Karlsruhe Reports in Informatics 2024,1
Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

Proseminar Mobile Computing SS 2023

Mobile und Verteilte Systeme
Ubiquitous Computing
Teil XX

Herausgeber:
Alexander Studt, Paul Tremper, Chaofan Li,
Haibin Zhao, Michael Beigl

2024

Karlsruhe Institute of Technology

Proseminar Mobile Computing
SS 2023

Mobile und Verteilte Systeme
Ubiquitous Computing

Teil XX

Herausgeber

Alexander Studt, Paul Tremper
Chaofan Li, Haibin Zhao
Michael Beigl

Karlsruhe Institute of Technology (KIT)
Fakultät für Informatik
Lehrstuhl für Pervasive Computing Systems (PCS) und TECO

Interner Bericht 2024,1
ISSN 2190-4782

Vorwort

Die Seminarreihe Mobile Computing und Ubiquitäre Systeme existiert seit dem Wintersemester 2013/2014. Seit diesem Semester findet das Proseminar Mobile Computing am Lehrstuhl für Pervasive Computing System statt.

Das Proseminar Mobile Computing wird seit dem Wintersemester 2013/2014 in jedem Semester durchgeführt. Seit dem Wintersemester 2003/2004 werden die Seminararbeiten als KIT-Berichte veröffentlicht. Ziel der Seminarreihe ist die Aufarbeitung und Diskussion aktueller Forschungsfragen.

Dieser Seminarband fasst die Arbeiten der Seminare des Sommersemesters 2023 zusammen. Die Themen der hier zusammengefassten Aufsätze umfasst die Themen "Neural Network Quantization", "Gaussian Process Regression", "Active Mobile Exoskeletons", "Path Planning for WSN Sensor Nodes" und "Investigating AI Techniques to Play Games". Wir danken den Studierenden für ihren besonderen Einsatz, sowohl während des Seminars als auch bei der Fertigstellung dieses Bandes.

Karlsruhe, den 27. Februar 2024

Alexander Studt
Paul Tremper
Chaofan Li
Haibin Zhao
Michael Beigl

Inhaltsverzeichnis

Investigating Artificial Intelligence techniques to play Poker

Bastian Franze

Karslruher Institut für Technologie, Karlsruhe, Germany
bastian.franze@student.kit.edu

Abstract Poker, like many games, is being used to measure the progress of artificial intelligence (AI). The particular problem with poker is incomplete information. Research in this area began with a game theory approach by Nash, who tried to find the optimal move. This was followed by the first hard-coded programs such as Loki, which were soon extended by neural networks to create programs such as Poki. Liberius beats professional players in two-player games by calculating blueprint strategies for the whole game, then solving the subgames to improve itself later to fix potential weaknesses. The current state-of-the-art approach, Pluribus, is capable of winning 6-player Texas Holdem unlimited against professional players. The Pluribus strategy is computed by self-play and improved during the game.

1 Introduction

Over the past few decades, many games such as chess, go or poker have been used to track the progress of AI and to benchmark its performance. Poker is widely used due to the nature of incomplete information and competitive multiplayer character in game theory[1] [2] and computer science [3] to provide explanations and examples on research on incomplete information scenarios. Over the past decade, AI capabilities have increased in complexity, playing poker with fewer constraints and winning at a high quality against professional players with significant winnings.
In game theory, a Nash equilibrium is found for a simplified version of three-player poker. A Nash equilibrium models a state in which no player can gain by unilaterally changing his strategy. This approach has limitations due to the increasing computational complexity of increasing the number of players or removing restrictions.
At the end of the 20th century, expert systems like *Loki* were created. These were strictly algorithmic AIs with some probabilistic modelling of opponents. These basic algorithms were soon improved with AIs like *Poki*. They mainly improved by modelling the opponents via a neural network to better adapt to their behaviour. Such AIs were not able to beat professional human players, but were able to get some results against average human players.
Libratus was one of the first poker AIs to defeat professional players in heads-up, two-handed no-limit texas hold'em. To achieve this, *Libratus* uses a pre-

calculated blueprint strategy, which it adapts based on the current game situation, and a self-improving algorithm to counter any weaknesses in the blueprint strategy.
The current state-of-the-art approach, *Pluribus*, is capable of beating professional players in six-player no-limit texas hold'em. This has been achieved by creating a blueprint strategy for self-play. During play, *Pluribus* tries to improve its current blueprint strategy by searching for a better strategy based on the current game situation. To achieve this, *Pluribus* uses a version of Monte Carlo Counterfactual Regret Minimisation (MCCRF) to compute its strategy.

2 Early research

Papp's master's thesis [4] gives a good example of a basic expert system. Expert systems are designed to react to predefined rules and are quite common. This requires a good knowledge and modeling of the current gamestate and the specific rules of the game. Implementations of such expert systems can vary from simple if statements based on one's own hand, to look-up tables that calculate the potential of one's own hand combined with randomised actions, to a theoretical Nash equilibrium. Due to the static nature of such an expert system, an experienced human player has a good chance of beating even complex implementations. Such expert systems are often improved by extending them with modern AI approaches. In game theory, ideal expert strategies are calculated using the Nash equilibrium. This means that no player can gain an advantage by switching strategy single-sided. Such approaches are limited to game theory due to the high computational cost.

2.1 Nash equilibrium

An equilibrium strategy is a list of actions for each player in which no player can improve by switching to another strategy.
An equilibrium strategy was found by Nash [1] for a simplified version of a three-player version of poker. He made the following simplifications in his example:

1. Every hand consists of one card
2. Each card is either low or high
3. Two chips are used to ante, call, or open
4. The player play in rotation till eighter everybody had the chance to call on an opened game or every player passed
5. The pot is divided equally among the highest hands equally
6. If no one bets the antes a retrieved

To calculate Nash equilibria, you need to model different game scenarios and calculate the payoffs of different strategies. There is no general formula. This makes using Nash equilibria to solve poker difficult. For this scenario, an equilibrium

strategy can be calculated using the given constraints. Nash mentions that because of the card constraint (Constraint #2) and the game ending condition (Constraint #4) it is pointless to pass with a high on the last move. Therefore the game can be reduced further. Based on the restrictions and the reduction of the pass, Nash created a table 2.1 showing the game with all possible moves.

	First Move	Second Move
Player I	Open on high Open on low	Call III on low Call II on low Call II and III on low
Player II	Cal I on low Open on high Open on low	Call III on low Call III and I on low
Player III	Open on low Call I on low Call II on low	Not a valid move

Due to the limited number of possible actions each player can take and the limited duration of the game, it is possible to further restrict the number of actions that lead to an equilibrium[5].

2.2 Loki

A more complete approach was explained in detail in the master's thesis of Papp [4] and in the book "The challenge of Poker" [6]. The expert system developed by papp, with its good documentation and continuous development, will serve as an example for early software expert systems. Loki gives a good example of poker programs without the implementation of modern AI technologies. Systems like *Loki* are the most common form of poker AIs. If you turn on your computer and play against a computer, chances are high that the AI you encounter will work in a similar way. *Loki* consists of three components: hand evaluation, betting strategy and opponent modelling, as shown in Figure 1.

Hand evaluation To evaluate the hand, *Loki's* performs two main steps, calculating the current hand strength, which describes the probability that the hand is the strongest at the current state of the game, and calculating the hand's potential, which describes the probability that the hand will become the strongest as the game progresses.

Strength is calculated based on pre-calculated win rates for each starting hand. Then the probability of each other starting hand interacting positively with the current cards on the board is assessed. For multiplayer, this method needs to be extended by the number of opponents and weighted based on the opponent

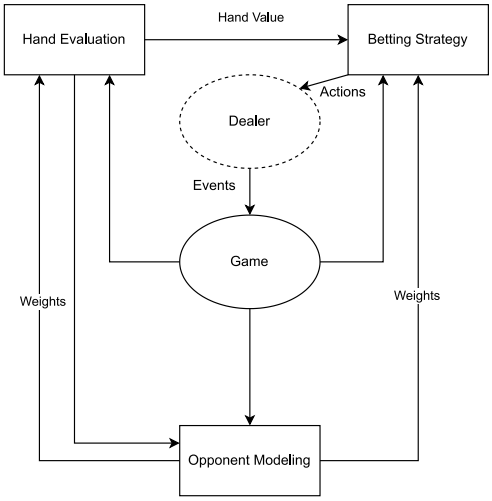


Figure 1. Overview of *Lokis* system provided by Papp[4]. This figure shows how the different modules of *Loki* work together, how they interact and how they influence each other.

models. To get the potential of the hand, the negative and positive potential is calculated with up to two cards lookahead in order to assess the next rounds and thus get a more precise view of the possibility of holding the strongest hand at the moment. When playing with multiple opponents, models and hand strength must be included, which increases the complexity of the calculation.

Betting strategy *Loki's* betting strategy is mainly based on it's calculated hand strength and hand potential. The strategy is also influenced by hardcoded factors such as the initial bet and manoeuvres such as semi-bluffing. *Loki's* betting and strategy does not take into account the behaviour of his opponents and therefore does not take advantage of their weaknesses, saving complexity in multiplayer scenarios.

Opponent modeling Opponents are modelled by *Loki* using weight arrays. The weight is adjusted based on the opponent's actions and their frequency. *Loki's* opponent modelling does not take into account how recently the actions were played and the covariance of each opponent. Papp[4] himself mentions that opponent modelling is one of the biggest opportunities for improvement and a potential weakness of *Loki*.

Improvements Improvements [7] have been made to the betting system by applying randomised actions, adapting the reweighting system to use probability triples, and using simulation to adjust the value of the bet. This more randomised

and statistical approach further improves the performance of *Loki* by making it less predictable and increasing the number of scenarios evaluated.

2.3 Limitations

The first approaches encountered inherent limitations. Nash equilibrium is proving to be a robust tool for formulating equilibrium strategies in incomplete information games. However, the complex interplay of computational complexity and the scarcity of straightforward strategy elimination methods makes the computation of Nash equilibria in complex scenarios a formidable task.
When dealing with expert systems that rely on probability tables and game state driven decisions, the challenge of modelling opponents while seamlessly integrating their behaviour and strategies into one's own optimisation framework becomes obvious. The lack of understanding of hidden information and the dynamics of static configurations creates vulnerabilities in systems such as Loki, allowing for potential exploitation. In response, contemporary AI methodologies have shifted towards a paradigm centred on the analysis of hidden data, a focus that aims to strengthen strategic decision-making in a way that transcends the limitations of its predecessors.

3 Modern AI appraoches

Poki [6] serves as a compelling case study in the transition to modern AI frameworks. Billings [6] extended the foundation of *Loki-2* [7] by integrating neural network-based opponent modelling into the architecture that created *Poki*. This development aimed to address the challenges of interpreting the hidden information in the game by modelling opponents and their behaviour.
Liberius [8] gives the example of a modern AI system beating professional players in two-player poker. To achieve this result, *Liberius* uses a blueprint strategy that has been pre-calculated using an MCCFR. To improve its strategy, *Liberius* also uses self-improvement algorithms and real-time search to improve on the current blueprint strategy.

3.1 Poki

Poki [6] is an improvement of *Loki-2* [7] by Billings. The main improvement of *Poki* compared to *Loki-2* is the implementation of an opponent modelling, which is based on a combination of statistics and a neural network that estimates the opponent's next move. This allows *Poki* to take advantage of your opponent's mistakes in multiplayer games. For betting, *Poki* uses the improvements already partially implemented in *Loki-2* to add simulation-based betting strategies and to adapt the weighting matrices and probability triples to take into account the opponent models.

Betting Strategie *Poki's* betting strategies like *Loki* still rely on the calculated hand strength and hand potential as already described in the section on *Loki*. These calculations are extended weighting matrices that contain a probability value of each possible hand being played at the current point in the game. The weighting matrices are updated in a re-weighting process after each betting action. To calculate the final betting strategy, the various hand strengths and weight matrices are combined to produce a probability triplet describing the fold, call and raise actions. Based on the thousands of probabilities, an action is selected and executed. *Poki* has been further improved with a simulation based betting strategy.

Oponent modeling Poki implements a neural network to create a more versatile opponent modelling that differs from conventional statistical methods. Poki's neural network is trained to predict the subsequent actions of opponents based on the current state of the game.
Bellings, a standard feed-forward neural network, is trained using a dataset collected from online matches against human players. This network architecture is designed to receive nineteen inputs, each corresponding to an aspect of the game context. This includes vital information such as the number of active players, the current state of the board, and other publicly available data points that inherently influence the players' decision-making process. The output layer consists of three distinct nodes, each of which corresponds to a possible action: fold, call and raise.

Evaluation To measure the performance of *Poki*, a unit called small sb/game has been used. This unit measures how many small blinds are won on average per game. *Poki* has been tested on over 20,000 hands of online poker. *Poki's* performance in online poker, mainly against amateur level players, was between +0.1 sb/game and +0.2 mbb/game. Against more intermediate players, *Poki's* advantage drops to +0.07 sb/game and +0.1sb/game. This performance can be considered extremely good even by modern AI standards, but a few considerations must be made. First of all, measuring in sb/game results in rather large numbers when rounded to the nearest hundred. Secondly, the skill level of *Poki's* opponents in online poker is not well known. Nevertheless, *Poki* beats *Loki* by 0.02 to 0.13 sb/game. Using the same validation technique, the AI results are fairly comparable, so a statement can be made about the doubling of performance by implementing an AI that models opponents.

3.2 Libratus

Brown and Sandholm developed *Libratus* [8], an AI that beats professional human players at two-player poker. According to the authors, the *Libratus* AI consists of three main modules:

1. Pre-calculated Blueprint strategy on an abstracted version of the game.

2. Subgame solver that traverses the game tree during the game. Trying to improve the blueprint strategy
3. Self-enhancement, filling in missing branches in the blueprint strategy

Unlike *Poki*, this AI does not rely on hard-coded strategies and opponent modelling, but attempts to solve an abstracted version of the game by playing itself.

Game abstraction According to the authors of *Libratus*, there are 10^{161} design points in an entire game. In order to keep this number calculable, a number of abstractions have to be made. The most common abstractions are bet size increments, which means that not every bet size is considered, only the most common ones. *Libratus* also groups hands with almost no difference and treats them as identical. This reduces the number of possibilities to a few million. MCCFR is used to calculate a blueprint strategy. When playing itself, the AI is more likely to choose the action with the highest regret in order to improve its overall performance. To find the best possible strategy and converge to an equilibrium, each self-played game is played from both perspectives. This means that the MCCFR is first traversed with a randomly chosen player, and then the algorithm changes the role of the player and traverses the game tree again.

Subgame solving Libratus plays according to its blueprint strategy only in the early part of the game, when the number of possible states is small and the abstraction is detailed. When the remaining game tree becomes too small, Libratus starts to create and solve an abstraction for the remaining subgame. Brown and Sandholm mention that solving subgames is a particularly hard problem in imperfect information games, because the optimal strategy may depend on unreached subgames. The subgame is solved by looking at the blueprint strategy and comparing the payoff with a simulated augmented subgame. Based on this, a new strategy is chosen. The subgame solving process is repeated for each subsequent off-tree action, this process is called nested subgame solving.

Self-enhancement *Libratus* self-improvement improves the blueprint strategy during the game. The module fills in missing branches in the blueprint strategy and calculates strategies for each of them. This module is necessary because the size of the game tree makes it impossible to compute the entire tree in advance. The way in which *Libratus* adds new branches tries to avoid exploitation by not exploiting the opponent. Instead, *Libratus* uses the bet sizes to add new branches to the blueprint. Self-improvement computes an answer by adding the given bet size to the game tree, but to avoid exploitation, Libratus only uses the off-tree strategy if the opponent uses the bet size.

Evaluation The performance of modern AIs and professional players is measured in mbb/game. Milli big blinds per game is the average number of big blinds won per 1000 games. *Libratus* has been tested against the best AI until

than, *Baby Tartanian8* [9]. Using the blueprint strategy alone, *Baby Tartanian8* outperformed *Libratus* by 8 MBB/game. Including the nested subgame solution, *Libratus* outplays *Baby Tartanian8* by 63 mbb/game.
Libratus played 120000 hands against top human professional players. During these games *Libratus* averaged 147 mbb/game, beating many top professional human players.

3.3 Limitations

Poki showed a significant improvement over its predecessors and achieved good results in online poker, but its performance drops significantly against better opponents. *Libratus* could beat professional humans and other high-performing AI systems, but is limited to two-player poker.

4 State of the Art AI System

Pluribus is an artificial intelligence created by Brown and Sandholm [3] that is capable of playing poker in real time at the highest competitive level, winning against professional human players. *Pluribus* strategies were calculated by playing against five copies of itself. Unlike *Libratus*, *Pluribus* mastered six-player poker using an MCCFR. To improve the blueprint strategy and achieve superhuman performance, Pluribus also performs a real-time search for a better strategy in the game tree.

4.1 Monte Carlo counterfactual regret minimization

Monte Carlo CFR [10][11][12] simulates a game by choosing a player to traverse the game tree. The traversal of the tree is initially random and gradually improves towards a good decision tree. At the end of the game, the algorithm determines how much better the traverser could have done by choosing different actions. The difference between the gain and the possible gain is added as regret. This allows the algorithm to later choose actions with higher regret with a higher probability of maximising the gain. This process is repeated for hypothetical decision points in the game. Exploring the tree is possible because the AI usually knows the strategy of all players. For two-player games, CFR is guaranteed to converge to a Nash equilibrium. For games with more than two players, it is guaranteed that CFR averages the performance of the best single fixed strategy in hindsight. [3] An example of the traverse can be seen in Figure 2. You can find an example algorithm in Algorithm 1, which is also used in *Pluribus*.

4.2 Pluribus

Game abstraction To reduce the number of desition points *Pluribus* has to face in the game, a game abstraction is applied, as in *Libratus*. *Pluribus* uses two different abstractions. The first one abstracts the actions in the game, by

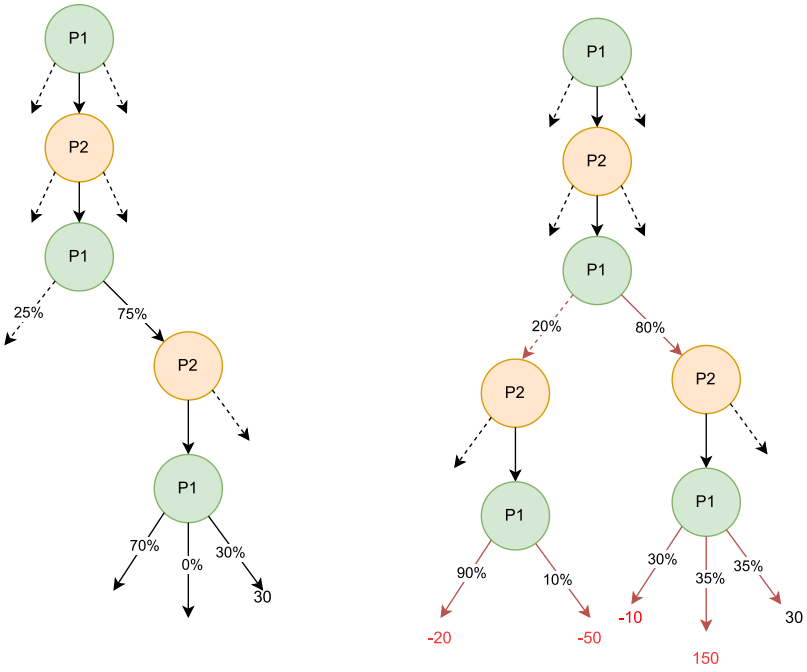


Figure 2. A game tree traversed by MCCFR. In the left tree, player 1 is traversing the game tree. The traversal simulates the game until an outcome is reached. Each decision point of player 1 is explored with every possible action that player 1 can take. Each of these actions is then simulated until an outcome is reached. Once the outcomes of the possible decisions are known, the probabilities in the game tree are updated by assigning higher probabilities to paths with higher gains (right tree).

Algorithm 1 MCCFR

function CALCULATE-STRATEGY($R(I_i), I_i$) ▷ Calculates the strategy based on regrets
function TRAVERSE-MCCRF(h, P_i)
  if h is a leaf node **then**
    return The game's payoff
  else if P_i is not a possible/valid option **then**
    return TRAVERSE-MCCFR($h \cdot 0, P_i$) ▷ This action remains irrelevant
  else if h is a chance node **then**
    Select an possible actions a based on the probabilities of this node
    return TRAVERSE-MCCFR($h \cdot a, P_i$)
  else if $P(h) = P_i$ **then**
    Get the Infoset I_i for P_i at this node
    CALCULATE-STRATEGY($R(I_i), I_i$) ▷ Determine the strategy at the Infoset I_i (returns Probability distribution for actions)
    Initialize expected value at zero
    for action a in $A(h)$ **do**
      TRAVERSE-MCCFR($h \cdot a, P_i$) ▷ Traverse each action
      Update the expected value based on traversal and probabilities
    for action a in $A(h)$ **do**
      Update the regrets $R(I_i, a)$ based on the expected values
    return the expected value $v(h)$
  else
    Get the $P_{P(h)}$ infoset of this node
    CALCULATE-STRATEGY($R(I_{P(h)}), I_{P(h)}$)
    Sample an action a from the probability distribution of the strategy
    return TRAVERSE-MCCFR($h \cdot a, P_i$)

reducing the number of bet sizes considered in the game. This is done because increasing a bet in the dollar range doesn't make much difference to the state of the game and the decision-making process. Second, *Pluribus* also reduces the amount of information it needs to consider by treating strategically similar hands identically, which reduces the number of different situations for which a strategy must be determined.

Blueprint Strategie Two-player self-play algorithms do not converge to an equilibrium strategy, but practice has shown that they produce reasonable results. *Pluribus* produces a blueprint strategy for the entire game offline via self-play. *Pluribus* uses an MCCFR to compute the self-play blueprint strategy, which samples actions in the game tree instead of traversing the whole game tree in each iteration. *Pluribus* plays according to this blueprint strategy only in the first round, where the number of decision points is small enough that no information abstraction is needed.

Realtime search Due to the limitations of the Blueprint strategy, *Pluribus* only plays it during the first of four betting rounds. After the first round, *Pluribus* performs a real-time search to determine a better strategy for the current state of the game. The real-time search looks ahead until it reaches a leaf node at the depth limit of the algorithm's lookahead. For perfect information games, each leaf node is then evaluated over an evaluation function, assuming that both players continue to play an equilibrium strategy from the evaluation point. For imperfect information games, this assumption is broken because leaf nodes do not have fixed values. Instead, the value of a leaf node depends on the strategy that the searcher chooses for the subgame. *Pluribus* solves this problem by using an approach that considers that any or all players can change their strategy beyond the leaf node of a subgame. The premise of *Pluribus* in this scenario is that the player has a choice of four different strategies.

1. The precomputed blueprint strategy
2. A blueprint strategy biased towards folding
3. A blueprint strategy biased towards calling
4. A blueprint strategy biased towards raising

This allows *Pluribus* to find a more balanced strategy, as otherwise it would be punished by opponents changing their strategy.

Keeping sensitive information One of the major challenges for AI in imperfect information games is protecting its private information from opponents. For example, if *Pluribus* has an exceptionally strong hand, it must refrain from placing an overly large bet, as this could cause other players to fold in anticipation. To overcome this dilemma, *Pluribus* adopts a strategy that involves tracking the probabilities of each potential hand reaching the current state of the game,

while maintaining its current strategy. In determining the appropriate move for its current hand, *Pluribus* adopts a method of evaluating its response across all potential hands, thus achieving a balanced strategy within the existing game scenario. This calculated approach ensures that *Pluribus* remains unpredictable for its opponents.

4.3 Evaluation

Pluribus was evaluated through matches against human professionals. The evaluation process included two scenarios: first, one instance of *Pluribus* competing against five human professionals; second, five instances of Pluribus competing individually against a single human professional. To reduce the influence of luck inherent in poker games, the experiments used a technique known as AIVAT to minimise variance.

One Human, Five copies of Pluribus Over the course of the research conducted by Brown and Sandholm, a total of 10,000 poker games were played by two different human professionals. Each of these players played 5,000 hands of poker against five iterations of *Pluribus*. Remarkably, *Pluribus*, operating without any knowledge of its opponents and thus without the ability to adjust its strategy in response, refrained from cooperating with its other copies.
The result was remarkable: *Pluribus* demonstrated its prowess by winning by an average of 32 mbb/game.

Five Human, One copy of Pluribus A series of 10,000 poker hands were meticulously played, placing *Pluribus* in direct confrontation with a group of five human professionals. To mitigate the potential impact of pre-existing knowledge, a concerted effort was made to keep all participants in the dark as to the identity of their rivals. This was achieved by assigning each participant an unchanging pseudonym, which was maintained throughout the experiment. This strategic measure was designed to prevent any prior bias from influencing the results. In particular, this approach allowed each player to observe and analyse the strategies of their peers. From the pool of human experts, a group of 13 eager volunteers, five participants were selected for each day. *Pluribus* achieved a remarkable average of 48 mbb/game.

The results of both segments of the evaluation have attracted considerable attention due to the high win rates achieved in the area of professional six-player No Limit Texas Hold'em poker.
Notably, *Pluribus* has demonstrated a remarkable ability to outplay even seasoned poker veterans. In particular, it defeated renowned player Chris "Jesus" Ferguson, an individual who boasts an impressive six World Series of Poker wins. This achievement underscores the AI's prowess by demonstrating its ability to outplay even the most illustrious human competitors in a game known for its complexity and nuance due to imperfect information.

4.4 Results

Pluribus demonstrates an impressive ability to beat skilled human professionals at six-player No-Limit Texas Hold'em. Its proficiency is the result of being trained through self-play, creating a basic blueprint strategy. This blueprint is then refined in real time through a continuous search for improved strategies, facilitated by MCCFR. The remarkable results achieved by *Pluribus* underline its ability to significantly outperform even seasoned professionals.

5 Conclusion

Game theory establishes the existence of equilibrium strategies in poker. However, the practical application of this insight is hampered by the lack of an efficient algorithm capable of uncovering such strategies, coupled with the colossal computational complexity required to compute them. As a result, this theoretical construct remains in the game theorie and is limited to simplified variants of poker.
Loki is a well-researched and comprehensible example of an expert system in action. *Loki* tries to solve poker by modelling its opponent based on their privileged actions and probabilities. Its decisions about which actions to take are based on predictive facts about the game and the probability of holding the best hand relative to its opponents.
Poki departs from the traditional approach of modelling opponents based solely on their past actions and associated probabilities. Instead, *Poki* uses a neural network to predict the opponent's upcoming moves, improving the probabilistic strategy and leading to more refined and effective strategic decisions. Both AIs, *Loki* and *Poki*, are capable of achieving good results in online poker.
Liberatus achieved a remarkable milestone by outperforming professional players and even leading AI opponents in two-player poker during its development period. This feat was made possible by using MCCFR as the backbone of its strategy calculation. This approach was augmented by a process of self-improvement and real-time search capabilities that further enhanced its strategy.
Pluribus shares similarities with *Libratus* in its training methodology. Like *Libratus*, *Pluribus* uses self-play as a fundamental training process, with the implementation of the MCCFR algorithm to create its blueprint strategy. *Pluribus* uses real-time game tree search to search for better strategies. This enables *Pluribus* to achieve a level of performance beyond human capabilities, leading to its triumph over professional players in the challenging domain of six-player no-limit Texas Hold'em.

6 Appendix

6.1 Poker Rules

[13] At the start of each round, a player is designated to post a mandatory bet called the "small blind" before any cards are dealt. Next to the small blind,

another player is designated to post a "big blind" (usually double the small blind), which is another mandatory bet. Following these initial bets, the cards are dealt. Each player is dealt two cards face down.
This is followed by a decision period in which each player evaluates their cards and considers their course of action. The options available are to match the value of the Big Blind (a 'Call'), to increase the bet (a 'Raise') or to fold (a 'Fold'). Players who choose to proceed then witness the "flop", where three community cards are revealed face up on the table.
This is followed by a betting round, giving players another opportunity to bet, raise or fold. At the end of this round, another card is dealt face up to add to the community cards on the table. This process is repeated, with additional betting rounds and subsequent card reveals, until five cards are revealed.
At the end of the final betting round, the players still actively participating use their two face up cards in combination with the community cards on the table. This combination is used to determine the best hand and the winner of the current hand.

References

[1] John Nash. "Non-Cooperative Games". In: *Annals of Mathematics* 54.2 (1951), pp. 286–295. ISSN: 0003486X. URL: http://www.jstor.org/stable/1969529 (visited on 07/15/2023).

[2] Matej Moravčík et al. "DeepStack: Expert-level artificial intelligence in heads-up no-limit poker". In: *Science* 356.6337 (2017), pp. 508–513. DOI: 10.1126/science.aam6960. eprint: https://www.science.org/doi/pdf/10.1126/science.aam6960. URL: https://www.science.org/doi/abs/10.1126/science.aam6960.

[3] Noam Brown and Tuomas Sandholm. "Superhuman AI for multiplayer poker". In: *Science* 365.6456 (2019), pp. 885–890. DOI: 10.1126/science.aay2400. eprint: https://www.science.org/doi/pdf/10.1126/science.aay2400. URL: https://www.science.org/doi/abs/10.1126/science.aay2400.

[4] Denis Richard Papp. "Dealing with Imperfect Information in Poker". In: (1998). URL: https://webdocs.cs.ualberta.ca/~games/poker/publications/papp.msc.pdf.

[5] Branislav L Slantchev. "Game theory: Dominance, Nash equilibrium, symmetry". In: *Department of Political Science, University of California–San Diego* (2008).

[6] Darse Billings et al. "The challenge of poker". In: *Artificial Intelligence* 134.1 (2002), pp. 201–240. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(01)00130-8. URL: https://www.sciencedirect.com/science/article/pii/S0004370201001308.

[7] Darse Billings et al. "Using Probabilistic Knowledge and Simulation to Play Poker." In: Jan. 1999, pp. 697–703.

[8] Noam Brown and Tuomas Sandholm. "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals". In: *Science* 359.6374 (2018), pp. 418–424. DOI: 10.1126/science.aao1733. eprint: https://www.science.org/doi/pdf/10.1126/science.aao1733. URL: https://www.science.org/doi/abs/10.1126/science.aao1733.

[9] Noam Brown and Tuomas Sandholm. "Baby Tartanian8: Winning Agent from the 2016 Annual Computer Poker Competition". In: *International Joint Conference on Artificial Intelligence*. 2016. URL: https://api.semanticscholar.org/CorpusID:9343634.

[10] Neil Burch et al. "Efficient Monte Carlo Counterfactual Regret Minimization in Games with Many Player Actions". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/3df1d4b96d8976ff5986393e8767f5b2-Paper.pdf.

[11] Marcin Ziemiński. "Deep-Hedge MCCFR: An algorithm for solving imperfect information games." eng. In: ().

[12] Martin Schmid et al. "Variance reduction in monte carlo counterfactual regret minimization (VR-MCCFR) for extensive form games using baselines". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 2157–2164.

[13] *Wikipedia Texas hold 'em*. https://web.archive.org/web/20230816122501/https://en.wikipedia.org/wiki/Texas_hold_\%27em. Accessed: 2023-08-16.

Investigating Artificial Intelligence Techniques to Play (Contract) Bridge

Simeon Schrape

Karlsruhe Institute of Technologie

Abstract. Artificial intelligence (AI) has achieved significant advancements in games that involve complete information. However, Bridge, a multiplayer game with imperfect information, remains highly challenging. Bridge comprises two main components: bidding and playing. When it comes to playing ability, expert-level teams are often on par with each other, making bidding the crucial factor in determining victory or defeat. This paper delves into the progress of AI in bridge bidding throughout the existence of computers. While notable achievements have been made in bridge bidding AI, the ultimate goal of surpassing human experts has not yet been attained. However, a double synergy neural network shows promise in approaching the expertise level by being capable of bidding in diverse bidding systems.

Keywords: bridge-bidding · neural-networks · literature-review

1 Introduction

Games have long been a challenging domain for testing artificial intelligence. Some AI research focuses on complete-information games like chess and go, while others study incomplete-information games such as poker and bridge. Bridge, a cooperative and competitive card game, poses challenges due to its partial-information nature. The game of contract bridge is played in two phases. The first one is the bidding phase and the second one is the playing phase. This literature review focuses on the advancement of the bidding in bridge, primarily due to bidding constituting approximately 75% of the game, with playing comprising only about 25%[8]. Moreover, at the expert level, players often exhibit comparable skill levels during the playing phase. As a result, the bidding phase plays a crucial role in determining the victor of the game. The bidding phase in bridge poses a challenging problem for a computer, given that the objective is not as easily discernible as in the playing phase. In the playing phase, the main objective is to obtain as many tricks as possible. In contrast, during the bidding phase, there is a necessity for direct player communication to exchange information about their respective hands and to ultimately arrive at the optimal contract for each of their hands.

Rule-based approaches, such as those by Wasserman [5] and Ginsberg [1], sought to mimic the meaning of bids in human bridge bidding systems. A bidding system consists of rules and constraints that are used by team partners to interpret bids and communicate abstract features to help in deciding on the next bid. Those features include the High Card Points, a numerical score of the highest cards in a hand, as well as the balance and suit length of a particular suit in a hand. Resolving bid ambiguity poses a significant challenge when simulating human bidding systems, as the rules from such systems can overlap, resulting in conflicting suggestions [7]. An intelligent resolution of ambiguity is crucial for human players. They achieve this by devoting a considerable amount of time to practising their bidding with their partners, aiming to attain a flawless mutual understanding.

Other approaches like the one by Yeh [7] tried to teach AI to bid without referencing human bidding systems. Inspired by the success of deep learning, they input the raw data into the neural networks without any abstraction by the bidding systems. The goal of these frameworks is to resolve bid ambiguity by learning sophisticated bidding rules and achieve improved bidding performance by leveraging all available information.
The state-of-the-art technique proposed by Zhang [8] uses the synergy of two deep neural networks and feature extraction algorithms to determine the most appropriate bid. The method has shown promising results against human players.

2 Algorithmic Methods

Research on Bridge Algorithms has been underway since the advent of computers. The first bridge bidding program was already developed in 1953. There are various obstacles to overcome when designing an algorithm for bridge. The first challenge is the decision-making process within a vast search space, particularly in relation to the numerous possible bidding sequences. Additionally, the algorithm must account for resource constraints and not indefinitely pursue results. The other challenge arises from bridge being an incomplete information game, which means that there is only partial information available. Another challenge in bridge is its competitive and cooperative nature. During the bidding phase, it is necessary to distinguish between friendly and hostile information, even with communication generally restricted between partners. Another issue is the need for a certain level of generality since different pairs may utilise different bidding systems. The final major challenge is the lack of indication of good bids during the bidding phase, with the indicator of a good bid occurring only after the playing phase, when it is clear that the round has been won.

2.1 Rule Based Bidding System

In 1970, Anthony Wasserman [5] created an algorithm for playing bridge. This program was the first to demonstrate a considerably strong bridge bidding performance. Initially, the subproblem of Partnership Bidding was addressed followed by the extension of the algorithm to tackle the competitive bidding problem. Partnership Bidding or non-competitive bidding is a unique instance of bridge bidding where the process occurs without interference or competition in a peaceful environment. That means the opposing team passes the bid every time during the bidding phase. The general tactic was to bid in the same way as a human player would bid, which required the program to simulate human judgement by considering the same conditions and making the same evaluations.

Decision Tree Model The method involved constructing a decision tree that includes all plausible bidding sequences as paths, resulting in an approximate

total of 10^{10} partnership bidding possibilities and 10^{47} competitive bidding possibilities. Due to limited resources, an algorithm is incapable of navigating through such an extensive decision tree. Therefore, a set of rules had to be created to reduce the number of possibilities. The rules are based on the bidding systems used in the bridge community. Wassermann considered three systems, namely the Standard American Bidding System, the Schenken System, and the Kaplan Scheinwold System. Thus, Wasserman narrowed down the root node's child to 8 opening bid classes instead of the 38 potential bids. The opening bid classes represent the bidding sequences that start with the opening bids that are defined in the different bidding systems. These sequences are then further categorised into subclasses based on the subsequent bidding actions. Because the length of a bidding sequence rarely exceeds 11, any paths longer than 11 are removed from the decision tree. Finally, Wasserman merged certain sequences, as some bidding sequences are practically equivalent and can, therefore, be treated as identical subclasses. This resulted in a transition from a tree structure to a cycle-free, directed graph.

Hand Evaluation To determine the next bid at each node of the decision tree, the algorithm reevaluates the strength of the partnership by computing a score. The partnership receives extra points for having longer cards in the suits their partner has bid, and for having fewer cards in the suits they haven't bid. Points are taken away from their total if they lack cards in the suits their partner has bid. Additionally, if the partnership agrees on a trump suit, there are further adjustments made to the points.

Testing After the pruning of the decision tree, the program was able to achieve good results against human players. In the context of non-competitive bidding, the program demonstrated a level of skill equivalent to that of a human expert. It was expected that the program would perform less effectively in the context of competitive bidding, as objective evaluations suggest that non-competitive bidding is a less challenging environment. When one team comprised the Wassermann program and the other comprised two expert bridge players, the program was estimated to perform slightly more skillfully than the average bridge player.

Conclusion In conclusion, the Wasserman approach was the initial program that achieved favourable results against human players. It addressed several of the challenges involved in creating such a program. Like the large search space, the processing of friendly and hostile information and the generality of multiple bidding systems.

2.2 Ginsberg's Intelligent Bridgeplayer

The next major breakthrough was GIB [1] by Matthew Ginsberg. At the time of its development, it was called the first bridge-playing program to approach the

level of a human expert. GIB secured twelfth place among a carefully selected group of thirty-four skilled professionals during an exclusive invitational event held at the 1998 World Bridge Championships. However, the type of bridge that was played at the event was without the bidding phase. GIB's bidding skills are the weakest part of the program, as they rely on an extensive set of rules stored in a database to define the interpretation of various auctions.

Borel Simulation The bidding algorithm employed by GIB is referred to as the Borel simulation, which utilizes a Monte Carlo-style simulation approach. The initial stage involves creating a set of deals, denoted as D, that aligns with the ongoing bidding. For every bid $b \in B$ and each deal $d \in D$, the database Z is utilized to anticipate the future course of the auction. Each contract is assessed based on its double dummy result. The bid b with the highest double dummy result is then selected and returned as the optimal choice.

Double Dummy Solver The double dummy result refers to the number of tricks achieved by the contractor during gameplay, assuming that all information is fully visible to all players. However, relying solely on the double dummy result leads to suboptimal performance, as bridge is not actually played with complete knowledge of all cards.

Limitations The algorithm neglects the strategic aspect where experts intentionally choose to conceal information by refraining from making specific bids, which is an essential factor to consider. The conservative nature of the database presents a challenge as the player assumes their partner bids conservatively. This leads the player to compensate by bidding aggressively, resulting in an overcompensating partnership. However, a significantly more substantial issue arises when the database contains omissions of data. For example, if the database suggests a bad choice every time a specific bid has been made. GIB tries to abuse this by bidding the specific bid because it thinks that this will result in a mistake made by the opponents. But in reality this will of course not occur. This is generally a significant issue in heuristically modelling an adversary's behaviour because it is challenging to differentiate a favourable decision that succeeds because the opponent has no winning options from a poor decision that seems successful due to the opponent's error. There are ways to address that problem but none of them are perfect.

Evaluation After this optimization, the bidding of GIB is a lot better than earlier programs but still far away from an expert level. In a bridge bidding competition, the final score achieved was +2 IMPS, which significantly outperformed the second-place contestant who scored -35 IMPS. Notably, in the previous year's contest, the victorious program attained an even lower score of -37 IMPS. Overall, GIB represents a significant breakthrough in the field of bridge bidding.

2.3 Noteworthy Approch

In 1999, the Wojciech Jamroga [3] proposed a new unique approach to modeling information exchange in bridge bidding. He viewed bidding as a conversation between four players exchanging information about their cards and decisions. Natural language processing techniques were used to interpret and model the bidding. The conversation involved at least one producer that makes the bid and one comprehender, with six utterance functions representing the semantic structure [2]. The model emphasized the information-processing nature of the game. It also introduced a prototype bidding system and a methodology for evaluating performance. The approach addressed the challenges posed by games with incomplete information and provided insights into cooperative communication and reasoning in bridge bidding.

2.4 Conclusion

The two approaches already achieved promising results in the realm of bridge bidding. They also surpassed multiple challenges in creating a bridge bidding Algorithm. However, they both are far away from surpassing humans in competitive bridge tournaments.

3 Early AI Methods

As computing power improves, machine learning methods are becoming more efficient and producing more accurate results. In the case of bridge bidding research, modern machine learning techniques have also been employed to solve the issue. One method involved employing a neural network to predict the opening bid, while the other approach utilised a deep neural network to solve the subproblem of non-competitive bidding.

3.1 Opening Bid Problem

In 1996, Yegnanarayana et al conducted a study[6], which aimed to explore the possibility of automating bridge bidding using modern AI techniques. The objective of the research was to investigate the reasoning process involved in a player's opening bid by analyzing the card patterns presented. The focus was specifically on evaluating the simplicity of the opening bid scenario, as it solely relies on the information provided by the hand cards held by the player.

Data Representation Yegnanarayana et al. used deep learning techniques with an artificial neural network (ANN). An integral aspect of training an ANN is the data representation phase, where there are several approaches to representing the 13 cards in a player's hand. These representations can be either feature patterns or a binary coding of 52 nodes that can be activated or deactivated. A feature pattern is, for example, the representation of the hand cards solely with

their high card point values. This encapsulation of relevant information offers an enhanced level of abstraction. However, in order to maintain the performance of the network and ensure effective generalization, Yegnanarayana et al. chose to use the raw shape representation using the 52-node scheme. This decision was motivated by the recognition that an imprecise abstraction could hinder the network's ability to generalize meaningfully.

Training Data Obtaining suitable training data is an essential part of training the neural network. In this scenario, Yegnanarayana et al. created training data using a program that creates a set of randomly distributed cards. To label the cards, Yegnanarayana et al. then individually made opening bids based on the 13 cards generated by the simulation. The backpropagation algorithm was used to train the multilayer feedforward networks. This process optimised the network's weights and biases by progressively adjusting them to minimise the difference between predicted and desired outcomes.

Testing To evaluate their approach, Yegnanarayana and his colleagues carried out three distinct experiments, each involving a different interpretation of the results.
The first experiment utilised a total of 13 nodes to represent the output. Among these nodes, one indicated the 'pass' bid, five represented the various suits, and seven identified the bid level. It should be noted that each set of cards activated precisely two output nodes, with the exception of the 'pass' bid scenario. When two nodes were activated, one was assigned to the bid level and the other to the bid suit. For this study, the authors utilised a training set consisting of 900 hands. Nevertheless, the performance of the network showed suboptimal convergence, which could be attributed to the inadequate size of the training set. Moreover, the occurrence of opening bids with levels exceeding three was infrequent, resulting in a low frequency of such patterns in the training data. Thus, the network faced difficulties in learning and generalising effectively from these rare patterns.
To address the constraints identified in the initial experiment, Yegnanarayana et al. developed a second experiment using a network architecture that comprised seven output nodes. This design was customised to predict level 1 bids. The seven output nodes included five nodes committed to the various suits, one node for the pass bid, and another node designed for situations where the bid level exceeded 1. Activation of this additional node signalled bids beyond level 1. The network architecture achieved optimal performance by integrating a hidden layer containing 60 nodes. Notably, this configuration produced favourable outcomes, with the network demonstrating commendable predictive accuracy when making bid predictions.
Recognising the inadequacy of restricting predictions solely to level 1 bids, Yegnanarayana et al. conducted a third experiment that involved an extended output space consisting of 12 nodes. This setup included one node solely for the pass bid, 5 nodes for first-level bids and another 5 nodes for second-level bids, as

well as an extra node to accommodate unfamiliar bids. Once again, the most favourable performance was observed when utilising a hidden layer consisting of 60 nodes. However, to guarantee optimal training outcomes, modifications have been made to the training data. Specifically, the frequency of level 2 bids has been increased within the training data. This modification was made to account for the infrequency of level 2 opening bids in hands with a normal distribution. Without increasing the representation of level 2 bids, a training set consisting of only 1600 hands would lack sufficient instances of level 2 bids, making it difficult to achieve satisfactory training results.

Conclusion In conclusion, this paper confirms the effectiveness of neural networks in acquiring the fundamental reasoning used for making opening bids in Bridge. The study revealed several important findings. Initially, training larger networks proved to be difficult. Nevertheless, networks with lower complexity demonstrated favourable performance outcomes. Additionally, the study showed the restricted learning capacity of a network that concentrates on 2-level bids. This limitation is mainly due to the insufficiency of data available for these bids. However, Yegnanarayana et al. exhibited that improved performance could be attained by adding specific data associated with these bids. Nevertheless, constructing a comprehensive bidding system requires a more advanced network structure capable of integrating raw and processed information. Several research considerations arose, including determining the most effective representation of raw input data in card games such as Bridge and addressing the complexities of training networks to detect patterns that occur with different frequencies.

3.2 Non-competitive Bidding with Deep Neural Networks

In 2016, Chaih-Kuan Yeh and Hsuan-Tien Lin[7] conducted a study at the National Taiwan University, which focused on the sub-problem of non-competitive bidding. Previous studies had reported that using raw data as input resulted in poor performance, but considering the success of deep learning in automatically building useful features from raw and abstract data, they decided to provide neural networks with only raw data as input. The framework developed uses deep reinforcement learning to automate bridge bidding. The framework enhances two aspects of bridge bidding AI.

Data Representation The first one is the learning of data representation. With the use of deep neural networks for feature extraction, there is no need for human-based bidding systems or human-designed features anymore. This allows the usage of the full power of machines because they can use all possible information without any filters.

Bid Ambiguity The second aspect is to resolve the bid ambiguity. Bid ambiguity poses a significant challenge in replicating human-based bidding systems.

Human systems have overlapping rules, resulting in conflicting suggestions based on player cards and bids. Resolving this ambiguity requires intelligent decision-making from human players, who also practice extensively to reduce ambiguity through mutual understanding. AI players face challenges in reaching the same level of mutual understanding as humans, making them inferior in the bidding phase. Using the framework, sophisticated bidding rules can be learned automatically to address bid ambiguity. This framework emulates the mutual understanding established by human players through practice, but in this case, it achieves it autonomously by practising with itself. At the time, it was the first known framework capable of achieving reliable mutual understanding solely through machine-based bridge bidding. This prevented the framework from being generally inferior to humans.

Problem Setup To represent the bidding problem, a two-player setting with Player 1 and Player 2 sitting at the North-South positions is considered. It is assumed, that Player 1 bids in odd rounds, and Player 2 bids in even rounds. The state $s^{(t)}$ is defined as a combination of the player's cards and the bidding history. The bidding strategy is represented by a function $G(s^{(t)}) = a^{(t)}$ that maps the state to a bid.

Training Data To learn a good bidding strategy, data is generated by simulating bridge games. Each instance in the data consists of the players' card vectors and the corresponding score for each possible contract. After that double dummy analysis is used to estimate the score for each contract without carrying out the playing phase. The expected score is approximated by randomly dealing the remaining cards to the opponents and performing a double dummy analysis multiple times.

Training For training the bidding strategy the reinforcement Q-learning algorithm is used. Q-learning is a form of reinforcement learning that does not require modelling of the environment. The Q-function represents the value of taking a particular action in a given state. The Q-function is updated iteratively based on feedback from previous actions. The Q-function is approximated using a deep neural network, and the loss is minimized through gradient descent.
The bridge bidding problem is modelled as a two-player partial-information cooperative game with multiple stages. To address the challenges of bid evaluation a penetrative Bellman's equation is introduced, which helps in estimating the cost of each bid accurately. Communication between partners is crucial in bridge bidding, and therefore a contextual bandit algorithm to balance exploration and exploitation is used. This approach enables bids that convey information effectively.

Conclusion In conclusion, this method introduces a model that combines deep reinforcement learning with enhanced exploration and update techniques to autonomously learn bidding strategies from raw hand data. At that time the model

was the first of its kind to address automatic bridge bidding solely based on raw data, without relying on additional human expertise. The model also surpassed both champion-winning programs and existing state-of-the-art models by a significant margin. This remarkable performance validates the potential of deep learning in achieving a highly competitive bidding system independently.

3.3 Conclusion

The initial attempts to apply modern AI techniques were promising but were only applied to sub-problems of bridge, such as the opening bid problem or non-competitive bidding. Further research is required to tackle the challenge of competitive bidding.

4 State of the Art Method

The current state-of-the-art method is the synergy of Double Neural Networks for Bridge bidding by Xiauyu Zhang[8]. The research is focused on bridge bidding with competition. It is the first approach that uses two different Neural networks and it takes multiple bidding systems into account. Zhang et al. have identified two major challenges that they need to address. The first is understanding the bidding systems used by the opponents, even if it is a different system to the one the AI uses, and the second is the large game state space. The challenges were addressed by implementing feature extraction algorithms for the bidding systems and neural networks to get to a result quickly.

4.1 Multi-bidding System

In human bridge games, it is common for pairs to have different bidding systems. However, the opposing teams still understand each other's bidding system. If one team consists of AI and the opponents use a different bidding system, the AI's ability to understand the bids and to make good bids for itself is greatly reduced because it is unable to understand the bids made by the opponents. Previous research has never considered multiple bidding systems. To solve this problem, Zhang et al.[8] have introduced feature extraction algorithms.
The information one uses the most when bidding, consists of three parts. The hand information, the situation information and the historical bidding sequences. A player's 13 cards are the hand information. The situation information consists of the vulnerability, the player's turn and whether the player is the opener. Of the three parts, the bidding sequence is the most difficult to handle. Previous work has inserted the bidding sequence directly into the neural network, as in Yeh and Lin's approach [7]. However, in order to function effectively, the network requires training on a specific bidding system since the same bidding sequence may have different interpretations in varying bidding systems. To solve this problem, a feature extraction algorithm is implemented. The information of a bidding sequence can be abstracted into 30 general features that apply to

any bidding system. With the help of bridge bidding experts and the analysis of 2.5 million bidding instances, a feature extraction algorithm F is designed for a specific bidding system to convert a sequence L into these general features through a fixed logic. As the rules for bidding systems can be very complex, 30 features cannot fully capture the meaning of the bidding sequences. However, it is still more precise than providing the Neural Network with the unprocessed bidding sequences for training. The general features I_L are represented as $I_L = F(L)$ given a bidding sequence L and an extraction Algorithm F.

4.2 Computational Complexity

A huge game state space is a difficult task. To explore the large number of possibilities, hardware performance and algorithm design are put to the test. To speed up the computation, deep neural networks are used to find the optimal bid in the current bidding state. The bidding decision model consists of two core artificial neural networks. The first one is for the bid selection and the other one is for bid evaluation.

Bid Selection Model The task of the bid selection model is to select the candidate bids for the current state of the game. It must respect the bidding constraints in such a way that all the bids it proposes are legal in the current bidding state. The input to the neural network is a 1×1284 vector. One part is for the general features produced by the feature extraction algorithm. The general features contain information about the other players that is extracted from the bidding sequence. For instance, one of the 30 features is the high card points, while another is the suit. Another feature indicates whether the bid resulted in a slam. It contains information about the historical bidding sequence. The next part is for the player cards h_i. The cards are encoded in a 4×13 matrix, which is then flattened into a 1×52 vector. The last part is for the vulnerability v, the dealer d and the player i. Each of these is encoded into a 1×4 vector. The output of the bidding model is a 1×38 vector. It contains the probability distribution P over the 38 possible bids. The formal definition of the bid selection:

$$P = bid_{slector}(v, d, i, h_i, I_L) = \{b \to p | b \in B \text{ and } p = 1\} \tag{1}$$

The structure of the network is shown in figure 1.

Evaluation Model The role of the evaluation model is to determine which of the candidate bids is the best. Different from other games like chess and Go there is no real-time reward that determines if a bid was a good choice. One only gets a reward after the bidding and the playing phase is over. The evaluation Network now tries to guess the IMP based on the information it gets as input.

$$IMP = situation_{evaluator}(v, d, i, h_i, I_L) \tag{2}$$

The structure of the evaluation model is almost the same as the bid selection model. The only two differences are in the input and output of the model. The

input of the evaluation model now contains the bid from the bid selection model. Since the bid sequence L is now $L + b$, the extracted features I_L contain the new information. The difference in the output is the output vector. The output vector from the evaluation model is a 1×49 one-hot vector with a length of 49, representing the IMP values from -24 to +24.

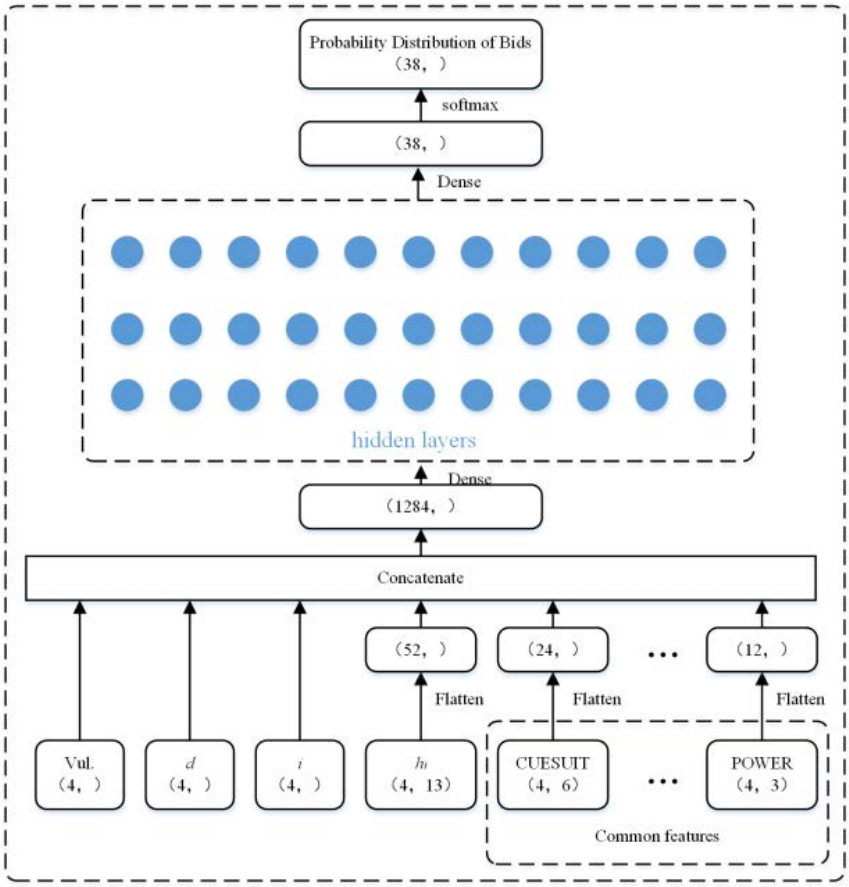


Fig. 1. The illustration visualizes the structure of the bid selection network. It shows the different input vectors with the different input parameters and the output vector, which is a probability distribution of all the possible bids. The structure of the evaluation model is almost the same. The only difference is the output vector.[8]

4.3 General Process

The general process, as illustrated in the pseudocode algorithm 1, for the entire system is as follows. First, the bidding sequence L is inserted into the feature extraction algorithm F and the general features I_L are extracted. Together with the hand cards H_i of the player, they are then encoded. The result of this is then fed into the bid selection network $bid_{selector}$. The output is the probability distribution P of all possible bids B that can be made. All bids that have a probability are then evaluated by the evaluation network $situation_{evaluator}$. The resulting International Match Point values of the bids are then compared and the bid with the highest IMP value is selected as the next bid.

Pseudo-code The following pseudo-code shows the process of the system.

Algorithm 1: General Process of the Bid Selection Model

Data: player cards h_i + bidding sequence L
Result: next bid
Function `Main`(h_i, L, v, d, i)**:**
  $I_L \leftarrow F(L)$;
  $P \leftarrow bid_{selector}(v, d, i, h_i, I_L)$;
  $IMP \leftarrow \emptyset$;
  for *bid b in Bids* **do**
    if $p > 0$ **then**
      $I_L \leftarrow F(L + b)$;
      $IMP \leftarrow situation_{evaluator}(v, d, i, h_i, I_L)$;
    end
  end
  next bid $\leftarrow \max(IMP)$;
  return next bid;

4.4 Training

In order to train the evaluation model, the IMP score needs to be calculated since it is not present in the training data. IMP calculation requires two scores: the actual score after the game and the score of the best contract. The best contract refers to the balanced point where neither the winner nor the loser can achieve a higher or lower score, ensuring fairness. The score under the contract is obtained through double-dummy analysis in the playing stage. The declarer's score function is determined based on the number of tricks won, the contract, the vulnerability of the contracting party, and whether it is doubled or not. Both scores are converted into IMP using the program Score2IMP.

4.5 Evaluation

The Evaluation of two Neural networks was done separately.

Bid Selection Model Initially, Zhang et al. analysed the network structure, the activation function, and the optimizer. They selected three metrics to investigate the network: The total precision, the real precision, which is the precision of the ordered bids, and the unreal precision, which is the precision of no-bids. After testing various setups, the network with four hidden layers, each containing 2048 neurons, the ReLU activation function, and the Adam optimizer produced the highest total precision. The network achieved an overall precision of 95% with a bidding sequence length of 4-8 which reached the expectation.

Evaluation Model The evaluation model metric is the average IMP score of 10,000 decks of cards. A reinforcement training algorithm was used to train the evaluation model. The results of the network improved with more iterations. The network achieved an average IMP score exceeding 0.7 after 1 million iterations.

4.6 Conclusion

In Conclusion, the state-of-the-art method presents a bidding model that supports multi-system bidding using a double neural network. Historical bidding sequences are converted into general feature information to reduce system constraints. The model includes a bid selection network and a state evaluation network for making optimal bidding decisions. The model performs as expected against different bidding systems.[8]

5 Conlusion

To summarise, this literature review presents a thorough examination of the progression of AI techniques in the realm of bridge bidding. The partial-information nature of the game, coupled with the complexity of bidding systems and the strategic decision-making process, has led to the development of innovative approaches.
Early attempts, such as rule-based systems and the introduction of AI players like Wasserman's [5], laid the foundation for bid evaluation understanding and imitation of human judgement. Ginsberg's Intelligent Bridgeplayer[1] marked a significant milestone by demonstrating the potential of simulation-based bidding algorithms. Yegnanarayana's[6] work on opening bids and the use of deep neural networks showcased the promise of machine learning techniques for data representation and feature extraction. Zhang's double neural network model[8], which represents the current state-of-the-art, addresses the difficulties related to multi-system bidding, feature extraction and bid evaluation. This method employs neural networks to automate the decision-making process for bidding, while achieving competitive results against different bidding systems.
These various methods cumulatively contribute to a deep understanding of the complexities involved in bridge bidding. They pave the way for AI systems to play a significant role in this challenging domain. As AI techniques persist to progress, the bridge bidding sphere is likely to encounter more revelations, which

will eventually close the distance between human proficiency and machine capabilities in competitive bridge contests.

References

1. Ginsberg, M.L.: Gib: Steps toward an expert-level bridge-playing program. In: IJCAI. pp. 584–593. Citeseer (1999)
2. Jakobson, R.: Linguistics and poetics. In: Style in language, pp. 350–377. MA: MIT Press (1960)
3. Jamroga, W.: Modelling artificial intelligence on a case of bridge card play bidding. In: Proceedings of the 8th International Workshop on Intelligent Information Systems. pp. 267–277. Citeseer (1999)
4. Tang, J.: How to play contract bridge. Website (August 2023), https://bridgebum.com/how_to_play_contract_bridge.php
5. Wasserman, A.I.: Realization of a skillful bridge bidding program. In: Proceedings of the November 17-19, 1970, fall joint computer conference. pp. 433–444 (1970)
6. Yegnanarayana, B., Khemani, D., Sarkar, M.: Neural networks for contract bridge bidding. Sadhana **21**, 395–413 (1996)
7. Yeh, C.K., Hsieh, C.Y., Lin, H.T.: Automatic bridge bidding using deep reinforcement learning. IEEE Transactions on Games **10**(4), 365–377 (2018)
8. Zhang, X., Lin, R., Bo, Y., Yang, F.: The synergy of double neural networks for bridge bidding. Mathematics **10**(17), 3187 (2022)

6 Appendix: Rules of Bridge[4]

6.1 The Setup

Contract Bridge is an auction and trick-taking card game for four players. The aim of the game is to earn points by winning as many tricks as possible. The players are divided into two teams of two. Each player is designated a compass direction, with partners sitting opposite each other (i.e. East-West will play against North-South). The game is played with a single standard 52-card deck, although having two decks available can move the game along more quickly.
Bridge is played over a series of hands. Each hand consists of two parts: the auction, where the partnerships bid against each other based on how strong they think their cards are; and the play, where teams try to win enough tricks to make or defeat the contract reached in the auction. Points are awarded depending on success or failure, and then another hand begins. Winning enough points earns a partnership a game. Winning two games makes a rubber (except in Duplicate Bridge, where each hand is treated independently).

The Deal One player is designated the dealer. He shuffles the deck and deals them all out clockwise, starting with the player to his left. Each player takes their 13 cards, looks at them and then sorts them. Cards are divided by suit and ordered by rank, with ace being the highest and 2 the lowest. The difference between the suits is important for the auction and scoring, but not for play.

Once the hand is completed, the player to the dealer's left becomes the new dealer and a new hand begins. If you're playing with two decks, the deck not being used for the hand can be shuffled and placed ready for the next dealer to make the game flow more quickly.

6.2 The play of the Hand

Bridge is a trick-taking game. If you know how to play other popular games like Whist, Hearts or Spades then you'll find that the cardplay of Bridge is very similar.
Although the auction comes first, it's easier to understand once you know the basics of play. All you need to know for now is that the auction will determine the contract for the hand. The contract will specify a number of tricks and a suit for trumps (although there may be No Trumps). The team that bids the highest is the declaring side. The other team is the defending side. The declaring side will try to win the number of tricks bid, while the defending side will try to stop them.
A trick is a set of four cards, one from each player. The player to the left of the declarer leads the first card of the hand and play proceeds clockwise until all four players have played. The first card of each trick determines the suit. If possible, players must follow suit by playing a card of the same suit from their hand. If no such card is available, then another card may be discarded, but it cannot contribute to the trick unless it's a trump. The player who contributed the highest ranking card wins the trick for their team and gets to start the next trick.
Once all thirteen tricks have been played, each pair counts up how many they've won. If the declaring sidehas made their bid (or higher) then the contract was successful. If not, the contract was defeated and the defending team will score penalty points.

Trumps If the auction resulted in a suit contract, then than suit will be trumps for the entire hand. Players must still follow suit if they can, but if they've run out of the chosen suit they may play a card from the trump suit instead. This is known as ruffing.
A trump beats cards from all other suits, so ruffing is a very powerful technique. If two or more trumps are played in the same hand, then the highest trump wins the trick.

Dummy The dummy separates Bridge from most other trick-taking games. Every hand, one member of the declaring side sits out as the dummy. Their entire hand is placed faced up on the table for all players to see. Although the order of play doesn't change, the declarer decides which cards to play from both his hand and dummy's.
Once the contract has been decided, the pairs are split into the declaring and

defending sides. The player on the declaring team who first mentioned the contract suit during the auction is the declarer. Their partner will be the dummy. The player to the left of the declarer leads the first card blind, just based on what they can see in their hand. But once this card has been led, the dummy hand is placed face-up on the table for all players to see. For convenience, the cards are arranged in columns with alternating suit colors and the trump suit on the right (declarer's left).

6.3 The Auction

For many players, the auction is the heart of Bridge. This is where the contract that governs the play of the hand will be decided.
Starting with the dealer and proceeding clockwise, players take it in turns to make a bid. A bid will have two components: the number of tricks the partnership is aiming to make, and the suit that will be trumps. The number of tricks will be between 1 and 7. This represents the number of tricks above six that the player thinks his team can make. (This is done because every bid also carries an implicit intention to win most of the tricks. With 13 tricks each hand, 7 is the lowest number that will give one team a majority.)
For example, a bid of 1C means taking at least 7 tricks with clubs as trumps. A bid of 4H means at least 10 tricks with hearts as trumps. A bid of 7NT means taking all thirteen tricks with no trump suit.
Like any other auction, each bid must be higher than the last. The numbers are treated as you would expect, with bids at the 7 level being the highest and bids at the 1 level the lowest. But within each level, the suit rankings come into play. The suits are ranked Clubs, Diamonds, Hearts, Spades, No Trumps, from lowest to highest. (An easy way to remember this is that the suit names are in alphabetical order.) So the lowest possible bid is 1C and the highest is 7NT. The suit ranking means that if, for example, the bid currently stood at 3D, a bid of 3C would not be allowed. If someone wanted clubs to be trumps they would have to bid at the 4 level or higher.
If you do not want to bid you can pass. Unlike some other auction games, passing doesn't forfeit your right to re-enter the auction later. Once three players have passed in a row, the last bid made is the contract for the hand. The person in the winning team who first bid the trump suit will be declarer. Be careful here: often this will be the person who made the final bid, but not always!
If all four players pass before making a bid, the hand has been passed out. No play occurs, and the next dealer deals a fresh hand.
The auction is the chance for players to share information about the strength and weakness of their hand with their partners (naturally, no other communication about the contents of a player's hand is permitted). Most bids will be natural, with the suit choice indicating a preference for that suit and the number indicating hand strength. However, bids can also be used to signal other kinds of information, sometimes completely unrelated to the suit and number of the bid. These prearranged bid signals are collectively known as a bidding system. There are many different kinds of bidding systems with varying levels of complexity.

Players who regularly play together in a partnership will try to employ a bidding system that complements their play style.

Doubeling and Redoubeling Another bidding option is to double. If a player doesn't think their opponents can meet their bid, but doesn't want to be in a contract of his own, he can double. This drastically increases the penalty points for defeating a contract, but it also doubles its value and increases the rewards for making overtricks. A doubled contract can then be redoubled by the other team. Bonuses and penalties for a redoubled contract are twice those for a doubled one.
A double or redouble only stands if the other three players then pass. Any other bid cancels the double, although there's nothing to stop a player from doubling this new bid.
In rubber bridge there is a score bonus for successfully making a doubled or redoubled contract. These are known as points for the insult. The bonus is 50 for a doubled contract and 100 for a redoubled one.

6.4 Scoring

Bridge scoring can seem a little complicated at first, but you'll soon get used to it. Many packs of cards come with a special bridge scoring card that you can use as a reference.
One player is in charge of the scoring. Two columns are headed "WE" and "THEY" to designated the scorer's team and their opponents, respectively. A horizontal line is drawn half-way down. Points that contribute towards a game are written below this line, while bonuses, penalties and other points are written above it. The difference between these two affects the course of play, but at the end of play all of the points are added up, with none being more important than the others.
Contracts are worth different amounts depending on the number of tricks and the suit bid. Clubs and Diamonds are known as the minor suits, and tricks bid and made here are worth 20 points each. Hearts and Spades are the major suits, and are worth 30 points per trick. A No Trump contract is the most valuable and calculating the score can be a little tricky: the first trick is worth 40, whereas subsequent tricks are worth 30.
To match the bidding levels, only the tricks after the sixth are counted for contract points. So a contract of 1C bid and made requires seven tricks but is worth 20 points (1 x 20). A contract of 3H requires nine tricks and is worth 90 points (3 x 30). A contract of 5NT requires eleven tricks and is worth 160 points (40 + 4 x 30).
Points for tricks bid and made go below the line (that horizontal line across the two columns). Once a team has 100 points or more below the line they have won a game. As soon as that happens, a new line is drawn beneath all the scores and the scores below the line are reset. Winning a second game ends the rubber, and the winning team gets a large bonus. This is worth 500 points if the team wins

two games to one, or 700 if they win two games to nil.
Tricks which are made but not bid are known as overtricks. These score the same number of points as bid tricks, but these points are placed above the line and don't count towards making game. If the contract was doubled or redoubled, the points for overtricks are increased to 100 and 200, regardless of suit. This is doubled again if the team making the overtricks is vulnerable.
Because the aim of Bridge is to win games and rubbers, it's helpful to consider the lowest bids for each suit that can win a game outright. For the minor suits this is at the 5 level (5 x 20 = 100). For majors it's at the 4 level (4 x 30 = 120). But the extra 10 points for the first No Trumps trick means that 3NT is sufficient (40 + 3 x 30 = 100). It's also important to remember that doubling and redoubling affects the base value of the contract as well as the bonus for overtricks. This doubled or redoubled value goes below the line, and can allow a team to score game at a much lower level than usual!

Slams Because only a hundred points are needed for game, it may seem that there is little point in bidding at the higher levels. But there are special incentives for bidding and making contracts of twelve or thirteen tricks. Making twelve tricks is known as a small slam and is worth 500 points. Bidding and winning all thirteen tricks is known as a grand slam and gets a bonus of 1000. These bonuses are not affected by doubling, but if the pair making the slam is vulnerable then the bonus is increased by 50

Vulnerability Once a team has won one game they become vulnerable. A vulnerable side faces far stiffer penalties for missing their contracts. Conversely, the rewards for slams and overtricks are increased. Vulnerability is a key bridge concept that affects a lot of the strategy, especially when both sides are at different vulnerabilities. The tables below show the different points and penalties available at different vulnerabilities.

Honor Points In rubber bridge there are a few miscellaneous bonuses for holding certain high cards. The top five cards in a suit (T, J, Q, K, A) are known as the honors. Having four trump honors in a single hand is worth 100 points, and having all five is worth 150. In a No Trump hand, having all four aces scores 150 points.

Optimale Kernelauswahl für Gauß Prozess Regression

Arne Fynn Haß[1]

Karlsruher Institut für Technologie,
Kaiserstraße 12, 76131 Karlsruhe, Germany
https://www.kit.edu

Abstract. In der Gauß Prozess Regression (GPR) ist die Auswahl eines geeigneten Kernels von entscheidender Bedeutung für die Leistungsfähigkeit und Genauigkeit des Modells. Die Wahl des richtigen Kernels bestimmt maßgeblich, wie gut das GPR-Modell die zugrunde liegenden Beziehungen zwischen den Datenpunkten erfasst und präzise Vorhersagen trifft. In diesem Paper wird Gauß Prozess Regression vorgsetllt und dargeboten wie man auf einen optimalen Kernel für das gegebene Modell kommt.

Keywords: Gauß Prozess Regression · Kernel · Hyperparameter

1 Einleitung

Das Maschinelle Lernen hat in den letzten Jahren eine rasante Entwicklung erlebt und gilt heute als eine der spannendsten und vielversprechendsten Disziplinen in der Informatik und Datenwissenschaft. Es befasst sich mit der Entwicklung von Algorithmen und Techniken, die es Computern ermöglichen, aus Daten zu lernen und auf Grundlage dieses erworbenen Wissens Vorhersagen zu treffen und Aufgaben zu automatisieren.

Ein besonders interessanter Ansatz im Bereich des Maschinellen Lernens ist die sogenannte "Gauß Prozess Regression" (GPR). Sie stellt eine leistungsstarke Methode dar, um komplexe Zusammenhänge in Daten zu erfassen und präzise Vorhersagen zu treffen. Anders als viele traditionelle Modelle bietet GPR eine flexible und probabilistische Herangehensweise, die nicht nur genaue Vorhersagen liefert, sondern auch die Unsicherheit in den Vorhersagen berücksichtigt.

Ob in der medizinischen Diagnostik [1], in der Robotik [3], der Finanzanalyse [8] oder in der Umweltüberwachung [6] - Maschinelles Lernen und Gauß Prozess Regression haben das Potenzial, unser Verständnis von Daten zu revolutionieren und innovative Lösungen für komplexe Probleme zu liefern.

2 Gauß Prozess Regression

Gauß Prozess Regression (GPR) ist ein leistungsstarkes und vielseitiges Werkzeug zur Modellierung und Vorhersage von Daten. Es gehört zur Klasse der nichtparametrischen Methoden, was bedeutet, dass es keine festen Funktionsformen

oder feste Anzahl von Parametern für das Modell gibt. Stattdessen basiert GPR auf der Idee der probabilistischen Modellierung von Funktionen.

Im Gegensatz zu klassischen linearen Regressionen, die eine einzige Funktion zur Approximation von Daten verwenden, modelliert GPR eine ganze Verteilung von Funktionen. Genauer gesagt, wird ein Gauß Prozess (GP) als eine unendliche Sammlung von Zufallsvariablen betrachtet, wobei jede dieser Variablen eine Funktion repräsentiert [5]. Die Verteilung dieser Funktionen wird durch einen Kernel definiert[9].

Der Kernel (Kovarianzfunktion) spielt eine zentrale Rolle in der GPR, da er die Ähnlichkeit zwischen verschiedenen Eingabedatenpunkten quantifiziert [5]. Wenn zwei Datenpunkte ähnlich sind, wird ihre Kovarianz hoch sein, und wenn sie sich stark unterscheiden, wird ihre Kovarianz niedrig sein. Dieser Ansatz ermöglicht es GPR, flexible Modelle zu erzeugen, die komplexe nicht-lineare Zusammenhänge zwischen den Eingangsvariablen und den Ausgaben erfassen können.

Ein weiterer entscheidender Vorteil von GPR ist die Möglichkeit, die Vorhersagen mit Unsicherheiten zu quantifizieren. Da GPR eine probabilistische Methode ist, liefert sie für jede Vorhersage eine Wahrscheinlichkeitsverteilung. Dies ermöglicht eine robuste und zuverlässige Schätzung der Vorhersagegenauigkeit und ist besonders nützlich in Situationen, in denen Unsicherheiten eine Rolle spielen, wie bei den Bespielen in der Einleitung genannt.

2.1 Gauß Prozess

Wie eben beschrieben ist ein GP eine Sammlung von Zufallsvariablen, von denen jede eine gemeinsame Gauß Verteilung hat, wenn eine endliche Anzahl von Zufallsvariablen betrachtet wird [5].

Ein Gauß Prozess wird vollständig durch seine Mittelwertfunktion und seinen Kovarianzfunktion definiert [5]. Die Mittelwertfunktion $m\left(x\right)$ und die Kovarianzfunktion $k(x, x^{'})$ für einen realen Prozess $f(x)$ sind folgendermaßen definiert:

$$m(x) = \mathbb{E}[f(x)] \tag{1}$$

$$k\left(x, x^{'}\right) = \mathbb{E}[(f\left(x\right) - m\left(x\right))\left(f\left(x^{'}\right) - m\left(x^{'}\right)\right)] \tag{2}$$

Mithilfe dieser Funktionen, wobei $\mathbb{E}$ die Erwartung repräsentiert, können wir eine Verteilung über die Funktionen $f(x)$ definieren:

$$f(x) \sim \mathcal{GP}\left(m(x), k\left(x, x^{'}\right)\right) \tag{3}$$

Die Mittelwertfunktion gibt das durchschnittliche Verhalten der vom Prozess generierten Funktionen an. Man kann sehen, dass die Kovarianzfunktion der

Schlüssel zur Modellierung der Abhängigkeiten zwischen den Datenpunkten ist [9]. Sie gibt an, wie ähnlich oder korreliert die Funktionswerte an verschiedenen Stellen im Eingaberaum sind.

2.2 Vorhersagen

Um realistische Vorhersagen aus den Trainingspunkten $X = \{(x_i, y_i)|i = 1, ..., n\}$, mit den Funktionswerten von x_i mit Rauschen $y_i = f(x_i) + \epsilon$, zu erhalten benutzen wir das Modell aus [5]. Unter der Annahme von additiven, unabhängigen und identisch verteilten gaußschem Rauschen ϵ mit Varianz σ_n^2 wird die Kovarianz der rauschbehafteten Beobachtungen zu

$$cov(y) = K(X, X) + \sigma_n^2 I \tag{4}$$

wobei $K(X, X)$ die Kovarianzmatrix ist, welche durch $K_{ij}(X, X) = k(x_i, x_j)$, mit $x_i, x_j \in X$, definiert ist. Die gemeinsame Verteilung der beobachteten Trainigswerte y und die Funktionswerte f_* der Testpunkte X_* gemäß der Kovarianz ist

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) \tag{5}$$

Die wesentlichen Vorhersagenverteilungen, die wir in [5] gegeben haben, sind

$$f_*|X, y, X_* \sim \mathcal{N}\left(\bar{f}_*, cov\left(f_*\right)\right), mit \tag{6}$$

$$\bar{f}_* = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} y, \tag{7}$$

$$cov\left(f_*\right) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]_{-1} K(X, X_*). \tag{8}$$

2.3 Eigenschaften vom Kernel

Eine valider Kernel ist positiv semidefinit [5]. Die meisten Kernel haben Hyperparameter, die eingestellt werden müssen, um ihre Leistung zu optimieren. Die Wahl und Anpassung dieser Hyperparameter ist ein wichtiger Schritt bei der Auswahl eines Kernel-Typs.

2.4 Typische Kernel-Typen in der GPR

Um Kernel besser verstehen zu können werden im folgenden Beispiele für diese aufgelistet:

SE-Kernel (Squared Exponential Kernel) [4]:

$$k_{SE}\left(x, x^{'}\right) = \sigma^2 \exp\left(-\frac{\left(x - x^{'}\right)^2}{2l^2}\right) \tag{9}$$

Hierbei repräsentieren x und $x^{'}$ die Eingabewerte der Datenpunkte, l ist der Längenmaßstab, der die Reichweite der Korrelation zwischen den Funktionswerten bestimmt. σ bestimmt die durchschnittliche Entfernung der Funktion von ihrem Mittelwert. Jeder Kernel hat diesen Parameter vorne, es ist nur ein Skalierungsfaktor. [4]

Der SE Kernel, wird oft für glatte, kontinuierliche Funktionen verwendet.

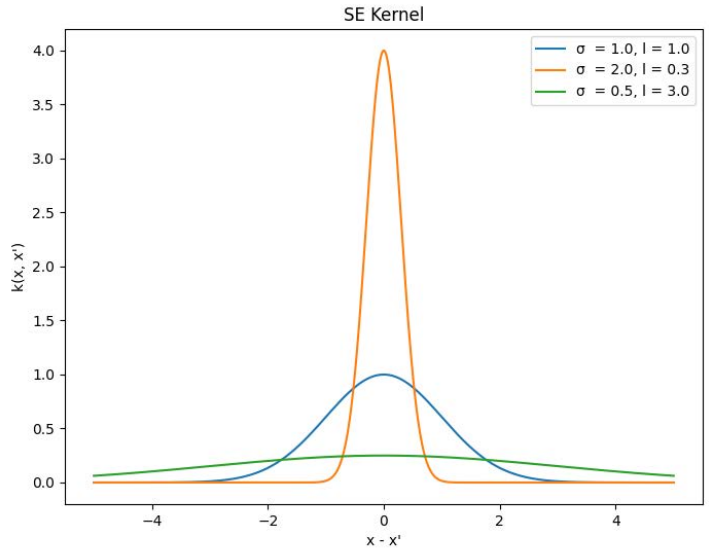


Fig. 1. Die SE-Kernel geplottet mit $x^{'} = 0$ und verschiedenen Hyperparamtern.

Periodic Kernel [4]:

$$k_{Per}\left(x, x^{'}\right) = \sigma^2 \exp\left(-\frac{2sin^2\left(\pi|x - x^{'}|/p\right)}{l^2}\right) \tag{10}$$

Modelliert Funktionen die sich genau wiederholen. Die Hyperparameter sind p, welcher die Periode bestimmt und l, der wie beim SE der Längenmaßstab ist.

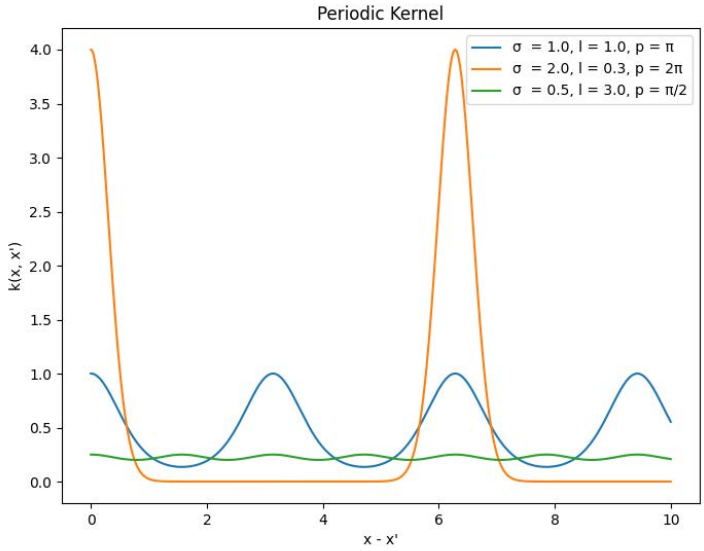


Fig. 2. Die Periodic-Kernel geplottet mit $x^{'} = 0$ und verschiedenen Hyperparamtern.

Matérn Kernel [5]:

$$k_{Matern}\left(x, x^{'}\right) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\left(x - x^{'}\right)}{l} \right)^{\nu} K_{\nu} \left(\sqrt{2\nu} \frac{\left(x - x^{'}\right)}{l} \right) \tag{11}$$

Mit positiven Hyperparametern ν und l. K_{ν} ist eine modifizierte Bessel Funktion. Hierbei ist ν ein Glätteparameter und l wie auch schon oben unser Längenmaßstab. Wie in [5] erhalten wir für $\nu \to \infty$ den SE-Kernel. Der Matérn Kernel ist flexibler als der SE Kernel und eignet sich für Datensätze mit einer gewissen Unregelmäßigkeit. Er kann genutzt werden, um Prozesse zu modellieren, die oft nicht perfekt glatt sind.

Es gibt auch andere Kernel-Typen, die verschiedene Modellierungseigenschaften aufweisen. Zudem können wir verschiedene Kernel kombinieren, zum Beispiel kann man Kernel miteinander addieren und multiplizieren wie in [4] gezeigt. Die Wahl des Kernels hängt von der Struktur der Daten und den gewünschten Modellierungseigenschaften ab. Eine sorgfältige Kernel-Auswahl kann die Leistung des GPR-Modells erheblich verbessern und eine effektive Modellierung komplexer Beziehungen zwischen den Datenpunkten ermöglichen. Durch die Definition des Kernels können wir die Funktionsweise des GPR-Modells steuern und es an die spezifischen Anforderungen unserer Daten anpassen.

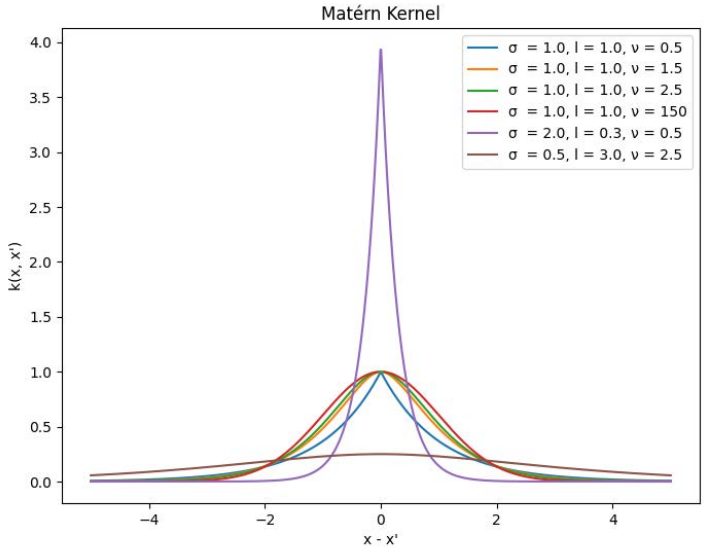


Fig. 3. Die Matern-Kernel geplottet mit $x^{'} = 0$ und verschiedenen Hyperparamtern.

3 Datenanalyse und Verständnis

Die Datenanalyse und das Verständnis der zugrunde liegenden Daten sind von entscheidender Bedeutung bei der Anwendung der GPR. Diese Phase bildet den Grundstein für die erfolgreiche Modellierung und die Auswahl eines geeigneten Kernels, um präzise Vorhersagen zu treffen.

In der Phase der Datenanalyse beginnen wir damit, die Struktur und Eigenschaften der Daten zu erkunden. Dazu gehört die Untersuchung der Datentypen, Skalierungen, Verteilungen, Korrelationen und möglichen Ausreißern. Ein gründliches Verständnis der Daten ermöglicht es uns, potenzielle Herausforderungen oder Besonderheiten zu identifizieren, die bei der späteren Modellierung berücksichtigt werden müssen.

Dabei spielt Visualisierung der Daten eine entscheidende Rolle. Streudiagramme, Histogramme, Boxplots und Zeitreihenplots ermöglichen es uns, Muster, Trends und mögliche Abhängigkeiten zwischen den Variablen zu erkennen. Die Visualisierung der Daten hilft uns, ein intuitives Gefühl für die zugrunde liegende Struktur zu entwickeln und wichtige Einsichten zu gewinnen.

Die Identifizierung von Mustern und Abhängigkeiten in den Daten ist ein weiterer wichtiger Aspekt der Datenanalyse. Wir suchen nach linearen und nicht-

linearen Zusammenhängen, periodischen Mustern und Autokorrelationen [5].

Das Verständnis der Datenstrukturen ist unerlässlich für die Auswahl eines geeigneten Kernels. Unterschiedliche Kernel-Typen sind für verschiedene Datentypen und Muster besser geeignet. Zum Beispiel ist der SE-Kernel gut geeignet, um kontinuierliche Muster zu erfassen, während der periodische Kernel besonders gut für Daten mit periodischen Schwankungen geeignet ist.

Insgesamt spielt die Datenanalyse und das Verständnis der Daten eine zentrale Rolle bei der Anwendung der Gauß Prozess Regression. Durch eine gründliche Analyse und Visualisierung der Daten sowie die Identifizierung von Mustern und Abhängigkeiten können wir spezifische Anforderungen und Eigenschaften an unser GPR-Modell stellen.

4 Modellierungseigenschaften

Diese eben erwähnten Modellierungseigenschaften müssen, für einen geeigneten Kernel klar definiert sein. Sie legen fest, welche Charakteristiken das GPR-Modell haben soll, um den Anforderungen der Anwendung gerecht zu werden. Aspekte wie Flexibilität, Berücksichtigung von Unsicherheiten und Zeit/Raum-Abhängigkeiten sollten beachtet werden [5].

Die Flexibilität des GPR-Modells bezieht sich darauf, wie gut es sich an die zugrunde liegenden Datenstrukturen anpassen kann. Ein guter Kernel sollte in der Lage sein, sowohl einfache als auch komplexe Muster in den Daten zu erfassen. Die gewählte Modellierungseigenschaft sollte sicherstellen, dass das GPR-Modell ausreichend flexibel ist, um die Vielfalt der Daten angemessen zu berücksichtigen, also sollte es nicht zur Unteranpassung bzw. Überanpassung kommen [5]. Unteranpassung tritt auf, wenn ein Modell zu einfach ist, um die zugrunde liegenden Muster in den Daten zu erfassen. Dies führt zu einer Situation, in der die Leistung des Modells sowohl auf den Trainings- als auch auf den Testdaten schlecht ist. Das Modell fehlt an der benötigten Komplexität, um die Beziehungen innerhalb der Daten darzustellen, was zu geringer Genauigkeit führt (siehe Fig. 4). Überanpassung können wir beobachten, wenn ein Modell lernt, auf den Trainingsdaten außergewöhnlich gut abzuschneiden, jedoch nicht in der Lage ist, auf neue, ungesehene Daten zu verallgemeinern. Mit anderen Worten, das Modell hat das Rauschen in den Trainingsdaten gelernt anstelle der zugrunde liegenden Muster. Es kann selbst die kleinsten Schwankungen in den Trainingsdaten erfassen, was zu einem komplexen Modell mit vielen Parametern führt. Diese Komplexität macht das Modell anfällig für Fehler bei neuen Daten, was zu einer schlechten Vorhersageleistung führt (siehe Fig. 4).

In vielen Anwendungen sind Unsicherheiten in den Daten vorhanden, sei es aufgrund von Messfehlern oder begrenzter Datenverfügbarkeit. Die Definition der Modellierungseigenschaften sollte sicherstellen, dass das GPR-Modell die Un-

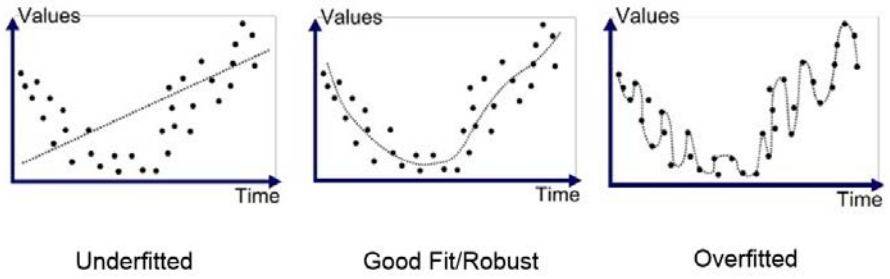


Fig. 4. Links Unteranpassung, Mitte Gute Abdeckung, Rechts Überanpassung. [2]

sicherheiten in den Vorhersagen berücksichtigt und plausible Konfidenzintervalle liefert [5].

Je nach Anwendung kann es wichtig sein, die zeitliche oder räumliche Struktur der Daten zu berücksichtigen. Ein passender Kernel sollte die Abhängigkeiten zwischen den Datenpunkten entsprechend ihrer zeitlichen oder räumlichen Nähe angemessen erfassen [5].

Die Definition der gewünschten Modellierungseigenschaften hilft uns, den Fokus bei der Kernel-Auswahl zu setzen und im nächsten Schritt den geeigneten Kernel-Typ sowie die entsprechenden Hyperparameter auszuwählen.

5 Kernel-Auswahl und Hyperparameter-Tuning

Nach der Bestimmung der gewünschten Modelleigenschaften beginnt das iterative Verfahren für die Kernel Auswahl. Es wird als iterativ bezeichnet, da sehr wahrscheinlich nicht der erste gewählte Kernel der optimalste für das Modell ist.

Wir wählen nun einen geeigneten Kerneln für unsere Daten. Sobald der Kernel gewählt wurde müssen wir die optimalen Hyperparameter für das Modell finden.

Der erste Schritt beim Hyperparameter-Tuning ist die Festlegung eines geeigneten Bereichs für jeden Hyperparameter. Je nach Art des Kernels können die Hyperparameter beispielsweise die Längenmaßstäbe, die Rauschintensität oder die Parameter, die die Form des Kernels steuern, umfassen. Es ist wichtig, den Bereich so zu wählen, dass er plausible und realistische Werte abdeckt.

Eine der verbreitetsten Methoden für die Bestimmung der Hyperparameter, ist die Maximierung der log marginal likelihood, welche in [5] folgendermaßen definiert ist:

$$\log p(y|X,\theta) = -\frac{1}{2}y^\top K_y^{-1} y - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi \tag{12}$$

Hierbei ist $K_y = K + \sigma_n^2 I$ die Kovarianzmatrix für die mit Rauschen behafteten Trainingspunkte. θ ist die Menge aller Hyperparameter, zum Beispiel bei der oben gezeigten SE-Kernel wäre $\theta = \{\sigma, l\}$.

Die log marginal likelihood kann in drei Teile aufgeteilt werden: Der data-fit-Term $-y^\top K_y^{-1} y/2$; der complexity penalty $\log|K_y|/2$ und der normalization constant $n\log(2\pi)/2$. Wie in [5] gezeigt nimmt der data-fit-Term monoton ab, je größer der Längenmaßstab wird, da das Modell immer weniger flexibel wird. Der negative complexity penalty nimmt mit wachsendem Längenmaßstab zu, da das Modell mit zunehmendem Längenmaßstab weniger komplex wird.

Da bei großen Datensätzen die Berechnung der log marginal likelihood sehr zeitaufwendig ist, da die Funktion darauf aufbaut die Matrix K zu invertieren, was eine Laufzeit von $\mathcal{O}(n^3)$, für eine $n * n$-Matrix hat. In solchen Fällen können Hyperparameter-Optimierungsalgorithmus effizienter sein.

Einige solcher Algorithmen werden von Yang und Shami in [10] vorgestellt. Die meist genutzten werden wir nun ebenfalls kurz vorstellen.

Der verbreitetste Hyperparameter-Optimierungsalgorithmus ist Grid-Search(GS). Hierbei wird der gesamte Wertebereich für jeden Hyperparameter in diskrete Abschnitte unterteilt. Das GPR-Modell wird dann für jede Kombination von Hyperparameterwerten trainiert und evaluiert, basierend auf Error-Metriken wie z.B dem Mean Squared Error (MSE). Die beste Kombination von Hyperparametern, die die besten Leistungsergebnisse liefert, wird ausgewählt. Ein Vorteil von GS ist, dass es sehr einfach zu implementieren und paralellisiert ist. Der große Nachteil von GS ist, dass der Algorithmus ineffizient für Hyperparameter mit mehreren Dimensionen ist wie in [10] beschrieben.

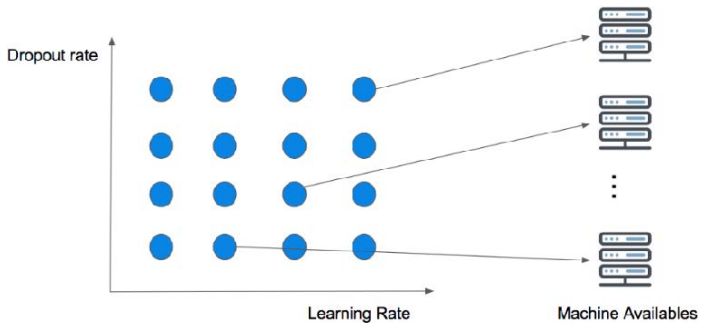


Fig. 5. Grid search über Ausfallraten und Lernraten in einer parallelen Ausführung. [11].

Random Search(RS) ist ein weiterer Optimierungsalgorithmus, er funktioniert ähnlich wie Grid Search, jedoch nimmt er eine feste Anzahl an zufälligen Kombinationen von Hyperparameter. Mit diesen wird das GPR-Modell wie bei GS trainiert und evaluiert. Die beste Kombination wird, dann ausgewählt. RS ist mehr effizient als GS, wie in [10] gezeigt. Der große Nachteil den GS und RS ist, dass sie nicht aus vorherigen Ergebnissen in der Iteration lernen, da alle Iteration unabhängig von den davor ist. Dadurch wird viel Zeit verschwendet, im Gegensatz zu Algorithmen die ihre nächste Evaluierung auf der vorherigen aufbauen.

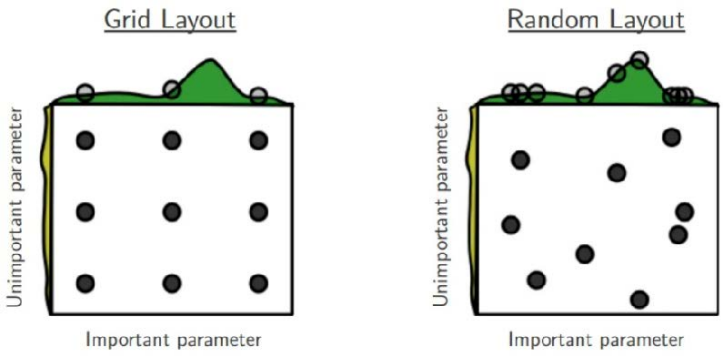


Fig. 6. Anordnungsvergleich von grid search and random search. [11]

Ein solcher Algorithmus ist die Bayesian Optimization. Die BO nutzt ein probabilistisches Modell, um Vorhersagen über die Leistung des GPR-Modells für verschiedene Hyperparameterkombinationen zu treffen. Basierend auf den gemachten Vorhersagen werden neue Hyperparameterwerte ausgewählt, die voraussichtlich die beste Leistung erzielen. Dieser Prozess wird iterativ wiederholt, um die Hyperparameter kontinuierlich zu verbessern (siehe Fig. 7). Für eine genauere Erklärung siehe [10]. Durch die beschriebene Funktionsweise der BO ist dieser Algorithmus schneller als GS uns RS. Ein Nachteil der BO ist jedoch, dass er schwierig zu parallesieren ist.

Eine kompakte Darstellung der Vor-, Nachteile und Laufzeit aller Algorithmen, welche in [10] vorgestellt werden, ist in Fig. 8.

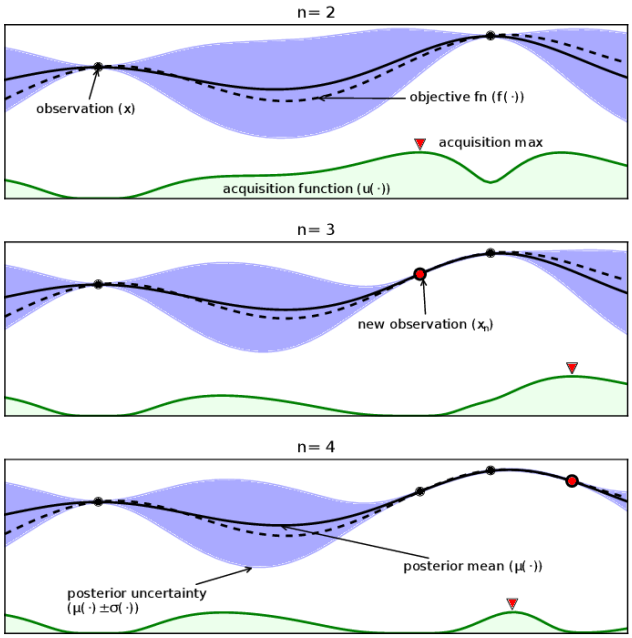


Fig. 7. Illustration von Bayesian optimization über drei Iterationen. [7]

HPO Method	Strengths	Limitations	Time Complexity
GS	· Simple.	· Time-consuming, · Only efficient with categorical HPs.	$O(n^k)$
RS	· More efficient than GS. · Enable parallelization.	· Not consider previous results. · Not efficient with conditional HPs.	$O(n)$
Gradient-based models	· Fast convergence speed for continuous HPs.	· Only support continuous HPs. · May only detect local optimums.	$O(n^k)$
BO-GP	· Fast convergence speed for continuous HPs.	· Poor capacity for parallelization. · Not efficient with conditional HPs.	$O(n^3)$
SMAC	· Efficient with all types of HPs.	· Poor capacity for parallelization.	$O(nlogn)$
BO-TPE	· Efficient with all types of HPs. · Keep conditional dependencies.	· Poor capacity for parallelization.	$O(nlogn)$
Hyperband	· Enable parallelization.	· Not efficient with conditional HPs. · Require subsets with small budgets to be representative.	$O(nlogn)$
BOHB	· Efficient with all types of HPs. · Enable parallelization.	· Require subsets with small budgets to be representative.	$O(nlogn)$
GA	· Efficient with all types of HPs. · Not require good initialization.	· Poor capacity for parallelization.	$O(n^2)$
PSO	· Efficient with all types of HPs. · Enable parallelization.	· Require proper initialization.	$O(nlogn)$

Fig. 8. Der Vergleich gängiger Hyperparameter-Optimierungsalgorithmen (n ist die Anzahl der Hyperparameter-Werte und k ist die Anzahl der Hyperparameter). [10]

6 Modellvalidierung und -vergleich

Nachdem wir den geeigneten Kernel ausgewählt und die Hyperparameter feingetunt haben, ist es entscheidend, das erstellte GPR-Modell auf seine Leistungsfähigkeit zu überprüfen und mit anderen Modellen zu vergleichen. Die Modellvalidierung und der -vergleich helfen uns dabei, die Vorhersagegenauigkeit und die Stabilität unseres Modells zu bewerten und sicherzustellen, dass es für unsere spezifische Anwendung angemessen ist. Was wichtig ist hier anzumerken ist, dass dies nicht gemacht werden muss, wenn ein Hyperparameter-Optimierungsalgorithmus angewendet wurde, da die erstellten Modelle automatisch miteinander verglichen und validiert werden, auch mit den Methoden die in diesem Abschnitt vorgestellt werden.

Die Kreuzvalidierung ist eine leistungsfähige Technik, um die Modellgenauigkeit zu verbessern und die Variabilität der Ergebnisse zu reduzieren. Beim k-Fold Cross-Validation-Verfahren wird der Trainingsdatensatz in k Teilmengen aufgeteilt, und das Modell wird k-mal trainiert und getestet, wobei jeweils eine andere Teilmenge als Testdatensatz verwendet wird. Die Ergebnisse werden dann gemittelt, um eine zuverlässigere Leistungsbewertung zu erhalten [5].

Um die Genauigkeit der Vorhersagen zu bewerten, verwenden wir verschiedene Bewertungsmetriken, welche in [5] erwähnt werden, wie den Mean Squared Error (MSE), den Standardized Mean Squared Error (SMSE) oder den Mean Standardized Log Loss (MSLL). Diese Metriken helfen uns dabei, die Unterschiede zwischen den tatsächlichen und vorhergesagten Werten zu quantifizieren und die Leistung des Modells zu beurteilen.

Die Modellvalidierung und der -vergleich sind kritische Schritte, um sicherzustellen, dass unser GPR-Modell zuverlässige Vorhersagen liefert und gut auf unsere spezifische Anwendung zugeschnitten ist. Durch die sorgfältige Bewertung der Modelleistung und den Vergleich mit anderen Modellen können wir die Stärken und Schwächen unseres GPR-Modells besser verstehen und seine Vorhersagegenauigkeit optimieren. Ein gut validiertes und verglichenes Modell gibt uns das Vertrauen, dass es präzise Vorhersagen liefert und uns wertvolle Erkenntnisse aus unseren Daten liefert.

Nach all diesen Schritten haben wir nun einen Kernel, der der unsere gegebenen Trainingspunkte abdeckt und trotzdem gute Vorhersagen für unsere Testpunkte liefert.

7 Anwendungsbeispiele

7.1 Schätzung des Alters als Maß für Gehirn Abnormalitäten

In ihrem Paper zeigen Becker et al. [1] zeigen wie die Anwendung von Gauß Prozessen verwendet werden kann um Altersschätzungen zu liefern. Die Studie konzentriert sich darauf, wie Vorhersagenverteilungen, wie die oben beschriebene Eq.(8), die von Gauß'schen Prozessen bereitgestellt werden, als Indikator für Abnormalitäten im Gehirn dienen können. Dies passiert durch die Annahme, dass die geschätzte Kovarianz als ein Maß der Unsicherheit für einen getesteten Punkt dient.

Becker et al. [1] verwenden eine Summe von mehreren SE-Kerneln als ihren schlussendluchen Kernel. Dies ist eine gute Wahl, da sie dadurch die im gesamten stetige und glatte Steigung in der Beziehung zwischen geschätztem und wahrem Alter darstellen können. Der lineare Kernel siehe [4] wurde hier nicht verwendet, obwohl er hier logisch erscheint, da er ein non-stationary Kernel ist, im Gegensatz zu SE-Kerneln. Ein nicht-stationärer Kernel modelliert Variationen und Veränderungen im Datenraum. Das bedeutet, dass die Beziehung zwischen den Datenpunkten nicht überall gleich ist. Diese Eigenschaft haben wir hier nicht gegeben, deshalb wird der Lineare Kernel nicht verwendet.

7.2 Automatisierte Überwachung der Grundwasserqualität

Das Paper [6] beschäftigt sich mit einer automatisierten Methode zur Überwachung der Grundwasserqualität unter Verwendung einer Kombination aus GPR und dem Bayesischen Informationskriterium (BIC). Shadrin et al. adressieren die Herausforderung, hochauflösende Karten der Grundwasserqualität zu erstellen, um Umweltüberwachung und -management zu unterstützen. Sie verwenden GPR, um räumliche Muster in den Grundwasserqualitätsdaten zu modellieren und Vorhersagen für unbekannte Orte zu generieren. Das BIC wird eingesetzt, um die optimale Anzahl von GPR-Modellen und deren Hyperparametern zu bestimmen, was zur Reduzierung von Überanpassung und zur Verbesserung der Modellgenauigkeit beiträgt.

Wenn die Methode auf die vorhanden Daten im Paper angewendet wird erhalten Shadrin et al. einen optimalen Kernel als Summe von Summen von SE-Kerneln und Summen von Periodic-Kerneln. Dies ist passend, da dadurch die periodischen und zugleich die kontinuierlichen Eigenschaften in den Daten abgedeckt werden.

8 Zusammenfassung

Die Auswahl eines geeigneten Kernels ist ein entscheidender Schritt bei der Anwendung der Gauß Prozess Regression auf ein bestimmtes Problem. Eine sorgfältige Kernel-Auswahl ermöglicht es uns, die Struktur der Daten angemessen zu erfassen und ein GPR-Modell zu erstellen, das genau und zuverlässig ist. Die Kernel-Auswahl in der Gauß Prozess Regression erfordert eine Kombination aus Datenanalyse, Bestimmung der Modelleigenscheiften, iterativer Verbesserung des Modells und kontinuierlichen Modellvergleichen. Indem wir diese Schritte systematisch durchlaufen, können wir ein GPR-Modell entwickeln, das die zugrunde liegende Struktur der Daten effektiv erfasst und zu präzisen Vorhersagen führt.

References

1. Becker, B.G., Klein, T., Wachinger, C., Initiative, A.D.N., et al.: Gaussian process uncertainty in age estimation as a measure of brain abnormality. NeuroImage **175**, 246–258 (2018)
2. Bhande, A.: What is underfitting and overfitting in machine learning and how to deal with it. URL: https://medium. com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it (2018)
3. Deisenroth, M.P., Fox, D., Rasmussen, C.E.: Gaussian processes for data-efficient learning in robotics and control. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(2), 408–423 (2015). https://doi.org/10.1109/TPAMI.2013.218
4. Duvenaud, D.: The kernel cookbook: Advice on covariance functions. URL https://www.cs.toronto.edu/ duvenaud/cookbook/k (2014)
5. Rasmussen, C.E., Williams, C.K., et al.: Gaussian processes for machine learning, vol. 1. Springer (2006)
6. Shadrin, D., Nikitin, A., Tregubova, P., Terekhova, V., Jana, R., Matveev, S., Pukalchik, M.: An automated approach to groundwater quality monitoring—geospatial mapping based on combined application of gaussian process regression and bayesian information criterion. Water **13**(4), 400 (2021)
7. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N.: Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE **104**, 148–175 (2016), https://api.semanticscholar.org/CorpusID:14843594
8. Wang, J., Yam, W.K., Fong, K.L., Cheong, S.A., Wong, K.M.: Gaussian process kernels for noisy time series: Application to housing price prediction. In: Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part VI 25. pp. 78–89. Springer (2018)
9. Wilson, A., Adams, R.: Gaussian process kernels for pattern discovery and extrapolation. In: International conference on machine learning. pp. 1067–1075. PMLR (2013)
10. Yang, L., Shami, A.: On hyperparameter optimization of machine learning algorithms: Theory and practice. arXiv preprint arXiv:2007.15745 (2020)
11. Yu, T., Zhu, H.: Hyper-parameter optimization: A review of algorithms and applications. arXiv preprint arXiv:2003.05689 (2020)

Training Gaussian Process Regression: Hyper-parameter Optimization Methods and the Indefinite Covariance Matrix

Beyza Keskin[1]
Supervisor: Paul Tremper[2]

[1] uutcl@student.kit.edu
[2] tremper@teco.edu

Abstract. This paper delves into the training phase of Gaussian Process Regression (GPR), with a primary focus on optimizing hyper-parameters and addressing the challenge of the indefinite covariance matrix. At the beginning a comprehensive insight into GPR will be provided. To tackle the problem of hyper-parameter optimization, we introduce several widely employed methods, including grid search, random search and Bayesian optimization. By analysing the strengths and weaknesses of each method, optimal use cases of these methods are highlighted. In the final section, we will introduce a common technique known as regularization, which is employed to handle indefinite covariance matrices.

Keywords: Gaussian Process Regression · Hyper-parameter Optimization · Grid Search · Random Search · Bayesian Optimization · Indefinite Covariance Matrix · Regularization

1 Introduction

Gaussian Process Regression (GPR) is an essential interpolation method that uses Bayesian Interference to provide predictions with a measure of uncertainty. It is used in various fields, such as machine learning, surrogate modelling, robotics and computer vision due to its high flexibility and adaptability.

The training phase of the GPR is crucial to make accurate predictions and associated uncertainties. However GPR depends significantly on hyper-parameters such as kernel type, length scales, noise variance and regularization parameter which directly impact the accuracy, flexibility, and generalization ability of the GPR model. To combat this problem, there are a wide variety of hyper-parameter optimization methods used throughout the training phase of the GPR. In this paper some of popular hyper-parameter optimization methods used in GPR will be introduced and their strengths, weaknesses and optimal use cases will be discussed.

Another challenge that emerges during the training phase of the GPR is the covariance matrix becoming indefinite, which can lead to numerical instabilities and poor model performance. At the end of the paper, appropriate techniques that have been proposed and widely used in practice to deal with this problem will be introduced.

2 Gaussian Process Regression

Before explaining various hyper-parameter optimization methods used in GPR, this section will explain the fundamental concept behind the GPR and the necessity for effective optimization methods for hyper-parameters.

2.1 Overview

GPR is a powerful tool in machine learning because it enables estimation of both predictions and associated uncertainties. GPR models the function as a distribution over functions and not as a fixed parameterized model, which makes it highly flexible and adaptable.

2.2 Gaussian processes regression model

The purpose of the regression tasks is to estimate a function f given a random number of input variables, in other words a n-dimensional vector x. In this context, it is important to acknowledge that there can be an infinite number of functions that can fit the given data. To address this challenge, GPR places a Gaussian prior on function values $y = f(x)$, which means that it assumes a Gaussian distribution over all possible functions that fit the given data:

$$p(y) = \mathcal{N}(y \,|\, \mu(x), k(x, x)) \tag{1}$$

Here, $p(y)$ is the probability distribution of the target variable y, which is assumed to follow a Gaussian distribution $\mathcal{N}$ with a mean function $\mu(x)$ and a covariance matrix K. The mean function $\mu(x)$ represents the average behavior of the function $f(x)$ at the input point x.

The covariance function k reflects the correlation between different input points. In that matter similar inputs correspond to similar outputs.

In prior distribution is no data observed yet. According to Wang in [1], when we start to have observations, instead of infinite numbers of functions, we only keep functions that fit the observed data points. This process is based on Bayesian inference principles and the updated distribution called the posterior distribution. We aim to investigate the connection between the target variable observed during training y and the target variable predicted for new test points y^*. The joint probability can be expressed as follows:

$$p\begin{pmatrix} y \\ y^* \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} y \\ y^* \end{bmatrix} \middle| \; 0, \begin{bmatrix} K(X,X) & K(X,x_*) \\ K(X,x_*)^T & k(x_*,x_*) \end{bmatrix}\right)$$

Here, the mean vector is is assumed to be zero to establish a baseline, $K(X,X)$ is the covariance matrix between the observed target variables, based on the inputs X. $K(X,x_*)$ is the covariance between the observed target variables and the target variables at new test points and finally $K(x_*,x_*)$ represents the covariance between the target variables at new test points.

By conditioning on the observations, we obtain the posterior distribution on the test output y_*:

$$p(y_*|x_*,x_1,\ldots,x_M,y) = \mathcal{N}(y_*|\mu_*,\sigma_*)$$

Here, μ, the mean of the predicted target variable y_* is typically represented as:

$$\mu_*(x_*,x_1,\ldots,x_M,y) = K(x_*,X)K(X,X)^{-1}y$$

and σ represents the uncertainty associated with the prediction:

$$\sigma_*(x_*,x_1,\ldots,x_M,y) = K(x_*,X)K(X,X)^{-1}K(X,x_*)$$

In the GPR, the kernel function, which is also referred to as the covariance function, exhibits several differential forms. For example, the heterogeneous square exponential covariance function is as follows:

$$k(x,x') = \theta\exp(-\lambda||x-x'||^2)$$

where, θ and λ denote the parameters of the covariance function as seen in [4].

Having described the kernel function, it is of importance to estimate the hyper-parameters involved in the kernel from the training data. Most kernels have hyper-parameters that determine the prior of a Gaussian process, and according to Chen et al. in [2] the choice of prior significantly influences the performance for a given amount of data. The log likelihood function is typically maximized to optimize the hyper-parameters. If the hyper-parameters are chosen wisely, GPR can achieve a performance at least as good as that of a neural network, which is a prominent machine learning method, with fewer data. According to Manzhos et al. in [3] the significance lies in the fact that sparsity of data is very common in sufficiently high-dimensional spaces due to the curse of dimensionality, which refers to the problem where the possible data points increases so fast that the available data becomes sparse. Relatedly, Manzhos et al. [3] add that the computational cost of GPR increases rapidly with the number of training data.

To address this challenge, various hyper-parameter optimization methods have been developed, which will be explained in Chapter 3. These methods optimize the important hyper-parameters involved so that they work optimally, leading to improved performance and precise modeling. These hyper-parameters can be data-driven, determined based on the characteristics of the training data or model-driven, selected based on the desired behavior and complexity of the model (e.g. number of layers and neurons in a neural network, learning rate, activation function, regularization).

3 Hyper-parameter Optimization Methods

Hyper-parameter optimization plays a crucial role in model performance. This section provides an overview of various hyper-parameter optimization methods such as grid search, random search and Bayesian optimization. Some of these methods involve computational complexity, others are not efficient enough or have difficulty in tuning interdependent hyper-parameters. Therefore, more effective methods were also researched to optimize hyper-parameters such as particle swarm optimization, genetic algorithm, and differential algorithm as described by Kang et al. in [4].

3.1 Grid Search

Grid search is a commonly used and straightforward hyper-parameter optimization method that systematically builds and tests models for each combination of given hyper-parameters.

After evaluating all possible hyper-parameter combinations, grid search selects the combination that achieves the best performance based on the chosen error metric (e.g. Mean Squared Error, Mean Absolute Error).

According to Yu et al. in [5], the feature that makes grid search one of the most used methods is that it is possible to run it in parallel. That's because every trial runs individually without the influence of time sequence, which means that computational resources can be allocated in a highly flexible manner (Figure 1).

However, Yu et al. [5] also point out that one of the crucial drawbacks of the grid search is the computational cost, especially when dealing with a high number of hyper-parameters or when hyper-parameters have a large number of possible values. This is because the number of possible combinations of hyper-parameters grows exponentially with the number of values. Therefore, grid search can be considered as a viable approach when the number of hyper-parameters and their corresponding values remains relatively small.

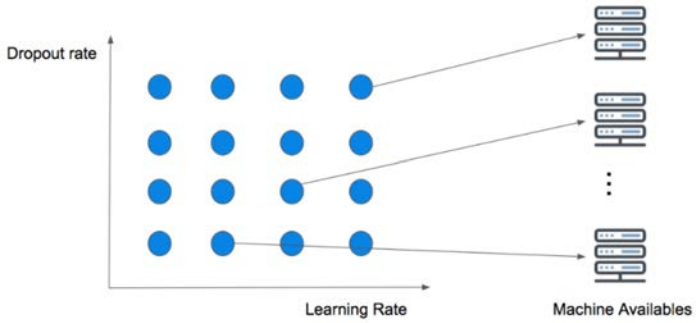


Fig. 1. Grid search on dropout rate and learning rate in a parallel concurrent execution [5]

3.2 Random Search

As the name suggests, random search tries out hyper-parameter values randomly from their defined ranges. Contrary to grid search which systematically evaluates all combinations of hyper-parameters, random search selects a random subspace of hyper-parameter values, evaluates the performance of each for predefined number of iterations or until a satisfying solution is found, and selects the best one.

Compared to grid search, random search can be considered less expensive since it tests randomly selected subsets of values instead of evaluating every possible combination. However, the key advantage of random search lies in its ability to explore a wide range of hyper-parameter values without limitations. It samples hyper-parameter values randomly from their defined ranges, while grid search follows a predefined pattern. This random sampling allows random search to potentially discover optimal configurations in areas that may lie between the grid points, which grid search might overlook (Figure 2).

Despite the fact that both grid search and random search are commonly used optimization methods due to their simplicity of implementation, they share an important drawback that has prompted researchers to seek more advanced hyper-parameter optimization methods. This drawback is their memorylessness, meaning they don't learn from their previous observations. Therefore, useful inter-dependencies between the hyper-parameters may be overlooked, leading to sub-optimal results.

To address this issue researchers have explored alternative methods such as Bayesian optimization, utilizing past observations and learning from them, leading to a more efficient exploration of the hyper-parameters.

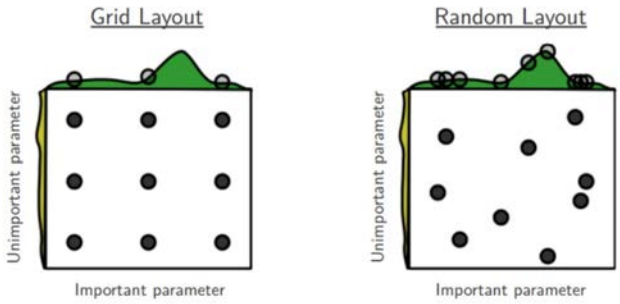


Fig. 2. Layout comparison between grid search and random search [5]

3.3 Bayesian Optimization

Definition 1. *Bayesian optimization is a sequential model-based method aimed at finding the global optimum with the minimum number of trials, while balancing exploration and exploitation to avoid trapping into the local optimum [5].*

In the exploitation phase, the model focuses on refining the current best-known solution. On the other hand, exploration involves delving into regions where predictive uncertainty is high and which may potentially yield better hyper-parameter combinations (Figure 3).

It is crucial to maintain a balance between exploitation and exploration for efficient hyper-parameter optimization. If the model relies too heavily on exploration, it may overlook optimal solutions, and ultimately resulting in sub-optimal outcomes.

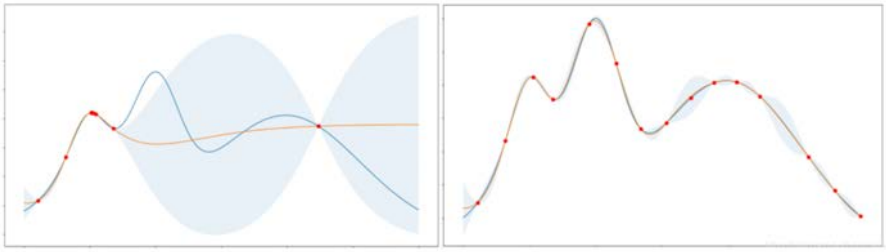


Fig. 3. Exploration-oriented (left) and exploitation-oriented Bayesian optimization (right); the shade indicates uncertainty [5].

Bayesian optimization has 2 key components (Figure 4). The first component is a surrogate model, in our case the Gaussian Process (GP) model which is used to approximate the underlying objective function, and the second component is an acquisition function which balances the trade-off between exploration and exploitation by deciding where to sample next.

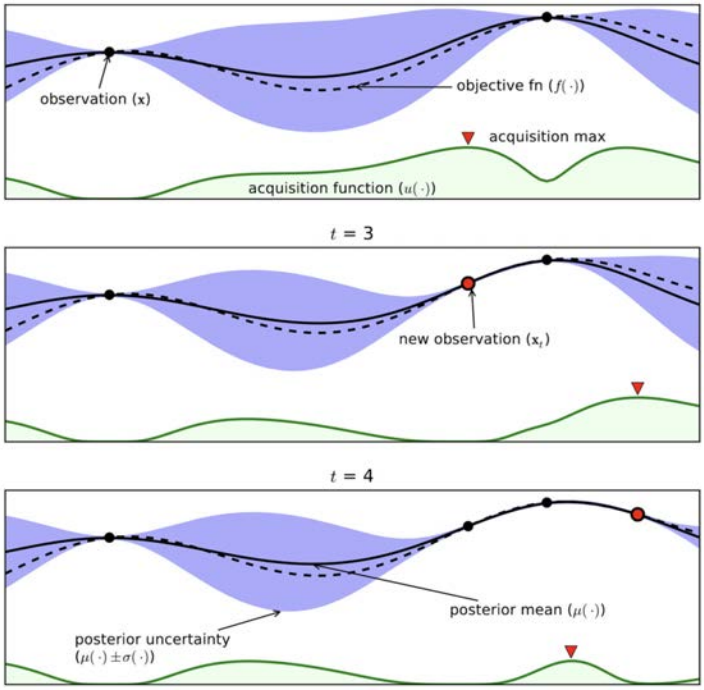


Fig. 4. Bayesian Optimization procedure over three iterations, with the mean and confidence intervals estimated with the surrogate model (GP) of the objective function. The acquisition functions in the lower shaded plots are showed [6].

Snoek et al. [8] describe how greedy acquisition procedures, strive to make significant progress in each subsequent function evaluation. These methods drive the optimization process by selecting points that are expected to yield valuable information, contributing to both improving the current solution and exploring uncharted regions of the search space. Relatedly, Yu et al. [5] emphasize that the Expected Improvement is the most prevalent selection for the acquisition function due to its strong performance and user-friendly nature:

$$E[I(\lambda)] = E[\max(0, f_{\min} - y)] \tag{2}$$

In this equation, $f_{\min}$ represents the current best observed function value, and y represents the predicted function value at the parameter value λ.

The improvement function is defined as $I(x, v, \theta) = (v - \tau)I$. I is larger than zero when an improvement exists; v is a normally distributed random variable; and τ is the target per the description in [5].

Before explaining the process in detail, the entire process is outlined in Algorithm 1, as depicted in the pseudo-code by Hamoudi et al. in [7]:

Algorithm 1. Pseudocode of Bayesian optimization with the GP surrogate model.

1: Input data D = $\{X, Y\}$
2: Initial small set of data randomly chosen $D_0 = \{X_0, Y_0\}$ from D
3: Compute a probabilistic Surrogate Function (Gaussian Process) g in such a way to find $Y_0 = g(X_0)$
4: for iterations $t = 1, 2, \ldots$ do
5: Select a new data set $D_t = \{X_t, Y_t\}$ from D by optimizing the Acquisition Function α (expected improvement per second plus. Equation (14)):

$$X_t = \arg\max_x \alpha(X; D_t)$$

4: Augment data set $D_t = D_{t-1} \bigcup \{(X_t, Y_t)\}$
5: Evaluate the Surrogate Function g for D_t to find $Y_t = g(X_t)$
7: end for

Fig. 5. Pseudo-code of Bayesian Optimization

In their study, Galuzzi et al. [6] describe the optimization process as starting with the initialization of a set of configurations X_0 and their corresponding function values: $D_0 = \{X_0, y_0\}$. At each iteration, the GP model will be updated using the Bayes rule, to obtain posterior distribution conditioned on the current training set, containing the past evaluated configurations and observations.

This iterative process helps in optimizing the model's objective function, which represents the performance metric that we are trying to maximize or minimize during the optimization process.

After evaluating a set of candidate points, a new data point with the highest expected improvement, in other words the point which maximizes the acquisition function will be selected as the next point to be evaluated.

The selected hyper-parameter configuration will be added to the prior data set and the GP is evaluated with the updated data set.

Until a convergence criteria is satisfied or a certain number of iterations have been completed, the procedure is continued. The final configuration of the hyper-parameters is the one that produces the best observed performance.

Snoek et al. [8] points out a key issue in Bayesian optimization regarding how important it is to design the initial assumptions effectively for efficient Bayesian

optimization. Many times, we don't have much information about the objective function, and getting data from it is also costly. This practical situation leads to two options: either we make strong assumptions without being completely sure they're accurate when we lack enough data, or we go with a less informative initial assumption. Additionally, it's often not clear how to find the right balance between exploring new possibilities and sticking with what we already know in the acquisition function.

Another critical problem of Bayesian optimization is that it progresses sequentially. Although there are some solutions to the parallelism problem, it can be very resource intensive as parallelizing Bayesian optimization requires running multiple instances of the Bayesian optimization process simultaneously, making an already potentially expensive optimization technique even less scalable.

Now that the 3 most popular hyper-parameter optimization algorithms have been introduced, we will now compare the grid search, random search and Bayesian optimization algorithms in table 1.

Table 1. Comparison of Hyper-parameter Optimization Algorithms

Algorithm	**Positives**	**Negatives**	**Optimal To Use**
Grid Search	– Systematic search guarantees full coverage – Simple to implement – Straightforward to understand – Possible to run parallel	– Computationally expensive and impractical for high-dimensional spaces (curse of dimensionality) – Memorylessness, it doesn't learn from it's previous observations. Therefore, useful inter-dependencies between the hyper-parameters may be overlooked, leading to sub-optimal results	– When the number of hyper-parameters and their corresponding values is relatively small and the resources are sufficient to explore all combinations thoroughly – When the objective function doesn't have useful inter-dependencies between hyper-parameters or when the relationship between hyper-parameters and the objective function is well-understood
Random Search	– Ability to explore a wide range of hyper-parameter values without limitations – Can handle high-dimensional spaces – Less computationally expensive than grid search – Possible to run parallel	– May still require many samples to find optimal hyper-parameters – Less systematic than grid search – No guarantee that random search will converge to optimal or near-optimal hyper-parameters. – Memoryless	– When the hyper-parameter search space is large and exhaustive search is computationally impracticable – When computational resources are limited and you have a large hyper-parameter space
Bayesian Optimization	– Efficient with expensive objective functions – Considers previous observations and inter-dependencies between hyper-parameters – Fewer evaluations required – Better suited for high-dimensional spaces	– Design of prior is critical to efficiency – In many cases, little is known about the objective function, and, it is expensive to sample from – Unclear trade-off between exploration and exploitation in acquisition function – Progresses sequentially, potentially slower than parallel methods – Parallelization can be resource-intensive and may reduce scalability	– When the objective function is expensive to evaluate – When the objective function is a black-box and its underlying behavior is not well-known – When computational resources are available for parallel execution and the search space is high-dimensional

4 Dealing with the Indefinite Covariance Matrix

Another problem that emerges during the training phase of the GPR is the covariance matrix becoming indefinite, which means that the matrix loses it's positive definiteness property. One possible cause of the indefinite covariance matrix can be redundant variables in the data, as their correlations approach zero. In that case the covariance matrix can have eigenvalues close to zero or even negative, which makes the matrix indefinite. An indefinite matrix leads to computational problems since it cannot be inverted, making it difficult to calculate uncertainty and predictive mean values:

$$\sigma_*(x_*, x_1, \ldots, x_M, y) = K(x_*, X)K(X, X)^{-1}K(X, x_*)$$

$$\mu_*(x_*, x_1, \ldots, x_M, y) = K(x_*, X)K(X, X)^{-1}y$$

To tackle this issue, numerous approaches have been proposed and widely adopted in practical applications. In this context, we will delve into a significant technique known as regularization, which plays a crucial role in addressing the problem at hand.

Regularization

Regularization is a general concept to improve the performance and prevent over-fitting. To prevent complex or extreme solutions where the training data fits perfectly, but do not generalize well to new, unseen data, additional constraints or penalties are added to the model's objective function. As two common regularisation strategies, Mohammadi et al. [9] present the pseudo-inverse (PI) and nugget methods. In this section an introduction of these methods will be presented based on the paper of Mohammadi et al.

A PI method is used to obtain an approximate inverse of a matrix that has numerical stability issues. PI's are generalizations of the matrix inverse for matrices with non-squares or inverses that might not exist. It allows you to compute an optimized inverse for matrices that don't have a true inverse. Moore-Penrose PI is the most well-known PI. Given a matrix A, the Moore-Penrose PI A^+ satisfies the following conditions as described by Barata et al. in [10]:

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. $(AA^+)^T = AA^+$
4. $(A^+A)^T = A^+A$

The nugget method on the other hand uses a small positive constant (σ^2, the nugget) that has been tacked onto the covariance matrix's K diagonal. The adjusted covariance matrix K_{adjusted} is given by:

$$K_{\text{adjusted}} = K + \sigma^2 I$$

By successfully adding some noise to the data, this can assist solving problems caused by closely spaced or perfectly aligned data points. It prevents the model from fitting the noise in the training data too closely. An essential step in using the nugget approach is determining the noise variance. Cross-validation or maximum likelihood are frequently used to calculate this value.

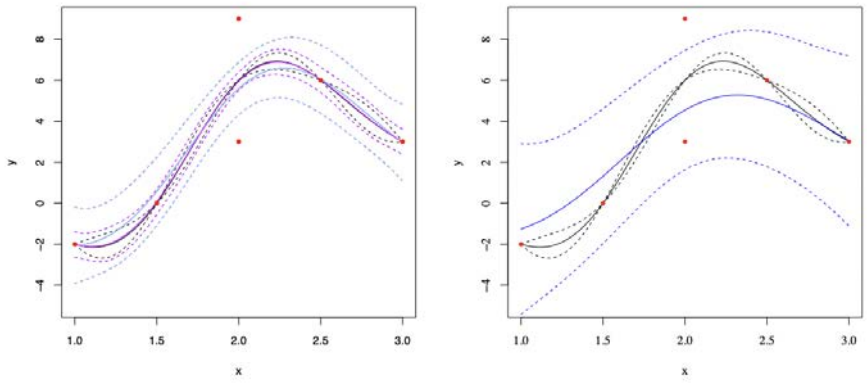


Fig. 6. PI regularization is plotted in black. Left: the nugget value is 1 (cyan) and 0.1 (magenta). For nugget values smaller than 0.01, Pseudo-inverse and nugget regularizations cannot be visually distinguished. Right: the nugget estimated by maximum likelihood is 7.07 (blue)[9].

The models produce predictions for PI regularisation that precisely span the given data points and also follow the average trend at redundant points. It's like perfectly joining the dots. The approach ensures that predictions vary gradually and are not too unpredictable. The degree of uncertainty surrounding predictions decreases when we use PI regularisation.

Nugget regularization is a bit different. It makes predictions that don't exactly match the data points and don't average out smoothly either. It adds more randomness to the predictions. When using nugget regularization, there's still some uncertainty left around the predictions.

The method we choose relies on the issue we're trying to solve. It is preferable to use PI regularisation or a little nugget if our GP model fits the data pretty well. A little nugget or PI will work if there is a minor discrepancy between

the model and the data. However, if the difference is greater, we must choose the approach depending on what is crucial to solving the problem. We might favour PI if we want the predictions to easily follow the trend. We might select the nugget if we wish to take uncertainty into account more when making our predictions.

5 Conclusion

After examining the training phase of GPR, we have reached the conclusion that the selection of proper hyper-parameters plays a pivotal role in achieving accurate predictions. We have provided a comprehensive review of techniques such as grid search, random search, and Bayesian optimization, analyzing their mechanisms, strengths, and limitations. Our analysis has highlighted the importance of adaptive strategies that learn from past evaluations, ultimately leading to improved hyper-parameter optimization.

In the context of addressing the challenge posed by indefinite covariance matrices, we have observed the significance of regularization within GPR, which also comes with its limitations. By employing regularization techniques, such as the utilization of pseudo-inverse and nugget methods, we have explored effective strategies aimed at mitigating the complexities arising during covariance matrix inversion. This endeavor not only empowers the model but also enhances its capability to generate dependable predictions and precise estimates of uncertainty.

The methods and insights presented here pave the way for more efficient and effective implementations of GPR, enabling its broader applicability across various domains.

References

1. Wang, Jie. *An Intuitive Tutorial to Gaussian Processes Regression*. arXiv preprint arXiv:2107.05174, 2021. [Online]
2. Chen, Zexun and Wang, Bo. *How priors of initial hyperparameters affect Gaussian process regression models*. arXiv:1605.07906 [stat.ML], 2023. [Online]
3. Sergei Manzhos and Manabu Ihara. *On the optimization of hyperparameters in Gaussian process regression with the help of low-order high-dimensional model representation*. arXiv preprint arXiv:2112.01374, 2021. [Online]
4. L. Kang, R.-S. Chen, N. Xiong, Y.-C. Chen, Y.-X. Hu, and C.-M. Chen, "Selecting Hyper-Parameters of Gaussian Process Regression Based on Non-Inertial Particle Swarm Optimization in Internet of Things," *IEEE Access*, vol. 7, pp. 59504-59513, 2019. DOI: 10.1109/ACCESS.2019.2913757. [Online]
5. T. Yu and H. Zhu, *Hyper-Parameter Optimization: A Review of Algorithms and Applications*, arXiv preprint arXiv:2003.05689, cs.LG. [Online]
6. Galuzzi, B.G.; Giordani, I.; Candelieri, A.; Perego, R.; Archetti, F. *Hyperparameter optimization for recommender systems through Bayesian optimization. Computational Management Science*, 17(4), 495–515, 2020. [Online]
7. Hamoudi, Y.; Amimeur, H.; Aouzellag, D.; Abdolrasol, M.G.M.; Ustun, T.S. *Hyperparameter Bayesian Optimization of Gaussian Process Regression Applied in Speed-Sensorless Predictive Torque Control of an Autonomous Wind Energy Conversion System. Energies*, 16(12), 4738, 2023. [Online]
8. Jasper Snoek, Hugo Larochelle, and Ryan P. Adams, *Practical Bayesian Optimization of Machine Learning Algorithms*, 2012. [Online]
9. Hossein Mohammadi, Rodolphe Le Riche, Nicolas Durrande, Eric Touboul, and Xavier Bay, *An analytic comparison of regularization methods for Gaussian Processes*, 2017. [Online]
10. J.C.A. Barata and M.S. Hussein, *The Moore–Penrose Pseudoinverse: A Tutorial Review of the Theory*, *Brazilian Journal of Physics*, vol. 42, pp. 146–165, 2012. [Online]

Gaussian Process Regression: Different Likelihoods in Various Applications

Annemarie Schaub

Karlsruhe Institute of Technology
Kaiserstraße 12, 76131 Karlsruhe, Germany
https://www.kit.edu

Abstract. Gaussian Process Regression (GPR) is a versatile framework used in regression analysis, allowing for flexible modeling and probabilistic predictions. One intriguing aspect of GPR is its incorporation of different likelihood functions, which govern the noise properties of observed data. This paper explores the use of different likelihoods in two specific scenarios: noise modeling and classification, showcasing the adaptability of GPR to various applications.

1 Introduction

Regression analysis is a statistical method used to examine the relationship between a dependent variable and one or more independent variables. It aims to model and predict the value of the dependent variable based on the independent variables. The process involves fitting a regression equation to the data, which determines the slope and intercept of the line that best represents the relationship. The analysis helps quantify the strength and direction of the relationship, allowing for predictions and inference.

Unlike traditional regression methods that assume a specific functional form for the relationship between variables, Gaussian Process Regression (GPR) is a non-parametric approach that allows for more flexible modeling. It assumes a Gaussian distribution over functions and provides a posterior distribution of possible functions given the data. This distribution captures uncertainty and allows for probabilistic predictions, providing not only point estimates but also confidence intervals. GPR can handle noisy and sparse data, adapt to various types of relationships, and is capable of capturing complex patterns. It is widely used in machine learning, particularly in areas such as spatial modeling, time series analysis, and optimization (e.g., see [15]).

Gaussian distributions assume symmetry and have a bell-shaped curve. In real-life scenarios, many variables exhibit skewness, where the distribution is asymmetrical. Similarly, Gaussian distributions have finite kurtosis, meaning they have lighter tails compared to some real-life data that can have heavy tails or outliers. Consequently, Gaussians are sensitive to outliers, meaning that a single extreme value can significantly impact the mean and standard deviation. Moreover, they are continuous and assume that the underlying variable can

take any value within a range. However, many real-life phenomena are discrete, such as the number of occurrences, counts, or categorical data, which cannot be accurately represented by a Gaussian distribution.

Concrete examples for non-Gaussian distributions are the volume of internet traffic and the occurrence of natural disasters. The volume of internet traffic often exhibits a non-Gaussian distribution because it experiences bursts of high activity and periods of low activity, resulting in a distribution that is skewed and exhibits heavy tails [1]. The occurrence of natural disasters, such as earthquakes or hurricanes, tends to follow a non-Gaussian distribution, because, while most days are relatively calm, extreme events with large magnitudes occur less frequently but can have a significant impact [14].

2 Core principles and equations

Over the course of this chapter, we lay the groundwork for understanding the fundamental principles and equations that underpin GPR. By exploring the core concepts and mathematical formulations, we aim to provide a comprehensive overview of the key components involved in GPR, including the Gaussian process prior, the mean and covariance functions and the predictive equations.

2.1 Notations

The notations are mainly adopted from Rasmussen et al. in [15].

Symbol	Meaning
$\propto$	proportional to
$\sim$	distributed according to
$\simeq$	asymptotically equal to
$f(x)$ or $\mathbf{f}$	Gaussian process
$\mathbf{f}_*$	Gaussian process (posterior) prediction
$\hat{\mathbf{f}}_*$	Gaussian process posterior mean
$\mathcal{GP}$	Gaussian process
X	training inputs
X_*	test inputs

2.2 Gaussian Processes

Definition 1. *A* Gaussian process *is a stochastic process, or rather a collection of random variables, any finite number of which have a joint Gaussian distribution* [15].

As seen in the book by Rasmussen [15], we will write a Gaussian process as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{1}$$

where the mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ of $f(\mathbf{x})$ are defined as

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{2}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{3}$$

with the expectation value $\mathbb{E}[\mathbf{x}]$. The mean function specifies the average behavior of the functions generated by the process, while the covariance function captures the pairwise relationships between the function values at different input points. The covariance function, also known as the kernel function, determines the smoothness of the functions generated by the Gaussian process.

2.3 Prediction

We aim to predict function values from n given data points, also called training points $X = \{(\mathbf{x}_i, \mathbf{y}_i) | i = 1, ..., n\}$ with $\mathbf{y}_i = f(\mathbf{x_i}) + \varepsilon$ being the observed output value at $\mathbf{x}_i$. Therefore, ε allows us to compensate for noisy observations.

Assuming the noise to be independent and identically distributed Gaussian with variance σ_n^2, the covariance, also referred to as the prior on the noisy observation, becomes

$$\text{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 I \tag{4}$$

where, as seen in [15], $K(X, X)$ is the $n \times n$ matrix of the covariances determined at all pairs of training points in X.

The predictive distribution is a Gaussian $\mathbf{f}_* | X_*, X, \mathbf{y} \sim N(\hat{\mathbf{f}}_*(X_*),$ $\text{cov}(X_*))$ with parameters

$$\hat{\mathbf{f}}_*(X_*) = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y} \quad \text{and} \tag{5}$$
$$\text{cov}(X_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*). \tag{6}$$

By performing an integration over the latent function values, the marginal likelihood $p(\mathbf{y}|X)$ is obtained as the result of combining the prior and likelihood:

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f}. \tag{7}$$

It is a measure of how well the chosen hyperparameters fit the observed data and quantifies the trade-off between model complexity and data fit by yielding a scalar value that represents the probability of the observed data given the model and hyperparameters.

2.4 Latent variables

Latent variables are unobserved variables that are inferred from the observed data to model the hidden patterns or underlying factors driving the data. They are used on regression models with non-Gaussian noise and classification problems. The Gaussian process is therefore defined for the latent variables, which then define a distribution for the target [13]. An application of this approach will be presented in chapter 4.2.

2.5 Markov chain Monte Carlo methods

Definition 2. *A* Markov chain *is a sequence of configurations, where each configuration depends only on the previous configuration, and the future states are independent of the past states given the present state* [2].

Markov chain Monte Carlo (MCMC) methods encompass a family of algorithms used to generate samples from a probability distribution by constructing a Markov chain with a desired stationary distribution [2]. Therefore, the space of possible samples from a target probability distribution is explored. The chain transitions from one state to another is based on a set of transition probabilities.

Gibbs sampling Gibbs sampling is a MCMC method used to generate samples from a joint probability distribution when the conditional distributions of the variables are known. It iteratively updates each variable by sampling from its conditional distribution given the current values of the other variables, effectively exploring the entire joint distribution. This process continues for a sufficient number of iterations until convergence is achieved and samples from the desired distribution are obtained [9].

Algorithm 1 Gibbs sampling for two variables x, y

Ensure: (x^k, y^k) are already obtained
  while true **do**
    generate x^{k+1} following $p(x^{k+1}|y^k)$.
    generate y^{k+1} following $p(y^{k+1}|x^{k+1})$.
  end while

The key aspect of this algorithm is that, instead of modifying x^k slightly to obtain x^{k+1} while keeping y^k fixed, x^{k+1} is generated independently of x^k, without any reference to its previous value.

3 Non-Gaussian noise

In this chapter, we delve into the intriguing realm of non-Gaussian noise, where the assumptions of traditional Gaussian noise models no longer hold. Specifically, we explore the challenges posed by input-dependent noise and the presence of outliers in regression problems.

3.1 Input-dependent noise

Goldberg et al. [8] describe an approach for regression problems with noise on the output and, different from standard approach looked at in chapter 2, the variance of the noise being input-dependent, so that we have to predict two functions: The noise level $r(\mathbf{x})$ and the function values themselves.

Model According to the book by Goldberg et al. [8], in this scenario, it is convenient to combine two Gaussian processes for making predictions. The first, so called y-process

$$f_y(\mathbf{x}) \sim \mathcal{GP}(m_y(\mathbf{x}), k_y(\mathbf{x}, \mathbf{x}')) \tag{8}$$

is to predict the function values as usual. Due to the noise not being negative, a Gaussian process prior is placed over the log noise rate $\mathbf{z}$ which leads to $r = (\exp z_1, ..., \exp z_n)$ and results in the second, so called z-process

$$f_z(\mathbf{x}) \sim \mathcal{GP}(m_z(\mathbf{x}), k_z(\mathbf{x}, \mathbf{x}')) \tag{9}$$

to predict the noise variance.

The n training points that we have got now are $X = \{(\mathbf{x_i}, \mathbf{y_i})|i = 1, ..., n\}$ with $\mathbf{y_i} = f_y(\mathbf{x_i}) + r(\mathbf{x_i}) = f_y(\mathbf{x_i}) + \exp f_z(\mathbf{x_i})$.

Noise sampling Goldberg et al. sample from the distribution $p(\mathbf{y}, \mathbf{z}|X)$ and afterwards ignore the $\mathbf{y}$-values to get $p(\mathbf{z}|X)$. To achieve this, Gibbs sampling (as introduced in section 2.5) is used. The distinguishing characteristic of this approach is alternating sampling from $p(\mathbf{y}|\mathbf{z}, X)$ and $p(\mathbf{z}|\mathbf{y}, X)$. This corresponds to alternating the "fitting" of the curve (y-process) with "fitting" the noise level (z-process).

Predictions As described by Goldberg et al. [8], the predictive distribution for the output $\mathbf{f_{y*}}$ at point $\mathbf{x}_*$ is

$$p(\mathbf{f_{y*}}|X) = \int p(\mathbf{f_{y*}}|X, r(\mathbf{z}))p(\mathbf{z}|X)d\mathbf{z} \tag{10}$$

The integral seen above in equation (10) cannot be solved analytically anymore, owing to $p(\mathbf{z}|\mathbf{y})$ not to follow a Gaussian Distribution. Therefore, Goldberg et al. use Monte Carlo approximation, which is a sampling technique that involves random sampling of the input space and calculating the average of function evaluations at these sampled points, making it useful for estimating integrals and other quantities [5].

The result of the Monte Carlo approximation, accomplished by Goldberg et al. in [8] can be seen in equation (11).

$$p(\mathbf{f_{y*}}|X) \simeq \frac{1}{k} \sum_j p(\mathbf{f_{y*}}|X, r(z_j)) \tag{11}$$

To subsequently approximate $p(\mathbf{f}_{\mathbf{y}*}|X, r(z_j)$, Goldberg et al. take 10 samples of $p(\mathbf{f}_{\mathbf{z}*}, z_j)$, resulting in a mixture of 10 Gaussians representing the approximating distribution. Finally, to approximate $p(\mathbf{f}_{\mathbf{y}*}, X)$, they computed the average over the k samples (z_1, ..., z_k).

3.2 Student's t-distribution for a regression problem with outliers

Neal [13] made a similar approach to the one discussed in the previous chapter. In an experiment described in [13], a Gaussian process model with non-Gaussian noise was applied to a regression problem with outliers.

Setup Sampled from a Gaussian distribution, the input variable, $\mathbf{x}$, was associated with corresponding target values that followed a distribution determined by a specific mean function. The majority of cases had a Gaussian distribution around this mean with a standard deviation of 0.1. However, with a 0.05 probability, outliers were introduced with a larger standard deviation of 1.0.

Model To model this data, a Gaussian process was employed. Assuming the noise originates from a t-distribution with $\nu = 4$ degrees of freedom, the model predicts the expected value of the target (see equation (12)).

$$f(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}} \tag{12}$$

The t-distribution, characterized by its degrees of freedom parameter, exhibits heavier tails compared to the Gaussian distribution. This means that extreme values are more likely to occur in the t-distribution, leading to a wider spread of data. As can be observed in figure 1, the Gaussian distribution has lighter tails compared to the t-distribution, implying that extreme values are less likely to occur. By comparing these two distributions, it becomes evident that the t-distribution provides a more flexible modeling framework, accommodating data with potential outliers or deviations from normality. The Gaussian distribution, on the other hand, represents a more standard and narrow range of possibilities, assuming a stricter adherence to normality.

While this noise distribution, as stated in the paper by Neal [13], did not precisely match the actual noise distribution, the heavy tails of the t-distribution allowed the model to capture the data without being overly influenced by outliers. To draw a comparison, Neal [13] additionally performed modeling assuming the presence of Gaussian noise.

Noise sampling Similar to the noise sampling for the input-dependent noise, explored in the previous section, it is done by Neal [13] by using MCMC involving hybrid Monte Carlo updates for the hyperparameters, along with updates for the individual noise variances.

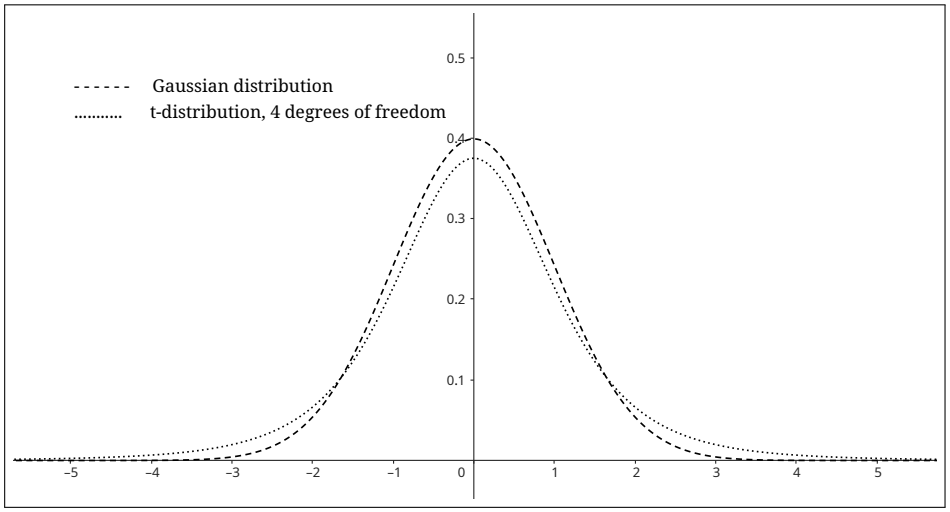


Fig. 1. The plot shows a comparison between a t-distribution with four degrees of freedom and a Gaussian distribution with $\mu = 0, \sigma^2 = 1$.

According to Neal in [13], it can be observed that the predictions generated by the model incorporating t-distributed noise appear to be more plausible and closer to the true function compared to the model incorporating Gaussian noise.

4 Classification

In the following sections, we explore the application of GPR to two distinct classification scenarios: ordinal regression and three-way classification. Unlike traditional classification methods, we extend GPR by incorporating non-Gaussian likelihood functions to capture the inherent characteristics of the data.

4.1 Ordinal regression

Ordinal regression, also known as ranking learning, is a machine learning approach that focuses on predicting the relative ordering or ranking of data instances rather than their absolute values or class labels.

Chu and Ghahramani [3] use Gaussian processes for the ordinal regression problem, which will be delved into henceforth.

Model In this setting, the data points $X = \{(\mathbf{x}_i, \mathbf{y}_i)|i = 1, ..., n\}$ consist of the $\mathbf{x}_i$ being any real number, but the $\mathbf{y}_i$ coming from $\mathcal{Y} = \{1, 2, ..., r\}$, the set of r ordered categories. Chu's and Ghahramanis's main idea in [3] is to postulate an unobservable latent function $l(\mathbf{x_i})$ with a Gaussian prior and then deriving the

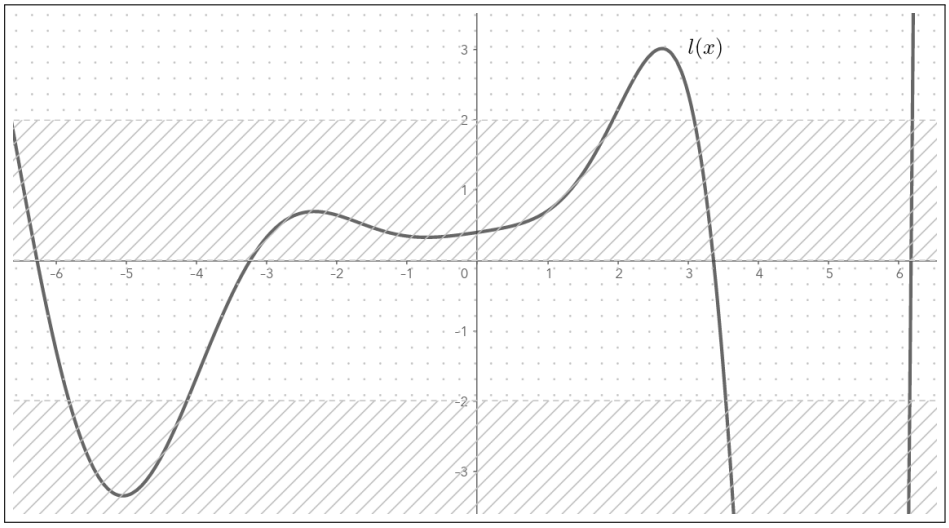


Fig. 2. The plot shows an illustrative example of a latent function $l(\mathbf{x})$ along with the corresponding categorization into four distinct intervals, each representing a different category. The latent function provides a continuous representation of the underlying data, while the categorized intervals offer a discrete classification scheme with four distinct categories.

$\mathbf{y}_i$-values from $l(\mathbf{x_i})$ by representing the categories $\mathcal{Y}$ as intervals along the real number line (see fig. 2).

The likelihood function $p(\mathbf{y}_i|l(\mathbf{x}_i)$ is further defined by Chu and Gharamani [3] as

$$p(\mathbf{y}_i|l(\mathbf{x}_i) = \begin{cases} 1 & \text{if } b_{\mathbf{y}_i - 1} < l(\mathbf{x_i}) \leq b_{\mathbf{y}_i} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

with b_i being the interval boundaries, where $b_0 = -\infty$ and $b_r = +\infty$.

Model adaption is subsequently achieved by integrating the hyperparameters θ over the θ-space. This can be done using MCMC methods as described earlier, but Chu and Ghahramani in [3] state, that this might be expensive to use in practice. Therefore, they explore two distinct approaches: MAP Approach with Laplace approximation similar to what Williams and Barber did in [17] and Expectation propagation with variational methods, similar to Minkas approach in [12].

MAP Approach with Laplace approximation The Maximum A Posteriori (MAP) approach with Laplace approximation is a method that approximates the posterior distribution by finding the mode using the Laplace approximation technique, providing an estimate of the most probable parameters in Bayesian inference [7].

Expectation propagation with variational methods Another technique to approximate the posterior distribution in Bayesian inference is expectation propagation (EP) with variational methods. It's done by iteratively updating and refining a variational distribution to better match the true posterior distribution [12].

4.2 A three-way classification problem

A paper published by Neal in 1997 [13] describes an example of a three-way classification problem and the process used to model and predict the target classes.

Model Because there are three classes, $k = 3$ latent values $y_{0,i}, y_{1,i}, y_{2,i}$, one for each case, are introduced which define the following class probabilities:

$$p(t_i = k) = \frac{\exp(-y_{k,i})}{\sum_{m=0}^{2} \exp(-y_{m,i})} \tag{14}$$

In the next step, those k latent values are then given independent Gaussian priors, such that

$$\mathrm{cov}(y_i, y_j) = \eta^2 \exp(-\sum_u \rho_u^2 (x_{u,i} - x_{u,j})^2 + \delta_{ij} J^2. \tag{15}$$

According to Neal et al. in [13], a small amount of jitter, J, has to be added for computational reasons.

Data generation Data points, (x_i, t_i), were generated by randomly choosing values from a uniform distribution over the interval [0, 1]. The target, t_i, was determined based on certain conditions involving the Euclidean distance and linear combination of the input values.

Neal generated a total of 1000 cases, out of which 400 were used as training data and 600 were used for testing the predictive performance.

Sampling A form of hybrid Monte Carlo was used for sampling and latent values were resampled between each update of the hyperparameters. Gibbs sampling scans were performed to update the latent values associated with training cases.

The convergence of the Markov chain simulation was imposed by monitoring the values of the hyperparameters and the latent values during the course of the simulation.

Prediction Predictive probabilities for test cases were obtained by averaging the probabilities from iterations after reaching equilibrium. The predictive mean and variance were calculated using the latent values from training cases saved for each iteration. A Monte Carlo estimate was used to generate predictive probabilities for each class, and the class with the highest probability was selected as the prediction.

Result According to the Neal et al. [13], the classification error rate on the 600 test cases was 0.13, which the paper refers to as comparable to an analogous neural network model. However, a comprehensive comparison with other classification methods was not conducted in this example.

The time required for the procedure ranges depending on the number of training cases. With 100 training cases, the simulation time was approximately 16 minutes, and the prediction time for the 600 test cases was under a minute. Using only those 100 training cases, a classification error rate of 0.17 was calculated by Neal et al. [13].

5 Further applications

Throughout this chapter, we briefly look into additional domains and problem scenarios where GPR with non-Gaussian likelihoods finds valuable applications. Building upon the foundational knowledge, this chapter digs into novel areas where the combination of GPR and non-Gaussian likelihoods unlocks new possibilities and addresses specific challenges. We delve into diverse applications such as residual contamination from nuclear weapons, oil flow, volatility forecasting, count data analysis and survival analysis.

5.1 Residual contamination from nuclear weapons

Diggle et al. [4] extended GPR to address settings where the stochastic variation in the data is encountered to be non-Gaussian. Specifically, they applied this extension to modeling geographically measured count data through a Poisson likelihood model with a rate that varies across space [15]. Their approach involved a Gaussian prior over the log Poisson rate.

The application of this methodology was focused on studying residual contamination from nuclear weapons. By employing the Poisson likelihood with a spatially varying rate and utilizing the Gaussian prior, Diggle et al. aimed to analyze the count data obtained from geographically diverse locations and capture the spatial variation in contamination intensity.

5.2 Oil flow

A latent variable model is applied to multi-phase oil flow data consisting of 1000 observations belonging to three known classes by Titsias et al. in [16].

The model with 10 latent dimensions using the ARD SE (Automatic Relevance Determination Squared Exponential) kernel is employed. The algorithm automatically switches off 7 out of 10 latent dimensions, resulting in a high-quality two-dimensional visualization of the data. In contrast, the standard sparse model assuming only 2 latent dimensions provides a less informative visualization. The model achieves a nearest neighbor error of 3 out of 1000 data points, whereas the standard model yields 26 errors, demonstrating the superiority of the approach in improving classification accuracy.

5.3 Volatility forecasting

Heteroscedastic GPR, similar to section 3.1, is according to Lázaro-Gredilla et al. in [11] a natural choice for modeling and predicting volatility, particularly in financial time series. Volatility, representing the standard deviation of a return series, can be estimated using Heteroscedastic GPR by considering the log-return series obtained from price series. By treating the return series as a zero-mean noise-only process and assuming discrete time intervals, such as days, Heteroscedastic GPR can estimate historical volatility and make forecasts in datasets where the noise level (volatility) varies over time and is further described by Lázaro-Gredilla et al. in [11].

5.4 Count data analysis

In their article, Jia et al. [10] propose a method for modeling stationary count time series using Gaussian transformations. Their approach employs a latent Gaussian process and distributional transformation to create stationary series with adaptable correlation features. These features can conform to various prespecified marginal distributions, such as Poisson, generalized Poisson, negative binomial, and binomial structures. Likelihood estimation is conducted using particle filtering and sequential Monte Carlo methods. This research presents innovative techniques for modeling stationary count time series with flexible correlation properties and diverse marginal distributions.

5.5 Survival analysis

In a study by Fernández et al. [6]., a semi-parametric Bayesian model for survival analysis is introduced. The model incorporates a parametric baseline hazard and utilizes a Gaussian process to capture nonparametric variations around it, as well as the influence of covariates. Unlike many other methods, this framework avoids imposing unnecessary constraints on the hazard rate or survival function. Additionally, the model accommodates left, right, and interval censoring mechanisms commonly encountered in survival analysis. The authors propose an MCMC algorithm for inference and a computation-efficient approximation scheme based on random Fourier features.

6 Strengths and Limitations

Within this chapter, we delve into the advantages and challenges associated with using non-Gaussian likelihoods. By expanding beyond the traditional Gaussian assumptions, non-Gaussian likelihoods allow for more flexible modeling and improved representation of complex data characteristics. However, their adoption also brings potential limitations and considerations to GPR that need to be carefully addressed.

On the one hand, GPR provides a measure of uncertainty for its predictions. This can be beneficial in classification tasks as it allows for quantifying the confidence or uncertainty associated with each prediction. Uncertainty estimates can be valuable in decision-making, risk assessment, or when dealing with imbalanced or ambiguous datasets.

Furthermore, GPR can effectively handle classification tasks even with a small training set. Unlike some other classification algorithms that require a large number of labeled samples, GPR can leverage prior knowledge and make use of a limited number of training examples.

GPR also provides insights into the decision-making process by analyzing the learned kernel function. This interpretability can be useful in understanding the underlying patterns or relationships in the data, which can aid in model validation and domain knowledge integration.

On the other hand, GPR can be computationally expensive, especially when dealing with large datasets. The training and inference time can be significant, particularly as the number of training samples increases. This can limit its scalability in certain applications where real-time or high-speed classification is required. It also relies on tuning hyperparameters such as the kernel length scale or noise level. The performance of GPR is sensitive to these hyperparameters, and selecting optimal values can be challenging, requiring expertise and experimentation.

Overall, GPR for classification tasks offers valuable advantages such as uncertainty estimation and flexibility in kernel choice. However, it also poses challenges related to computational complexity, hyperparameter tuning, and handling imbalanced datasets. Assessing the trade-offs and considering the specific requirements of the classification problem at hand is important when deciding whether to use GPR as the classification method.

7 Executive Summary

In conclusion, GPR is a versatile and powerful method that finds applications in diverse fields. We discovered its strength in the flexibility to accommodate different likelihood functions, enabling adaptable modeling to match specific data characteristics.

To begin the discussion of selecting an appropriate likelihood, we saw in chapter 3 that GPR can be tailored for regression problems. Then chapter 4 demonstrates the use of GPR for classification problems. Finally, to round up

the discussion and as outlook for potential future research, chapter 5 enumerates various further applications of GPR with non-Gaussian likelihoods.

The choice of likelihood function should be chosen with care, with consideration of the properties of the data set at hand. Each likelihood makes specific assumptions and carries implications, and an inappropriate choice can lead to suboptimal outcomes or inaccurate modeling.

Understanding the advantages and disadvantages of different likelihoods and their suitability for the task at hand empowers practitioners to fully exploit the potential of GPR. By leveraging this knowledge, meaning insights can be extracted, accurate predictions made, and real-world problems effectively addressed. In-depth researches and experimentations with different likelihoods within the GPR framework will further enhance its applicability and effectiveness across diverse domains.

References

1. Alasmar, M., Clegg, R., Zakhleniuk, N., Parisis, G.: Internet traffic volumes are not gaussian—they are log-normal: An 18-year longitudinal study with implications for modelling and prediction. IEEE/ACM Trans. Netw. **29**(3), 1266–1279 (feb 2021), https://doi.org/10.1109/TNET.2021.3059542
2. Brooks, S., Gelman, A., Jones, G., Meng, X.: Handbook of Markov Chain Monte Carlo. Chapman & Hall/CRC Handbooks of Modern Statistical Methods, CRC Press (2011)
3. Chu, W., Ghahramani, Z.: Gaussian processes for ordinal regression. Journal of Machine Learning Research **6**, 1019–1041 (07 2005)
4. Diggle, P.J., Tawn, J.A., Moyeed, R.A.: Model-based geostatistics. Journal of the Royal Statistical Society. Series C (Applied Statistics) **47**(3), 299–350 (1998), http://www.jstor.org/stable/2986101
5. Evans, M., Swartz, T.: Approximating Integrals via Monte Carlo and Deterministic Methods. Oxford Statistical Science Series, OUP Oxford (2000), https://books.google.de/books?id=GMHKfx84L4MC
6. Fernández, T., Rivera, N., Teh, Y.W.: Gaussian processes for survival analysis (2016)
7. Gauvain, J.L., Lee, C.H.: Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. IEEE Transactions on Speech and Audio Processing **2**(2), 291–298 (1994). https://doi.org/10.1109/89.279278
8. Goldberg, P., Williams, C., Bishop, C.: Regression with input-dependent noise: A gaussian process treatment. In: Jordan, M., Kearns, M., Solla, S. (eds.) Advances in Neural Information Processing Systems. vol. 10. MIT Press (1997), https://proceedings.neurips.cc/paper_files/paper/1997/file/afe434653a898da20044041262b3ac74-Paper.pdf
9. Hanada, M., Matsuura, S.: MCMC from Scratch: A Practical Introduction to Markov Chain Monte Carlo. Springer Nature Singapore (2022), https://link.springer.com/book/10.1007/978-981-19-2715-7
10. Jia, Y., Kechagias, S., Livsey, J., Lund, R., Pipiras, V.: Latent gaussian count time series. Journal of the American Statistical Association **118**(541), 596–606 (2023)
11. Lázaro-Gredilla, M., Titsias, M.K.: Variational heteroscedastic gaussian process regression. In: ICML. pp. 841–848 (2011)
12. Minka, T.P.: Expectation propagation for approximate bayesian inference (2013)
13. Neal, R.M.: Monte carlo implementation of gaussian process models for bayesian regression and classification (1997)
14. Poirion, F., Zentner, I.: Non-gaussian non-stationary models for natural hazard modeling. Applied Mathematical Modelling **37**, 5938–5950 (04 2013). https://doi.org/10.1016/j.apm.2012.11.021
15. Rasmussen, C.E., Williams, C.K.: Gaussian processes for machine learning. The MIT Press (2006)
16. Titsias, M., Lawrence, N.D.: Bayesian gaussian process latent variable model. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9, pp. 844–851. PMLR, Chia Laguna Resort, Sardinia, Italy (13–15 May 2010), https://proceedings.mlr.press/v9/titsias10a.html
17. Williams, C., Barber, D.: Bayesian classification with gaussian processes. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(12), 1342–1351 (1998). https://doi.org/10.1109/34.735807

The Influence Of Variance In GPR And How To Interpret It

Lukas Yikai Xu

Karlsruher Institute for Technologie, 76131 Karlsruhe, Germany

Abstract. Gaussian process regression(GPR) are a generic supervised learning method designed to solve regression and probabilistic classification problems. It returns an estimation in form of a variance that predicts the possible outcome for a coordinate. We analyze which aspects of the GPR affect the resulting variance. This includes the signal variance of the kernel function and noise inflicted input values. Another aspect we are going to take a look at is, that even tho a variance predicts a result, not every prediction should be trusted and under which parameters the prediction becomes misleading.

1 Introduction

GPR is a powerful statistical technique that allows us to model the relationship between variables and make predictions based on observed data. It can be used in multitude of areas for regression like for example in measuring soil temperature [8]. Variance plays a fundamental role in GPR as it measures the spread or dispersion of the data around the regression line. It provides insights into the uncertainty associated with our predictions and allows us to assess the reliability and accuracy of the regression model. Understanding and estimating the variance in GPR is essential for assessing the goodness of fit of our regression model and quantifying the uncertainty associated with our predictions. That's why we are focusing on aspects that influence the variance and analyze under which cases the prediction is sensible.

2 Kernel Function

In this paper we are solely going to use the squared exponential function as the kernel function:

$$k(x, x') = \partial^2 exp(-\frac{(x - x')^2}{2l^2})$$

We focus on the squared exponential function because it is the most common kernel function. It is very common because of its flexibility with its hyper parameters ∂^2 and l which can be adjusted to fit many functions.
The parameter ∂^2 is called signal variance and the parameter l is called length scale. These two parameters have no inherent meaning but are used to configure the function. The kernel function is responsible for the smoothness of the

regression. It determines that by looking at the distance between the two points inserted into the function. When the two points x and x' are identical the function would return ∂^2 and when the two points are very far away the function returns 0. The upper and lower limit are therefore:

$$0 < \partial^2 exp(-\frac{(x-x')^2}{2l^2}) < \partial^2$$

Another important property is that ∂^2 is never negative because otherwise the covariance matrix would not be positive definite anymore which is a important property for the covariance matrix. The covariance matrix is explained in the following section.

3 Gaussian Process Regression

This section we explain roughly how GPR functions. GPR is split into two parts the prior and posterior. The explanations are based of the paper [7]. For a more thorough explanation refer to that paper.

3.1 Prior

In the prior phase no observations are considered. Our model simply returns a multivariate normal distribution. An multivariate normal distribution are simply multiple uni-variate normal distribution combined with each other. An uni-variate normal distribution of random variable X is written as $X \sim \mathcal{N}(\mu, \partial^2)$ with mean μ and variance ∂^2. A uni-variate normal distribution looks like the left graph in the image below:

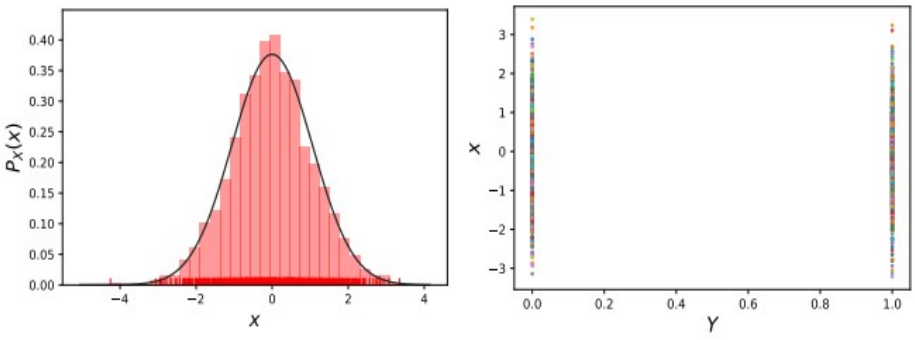


Fig. 1. Left Side: Uni-variate distribution, Right Side: Two uni-variate distributions plotted on y-axis. Image from [7]

By plotting the normal distribution on the y-axis and connecting randomly selected pair of points we get multivariate normal distributions.

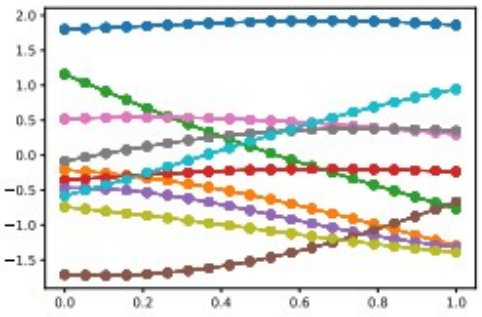


Fig. 2. Multivariate distribution. Image from [7]

3.2 Posterior

In the posterior we will adjust the posterior with newly obtained observations. For a given amount of input values $\{x_1, .., x_n\}$ we have a set of y_i for each x_i where $x_i, y_i \in \mathbb{R}$. We write x = $[x_1, .., x_n]^\intercal$, y = $[y_1, .., y_n]^\intercal \in \mathbb{R}^{N\times 1}$ for the vector variant. The goal of is to find a function f(x) for input x that approximates y. We usually also expect f to be influenced by some noise since most measurements in reality are not 100% accurate :

$$y = f(x) + \epsilon_y$$

Where $\epsilon_y \sim \mathcal{N}(0, \eta^2)$, $f(x) \sim \mathcal{N}(0, K)$. $\mathcal{N}(\mu, K)\mu = E(x)$ K is the covariance matrix using the squared exponential kernel: Which makes the covariance matrix the following:

$$K = k(X, X) = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix}$$

The results of the prior gets filtered. Only the functions that fit the input values will be kept. After that we want to find out the y coordinate for coordinates x_{*i} for which we don't know the y values yet. The group of x_{*i} is denoted as $X_* = \{x_{*1}, ..., x_{*n}\}$. By solving the joint distribution of f with X and X_* as inputs we get the computed mean and variance of the posterior. They are denoted as:

$$\mu_* = m(X_*) + k(X_*, X)(k(X, X) + \eta^2 I_n)^{-1}(Y(X) - m(X))$$

$$\partial_*^2 = k(X_*, X_*) - k(X_*, X)(k(X, X) + \eta^2 I_n)^{-1} k(X_*, X)^\intercal$$

μ_* is a vector where each entry corresponds to the predicted y value for the respective x_{*i}.
∂_*^2 is a matrix where each diagonal entry is the variance for their respective x_{*i}.

$k(X_*, X_*)$ has the same structure as K but just with entries of K_*.
While $k(X_*, X)$ is defined as:

$$K_* = k(X_*, X) = \begin{pmatrix} k(x_{*1}, x_1) & \cdots & k(x_{*1}, x_n) \\ \vdots & \ddots & \vdots \\ k(x_{*m}, x_1) & \cdots & k(x_{*m}, x_n) \end{pmatrix}$$

4 Influences on the Posterior Variance

There are multiple factors that can influence the variance of the posterior. In this section we go into how the signal variance of the kernel function and how noise in the input data affects the posterior variance and what consequences it has for the evaluation of the result.

4.1 Signal Variance

To understand what part the signal variance plays in the posterior variance we assume that X_* has only one element and apply it to the posterior variance formula. For the following calculation we assume $\eta^2 = 0$ for simplicity. There are n input data in X. If we use the formula we get:

$$\begin{aligned} \partial_*^2 &= k(x_*, x_*) - K_* K^{-1} K_*^\intercal \\ &= \partial^2 - (\partial^2 K_*')(\frac{1}{\partial^2} K^{'-1})(\partial^2 K_*'^\intercal) \\ &= \partial^2 - \partial^2 K_* K^{'-1} K_*'^\intercal \\ &= \partial^2 (1 - K_* K^{'-1} K_*'^\intercal) \\ &\leq \partial^2 \end{aligned} \tag{1}$$

We first took out the constant of ∂^2 out of all the covariance matrices. We can do that since every entry of the matrices is of the form of the kernel function which has a ∂^2 as a factor. After that we simplify the term and exclude the ∂^2. The term in the bracket is in the range of [0,1] since $K^{'-1}$ is a positive definite matrix. Because K is positive definite and invertible, K^{-1} is positive definite as well. By excluding ∂^2 the matrix stays positive definite. Therefore the term $K_* K^{'-1} K_*'^\intercal$ is always bigger than zero. If we size it down we get the upper limit which is ∂^2.
This is also the case if we add η^2 to K before inverting. Since η^2 is a positive number because it's the variance of ϵ_y and a variance must be a positive number. If we add a positive number to the diagonal of K then we are not affecting its positive definite property.

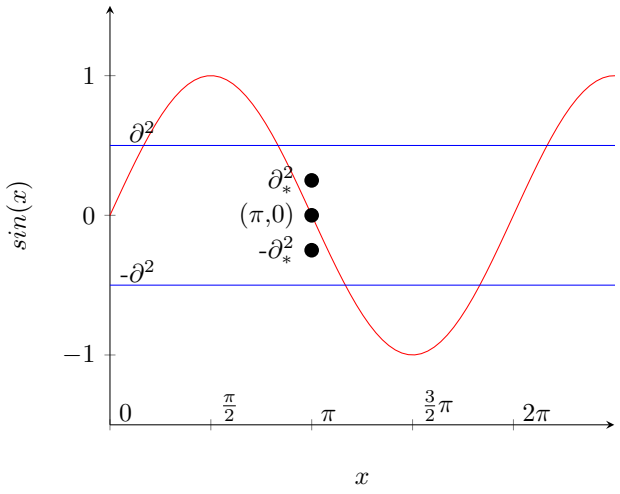
This equation equates to getting the variance for that one point in X_*. Picking the signal variance decides the upper limit that the variance can reach in this model.

The lower limit is 0, and we can confirm that with:

$$\begin{aligned}\partial_*^2 &= \partial^2(1 - K_*' K^{'-1} K_*'^{\intercal}) \\ &> \partial^2(1 - \begin{pmatrix}1 & \cdots & 1\end{pmatrix} M \begin{pmatrix}1 \\ \vdots \\ 1\end{pmatrix}) \\ &> \partial^2(1-1) = 0\end{aligned} \tag{2}$$

Where M is a structure introduced in Section 5 (Eq:3). By maximizing the kernel function terms we can get the lowest number in the right factor. Picking ∂^2 does not change the lower limit.

Graphical Evaluation :
Let's assume we were trying to approximate the sine curve. We already got the mean from our prior which is the red line and one of our unknown points in X^* would be π. For that point a variance is calculated. The blue line in the graph marks the upper/lower limit of the posterior variance for point in π. It should not be confused with the actual variance ∂_*^2 which is smaller than ∂^2 in most cases. By adding/subtracting the variance to/from the mean at x = π we get the range of possible y-values. The evaluation on a graph would look like this:



Since we are working with gaussian distributions, the posterior variance tells us that there is a 68% chance that the y-value of a x_* will be in the range of [mean - variance, mean + variance] which in our case would be $[-\partial_*^2, \partial_*^2]$. For a 95% confidence interval simply use the range [mean - $2\partial_*^2$, mean + $2\partial_*^2$]

If we simply repeat this process for other points in X_* and draw a line trough

all the calculated variance points then we would get the range of the covariance for each point.

In Summary :
We showed that choosing the signal variance ∂^2 with squared exponential function as the kernel determines the maximal achievable posterior variance and limits the posterior variance in the range of $\partial_*^2 \in [0,\partial^2]$. That means, that no matter how bad the initial sample point or parameters are, the variance will never be worse than ∂^2

4.2 Inputdata with Noise

Now we assume that the input variables in X are affected by noise ϵ. We used the model in section 2 of the paper by McHutchon, Andrew, and Carl Rasmussen [1] for this section. Let $X = \{x'_1, ..., x'_n\}$. Each element x'_i in X is the original sample point x which was affected with noise ϵ_{x_i}. Thus we get:

$$x'_i = x_i + \epsilon_{x_i}$$

With these input values the y becomes:

$$y = f(x' + \epsilon_x) + \epsilon_{y_i}$$

Where $\epsilon_y \sim \mathcal{N}(0, \eta^2)$, $\epsilon_x \sim \mathcal{N}(0, \Sigma_x)$.
$\epsilon_y \sim \mathcal{N}(0, \eta^2)$ means that the noise of the y value is normally distributed with variance η^2.
$x\prime$ is the vector with the entries x'_i for each row and ϵ_x is a vector with ϵ_{x_i} as entry for each row. Σ_x is a diagonal matrix that corrupts each input dimension independently with noise. This is just a another way to write that each ϵ_{x_i} is normally distributed with a different variance.

To get the posterior through the equation, the term $f(x_i + \epsilon_{x_i})$ was approximated with the following term:

$$y = f(x) + \epsilon_x^\intercal f'(x) + \epsilon_y$$

Where $f'(x)$ is the derivative of the estimated function f(x) by the prior. The derivative of f(x) is the rise per x-value and by multiplying it with ϵ_x we add the rise that was not considered in f(x). Now this results in the mean and variance of the posterior to be the following:

$$\mu_* = m(X_*) + k(X_*, X)(k(X, X) + \eta^2 I_n + diag(\partial_{f'} \Sigma_x \partial_{f'}^\intercal))^{-1}(Y(X) - m(X))$$

$$\partial_*^2 = k(X_*, X_*) - k(X_*, X)(k(X, X) + \eta^2 I_n + diag(\partial_{f'} \Sigma_x \partial_{f'}^\intercal))^{-1} k(X_*, X)^\intercal$$

The notation $diag\{.\}$ results in a diagonal matrix, the elements of which are the diagonal elements of its matrix argument.

To get a better understanding what noise in input data does to the posterior variance, we assume that in the following calculations that $|X| = |X_*| = 1$. This results in:

$$\begin{aligned}
\partial_*^2 &= k(x_*, x_*) - k(x_*, x)(k(x, x) + \eta^2 I_n + diag(\partial_{f'} \Sigma_x \partial_{f'}^\intercal))^{-1} k(x_*, x)^\intercal \\
&= \partial^2 - k(x_*, x)(\partial^2 + \eta^2 + diag(\partial_{f'} \Sigma_x \partial_{f'}^\intercal))^{-1} k(x_*, x)^\intercal \\
&= \partial^2 - k(x_*, x)(\partial^2 + \eta^2 + z)^{-1} k(x_*, x)^\intercal \\
&= \partial^2 - \alpha^2 (\partial^2 + \eta^2 + z)^{-1}
\end{aligned}$$

We denoted $diag(\partial_{f'} \Sigma_x \partial_{f'}^\intercal)$ as $z \in \mathbb{R}$ where $\partial_{f'}$ is denoted as:

$$\partial_{f'} = [\frac{f(x_i)}{\partial x_i^1}, ..., \frac{f(x_i)}{\partial x_i^n}]$$

The term $(\partial^2 + \eta^2 + z)^{-1}$ is always greater or equal 0. Σ_x is always positive since it's a variance and multiplying it with $\partial_{f'}$ twice doesn't change it. Since $\alpha \in [0, \partial^2]$ we can round off the entire term in the last line to $\partial^2 - (\partial^2)^2(\partial^2 + \eta^2 + z)^{-1} = \partial^2(1 - \partial^2(\partial^2 + \eta^2 + z)^{-1})$. By rounding up we get the upper limit of ∂^2:

$$\partial^2(1 - \partial^2(\partial^2 + \eta^2 + z)^{-1}) < \partial^2 - \alpha^2(\partial^2 + \eta^2 + z)^{-1} < \partial^2$$

That means that the noise in the input variable shifts the variance at point x_* around, but never goes outside of the range of $[\partial^2(1 - \partial^2), \partial^2]$.
If we remove the noise η^2 and z then we get:

$$0 = \partial^2 - \partial^2 < \partial^2 - \alpha^2(\partial^2)^{-1} < \partial^2$$

If we think about the lower limit this way. The rounding up of α^2 to ∂^2 means that x_* and x are identical. Since in case when x_* and x are identical a variance other than zero means that there is a chance that the y-value of x is not the same as x_* which obviously doesn't make sense. So the lower limit must be zero. Adding the noise simply shifts the lower limit up, but doesn't influence the upper limit.

Now if we apply that knowledge to multi dimensional case we would get a similar

term as we have done in Section 3.1:

$$\begin{aligned}
\partial_*^2 &= k(X_*, X_*) - K_*(K + \eta^2 I_n + diag(\partial_{f'} \Sigma_x \partial_{f'}^\intercal))^{-1} K_*^\intercal \\
&= \partial^2 - (\partial^2 K'_*)(\partial^2(K' + \frac{\eta^2}{\partial^2} I_n + \frac{1}{\partial^2} z))^{-1}(\partial^2 K_*'^\intercal) \\
&= \partial^2 - (\partial^2 K'_*)(\frac{1}{\partial^2}(K' + \frac{\eta^2}{\partial^2} I_n + \frac{1}{\partial^2} z)^{-1})(\partial^2 K_*'^\intercal) \\
&= \partial^2 - \partial^2(K'_*(K' + \frac{\eta^2}{\partial^2} I_n + \frac{1}{\partial^2} z)^{-1} K_*'^\intercal) \\
&= \partial^2(1 - (K'_*(K' + \frac{\eta^2}{\partial^2} I_n + \frac{1}{\partial^2} z)^{-1} K_*'^\intercal)) \\
&\leq \partial^2
\end{aligned}$$

With the exception that instead of K^{-1} we would have $(K'+\eta^2 I_n+diag(\partial_{f'} \Sigma_x \partial_{f'}^\intercal))^{-1}$. This doesn't change the fact that the term $(K'_*(K' + \frac{\eta^2}{\partial^2} I_n + \frac{1}{\partial^2} z)^{-1} K_*'^\intercal))$ is a positive definite matrix. Therefore that term is always greater than zero and the upper limit is still ∂^2. So that means that even with noise in our input values, the upper limit of the variance stays ∂^2

In Summary :
Even if the initial sample points are afflicted with noise that will not change the maximal achievable posterior variance but only shift the value around in the variance range. The posterior variance is also guaranteed to not fall under 0. This can be useful when the variance for the results are already known. By choosing ∂^2 as that variance it can be guaranteed that the results will fall in that range.

5 Trustworthiness Of The Variance

In the previous sections we talked about the range of possible variance for the posterior in scenarios with and without noise. The variance tells us in a guassian distribution context that there is a chance of 68% that the computed value at a point will be in the range of that variance range but is that really the case? In this section we are going into scenarios where the variance can be misleading.

5.1 The Objective Function

First we need a model that evaluates how good the results are. This following presented approach is based on the section "Understanding the objective function" of the article [3]. The objective function returns a number which represents how fitting the chosen parameters ∂ and l are given the input values of X. The author focuses on how the length scale l influences the posterior variance, and

he used the logarithmic marginal likelihood function as the model to evaluate the quality of the length scale l:

$$
\begin{aligned}
&log(p(y(x))) \\
&= log(\frac{1}{(2\pi)^{n/2} det(K+\eta^2 I_n)^{n/2}} exp(-\frac{1}{2}(y(X)-m(X))^\intercal (K+\eta^2 I_n)^{-1}(y(X)-m(X)))) \\
&= -\frac{1}{2}log(det(K+\eta^2 I_n)) - \frac{1}{2}(y(X)-m(X))^\intercal (K+\eta^2 I_n)^{-1}(y(X)-m(X)) - \frac{n}{2}log(2\pi)
\end{aligned}
$$

We denote the term $-\frac{1}{2}log(det(K+\eta^2 I_n))$ as model complexity term and the term $-\frac{1}{2}log(det(K+\eta^2 I_n)) - \frac{1}{2}(y(X)-m(X))^\intercal (K+\eta^2 I_n)^{-1}(y(X)-m(X))$ as data fit term. Maximizing this function will result in ideal results. To understand how this function behaves we inspect the two terms in their most extreme cases. The term at the end of the function will be ignored since it is a constant.

The Model Complexity Term :
The model complexity term basically consists of the determinant of K. We assume in the following that $\eta^2 = 0$, $\partial^2 = 1$ and that we have two training points (x_1, y_1) and (x_2, y_2).
If l approaches 0:

$$
\begin{aligned}
&-\frac{1}{2}log(det(K+\eta^2 I_n)) \\
&\approx -\frac{1}{2}log(det(\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix})) \\
&\approx -\frac{1}{2}log(1) \\
&\approx 0
\end{aligned}
$$

If l approaches ∞:

$$
\begin{aligned}
&-\frac{1}{2}log(det(K)) \\
&\approx -\frac{1}{2}log(det(\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix})) \\
&\approx -\frac{1}{2}log(0+\epsilon) \\
&\approx \infty
\end{aligned}
$$

The term $\log(0 + \epsilon)$ represents a term which approaches $-\infty$ and multiplied with $-\frac{1}{2}$ makes it positive.

The effect in these equations can also be simulated with picking input points that are all either very far away from each other or if they are all very close to each other.

5.2 Data Fit Form

The data fit form doesn't have a meaning in of itself but does well in balancing the model complexity term. We assume that $\eta^2 = 0$, $\partial^2 = 1$ and that we have two training points (x_1, y_1) and (x_2, y_2). y(X) simply represents a vector with all the y values for all sample points in X.
For l approaching 0 $l \to 0$:

$$\begin{aligned}
&-\frac{1}{2}(y(X))^\intercal (K)^{-1}(y(X)) \\
&= -\frac{1}{2}(y_1, y_2)\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}^{-1}\begin{pmatrix}y_1\\ y_2\end{pmatrix} \\
&= -\frac{y_1^2 + y_n^2}{2}
\end{aligned}$$

For l approaching ∞:

$$\begin{aligned}
&-\frac{1}{2}(y(X))^\intercal (K)^{-1}(y(X)) \\
&= -\frac{1}{2}(y_1, y_2)\begin{pmatrix}\partial^2 & \partial^2\\ \partial^2 & \partial^2\end{pmatrix}^{-1}\begin{pmatrix}y_1\\ y_2\end{pmatrix} \\
&= -\frac{1}{2}(y_1, y_2) - \frac{1}{(\partial^2)^2 - (\partial^2)^2}\begin{pmatrix}\partial^2 & -\partial^2\\ -\partial^2 & \partial^2\end{pmatrix}\begin{pmatrix}y_1\\ y_2\end{pmatrix} \\
&= -\frac{y_1^2 + 2\partial^2 y_1 y_2 + y_2^2}{2((\partial^2)^2 - (\partial^2)^2)} \\
&= -\infty
\end{aligned}$$

Evaluation :
Since the goal is to maximize the objective function, picking a large l would seem logical at first because through that the complexity term approaches ∞. But the author states that the data fit term approaches $-\infty$ faster [3]. Picking a small length scale on the other hand results in a negative number. Therefore simply picking a large/small length scale will not result in a good posterior variance. The ideal length scale is somewhere in between of these two extremes when the data term is still smaller than the complexity term which is big enough to make the entire equation positive.

5.3 Overfitting

Overfitting happens in the scenario where our model is too uncertain about the results. A very complex model with very small l maximizes the data fit term, however that results in the kernel function returning 0 for all inputs that are not identical regardless of their distance from each other. The result is that the model is not able to make predictions for new points x_*. For the equation we

simplify m(x) to the null function,$\eta^2 = 0$ and $|X_*| = 1$.
This is reflected through the variance of the posterior:

$$\begin{aligned}\partial_*^2 &= K_{**} - K_* K^{-1} K_*^\intercal \\ &= \partial^2 - 0 \begin{pmatrix} \partial^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \partial^2 \end{pmatrix}^{-1} 0 \\ &= \partial^2\end{aligned}$$

Unlike in equation 1 the posterior is identical to ∂^2 and ∂^2 is not the upper limit. As we can see the variance maximizes and thus the model is very uncertain about its prediction.

5.4 Underfitting

Underfitting is the exact opposite of overfitting. In this case the model is too certain about its results. If we have a simple model with a large l then the kernel function will approach ∂^2 regardless of the input values. We simplify m(x) to the null function, $\eta^2 = 0$ and $|X_*| = 1$.
To make the following calculations easier we are gonna introduce a certain structure.

$$\begin{aligned} M &= \begin{pmatrix} 1 & \cdots & 1+\epsilon \\ \vdots & \ddots & \vdots \\ 1+\epsilon & \cdots & 1 \end{pmatrix}^{-1} \\ &= \begin{pmatrix} -\frac{\epsilon+(n-1)}{(n-1)\epsilon^2+n\epsilon} & \cdots & \frac{x+1}{(n-1)\epsilon^2+n\epsilon} \\ \vdots & \ddots & \vdots \\ \frac{x+1}{(n-1)\epsilon^2+n\epsilon} & \cdots & -\frac{\epsilon+(n-1)}{(n-1)\epsilon^2+n\epsilon} \end{pmatrix} \end{aligned} \tag{3}$$

$$\begin{pmatrix}1 \cdots 1\end{pmatrix} M \begin{pmatrix}1 \\ \vdots \\ 1\end{pmatrix} = \frac{n}{\epsilon + n} \tag{4}$$

M is a matrix with 1 on the diagonal and $1 + \epsilon$ in the other entries. The ϵ is a small negative real number.

This results in the following prediction for the posterior variance:

$$
\begin{aligned}
\partial_*^2 &= K_{**} - K_* K^{-1} K_*^\intercal \\
&= \partial^2 - \partial^2 \begin{pmatrix} 1+\epsilon \cdots 1+\epsilon \end{pmatrix} \frac{1}{\partial^2} \begin{pmatrix} 1 & \cdots & 1+\epsilon \\ \vdots & \ddots & \vdots \\ 1+\epsilon & \cdots & 1 \end{pmatrix}^{-1} \partial^2 \begin{pmatrix} 1+\epsilon \\ \vdots \\ 1+\epsilon \end{pmatrix} \\
&\approx \partial^2 - \partial^2 \begin{pmatrix} 1 \cdots 1 \end{pmatrix} M \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \\
&\approx \partial^2 - \partial^2 (\frac{n}{\epsilon + n}) \\
&\approx 0
\end{aligned}
$$

With over- and underfitting we see that choosing bad values for length scale l or input values will result in a model that isn't able to make good predictions. That's why we introduced the objective function. By maximizing it we can find sensible parameters

In Summary :
This shows that even if our variance predicts that our value fall within a certain range, that the predicted range might not be even close to the real values.
Keep in mind that this is only one possible method to judge the results of the model and is only supposed to demonstrate a method on how that could be accomplished.

6 Implications in Practice

In this small section we are going to show through an example what implications our conclusions have in practice and highlight some properties of the squared exponential function(RBF) as kernel. The following graph(Fig.3) uses the squared exponential function as kernel with sample points in x $\in \{-2, -1, 1, 1.5\}$.
The posterior variance gives us a range of possible y-values for a random variable x'. The distance between the mean function at point x = -2 for example and the outer black line is the variance of that point
In this paper we talked about the upper/lower limits of the posterior variance. We have an upper limit of ∂^2 which can be chosen and a lower limit of 0. The closer a random variable is to a sample point the closer the variance is to 0. This can be seen in the graph where the variance has a wave like structures between to sample points and till it becomes 0 once it reaches the sampling point.
We know that the upper limit is ∂^2 but in most cases a variance that big does not occur. The variance increases the bigger the distance is between a random variable and a sample point. This is also reflected in this graph. If we look at the behaviour between two sample points we see that the variance is at its maximum in the middle between the two sample points. Another case where we have

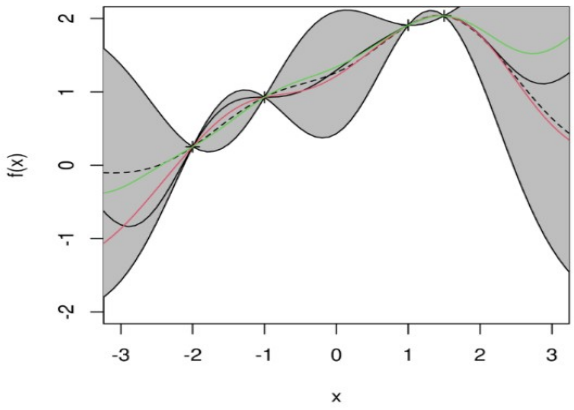


Fig. 3. The variance is the grey area drawn by the outer black line. The other colored lines are possible functions from the prior.The line with dots is the mean function. Image from [6]

high variance is when we inspect -100 as the random variable for example. That would result in a variance that approaches ∂^2.
The length scale influences how fast the posterior variance increases/decreases as the distance between a random variable and a sample point grows. A small length scale leads to wigglier graphs while a larger length scale leads to a smoother graph which can be seen in the graph below.

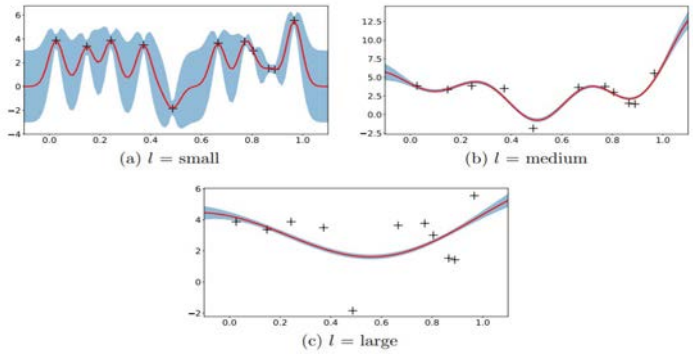


Fig. 4. The red line is the mean function and the blue area is the range of the posterior variance. Image from [7]

7 Conclusion

After inspecting the extreme cases of the kernel function we arrived at the conclusion that the the upper and lower limits of the variance are restricted in a rigid interval which is independent of the input values. This property persists even when we expect the input values and results to be inflicted with noise.

We also looked at an approach on how to determine whether a prediction of a model was sensible or not and how under poorly chosen conditions the prediction was affected. On one end the model could be too certain about it's predictions which resulted in a variance that equals zero across all values or on the other end where the model was too uncertain about it's predictions which resulted in the maximization of the variance.

References

1. McHutchon, Andrew, and Carl Rasmussen. "Gaussian process training with input noise." Advances in neural information processing systems 24 (2011).
2. Johnson, Juan Emmanuel, Valero Laparra, and Gustau Camps-Valls. "Accounting for input noise in Gaussian process parameter retrieval." IEEE Geoscience and Remote Sensing Letters 17.3 (2019): 391-395.
3. Yi.W (2019). Understanding Gaussian Process, the Socratic Way. Toward Data Science
4. Rasmussen, C. E.,Williams, C. K.(2006)"Gaussian processes for machine learning", MIT Press
5. De Freitas,Nandos(2013, 02 05).Machine learning - Introduction to Gaussian processes.Youtube. https://www.youtube.com/watch?v=4vGiHC35j9s
6. Connor Crilly(2022), Graph of posterior distribution, digital image, accessed 21.08.2023
   https://compass.blogs.bristol.ac.uk/2022/01/25/gaussian-process-emulation/
7. Wang, Jie. "An intuitive tutorial to Gaussian processes regression." arXiv preprint arXiv:2009.10862 (2020).
8. Mihoub, Redouane, Nabil Chabour, and Mawloud Guermoui. "Modeling soil temperature based on Gaussian process regression in a semi-arid-climate, case study Ghardaia, Algeria." Geomechanics and Geophysics for Geo-Energy and Geo-Resources 2 (2016): 397-403.

Exoskelette in der Industrie*

Elias Lio Benesch

Karlsruher Institut für Technologie, 76131 Karlsruhe, Deutschland

Abstract. Die Anwendung von Exoskeletten auf dem Arbeitsplatz ist noch ein relativ neues Konzept und man sieht sie noch selten im Betrieb. Ziel dieser Ausarbeitung ist es die pro und kontras einer Integrierung der aktiven mobilen Exoskelette in die Industrie sich anzuschauen, und durch dieser Beobachtung eine Schlussfolgerung zu ziehen, ob es sich für einen Arbeitsplatz lohnt in aktiven mobilen Exoskeletten zu investieren.

Keywords: Exoskelette · Arbeitsplätze · Verletzungsprävention.

1 Einleitung

In den 1960ger Jahren wurde von der amerikanischen Firma, General Electric, ein Prototypen eines robotischen Anzugs entwickelt, der den Trägern verstärken bzw beim heben schwerer Objekte assisstieren sollte. Der Anzug hieß Hardiman und er war das erste Modell eines aktiven mobilen Exoskeletts [1, 2]. Wegen den technologischen Einschränkungen der Zeit war Hardiman leider nicht sehr erfolgreich, seit dem versucht man jedoch die aktiven mobilen Exoskelette weiterzuentwickeln, sodass sie eine Norm in unserer Gesellschaft werden können [2]. Neben dem Militär und der Medizin werden auch in der Industrie nach aktiven mobilen Exoskeletten gefragt. Sie sollen die Gesundheit der Arbeiter schonen, Arbeitsunfälle vermeiden und durch die Verringerung der Anzahl der verletzten Arbeiter auch mengenweise Geld sparren [2–4]. Es ist eine sehr effektive Methode die Kraft einer Machine mit der Denkfähigkeit eines Menschens zu kombinieren. Bevor man aber die Exoskelette auf den Arbeitsplätzen einführt, müssen die Sicherheits- und Gesundheitsabteilungen, Ergonomisten und andere Interessengruppen überzeugt werden, dass die Einfuhrung von aktiven mobilen Exoskeletten Sinn macht [3]. Dies zu zeigen ist das Ziel dieser Ausarbeitung.

2 Recherchemethode

Da die aktiven mobilen Exoskelette in der jetzigen Zeit noch relativ junge technologien sind, bin ich davon ausgegenagen, dass sie sich im Moment noch sehr schnell entwickeln. Es wurden deshalb Artikel aus dem letzten fünf Jahren bevorzugt, um zu vermeiden, dass veraltete Information in die Bearbeitung mit einbezogen wird. Es wurden jedoch auch ältere Artikel im Ausnahmefall akzeptiert,

* Proseminar: Mobile Computing

ins besondere in Fällen wo es sich in den Artikeln nicht direkt um die Rolle der Exoskelette auf den Arbeitsplätzen handelte. Die Rechereche wurde ausschließlich auf Englisch betrieben, alle Artikel auf einer anderen Sprache wurden also nicht angenommen. In den ersten Phasen der Recherche wurden die Schlüsselwörter "mobile active exoskeletons", "industry" und "workplaces" verwendet. Es war wichtig zu bedenken dass es sich in dieser Ausarbeitung um aktive mobile Exoskelette in der Industrie handeln sollte, also wurden alle Artikel, in dem es sich hauptsächlich um passive oder feststehende Exoskelette handelte, ausgeschlossen, und es wurden nur Artikel mit einbezogen, indem es sich um Exoskelette handelten, die für die Arbeit gemacht wurden sind. Während der Recherche wurde mir bewusst, dass das Wort "work-related musculo-skeletal disorders" öfters erschienen ist, also wurde dies auch zu einem wichtigen Schlüsselwort. Sollte ein Artikel zum Thema passen und die Kriterien erfüllen, wurde der Abstract, die Einleitung und die Schlussfolgerung gelesen, und wenn es zur Ausarbeitung passend schien, wurde es durchgelesen und eventuell als Quelle für diese Ausarbeitung mit einbezogen.

3 Motivation

3.1 Die Gesundheit der Arbeiter

Viele Berufe sind offensichtlich sehr belastend für den Körper der Arbeiter. Solche Berufe führen nicht nur zu Arbeitsunfällen, sie sind auch langfristig schädlich für die Gesundheit der Arbeiter. Insbesondere entstehen öfters bei Arbeitern diverse Muskel-Skelett-Erkrankungen [2, 3]. Ein gutes Beispiel ist die Auto Industire. Es wurde zwar in der Automanufaktur schon die Arbeit viel leichter und sicherer gemacht, indem vieles des Aufbaus von Maschinen übernommen wurde, es kann aber nicht der ganze Prozess durch Automatisierung ersetzt werden [5, 6]. Viele Schritte beim Aufbau eines Autos werden noch manuell von Menschen geleistet, und in der Manufaktur werden nicht mehr als ein paar Minuten pro Wagen verwended, dadurch sind also Arbeiter gezwungen mit einer hohen Frequenz, und einer unangenehme Körperaltung, Autoteile zu montieren, was langfristig zu Störungen im Bewegunsapperat führt [3]. Solche Muskel-Skelett Erkrakungen, die in der Arbeit entstehen, betreffen nicht nur offensichtlich körperlich belastende Arbeitsplätze wie die Manufaktur, Logistik, Bauarbeit oder die Landwirtschaft, sie können auch in Arbeitsplätzen wie im Büro oder in der Medizin entstehen. Beispielsweise erleiden Zahnärzte in den meisten Fällen nach einiger Zeit Schmerzen im Nacken, Schultern oder Rücken, obwohl sie auf dem ersten Blick ihre Körper nicht so stark belasten müssen [7]. Dies liegt daran, dass sie beim behandeln ihrer Patienten eine umständliche Haltung annehmen, um in den Mund ihrer Patienten hineinzuschauen oder zu operieren. 39% der Arbeitsbedingten Verletzungen sollen Muskel-Skelett-Erkrankungen sein [8]. Dies senkt nicht nur die Lebensqualität Arbeiter selber, doch die Finanzen der Arbeitspläzen werden auch massiv beeinflusst. Durch Kompensationskosten für die Erkranketen, Verlust in Arbeiter und die Notwedigkeit neue Arbeiter zu rekrutieren und zu trainieren, verlieren die Arbeitsplätze auch große Mengen an Geld

[3]. Es wurde geschätzt dass 0,5-2%, evtl bis zu 4%, vom Bruttonationaleinkommen der EU nur wegen Muskel-Skelett-Erkrankungen ausgegeben werden [2, 8]. Zusätzlich führen solche Verletzungen auch zu einer massiven Senkung der Produktivität in der Wirtschaft. Arbeitsbedingte Muskel-Skelett-Erkrankungen gehören also zu einem Problem den man möglichst früh Lösen sollte, ins Besondere da die Anzahl der betroffenen Arbeiter auch zu wachsen scheint [8, 9].

3.2 Exoskelette als Lösung

Um die Anzahl der arbeitsbedingten Muskel-Skelett-Erkrankungen zu reduzieren werden ergonomische Lösungen vorgeschlagen, wie Programme die eine gesunde Körperbewegung fordern oder alternative Arbeitsgegenstände mit ergonomischen Designs [9]. Einer der größeren ergonomischen Lösungen, die sowohl langfristige Schäden am Bewegungsapperat als auch Arbeitsunfälle verhindern soll, ist die Integration der Exoskelette auf Arbeitsplätzen. Diese leichte robotische Anzüge sind nun auf dem Markt verfügbar [12, 13], und sie sollen sowohl die Produktivität der Arbeiter erhöhern als auch den Arbeitsplatz zu einem sichereren und ergonomischeren Ort transformieren. Sie sollten auch eine Firma finanziell verstärken, indem Kompensationskosten und Krankheitstage verringert werden.

4 Arten von Exoskeletten

Es gibt viele Wege Exoskelette in Untergruppen einzuteilen [4]. Erst mal kann man sie in deren Aufgaben unterscheiden, d.h. ob sie in der Medizin, im Militär oder in der Industrie verwendet werden. Dann kann man sie in passive und aktive bzw semi aktive Exoskelette unterteilen [3, 10]. Der Unterschied liegt in der Energiequelle. Aktive Exoskelette sind robotisch und beinhalten Teile die eine Stromquelle brauchen, wie Aktoren oder Motoren, Passive Exoskelette beinhalten jedoch keine Teile die sich von selber bewegen und verlassen sich stattdessen auf Federn oder Dämpfer um die Kraft des Trägers zu vergrößern [10]. Beide Arten werden in der Industrie verwendet, passive Exoskelette sind jedoch weiterentwickelter als ihre robotischen Gegenstücke, da passive Exoskelette weniger komplexere Probleme mit sich bringen [3]. Eine weitere Unterteilung, die wichtig für dieser Ausarbeitung ist, ist die Mobilität des Exoskeletts, d.h. ob sie sich mit dem Trägern bewegen kann oder ob sie an einem Ort festgebunden ist. Wie schon erwähnt, wird in dieser Ausarbeitung der Fokus exklusiv auf industriellen, aktiven und mobilen Exoskeletten gelegt. Aktive mobile Exoskelette können in weitere Untergruppen eingeteilt werden, wie wo sie Gebaut wurden sind (von einer Firma, von der Regierung, experimentell in einem Labor usw), welche Form sie hat (ist es complet flexibel ist oder besteht es aus starren Teilen) oder für welchen Körperteilen sie gedacht ist (Oberkörper, Beine, Rücken usw) [4]. Jedes Exoskelett sollte für maximalle effizienz eigenschaften haben die genau für einen Job passt, d.h. es sollten so viele Arten von Exoskelette geben wie es Arbeiten gibt, indem ein Exoskelett integriert werden kann [4].

5 Potenzielle Nachteile

5.1 Sicherheit

Auch wenn einer der Hauptmotivationen der Exoskelette es ist, den Arbeitern Sicherheit zu liefern und Arbeitsunfällen zu verhindern, können Exoskelette Quellen für weitere Gefahren sein, die es auf dem Arbeitsplatz davor nicht gab. Ins Besondere gilt dies für aktive Exoskelette, da man beim Tragen eines aktiven Exoskeletts weniger Kontrolle über die beweglichen Teile hat, als beim Tragen eines passiven Exoskeletts, dessen Bewegung ausschließich vom Nutzer abhängig ist. Das Tragen eines robotischen Anzugs kann also zu schweren maschinellen Unfällen führen, wenn sie nicht richtig gebaut werden. Es wurde auch Bedenken geäußert, dass gerade das Sicherheitsgefühl beim Tragen eines Exoskeletts den Arbeitern anfälliger für Gefahren machen könnte. Das Ausrutchen oder Stolpern ist viel gefährlicher wenn man einen Exoskelett trägt. Weiter Gefahren auf die man achten muss ist das Risiko dass der Träger sehr nah an einer lebensbedrohlichen Energiequelle ist, da aktive mobile Exoskelette elektrisch betrieben werden müssen [4]. Außredem heißt dies auch, das Teile des Exoskeletts sich durch der Energiezufuhr erhitzen kann. Um Verbrennungen zu vermeiden müssen diese Teile von dem Trägern fern gehalten werden [4]. Diese Probleme sind aber leicht lösbar, man muss nur sicherstellen, dass die Technologie, die im Exoskelett implementiert wurden ist, fortgeschritten genug ist.

5.2 Ergonomie

Neben Unfallvermeidung sollen industrielle Exoskeletten auch die Anzahl der Muskel-Skelett-Erkrankungen unter Arbeitern verringern. Es ist also wichtig dass die Form der robotischen Anzüge so entworfen werden, dass der Nutzer nach längerem Tragen des Exoskeletts nicht unter Unbequemlichkeiten oder Schmerzen leiden, sonst hat das Exoskelett bei seiner Aufgabe versagt. Eine Herausforderung für die Designer besteht darin, dass, im Gegensatz zu einer Maschine, ein Mensch sich in vielen verschieden Arten bewegen kann [3]. Beim mehrmaligen durchführen einer Aufgabe kann eine Maschine die selbe Bewegung bis zum Ende ihrer Lebenszeit wiederholen, ein Mensch wird sich jedoch jedes Mal anders bewegen. Deshalb sollte vor dem designen eines Exoskeletts die Kinematik (Bewegung) des Menschens kennen, damit der Nutzer nicht davon gehindert wird sich frei zu bewegen und damit seine Produktivität vermindert wird [4]. In dem Feld wurden einige Körperteile mehr erforscht als andere. Unterstützung für die Arme und unteren Rücken wurden zum Beispiel mehr erforscht als für die Handgelenke oder Knie [3]. Der ergonomische Aspekt eines idealen Exoskeletts macht den Aufbau also viel komplizierter.

5.3 Kosten

...

5.4 Fehlende Standards

6 Verfügbare Exoskelette

Nachdem wir uns angeschaut haben, was die Vorteile und Nachteile der aktiven mobilen Exoskeletten sein könnten, macht es Sinn sich anzuschauen, was es für Modelle in der reellen Welt gibt, und wie sie mit den potentiellen Nachteilen umgehen.

6.1 Guardian XO (Sarcos)

Sarcos ist eine amerikanische Firma die sich auf mobilen robotischen Systemen speziallisiert, die in diversen Arbeitsplätzen, wie auf der Baustellen, auf dem Feld oder sogar unter Wasser, angewandt werden sollen. Im Moment haben sie einen Modell eines Exoskeletts unter ihren Namen, nämlich den Guardian XO [12]. Der Guardian XO ist ein aktives mobiles ganz-Körper Exoskelett der bis zu 90kg mehrmals hintereinander tragen kann, ohne dass der Träger das Gewicht in irgendeiner Art und Weise spürt, weder vom Gegenstand was getragen wird noch vom Anzug. Es gibt jedoch die Option den Trägern ein Teil des Gewichts des Gegenstandes spüren zu lassen, um zu vermeiden dass der Träger zu wenig vom Umfeld wahrnimmt. Es sollte jeweils nur eine halbe Minute dauern in den Anzug hineinzusteigen, und ihn wieder auszuziehen. Man kann mit ihn 4.8km/h laufen, so schnell wie ein durchschnittlicher Mensch. Die Aufgabe des Guardian XOs ist einem Arbeitern zu helfen, schwere Gegenstänge zu tragen, ohne das er körperlich belasted wird. Das Exoskelett kann in der Manufaktur, auf der Baustelle, im Lager und im Vertrieb, in der Öl- und Gasindustrie, in maritimen Bereichen, im Militär und in der Luft- und Raumfahr benutzt werden. [12]

Fig. 1. Werbefoto für Guardian XO von Sarcos [12]

6.2 Cray X und Apogee (German Bionic)

Eine weitere Firma die Exoskelette herstellt ist die deutsche Firma, German Bionic [13]. Diese Firma speziallisiert sich auf aktiven mobilen Exoskelette, und stellt sowohl für die Pflege als auch für die Industrie Exoskelette her. Zwei ihrer industriellen Exoskeletten heißen Cray X und sein Nachfolger, Apogee. Sie sind, so wie der Guardian XO, aktive mobile Exoskelette, die den Arbeitern beim Tragen schwerer Gegenstände unterstützen. Im Gegensatz zum Guardian XO sind beide Anzüge jedoch keine ganz-Körper Anzüge, sondern sie bedecken nur den unteren Rücken und die Oberschenkel. Diese kompaktere Form dient als Vorteil, dafür sind aber beide Modelle nur für das Tragen von 30kg pro Hebevorgang gedacht, ein Drittel vom Guardian XO, es gibt also einige Aufgaben die nicht für Cray X oder Apogee gedacht sind.

Fig. 2. Werbefoto für Apogee von German Bionic [13]

7 Kann man Exoskelette auf Arbeitsplätzen integrieren?

7.1 Meinung der Arbeiter

Bevor Exoskelette richtig auf einem Arbeitsplatz integriert werden können, brauchen die Interesengruppen Beweise dass solch eine Investition Sinn macht [3]. Einer

der wichtigeren Aspekte bei der Einführung einer neuen Technologie im Arbeitsplatz ist wie wohl sich die Arbeiter selber mit der neuen Technologie fühlen. Dies ist besonders der Fall mit Exoskeletten, da deren Hauptaufgabe es ist die ergonomische Lage der Arbeiter zu verbessern. Um zu untersuchen ob die Arbeiter bereit sind aktive mobile Exoskelette als Teil ihrer Arbeit zu haben, wird gerne das UTAUT Modell verwendet [15].

Das UTAUT Modell (Unifying Theory of Acceptance and Use of Technology) enthält vier Kernfaktoren die determinieren sollen, ob Arbeiter zufrieden mit einer neuen Technologie sind, oder ob diese neue Technologie bei bestimmten Aspekten verbesserungen braucht. Die vier Kernfaktoren sind:

- Leistungserwartung: Hilft die neue Technologie den Arbeitern effizienter zu arbeiten?
- Aufwandserwartung: Ist es einfach zu lernen, wie man diese neue Technologie verwendet?
- sozialer Einfluss: Fühlt man sich wohl unter den Kollegen, wenn man mit der neuen Technologie arbeitet?
- erleichternde Bedingungen: Werden die Arbeiter richtig auf dieser neuen Technologie vorbereitet? [4, 15]

Dazu werden noch die Prozentanteile für Geschlecht, Alter, Erfahrung und Freiwilligkeit die Technologie zu verwenden dokumentiert, da diese Verhältnisse die Ergebnisse der Umfragen beeinflussen könnten [15]. Da industrielle active mobile Exoskelette noch in der praxis sehr neu sind, handelte es sich bei den meisten Umfragen zum Thema Exoskelette um passive Exoskelette. Es stehen wenige Resultate zur Zufriedenheit der aktiven industriellen Exoskelette zur verfügung. Um klar sagen zu können ob Arbeiter bereit wären aktive mobile Exoskelette auf der Arbeit zu verwenden, müsste man sie weiter in der Praxis sich anschauen.

7.2 Meinungen der Interessengruppen

8 Schlussfolgerung

References

1. Mosher, Ralph S. “Handyman to Hardiman.” SAE Transactions, vol. 76, 1968, pp. 588–97. JSTOR
2. Bogue, R. (2019), ”Exoskeletons – a review of industrial applications”, Industrial Robot, Vol. 45 No. 5, pp. 585-590. https://doi.org/10.1108/IR-05-2018-0109
3. Crea, S., Beckerle, P., De Looze, M., De Pauw, K., Grazi, L., Kermavnar, T., . . . Veneman, J. (2021). Occupational exoskeletons: A roadmap toward large-scale adoption. Methodology and challenges of bringing exoskeletons to workplaces. Wearable Technologies, 2, E11. https://doi.org/10.1017/wtc.2021.11
4. F. O. Dengiz, ”Research and Development on Mobile Powered Upper-Body Exoskeletons for Industrial Usage,” 2022 21st International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 2022, pp. 1-6, https://doi.org/10.1109/INFOTEH53737.2022.9751255.

5. Spallek, M., Kuhn, W., Uibel, S. et al. Work-related musculoskeletal disorders in the automotive industry due to repetitive work - implications for rehabilitation. J Occup Med Toxicol 5, 6 (2010). https://doi.org/10.1186/1745-6673-5-6
6. A. Voilqué, J. Masood, J. Fauroux, L. Sabourin and O. Guezet, "Industrial Exoskeleton Technology: Classification, Structural Analysis, and Structural Complexity Indicator," 2019 Wearable Robotics Association Conference (WearRAcon), Scottsdale, AZ, USA, 2019, pp. 13-20 https://doi.org/10.1109/WEARRACON.2019.8719395
7. Ohlendorf, D.; Naser, A.; Haas, Y.; Haenel, J.; Fraeulin, L.; Holzgreve, F.; Erbe, C.; Betz, W.; Wanke, E.M.; Brueggmann, D.; et al. Prevalence of Musculoskeletal Disorders among Dentists and Dental Students in Germany. Int. J. Environ. Res. Public Health 2020, 17, 8740. https://doi.org/10.3390/ijerph17238740
8. Van Eerd D, Munhall C, Irvin E, et al. Effectiveness of workplace interventions in the prevention of upper extremity musculoskeletal disorders and symptoms: an update of the evidenceOccupational and Environmental Medicine 2016;73:62-70. https://doi.org/10.1136/oemed-2015-102992
9. Sundstrup, E., Seeberg, K.G.V., Bengtsen, E. et al. A Systematic Review of Workplace Interventions to Rehabilitate Musculoskeletal Disorders Among Employees with Physical Demanding Work. J Occup Rehabil 30, 588–612 (2020). https://doi.org/10.1007/s10926-020-09879-x
10. Fournier DE, Yung M, Somasundram KG, Du BB, Rezvani S, Yazdani A (2023) Quality, productivity, and economic implications of exoskeletons for occupational use: A systematic review. PLoS ONE 18(6): e0287742. https://doi.org/10.1371/journal.pone.0287742
11. Pesenti M, Antonietti A, Gandolla M, Pedrocchi A. Towards a Functional Performance Validation Standard for Industrial Low-Back Exoskeletons: State of the Art Review. Sensors. 2021; 21(3):808. https://doi.org/10.3390/s21030808
12. Sarcos Homepage, https://www.sarcos.com/ Last accessed 15 July 2023
13. German Bionic Homepage, https://germanbionic.com/ Last accessed 16 July 2023
14. Sunwook Kim, Albert Moore, Divya Srinivasan, Abiola Akanmu, Alan Barr, Carisa Harris-Adamson, David M. Rempel & Maury A. Nussbaum (2019) Potential of Exoskeleton Technologies to Enhance Safety, Health, and Performance in Construction: Industry Perspectives and Future Research Directions, IISE Transactions on Occupational Ergonomics and Human Factors, 7:3-4, 185-191 https://doi.org/10.1080/24725838.2018.1561557
15. Elprama, S. A., Vannieuwenhuyze, J. T. A., De Bock, S., Vanderborght, B., De Pauw, K., Meeusen, R., & Jacobs, A. (2020). Social Processes: What Determines Industrial Workers' Intention to Use Exoskeletons? Human Factors, 62(3), 337–350. https://doi.org/10.1177/0018720819889534

Sensoren in aktiven und mobilen Exoskeletten: Eine Review

Damian Reich

Karlsruher Institut für Technologie

Abstract. In der Steuerung von Exoskeletten spielt die Sensorik eine wichtige Rolle. Als Informationsquelle für die Steuerung des Skeletts und des damit verbundenen Menschen ist die richtige Wahl der Sensoren entscheidend, um eine präzise Einschätzung der Intentionen des Nutzers anhand von Echtzeitdaten zu erhalten. Diese Review präsentiert einen Überblick über den Einsatz von Sensoren in aktiven und mobilen Exoskeletten aus sowohl medizinischen als auch nicht-medizinischen Anwendungsbereichen. Hierbei wird sowohl umfassend auf weitverbreitete Beispiele in Forschung und Praxis der letzten Jahre eingegangen, als auch auf den technischen Aufbau der darin verwendeten Sensorik. Da mobile Exoskelette meistens nah am menschlichen Körper angebracht werden, können die Beispiele verlässlich in spezifische Körperregionen unterteilt werden. In dieser Review werden vorkommende Beispiele zur Übersicht in drei Bereiche gegliedert: Hand, Unterkörper und Oberkörper. Zudem erfolgt ein Einblick in Prinzipien der Gangerkennung und der Sensordatenfusion, welche fallabhängig eine große Rolle im Design des Exoskeletts und dessen Sensorik spielen können.

Keywords: aktive Exoskelette · mobile Geräte · Sensor · Gangunterstützung

1 Einleitung

Maschinelle Exoskelette sind tragbare Roboter, welche dem Träger bei einer körperlichen Tätigkeit unterstützen sollen. Damit können die physischen Fähigkeiten eines Menschen über dessen Maßstab angehoben werden und es eröffnen sich neue, vielfältige Möglichkeiten. Besonders unter Anbetracht der Fortschritte in der technischen Forschung besitzt dieses Gebiet der Robotik die Möglichkeit, körperliche Einschränkungen zu überwinden und die Fähigkeiten eines Menschen neu zu erfinden.

Exoskelette werden oft unter zwei Gesichtspunkten klassifiziert: Lokalisierung am Körper und Anwendungsbereich. Unter der Lokalisierung versteht man die Klassifizierung der Exoskelette in die entsprechende Körperregion, in welcher das Skelett entsprechende Gliedmaßen in ihrer Bewegung unterstützt. Da mobile Exoskelette meist möglichst kompakt am Körper sitzen müssen, ist diese Unterteilung gewöhnlich einfach vorzunehmen. Man unterscheidet zwischen Handexoskelett, Unterkörperexoskelett (engl. Lower Limb Exoskeleton, LLE) und Oberkörperexoskelett (engl. Upper Limb Exoskeleton, ULE). Die andere Möglichkeit ist die Klassifizierung nach Anwendungsbereich. Dieser bezieht sich spezifischer auf das zu bewältigende Problem, welches von dem Exoskelett gelöst werden soll. Man unterscheidet dabei in zwei große Gruppierungen, die medizinischen und die nicht-medizinischen Anwendungen.

Im medizinischen Fall werden die Exoskelette von Patienten verwendet, die aufgrund einer körperlichen Behinderung in ihrer Bewegungsfreiheit eingeschränkt sind. Sie sollen dabei meist die Fortschritte in der Phase der Rehabilitation beschleunigen, indem betroffene Gliedmaßen der Patienten durch die Unterstützung verbessert gesteuert werden können. Ein großer Teil der Forschung behandelt dabei LLEs. Diese werden für die Hüfte und Beine des Nutzers gestaltet und sollen der Person bei der Fortbewegung beisteuern oder diese sogar erst ermöglichen [1, 2]. Ein anderer großer Einsatzbereich ist die Entwicklung von Handexoskeletten, die einen Teil der Aktivitäten wieder möglich machen sollen, welche die Verwendung der menschlichen Hand voraussetzen. Dies umfasst meistens das Greifen von Sachen und ist durch die hohe Komplexität der menschlichen Hand erschwert [3].

Nicht-medizinische Anwendungen finden sich bei Personen wieder, dessen körperlichen Fähigkeiten funktionstüchtig sind und durch Nutzung des Roboters verbessert werden sollen. In der Industriebranche können beispielsweise Arbeiter mit Exoskeletten ausgestattet werden. Diese dienen zum einen der Reduzierung der körperlichen Last und damit auch der Unfallrate [4]. Sie können jedoch auch die vom Nutzer ausgeübte Kraft um ein Vielfaches verstärken und somit Arbeitsprozesse beschleunigen oder erzeugen. Ein weiterer Verwendungszweck findet sich in militärischen Anwendungen wieder. Ähnlich zum industriellen Zweck können Soldaten mit den Anzügen die selbst auszuübende Kraft verringern, welche ansonsten zum Tragen der Ausrüstung benötigt werden würde [5]. Damit kann die Ausdauer deutlich erhöht werden. Außerdem wird somit auftretenden körperlichen Problemen durch physischer Belastung vorgebeugt.

Bei aktiven Exoskeletten handelt es sich um Roboter, dessen mechanische Struktur mithilfe von Aktuatoren gesteuert und manipuliert werden kann. Um diese Elemente anzusteuern benötigt es ein System an Sensoren, welche an dem Skelett selber oder an dem Nutzer angebracht sind. Damit können Kräfte und Signale des Nutzers, sowie auch teilweise der Umwelt, möglichst effektiv und zuverlässig gemessen werden. Auf Basis dieser Informationen findet eine Evaluation in der Steuereinheit des Exoskeletts statt, welche die Bewegungsintention des Nutzers approximieren soll. Infolge eines Algorithmus erzeugt die Steuerung passende Steuersignale, welche schließlich an die Aktuatoren gesendet werden und das Exoskelett bewegt sich.

Diese Review dient dazu, einen Überblick über die Sensoren in aktiven und mobilen Exoskeletten zu schaffen. Dabei werden Beispiele aus Forschungsergebnissen und praktischen Anwendungen in den zuvor genannten Klassifikationen betrachtet. [6]

2 Prinzip der Sensorik

Das grundsätzliche Ziel der eingebauten Sensorik ist es, die Mensch-Maschine-Interaktion zu quantifizieren und zu messen. Dadurch sollen Intentionen des Nutzers in seiner Bewegung eingeschätzt und diese dementsprechend unterstützt werden. Die Präzision sollte dabei möglichst hoch sein, wobei negative Auswirkungen auf die Bewegungsfreiheit des Menschen möglichst gering sein sollen. Die benötigte Information ist neben dem Einsatz des Exoskeletts auch abhängig von Faktoren wie Design, Größe und Kosten. Somit kann sich das Sensorsystem in gleichen Anwendungsfällen stark unterschieden.

Tiboni et al. [7] beschreibt mit einer grundlegenden Architektur für aktive Exoskelette den Verlauf von Informationen im System. Dabei findet die Datenerhebung in den Sensoren statt, die bei mobilen Exoskeletten intern angebracht sind. Die Daten stammen entweder aus der Interaktion des Nutzers mit dem Sensor, Interaktionen zwischen maschinellen Bauteilen (vgl. Zoss und Kazerooni [8]) oder Interaktionen zwischen Exoskelett und externen Einflüssen (vgl. Imtiaz et al. [9]). Alle Werte werden an eine Kontrolleinheit weitergeleitet, welche daraus die Intention des Nutzers ableiten soll. Daraufhin werden die berechneten Signale an die Aktuatoren gesendet. Diese üben dann, je nach Typ des Aktuators, eine Kraft auf die mechanische Vorrichtung aus und manipulieren diese so, dass die Bewegung möglichst sicher ausgeführt wird.

2.1 Gangerkennung

Neben einfacher Datenerhebung existiert in Lower Limb Exoskeletten für die Rehabilitation meist ein weiteres Prinzip in der Intentionserkennung. Die Gangart (engl. gait) ist eine Beschreibung der Art und Weise, wie ein Mensch geht. Sie kann als sich wiederholender Zyklus untersucht werden. Der Zyklus wird für die weitere Analyse in Phasen aufgeteilt (siehe Abb. 1). Das Verständnis dieser

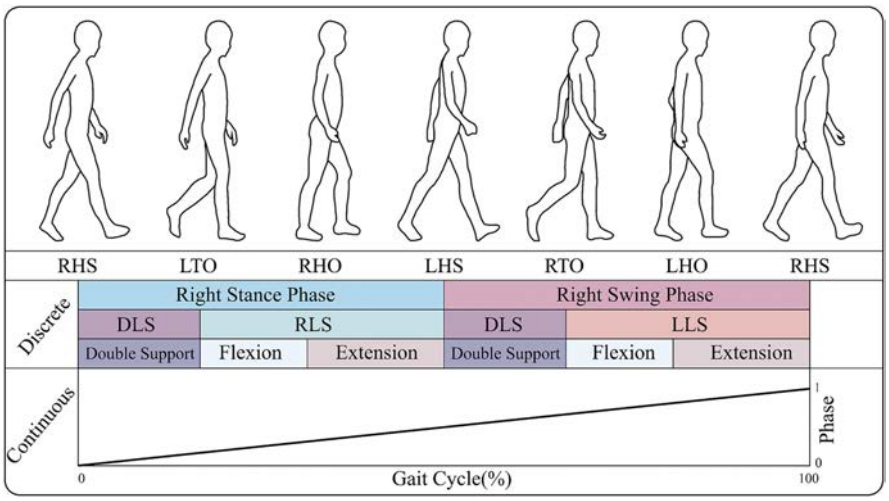


Abb. 1. Mögliche Aufteilung der Gangphase in auswertbare Segmente aus [6].

Phasen kann dabei helfen, das Skelett besser bei der Umsetzung des Ganges zu steuern.

Der Gang ist jedoch abhängig von vielfältigen Faktoren, wie Anatomie, physiologischer Funktionen, Bewegungskontrolle oder aber auch Schrittweite und Geschwindigkeit [6, 10]. Somit sind zusätzlich personenbezogene Anpassungen erforderlich.

2.2 Sensordatenfusion

Bei der Steuerung eines Exoskelett-Roboters ist es wichtig, eine möglichst gute Einschätzung der Umgebungs- und Nutzerinformationen zu erhalten und miteinzubeziehen. Dabei ist die Verwendung von nur einem Sensor unüblich, da die erhaltenen Informationen meist nicht ausreichend sind, um eine hinreichend präzise Einschätzung zu erlangen.

Sensordatenfusion (auch Sensor Fusion) bezeichnet das Zusammenfügen von Informationsdaten aus mehreren Sensoren. Hierbei kann zwischen unimodalen Systemen und multimodalen Systemen [11]. Unimodale Systeme verwenden nur einen Typen von Sensor in der gesamten Struktur des Exoskeletts. In multimodalen Systemen hingegen existieren unterschiedliche Sensortypen, wodurch Schwachstellen in einer Art von Sensor überkommen werden können. Die Schwierigkeit hier ergibt sich durch die größeren Unterschiede der resultierenden Daten. Dadurch werden neue Fusionsalgorithmen benötigt, welche auf das System spezialisiert sein müssen.

Ein Beispiel hierzu ist eine Kontrollmethode von Kiguchi et al. [12], welche Daten sowohl von Elektromyografie-Sensoren, die elektrische Muskelreize messen, als auch von Kraftsensoren am Handgelenk fusioniert. Mithilfe eines neuronalen

Netzwerkes, welches als Eingabe die genannten Daten erhält, können Steuersignale für das Arm-Exoskelett erzeugt werden.

3 Überblick auf die Sensoren

Bei den nun vorgestellten Sensortypen handelt es sich um solche, die in Designs und Anwendungen von Exoskeletten weit verbreitet sind. Die Einschätzung der Verbreitung eines Sensortypen erfolgte mithilfe einer systematischen Literaturrecherche [7].

Der Überblick über einen Sensor beginnt mit einer Erklärung der technischen Zusammensetzung. Darauf folgt eine geordnete Sammlung aus Beispielen von Einsätzen dieses Sensors in Exoskeletten. Die Exoskelette werden hierbei nach der entsprechenden Körperregion kategorisiert, in welcher sie den Nutzer aktiv in der Bewegung unterstützen sollen. Die Gruppierung erfolgt in ULEs, LLEs und Handexoskelette.

3.1 Kraftsensoren

Kraftsensoren (engl. force sensor) werden eingesetzt, um die an der Oberfläche des Sensors auftretende Kraft zu messen. In Exoskelett-Anwendungen wird meist die Kraft vom Nutzer direkt ausgeübt. Sie kann jedoch auch zwischen mechanischen Teilen oder außerhalb des Skeletts entnommen werden, um Einwirkungen von externen Objekten aufzunehmen [9].

Für die Kraftsensoren in Exoskeletten werden häufig Kraftmesswiderstände (engl. force sensing resistors, FSR) verwendet. Bei einem FSR handelt es sich um einen dünnen leitfähigen Film, welcher unter Krafteinwirkung seinen internen Widerstand verändert [13]. Damit kann die Krafteinwirkung quantifiziert werden (siehe Abb. 2). FSRs werden aufgrund ihrer schmalen Breite und ihrer geringen Kosten gegenüber anderen Sensoren oft bevorzugt (vgl. [14, 15]).

Eine weitere beliebte Methode umfasst die Verwendung von Dehnungsmessstreifen (engl. strain gauge) [16]. Dieser ähnelt in dessen Aufbau und Funktionsweise einem FSR. Der Unterschied bei dem Dehnungsmessstreifen liegt darin, dass dieser die einwirkende Kraft durch Verformung eines Substrates im Streifen misst. Ein FSR hingegen misst die Einwirkung auf das Sensorelement selbst. Dehnungsmessstreifen können in anderen Sensoren angewendet werden. So findet man auch Verwendungen von Ladezellen (engl. load cell) in Exoskelett-Designs [17]. Diese bestehen aus einer Konfiguration von mehreren Dehnungsmessstreifen und können genauere Messungen durchführen, aufgrund ihrer erweiterten Menge an Messdaten.

Aufgrund der flexiblen Einsetzbarkeit werden in einigen Anwendungen Kraftsensoren für das Messen von Druck oder Drehkraft umfunktioniert [8].

Um beim Tragen unterschiedlich schwerer Objekte die vom Nutzer auszuübende Kraft konstant zu halten, wird von Wang et al. [18] ein Sensor am Griff am Ende des Exoskelettarms installiert. Dieser Sensor misst die Kraft, welche durch

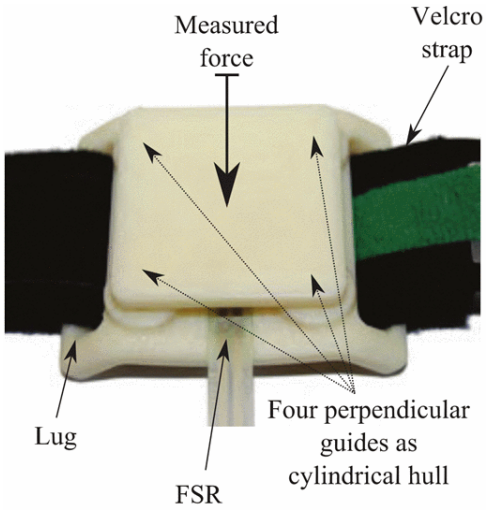


Abb. 2. Einsatz eines FSRs unter einer beweglichen Platte zur Messung der Muskelspannung am Bein aus [14].

das Gewicht des getragenen Objektes entsteht. Die Aktuatoren können so ihre Kraftübertragung auf den Arm anpassen in, Abhängigkeit zum getragenen Objekt. Das Design von Liu et al. [19] zielt auf eine alltägliche Nutzung ab und legt Wert auf präzise Sensitivität und einen hohen Freiheitsgrad. Hierzu wurde ein 3-Achsen Kraftsensor unter der Handfläche platziert. Dieser misst Kraftausübungen durch Bewegung der Hand des Nutzers in alle Richtungen und ermöglicht so eine präzise Bewegung des Exoskelett-Arms.

Kraftsensoren in Lower Limb Exoskeletten finden häufige Verwendung in den Sohlen des Nutzers. Pappas et al. [20], Kim et al. [21] und Yang et al. [22] beziehen sich in ihrem Entwurf auf die Platzierung von mehreren Sensoren unter den Fußsohlen. Dabei kann die Übertragung des Gewichts verteilt gemessen und anhand der Daten die Gangphase approximiert werden. Zusätzlich zur Kraftmessung zwischen Fuß und Boden besitzt das Konzept von Chen et al. [23] für jedes Bein zwei weitere zweidimensionale Kraftsensoren in Schienbein- und Schenkelregion. Diese sind über Aluminiumbänder jeweils mit den Beinen verknüpft und messen die Intention über den Bewegungsunterschied zwischen Mensch und Skelett. Aufgrund der Vorteile, die Kraftsensoren bieten, werden sie auch als Ersatz anderer Sensortypen verwendet. Zoss und Kazerooni [8] greifen in der Strukturierung des Berkeley Lower Extremity Exoskeleton (BLEEX) darauf zu. Beim Erfassen des Drehmoments im Knie verzichten sie auf einen entsprechenden Drehmoment-Sensor, da dieser die Breite des Gelenks zu sehr vergrößern würde. Stattdessen setzen sie einen Kraftsensor ein, welcher über einen erweiterten Bezugspunkt den Winkel misst, ohne einer Biegung ausgesetzt zu sein.

Das Bewegen einzelner Finger von Handexoskeletten kann durch Kraftsensoren an jeder Fingerspitze erfolgen. So setzt das Exoskelett iHandRehab [24] zur Erfassung eines Fingers ein Bauelement aus drei Kraftsensoren und einer Metallplatte ein. Dies soll die Messung zuverlässiger machen, da das Gewebe am Kontaktpunkt der Hand weich ist. Zusätzlich verfügt das Design von Imtiaz et al. [9] auf den Fingerspitzen des Skeletts selbst über Kraftsensoren. Bei Kontakt mit einem Objekt kann damit die Bewegung des Fingers jeweils automatisch stoppen. Das Konzept HANDEXOS [25] beschreibt für jeden Finger ein Modul, welches für die einzelnen Fingerglieder jeweils einen Kraftsensor einsetzt. Dies verbessert die Bewegungsfreiheit, die der Roboter bietet. Durch dazukommende Aktuatoren nimmt jedoch ebenso die Dimension des Exoskeletts zu.

3.2 Drehmomentsensoren

Drehmomentsensoren (engl. torque sensor) finden am häufigsten Anwendung in LLEs. Hier werden sie meistens in Verbindung mit entsprechenden Drehmomentaktuatoren an Gelenken des Exoskeletts verwendet [7].

Die Sensoren bestehen meist aus einem äußeren und einem inneren Ring. Der innere Ring ist dabei so mit einem entsprechenden Bauteil verbunden, dass das ausgeübte Drehmoment auf den Ring übertragen wird. Bei dem Bauteil können unterschiedliche Ansätze verwendet werden. So wird hierzu oft eine Achse verwendet, welche direkt mit dem Sensor verbunden wird [21]. Toxiri et. al [26] verwenden in ihrem Design ein elastisches Seil, welches über mehrere Bolzen eine Kraft übertragen kann, die sich in das Drehmoment übersetzen lässt. Das ausgeübte Drehmoment kann anschließend zwischen den beiden Ringen aufgenommen werden. In vielen Sensoren sind hierzu Dehnungsmessstreifen eingebaut, welche die erzeugte Belastung messen. Beispiele dazu sind die TRT-200 Sensoren in Schabowsky et al. [27] oder die Drehmomentsensoren im Design von Kim et al. [21] (siehe Abb. 3).

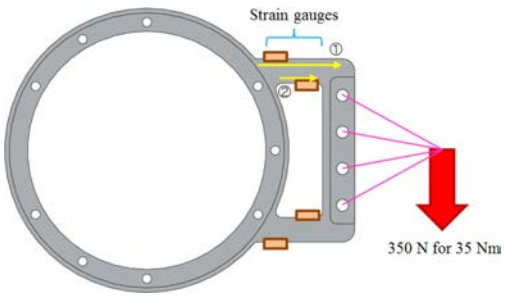


Abb. 3. Implementierung eines Drehmomentsensors unter Verwendung von Dehnungsmessstreifen aus [21]. Die Streifen messen das auf den äußeren Ring ausgeübte Drehmoment.

Ein von der Norm abweichender Sensor wird im Design von Gabert und Lenzi vorgestellt [28]. Zur Messung von Krafteinwirkung und Drehmoment benutzen sie zwei Magnete und Hall-Sensoren, welche zur Messung von Magnetfeldern eingesetzt werden. Bei Deformation des Bauteils bewegen sich die angebrachten Magneten, was eine Veränderung des Magnetfeldes mit sich bringt (siehe Abb. 4). Diese Veränderung wird über die Hall-Sensoren wahrgenommen und es wird eine Spannung erzeugt, welche sich in die ausgeübte Kraft beziehungsweise das erzeugte Drehmoment übersetzen lässt.

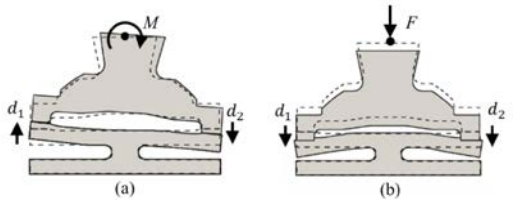


Abb. 4. Deformationen des Adapters resultieren durch Krafteinwirkungen aus [28]. Hall-Sensoren messen Veränderungen der Magentfelder an d_1 und d_2.

Drehmomentsensoren in aktiven Exoskeletten werden primär zwischen einem entsprechenden Aktuator, wie einem Motor, und dem Verbindungsstück zum Nutzer eingebaut. Damit kann zum einen der vom Aktuator erzeugte Output direkt gemessen werden. Zum anderen ist es gleichzeitig weiterhin möglich, die Intention des Nutzers am jeweiligen Gelenk zu quantifizieren und zu messen.

Das EXO-UL8 ist ein ULE, welches die Rehabilitation der Bewegung beider Arme nach einem Schlaganfall unterstützt [29]. Hierzu wurden in beiden Exoskelett-Armen jeweils drei FT-Sensoren (ATI mini 40) verwendet. Diese befinden sich am Oberarm, Unterarm und am Handgelenk an Kontaktpunkten zwischen Skelett und dem Nutzer. Diese nehmen unter anderem das Drehmoment an dem entsprechenden Gelenk auf.

Auch in LLEs werden Drehmomentsensoren zur Messung des resultierenden Drehmoments in unterschiedlichen Gelenken verwendet. Yu et al. [30] nehmen im Design eines Hüft-Exoskeletts Bezug auf einen Drehmomentsensor, welcher direkt am Flansch, dem Verbindungsstück, des Aktuators angebracht ist. Ähnlich existieren auch Designs, in denen am Kniegelenk [21] oder am Fußgelenk [31] das vom Nutzer verursachte und vom Aktuator angewandte Drehmoment gemessen wird. Die Nutzerintention kann hierbei aus dem resultierenden Ergebnis durch ermittelt werden [32].

3.3 Trägheitsmesseinheit

Die Trägheitsmesseinheit (engl. inertial measurement unit, IMU) besteht aus zwei unterschiedlichen Sensortypen: einem Beschleunigungssensor und einem

Gyroskop [33]. Ein Gyroskop ist ein Messgerät, welches die Winkelgeschwindigkeit von Rotationen bestimmen kann. Es lassen sich zudem Anwendungen finden, in denen die IMU zusätzlich über einen Magnetometer verfügt [34]. Der Beschleunigungssensor misst lineare Beschleunigung und kann die durch die Erde ausgeübte Gravitationskraft entlang einer Richtung bestimmen. Mithilfe dieser und den Daten aus dem Gyroskop ist es demnach möglich, die Winkelposition des Sensors in Referenz zur Erdanziehungskraft zu erfassen [35].

Aufgrund der Menge an messbaren Datenwerten werden IMUs fast ausschließlich in LLEs eingesetzt. Hier übernehmen sie nämlich die Aufgabe der Gangphasenerkennung des Nutzers [7]. So lassen sich anhand der Daten, wie Winkel und Beschleunigung einzelner Beinpartien, effizient die Phasen des jeweiligen Gangzyklus ermitteln. Hierzu kommen oft andere Sensoren wie Kraftsensoren hinzu, dessen Messdaten mit denen der IMUs kombiniert werden [36, 37, 38]. Damit steigt die Genauigkeit der Daten und es kann eine bessere Einschätzung der Gangphase erfolgen.

So existieren Anwendungen, in denen die Gangphasenerkennung bereits durch die Verwendung von IMUs abgedeckt ist. Susanto et al. [33] haben zwei IMUs an den beiden Knien des Exoskeletts eingesetzt. Sie übergeben den gemessenen Steigungswinkel des Knies an die Steuereinheit. Diese bestimmt unter Verwendung eines neuronalen Netzwerkes eine einfache Einteilung der Gangphase während des Laufens. Das durch Vakuum-Aktuatoren gesteuerte Knie-Exoskelett von Zhang et al. [39] integriert insgesamt vier IMUs. Für beide Beine jeweils ein Sensor über und ein Sensor unter dem Knie. Die aufgenommene Beschleunigung und die Orientierung werden für die Ermittlung der Gangphase eingesetzt.

In vielen Anwendungen wird allerdings hierfür eine Kombination aus mehreren Sensortypen angewendet. So kann die Bestimmung der Phase mit erhöhter Sicherheit oder einer größeren Anzahl an Zyklusunterteilungen erfolgen. Das LLE Gamma prototype [40] nutzt ebenfalls vier IMUs, welche an Schienbein und Oberschenkel integriert sind. Hier unterscheidet das System sich jedoch insofern, dass zusätzliche Sensoren in den Schuhen die Kraftverteilung der Füße mit dem Boden messen. Der Gangzyklus lässt sich aus den Daten beider Sensorgruppen in sechs Phasen unterteilen. Das vorgestellte Exoskelett von Aguirre-Ollinger und Yu [41] besitzt zwei IMUs, welche sich jeweils hinter den Oberschenkeln befinden. Diese nehmen Daten zur Rotation der Beine auf. Der Unterschied zwischen den Rotationen wird zusammen mit den Messwerten der Encoder an den Kniegelenken zur Bestimmung der Phase verrechnet.

Tan et al. [42] beschäftigen sich in ihrem Exoskelett-Design mit einer Gangphasenerkennung, welche durch einen gestörten Rhythmus beim Gehen des Nutzers erschwert wird. An den Schuhlaschen beziehungsweise den Schuhzungen des Exoskeletts wurden für jedes Bein eine IMU (LPMS-B2, LP-RESEARCH Inc., Japan) verwendet. Die aufgenommenen kinematischen Daten der Füße werden zur Bestimmung der Gangphase eingesetzt. Im Sensorsystem von Yue et al. [36] erfolgt die Gangphasenerkennung hingegen primär durch Kraftsensoren in den Sohlen. Beide eingebauten IMUs sind außen an den Sohlen angebracht. Sie

können die Winkelgeschwindigkeit der jeweiligen Sohle messen und werden dazu verwendet, die momentane Bodenbeschaffenheit aus vier definierten Kategorien zu ermitteln. Diese Kategorisierung hilft zusätzlich bei der Gangphasenermittlung.

Eine untergeordnete Rolle spielt die IMU auch im Design eines Exoskelett-Steuersystems von Vantilt et al. [43]. Hier wird eine IMU in der Region der Hüfte platziert, welche zusätzlich Daten zur Beschleunigung und Rotation misst. Die Daten sollen mit den Messungen der anderen verwendeten Sensoren kombiniert werden, sodass die momentane Orientation des Exoskeletts in Weltkoordinaten abgeschätzt werden kann. Hier findet also keine Gangphasenerkennung statt.

3.4 Encoder

Bei Encodern handelt es sich um eine Gruppe an Sensoren, die die Bewegung von mechanischen Teilen in ein analoges oder digitales Signal kodieren (engl. encode). Dieses wird an eine Kontrolleinheit weitergeleitet, welche das Signal übersetzt und weiterverarbeitet.

Encoder lassen sich ihrer Funktionsweise nach in unterschiedliche Klassen kategorisieren. So existieren zwei Obergruppen an Encodern. Das sind zum einen lineare Encoder, welche Änderungen entlang einer Achse messen können. Die andere Gruppe besteht aus rotierenden Encodern. Diese messen im Gegensatz zu den linearen Encodern Änderungen in einer Rotation. In Exoskeletten finden sich hauptsächlich Anwendungen von rotierenden Encodern. Diese werden entweder in die Motoren der Aktuatoren eingebaut, um diese präziser steuern zu können, oder aber sie befinden sich an Teilen des Exoskeletts, um dort die Winkel der Gelenke bestimmen zu können [6]. Für die hohe Präzision der Sensoren ist eine genaue Installation dieser vorausgesetzt. Aufgrund dessen werden sie nicht an Körperteilen des Nutzers angebracht.

Die Sensoren lassen sich allerdings noch spezifischer unterteilen. So bezeichnet man den Typ des Encoders beispielweise abhängig von deren Ausgabedaten als entweder absolut oder inkrementell [7]. Absolute Encoder geben in ihrem Signal ihre aktuelle Position aus, was in einem rotierenden Encoder die Winkelposition ist. Inkrementelle Encoder hingegen können nur Änderungen in ihrer Position wahrnehmen und diese in einer Rotation als Winkelgeschwindigkeit ausgeben. Zudem lassen sich die Sensoren bezüglich ihrer Vorgehensweise bei der Datenerhebung kategorisieren. Mechanische rotierende Encoder verwenden leitende, entlang der Achse rotierende Ringe, welche mithilfe einer Spannung durch ihre Hohlräume einen binären Code generieren. Optische Encoder benutzen mehrere Lichtquellen und entsprechende Sensoren. Das Unterbrechen unterschiedlicher Lichtimpulse durch zirkuläre Platten bei einer Rotation der Achse kann schließlich als Signal des Encoders codiert werden. Bei Verwendung eines magnetischen Encoders wird ein Magnetfeld an der Achse platziert. Rotationen der Achse verändern die Auswirkungen des Magnetfeldes auf den Sensor, welcher daraus ein analoges beziehungsweise digitales Signal erzeugt.

Hsieh et al. präsentieren ein Exoskelett für die Rehabilitation der Schultern [44]. Hierbei wird ein linearer Encoder in den Aktuator eingesetzt. So kann die durch den Motor resultierende Verschiebung hinter der Feder präzise entnommen werden. Ähnlich wird das Problem der Präzision bei elastischen Bauteilen von Jarrett und McDaid gelöst [45]. Sie konzipieren einen Aktuator für die Einbindung in ein ULE zur Bewegung des Ellenbogens. Dabei benutzen sie verformbare Elastomere. Die tatsächlich resultierende Winkelposition wird hier durch einen rotierenden Encoder gemessen. Encoder werden in unterschiedlichen Multi-Sensor-Systemen eingesetzt. So ergänzen Elnady et al. in ihrem Design eines BCI-getriebenen Arm-Exoskeletts [46] ihr EEG-Sensorsystem um einen zusätzlichen Encoder zum Ermitteln der Gelenkposition.

Gelenke eines LLEs lassen ebenso mithilfe von Encodern erfassen. Das dargestellte Exoskelett von Nomura et al. [47] setzt insgesamt vier rotierende Encoder ein, jeweils zwei für Kniegelenk und Hüftgelenk des Roboters. Der Aktuator eines Knie-Exoskeletts von Chen et al. [48] wird zusammen mit einem rotierenden inkrementellen Encoder implementiert. Dieser misst die Rotationsgeschwindigkeit im Gelenk des Skeletts. Vantilt et al. [43] präsentiert ein Aktuator-Design eines Exoskeletts, welches sowohl magnetische als auch optische Encoder einsetzt. Dieser Aktuator wird für Hüft-, Knie- und Knöchelgelenk verwendet und besitzt jeweils einen absoluten magnetischen Encoder und einen inkrementellen optischen Encoder.

Das Hand-Exoskelett von Ben-Tzvi et al. [49] setzt drei absolute magentische Encoder für jeden Finger ein. Die Konstruktion erlaubt es, die Form und Position der Finger mit einem Fehler von unter 0,8% der maximalen Bewegungsfreiheit zu bestimmen. Aubin et al. legt das Design eines Exoskeletts zur Rehabilitation des Daumens dar [50]. Zur direkten Bestimmung der Winkel in den unteren beiden Gelenken des Daumen werden hier optische Encoder eingesetzt.

3.5 EMG

Die Elektromyografie (EMG) ist eine Methode, bei der elektrische Signale bei der Aktivität ausgewählter Muskelpartien aufgezeichnet werden. Wenn die Muskelzellen der Skelettmuskulatur erregt werden, versuchen die EMG-Sensoren das emittierte elektrische Potenzial aufzunehmen [7].

Man unterscheidet bei der EMG zwischen zwei Methoden, der intramuskulären und der Oberflächen-EMG. Die intramuskuläre EMG benutzt zur Signalmessung eine Nadel, die in das Muskelgewebe eingeführt wird. Dieses Verfahren ist nützlich darin, möglichst exakt Daten zu einzelnen Muskeln zu erfassen. Da es sich bei der Methode aber um ein medizinisch invasives Verfahren handelt, wird diese Methode im Zusammenhang mit Exoskeletten nicht angewendet [7].

Das Oberflächen-EMG (engl. surface EMG, sEMG) verwendet hingegen Elektroden, welche auf der Haut im Bereich des betroffenen Muskels platziert werden. Im Vergleich zur intramuskulären Alternative birgt diese Methode jedoch Nachteile. So müssen die Elektroden bei jedem Einsatz des Exoskeletts neu und möglichst genau platziert werden. Das Signal ist beispielsweise auch abhängig von äußeren Einflüssen, wie der Hautfeuchtigkeit und der Muskelgröße des Trägers

[16]. Die größere Entfernung wirkt sich zudem negativ auf das aufgenommene Signal aus. Dennoch ist sEMG eine intuitive und effektive Methode zur Analyse der Intention des Nutzers.

In ULEs lassen sich einige Anwendungen von sEMG-Sensoren finden, bei denen das Skelett jedoch stationär angebracht ist und somit nicht mobil verwendet werden kann. Ein mobiles Beispiel, welches sEMG-Sensoren einbezieht, ist das Exoskelett-Design von Copaci et al. [51]. Hier werden zwei sEMG-Elektroden am Bizeps und eine am Schulterblatt platziert. Zusammen mit FSR-Sensoren kann die vom Nutzer angestrebte Bewegung des Ellenbogengelenks approximiert und unterstützt werden.

Chandrapal et al. stellen ein LLE vor, welches die Belastung am Knie des Nutzers reduzieren soll [52]. Neben einem Encoder zum Messen des Kniewinkels verwenden sie fünf sEMG-Sensoren, positioniert an den Muskeln des Oberschenkels, welche am meisten zur Beugung und Streckung des Kniegelenks beitragen. Das Design von Aguilar-Sierra et al. [53] platziert an jedem Bein acht sEMG-Sensoren. Auf Basis dieser breiten Überdeckung an Muskelpartien soll die Intention des Nutzers beim Gehen möglichst umfassend beschrieben werden. Die EMG-Daten werden dafür verwendet, die Gangphase anhand von Mustern in den Messwerten zu erkennen und darauf basierend die Aktuatoren anzusteuern. Vorgestellt wurde bereits der Nachteil von sEMG, dass die Elektroden bei jeder Verwendung neu platziert werden müssen und diese möglichst die gleichen Positionen über den Muskelpartien einnehmen sollen. Um diesem Problem zu entgehen, verwendet das Knie-Exoskelett von Moon et al. einen Gurt, welcher oberhalb des Knöchels festgemacht wird [54]. Dieser Gurt besitzt sEMG-Sensoren an der Innenseite und misst somit die Muskelsignale bei jeder Verwendung möglichst an gleicher Position.

Maestro ist ein durch sEMG-Sensoren betriebenes Hand-Exoskelett für Patienten bei Rückenmarksverletzungen [55]. Dabei werden unterschiedliche Handposen nach ihrem EMG-Muster in drei Gruppen unterteilt. Es werden drei sEMG-Sensoren an Hand und Unterarm des Nutzers angebracht. Bei der Platzierung wird auf das akkurate Messen der Beugung und Streckung der Finger und die des Daumens abgezielt. Eine andere Methode der Rehabilitation findet sich in Li et al. [56]. Hier wird sich auf die bisherige Forschungen in der Rehabilitation von Schlaganfallpatienten durch Spiegeltherapie gestützt [57]. Dabei werden sechs sEMG-Sensoren an Hand und Unterarm des gesunden Arms positioniert. Diese nehmen die Signale bei Bewegung der Hand auf und leiten diese an das Gerüst des Exoskeletts, welches sich an der betroffenen Hand befindet.

3.6 EEG

Die Elektroenzephalografie (EEG) beschäftigt sich, ähnlich wie die EMG, mit dem Messen von elektrischen Signalen des menschlichen Körpers zur Intentionserkennung. Hierbei werden die Summen der von den Neuronen weitergeleiteten elektrischen Potenziale außerhalb des Schädels beim Bewegen der Muskeln aufgezeichnet und verarbeitet [16]. Ebenfalls wie bei der EMG existieren hier

invasive und nicht-invasive Methoden der Signalerfassung. Im Bereich der Exoskelette wird allerdings generell nicht-invasiv geforscht. Dabei werden Elektroden außerhalb der Kopfhaut angebracht, oft in Form einer Kappe, welche vom Nutzer aufgesetzt werden kann. Bezüglich der Platzierung existieren internationale Standards, wie das 10-20-System, auf welches in den meisten Anwendungen zurückgegriffen wird [58] (siehe Abb. 5). Die Zahlen beziehen sich dabei auf die prozentuale Entfernung einzelner Elektroden auf der Kopfhaut voneinander.

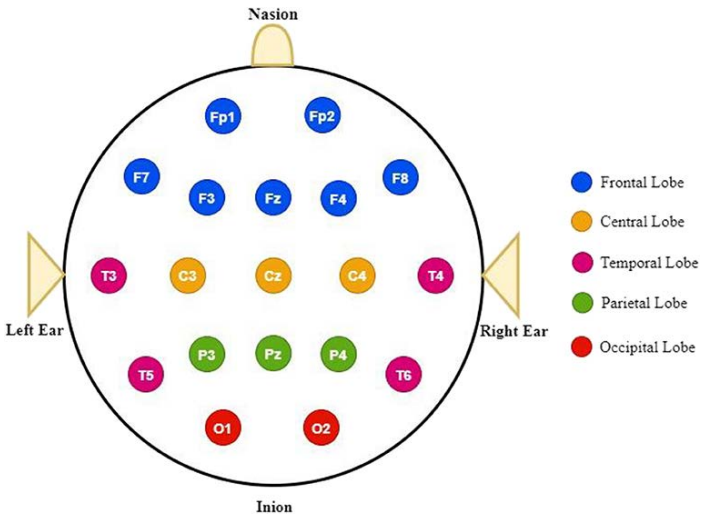


Abb. 5. 10-20-System für eine effektive Aufteilung der Elektroden zur EEG-Messung aus [58]. Üblich sind auch Elektroden hinter den Ohren, bezeichnet als $A1$ und $A2$.

Bei Nutzung von EEG-Signalen in der Steuerung von Exoskeletten werden sogenannte Gehirn-Computer-Schnittstellen (engl. Brain-Computer Interface, BCI) verwendet. BCIs sind Teil der Steuerung und umfassen die Aufnahme der elektrischen Signale von den EEG-Sensoren, die Weiterverarbeitung dieser und das Übersetzen in Steuersignale für die Aktuatoren [58].

Das Design eines Arm-Exoskeletts von Elnady et al. richtet sich an die Rehabilitation einfacher Armbewegungen bei Schlaganfallpatienten [46]. Sie legen dem Nutzer dafür ein Headset um den Kopf, welches mit EEG-Sensoren ausgestattet ist. Das eingesetzte BCI-System verwendet diese Signale, um den momentanen Zustand einer von zwei Klassen zuzuordnen: 0 für ruhend und 1 für eine Bewegungsintention. Abgeleitet von dieser Klassifizierung wird der Exoskelett-Arm durch Aktuatoren geöffnet oder aber eine FES-Einheit (funktionale Elektrostimulation) wird angesprochen, welche dem Patienten über Muskelstimulation mit elektrischen Impulsen hilft.

Choi et al. [59] befassen sich mit der Forschung einer BCI, welche mithilfe von EEG-Sensoren drei Bewegungzustände des Nutzers unterscheiden kann: Aufstehen, Vorwärtsgehen und Hinsetzen. Sie verwenden hierfür eine EEG-Haube mit 31 Elektroden, welche nach dem bereits benannten 10-20 System positioniert werden. Ein weiteres BCI zur Steuerung eines LLEs stellen Ferrero et al. vor [60]. Auch für dieses BCI wird eine Haube für die Anbringung der 32 EEG-Sensoren eingesetzt. Hierbei wird das 10-10 System angewendet. Dabei handelt es sich ebenfalls um einen internationalen Standard, welcher mit einer größeren Anzahl an näher aneinander platzierten Sensoren die Auflösung der Daten verbessern soll. Zur Zustandsberechnung umfasst das Design zudem vier Elektroden im Bereich der Augen, welche elektrookulographische Signale aufzeichnen, also Bewegungen der Augen. EMG in Exoskeletten wird zudem öfters zusammen mit EEG kombiniert. Li et al. [61] präsentieren ein Exoskelett, welches das Treppensteigen erleichtern soll. Dazu erfolgt eine Gangphasenbestimmung, welche allein auf intrinsische Signale des Nutzers erfolgen kann und keine mechanischen Sensoren verwendet. Hier werden die Messdaten aus der EEG-Haube und zwei EMG-Sensoren an den Unterarmen bei Nutzung des Geländers der Treppe kombiniert.

Auch für Handexoskelette eignet sich die Verwendung von Technologie der EEG. Die komplizierten Bewegungen der einzelnen Finger und Handgesten lassen sich mithilfe der EEG-Muster in Gruppen klassifizieren. So konnte mithilfe des Handexoskeletts mano [62] von der computergestuerten Beugung und Streckung der Finger in vier Aufgaben auf dabei gemessene EEG Signale zurückgeschlossen werden.

4 Schlussfolgerung

In dieser Review wurden die Ziele der Sensorik in Exoskeletten vorgestellt. Dabei wurde näher auf die Methode der Gangphasenerkennung eingegangen, welche bei dem Einsatz von Sensoren in LLEs eine entscheidende Rolle spielt. Zudem wurde die Sensordatenfusion aufgegriffen und erläutert, die bei der Nutzung mehrerer Sensoren in einem System unumgänglich ist.

Die Arbeit stellte eine Reihe von Sensortypen vor, auf welche am häufigsten in der Anwendung von Exoskeletten zugegriffen wird. Hierbei wurden die Sensortypen in ihrer Funktionsweise dargestellt und für alle Typen wurde eine breite Menge an Beispielen in Exoskelett-Designs vorgestellt. Bei diesen Exoskeletten handelt es sich um Anwendungen aus sowohl medizinischen als auch nicht-medizinischen Bereichen. Zudem wurden Modelle aus der Praxis wie auch aus der Forschung verwendet. Hierbei wurden Beispiele vorgestellt, die in ihrer Gesamtheit einen möglichst umfassenden Einblick in potenziellen Anwendungen dieser Sensoren bieten sollen.

Die Schwierigkeiten beim Entwerfen eines Exoskeletts und des damit zusammenhängenden Sensorsystems sind komplex und von Fall zu Fall unterschiedlich zu bewältigen. Eine direkte Anwendung der dargestellten Systeme in den Beispielen ist demnach oft nicht die effizienteste Lösung. Aus den Ergebnissen lässt sich

allerdings ableiten, welche Methoden zuvor funktioniert haben und es lassen sich neue Zusammensetzungen und Alternativen für Sensorsysteme bilden. Mit der weiter fortschreitenden technologischen Entwicklung lassen sich zudem in Zukunft neue Sensortypen entwickeln, welche bisherige Schwachstellen nichtig machen und das Potenzial von Exoskeletten steigern werden.

Zusammenfassung

[1] Dana Terrazas-Rodas et al. "Control Systems in Lower-limb Exoskeletons for Rehabilitation and/or Assistance: A Brief Review". In: *2022 International Conference on Smart Systems and Power Management (IC2SPM)*. Nov. 2022, pp. 117–123. DOI: 10.1109/IC2SPM56638.2022.9988843.

[2] Usama Umar et al. "Design and Simulation of Lower-Limb Exoskeleton to Assist Paraplegic People in Walking". In: *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*. Vol. 1. ISSN: 2576-3555. May 2022, pp. 855–860. DOI: 10.1109/CoDIT55151.2022.9804158.

[3] Tiaan du Plessis, Karim Djouani, and Christiaan Oosthuizen. "A Review of Active Hand Exoskeletons for Rehabilitation and Assistance". In: *Robotics* 10.1 (Mar. 2021), p. 40. DOI: 10.3390/robotics10010040.

[4] Sunwook Kim et al. "Potential of Exoskeleton Technologies to Enhance Safety, Health, and Performance in Construction: Industry Perspectives and Future Research Directions". In: *IISE Transactions on Occupational Ergonomics and Human Factors* 7.3-4 (Oct. 2019), pp. 185–191. ISSN: 2472-5838. DOI: 10.1080/24725838.2018.1561557. URL: https://doi.org/10.1080/24725838.2018.1561557 (visited on 08/26/2023).

[5] Man Bok Hong, Gwang Tae Kim, and Yeo Hun Yoon. "ACE-Ankle: A Novel Sensorized RCM (Remote-Center-of-Motion) Ankle Mechanism for Military Purpose Exoskeleton". en. In: *Robotica* 37.12 (Dec. 2019), pp. 2209–2228. ISSN: 0263-5747, 1469-8668. DOI: 10.1017/S0263574719000845. URL: https://www.cambridge.org/core/journals/robotica/article/aceankle-a-novel-sensorized-rcm-remotecenterofmotion-ankle-mechanism-for-military-purpose-exoskeleton/5660F1B670E90FA5B66634D12F6C9C70 (visited on 08/08/2023).

[6] Canjun Yang et al. "Current developments of robotic hip exoskeleton toward sensing, decision, and actuation: A review". en. In: *Wearable Technologies* 3 (2022), e15. ISSN: 2631-7176. DOI: 10.1017/wtc.2022.11. URL: https://www.cambridge.org/core/journals/wearable-technologies/article/current-developments-of-robotic-hip-exoskeleton-toward-sensing-decision-and-actuation-a-review/8557C3F7209A9E7EE45BA248F0E68037 (visited on 05/24/2023).

[7] Monica Tiboni et al. "Sensors and Actuation Technologies in Exoskeletons: A Review". In: *Sensors* 22.3 (Jan. 2022), p. 884. DOI: 10.3390/s22030884.

[8] Adam Zoss and H. Kazerooni. "Design of an electrically actuated lower extremity exoskeleton". In: *Advanced Robotics* 20.9 (Jan. 2006), pp. 967–988. ISSN: 0169-1864. DOI: 10.1163/156855306778394030. URL: https://doi.org/10.1163/156855306778394030 (visited on 07/07/2023).

[9] Muhammad Saad bin Imtiaz et al. "Design of Portable Exoskeleton Forearm for Rehabilitation of Monoparesis Patients Using Tendon Flexion Sensing Mechanism for Health Care Applications". en. In: *Electronics* 10.11 (Jan. 2021), p. 1279. ISSN: 2079-9292. DOI: 10.3390/electronics10111279. URL: https://www.mdpi.com/2079-9292/10/11/1279 (visited on 06/19/2023).

[10] Di Shi et al. "A Review on Lower Limb Rehabilitation Exoskeleton Robots". In: *Chinese Journal of Mechanical Engineering* 32.1 (Aug. 2019), p. 74. ISSN: 2192-8258. DOI: 10.1186/s10033-019-0389-8. URL: https://doi.org/10.1186/s10033-019-0389-8 (visited on 05/30/2023).

[11] Domen Novak and Robert Riener. "A survey of sensor fusion methods in wearable robotics". en. In: *Robotics and Autonomous Systems*. Wearable Robotics 73 (Nov. 2015), pp. 155–170. ISSN: 0921-8890. DOI: 10.1016/j.robot.2014.08.012. URL: https://www.sciencedirect.com/science/article/pii/S0921889014001705 (visited on 05/30/2023).

[12] K. Kiguchi et al. "An exoskeletal robot for human elbow motion support-sensor fusion, adaptation, and control". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 31.3 (June 2001), pp. 353–361. ISSN: 1941-0492. DOI: 10.1109/3477.931520.

[13] A. S. Sadun, J. Jalani, and J. A. Sukor. "Force Sensing Resistor (FSR): a brief overview and the low-cost sensor for active compliance control". In: *SPIE Proceedings*. Ed. by Xudong Jiang et al. SPIE, July 2016. DOI: 10.1117/12.2242950.

[14] Jonas Beil, Gernot Perner, and Tamim Asfour. "Design and control of the lower limb exoskeleton KIT-EXO-1". In: *2015 IEEE International Conference on Rehabilitation Robotics (ICORR)*. ISSN: 1945-7901. Aug. 2015, pp. 119–124. DOI: 10.1109/ICORR.2015.7281186.

[15] Hyundo Choi et al. "Compact Hip-Force Sensor for a Gait-Assistance Exoskeleton System". en. In: *Sensors* 18.2 (Feb. 2018), p. 566. ISSN: 1424-8220. DOI: 10.3390/s18020566. URL: https://www.mdpi.com/1424-8220/18/2/566 (visited on 07/14/2023).

[16] Slávka Neťuková et al. "Lower Limb Exoskeleton Sensors: State-of-the-Art". eng. In: *Sensors (Basel, Switzerland)* 22.23 (Nov. 2022), p. 9091. ISSN: 1424-8220. DOI: 10.3390/s22239091. URL: https://europepmc.org/articles/PMC9738474 (visited on 05/24/2023).

[17] Pilwon Heo and Jung Kim. "Power-Assistive Finger Exoskeleton With a Palmar Opening at the Fingerpad". In: *IEEE Transactions on Biomedical Engineering* 61.11 (Nov. 2014), pp. 2688–2697. ISSN: 1558-2531. DOI: 10.1109/TBME.2014.2325948.

[18] Xin Wang et al. "Multi-connection load compensation and load information calculation for an upper-limb exoskeleton based on a six-axis force/torque

sensor". en. In: *International Journal of Advanced Robotic Systems* 16.4 (July 2019), p. 1729881419863186. ISSN: 1729-8806. DOI: 10.1177/1729881419863186. URL: https://doi.org/10.1177/1729881419863186 (visited on 07/07/2023).

[19] Chang Liu et al. "Functional Evaluation of a Force Sensor-Controlled Upper-Limb Power-Assisted Exoskeleton with High Backdrivability". en. In: *Sensors* 20.21 (Jan. 2020), p. 6379. ISSN: 1424-8220. DOI: 10.3390/s20216379. URL: https://www.mdpi.com/1424-8220/20/21/6379 (visited on 07/07/2023).

[20] I.P.I. Pappas et al. "A reliable gait phase detection system". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 9.2 (June 2001), pp. 113–125. ISSN: 1558-0210. DOI: 10.1109/7333.928571.

[21] Jung-Hoon Kim et al. "Design of a Knee Exoskeleton Using Foot Pressure and Knee Torque Sensors". en. In: *International Journal of Advanced Robotic Systems* 12.8 (Aug. 2015), p. 112. ISSN: 1729-8806. DOI: 10.5772/60782. URL: https://doi.org/10.5772/60782 (visited on 06/19/2023).

[22] Wei Yang et al. "Lower Limb Exoskeleton Gait Planning Based on Crutch and Human-Machine Foot Combined Center of Pressure". en. In: *Sensors* 20.24 (Jan. 2020), p. 7216. ISSN: 1424-8220. DOI: 10.3390/s20247216. URL: https://www.mdpi.com/1424-8220/20/24/7216 (visited on 07/07/2023).

[23] Feng Chen et al. "WPAL for Enhancing Human Strength and Endurance during Walking". In: *2007 International Conference on Information Acquisition*. July 2007, pp. 487–491. DOI: 10.1109/ICIA.2007.4295782.

[24] Jiting Li et al. "iHandRehab: An interactive hand exoskeleton for active and passive rehabilitation". In: *2011 IEEE International Conference on Rehabilitation Robotics*. ISSN: 1945-7901. June 2011, pp. 1–6. DOI: 10.1109/ICORR.2011.5975387.

[25] Azzurra Chiri et al. "Mechatronic Design and Characterization of the Index Finger Module of a Hand Exoskeleton for Post-Stroke Rehabilitation". In: *IEEE/ASME Transactions on Mechatronics* 17.5 (Oct. 2012), pp. 884–894. ISSN: 1941-014X. DOI: 10.1109/TMECH.2011.2144614.

[26] Stefano Toxiri et al. "A Parallel-Elastic Actuator for a Torque-Controlled Back-Support Exoskeleton". In: *IEEE Robotics and Automation Letters* 3.1 (Jan. 2018), pp. 492–499. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2768120.

[27] Christopher N. Schabowsky et al. "Development and pilot testing of HEXORR: Hand EXOskeleton Rehabilitation Robot". en. In: *Journal of NeuroEngineering and Rehabilitation* 7.1 (July 2010), p. 36. ISSN: 1743-0003. DOI: 10.1186/1743-0003-7-36. URL: https://doi.org/10.1186/1743-0003-7-36 (visited on 07/27/2023).

[28] Lukas Gabert and Tommaso Lenzi. "Instrumented Pyramid Adapter for Amputee Gait Analysis and Powered Prosthesis Control". In: *IEEE Sensors Journal* 19.18 (Sept. 2019), pp. 8272–8282. ISSN: 1558-1748. DOI: 10.1109/JSEN.2019.2920179.

[29] Yang Shen et al. “Asymmetric Dual Arm Approach For Post Stroke Recovery Of Motor Functions Utilizing The EXO-UL8 Exoskeleton System: A Pilot Study”. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. ISSN: 1558-4615. July 2018, pp. 1701–1707. DOI: 10.1109/EMBC.2018.8512665.

[30] Shuangyue Yu et al. “Quasi-Direct Drive Actuation for a Lightweight Hip Exoskeleton With High Backdrivability and High Bandwidth”. In: *IEEE/ASME Transactions on Mechatronics* 25.4 (Aug. 2020), pp. 1794–1802. ISSN: 1941-014X. DOI: 10.1109/TMECH.2020.2995134.

[31] Greg Orekhov, Jason Luque, and Zachary F. Lerner. “Closing the Loop on Exoskeleton Motor Controllers: Benefits of Regression-Based Open-Loop Control”. In: *IEEE Robotics and Automation Letters* 5.4 (Oct. 2020), pp. 6025–6032. ISSN: 2377-3766. DOI: 10.1109/LRA.2020.3011370.

[32] Beomsoo Hwang and Doyoung Jeon. “A Method to Accurately Estimate the Muscular Torques of Human Wearing Exoskeletons by Torque Sensors”. en. In: *Sensors* 15.4 (Apr. 2015), pp. 8337–8357. ISSN: 1424-8220. DOI: 10.3390/s150408337. URL: https://www.mdpi.com/1424-8220/15/4/8337 (visited on 07/28/2023).

[33] Susanto Susanto et al. “Real-Time Identification of Knee Joint Walking Gait as Preliminary Signal for Developing Lower Limb Exoskeleton”. en. In: *Electronics* 10.17 (Jan. 2021), p. 2117. ISSN: 2079-9292. DOI: 10.3390/electronics10172117. URL: https://www.mdpi.com/2079-9292/10/17/2117 (visited on 08/01/2023).

[34] Maja Goršič et al. “Online Phase Detection Using Wearable Sensors for Walking with a Robotic Prosthesis”. en. In: *Sensors* 14.2 (Feb. 2014), pp. 2776–2794. ISSN: 1424-8220. DOI: 10.3390/s140202776. URL: https://www.mdpi.com/1424-8220/14/2/2776 (visited on 08/01/2023).

[35] Zdzisław Kowalczuk and Tomasz Merta. “Evaluation of position estimation based on accelerometer data”. In: *2015 10th International Workshop on Robot Motion and Control (RoMoCo)*. July 2015, pp. 246–251. DOI: 10.1109/RoMoCo.2015.7219743.

[36] Chunfeng Yue et al. “Design and Performance Evaluation of a Wearable Sensing System for Lower-Limb Exoskeleton”. en. In: *Applied Bionics and Biomechanics* 2018 (Sept. 2018), e8610458. ISSN: 1176-2322. DOI: 10.1155/2018/8610458. URL: https://www.hindawi.com/journals/abb/2018/8610458/ (visited on 08/01/2023).

[37] Chao-Feng Chen et al. “Development and Hybrid Control of an Electrically Actuated Lower Limb Exoskeleton for Motion Assistance”. In: *IEEE Access* 7 (2019), pp. 169107–169122. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2953302.

[38] Taehoon Lee, Inwoo Kim, and Yoon Su Baek. “Design of a 2DoF Ankle Exoskeleton with a Polycentric Structure and a Bi-Directional Tendon-Driven Actuator Controlled Using a PID Neural Network”. en. In: *Actuators* 10.1 (Jan. 2021), p. 9. ISSN: 2076-0825. DOI: 10.3390/act10010009. URL: https://www.mdpi.com/2076-0825/10/1/9 (visited on 08/01/2023).

[39] Liancun Zhang et al. “A Wearable Soft Knee Exoskeleton Using Vacuum-Actuated Rotary Actuator”. In: *IEEE Access* 8 (2020), pp. 61311–61326. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2983790.

[40] Christian Di Natali et al. “Pneumatic Quasi-Passive Actuation for Soft Assistive Lower Limbs Exoskeleton”. In: *Frontiers in Neurorobotics* 14 (2020). ISSN: 1662-5218. URL: https://www.frontiersin.org/articles/10.3389/fnbot.2020.00031 (visited on 08/01/2023).

[41] Gabriel Aguirre-Ollinger and Haoyong Yu. “Lower-Limb Exoskeleton With Variable-Structure Series Elastic Actuators: Phase-Synchronized Force Control for Gait Asymmetry Correction”. In: *IEEE Transactions on Robotics* 37.3 (June 2021), pp. 763–779. ISSN: 1941-0468. DOI: 10.1109/TRO.2020.3034017.

[42] Xiaowei Tan et al. “Cadence-Insensitive Soft Exoskeleton Design With Adaptive Gait State Detection and Iterative Force Control”. In: *IEEE Transactions on Automation Science and Engineering* 19.3 (July 2022), pp. 2108–2121. ISSN: 1558-3783. DOI: 10.1109/TASE.2021.3066403.

[43] Jonas Vantilt et al. “Model-based control for exoskeletons with series elastic actuators evaluated on sit-to-stand movements”. In: *Journal of Neuro-Engineering and Rehabilitation* 16.1 (June 2019), p. 65. ISSN: 1743-0003. DOI: 10.1186/s12984-019-0526-8. URL: https://doi.org/10.1186/s12984-019-0526-8 (visited on 08/01/2023).

[44] Hsiang-Chien Hsieh et al. “Design of a Parallel Actuated Exoskeleton for Adaptive and Safe Robotic Shoulder Rehabilitation”. In: *IEEE/ASME Transactions on Mechatronics* 22.5 (Oct. 2017), pp. 2034–2045. ISSN: 1941-014X. DOI: 10.1109/TMECH.2017.2717874.

[45] Christopher Jarrett and Andrew McDaid. “Modeling and Feasibility of an Elastomer-Based Series Elastic Actuator as a Haptic Interaction Sensor for Exoskeleton Robotics”. In: *IEEE/ASME Transactions on Mechatronics* 24.3 (June 2019), pp. 1325–1333. ISSN: 1941-014X. DOI: 10.1109/TMECH.2019.2906918.

[46] Ahmed Mohamed Elnady et al. “A Single-Session Preliminary Evaluation of an Affordable BCI-Controlled Arm Exoskeleton and Motor-Proprioception Platform”. In: *Frontiers in Human Neuroscience* 9 (2015). ISSN: 1662-5161. URL: https://www.frontiersin.org/articles/10.3389/fnhum.2015.00168 (visited on 08/05/2023).

[47] Shinnosuke Nomura et al. “Power Assist Control Based on Human Motion Estimation Using Motion Sensors for Powered Exoskeleton without Binding Legs”. en. In: *Applied Sciences* 9.1 (Jan. 2019), p. 164. ISSN: 2076-3417. DOI: 10.3390/app9010164. URL: https://www.mdpi.com/2076-3417/9/1/164 (visited on 08/08/2023).

[48] Bing Chen et al. “Design and characterization of a magneto-rheological series elastic actuator for a lower extremity exoskeleton”. en. In: *Smart Materials and Structures* 26.10 (Sept. 2017), p. 105008. ISSN: 0964-1726. DOI: 10.1088/1361-665X/aa8343. URL: https://dx.doi.org/10.1088/1361-665X/aa8343 (visited on 08/01/2023).

[49] Pinhas Ben-Tzvi and Zhou Ma. "Sensing and Force-Feedback Exoskeleton (SAFE) Robotic Glove". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 23.6 (Nov. 2015), pp. 992–1002. ISSN: 1558-0210. DOI: 10.1109/TNSRE.2014.2378171.

[50] Patrick Aubin et al. "A pediatric robotic thumb exoskeleton for at-home rehabilitation : The isolated orthosis for thumb actuation (IOTA)". In: *International Journal of Intelligent Computing and Cybernetics* 7.3 (Jan. 2014). Ed. by Dr Guilherme N. DeSouza, pp. 233–252. ISSN: 1756-378X. DOI: 10.1108/IJICC-10-2013-0043. URL: https://doi.org/10.1108/IJICC-10-2013-0043 (visited on 08/08/2023).

[51] Dorin Copaci et al. "A High-Level Control Algorithm Based on sEMG Signalling for an Elbow Joint SMA Exoskeleton". en. In: *Sensors* 18.8 (Aug. 2018), p. 2522. ISSN: 1424-8220. DOI: 10.3390/s18082522. URL: https://www.mdpi.com/1424-8220/18/8/2522 (visited on 08/03/2023).

[52] Mervin Chandrapal et al. "Preliminary Evaluation of Intelligent Intention Estimation Algorithms for an Actuated Lower-Limb Exoskeleton". en. In: *International Journal of Advanced Robotic Systems* 10.2 (Feb. 2013), p. 147. ISSN: 1729-8806. DOI: 10.5772/56063. URL: https://doi.org/10.5772/56063 (visited on 08/05/2023).

[53] Hipolito Aguilar-Sierra et al. "Design and control of hybrid actuation lower limb exoskeleton". en. In: *Advances in Mechanical Engineering* 7.6 (June 2015), p. 1687814015590988. ISSN: 1687-8132. DOI: 10.1177/1687814015590988. URL: https://doi.org/10.1177/1687814015590988 (visited on 08/05/2023).

[54] Dae-Hoon Moon, Donghan Kim, and Young-Dae Hong. "Intention Detection Using Physical Sensors and Electromyogram for a Single Leg Knee Exoskeleton". en. In: *Sensors* 19.20 (Jan. 2019), p. 4447. ISSN: 1424-8220. DOI: 10.3390/s19204447. URL: https://www.mdpi.com/1424-8220/19/20/4447 (visited on 08/05/2023).

[55] Youngmok Yun et al. "Maestro: An EMG-driven assistive hand exoskeleton for spinal cord injury patients". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 2904–2910. DOI: 10.1109/ICRA.2017.7989337.

[56] Chaoyang Li et al. "Real-Time Gait Event Detection for a Lower Extremity Exoskeleton Robot by Infrared Distance Sensors". In: *IEEE Sensors Journal* 21.23 (Dec. 2021), pp. 27116–27123. ISSN: 1558-1748. DOI: 10.1109/JSEN.2021.3111212.

[57] I. Emerson, J. Potgieter, and W.L. Xu. "Control implementation for an integrated robotic and virtual mirror therapy system for stroke rehabilitation". In: *2016 IEEE 14th International Workshop on Advanced Motion Control (AMC)*. ISSN: 1943-6580. Apr. 2016, pp. 479–485. DOI: 10.1109/AMC.2016.7496396.

[58] Mamunur Rashid et al. "Current Status, Challenges, and Possible Solutions of EEG-Based Brain-Computer Interface: A Comprehensive Review". In: *Frontiers in Neurorobotics* 14 (2020). ISSN: 1662-5218. URL: https:

//www.frontiersin.org/articles/10.3389/fnbot.2020.00025 (visited on 08/05/2023).

[59] Junhyuk Choi et al. "Developing a Motor Imagery-Based Real-Time Asynchronous Hybrid BCI Controller for a Lower-Limb Exoskeleton". en. In: *Sensors* 20.24 (Jan. 2020), p. 7309. ISSN: 1424-8220. DOI: 10.3390/s20247309. URL: https://www.mdpi.com/1424-8220/20/24/7309 (visited on 08/05/2023).

[60] Laura Ferrero et al. "A BMI Based on Motor Imagery and Attention for Commanding a Lower-Limb Robotic Exoskeleton: A Case Study". en. In: *Applied Sciences* 11.9 (Jan. 2021), p. 4106. ISSN: 2076-3417. DOI: 10.3390/app11094106. URL: https://www.mdpi.com/2076-3417/11/9/4106 (visited on 08/05/2023).

[61] Zhijun Li et al. "Hybrid Brain/Muscle Signals Powered Wearable Walking Exoskeleton Enhancing Motor Ability in Climbing Stairs Activity". In: *IEEE Transactions on Medical Robotics and Bionics* 1.4 (Nov. 2019), pp. 218–227. ISSN: 2576-3202. DOI: 10.1109/TMRB.2019.2949865.

[62] Luca Randazzo et al. "mano: A Wearable Hand Exoskeleton for Activities of Daily Living and Neurorehabilitation". In: *IEEE Robotics and Automation Letters* 3.1 (Jan. 2018), pp. 500–507. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2771329.

Actuation technology in Active Mobile Exoskeletons *

Yavuz Karaca[1]

Karlsruhe Institute of Technology, Karlsruhe 76131, Germany, uxfix@kit.edu

Abstract. Exoskeletons are witnessing an increase in commercialization as their utility in rehabilitation and industrial aid expands, driving the growth of the exoskeleton market. Creating an efficient and smoothly operating exoskeleton, involves optimizing numerous variables, one of the key ones being the selection of an appropriate actuator. This paper aims to explore the current state of actuation technology, focusing on the various motors, mechanical design, and actuation principles employed. By establishing key criteria, the trade-offs inherent in their design will be identified. Additionally, the significant role of biomimicry in exoskeleton development will be discussed.

Keywords: Actuators · Exoskeletons · Biomimicry · Mechatronic Devices · Wearable Devices

1 Introduction

Throughout the course of history, people have been engaging in physically demanding tasks alongside their daily lives. However, the execution of any physical activity can be impeded by factors such as injuries or the natural aging process. Moreover, even the most physically fit individuals have their limitations, as certain actions can surpass their capabilities. In response to this inherent vulnerability and restrictions in human nature, a solution has been devised by researchers and engineers in the form of an external wearable device known as an exoskeleton.

In the realm of zoology, the term "exoskeleton" pertains to any rigid external structure present in certain organisms [1], but in the context of this paper, it specifically pertains to exoskeletal robots. These innovative wearable devices are meticulously designed with a mechanical framework to imitate the anatomical structure of a limb or the particular body part that they are intended to enhance [2]. Hence, their applications span a broad spectrum – from enhancing worker performance and minimizing health complications to accelerating the recovery of individuals requiring medical assistance [3].

In industrial settings, exoskeletons effectively bolster worker strength during tasks involving long-distance walking or heavy lifting [4]. For instance, the

* Supported by organization Teco.

Berkeley Lower Extremity Exoskeleton, a pioneering human exoskeleton, demonstrated its capabilities at U.C. Berkeley by walking at an average speed of 0.9 m/s (2 mph) while carrying a 34 kg (75 lb) payload [5]. Another common industrial application involves enhancing workers' endurance by reducing the strain on their arms, allowing them to hold objects above their heads for extended periods (Fig. 1)[6].

In the medical field, exoskeletons play a vital role in assisting individuals with mobility impairments, enabling them to partially regain their functional independence [7]. They are also utilized for rehabilitation purposes, aiding in improving exercise effectiveness and shortening recovery durations [8]. Ekso Bionics specializes in developing bionic suits that provide power to both upper and lower body for such patients. Their exoskeletons have enabled patients with lower extremity disabilities to take over 180 million steps. They shared that their database has recorded over 3,000 individuals benefiting from the device in less than 2 years [9].

As of 2022, the global exoskeleton market reached a valuation of USD 334.5 million. This market is poised for significant revenue growth, projected at a compound annual growth rate (CAGR) of 16.9 percent, as indicated by the same report [10, 11]. This growth can be attributed to the increasing reliance on exoskeletons by a larger population.

One of these categorization criteria is based on the anatomical areas targeted by the exoskeletons. This encompasses upper limb exoskeletons (ULE), lower limb exoskeletons (LLE), full-body exoskeletons, as well as devices designed for specific anatomical regions. Classic examples for specific segment classes include hands and the trunk, each characterized by unique attributes and distinct optimization goals. For instance, precision is paramount for hand exoskeletons, while trunk exoskeletons focus on delivering a wide range of motion. [2]

Fig. 1. The "EXO-O1 Overhead Exoskeleton" manufactured by Hilti serves as an example of the Passive Upper Limb Exoskeleton (ULE) and the industrial application of exoskeletons. Here, the lack of sensors, electronics and motors is evident, as the system solely relies on the energy stored in its elastic binders to generate an upward force on the component connected to the elbows. Illustrations retrieved from [12].

Exoskeletons can also be categorized as either active or passive, depending on whether they utilize a power source. Passive exoskeletons rely solely on elastic materials, such as springs, to store and release energy as needed [13]. In some cases, they also incorporate dampers, which serve as shock absorbers or vibration reducers [14]. These exoskeletons can be employed to counter the effects of gravity; for instance, supporting the weight of a worker's arm (Fig. 1) [15, 16].

Active exoskeletons, on the other hand, are more intricate, incorporating advanced electronics, sensors, actuators, and sophisticated control strategies. The control unit, incorporating a microprocessor for command management, receives inputs from the sensor data, determines the power delivery for the actuators [17]. Hence, these exoskeletons are commonly known as powered exoskeletons. This arrangement enables the detection of user intention, safe movements and higher force output. [18, 19]

Actuators produce the essential physiological torques to maneuver the exoskeleton's joints, but they contribute a considerable amount of weight to the wearable device. The selection of mechanical power generation technology becomes a pivotal choice, impacting not only weight but also factors such as energy consumption and performance. Various advantages and considerations arise from these choices in the design and functioning of exoskeletons. [20, 2]

2 Actuation Technology Analysis

The primary objective of this study was to comprehensively explore the current status of actuation technologies within active exoskeletons. This inquiry, using targeted keywords, covers a wide scope, without any specific emphasis on devices dedicated to distinct purposes or confined to specific anatomical regions. The investigation considered a variety of 66 sources, comprising research papers, journal articles, and reviews, without imposing any temporal limitations, restricted to the English language.

2.1 Important Aspects and Key Criteria

The focus lies on optimization goals of performance, efficiency, safety, and user comfort. Other factors like portability, durability, accessibility, adaptability to different users, alignment and costs are not directly covered here, although they are addressed in various articles and briefly mentioned in this study [21].

Performance The operational competence of exoskeleton systems is fundamentally reliant on their performance, which is strongly determined by the capacity of actuators to produce the requisite torques for an array of joints [22]. Performance encompasses actuation kinematics, response latency, force projection, and system compliance to user feedback, which includes agility [21].

Evaluation of a mechanical design's performance employs several key metrics such as mean absolute error, standard deviation, and relative errors. These

metrics are used to interpret and analyze the data gathered from the measurement of angular displacement, velocity, and acceleration of the moving joints of an exoskeleton during different experimental or simulation scenarios. Regarding actuator performance, a key element is residual torque, representing the difference between reference and actual torque. It offers insights into the actuator's efficiency and precision in achieving intended mechanical outcomes. [22]

Efficiency Exoskeleton systems' efficiency is tied to their ability to use energy optimally, minimize energy wastage, and maximize power transmission. This efficiency has a direct and substantial influence on the performance of exoskeleton systems. Various factors, including the energy density of power sources, power transmission, principles of thermodynamics, and the use of energy recapturing methods, illustrate this relationship. [23]

In light of this, the use of actuators with a high power-to-weight ratio is crucial to design lightweight exoskeletons capable of generating the necessary force and speed to manipulate joints accurately. Incorporating passive or quasi-passive elements with conventional actuators reduces their workload. Passive elements, as previously described, consist of elastic materials. Quasi-passive elements share the same passive nature but can be activated or deactivated as required [24]. This mixed strategy promotes an optimization of system performance while ensuring energy use is kept at efficient levels. [21]

Safety Safety is crucial in developing and operating exoskeletons, especially for users with motor disabilities. Given the intimate physical connection between the exoskeleton and the user, any minor flaw in the device's structure or function could lead to severe injuries. Hence, ongoing device variable monitoring is essential. Incorporating mechanical safety measures, like joint stops, is vital to avoid surpassing desired motion limits. Elimination of sharp edges is another design aspect that directly contributes to user safety. [21]

To further ensure safety, protocols like emergency disengagement systems, redundancy safeguards, and fault tolerance mechanisms need to be incorporated into the system. Additionally, the use of both passive and active damping techniques is crucial to mitigate abrupt actuator movements and maintain the system's overall stability. Lastly, the exoskeleton system must have the capability to respond to unexpected disturbances or external forces effectively. This ability contributes to maintaining the system's dynamic stability, thereby further enhancing user safety during operation. [25]

User Comfort The acceptance and continuous use of exoskeleton systems are greatly influenced by user comfort, optimal weight distribution, and ergonomic design [26]. Comfort is closely tied to precise individual adjustments. Users consistently locate their preferred settings with reliability [27]. This emphasizes the importance of providing users the means to find and set their optimal parameters autonomously, contributing to overall comfort and usability.

Beyond these, factors such as noise reduction [28], heat dissipation [29], and vibration control directly impact user comfort. In relation to actuation technology, the choice of actuator could directly influence these factors thus affect overall user comfort and device acceptance [30].

2.2 Structural Design and Material Selection

Exoskeletons form a unique area within robotics, where the conventional "stiffer is better" approach doesn't always apply [2]. Rigid exoskeletons offer precision and responsiveness for structured tasks, driven by the ease of controlling the material and the inherent stability they provide. However, these qualities might not suffice in addressing the requirements for safety and adaptability in unpredictable environments. The heavier weight of rigid exoskeletons also significantly affects comfort and endurance negatively. [31]

Furthermore, rigid exoskeletons require either careful alignment of the biological joint with the exoskeleton joint or a mechanism to overcome joint axes misalignment problems, and they are difficult to customize for a specific user. The introduction of soft exoskeletons aims to alleviate these complications by enabling certain flexibility that conforms to the shape of their user (Fig. 2). [20]

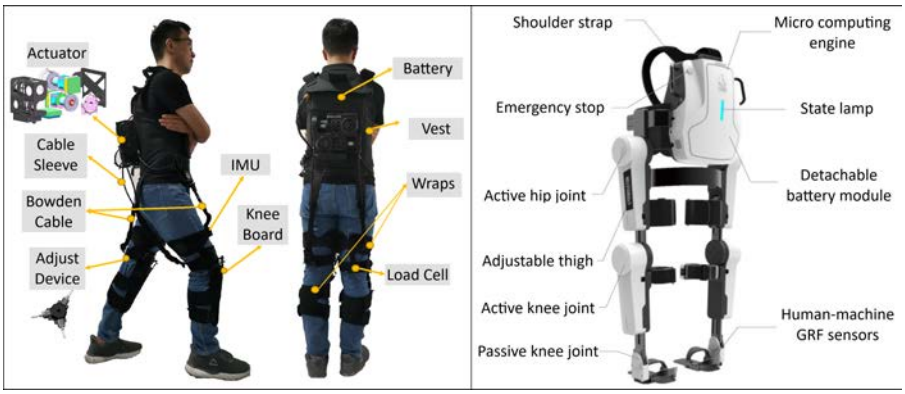


Fig. 2. The image comprises two illustrations of lower limb exosekeltons: (a) depicts a soft LLE. This exoskeleton introduces a soft system design. The Bowden cable generates tensile force and torque for movement, while load cells measure force and the IMU (Inertial Measurement Unit) sensor collects orientation, velocity, and acceleration data, enhancing control and response. Retrieved from [32]. (b) showcases a rigid LLE, UGO. This exoskeleton aids balanced walking for patients with spinal cord injuries or stroke. It features powered hip and knee joints and a passive ankle joint. Sensors and encoders provide data on joint angle and torque. Retrieved from [33].

Utilizing 3D scanning technology, custom-fit and lightweight exoskeletons are being manufactured to ensure optimal user adaptation, alignment and comfort.

High-strength, lightweight materials, such as aluminum alloys and carbon fiber composites, are usually preferred. [34]

Another crucial consideration in mechanical design is determining the number of degrees of freedom (DoF) in the system. DoF refers to the independent movements available in the joints, including translations and rotations. It plays a vital role in precisely defining the position and orientation of each device segment. Increasing the number of degrees of freedom enhances both mechanical complexity and manipulation capabilities. [35]

This challenge becomes particularly pronounced in the design of hand exoskeletal devices (HED). Hands play a crucial role in activities of daily living (ADLs) and possess the largest somatosensory mapping representation in the brain, rendering them highly sensitive and accurate for their intended functions [36]. The human finger, for instance, possesses four DoF. Replicating such dexterity and fine motor control in exoskeleton technology is a complex task [37].

2.3 Generation of Mechanical Power

Actuators can be characterized based on their method of generating mechanical power. Pneumatic, hydraulic, and electric actuators represent the three most commonly employed types. They drive motion to execute actions such as lifting, blocking, or clamping. [38] Furthermore, smart materials such as Shape Memory Alloy (SMA) are also being utilized [39].

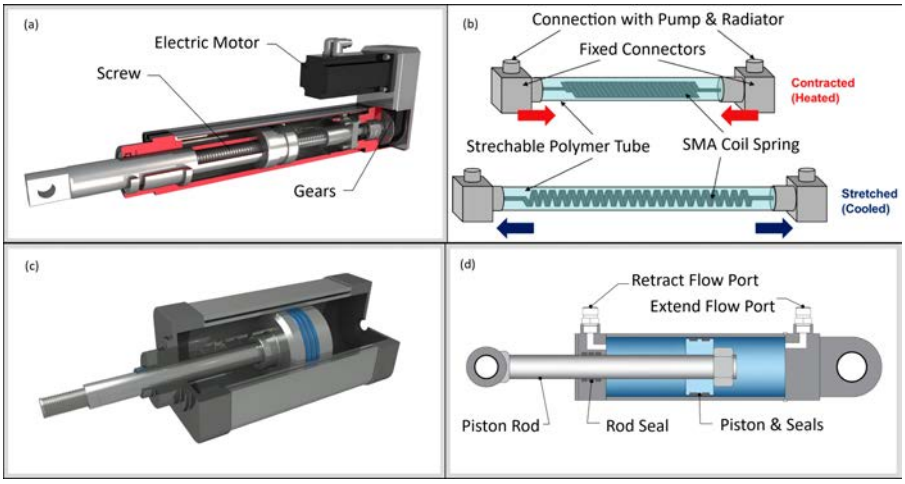


Fig. 3. Different Actuators: (a) Electric Actuator. Retrieved from [40]. (b) Shape Memory Alloy. Retrieved from [41]. (c) Pneumatic Actuator. Retrieved from [42]. (d) Hydraulic Actuator. Retrieved from [43].

Electric Actuators Electric actuators are powered by electricity and generate movement through current, which drives a motor that creates torque, activating the actuator. They are recognized for speed and precision, but they can be vulnerable to sudden power shifts [44]. Electric actuators often include elements such as lead screws and gears and typically contain the motor within the actuator assembly itself (Fig. 3.(a)). Electric motors can be further broken down into Alternating Current (AC), Alternating Current direct (AC direct), Brushless Direct Current (DC), and Brushed Direct Current (DC) motors. [45]

Brushed DC motors were initially popular due to their operational mechanism. These motors function by passing an electric current through coils, causing them to experience magnetic repulsion and attraction from magnets, thus initiating rotation. However, the rise of AC motors came later, driven by advancements in microcomputers and circuit elements. AC motors operate by periodically altering the direction of electric current flow, generating a changing magnetic field that propels motor rotation. [45, 46]

Pneumatic and Hydraulic Actuators Pneumatic actuators function through the compression of atmospheric air into a controlled, elevated pressure level that remains safe. A typical pneumatic actuator consists of "a primary motor, a compressor, a delivery hose network, an air storage tank, and the actuator component itself" (Fig. 3.(c)) [38]. Most systems utilizing pneumatic power operate at compression rates around 80 to 100 psi. [45]

Conversely, hydraulic actuators operate similarly but utilize fluid, typically oil, in place of air. The oil's property of incompressibility, along with its viscosity, enables efficient energy transfer. A hydraulic actuator system comprises "a regulated throttle, a piston or ram, and a tube network for pressure control."(Fig. 3.(d)). [47]

Both pneumatic and hydraulic actuators excel in enabling diverse rotational movements and handling substantial loads without overheating, yet their power sources are heavy, posing challenges to their portability [44]. Pneumatic actuators have gained popularity due to their affordability, cleanliness, and safety aspects. The cleanliness is particularly advantageous in comparison to hydraulic actuators, which come with a potential risk of fluid leakage. However, the force output of pneumatic actuators tends to be limited when compared to their hydraulic counterparts. [45, 48]

Shape Memory Alloy Shape Memory Alloys (SMAs) are exceptional materials that can regain their original shape (the memorized shape) when heated above a specific transformation temperature (Fig. 3.(b)). They can undergo significant reversible deformations without suffering permanent damage. "This presents a good force-to-weight ratio, small volume, and noiseless operation, the SMA-based actuators being considered a good actuation solution for wearable and soft robotics applications and in particularly for rehabilitation devices.". However, SMAs do have drawbacks, including the challenging control and low

work frequency due to the hysteresis effect, high costs, and the need for heat dissipation mechanisms. [39]

2.4 Actuation Principle

Various actuation approaches are employed in wearable robotics, such as conventional actuation, series elastic actuation (SEA), and quasi-direct drive (QDD) actuation. Conventional actuation involves the use of high-speed, low-torque motors like Brushless DC coupled with large ratio gears to satisfy precise torque, velocity, and control requirements. They encounter significant mechanical impedance, meaning they resist natural human movements. Control algorithms partially reduce this impedance, yet complete elimination remains unfeasible due to inherent inertia and friction challenges. [49]

Direct actuation, a derivative of this concept, allows a motor to directly control a joint without an intermediary transmission system, but this causes high inertia [49]. Another approach is to use cables for actuation transmission, where motor torque is transmitted to joints through cables, winding wheels, and pulleys, leading to significant reductions in exoskeleton weight and required joint-level torque. Bowden cables are commonly used in this context (Fig. 2.(a)). Such systems position the motor and transmission often on the wearer's back for enhancing weight distribution. [2]

SEAs, also known as elastic actuators, combine a main actuator with an elastic element. They offer impact load absorption, passive mechanical energy storage, and lower mechanical output impedance compared to rigid actuators. [50] However, SEAs often have to trade-off performance factors such as control bandwidth, system complexity, system weight, and size. Additionally, they are often paired with cable transmission, where the spring element is linked with the Bowden cable, creating a serial elastic actuator Bowden cable drive. [2]

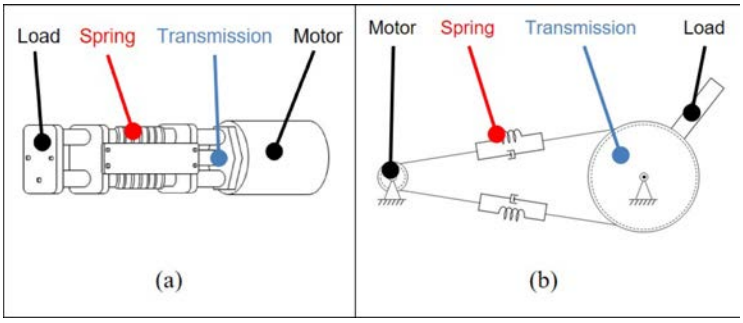


Fig. 4. Comparing FSEA (a) with RFSEA (b). Retrieved from [51]

The Force Sensing Series Elastic Actuator (FSEA) and the Reaction Force Sensing Series Elastic Actuator (RFSEA) are two common SEA designs. In the

former, the compliant element is located between the gearbox output and the load, while in the latter, the spring is situated between the motor housing and chassis ground (Fig. 4). RFSEAs are more compact and provide greater motion range, but offer less direct force sensing, reduced force tracking, and weaker impact load protection compared to FSEAs. [52]

Elastic actuators introduce nonlinear compliance, characterized by a nonlinear relationship between applied force and resulting displacement [53]. Nonlinear compliance often allows for more complex and versatile system behaviors. For example, it can enable adaptive responses where the system stiffness changes depending on the magnitude of the applied force, the rate of change of force, or other system states. [54, 55]

QDD actuation incorporates "a custom high torque density motor and low ratio gear transmission" [56]. They exhibit favorable attributes like minimal friction, toughness, simplicity, robust force control, and particularly selectable impedance, resulting in significant backdrivability. A noteworthy drawback is the decreased torque density of the entire actuation system, as it's considered to be heavy and bulky. [57]

2.5 Biomimicry in Actuation Technologies

Striving to accurately mimic the characteristics of human muscles, tendons, and insect exoskeletons, scientists are exploring actuation technologies that substantially borrow from these biological entities. Insects, with their rigid exoskeletons, comprised of laminated chitin fiber composites within a crossed matrix, act as a model for research [58]. These exoskeletons not only protect but adapt to the needs of the insect by modulating their thickness, stiffness, and fiber orientation based on various external stimuli. [59]

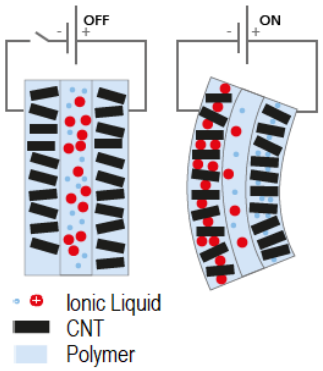


Fig. 5. Diagram of EAP Actuator. Retrieved from [60]

Inspired from biomimicry, Electroactive polymer actuators, also known as electromechanically active polymers (EAPs), alter their shape and size when subjected to electrical stimuli, resulting in significant changes in stress and strain. They stand out due to their high mechanical flexibility, lightweight nature, simple and adaptable structure, quiet operation, and absence of heat generation. Mimicking muscle-like behavior, dielectric elastomer actuators (DEAs) have established themselves as some of the most studied and promising soft actuators [61]. They function as compliant capacitors, with an elastic dielectric sandwiched between two flexible electrodes, enabling changes akin to those seen in mammalian skeletal muscles when stimulated electrically (Fig. 5) [62]. While EAPs hold great promise, their intricate manufacturing process and reliance on high voltages in the kV range present challenges [63]. These challenges include delivering and containing such power in batteries.

Due to intricate manufacturing, there has been an emphasis on efficient production of actuator elements powering these muscles, with advancements in components like carbon nanotubes (CNTs) and wires driving innovation in artificial muscle technology. CNTs are tubular structures composed of carbon atoms, with diameters on the nanometer scale and lengths up to several millimeters. Their primary application lies in the development of artificial muscles. CNT yarn or sheets can be engineered to actuate in response to thermal or electrical stimuli, providing movement and force in a similar manner to biological muscles. Their light weight, high strength, and potential for high strain make them an excellent material for this application. [64]

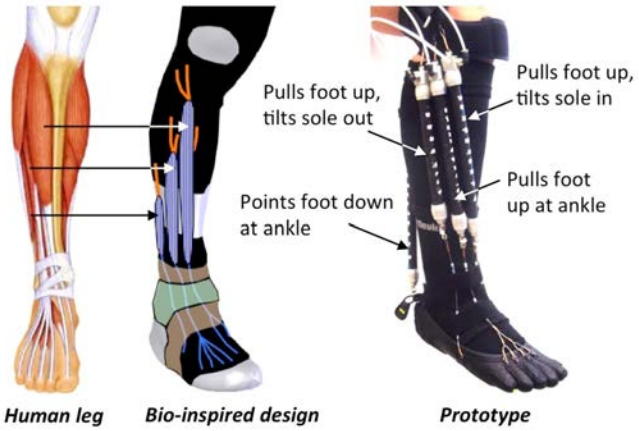


Fig. 6. Soft pneumatic exoskeleton with antagonistic actuation. Retrieved from [65]

Antagonistic actuation in exoskeletons is another approach that mirrors the human musculoskeletal system. In this setup, two motors move their transmissions in contrary directions, reflecting the functionality of muscles and tendons

in the human body (Fig. 6). This configuration enhances flexibility, compliance, and control but may induce high transient peak forces between the actuator and the exoskeleton, posing a risk to components. The risk can be reduced by incorporating compliant transmission elements between antagonistic actuators. Moreover, the integration of nonlinear compliant elements leads to a closer resemblance to biological systems, due to reduced resistance under light loads and the ability to adjust joint stiffness. [53]

3 Discussion

While hydraulic motors initially had a significantly higher power density compared to electric motors, technological advancements have gradually narrowed this gap. Presently, due to remarkable improvements in electric motor performance, the power density difference between hydraulic and electric motors has been reduced. The enhancement in electric motor performance is largely attributed to stronger permanent magnets, resulting in higher power density. [47] Currently, the torque generated by DC servo motors exhibits a relationship proportional to the motor weight raised to the power of 4/3. On the other hand, the torque produced by hydraulic motors directly correlates with their weight. [66]

4 Conclusion

Exoskeleton design and material choice establish the foundation for a system balancing strength, comfort, and durability, enhancing the user experience. A trend is evident towards creating lighter, more flexible, and softer exoskeletons.

In actuation, the goal is an optimal power-to-weight ratio while maintaining speed, precision, and safety. Pneumatic actuators prioritize speed over force output, whereas hydraulic actuators emphasize high force output and mobility, albeit with drawbacks of slower speed, expense, and leakage concerns. Electric actuators are precise and have higher speed in contrast to hydraulic actuators, yet they are hindered by their lower force output and intricate design. SMAs and EAPs aim to mitigate these trade-offs and seek comprehensive solutions, but their current state presents its own set of challenges.

Nature-inspired observations are instrumental in research, offering fresh ideas and models. This approach has led to muscle-like exoskeletons, and antagonistic actuation. The forthcoming frontier of exoskeleton actuation technology lies in this realm but addressing challenges related to energy delivery, cost, and manufacturing complexity is crucial for its advancement.

References

1. Douglas, Harper (2001). "exoskeleton". Online Etymology Dictionary. https://www.etymonline.com/word/exoskeleton. Archived from the original on 20 April 2013.

2. Tiboni M, Borboni A, Vérité F, Bregoli C, Amici C. Sensors and Actuation Technologies in Exoskeletons: A Review. Sensors. 2022; 22(3):884. https://doi.org/10.3390/s22030884
3. Pamungkas, D.S.; Caesarendra, W.; Soebakti, H.; Analia, R.; Susanto, S. Overview: Types of Lower Limb Exoskeletons. Electronics 2019, 8, 1283. https://doi.org/10.3390/electronics8111283
4. Malcolm, P.; Derave, W.; Galle, S.; de Clercq, D. A Simple Exoskeleton That Assists Plantarflexion Can Reduce the Metabolic Cost of Human Walking. PLoS ONE 2013, 8, e56137.
5. Kazerooni H, Steger R, Huang L. Hybrid Control of the Berkeley Lower Extremity Exoskeleton (BLEEX). The International Journal of Robotics Research. 2006;25(5-6):561-573. doi:10.1177/0278364906065505
6. Moeller T, Krell-Roesch J, Woll A, Stein T. Effects of Upper-Limb Exoskeletons Designed for Use in the Working Environment-A Literature Review. Front Robot AI. 2022 Apr 29;9:858893. doi: 10.3389/frobt.2022.858893. PMID: 35572378; PMCID: PMC9099018.
7. W. Banchadit, A. Temram, T. Sukwan, P. Owatchaiyapong and J. Suthakorn, "Design and implementation of a new motorized-mechanical exoskeleton based on CGA Patternized Control," 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO), Guangzhou, China, 2012, pp. 1668-1673, doi: 10.1109/ROBIO.2012.6491207.
8. Coenen, P.; van Werven, G.; van Nunen, M.P.M.; van Dieën, J.H.; Gerrits, K.H.L.; Janssen, T.W.J. Robot-assisted walking vs overground walking in stroke patients: an evaluation of muscle activity. J. Rehabil. Med. 2012, 44, 331–337
9. https://medicalfuturist.com/the-state-of-exoskeletons-in-2022/ (27.06.2023).
10. Exoskeleton Market Size, Share and Trends Analysis Report, By Mobility, By Technology, By Extremity, By End-use, By Region, And Segment Forecasts, 2023 - 2030 Report ID: GVR-1-68038-071-2.
11. https://www.sphericalinsights.com/reports/exoskeleton-market (15.08.2023).
12. https://shorturl.at/gstDQ (15.08.2023).
13. https://roboticsbiz.com/active-vs-passive-exoskeletons-explained/ (15.08.2023).
14. https://exoskeletonreport.com/2015/08/types-and-classifications-of-exoskeletons/ (14.08.2023).
15. Rahman, Tariq and Sample, Whitney and Jayakumar, Shanmuga and King, Marilyn and Wee, Jin and Seliktar, Rami (Rahamim) and Alexander, Michael and Scavina, Mena and Clark, Alisa. (2006). Passive exoskeleton for assisting limb movement. Journal of rehabilitation research and development. 43. 583-90. 10.1682/JRRD.2005.04.0070.
16. Yong-Ku Kong, Jeong Ho Kim, Hyun-Ho Shim, Jin-Woo Shim, Sang-Soo Park, Kyeong-Hee Choi, Efficacy of passive upper-limb exoskeletons in reducing musculoskeletal load associated with overhead tasks, Applied Ergonomics, Volume 109, 2023, 103965, ISSN 0003-6870, https://doi.org/10.1016/j.apergo.2023.103965.
17. Tiboni, M.; Legnani, G.; Lancini, M.; Serpelloni, M.; Gobbo, M.; Fausti, D. ERRSE: Elbow robotic rehabilitation system with an EMG-based force control. Mech. Mach. Sci. 2018, 49, 892–900
18. https://eksobionics.com/what-are-human-robotic-exoskeletons-made-of/ (30.06.2023).
19. https://orthexo.de/en/guidebook-2/what-is-an-exoskeleton/ (30.06.2023).
20. Agarwal, Priyanshu and Deshpande, Ashish. (2019). Exoskeletons: State-of-the-Art, Design Challenges, and Future Directions. 10.1093/oso/9780190455132.003.0011.

21. Pérez Vidal, A.F.; Rumbo Morales, J.Y.; Ortiz Torres, G.; Sorcia Vázquez, F.d.J.; Cruz Rojas, A.; Brizuela Mendoza, J.A.; Rodríguez Cerda, J.C. Soft Exoskeletons: Development, Requirements, and Challenges of the Last Decade. Actuators 2021, 10, 166. https://doi.org/10.3390/act10070166
22. Di Natali, C., Toxiri, S., Ioakeimidis, S., Caldwell, D., and Ortiz, J. (2020). Systematic framework for performance evaluation of exoskeleton actuators. Wearable Technologies, 1, E4. doi:10.1017/wtc.2020.5
23. J. Ortiz, T. Poliero, G. Cairoli, E. Graf and D. G. Caldwell, "Energy Efficiency Analysis and Design Optimization of an Actuation System in a Soft Modular Lower Limb Exoskeleton," in IEEE Robotics and Automation Letters, vol. 3, no. 1, pp. 484-491, Jan. 2018, doi: 10.1109/LRA.2017.2768119.
24. L. Mišković, M. Dežman and T. Petrić, "Modular quasi-passive mechanism for energy storage applications: towards lightweight high-performance exoskeleton," 2021 20th International Conference on Advanced Robotics (ICAR), Ljubljana, Slovenia, 2021, pp. 588-593, doi: 10.1109/ICAR53236.2021.9659353.
25. Duojin Wang, Xiaoping Gu, Wenzhuo Li, Yaoxiang Jin, Maisi Yang, Hongliu Yu, Evaluation of safety-related performance of wearable lower limb exoskeleton robot (WLLER): A systematic review, Robotics and Autonomous Systems, Volume 160, 2023, 104308, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2022.104308.
26. Yandell MB, Ziemnicki DM, McDonald KA, Zelik KE (2020) Characterizing the comfort limits of forces applied to the shoulders, thigh and shank to inform exosuit design. PLoS ONE 15(2): e0228536. https://doi.org/10.1371/journal.pone.0228536
27. https://innovationorigins.com/en/selected/achieving-comfort-in-exoskeletons-in-less-than-two-minutes/ (14.07.2023).
28. Medrano, R.L., Thomas, G.C., Margolin, D. et al. The economic value of augmentative exoskeletons and their assistance. Commun Eng 2, 43 (2023). https://doi.org/10.1038/s44172-023-00091-2
29. Liu Y, Li X, Lai J, Zhu A, Zhang X, Zheng Z, Zhu H, Shi Y, Wang L, Chen Z. The Effects of a Passive Exoskeleton on Human Thermal Responses in Temperate and Cold Environments. Int J Environ Res Public Health. 2021 Apr 8;18(8):3889. doi: 10.3390/ijerph18083889. PMID: 33917655; PMCID: PMC8067969.
30. Monner, Hans Peter (2005) Smart materials for active noise and vibration reduction. Noise and Vibrations - Emering Methods, 2005-04-18 - 2005-04-21, Saint-Raphael, France.
31. M. C. Carrozza et al. (Eds.): WeRob 2018, BIOSYSROB 22, pp. Comparison of a Soft Exosuit and a Rigid Exoskeleton in an Assistive Task. 415–419, 2019. https://doi.org/10.1007/978-3-030-01887-0_80
32. Chen, C.; Zhang, Y.; Li, Y.; Wang, Z.; Liu, Y.; Cao, W.; Wu, X. Iterative Learning Control for a Soft Exoskeleton with Hip and Knee Joint Assistance. Sensors 2020, 20, 4333. https://doi.org/10.3390/s20154333
33. Yang, W.; Zhang, J.; Zhang, S.; Yang, C. Lower Limb Exoskeleton Gait Planning Based on Crutch and Human-Machine Foot Combined Center of Pressure. Sensors 2020, 20, 7216. https://doi.org/10.3390/s20247216
34. Liu, F., Chen, M., Wang, L., Wang, X., Lo, CH. (2022). Custom-Fit and Lightweight Optimization Design of Exoskeletons Using Parametric Conformal Lattice. In: Yuan, P.F., Chai, H., Yan, C., Leach, N. (eds) Proceedings of the 2021 DigitalFUTURES. CDRF 2021. Springer, Singapore. https://doi.org/10.1007/978-981-16-5983-6-12
35. Sanjuan, J.D.; Castillo, A.D.; Padilla, M.A.; Quintero, M.C.; Gutierrez, E.E.; Sampayo, I.P.; Hernandez, J.R.; Rahman, M.H. Cable driven exoskeleton for upper-limb rehabilitation: A design review. Robot. Auton. Syst. 2020, 126, 103445.

36. Janko D, Thoenes K, Park D, Willoughby WR, Horton M, Bolding M. Somatotopic Mapping of the Fingers in the Somatosensory Cortex Using Functional Magnetic Resonance Imaging: A Review of Literature. Front Neuroanat. 2022 Jun 29;16:866848. doi: 10.3389/fnana.2022.866848. PMID: 35847829; PMCID: PMC9277538.
37. A. Wege and G. Hommel, "Development and control of a hand exoskeleton for rehabilitation of hand injuries," 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2005, pp. 3046-3051, doi: 10.1109/IROS.2005.1545506.
38. https://yorkpmh.com/resources/hydraulic-vs-pneumatic-vs-electric-actuators/ (15.07.2023).
39. Copaci, D., Arias, J., Moreno, L., and Blanco, D. (2022). Shape Memory Alloy (SMA)-Based Exoskeletons for Upper Limb Rehabilitation. IntechOpen. doi: 10.5772/intechopen.101751
40. https://www.iqsdirectory.com/articles/linear-actuator/electric-actuators.html
41. Jeong, J.; Yasir, I.B.; Han, J.; Park, C.H.; Bok, S.-K.; Kyung, K.-U. Design of Shape Memory Alloy-Based Soft Wearable Robot for Assisting Wrist Motion. Appl. Sci. 2019, 9, 4025. https://doi.org/10.3390/app9194025
42. https://www.linear-actuators.net/pneumatic-actuators/
43. https://www.hydrauliccylindermanufacturers.net/hydraulic-actuators/
44. Pérez Vidal, A.F.; Rumbo Morales, J.Y.; Ortiz Torres, G.; Sorcia Vázquez, F.d.J.; Cruz Rojas, A.; Brizuela Mendoza, J.A.; Rodríguez Cerda, J.C. Soft Exoskeletons: Development, Requirements, and Challenges of the Last Decade. Actuators 2021, 10, 166. https://doi.org/10.3390/act10070166
45. Hays, E.; Slayton, J.; Tejeda-Godinez, G.; Carney, E.; Cruz, K.; Exley, T.; Jafari, A. A Review of Rehabilitative and Assistive Technologies for Upper-Body Exoskeletal Devices. Actuators 2023, 12, 178. https://doi.org/10.3390/act12040178
46. https://www.powerelectric.com/motor-resources/motors101/ac-motors-vs-dc-motors (15.07.2023).
47. Sakama, S.; Tanaka, Y.; Kamimura, A. Characteristics of Hydraulic and Electric Servo Motors. Actuators 2022, 11, 11. https://doi.org/10.3390/act11010011
48. Jobin Varghese, Akhil V.M., Rajendrakumar P.K., Sivanandan K.S., A rotary pneumatic actuator for the actuation of the exoskeleton knee joint, Theoretical and Applied Mechanics Letters, Volume 7, Issue 4, 2017, ISSN 2095-0349, https://doi.org/10.1016/j.taml.2017.08.002.
49. Marconi, D.; Baldoni, A.; McKinney, Z.; Cempini, M.; Crea, S.; Vitiello, N. A novel hand exoskeleton with series elastic actuation for modulated torque transfer. Mechatronics 2019, 61, 69–82.
50. Wang, T.; Zheng, T.; Zhao, S.; Sui, D.; Zhao, J.; Zhu, Y. Design and Control of a Series–Parallel Elastic Actuator for a Weight-Bearing Exoskeleton Robot. Sensors 2022, 22, 1055. https://doi.org/10.3390/s22031055
51. Lee, C.; Kwak, S.; Kwak, J.; Oh, S. Generalization of Series Elastic Actuator Configurations and Dynamic Behavior Comparison. Actuators 2017, 6, 26. https://doi.org/10.3390/act6030026
52. N. Paine, S. Oh and L. Sentis, "Design and Control Considerations for High-Performance Series Elastic Actuators," in IEEE/ASME Transactions on Mechatronics, vol. 19, no. 3, pp. 1080-1091, June 2014, doi: 10.1109/TMECH.2013.2270435.
53. Jäger, M.; Helbig, T.; Goos, M.; Köhring, S.; Witte, H. Characterization of an Antagonistic Actuation System with Nonlinear Compliance for an Upper-Arm Exoskeleton. Actuators 2023, 12, 196. https://doi.org/10.3390/act12050196

54. Sanchez-Villamañan, M., Gonzalez-Vargas, J., Torricelli, D. et al. Compliant lower limb exoskeletons: a comprehensive review on mechanical design principles. J NeuroEngineering Rehabil 16, 55 (2019). https://doi.org/10.1186/s12984-019-0517-9
55. Davide Pilastro, Roberto Oboe, Tomoyuki Shimono, A Nonlinear Adaptive Compliance Controller for Rehabilitation, IEEJ Journal of Industry Applications, 2016, Volume 5, Issue 2, Pages 123-131, Released on J-STAGE March 01, 2016, Online ISSN 2187-1108, Print ISSN 2187-1094, https://doi.org/10.1541/ieejjia.5.123
56. S. Yu et al., "Quasi-Direct Drive Actuation for a Lightweight Hip Exoskeleton With High Backdrivability and High Bandwidth," in IEEE/ASME Transactions on Mechatronics, vol. 25, no. 4, pp. 1794-1802, Aug. 2020, doi: 10.1109/TMECH.2020.2995134.
57. Gealy, David & McKinley, Stephen & Yi, Brent & Wu, Philipp & Downey, Phillip & Balke, Greg & Zhao, Allan & Guo, Menglong & Thomasson, Rachel & Sinclair, Anthony & Cuellar, Peter & McCarthy, Zoe & Abbeel, Pieter. (2019). Quasi-Direct Drive for Low-Cost Compliant Robotic Manipulation.
58. Muthukrishnan S, Mun S, Noh MY, Geisbrecht ER, Arakane Y. Insect Cuticular Chitin Contributes to Form and Function. Curr Pharm Des. 2020;26(29):3530-3545. doi: 10.2174/1381612826666200523175409. PMID: 32445445; PMCID: PMC7755156.
59. Scuderi, Giuliana. "Adaptive building exoskeletons: A biomimetic model for the rehabilitation of social housing." ArchNet-IJAR: International Journal of Architectural Research 9.1 (2015): 134.
60. https://www.fpc-aist.fraunhofer.jp/en/eap.html (18.08.2023).
61. Behboodi A, Lee SCK. Benchmarking of a Commercially Available Stacked Dielectric Elastomer as an Alternative Actuator for Rehabilitation Robotic Exoskeletons. IEEE Int Conf Rehabil Robot. 2019 Jun;2019:499-505. doi: 10.1109/ICORR.2019.8779378. PMID: 31374679; PMCID: PMC8050934.
62. Oveissi, F.; Fletcher, D.F.; Dehghani, F.; Naficy, S. Tough hydrogels for soft artificial muscles. Mater. Des. 2021, 203, 109609.
63. Li, B.; Niu, S.; Li, B.; Wang, P.; Qiao, Y. Design and Analysis of Mechanical Characteristics of EAP Flexible Drivers. Machines 2022, 10, 1241. https://doi.org/10.3390/machines10121241
64. Craddock, M.; Augustine, E.; Konerman, S.; Shin, M. Biorobotics: An Overview of Recent Innovations in Artificial Muscles. Actuators 2022, 11, 168. https://doi.org/10.3390/act11060168
65. https://newatlas.com/soft-exoskeleton-ankle-rehabilitation/30542/ (18.08.2023).
66. Nakano, K.; Konno, Y. The comparative study on the characteristics of AC, DC Servomotors and hydraulic motors. Hydraul. Pneum. 1997, 28, 466–472

Applications of the ABC Algorithm in UAV-Aided WSNs

Tobias Kempf[0009−0001−3327−4652]

Karlsruher Institut für Technologie, 76131 Karlsruhe, Germany
tobias.kempf@student.kit.edu

Abstract. Wireless Sensor Networks (WSNs) play a role in various applications, including environmental monitoring, surveillance and agriculture. The effectiveness of WSNs can be further enhanced by incorporating mobility, especially by including a UAV as a mobile data sink. In recent years, the Artificial Bee Colony (ABC) algorithm has received considerable attention as a bio-inspired optimization technique for WSNs due to its simplicity, flexibility and efficiency. This paper aims to provide a comprehensive analysis of the various applications of the ABC algorithm in UAV-Aided WSNs. It examines how the algorithm can be adapted and optimized to maximize objectives and minimize convergence times. In addition, the review investigates the difficulties of adapting the ABC algorithm to these applications. To provide a comprehensive understanding of the topic, a review of relevant research papers and studies from reputable sources will be conducted. The results of this literature review contribute to a deeper understanding of the potential and limitations of the ABC algorithm.

Keywords: Artificial Bee Colony algorithm, Wireless Sensor Networks, UAV, Path Planning, Localization

1 Introduction

Due to their potential applications in various fields such as environmental monitoring, disaster management and precision agriculture, Wireless Sensor Networks (WSNs) have become very interesting in recent years [2]. The efficiency and effectiveness of WSNs can be enhanced by integrating Unmanned Aerial Vehicles (UAVs) into the network infrastructure [15]. UAVs can serve as mobile base stations, relaying data between the sensor nodes and base, thereby extending the coverage and connectivity of the network. They can also be deployed to perform sensing tasks, such as monitoring specific areas of interest or collecting data from otherwise inaccessible locations. Additionally, UAVs can be used to optimize network energy consumption by intelligently managing the movement and communication of the sensor nodes [15]. In this context, the Artificial Bee Colony (ABC) algorithm has emerged as a promising optimization technique for improving the performance of UAV-aided WSNs. The ABC algorithm is a bio-inspired optimization algorithm that mimics the swarm behavior of honeybees

[9]. It has been successfully applied to various optimization problems [10]. In recent years, researchers have recognized the potential of the ABC algorithm for addressing the unique challenges of UAV-aided WSNs. By leveraging the ABC algorithm's optimization capabilities, researchers aim to optimize key aspects of UAV-aided WSNs, such as energy efficiency, node deployment and data collection strategies [16]. This paper aims to provide a comprehensive analysis of the applications of the ABC algorithm in UAV-aided WSNs. It explores how the ABC algorithm can be applied to optimize different aspects of the network, addressing challenges such as limited energy resources, dynamic node deployment, and efficient data collection. By surveying the existing literature on this topic, this review paper aims to identify the key research trends, methodologies, and findings related to the integration of the ABC algorithm in UAV-aided WSNs. To achieve this, a review of relevant research papers and scholarly articles from reputable sources will be conducted. The findings from this paper will contribute to a deeper understanding of the potential and limitations of the ABC algorithm in optimizing UAV-aided WSNs. Ultimately, the aim is to enhance the efficiency, performance, and reliability of UAV-aided WSNs through the utilization of the ABC algorithm and contribute to the advancement of this rapidly evolving field.

2 Background

In this section the ABC Algorithm is explained and compared to other Algorithms commonly used in WSN's.

2.1 ABC Algorithm

The ABC Algorithm is a swarm intelligence optimization algorithm inspired by the foraging behavior of honey bees [7]. It was proposed by by Karaboga in 2005 for optimizing numerical problems [8]. The algorithm is based on the social behaviour of honeybees, where the colony works together to find the best food sources in their environment. In nature, a honeybee colony strives to make efficient use of scattered food sources. To do this, it uses forager bees that can take on roles and communicate with each other to optimize the outcome. The food sources are of different value based on the distance of the nest, its quantity and quality and nearby threats [18]. The ABC Algorithm mimics this behavior in order to find solutions in a search space. The foraging bees can be divided into three groups:

Employed foragers are currently exploiting a food source. On their return to the nest they communicate the state of their food source to the other bees by performing a waggle dance [20]. This indicates the direction of the food source and its distance from the nest. They also exchange information about the richness of the food source. They will continue to exploit the food source until it is depleted.

In the algorithm, each employed bee corresponds to a solution in the search space. The fitness of its solution is evaluated based on an objective function [8].

Onlooker foragers wait in the hives dance area. This is where the returning employed foragers exchange information about their current food sources. The onlookers will watch many employed foragers and try to pick out the one who is communicating the food source with the most value. They decide individually whether a food source is valuable enough to be exploited further. If so, they become employed foragers and use the shared information about the location of the food source to exploit it [20].

In the algorithm onlooker bees select the employed bees based on a probability distribution of their fitness values as their starting points for further exploration. employed bees with higher fitness have a higher probability of being chosen. The onlooker bees then evaluate the solutions provided by the employed bee and generate a new solution in a short ranged scope around the previous solution. If the solution has a higher fitness value than the previous one, it is adopted. If a solution of a employed bee does not improve over a predefined number of cycles it is abandond [8].

Scouting foragers are looking for new food sources. They discover new food sources through exploration [20]. As soon as they discover a new food source, they immediately begin to exploit it, becoming a employed forager.

In the algorithm Scout bees are responsible for discovering new areas of the search space by randomly generating new solutions. Every time a employed bee solution is abandond it is replaced by a scouts solution [8].

2.2 Other Optimization Algorithms

In the field of optimization algorithms, multiple approaches have been developed. This background section will provide a brief introduction to Bayesian Optimization, Reinforcement Learning, and Genetic Algorithm and compares them to the Artificial Bee Colony (ABC) algorithm.

Bayesian Optimization [6] is a sequential model-based optimization technique that uses Gaussian process regression. It constructs a surrogate of the objective function and iteratively updates this model to make informed decisions about where to sample next. Bayesian Optimization is particularly effective when the objective function is expensive to evaluate.

Reinforcement Learning [19] is a machine learning approach that focuses on learning optimal actions in a sequential decision-making process. Agents learn through trial and error by interacting with an environment and receiving feedback in the form of rewards or penalties, maximizing the cumulative numeric reward over time. The agent has to find a balance between exploiting rewarding actions and exploring new actions. This balance is affected by parameters.

Genetic Algorithm [14] is a population-based search and optimization technique inspired by the process of natural selection and evolution. It represents candidate solutions as individuals in a population and uses genetic operators such as selection, crossover, and mutation to mimic the evolutionary process. The algorithm iteratively generates new generations of solutions by applying these genetic operators, favoring individuals with higher fitness values. Genetic algorithms are particularly useful for solving optimization problems with large search spaces and multiple conflicting objectives.

Comparison with the ABC Algorithm
While Bayesian Optimization, Reinforcement Learning, and Genetic Algorithm are all powerful optimization approaches, they differ from the ABC algorithm in several key aspects. Unlike Bayesian Optimization and Reinforcement Learning the ABC algorithm is a population-based algorithm. The ABC algorithm is characterized by its simplicity, efficiency and robustness. It does not require extensive parameter tuning. Bayesian Optimization, Reinforcement Learning, and Genetic Algorithm, on the other hand, may involve more intricate modeling and parameter tuning processes, making them potentially more complex and computationally expensive. This allows the ABC algorithm to be used to further improve a feasible solution of another heavier algorithm, such as the Genetic Algorithm [13].

3 Related Work

In the field of optimization techniques, the Artificial Bee Colony (ABC) algorithm has gained significant attention due to its simplicity and efficiency. Various studies have explored the applications of the ABC algorithm in different domains, highlighting its effectiveness in solving complex optimization problems. One notable paper that provides a comprehensive survey on the applications of the ABC algorithm is titled "Applications of Artificial Bee Colony optimization technique: Survey" (Kaswan, Choudhary, Sharma, 2015) [10].

That paper conducted an extensive review of the literature to investigate the diverse applications of the ABC algorithm. The authors identified several domains where the ABC algorithm has been successfully applied, including but not limited to: structural optimization, face pose estimation, bioinformatics, and MR brain image classification. The survey paper presents a systematic analysis of the ABC algorithm's performance in each domain, providing insights into its strengths and limitations [10].

Due to the wide range of applications, only one of the reviewed papers [1] deals with WSNs and focuses only on node deployment. Therefore, the optimization possibilities regarding the path planning aspect of a UAV-Aided WSN are not mentioned. The present paper is an addition to the previous works by covering this specific area of research.

4 Application Areas

4.1 Review Methodology

The reviewed articles are retrieved using a four-step methodology similar to [17]. In the first step, articles are retrieved using a keyword search in Google Sholar. The keywords used are semantically identical to the keywords of this review. In the second step, the articles are screened to reduce the number of articles. The following restrictions are applied in the screening process:

1. The articles should be published after 2011, more than 5 years after the introduction of Karaboga's ABC algorithm [8], to ensure the use of his terminology in the articles.
2. The articles should be written in English.

In the third step, the keywords of the articles are analyzed. Articles that do not contain a keyword semantically similar to 'ABC algorithm' are excluded. Additionally, either the keyword 'WSN' or 'UAV' must be present.
In the fourth step, the remaining seven articles are classified into two categories. They are either in the category of UAV Path Planning (section 4.3) or Localization (section 4.4).

4.2 Overview

The following sections review the selected articles in detail, focusing on the modifications made to the ABC algorithm. As can be seen in table 1, there are differences in the severity of modification depending on the use case of the algorithm. In particular, path planning leads to more modified ABC stages and the use of different algorithms supporting the ABC algorithm. Localization problems, on the other hand, tend to work well with an less modified ABC algorithm.

Paper	Application Area	Combined Algorithm	ABC Stages Modified			
			Init	Employed	Onlooker	Scout
[16]	UAV Path Planning	PRM	X		X	X
[11]	UAV Path Planning	Logistics Map	X	X	X	
[13]	UAV Path Planning	AHP and AGA		X		
[3]	Node Localization	-	X			
[12]	Base Station Localization	-				
[4]	UAV Localization	-				
[5]	Node Localization	-				X

Table 1. Overview of the reviewed papers

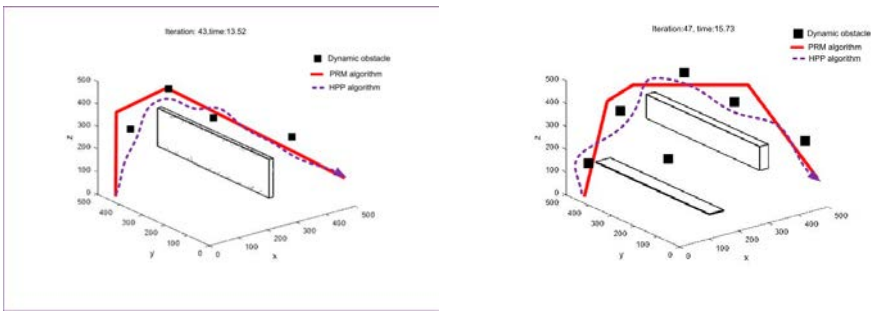
4.3 UAV Path Planning

Hybrid Path Planning for Efficient Data Collection in Uav-Aided Wsns for Emergency Applications

In this paper [16] a hybrid path planning algorithm for UAVs is proposed. It combines the probabilistic roadmap (PRM) algorithm and the optimized artificial bee colony (ABC) algorithm in order to find the shortest collision-free path for the UAV in an emergency situation. It takes both static obstacles and dynamic threats into account. In its WSN scenario, the positions of the collection zones are already predefined. However, it optimizes the path for a low altitude to increase the chance of a successful transmission. It proposes to first find a collision-free path with the PRM algorithm, avoiding static obstacles, and then optimize it with a modified ABC algorithm. (Fig 1)

It alters the ABC Algorithm as follows:
In the initialization phase it pre-calculates waypoints using the PRM algorithm
In the employed bee phase it employs bees to the generated positions
In the onlooker bee phase it generates new waypoints based on probability values. If the waypoint collides with a static obstacle or a threat is detected the waypoint is ignored.
In the scout bee phase the fitness values of the onlooker waypoints are compared to the employed waypoints. If they posses a higher fitness value, they become the new employed waypoint.
The phases starting with the employed bee phase are repeated several times. In the end, an optimized path is generated by connecting the waypoints of the employed bees.

Compared to the stand-alone algorithms, it successfully improves flight time, convergence time and energy efficiency.



(a) (b)

Fig. 1. Example of the HPP algorithm [16]

Path Planning for Unmanned Air Vehicles Using an Improved Artificial Bee Colony Algorithm

In this paper [11] a modified initialization strategy for the standard ABC is proposed for use in UAV path planning. It takes into account constraints such as threat areas, terrain, fuel and flight time. Threats are modelled as circles with the threat in the centre. While the UAV is inside the circle, it is exposed to the threat with a probability proportional to the distance from the threat centre. The path planning problem is transformed into a D dimensional function optimization problem by connecting the start and end points and then drawing lines L_1 ... L_D perpendicular to this axis. The lines are evenly spaced. (Fig 2)

The ABC algorithm is modified in the following way:
In the initialization phase, a set of starting positions is generated. Instead of generating them randomly the paper proposes to use of the logistic map to initialize the population to increase the population diversity. In addition, opposition-based learning is used to increase the speed of convergence. The opposite of a point is defined as the point generated by mirroring at the center of the space.
In the employed bee phase it employs bees to the generated positions. In each iteration an employed bee updates its position. In contrast to the standard approach, the step size changes with the number of iterations, decreasing towards the end.
In the scout bee phase employed sources are abandoned if they have not improved over a predefined number of iterations. For each abandoned source, a new one is randomly selected.
The phases starting at the employed bee phase are repeated for a number of times. The employed position with the highest fitness value is chosen as the estimated optimal position of the waypoint of the current vertical Line L.

Compared to the standard ABC algorithm, it achieves more optimised paths and faster convergence times.

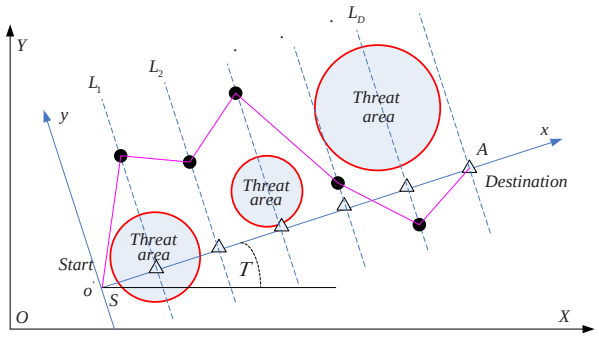


Fig. 2. Example of the threat modelling [11]

Multi-Uav Optimal Mission Assignment and Path Planning for Disaster Rescue Using Adaptive Genetic Algorithm and Improved Artificial Bee Colony Method

In this paper [13] a combination of the Analytic Hierarchy Process (AHP), an Adaptive Genetic Algorithm (AGA) and an Improved Artificial Bee Colony (IABC) is proposed to solve the mission assignment and path planning problem. After assigning mission points the IABC algorithm is used to plan a path between neighbouring mission points (Fig 3). It uses a balanced search strategy.

The ABC algorithm is modified in the following way:
In the initialization phase, a set of solutions is generated. Each solution reprents a path between the to mission points.
In the employed bee phase it employs bees to the generated positions. In each iteration an employed bee updates its position. The search radius for the new position shrinks as the number of iterations increases.
In the onlooker bee phase it uses a roulette selection to assign onlooker bees to the positions of the employed bees based on the fitness value of the position.
In the scout bee phase employed sources are abandoned if they have not improved over a predefined number of iterations. For each abandoned source, a new one is randomly selected.
The phases are repeated for a number of times, starting at the employed bee phase.

Compared to the standard ABC and GA algorithms, it achieves more optimised paths and faster convergence times.

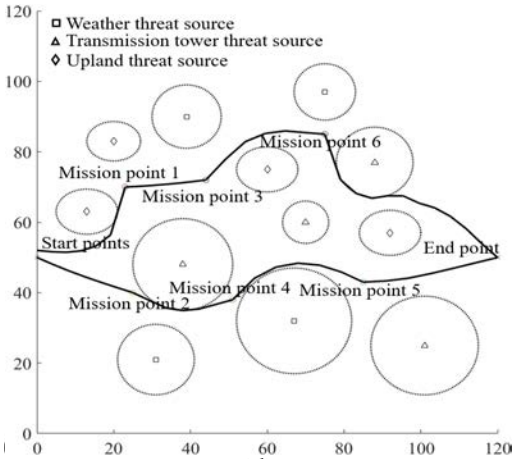


Fig. 3. Visualization results of AGA + IABC [13]

4.4 Localization

Implementation of an Efficient Artificial Bee Colony Algorithm for Node Localization in Unmanned Aerial Vehicle Assisted Wireless Sensor Networks

This paper [3] proposes the ABC algorithm as a solution to the node localization problem in wireless sensor networks. It relies on a mobile UAV to collect data from the nodes instead of using fixed ground anchors. This improves the distance calculation as a UAV can provide an almost clear line of sight to the nodes. The algorithm proposed by the paper places on node at a time using the ABC algorithm.

The ABC algorithm is modified in the following way:
In the initialization phase, it generates a set of random positions.
In the employed bee phase it employs bees to the generated positions.
In the onlooker bee phase it uses a roulette selection to assign onlooker bees to the positions of the employed bees based on the fitness value of the position. Then the onlookers generate a new food position in the neighbourhood of their current position. If the fitness of onlooker source has an equal or better quality than the employed source, the employed one is replaced by the onlooker one.
In the scout bee phase employed sources are abandoned if they have not improved over a predefined number of iterations. For each abandoned source, a new one is randomly selected.
The phases starting with the employed bee phase are repeated a number of times. The process is repeated for all other nodes.

It achieves an improvement in localization accuracy of around (10-35)% compared to classical techniques. (Fig 4)

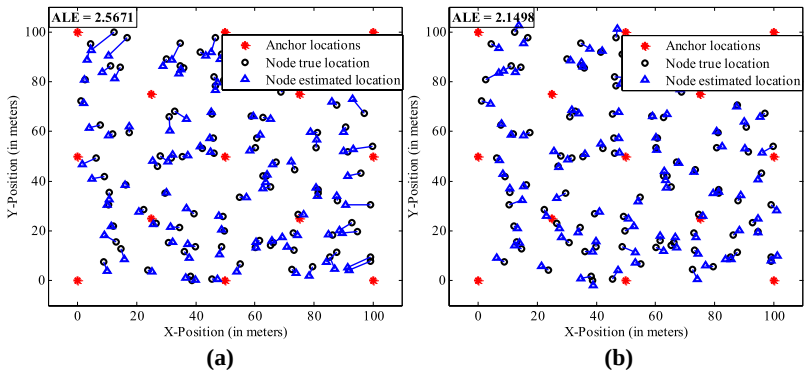


Fig. 4. Unknown node localization with 13 anchor nodes using: a DEA, b ABC (ALE = Average localization Error) [3]

Post-disaster Unmanned Aerial Vehicle Base Station Deployment Method Based on Artificial Bee Colony Algorithm

This paper [12] proposes the ABC algorithm to solve the problem of UAV base station deployment. In a post-disaster event, it aims to deploy mobile UAVs as base stations to maximize the connectivity and throughput of user equipment (UE). The variable flight altitude of the UAV-BS allows for improved coverage of the UE (fig 5). The algorithm proposed in the paper places all nodes simultaneously using the ABC algorithm.

The ABC algorithm is modified in the following way:
In the initialization phase generates a set of solutions where each solution is a set of coordinate-triples each triple representing a UAV position. The fitness of a solution is determined by network overall throughput provided by the UAVs.
In the employed bee phase each employed bee generates a new solution near the current solution randomly
In the onlooker bee phase, a roulette selection is used to assign onlooker bees to the positions of the employed bees based on the fitness value of the position. Then the onlookers generate a new food position in the neighborhood of their selected position. If the fitness of the onlooker solution has an equal or better quality than the employed solution, the employed one is replaced by the onloooker one.
In the scout bee phase employed solutions are abandoned if they have not improved over a predefined number of iterations. For each abandoned solution, a new one is randomly chosen.
The phases starting at the employed bee phase are repeated for a number of times. The employed position with the highest fitness value is chosen as the estimated optimal solution.

Compared to standard algorithms it achieves a higher network throughput.

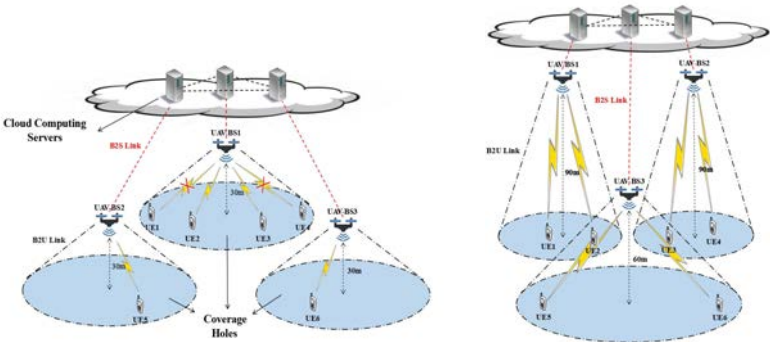


Fig. 5. Schematic of the UAV-BS flight altitude deployment (unified vs. variable)[12]

Solving Uav Localization Problem With Artificial Bee Colony (ABC) Algorithm

This paper [4] proposes the ABC algorithm as a solution to the UAV localization problem. In the paper's scenario, a user is located inside a tall building and a UAV is located outside the building (fig 6). The UAV acts as a base station that connects to the users in line-of-sight. It uses the ABC algorithm to determine an efficient location for the UAV.

The ABC algorithm is modified in the following way:
In the initialization phase a set of UAV locations is generated where each location is a coordinate triple. The fitness of a location is determined by its total path loss. Each location is assigned an employed bee
In the employed bee phase, each employed bee randomly generates a new solution near the current solution. If this solution has a higher fitness value the old solution is replaced. Otherwise a trial counter is incremented
In the onlooker bee phase a roulette selection is used to assign onlooker bees to the positions of the employed bees based on the fitness value of the position. Then the onlookers generate new locations simular to the employed bees.
In the scout bee phase, employed solutions are abandoned if the employed bee's trial counter exceeds a pre-defined limit. For each abandoned solution, a new one is randomly selected.
The phases starting from the employed bee phase are repeated a number of times.

Compared to localization based on the PSO algorithm, it reduces path loss.

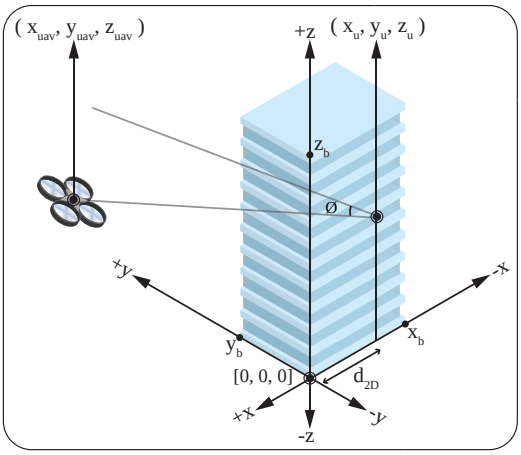


Fig. 6. Schematic of UAV localization [4]

Deployment in Wireless Sensor Networks by Parallel and Cooperative Parallel Artificial Bee Colony Algorithms

This paper [5] proposes the ABC algorithm for solving the deployment problem of sensor networks. In the paper's scenario, a set of sensor nodes must be placed to cover a region. Coverage is determined by dividing the area of interestinto equally sized subareas and checking covarge for the corners of those subareas. The paper aims to maximise the coverage of the network by correctly positioning sensor nodes. The parallel approach runs multiple colonies migrating some solutions between them. The propesed coop approach improves the migrated solution before migrating it.

The ABC algorithm is modified in the following way:
In the initialization phase a set of solutions is generated where each solution is a set of node positions
In the employed bee phase, each employed bee randomly generates a new solution near the current solution.
In the onlooker bee phase, onlooker bees are assignend to the positions of the employed bees based on the fitness value of the solution.
In the scout bee phase, new solutions are randomly generated replacing employed solutions that have not improved recently
In the migration phase the global best solution of all subcolonies is selected. For each subcolonie the best solution is compared to it. For each node the location in the global solution is changed if that improves its fitness value. Then for each subcolonie the worst solution is replaced by the global best solution.
The phases starting from the employed bee phase are repeated a number of times.

Compared to serial and parallel ABC it achieves a higher covarege at a faster convergence time (fig 7).

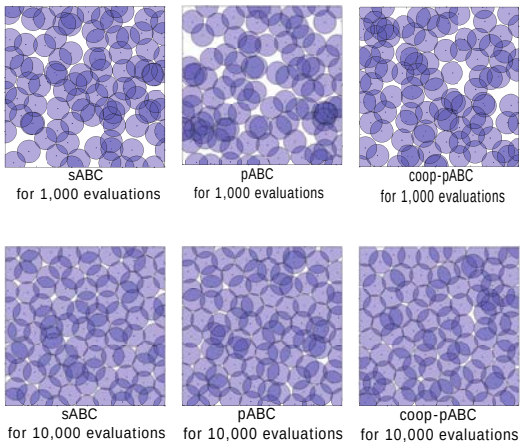


Fig. 7. covarage comparison of serial, parallel and coop [5]

5 Conclusion

The applications of the ABC algorithm in UAV-Aided Wireless Sensor Networks have demonstrated significant potential for optimizing network performance and addressing the challenges of these systems. Through this literature review, we have explored the various applications of the ABC algorithm in UAV-Aided WSNs, covering areas such as path planning, node localization and UAV localization. The ABC algorithm has proven to be effective in optimizing key aspects of UAV-Aided WSNs, producing high quality solutions at high convergence speeds. It has also proven to be compatible with other algorithms when it comes to path planning. However, while the ABC algorithm has demonstrated its effectiveness in UAV-Aided WSNs, there are still areas for further research and development. Future research could focus on optimizing the parameters of the algorithm, adapting it to specific network configurations. A combination of the improvements suggested by the reviewed papers, including stage-specific and overall optimisations such as parallelisation, can also lead to significant performance improvements.

In conclusion, the ABC algorithm shows promise as a valuable tool for optimizing UAV-Aided WSNs. Further research and development in this area can lead to significant improvements in data aggregation capabilities, energy efficiency and overall network performance.

References

1. Ajayan, A.R.: A modified abc algorithm & its application to wireless sensor network dynamic deployment. IOSR Journal of Electronics and Communication Engineering **4**, 79–82 (2013)
2. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Commun. Mag. **40**, 102–114 (2002)
3. Annepu, V., Rajesh, A.: Implementation of an efficient artificial bee colony algorithm for node localization in unmanned aerial vehicle assisted wireless sensor networks. Wireless Personal Communications **114**, 2663 – 2680 (2020)
4. Aslan, S., Demirci, S.: Solving uav localization problem with artificial bee colony (abc) algorithm. In: 2019 4th International Conference on Computer Science and Engineering (UBMK). pp. 735–738 (2019). https://doi.org/10.1109/UBMK.2019.8907034
5. Aslan, S.: Deployment in wireless sensor networks by parallel and cooperative parallel artificial bee colony algorithms. An International Journal of Optimization and Control: Theories & Applications (IJOCTA) (2018)
6. Frazier, P.I.: A tutorial on bayesian optimization (2018)
7. Karaboga, D.: Artificial bee colony algorithm. Scholarpedia **5**(3), 6915 (2010). https://doi.org/10.4249/scholarpedia.6915, revision #91003
8. Karaboğa, D.: An idea based on honey bee swarm for numerical optimization. In: AN IDEA BASED ON HONEY BEE SWARM FOR NUMERICAL OPTIMIZATION (2005)

9. Karaboğa, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of Global Optimization **39**, 459–471 (2007)
10. Kaswan, K.S., Choudhary, S., Sharma, K.: Applications of artificial bee colony optimization technique: Survey. In: 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom). pp. 1660–1664 (2015)
11. Lei, L., Shiru, Q.: Path planning for unmanned air vehicles using an improved artificial bee colony algorithm. In: Proceedings of the 31st Chinese Control Conference. pp. 2486–2491 (2012)
12. Li, J., Lu, D., Zhang, G., Tian, J., Pang, Y.: Post-disaster unmanned aerial vehicle base station deployment method based on artificial bee colony algorithm. IEEE Access **7**, 168327–168336 (2019). https://doi.org/10.1109/ACCESS.2019.2954332
13. Liu, H., Ge, J., Wang, Y., Li, J., Ding, K., Zhang, Z., Guo, Z., Li, W., Lan, J.: Multi-uav optimal mission assignment and path planning for disaster rescue using adaptive genetic algorithm and improved artificial bee colony method. Actuators **11**(1) (2022). https://doi.org/10.3390/act11010004, https://www.mdpi.com/2076-0825/11/1/4
14. Mirjalili, S.: Genetic Algorithm, pp. 43–55. Springer International Publishing, Cham (2019)
15. Popescu, D., Dragana, C., Stoican, F., Ichim, L., Stamatescu, G.: A collaborative uav-wsn network for monitoring large areas. Sensors (Basel, Switzerland) **18** (2018)
16. Poudel, S., Moh, S.: Hybrid path planning for efficient data collection in uav-aided wsns for emergency applications. Sensors (Basel, Switzerland) **21** (2021)
17. Qadir, Z., Ullah, F., Munawar, H.S., Al-Turjman, F.: Addressing disasters in smart cities through uavs path planning and 5g communications: A systematic review. Computer Communications **168**, 114–135 (2021). https://doi.org/https://doi.org/10.1016/j.comcom.2021.01.003, https://www.sciencedirect.com/science/article/pii/S0140366421000116
18. Seeley, T.D.: The Wisdom of the Hive. Harvard University Press, Cambridge, MA and London, England (1995). https://doi.org/doi:10.4159/9780674043404, https://doi.org/10.4159/9780674043404
19. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
20. Tereshko, V., Loengarov, A.: Collective decision-making in honey bee foraging dynamics. Computing Information Systems **9** (11 2004)

Reinforcement Learning for Path Planning in UAV-assisted WSN: A Review

Chenxin Tao[1]
urskl@student.kit.edu

and Tutor: Chaofan, Li[1]
chaofan.li@kit.edu

Karlsruhe Institute for Technology, Karlsruhe Kaiser Street. 12, 76131, Germany

Abstract. Reinforcement learning has been very popular in path planning. To collect data more efficiently, mobile data sinks such as UAV can be applied, which carries collected data but need path planning. The paper first introduced the basic idea of reinforcement learning and the standard components of the algorithm. Then the reason for choosing reinforcement learning instead of other algorithms is stated. In total 4 papers were reviewed, and their model and algorithm were introduced. Finally, this paper is summarized and future reseach direction is stated.

Keywords: Reinforcement Learning · Path finding · Wireless sensor network· Literature Review.

1 introduction

Wireless Sensor Networks (WSNs) have emerged as a groundbreaking technology that combines sensing, communication, and computation to enable the collection and transmission of data in various applications.[6] WSNs may consist of autonomous sensor nodes that are wirelessly interconnected to monitor and gather information from the surrounding environment. These networks have found extensive use in diverse fields, such as environmental monitoring, healthcare, industrial automation, and surveillance.

One of the key challenges in WSNs is efficiently gathering data from specific locations. This task can be accomplished by employing some mobile data sink node such as Unmanned Aerial Vehicle to collect data in case of possibly distant distributed sensor nodes instead of using direct wireless communication between the node and the base station. Thus, an efficient path-finding algorithm capable of identifying the most efficient paths that minimize energy consumption, and ensure timely delivery of data and etc. is vital.

For the reasons stated above, this paper reviewed several papers related to the topic "Reinforcement learning for path planning in WSN" and provide a wide view on different approaches to model the environment or to train the RL-network.

2 Background

2.1 Introduction to Reinforcement Learning

Reinforcement learning is a subfield of artificial intelligence (AI) and machine learning that focuses on the development of intelligent agents capable of making sequential decisions.[1] It provides a framework for learning through interaction with an environment, where an agent learns to take actions in order to maximize a cumulative reward signal. And it has been widely applied to solve the path-planning problem due to its characteristics such as learning from interaction, adaptability to uncertain and dynamic environments, and Exploration-exploitation trade-off.

In contrast to other machine learning approaches, such as supervised learning or unsupervised learning, reinforcement learning does not rely on labeled data or pre-existing knowledge about the environment. Instead, the agent learns through trial and error, receiving feedback in the form of rewards or punishments based on its actions. By exploring the environment and observing the consequences of its actions, the agent gradually develops a policy—a strategy for selecting actions—to optimize its performance and achieve long-term goals.

The environment in Reinforcement learning is typically stated in the form of a Markov decision process (MDP), where the next state is only related to the current state and the action taken in this state. The goal is to finally develop a policy that gives the actions regarding the current state to maximize the expected cumulative reward. The agent can be rewarded for taking actions according to the reward function depending on the action and the current state.

We can get the expected reward of the action taken in one state by computing the mean of that for all possible following states while the best actions according to the action policy are performed, called the Q-function. A V-function which indicates the expected reward of a certain state can be then computed by computing the mean of all Q-functions for actions possible. Instead of taking the maximum of the best move, the mean is used to encounter cases where some unpredictable factors prevent the agent from going along with the best strategy. Additionally, a trade-off between the current reward and future reward can be achieved by multiplying the Q/V function with a factor between 0 and 1.

Above is the implementation for the discrete environment. Limitations exist because RL algorithms share the same complexity issues as other algorithms: memory complexity, computational complexity, and, in the case of machine-learning algorithms, sample complexity. [1] Deep learning can help solve the probelms.

Gradient descent is a fundamental optimization algorithm used extensively in machine learning and deep learning. It is a powerful method that enables models to learn and improve their performance by iteratively adjusting their parameters based on the observed error or loss.

By computing the gradients of the cost function with respect to each parameter, gradient descent determines the direction in which the parameters should be updated to reduce the error. The algorithm takes small steps in the opposite direction of the gradient, adjusting the parameters until it reaches a point where further changes yield diminishing returns.

There are two primary variants of gradient descent: batch gradient descent and stochastic gradient descent (SGD). Batch gradient descent computes the gradients over the entire training dataset in each iteration, which can be computationally expensive for large datasets. On the other hand, SGD updates the parameters for each individual training example, making it more computationally efficient but introducing more noise into the optimization process. Both methods are used in the following introduced papers.

2.2 Other Path planning algorithm

There are several algorithms that are frequently deployed to optimize the trajectory planning of the data sink in the WSN, such as Bayesian Optimization, Genetic algorithm, Artificial Bee Colony and etc. They all have their own advantage.

Artificial Bee Colonies (ABC) are nature-inspired optimization algorithms that mimic the foraging behavior of honeybees. They employ a population of artificial bees to collectively explore and exploit the search space, exchanging information to find near-optimal solutions for complex optimization problems.

Genetic mimic the process of natural selection to iteratively search for optimal solutions to complex problems. By representing potential solutions as individuals within a population and applying genetic operators such as mutation, crossover, and selection, genetic algorithms can effectively explore the solution space and converge towards optimal or near-optimal solutions.

Bayesian involves modeling uncertain quantities using probability distributions and updating these distributions as new data becomes available. Bayesian learning starts with prior beliefs about the parameters of a model and then updates those beliefs using Bayes' theorem, which incorporates the likelihood of the data given the model and the prior beliefs.

But RL's Adaptability to Dynamic Environments, Capability of handling Complex Action Spaces, Transferability, and Generalization make it especially outstanding. Adaptability to Dynamic Environments is especially important in multi-UAV systems where the agents themselves are the variance. As mentioned before, deep neural network can be applied RL so that it may adapt to a continuous environment which could be very space-consuming and computation-expansive when stored in a discrete state space form. Transferability means that the algorithm developed may be used in another environment without redesigning or retraining and etc. This is greatly represented in paper[2]. Generalization is the capability to infer a general valid policy from the limited training dataset,

which allows agents to act properly even in scenarios that are new to the algorithm.

3 Review Methodology

3.1 Article Retrival

In order to find papers with related topics, a set of search engines has been used to collect related literature from online resources, Google Scholar and IEEE Xplore are included. The Keywords used for search papers are "WSN", "RL", "UAV" and "Path planning".After dropping out not desired articles, there are only a few fulfilling all three requirements left.

3.2 Screening

This paper set several assessment criteria to refine the search results. The following constraints are applied to retrieved articles: 1. All articles must be written in the English language. 2. No duplicated research articles should be retained. Each article must focus on unique research work. For similar papers, only the one with most citations is kept. 3. The type of each article must be a review or a standard research article.

4 Review

4.1 Actor-Critic Deep RL: Bayesian Optimization Enhanced Deep Reinforcement Learning for Trajectory Planning and Network Formation in Multi-UAV Networks[4]

Goal To avoid the energy shortage problem for data transmission regarding the QoS requirement, this paper aims to minized the energy consumption by employing UAV in an optimal flying manner.

The authors consider the problem of network formation and trajectory planning jointly in a multi-UAV system where the connection between UAVs is possible so that information transition from distant drone to the base station is possible. They find the two problems tight related because for example when two UAVs fly far apart, their UAV-to-UAV(U2U) channel becomes deteriorated and even disconnected. So they proposed a two-step iterative approach to solve the joint problem: Firstly, they devise the adaptive network formation scheme by using a heuristic algorithm to balance the UAVs' energy consumption and data queue size. Then, with the fixed network formation, the UAVs' trajectories are further optimized by using multi-agent deep reinforcement learning without knowing the Ground User(GU)s' traffic demands and spatial distribution. Moreover, to improve the learning efficiency, they proposed an action estimation mechanism by using Bayesian optimization to estimate more rewarding actions for each

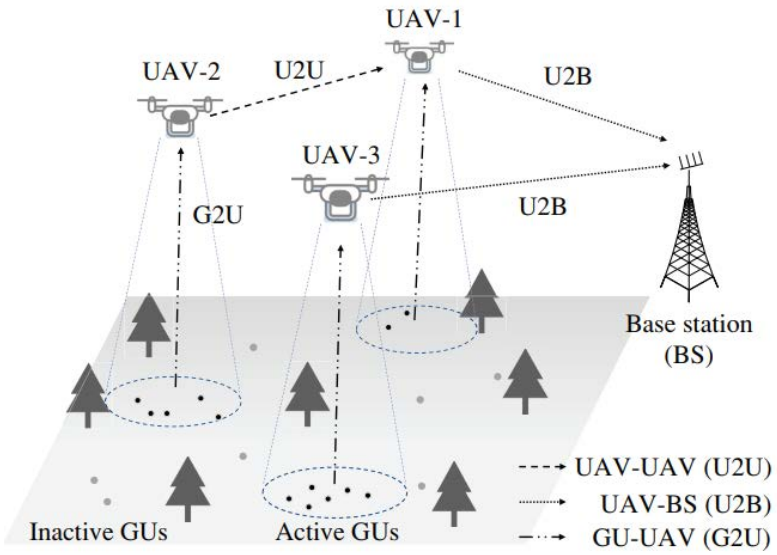


Fig. 1. Multi-UAV-assisted data offloading to the remote BS[4]

UAV. Based on the UAVs' past trajectories, the action estimation can avoid ineffective action exploration and potentially improve learning performance.

Model Time is divided into slots with unit length. One slot is subdivided into three intervals for flying, data sensing and offloading. While sensing has a fixed length, two other intervals can be jointly optimized. There are three types of communications: Ground-to-UAV, UAV-to-UAV and UAV-to-Base

Deep Reinforcement Learning is roughly implemented as followed:

Reward function depends on state and action. It is the sum of following parts: energy reward, transmission reward, sensing reward, and penalty for not keeping a safe distance.

State a joint of all UAV actions in this duration, its every element should contain that UAV's location, energy status, network formation, and buffer size

Action a joint of all UAV actions in this duration

The trajectory optimization problem is then handled in an actor-critic DRL framework using two sets of deep neural networks (DNNs) to approximate the policy and value functions respectively. The network is updated using Q-Learning, when the true Q-value of an action is not available, the critic network can give an estimation of the true reward.

The actor network is updated with gradient descent using the function for the summed Q-Value. While the actor network can be localized, the critic network would require global information and is therefore trained in the base station.

The Authors wanted to employ a multi-agent deep deterministic policy gradient(MADDPG) as the actor network to solve complex control problems in multi-agent systems. To do that, they faced two challenges:

- collecting all UAVs' observations and then adapting their flying actions jointly.
- the state and action spaces increase rapidly as the number of UAVs increases.

Therefore Bayesian Optimization is used to estimate the UAVs' optimal flying actions in the next time step. The multi-UAV trajectory planning problem is decomposed into N single-UAV trajectory planning problems based on local observations. For each problem, a function **f** mapping from location to collected data is defined. The goal is to build a probability model $\mathbf{P(f(l)|D(t))}$, where D(t) represents all collected data and location before time slot **t**, and l represents the current location.

Using Bayes rule, it is linear in $\mathbf{P(D(t)|f(l))*P(f(l))}$. $\mathbf{P(f(l))}$ is model with multi-variant Gaussian distribution, since the UAVs are unaware of the GUs' spatial distribution and their traffic demands. A **z** function is formulated to characterize the expected improvement of the function **f** value by computing the difference between current **f**-value and maximum of that from past samplings.

The critic network tries to minimize temporal difference error by the gradient descent direction.

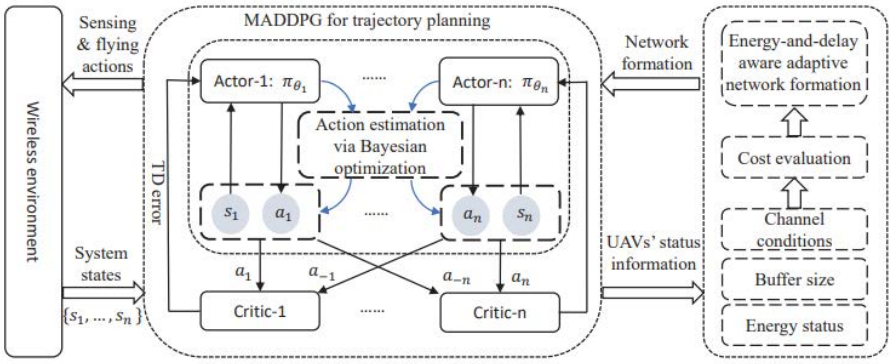


Fig. 2. The learning framework for joint network formation and trajectory optimization.[4]

4.2 Vanilla RL: Energy-efficient animal tracking with multi-unmanned aerial vehicle path planning using reinforcement learning and wireless sensor networks[3]

Goal The goal is to monitor wildlife species with the ability to observe key properties of animals such as the trend of animal movement and behaviors of animals within a herd. This is especially important for the study of some endangered animals where human intervention is almost impossible. The top priority of trajectory planning is to timely collect the information located in the sensor nodes, which is crucial for tracking the herd and analysing their habitat. Energy saving is placed in the second place.

Model The model can be described as a WSN-based network topology whereby a specific number of animals are equipped with a sensor node and a lightweight communication module (sensor node-enabled animal [SNEA]) is employed. The information is collected by the SENA, then sent to a local base station(BS), then collected by UAVs, which finally ensemble at the central BS.

This paper introduced a rare application of WSN in the context of animal tracking.

Fig. 3. Example of grid-based unmanned aerial vehicle-wireless sensor network model[3]

To predict more precisely, the large observation area is partitioned into small equal-length physical grids. A static BS is placed in each grid strategically to sense the animals in addition to collecting the data from relatively small WSN nodes fixed on trees, which provides scalability of the proposed approach.

Some challenges and their solutions:

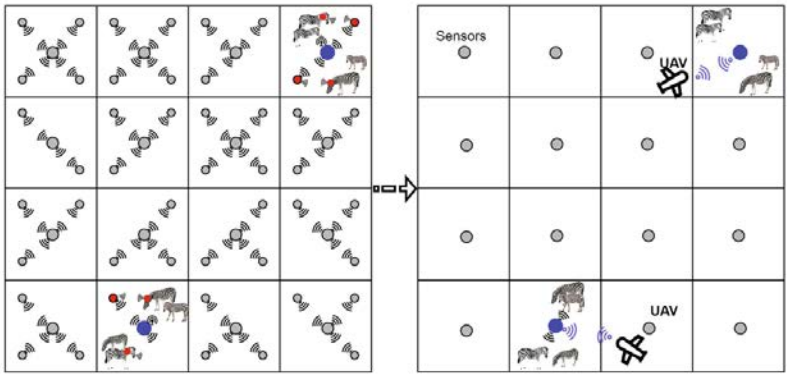


Fig. 4. Flow diagram of the whole operation for sensor node-enabled animal structure nodes[3]

limited energy resource for nodes on SENA a sleep mechanism is employed so that they only stay awake for some short and periodical time intervals for transmitting the data

when to transmit SNEA nodes follow a decreasingϵ-greedy strategy, where the possibility for transmission is positively correlated to its value. ϵ is normally at max, but starts to decrease when SENA stays in one location for too long and will be reset to max if it starts moving again. This also helps the UAVs to collect data more efficiently.

detect appearance of animals early Value-of-Information(VoL) mechanism is employed where the value of information is decreased as time pass by.

The Algorithm:

Reward function If VoL is greater than 0, then it is equal to that value plus a penalty for each of the other drones in this grid. Otherwise, it just gets a constant penalty.

State the coordinate of the drone plus its velocity.

Action each drone can move to all other neighboring grids.

4.3 Double Deep Q-learning and Combined Experience Relay: Multi-UAV Path Planning for Wireless Data Harvesting with Deep Reinforcement Learning[2]

Goal To harvest most data from Internet-of-Things(IoT) nodes in urban with UAVs using an algorithm that can without retraining adapt to changes in parameters that define the task (e.g., number of deployed UAVs, number,

position, and data amount of IoT devices). This is especially meaningful in real-life applications where even minor changes in the scenario would require repeating the full training process again.

Model The environment is modeled as a grid world with sets of designated landing positions, obstacles, and to-avoid locations (e.g. no-fly zone). The drone's position is modeled with its horizontal coordinate and altitude which can vary between drones but stay constant during the mission. The data collection mission is over after certain mission time steps for all UAVs due to energy constraints.

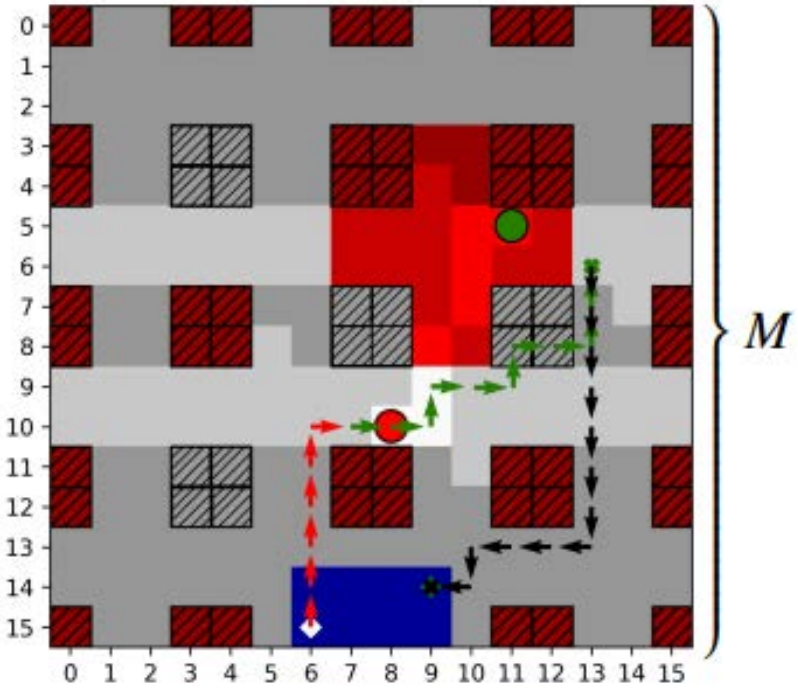
Allowed actions for drones are: hover, east, south, west, north, and land, and the drones are assumed to fly with constant speed throughout the mission(i.e. they can move only one gird per mission time step). If a drone lands, it stays still. Starting in the start position The mission is considered complete after all drones have landed in designated locations.

The path planning problem is translated into a decentralized partially observable Markov decision process (Dec-POMDP), which is then solved with a Deep Reinforcement Learning approach.

Algorithm:
MDP state space is the joint set of the following sets: Landing zones, No-fly zones, Obstacles, UAV positions, Flying times, Operational status, Device position and Device data
Additionally, a safety controller is introduced for avoiding collisions, if a possible collision is detected, the drone would stop the planned action and hover.
Reward function: Reward for data collected, Penalty for triggering safety controller, Penalty for not landing on time and Penalty for move
Algorithm:
Double Deep Q-learning and Combined Experience Relay are employed. One of the deep Q-network (DQN) is trained to minimize the expected temporal difference (TD) error. To prevent the deadly triad of function approximation, bootstrapping, and off-policy training from making its training unstable and causing divergence.

An experience relay mechanism is implemented as follows:
New experiences made by the agent are stored in the replay memory D. During training, a minibatch of size m is sampled uniformly from D and used to compute the loss. the latest experience the agent made is always added, so it is less sensitive to the batch size selection. The other network is for the estimation of the next maximum Q-value.

A multi-Agent Q-Learning is employed as follows:
Decentralized deployment with centralized training i.e. each agent has an individual but identical reward function. The experiences made by independently acting agents can be centrally pooled throughout the training phase While being recharged in BS, the UAVs would upload their experience data to a central



	Symbol	Description
DQN Input		Start and landing zone
		Regulatory no-fly zone (NFZ)
		Tall buildings* (UAVs cannot fly over)
		Small buildings* (UAVs can fly over)
		IoT device
		Other agents
		*all buildings obstruct wireless links
Visualization		Summation of building shadows
		Starting and landing positions during an episode
		UAV movement while comm. with green device
		Hovering while comm. with green device
		Actions without comm. (all data collected)

Fig. 5. Example of a single UAV collecting data from two IoT devices in an urban environment of size M × M with NFZs, a single start/landing zone, and buildings causing shadowing. Small buildings can be flown over and tall buildings act as navigation obstacles.[2]

server with larger memory and computation resources. And the central algorithm is trained with the latest collected information.

4.4 Pointer Network-A* Architecture: UAV Trajectory Planning in Wireless Sensor Networks for Energy Consumption Minimization by Deep Reinforcement Learning[6]

Goal Due to the limited energy source carried by UAVs, the service range of UAVs is constrained by the reality that they cannot travel very long distances or fly for long periods of time. Other than that, the battery life of sensor nodes in WSNs is typically limited, and in many cases, it is hard to regularly replace their batteries. Thus the authors of the paper aimed to design an algorithm that minimizes the total energy consumption.

Model Assume that devices on the ground have been clustered according to some specific criterion(which is not studied in the paper). Each Cluster contains a cluster head(CH), which will be selected by the proposed algorithm.

Only one rotary-wing UAV is dispatched to visit CHs to collect data from the ground network. After the UAV flies to the hovering position(above the CH) with a fixed speed, it hovers there and transfers a beacon frame to wake up the corresponding CH from sleep mode to active model.

Algorithm:
Since all clusters must be visited by the UAV sequentially, the visiting decisions problem is converted into a sequence-to-sequence prediction problem.

Then an A* Pointer Network is employed in which an A* search algorithm is used to find the a path with minimal energy consumption. The network is constructed with an Autoencoder model where an encoder and decoder are constructed with recurrent neural network(RNN), in this case, Long Short-Term Memory(LSTM) units to improve performance.

Encoder Start position or a cluster is converted into a high dimensional vector space. Then, the embedding vectors are fed into LSTM cells. At each encoding step, the LSTM cell reads one embedded item and outputs a latent memory state. Finally, the input sequence is transformed into a sequence of latent memory states.

Decoder The output of the encoder is given to the decoder network. At each decoding step, the LSTM cell outputs the hidden state, and the decoder employs the attention mechanism to output the visiting decision based on the hidden state and input.

A* Search build a search graph for all clusters according to the out sequence of the decoder, where each layer is composed of nodes of one cluster. The first and the last layer are both the start position.

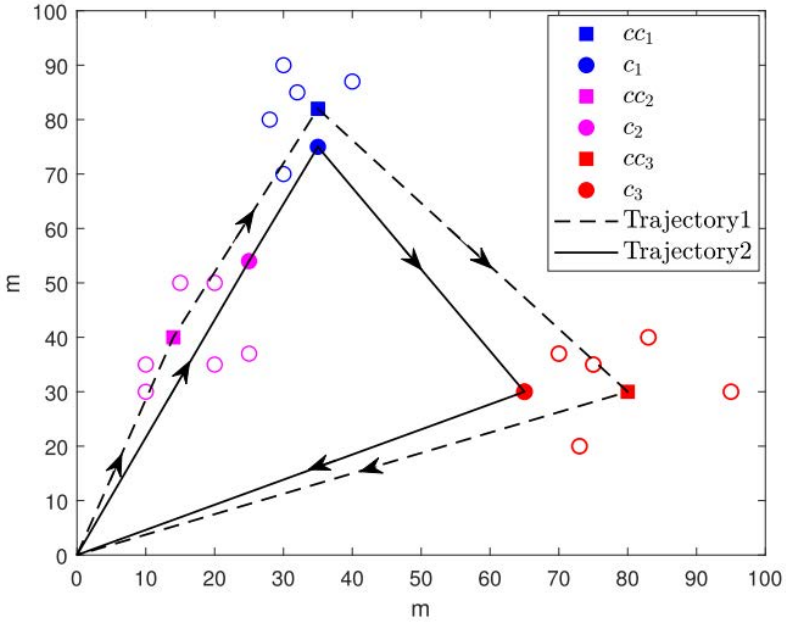


Fig. 6. Comparison of different UAV's trajectories[6]

The basic idea of the A* search is to find the estimated closest node to the destination with minimal cost to travel. Here, both the cost and the distance to the destination would be the energy consumption which is computed as follows:

Energy consumption computation At each hovering position, the energy consumption of the UAV includes two parts: communication-related energy and hover-related energy.

The energy consumption in the ground network includes two components. The first component is the communication energy consumption between CHs and their member nodes. The second component of the energy consumption of the ground network is the energy consumed by each CH to complete its data transmission to the UAV.

To optimize the parameter of the proposed network, actor-critic RL is employed. The actor here is exactly the proposed network and the critic has the same structure as the encoder of the Ptr-A* network. They use the policy gradient method and stochastic gradient descent to optimize the parameter of the actor(Ptr-A*), where the gradient is approximated with Monte Carlo sampling(take n samples and use the mean as the gradient). The parameters of the critic are trained with stochastic gradient descent on a mean squared error objective between its

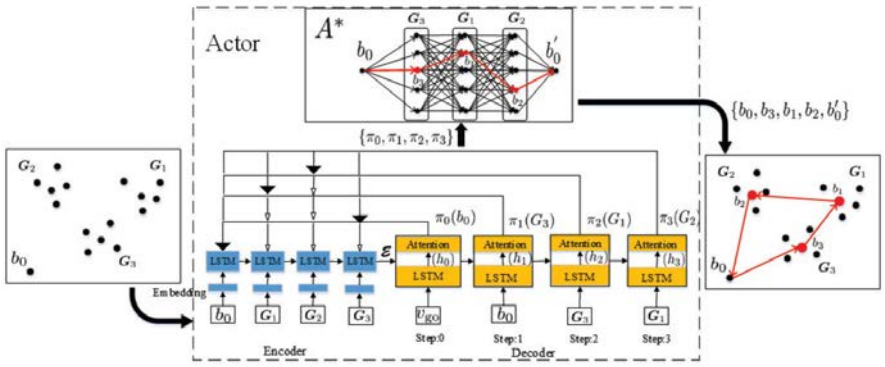


Fig. 7. Example of Ptr-A* architecture for a 3-clusters network[6]

predictions and the actual energy consumption. Finally, the Adaptive Moment estimator is applied to iteratively update both parameters.

4.5 Summary

In the first introduced paper, the authors decided to solve the joint problem of path optimization and path planning, which in most papers are solved separately. The problem occurs only when multi-UAV system is employed, which means a trade-off between complexity and efficiency. They therefore provide a new perspective of the problem. The Authors also employed a Bayesian optimization to estimate the position of the drones, which enhances the central training process that requires global information.

In the second paper, due to the requirement of monitoring the herd, there are plenty of BS placed across the whole area. Here the main task is to arrange the visiting order of the BS so that latest news may be delivered as soon as possible aside from minimization of energy consumption, making it outstanding from other papers.

Authors of the third paper claim their algorithm to be able to adapt to some degree of changes in the environment without retraining, which is quite remarkable. It shows the possibility of RL in generalization.

The fourth paper developed a transformer-alike approach by transforming the path planning problem into an sequence-to-sequence problem. RL in this paper serves as an unsuperivsed learning method to help the convergence. Other than ordinary Q-learning, this paper has a novel approach.

5 Conclusion

As can be seen above, there have been various models and implementations of RL in different scenarios which can adapt and have much better performance. RL is also capable of solving the generalized problem as proposed in [4].

As mentioned in [4], it is not often that the joint problem of trajectory planning and network formation. But considering the time efficiency of multi-UAV system could bring, the complexity added to the algorithm could be worthy. Other than that, UAV can do more than medium of information. Like in [5], the UAV also serves as a moving charger, which also requires a more delicated energy consumption strategy. This technology can for example be applied in [3]. Both of these two ideas can be future research direction.

References

1. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine **34**(6), 26–38 (2017). https://doi.org/10.1109/MSP.2017.2743240
2. Bayerlein, H., Theile, M., Caccamo, M., Gesbert, D.: Multi-uav path planning for wireless data harvesting with deep reinforcement learning. IEEE Open Journal of the Communications Society **2**, 1171–1187 (2021). https://doi.org/10.1109/OJCOMS.2021.3081996
3. Ergunsah, S., Tümen, V., Kosunalp, S., Demir, K.: Energy-efficient animal tracking with multi-unmanned aerial vehicle path planning using reinforcement learning and wireless sensor networks. Concurrency and Computation: Practice and Experience **35**(4), e7527 (2023). https://doi.org/https://doi.org/10.1002/cpe.7527, https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.7527
4. Gong, S., Wang, M., Gu, B., Zhang, W., Hoang, D.T., Niyato, D.: Bayesian optimization enhanced deep reinforcement learning for trajectory planning and network formation in multi-uav networks. IEEE Transactions on Vehicular Technology pp. 1–16 (2023). https://doi.org/10.1109/TVT.2023.3262778
5. Li, K., Ni, W., Tovar, E., Jamalipour, A.: On-board deep q-network for uav-assisted online power transfer and data collection. IEEE Transactions on Vehicular Technology **68**(12), 12215–12226 (2019). https://doi.org/10.1109/TVT.2019.2945037
6. Zhu, B., Bedeer, E., Nguyen, H.H., Barton, R., Henry, J.: Uav trajectory planning in wireless sensor networks for energy consumption minimization by deep reinforcement learning. IEEE Transactions on Vehicular Technology **70**(9), 9540–9554 (2021). https://doi.org/10.1109/TVT.2021.3102161

Path Planning in Wireless Sensor Networks with UAV-Based Data Gathering: A Literature Review with Genetic Algorithm Optimization

Daonan Zhang[1]

Karlsruhe Institute of Technology, Karlsruhe 76131, Germany
uqqww@student.kit.com

Abstract. The utilization of unmanned aerial vehicle (UAV) in data gathering within wireless sensor network(WSN) is gaining momentum, bringing significant advancements. However, considering the energy consumption and obstacles in specific environments, the complex path planning problem in UAV-based data gathering poses a challenge. This paper presents a literature review on the application of Genetic Algorithm (GA) optimization techniques to address path or trajectory planning in WSN with UAV-based data gathering. The review explores existing research studies, analyzing aspects such as problem and constraints formulation, encoding schemes, fitness functions, mutation operators. It discusses the advantages and limitations of different implementations of GA optimization in this context, identifies research gaps and challenges.

Keywords: Unmanned aerial vehicle(UAV) · Wireless sensor network(WSN) · Path planning heuristic · Genetic Algorithm (GA) · Literature review

1 Introduction

WSNs are interconnected sensor nodes that communicate in wireless method to collect data about the surrounding environment. Nodes are generally lower power and distributed in ad hoc, decentralized fashion. WSNs are mostly used in agriculture to monitor environmental conditions and control irrigation[1]. Recently, as enablers of many important applications, UAVs equipped with wireless communication platforms have attracted significant attentions[8]. When UAVs are applied to collect data in large-scale WSN, there are many challenges that affect the UAVs to manage WSN in large geographical areas in both academic and industrial fields to better utilize the use of UAV-assisted data collection in WSNs[2]. An illustration of the UAV-assisted data-gathering scenario is depicted in the Fig. 1.

However, WSN nodes are generally equipped with limited power resources that do not last long. Although the nodes are usually rechargeable, they are difficult to recharge in bulk[13]. As for UAVs, they require special equipment for charging, making their operations often "one-time", where they need to return to a depot for recharging after a mission. Considering the obstacles present in particularly harsh environments, path planning becomes exceedingly important.

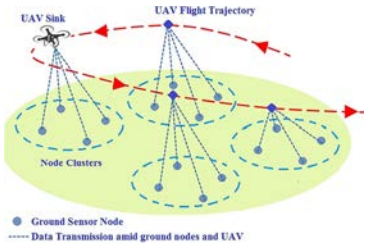


Fig. 1. Illustration of the scenario [13]

Given the mentioned constraints, the problem at hand naturally corresponds to the Traveling Salesman Problem (TSP). Nonetheless, it is important to note that the Traveling Salesman Problem (TSP) is known to be NP-hard, which implies that finding an optimal solution is computationally challenging. Therefore, in order to tackle this problem effectively, heuristic strategies are commonly employed.

2 Background

2.1 Heuristic strategies

In practical research, there are several heuristic strategies available in this context, the following four strategies are the most commonly used:

1. Genetic Algorithm (GA): GA is an evolutionary approach that uses genetic operators such as selection, crossover, and mutation to iteratively search for an optimal solution. It offers a balance between exploration and exploitation and is suitable for problems with complex search spaces.

2. Bayesian methods (Bayesian): Bayesian approaches utilize probabilistic frameworks to model uncertainty and update beliefs based on new information. They are effective in handling uncertain environments and can incorporate prior knowledge for improved decision-making.

3. Reinforcement Learning (RL): RL involves learning optimal policies through agent-environment interaction. It adapts to dynamic environments and can handle complex decision-making problems. RL algorithms learn from trial and error to maximize long-term rewards.

4. Artificial Bee Colony (ABC): ABC is inspired by the foraging behavior of honeybees. It utilizes the collective intelligence of a population of artificial bees to search for optimal solutions. ABC algorithms are robust, simple to implement, and suitable for continuous optimization problems.

Each of these heuristic strategies has its own advantages and limitations(Table 1). This literature review paper specifically centers around exploring and analyzing the use of Genetic Algorithm (GA) in the given context.

Table 1. Comparative Analysis of Four Path Planning Algorithms

Algorithm	Advantages	Disadvantages
GA	Effective for complex search spaces	Convergence to suboptimal solutions
	Ability to handle multiple objectives	Computationally expensive for large-scale problems
Reinforcement Learning(RL) (based on [3])	Adaptability to dynamic environments	Sensitivity to hyperparameter tuning
	Learns optimal policies through interaction	High computational complexity and training time
Bayesian Methods (based on [4])	Probabilistic framework for uncertainty modeling	Computational complexity increases with problem dimensionality
	Ability to update beliefs with new information	Reliance on prior assumptions and model selection
Artificial Bee Colony(ABC) (based on [9])	Simple and easy to implement	Limited global exploration capability
	Robustness against parameter settings and noise	Slow convergence for complex problems

2.2 Introduction of Genetic Algorithm

GA is a well-known algorithm inspired by the biological process of evolution. It mimics the concept of survival of the fittest in nature, and it was originally proposed by J.H. Holland in 1992. The fundamental components of GA include chromosome representation, fitness selection, and biologically-inspired operators.

The biologically-inspired operators consist of chromosome coding, selection, mutation, and crossover. Chromosome coding pertains to the systematic translation of solution components into a digital representation, commonly using binary or other encoding methods. In the selection process, chromosomes are chosen based on their fitness values for further processing. The crossover operator randomly selects a locus and exchanges subsequences between chromosomes to create offspring. In mutation, certain bits of the chromosomes are randomly flipped based on probability. The subsequent advancements of GA primarily focus on refining these elements, including operators, representation, and fitness.

A general framework of GA could be presented as Fig. 2. Multiple operators can be employed in each step.

3 Review methodology

This review methodology of this paper applies the strategy of [11], which can be divided into four phases.

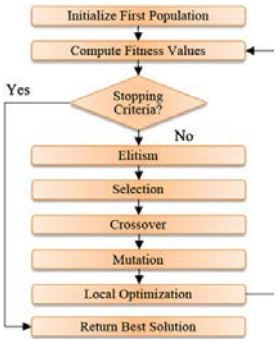


Fig. 2. The flow chart of GA[12]

3.1 Article retrieval

In order to find papers with related topics, a set of search engines has been used to collect related literature from online resources, Google Scholar and IEEE Xplore are included. Two groups of keywords referred to as G1, G2 are formulated for this purpose. G1 comprises keywords that primarily center around general path-planning algorithms in WSNs using the meta-heuristic approach of GA. The keywords used for this purpose include "WSN", "GA" and "Path planning". The keywords in G2 aim to narrow down the papers focused particularly on UAV-based data collection in WSNs so that G2 includes only the string "UAV". Furthermore, the "related work" part of several papers has also been used for retrieval of this paper. The total number of research papers retrieved is first 37.

Note: Some papers may not explicitly include the keyword "UAV" but might involve similar concepts or components, such as the utilization of a data mule.

3.2 Screening

This paper set several assessment criteria to refine the search results. The following constraints are applied to retrieved articles:

- The articles should be published on or after January 1, 2013
- All articles must be written in the English language only.
- No duplicated research articles should be retained. Each article must focus on unique research work.
- The type of each article must be a review or a standard research article. Other literature artifacts such as book chapters, letters, editorials, comments, or notes are excluded.

Only 20 out of the 37 articles are fitting these two constraints.
Note: 2. 3. 4. Constraints are exactly the same as the constraints in [11].

3.3 Content analysis

This paper is mainly focused on GA implementation in the context of path planning for UAV-based data collection in WSNs, so the articles should also focus on GA if more than one meta-heuristic method has been used for path planning. This rule also matches other keywords, if one of them is not found in the abstracts part and without any other similar characters, the article is dismissed. With these constraints, only 10 out of 20 articles still remain.

3.4 Categorization

After applying strict filtering based on all the constraints, it was found that all the articles met the specified conditions exceptionally well. However, considering the primary objective of this paper is to compare diverse implementations of GA across different phases, it was observed that certain articles exhibited remarkably similar implementations, making the comparison challenging. Hence, for the sake of clarity and conciseness, this paper only selects one representative article of them to exemplify such implementations in the analysis and discussion.

After the grouping process, only 7 out of the 10 articles remained for further analysis. It is worth noting that an additional literature review article specifically focused on GA(the paper [6]), which serves as the guiding source for this paper's "navigation," has been selected.

4 Comparative analysis

To better emphasize the distinctive characteristics of each paper and their variations in the specific implementation of GA, this study employs a progressive comparative approach. This approach entails two parts, namely the modeling and the diverse implementation methods adopted at each stage of GA.

4.1 Modeling

System model In the domain of path planning for UAV in WSNs, understanding and effectively modeling the system at hand is crucial for devising solutions. This comprehensive literature review explores the variations in system modeling strategies employed by different authors. this review aims to shed light on the strengths and limitations that arise from the varying modeling perspectives presented in the selected articles.

In [5] Grovind et al. have divided the sensor deployment area into same-size 3D grids/cells (SIG and flying cells), each Sensing Information Grid (SIG) may have a different number of sensor nodes, as depicted in Fig. 3, and the UAV follows a specific path through grid points to collect sensing information. A single UAV fly starts from the start node (one of the flying cells) and needs to fly through all SIG sell to collect data and then back to the start node. Due to the characteristic of each cell in the SIG grid having the same size, instead

of modeling as a TSP problem, they proposed scheme uses the DFS algorithm to find the optimal number of flying grids. This type of modeling renders the computation of fitness within class N; however, the observed disparity between the computed outcomes and actuality is a cause for concern.

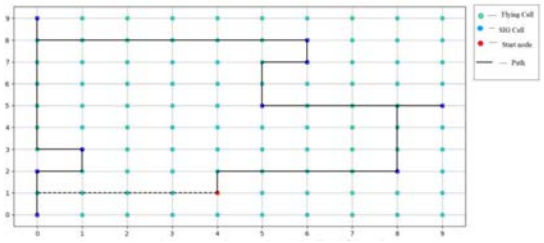


Fig. 3. SIG[5]

Grouping the nodes into multiple clusters is a widely adopted strategy for system modeling of such problems. Every cluster of nodes designates a suitable cluster member node as the cluster head (CH). The CH in a cluster gathers the packets from other cluster nodes. The CH further aggregates the collected packets into a single datum and transmits the datum to the sink[13].

In [7], the authors introduced a novel clustering approach. Unlike conventional methods that model the CH as a single node, the proposed approach considers the entire area of the cluster and treats each corner within this area as an individual node(see Fig. 4). This unique perspective transforms the path planning problem into a Traveling Salesman Problem with Neighborhoods (TSPN) formulation. This approach enhances the precision of the computational outcomes; however, it also significantly escalates the demand for computational resources.

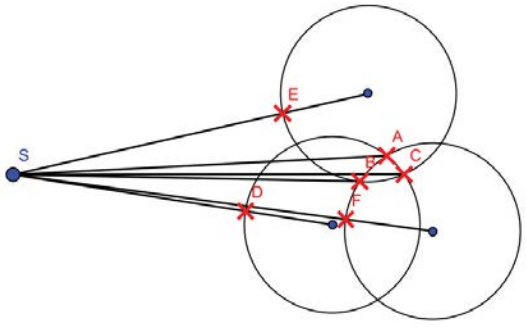


Fig. 4. Nodes in joint area [7]

In [12] and [2], the authors of the study incorporated the use of multiple UAVs in the mission, thereby formulating the path planning problem as a multiple Traveling Salesman Problem (mTSP). This methodology renders parallel computing feasible, aligning more closely with real-world scenarios. Regrettably, the author has omitted the utilization of heuristic algorithms specific to the Capacitated Vehicle Routing Problem (CVRP) for constructing initial solutions, a strategy that is particularly well-suited for models of this nature.

Table 2 presents a comparison of various papers employing different system formulation strategies.

Table 2. System Model

	Dimension	Node formulation	#UAV	TSP Type
[2]	2D	Cluster	multi	mTSP
[10]	3D	Cluster	single	TSPN
[5]	3D	Coordinates	single	DFS
[12]	semi-3D [1]	Cluster	single	mTSP
[7]	3D	Corner points in an area[2]	single	TSPN
[13]	3D	Cluster	single	MOOP-TSP

Objective function and Constraints The choice of objective functions and constraints is also vital for problem formulation and solution. This literature review examines how different researchers have approached and compared objective functions and constraints in their studies. By analyzing the variations in these components, this paper aim to gain insights into the trade-offs, strengths, and limitations of different approaches.

In [7], authors assume "for convenience's sake, the data mule has unlimited resources (like battery, memory) to collect all data from sensors in the whole network.", and all sensors deployed over a two-dimensional plane, the angle and speed of the data mule is not a factor considered in this study. Meanwhile, the authors also didn't study the energy consumption of WSNs or other communication details, the focus is on a linear optimization problem without constraints, with the sole objective of minimizing the path length. While this straightforward objective function benefits computational efficiency, disregarding the impact of energy consumption is deemed unacceptable.

In [2] Mesfin et al. proposed to minimize the path length with a limited energy budget of all sensors and UAVs, they proposed a UAV power consumption model based on the energy consumption of communication between UAVs and sink nodes, any solution over the total budget is infeasible. Instead of an energy budget, Ozgur set a minimum and maximum number of Control points(CP), which can be assumed as sensors or CH [Generation of Bezier Curve-Based Flyable Trajectories for Multi-UAV Systems with Parallel Genetic Algorithm] for a single UAV to make sure the energy limitation. but treat the CP as an area with an acceptable distance and angle from the CP instead of an exact point.

In the research paper [10], Ahmed et al. introduced an approach aimed at extending the network's lifetime, reducing energy consumption, and minimizing the path length. Their objective was to satisfy both the communication time constraint and the obstacle avoidance constraint while achieving these goals. They also considered the increased energy consumption by different angle changes, the increasing chance of turning angles of UAV for avoiding obstacles in the model makes this factor cannot be disregarded.

Govind et al. proposed that "the optimal path for UAV that need to minimize energy consumption, operational time and traveled distance and maximize information gain." [5]. They used Sensing quality, delay, and distance to quantify the fitness of one path.

They each offer distinct definitions of the objective function and extensively explore the impacts of specific factors on it from different angles. Unfortunately, within the references consulted for this paper, none have managed to integrate these factors to provide a more comprehensive computational approach.

Table 3 presents a comparison of various papers employing different objective functions and special constraints.

Table 3. Constraints

	Purpose	Special Constraints
[2]	Maximize the lifetime of UAVs and WSNs	
[10]	Minimize the energy consumption	Angle cost
[5]	Maximize the sensing Quality, minimize the delay and distance	Prohibited grid
[12]	Minimize the energy consumption	# Control points per UAV
[7]	Minimize the path length	Unlimited resource
[13]	Minimize the energy consumption, maximize the RSSI	

4.2 GA implementation

The following sections will present a comprehensive analysis of the implementation of GA from different articles, including the Chromosome representation and initial population, selection mechanisms, and the function to quantify the fitness of chromosomes, crossover, and mutation operators. By synthesizing and analyzing the information from these sources, this paper aims to provide a comprehensive overview of the different approaches and strategies employed in GA implementation.

Chromosome representation and initial population Each chromosome is a solution candidate for the system, and the set of these chromosomes constructs the population, which is also called a generation. In the evolution process, a new population is generated from the current one by using genetic operators. In the context of a path planning problem, a commonly adopted approach is to

represent each gene in the chromosome as an individual node, with the order of the genes reflecting the order of nodes in the path. An important consideration in this phase is to establish an appropriate population size. The size of the population is an important factor, and it determines the efficiency of the GA. If it is too small, the genetic algorithm can converge quickly to a local optimal solution and cannot reach a near-optimal solution. On the other hand, a too-large population may result in not finding a feasible solution in an acceptable time[12]. Meanwhile, depending on the system model, the representation of chromosomes can also be different.

Since multiple UAVs are applied in [2], the authors encode the chromosome as a set of paths for each UAV as Eq. 1. and the structure is shown in Fig. 5, where nodes divided into N groups, N is the size of UAVs, all of them start from the start point, and each node except start point appear only once. While in [10], each individual chromosome represents a complete UAV trajectory, which consists of a sequence of RPs (rendezvous points) and a sequence of IPs (intermediate traversal points),the genes encode the order in which the RPs are visited and the supporting points (IPs) between consecutive RPs to avoid obstacles. While in [13] chromosome consists of three genes, denoted as (Gi1, Gi2, and Gi3). These genes represent the random 3D coordinates of a tentative UAV Hover Point (UHP) location over the first cluster. This introduces greater flexibility but concurrently leads to an increase in computational time.

$$\#Gene = \#UAVs + CP - 1 \tag{1}$$

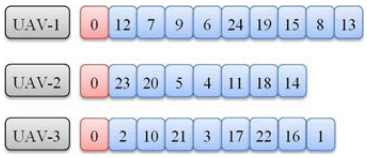


Fig. 5. Chromosome representation for multi UAVs [2]

In order to generate the initial population, researchers in the field have employed various methods. One such method is the Chromosome Generation Algorithm (CGA) in [7]. However, Random generation is more commonly employed. This entails randomly selecting nodes in the graph to serve as genes. In [12], Ozgur justified the utilization of random generation by stating, " Although, these methodologies can result in early converging to a feasible solution, their increased computation time is clearly seen as a disadvantage."

Table 4 presents a comparison of various papers employing different chromosome representations and initial population strategies.

Parent selection Parent selection in GA is the process of selecting individuals from the current population to be used as parents for generating the next gen-

Table 4. Chromosome representation and initial population

	Chromosome Representation	Initialization strategy
[2]	A Set of disjoint CHs	Without specific clarification.
[10]	RPs and IPs	Self-implemented random function
[5]	Flying cell	Total random
[12]	A Set of disjoint Control points	Total random
[7]	Corner points	CGA
[13]	3D-Coordinates	Self-implemented random function

eration. The purpose of parent selection is to guide the search towards better solutions by favoring individuals with higher fitness or promising characteristics.

In the selection operation, the most significant variations lie in how the population is grouped and how two or more individuals are chosen as parents for the next generation. Additionally, the consideration of applying elitism introduces another key distinction.

The authors of the [5] utilize tournament selection without elitism to enhance the exploration capability of the algorithm. This allows for a greater chance of discovering new feasible solutions and avoiding getting trapped in local optima. Conversely, the authors of the [5] paper argue that for their specific problem, elitism is preferred. By incorporating elitism, the best solutions from the previous generation are retained and carried forward, serving as a reference for achieving optimal solutions. The authors of the [7] on the other hand advocate for the use of Roulette Wheel Selection as the preferred parent selection method. They posit that Roulette Wheel Selection outperforms tournament selection in terms of convergence. Moreover, it's ease to implement and has the efficient time complexity of O(N), where N represents the size of the population.

In the context of multi-objective optimization, [5] and [13] adapt the NSGA-II algorithm to address the problem. One notable difference lies in how the population is divided into distinct Pareto groups. Each group consists of individuals that represent Pareto-optimal solutions for a particular objective function. By using NSGA-II, which incorporates non-dominated sorting and crowding distance, the population can be partitioned into multiple fronts based on their dominance relationships. Individuals within each front are considered Pareto-optimal with respect to their objective values. This grouping of the population into different Pareto groups allows for the identification of diverse trade-off solutions that span the entire Pareto front, providing decision-makers with a range of optimal solutions to choose from based on their preferences. The selection also only happens within the group or the front.

Table 5 presents a comparison of various papers employing different parent selection strategies.

Crossover After the selection of two-parent individuals in a GA, the subsequent operation involves modifying the genes of one or more children in the next generation. The purpose of this operation is to create offspring that can com-

Table 5. Selection operation

	Grouping	Selection strategy
[2]	Ordering base on fitness value	Only the individuals with the best fitness values
[10]	Without specific clarification.	Without specific clarification.
[5]	Pareto-Fronts	Tournament selection
[12]	Ordering base on fitness value	Tournament with Elitism
[7]	Ordering base on fitness value	Roulette wheel
[13]	Pareto-Fronts	Without specific clarification.

bine the favorable traits of both parents and potentially exhibit improved fitness values.

The Authors in [10] choose to use the SCX strategy to implement the crossover operation, which means each parent from its visiting order selects another RP not appearing in the child's chromosome currently. Then, the length of two path segments formed with the two selected RPs is compared and the better segment is added to the child's chromosome. An important aspect to highlight is the treatment of obstacles in the problem. Instead of using a simple feasibility check for solutions, the authors introduced IPs that enable UAVs to navigate through obstacles by adjusting their turning angles at these specific points. These IPs are dynamically generated, which means that the fitness value of an individual solution should be evaluated after the IPs are determined. Additionally, the authors incorporated a path-refining process, as depicted in Fig. 6, to improve the quality of the generated paths. This refinement step involves optimizing the paths to make them better solutions, and it should be considered in the determination of the fitness value. It is worth noting that these dynamic IPs and path refining process contribute to the complexity and effectiveness of the proposed approach in handling obstacles and generating high-quality solutions. The inclusion of these elements necessitates careful consideration during the evaluation of fitness values and further enhances the optimization process in solving the problem at hand.

In the paper [7], the authors explored various crossover operations to enhance the effectiveness of the path planning process. Specifically, they investigated the strengths of different crossover operators, including CSEX (Cycle-Based Sequential EXchange), CX (Cycle Crossover), MSCX (Modified Sequential Constructive Crossover), and SCX (Sequential Constructive Crossover). By considering these different crossover operations, the authors aimed to improve the exploration and exploitation capabilities of the genetic algorithm in generating high-quality paths. Each crossover operator has its own characteristics and mechanisms for combining genetic information from parent individuals to create offspring solutions.

In the paper [12], [5], and [13], the authors opted to utilize k-point crossover as the chosen crossover operation. Although the simulation results in the [7] section demonstrated that k-point crossover generally exhibits poorer convergence performance, it is important to note that genetic algorithms are not highly sen-

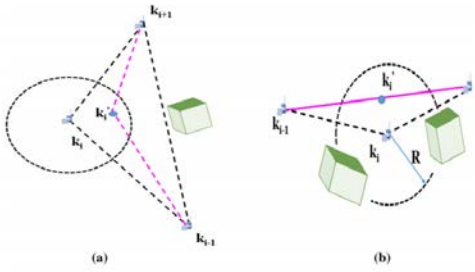


Fig. 6. path-refine process [10]

sitive to the specific type of crossover operation employed. Despite the potential limitations of k-point crossover, it remains a straightforward and easily implemented approach.

Table 6 presents a comparison of various papers employing different crossover strategies.

Table 6. Crossover

	Crossover strategy
[2]	Without specific clarification.
[10]	SCX
[5]	2-point crossover
[12]	1/2-point crossover
[7]	MSCX
[13]	1-point crossover

Mutation The mutation operation is also a probabilistic process in evolutionary algorithms for triggering diversity of the population. Mutation operation can extend the search space and avoid getting stuck in local optimum.[10]

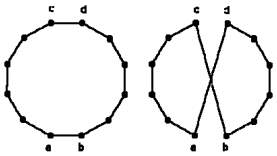


Fig. 7. 2-opt[12]

In the context of a TSP-like problem, the 2-OPT strategy is commonly employed as the mutation operation. Fig. 7 presented in the paper [7] illustrates the functioning of 2-OPT, which it involves swapping two genes (nodes) within an individual. Given the presence of multi-UAVs in the study described in [12], the authors decided to implement a 2-way mutation. This implies that the swap of nodes (genes) can occur either within a single UAV or across different UAVs in a solution set.

Table 7. Mutation

	Crossover strategy
[2]	Without specific clarification.
[10]	2-opt
[5]	2-opt
[12]	2-opt
[7]	2-opt
[13]	Random value mutation

5 Conclusion

In conclusion, this literature review provides a comprehensive analysis of different system models and implementations of genetic algorithms for the path planning of UAVs in WSNs. The following key insights can be drawn from the review:

Diverse System Models: The literature encompasses a range of system models, each offering unique perspectives on the problem of path planning. These models consider various factors and constraints, catering to different application scenarios. Selecting an appropriate system model is crucial for effective path planning and meeting specific requirements.

Diverse Implementations of GA: The reviewed literature showcases a wide range of approaches and strategies for implementing genetic algorithms in UAV path planning. These implementations differ in terms of encoding schemes, selection methods, crossover and mutation operations, and fitness evaluation techniques. Each implementation offers unique features and advantages, catering to different problem characteristics and optimization goals.

Though GA has proven effective in UAV-Aided WSNs path planning, current research still exhibits certain weaknesses. Future research can focus on optimizing GA parameters and adapting them to specific network configurations. Implementing improvements like stage-specific optimizations and parallelization, could lead to significant performance enhancements. Combining GA with other techniques, such as reinforcement learning, may further improve UAV path planning in dynamic environments.

References

1. Alkhatib, A.A.A., Baicher, G.S.: Wireless sensor network architecture. In: International conference on computer networks and communication systems (ICNCS 2012). vol. 35, pp. 11–15 (2012)
2. Betalo, M.L., Leng, S., Zhou, L., Fakirah, M.: Multi-uav data collection optimization for sink node and trajectory planning in wsn. In: 2022 IEEE 2nd International Conference on Computer Communication and Artificial Intelligence (CCAI). pp. 1–7. IEEE (2022)
3. Di Caro, G.A., Yousaf, A.W.Z.: Multi-robot informative path planning using a leader-follower architecture. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 10045–10051. IEEE (2021)
4. Gong, S., Wang, M., Gu, B., Zhang, W., Hoang, D.T., Niyato, D.: Bayesian optimization enhanced deep reinforcement learning for trajectory planning and network formation in multi-uav networks. IEEE Transactions on Vehicular Technology (2023)
5. Gupta, G.P., Chawra, V.K., Dewangan, S.: Optimal path planning for uav using nsga-ii based metaheuristic for sensor data gathering application in wireless sensor networks. In: 2019 IEEE international conference on advanced networks and telecommunications systems (ANTS). pp. 1–5. IEEE (2019)
6. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimedia tools and applications **80**, 8091–8126 (2021)
7. Liu, J.S., Wu, S.Y., Chiu, K.M.: Path planning of a data mule in wireless sensor network using an improved implementation of clustering-based genetic algorithm. In: 2013 IEEE Symposium on Computational Intelligence in Control and Automation (CICA). pp. 30–37. IEEE (2013)
8. Liu, J., Tong, P., Wang, X., Bai, B., Dai, H.: Uav-aided data collection for information freshness in wireless sensor networks. IEEE Transactions on Wireless Communications **20**(4), 2368–2382 (2020)
9. Poudel, S., Moh, S.: Hybrid path planning for efficient data collection in uav-aided wsns for emergency applications. Sensors **21**(8), 2839 (2021)
10. Pravija Raj, P., Khedr, A.M., Al Aghbari, Z.: Edgo: Uav-based effective data gathering scheme for wireless sensor networks with obstacles. Wireless Networks **28**(6), 2499–2518 (2022)
11. Qadir, Z., Ullah, F., Munawar, H.S., Al-Turjman, F.: Addressing disasters in smart cities through uavs path planning and 5g communications: A systematic review. Computer Communications **168**, 114–135 (2021)
12. Sahingoz, O.K.: Generation of bezier curve-based flyable trajectories for multi-uav systems with parallel genetic algorithm. Journal of Intelligent & Robotic Systems **74**, 499–511 (2014)
13. Singh, M.K., Choudhary, A., Gulia, S., Verma, A.: Multi-objective nsga-ii optimization framework for uav path planning in an uav-assisted wsn. The Journal of Supercomputing **79**(1), 832–866 (2023)

Quantifying gradients in deep Convolutional Neural Networks with the DoReFa-Net

Felix Ferber
Supervisor: Haibin Zhao

Karlsruher Institut für Technologie, 76131 Karlsruhe, Germany
uonzw@student.kit.edu

Abstract. In this paper we look into the work of DoReFa-Net. A method introduced to quantize gradient parameters of deep Convolutional Neural Networks with parameter of arbitrary bit-width. We ease the understanding for the proposed methods and ideas by providing the needed basic knowledge in this field in a brief manner, and explain the important related research which laid grounds for the DoReFa-Net. This paper is targeted for a broad audience with basic knowledge in computer science and aims to make the work of Zhou Shuchang and their team more accessible for students and those interested.

Keywords: Neural Network · Quantization · Review

1 Introduction

To introduce you to the general topic of DoReFa-Net [12] we need to provide some basic foundations. We dedicate this section to explain, in a brief manner, some needed basic knowledge. With the goal in mind, to not get distracted by confusing details later on.

We will start off by detailing some definitions and explanations regarding Neural Networks and explaining in that context the structure and generally neurons. We will dive into Convolutional Neural Network and deep Convolutional Neural Network, to understand the kind of Neural Networks DoReFa-Net is applicable to.

Following those explanations we will briefly explain the general training process and explain the terms forward- and backward-pass in this context. And in that manner we will also explain data sets and the important data sets mentioned in the works.

In the terms of Neural Network quantification we don't get around the terms size and computational complexity which we will explain in the last part of this section.

In the next Section we will use all those basics to introduce briefly some related and important works close to DoReFa-Net before we then detail the works of Zhou Shuchang and their team [12].

1.1 Neural networks

A Neural Network is a computational model inspired by the structure and function of the human brain. It consists of interconnected nodes, called neurons, organized in layers. Neural Networks are used for various tasks, such as pattern recognition, classification, regression, and optimization.

Neurons, synapses and gradients

In the context of Neural Networks, a neuron is a fundamental unit, that takes a group of weighted inputs, applies an activation function, and returns an output. Neurons are connected to each other through synapses, which can be thought of as roads in a Neural Network. Each connection between two neurons has a unique synapse with a unique weight and activation attached to it. Gradients are a fundamental concept in the training of Neural Networks. They represent the rate of change of a function with respect to the parameter weights of the network. Gradients are used in algorithms to update the parameters of the network.

Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specialized type of Neural Network designed for processing structured arrays of data, such as images. CNNs are widely used in computer vision tasks and have become the state-of-the-art for many visual applications, such as image classification and object detection.

Deep Convolutional Neural Networks

A Deep Convolutional Neural Network (DCNN) refers to a CNN with multiple convolutional layers stacked on top of each other. DCNNs have been shown to achieve better performance on complex visual tasks compared to shallow Convolutional Neural Networks with fewer layers.

1.2 Training

Training in the context of Neural Networks refers to the process of optimizing the model's parameters such that it can make accurate predictions on new, unseen data. During the training process, the model is presented with a set of labeled training examples and learns to adjust its parameters based on the errors it makes. The goal is to minimize the difference between the predicted outputs and the true outputs.

During training, the model goes through multiple iterations, also known as epochs, where it updates its parameters using an algorithm. The training process involves two main steps: the forward pass and the backward pass.

Forward pass

The forward pass, also known as forward propagation, is the calculation and storage of intermediate variables and outputs for a Neural Network. It involves processing the input data through the layers of the network, from the input layer to the output layer, to obtain the predicted outputs. Each layer performs a set of computations, typically involving matrix multiplications and activation functions, to transform the input data.

In the forward pass, the input data is multiplied by the weights of the connections between the neurons in each layer, and the resulting values are passed through activation functions to introduce non-linearity. This process is repeated for each layer until the final output is obtained. The intermediate variables and outputs calculated during the forward pass are stored for later use in the backward pass.

Backward pass

The backward pass, also known as backward propagation, is the process of calculating the gradients of the model's parameters with respect to the loss function. It involves traversing the computational graph in the reverse direction, from the output layer to the input layer, to update the parameters based on the errors made during the forward pass. These errors are represented as an error function or loss function.

In the backward pass, the gradients of the loss function with respect to the outputs of each layer are calculated using calculus. These gradients are then used to update the weights and biases of the model using an optimization algorithm like gradient descent. The gradients are propagated backwards through the layers of the network, hence the name "backpropagation".

1.3 Datasets

Datasets refer to a collection of data that is organized and structured for use in machine learning and data analysis. Datasets can vary in size and complexity, ranging from small datasets used for testing and experimentation to large datasets used for training deep learning models. Datasets can contain different types of data, including numerical, categorical, and text data. Datasets are crucial for building and evaluating machine learning models, as they provide the necessary information for the models to learn patterns and make predictions.

SVHN

SVHN [9] stands for Street View House Numbers. It is a real-world image dataset that consists of images of house numbers taken from Google Street View. The dataset is widely used for training and evaluating machine learning models for digit recognition tasks. Each image in the SVHN dataset contains a number from 0 to 9.

ImageNet

ImageNet [4] is a large-scale image dataset that contains millions of labeled images. It is widely used in the computer vision community for training and evaluating deep learning models. The dataset covers a wide range of object categories, including animals, vehicles, and everyday objects. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition that benchmarks the performance of models on ImageNet.

1.4 Complexity

In the context of Neural Networks, complexity refers to the computational resources required to train and run a Neural Network model. It involves measuring both the time complexity and space complexity of the algorithms used in Neural Networks. Time complexity refers to the amount of time it takes to train and make predictions with a Neural Network, while space complexity refers to the memory storage required during the training and inference processes. The complexity of a Neural Network is influenced by factors such as the number of layers, the number of neurons in each layer, and the size of the input data. Increasing the complexity of a Neural Network can lead to more accurate predictions but may also require more computational resources.

2 Recent efforts

We looked into the basic concepts for Neural Networks helping us to understand the DoReFa-Net. In this section we will briefly explain selected research topics preceding DoReFa-Net, which were crucial for the work of Zhou Shuchang and their team [12]. Beginning with the concepts of quantification and sparsification for reducing complexity of Neural Networks, followed by models which the DoReFa-Net was derived from. After that we will start introducing the concept of DoReFa-Net.

2.1 Quantification

Quantization in the context of reducing the complexity of a Neural Network involves representing the network's parameters and activations with reduced precision, such as using fewer bits to represent numerical values. By quantizing these values, according to the team of Wu [11], the memory requirements and computational complexity of the network can be significantly reduced. This can lead to benefits like faster inference, reduced energy consumption, and improved deployment on devices with limited resources, while still maintaining acceptable accuracy levels.

2.2 Sparsification

Sparsification in the context of reducing the complexity of a Neural Network involves inducing sparsity in the network's parameters or activations, meaning that a significant portion of the values become zero. By removing or "pruning" these zero values, the network's size and computational requirements can be reduced. Sparsification techniques, like in the work of Han and their team [6], aim to retain the most important connections or activations while discarding the less significant ones, allowing for more efficient memory usage, faster computations, and potential acceleration through sparse matrix operations.

2.3 AlexNet

AlexNet [8] is a CNN architecture that was designed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. It was the winning entry in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. AlexNet is considered a breakthrough in the field of computer vision and deep learning, as it demonstrated the effectiveness of DCNNs for image classification tasks.

2.4 Binary Neural Network

Binarized Neural Networks (BNNs), by the team of Courbariaux [3], are a type of Neural Network model that use binary values (usually -1 and +1) for both weights and activations, instead of traditional real-valued representations. This binary representation significantly reduces memory usage and accelerates computations by replacing costly multiplications with simple bitwise operations like XNOR and popcount.

In BNNs, the binarization process typically involves a step called "binarization function," which converts the real-valued weights and activations into binary values. This function can be deterministic, such as sign function, or probabilistic, where the probabilities of -1 and +1 are learned during training. During forward propagation, the binarized weights and activations are used for computations, and the results are accumulated using bitwise operations.

2.5 XNOR-Net

XNOR-Net [10] is a Neural Network model that extends the idea of BNNs by leveraging the XNOR operation. In XNOR-Net, both the weights and binary activations are binarized to -1 and +1, similar to BNNs, but the convolutional operations are replaced with XNOR and bit-count operations.

In XNOR-Net, the key innovation lies in performing convolutional operations using XNOR and bit-count operations. The multiplication operation between binary weights and binary inputs is effectively replaced with the simple bitwise XNOR operation, making the computations significantly faster. The result is then passed through a bit-count operation to accumulate the binary convolutional outputs.

3 DoReFa-Net

In this section we will discuss the DoReFa-Net, starting with the challenges, which were in mind of Zhou Shuchang and their team [12]. We elaborate on the proposed ideas in more detail leveraging the previous explanations. Experiments done in the scope of DoReFa-Net will be discussed briefly and we highly encourage you to look into the original work for more details in that regard.

3.1 Introducing DoReFa-Net

DoReFa-Net is a method for training CNNs with low bitwidth weights and activations using low bitwidth parameter gradients. This allows for significant reductions in the memory and computational requirements of CNNs, while still achieving comparable accuracy to 32-bit counterparts.

DoReFa-Net works by stochastically quantizing the parameter gradients to low bitwidth numbers before they are propagated to the convolutional layers. This is done during the backward pass of the training algorithm. As a result, the convolutions during the forward and backward passes can now operate on low bitwidth weights and activations/gradients respectively. This allows DoReFa-Net to use bit convolution kernels, which are much more efficient than traditional convolution kernels.

The main advantages of DoReFa-Net are its efficiency and accuracy. DoReFa-Net can significantly reduce the memory and computational requirements of CNNs, while still achieving comparable accuracy. This makes DoReFa-Net a promising approach for deploying CNNs on resource-constrained devices, such as mobile phones and embedded systems.

3.2 The contributions of DoReFa-Net

In this chapter we will list the contributions made by the paper of Zhou Shuchang and their team [12], without going into too many details. Subsequently we will start detailing the method of DoReFa-Net which includes detailed explanations to ease the understanding and provide the context for this work.
The paper on DoReFa-Net [12] detailed their contribution as follows:

- A generalized method of BNNs [3] such that it allows creating a DoReFa-Net, which is a CNN with customizable bitwidth for activation, weight and gradient respectively. This allows operating with low bitwidth kernels for accelerated backward and forward passes during training processes.
- Opening an accelerated way of training low bitwidth Neural Networks on Hardware like CPU, FPGA, ASIC and GPU. And even reducing costs by reducing considerably the energy consumption during training especially on FPGA and ASIC.

– Exploration and experiments on different bitwidth configurations of the weights, activation and gradient parameters for DoReFa-Net. Given the 1-bit weight, 1-bit activation and 2-bit gradient configuration which achieved 93% accuracy on the SVHN dataset.
– A release of a pre-trained DoReFa-Net model in TensorFlow [1], derived from AlexNet [8] trained on the SVHN [9] dataset.

Side note:
Field-Programmable Gate Arrays (FPGAs) are a programmable integrated circuit that can be configured after manufacturing, offering flexibility and adaptability for various applications. Application-Specific Integrated Circuits (ASICs) are a customized integrated circuit designed and optimized for a specific task or application, providing high performance and power efficiency. FPGAs can be reprogrammed and are therefore suitable for prototyping, rapid development, and applications requiring flexibility. ASICs are tailored for specific functions during manufacturing and cannot be reconfigured, offering optimized performance and power efficiency.

3.3 The formulation of DoReFa-Net

Now that we have introduced the idea of DoReFa-Net we will go into detail of the formulation. In this we will explain the bit convolution kernel, which is the base for the low bit operations of the parameters, in regards to the DoReFa-Net. We will detail how this is leveraged and explain the Straight Though Estimator and its use, before we go into detail of the quantization of each weight, activation and gradient parameter.

Following this chapter we will detail the pseudo code algorithm on how a DCNN could be trained using the DoReFa-Net method.

Bit convolution Kernels

Bit convolution kernels are convolutional filters that operate at the bit level. These small matrices help the Neural Network understand spatial patterns in the input. In these filters the convolution operation is performed on binary representations of the input data rather than the traditional numerical values.

In equation 1 we specify the 1-bit dot kernel. This can also be used to calculate the dot product in general and consequently convolution, for low bitwidth fixed-point integers.

$$\mathtt{x} \cdot \mathtt{y} = bitcount(and(\mathtt{x}, \mathtt{y})), x_i, y_i \in \{0, 1\} \forall i. \tag{1}$$

Which we are interested in because fixed-point number arithmetic reduces memory requirements and computational complexity compared to floating-point arithmetic. We can use this for more efficient hardware implementation, resulting in faster and more power-efficient inference. Additionally can we exploit fixed-point arithmetic for quantization of network parameters.

To show this Zhou Shuchang and their team [12] created the following example. Assume x is a sequence of M-bit fixed-point integers s.t. $x = \sum_{m=0}^{M-1} c_m(x)2^m$ and y a series of K-bit fixed-point Integers s.t. $y = \sum_{k=0}^{K-1} c_k(y)2^k$, where $c_m(x)_{m=0}^{M-1}$ and $c_k(y)_{k=0}^{K-1}$ are vectors containing only the number 0 and 1.

The dot product of x and y can be computed by the bit-wise operation as:

$$x \cdot y = \sum_{m=0}^{M-1} \sum_{k=0}^{K-1} 2^{m+k} bitcount[and(c_m(x), c_k(y))] \tag{2}$$

$$c_m(x)_i, c_k(y)_i \in \{0, 1\} \forall i, m, k. \tag{3}$$

The above equations have computational complexity on $O(NK)$ and are therefore directly proportional to the bitwidth of x and y

These equations extend the introduced 1-bit dot product kernel of equation 1 to a bit convolution Kernel for bit vectors of arbitrary length.

Straight Through Estimator

In order to represent real numbers with a set of finite numbers, we can naively achieve this with a step function, which maps numbers in specific ranges to a set value. Problems arise with this in the back propagation, simply because almost all sections of a stepping function have a gradient of zero.

To circumvent this problem DoReFa adopts the Straight-Through-Estimator method [7] [2]. This method allows DoReFa-Net to define arbitrary forward and backward operations.

The Straight Through Estimator extensively used is $quantize_k$, which is defined to quantize the real number input $r_i \in [0, 1]$ on an k-bit number output $r_o \in [0, 1]$.

The definition of the Straight-Through-Estimator is as follows:
Forward operation:

$$r_o = \frac{1}{2^k - 1} round((2^k - 1)r_i) \tag{4}$$

Backward operation:

$$\frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}. \tag{5}$$

The r_o in the forward operation is just the closest fraction to r_i in the form of $\frac{t}{2^k-1}$ for $t \in \mathbb{N}$ and $t < 2^k$. For the backward operation c denotes the function 4 from the forward pass, which makes $\frac{\partial r_i}{\partial r_o}$ not a differentiable function. But we leverage the fact the r_i is approximately equal to r_o and use the well defined gradient $\frac{\partial c}{\partial r_o}$ as an approximation for $\frac{\partial c}{\partial r_i}$.

Using this construction, we can represent a real number with k-bits. Also, with r_o being a fixed point integer, we can use the defined dot product function 2 to calculate the dot product of two such k-bit real numbers, by properly tweaking the equation.

Low bitwidth quantization

In this section we will look into the quantization of weights, activation and gradient parameters respectively. This will introduce the proposed method of gradient quantization. Each chapter and equation has explanations to ease the understanding. Each of the quantized parameters will use a Straight Through Estimator introduced in the previous chapter.

The quantization of the weight parameters

Previous work, like the BNN [3] or XNOR-Net [10] already introduced the binarization of the weight parameters using the Straight Through Estimator. In the BNN the Straight Through Estimator to binarize the weights looked like this:

Forward operation:

$$r_o = sign(r_i) \tag{6}$$

Backward operation:

$$\frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o} \mathbb{I}_{|r_i| \leq 1}. \tag{7}$$

Here the $\mathbb{I}_{condition}$ stands for the index function, which is defined to equal to 0 if the condition in the subscript is $false$ and equals 1 otherwise.

The function $sign(r_i)$ from equation 6 is defined as $sign(r_i) = 2\mathbb{I}_{r_i \geq 0} - 1$, which only returns the values $\{-1, 1\}$.

The XNOR-Net has a slightly different binarization equation for their weight parameters. The difference being scaling the weights after being binarized.
The Straight Through Estimator looks as follows:

Forward operation:

$$r_o = sign(r_i) \times \mathbf{E}_F(|r_i|) \tag{8}$$

Backward operation:

$$\frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}. \tag{9}$$

The function $\mathbf{E}_F$ in equation 8 is the absolute mean of all the input weights and used as the scaling factor. The reasoning for the decision to introduce a scaling factor is that it increases the value range for the weights while still exploiting bit convolution kernels.

This might seem like a preferable decision, in terms of DoReFa-Net however, it is left out. Solely for the fact that we find it impossible to exploit bit convolutions between gradient and weight in the back propagation.

We instead use constant scaling instead of channel-wise scaling. The Straight Through Estimator used for DoRoFa-Net binary weight parameters is as follows:

Forward operation:

$$r_o = sign(r_i) \times \mathbf{E}(|r_i|) \tag{10}$$

Backward operation:

$$\frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}. \tag{11}$$

Weight parameters which are not binary and instead use the k-bit representation for their weights, with $k > 1$, use the following Straight Through Estimator f_ω^k instead:

Forward operation:

$$r_o = f_\omega^k = 2quantize_k(\frac{tanh(r_i)}{2max(|tanh(r_i)|)} + \frac{1}{2}) - 1 \tag{12}$$

Backward operation:

$$\frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}. \tag{13}$$

Important to note is that the $\frac{r_o}{r_i}$ in 13 is well defined due to the used function in 12 being already the earlier defined Straight Through Estimator called $quantize_k$

In the function 12 the $tanh$ is used to limit the value range of the weight to $[-1, 1]$ before the quantization of the k-bit. The value of $\frac{tanh(r_i)}{2max(|tanh(r_i)|)} + \frac{1}{2}$ can only be in the range of $[0, 1]$, with the max-function taking the maximum over all weights of the current layer. The $quantize_k$ function, from equation 4, then turns this into an k-bit fixed point number in the $[0, 1]$ range. Finally we use the same transformation as the $sign$ function used in the definition from the equation 6 in 3.3, to transform the range to $[-1, 1]$.

We note as alternative way of binarizing weights by setting $k = 1$ in equation 12. Nonetheless Zhou Shuchang and their team [12] found the difference to be insignificant in their experiments.

The quantization of the activation parameters

In this section we will detail the approach to low bitwidth activations parameters which are input to the convolutions. This makes them crucially important when replacing the computational intensive floating-point operations.

In earlier research of BNNs [3] and XNOR-Net [10] the activation parameters were binarized in the same way as the weight parameters. This approach was found by Zhou Shuchang and their team [12] either non reproducible or bound to loss in severe amounts of accuracy especially when training against an ImageNet [4] models like AlexNet [8].

That's the reason why we use a Straight Through Estimator for input activations r of each weight layer. For this we assume the output of the previous layers to be bounded by their respective activation function h, which ensures us that the value of $r \in [0, 1]$. The DoReFa-Net quantization of the activations r to k-bit is then:

$$f_\alpha^k = quantize_k(r). \tag{14}$$

The quantization of the gradient parameters

Contrary to the deterministic approach of the activation and weights for the quantization, we find an stochastic approach to quantize the gradient parameter as necessary. As shown in the experiments of the Gubta and their team [5] already for 16-bit weights and gradients.

The quantization of gradients is different already in regard to the unbounded nature, therefore making the value range significantly greater than those of the activations.

Looking at Equation 14, we mapped activation simply onto $[0, 1]$, by using the earlier defined nonlinear differentiable function and passing our values through. Such a function, does not exist for gradients. For that reason we use the following function for the quantization of k-bit gradients:

$$\bar{f}_{\gamma}^{k} = 2max_0(|dr|)(quantize_k(\frac{dr}{2max_0(|dr|)} + \frac{1}{2}) - \frac{1}{2}). \tag{15}$$

In this equation it is $dr = \frac{\partial c}{\partial r}$, the back-propagated gradient of r for the corresponding layer. The maximum is taken over all axes of the gradient tensor dr, with exception for the mini-batch axis, so each mini-batch and it's axis will get its own scaling factor when applying this function.

The function above maps the gradient into the range $[0, 1]$, quantizes those values using the $quantize_k$ function and scales the function back to its original range.

This quantization can introduce strong biasing of the values and to compensate for that we introduce an additional noise function $N(k) = \frac{\sigma}{2^k - 1}$ where $\sigma \sim Uniform(-0.5, 0.5)$ which is the uniform distribution. It is noted that in this definition the endpoints of the uniform distribution are not clipped, due to the fact that those are almost never being attained.

This noise function has the same magnitude as the possible quantization error. In the experiments of Zhou Shuchang and their team [12], they found it to be critical to add this noise to achieving good performance. Adding this noise into the equation 15 we get the following:

$$f_{\gamma}^{k} = 2max_0(|dr|)(quantize_k(\frac{dr}{2max_0(|dr|)} + \frac{1}{2} + N(k)) - \frac{1}{2}). \tag{16}$$

Due to the Gradient only being quantized on the backward pass. We use the following Straight Through Estimator on the output of each convolutional layer:
Forward operation:

$$r_o = r_i \tag{17}$$

Backward operation:

$$\frac{\partial c}{\partial r_i} = f_{\gamma}^{k}(\frac{\partial c}{\partial r_o}). \tag{18}$$

3.4 The algorithm for DoReFa-Net

In this section we will see an example algorithm to train a DCNN with arbitrary bitwidth using the quantification methods introduced previously. Following that we will explain the workings of that algorithm a little more in detail, before we end this chapter with some exceptions in this method and a note on how to reduce run-time memory usage by fusing nonlinear functions with rounding.

Starting off with the algorithm taken from the original work from DoReFa-Net [12]:

Algorithm 1 Training a L-Layer DoReFa-Net with W-bit weights and A-bit activations using G-bit gradients. Weights, activations and gradients are quantized according to equation 12, 14 and 16, respectively

Require: a minibatch of input and targets (a_0, a^*), previous Weights W, learning rate η

Ensure: updated weights W^{t+1}

{1. Computing the parameter gradients:}

{1.1 Forward propagation:}

for $k = 1$ **to** L **do**
  $W_k^b \leftarrow f_\omega^W(W_k)$
  $\widetilde{a}_k \leftarrow \texttt{forward}(a_{k-1}^b, W_k^b)$
  $a_k \leftarrow h(\widetilde{a}_k)$
  if $k < L$ **then**
    $a_k^b \leftarrow f_\alpha^A(a_k)$
  end if
  Optionally apply pooling
end for

{1.2 Backward Propagation:}

Compute $g_{a_L} = \frac{\partial C}{\partial a_L}$ knowing a_L and a^*.

for $k = L$ **to** 1 **do**
  Back-propagate g_{a_K} through activation function h
  $g_{a_k}^b \leftarrow f_\gamma^G(g_{a_k})$
  $g_{a_{k-1}} \leftarrow \texttt{backward\_input}(g_{a_k}^b, W_k^b)$
  $g_{W_k^b} \leftarrow \texttt{backward\_weight}(g_{a_k}^b, a_{k-1}^b)$
  Back-propagate gradients through pooling layer if there is one
end for

{2. Accumulate the parameters gradients:}

for $k = 1$ **to** L **do**
  $g_{W_k} = g_{W_k^b} \frac{\partial W_k^b}{\partial W_k}$
  $W_k^{t+1} \leftarrow \texttt{Update}(W_k, g_{W_k}, \eta)$
end for

To ease the understanding we will recap the algorithm in more details here. This is purely to ease understanding this simple, but dense algorithm.

To begin in the forward propagation, we assign values to the parameter weights vector of this minibatch using the quantization function from equation

12 (`line2`). We calculate the new activations using an arbitrary `forward` function(`line3`), which now can operate on low bitwidth and fixed-point integers. Followed by assigning the new activation value after passing through the activation function h (`line4`) and finally the quantization of the activations of the layer using the function from equation 14 (`line6`). Important to note is that the layer 0 and L will never be quantized here, but we will come back to this in the next chapter.

The backward propagation starts with calculating the first gradient of the layer. Using the resulting gradient we start the loop with back propagating it through the activation function h (`line11`). We quantize the result using the quantization function from equation 16 and assign it to the gradient for this minibatch(`line12`). Then we use arbitrary $\texttt{backward}_{\texttt{i}}\texttt{nput}$ and $\texttt{backward}_{\texttt{w}}\texttt{eight}$ to firstly get the gradient for the next iteration (`line13`) and get the gradient which we need to update the weights parameters(`line14`).

In the last loop we collect the gradients (`line18`) from the minibatches and apply the arbitrary `Update`() function to get our updated weights parameters.

Computational intense are the arbitrary functions `forward`, $\texttt{backward}_{\texttt{i}}\texttt{nput}$ and $\texttt{backward}_{\texttt{w}}\texttt{eight}$, which all possibly are mapped on low bitwidth numbers and fixed-point integers with affine transformations. Which means those expensive operations are now accelerateable significantly by the fixed-point integer dot product kernel 2.

The First and last layers

As the previous explanation suggested, in a DCNN, the first and last layers appear to be different from the rest. As they are the layers interfacing the input and output. For the first layer it is often an image which might contain non redundant features in 8-bit or higher than the bitwidth of the features. As for the outputs from the last layer, they produce one hot vector. Which are $1 \times N$ vectors with a singular 1 and 0's on every other position, those are used for mapping to a vocabulary from a table. By this definition they are close to bit vectors and converting those layers to low bitwidth counter parts is left open in the work of DoRoFa-Net.

Arguments for not reducing the computational complexity in those layers are mainly based on observed degradation of accuracy, while looking into the sparsification of the convolutions in a Neural Network by Han and their team [6]. This observation and the fact that for a DCNN the first and last layer alone are a small fraction of the computational complexity, the quantization is left open in the experiments done by Zhou Shuchang and their team [12].

However even though the input of the first layer and the output of the last layer are fully connected. The parameters used by the preceding and following layers are again quantized by definition of the algorithm.

Reducing Run-time memory usage

Reducing Memory during run-time is one of the motivations for creating a low bitwidth Neural Network and is, as shown earlier, possible. Nevertheless we have to point out that a naive implementation of Algorithm 1 would store activations from the activation function $h(a_k)$ in full-precision and therefore use a lot of memory during run-time. Especially when using floating-point arithmetic, which also would introduce a lot of computational complexity back into the algorithm, due to it's non-bitwise nature.

To avoid this we can fuse `line3`, `line4` and `line6` with the goal in mind not storing the intermediate floating-point results in full precision, which would inevitably lead to non-bitwise arithmetic. For that we have to point out the fact, that if the function h is monotonic, the function $f_\alpha \cdot h$ is also monotonic. Which again implies that the few values for a_k^b, which in of themselves are non-overlapping, correspond to also non-overlapping ranges of a_k. Keeping this in mind, we can implement $a_k^b = f_\alpha(h(a_k))$ with computational simple fixed-point number comparisons, resulting in the desired avoidance of intermediate floating-point results.

Similarly can we fuse `line11`, `line12` and from the preceding iteration `line13` to again avoid the intermediate result of g_{a_k}, even though more complex when there are intermediate pooling layers, we can do the fusion nevertheless. As the function $quantize_k$ commutes with the max function, such that

$$quantize_k(max(a, b)) = max(quantize_k(a), quantize_k(b)) \tag{19}$$

implies that $g_{a_k}^b$ can be generated from g_{a_k} only using fixed-point number comparisons.

3.5 Experiments

In this chapter, we provide a brief overview of the experiments conducted by Zhou Shuchang and their team [12] and summarize some notable results. We strongly recommend referring to their work for more comprehensive details and data related to these experiments.

The experiments involved multiple sets of data with varying configuration spaces for the models, where each configuration represented different bitwidths of weight, activation, and gradient parameters. The first set of configurations focused on accuracy, storage size ranking, and complexity ranking for different types of configurations. Each configuration was trained on four models with varying channel counts.

For this set of configurations, the SVHN [9] dataset was utilized. We observed that the accuracy degradation was not significant for model A, which had full connectivity in all seven layers, when the gradient's bitwidth (G) was greater than or equal to 4. However, there were exceptions to this trend, as seen in the configuration data. For instance, the (1, 1, 2) configuration achieved accuracies of approximately 93.4%, 92.4%, 91%, and 80.3% for models A, B, C, and D,

respectively. Conversely, the (1, 4, 2) configuration, which would suggest higher accuracy with a higher bitwidth for one parameter, achieved accuracies of 81.5%, 89.8%, 91.1%, and 86.8% for the same models.

In general, we observed a trend where models with greater connectivity between layers tended to have higher accuracy. Additionally, increasing the gradient bitwidth led to improved accuracy, but the improvement became insignificant after 8 bits.

The second set of configurations utilized the ILSVRC12 [4] image classification dataset, which contains around 1.2 million natural images from approximately 1,000 object categories. Here, the focus was on different gradient bitwidths for 1-bit weights and activation with bitwidths of $A \leq 4$. We observed a clear improvement in accuracy with higher bitwidths and even similarity in accuracy between configurations with 32-bit and 6-bit gradients.

We recommend referring to the work of Zhou Shuchang and their team [12] for more detailed explanations and extensive data on these experiments.

4 Conclusion

In conclusion, the paper by Zhou Shuchang and their team [12] presents a highly impactful work that introduces a well-formulated and easily understandable idea. The paper effectively contextualizes the provided information and delivers a clear message. With over 2,000 citations to date, this work is widely recognized and highly influential in the field. We strongly recommend reading the original paper if you find the topic intriguing.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
2. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
3. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1 (2016)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
5. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. In: International conference on machine learning. pp. 1737–1746. PMLR (2015)
6. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding (2016)
7. Hinton, G., Srivastava, N., Swersky, K.: Neural networks for machine learning (2012)

8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks (alexnet)
9. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
10. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks (2016)
11. Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J.: Quantized convolutional neural networks for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4820–4828 (2016)
12. Zhou, S., Ni, Z., Zhou, X., Wen, H., Wu, Y., Zou, Y.: Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. CoRR **abs/1606.06160** (2016), http://arxiv.org/abs/1606.06160

Proseminar Wide Reduced-Precision Networks

Jan Langbecker
Supervisor: Haibin Zhao

Karlsruher Institut für Technologie, 76131 Karlsruhe, Germany
jan.langbecker@student.kit.edu

Abstract. One major problem of neural networks that often hinders them impractical outside of an experimental and scientific context are their resource requirements both during training and when applying the model after Training. There are many papers about techniques that aim to lessen some of the resource requirements of neural networks, many do so by quantising them thereby reducing the precision or limiting the range of input data and weights to the neural network when running the model.
One such quatisation technique is called "Wide Reduced Precision Networks"[1]. This paper compares WRPN to other similar techniques [7][3][8] in terms of their effectiveness in achieving their respective goals of reducing resource consumption. And explains key concepts and terms required to understand these techniques.

1 Introduction

Motivation In recent years there has been a trend of drastically increasing the size of the input layer of neural networks as well as their depth - especially with convolutional neural networks used in tasks like computer vision. This causes a massive increase in memory requirements and computational requirements to run and train such neural networks as well as storage requirements to store such a neural network. This causes the issues of such neural networks being difficult to deploy in the real world[10] as well as them being expensive to train.

One proposed solution to this problem is network quantisation: it aims to reduce those requirements while having a limited impact on the accuracy of neural networks. In recent years a lot of different quantisation techniques, each with their own set of advantages and disadvantages have been introduced. One such technique are Wide Reduced Precision Networks [1] which tries to overcome the main issues of previous quantisation techniques: Mainly them either having a major reduction in resource requirements or minimal loss in accuracy but not both.

In this paper we will explain terminology and concepts required to understand WRPN and quantisation in general, the quantisation technique used in WRPN and the quantisations schemes used in other similar techniques. We will also

compare WRPN to other previously used quantisation techniques on which WRPN are based.

2 Basic Terminology and Concepts

2.1 Neural networks

A neural network is a mathematical model that maps input(s) to a set of output values. It consists of 3 types of layers: An input layer containing the input value(s) to the neural network, an output layer containing the output values and at least one hidden layer, each consisting of nodes (values), and connections between nodes of the current layer and the nodes of the next layer, each connection having an associated weight. Those weights are also referred to as model parameters.

The neural network calculates the values of one layer based on the values of the previous layer and the weights of their connections. Most of the time this is done by simple multiplication and accumulation or in other words matrix multiplication, however other types of functions such as convolution can be applied as well. This is done for each layer until the output layer is reached, the values that are calculated as values of the output layer are the outputs of the entire network.[4]

2.2 Convolutional Neural Networks

Convolutional Neural Networks or CNN's are a type of neural network in which some layers calculations are not performed by the usual matrix multiplication used in general neural networks, but instead by performing convolution. This proved beneficial in a lot of classification tasks - especially in image recognition and natural language processing.[9]

2.3 Convolution

The discrete convolution of two complex-valued functions is defined as [2]:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

For our intents and purposes the two functions f and g map complex coordinates to one component of the inputs of the first function and to the weights of the layer for the second function.

2.4 Backpropagation

Backpropagation is the process used to improve the neural network during "training". It aims to adapt the weights of a layer to reduce the mean error of the network. This reduction of the mean error of the network is usually achieved using gradient descent.

2.5 Quantisation/Quantized Neural Networks

Quantisation is the process of mapping a larger input set to a smaller output set. The output often has a finite number of elements. In the context of neural networks quantisation can be applied to either input values, output values, network weights or any combination of these for any of the layers of the neural network.

"While it is possible to train a quantized [neural network] from scratch, the majority of research has shown that starting with a pre-trained model to be more effective at minimising accuracy loss when quantizing a [neural network]"[10]. This means that quantisation is almost always applied to pre-trained models (Meaning that the Network already has been trained using training data). Quantisation is usually either applied as so called quantisation-aware training or as post-training quantisation. While post-training quantisation involves just the quantisation of the network itself, quantisation-aware training also involves additional training after the quantisation "so that [the neural network can] adjust [..] to the newly quantized values."[10]

The goal of quantisation is to reduce the storage and computational requirements of neural networks at the cost of a loss in precision of the output layer. [10] Good quantisation techniques have a very limited precision loss impact on the output, while achieving a major reduction in either storage requirements or computational requirements or both.

A neural network with quantisation is also referred to as a quantized neural network. Quantisation can be achieved using a variety of techniques, each with their own output set and specific advantages and disadvantages.

2.6 The Clipping Function

One very important and often used function for the quantisation of a neural network is the clip-function [7]:

$$clip(x, u, l) = max(l, min(u, x)) \tag{1}$$

The clip function limits the set of a value of x to a range between an upper bound u and a lower bound l by "clipping" values above u to u and values below l to l or in other words: Replacing values larger than u with u and values smaller than l with l.

2.7 Reduced-Precision Networks

Reduced-precision networks are a special kind of quantized neural network, where the output set of the quantisation is finite and set up in a way causing the individual output values to be less precise than the input values. This is for example the case when mapping floating point values to fixed point values or integers. Or when limiting the range of possible integers, e.g. when applying the clipping function. [10]

2.8 Filter Maps

Filter maps are convolutional layers of a neural network that act as filters. They are often applied multiple times in succession and often there are multiple different filters applied after each other. They differ from general convolutional layers because they are not fully connected meaning not every input influences every output and some outputs may not even be connected at all. Filter maps enable the input data to conventional layers in convolutional neural networks to be more usable since after applying a good filter the parts of the input data relevant to the classification will be more highlighted, while those data parts less important will be less dominant in the overall result.

2.9 Straight-Through Estimator

If the weights of a neural network are quantized in a way that limits the amount of possible values to a set of finite values, we got a problem when trying to train the network e.g when applying quantisation-aware training [10]:
There is no meaningful gradient function that we can apply during backpropagation to "train" the network. The straight-through estimator as explained in [11] is a solution to this problem. It is an estimated continuous gradient function which can be used to apply gradient descent and train the network.

2.10 AlexNet

AlexNet is a specific eight layers deep convolutional neural network originally trained for the ImageNet competition on image recognition which it subsequently won. It has since them been trained on different datasets and used in research papers as one example the demonstrate the effect of techniques introduced in the papers.[6]

2.11 ResNet

A ResNet or a Residual Neural Network is a specific kind of neural network which has some skip connections that "bypass" some of the layers of the neural networks, this means the for some values the original value is added to the output of one or more subsequent layers before passing the result of the addition

as an input to the next layer. They were introduced after the winner of the next ImageNet competition after AlexNet won by including more layers - which causes issues because errors in one layer can compound with each following layer. ResNet's won the following ImageNet competition.[5]

3 Wide Reduced-Precision Networks

3.1 Explanation

Wide reduced-precision networks or WRPN are one proposed solution to the major resource requirements of modern neural networks, and more specific convolutional neural networks, during execution as well as to storing the model. The main issue is that modern neural networks use a large input layer and high precision values in all layers, causing the storage requirements to store the model parameters to be very large.

WPRN's aim is to reduce these requirements like other reduced-precision networks by limiting the range and precision of weights of the neural network. Additionally different from other reduced-precision networks, WRPN also quantize the activations which aims to further reduce the computational requirements of the network. Since this causes a lot of precision loss in the output layer compared to a conventional network WRPN include additional filter maps to compensate for this loss.

"Although the number of raw compute operations increases as [...] the number of filter maps in a layer [is increased], the compute bits required per operation is now a fraction of what is required when using full-precision operations"[1] This enables the use of cheaper and more scalable hardware, which means that WRPN can still be less computationally expensive then conventional full precision networks. WRPN requires some training for the additional filter maps, this can be done before quantizing or by applying quantisation aware training.

WRPN main contribution is the fact that it enables the usage of neural network quantisation with all its benefits while being able to mitigate most of, or in some cases even all the downsides that usually arise when applying quantisation like a large loss in accuracy. This replaces the optimization for minimal loss in accuracy after applying 'traditional' quantisation techniques with the optimisation of the reduction of the computational cost of the neural network. Also it introduces the downsides of having to optimise for each data set individually when applying the procedure, and the possibility of higher computational costs than the non-quantized Network when improperly optimising for the data-set.

3.2 Quantisation Scheme

WRPN quantisation scheme works differently for weights and activations. It's quantisation scheme for weights it is given by:

$$w_q = \frac{1}{2^{k-1}-1} round((2^{k-1}-1) * clip(w, 1, -1))$$

Where w_q is the quantized weight, w the original weight, *round* is rounding half down and *clip* is the clipping function (1).

The quantisation scheme applied to input (activation) values is:

$$a_q = \frac{1}{2^{k}-1} round((2^{k}-1) * clip(a, 1, 0))$$

Where a_q is the quantized input value, a the original input value, *round* is rounding half down and *clip* is the clipping function (1).

Because this quantisation scheme is very general which might result in a compounding error due to the increased number of filter maps, which results in worse accuracy of the network than just applying quantisation. [1] circumvents this issue by trying several different combinations of the 3 variables increase in filter maps, bit-length of weights and bit-length of activations.

The upper bounds for these values are indirectly set by the computational cost of the original network (if you include too many filters while not decreasing the bit-lengths enough the performance of the WRPN can be worse than the original network - in which case the whole process was useless). And the lower bounds are directly set by the original amount of filter maps and 1 bit weights and 1 bit activations. This is the lower bound because with less filter maps then the original network the accuracy of the network would suffer even more due to the quantisation then just quantizing the network and when using 1 bit weights/activations we effectively perform binary connect and we can't reduce the precision of weights anymore then binary connect does.

This means that the optimal bit-lengths and scaling factor for filter maps might differ greatly between different data sets which results in a lot of experiments being necessary to apply the WRPN technique effectively.

4 Other Quantisation techniques

The idea of reducing precision of a neural network by quantizing it is not unique to WRPN. This section will try to give an overview of some of the techniques on which WRPN are based. These techniques are similar to WRPN since they limit the precision of network weights by quantizing them, however WRPN uniquely also quantize the input layer, and include extra layers to compensate for that.

4.1 Binary connect

Similar to WRPN, neural networks using BinaryConnect(BC)[7] also reduce the precision of the calculations. However BC's main focus is decreasing computational cost and storage requirements for the model when running it - not during training. BC abuses the fact that the main computationally costly operations during both backward and forward propagation of neural networks are convolutions and matrix multiplications, both of which require a lot of multiply-accumulate operations[7].

BC achieves a significant reduction of these costs by limiting the space of possible weight values of the DNN to {-1,1}. Which makes it possible to replace a lot of multiplications with simple additions and subtractions and weights can now be stored in just a single bit per weight. This enables the use of simpler and easier to scale hardware to compute the neural network. The paper[7] mentions the use of fixed-point adders instead of multiply-accumulators as an example for this.

The major benefit of the simpler hardware required to compute the neural networks using BC is the fact that it is cheaper to run in a deployment. Or a larger model can be run at the same cost as a neural network not running BC. However these benefits come at a significant loss of precision of the neural network.

Quantisation Method: The binary weights for BC are computed from the original weights of a neural network by applying:

$$w_b = \begin{cases} +1 \text{ with probability } p = \sigma(w) \\ -1 \text{ with probability } 1 - p \end{cases}$$

with σ being defined as:

$$\sigma(x) = clip(\frac{x+1}{2}, 0, 1)$$

and clip being the clipping function (1)

4.2 Ternary-weight networks

Ternary-weight Networks (TWN)[3] are based on neural networks using BC, and share its goal of decreasing computational cost and storage requirements of running the model. Similar to Networks using BC this is also achieved by limiting the space of available values that are used as weights in the neural network.

TWN's try to reduce the significant precision loss that occurs in neural networks using BC by also including 0 as an available value for weights. The

main idea here is that this enables Networks to be(have) more similar to conventional neural networks while retaining a similar factor of reduction in computational costs and storage consumption as BC based Networks. TWN's inherit the capability of being computed on easier to scale and cheaper hardware from BC based Networks - since the extra zero states only eliminates some of the values that are added up in a node. And therefore does not significantly affect the computational costs of running the neural network compared to BC.

Since the main goal of TWN's is to more accurately depict the original weights off a DNN then BC, Weights should be chosen by quantising as optimal as possible. Therefore the Ternary Function used to quantize the weights is more complicated then the one used by BC. The paper on TWN's explains how they "solved" this optimisation Problem by modelling it in way that makes it possible to transform to the Problem of optimising a single threshold value which is then used to calculate the ternary weights by applying the following formula:

$$w_t = \begin{cases} +1 \text{ if } w > \Delta \\ -1 \text{ if } w < -\Delta \\ 0 \quad \text{else} \end{cases}$$

Where w is the original weight, w_t the ternary weight and Δ is the threshold value. The threshold value is the solution to the optimisation problem:

$$\Delta = \arg\min_{\Delta>0} \frac{1}{I_\Delta}(\sum_{i \in I_\Delta} |w_i|)^2 \tag{2}$$

Where $I_\Delta := \{i||w_i| > \Delta, w_i \in W\}$ and W is the ordered set of all original weights. The paper mentions that even though there is no straightforward solution to this problem, under the assumption that weights are normal or uniformly distributed the following is a very good approximation [3]:

$$\Delta \approx \frac{0.75}{|W|} * \sum_{w \in \mathrm{W}} |w| \tag{3}$$

4.3 Neural Networks with fixed point weights

Another approach to reduce the resource requirements of neural networks is quantizing their weights by storing them as fixed point numbers with limited precision. This works because "high-precision computation in the context of [machine] Learning is rather unnecessary"[8] since neural networks possess a certain "algorithm-level noise-tolerance"[8]. So we can save a lot of resources when not performing computations at an unnecessary high floating point precision over traditional non-quantized neural networks.

Because using fixed point numbers limits the precision over floating point numbers we need to round to perform the conversion. [8] results show that

choosing the right rounding function has a large influence on how accurate and similar to the non-quantized neural network the quantized neural network performs. They experimentally determined that stochastic rounding significantly outperforms the naive approach of rounding to the nearest number, unless a large amount of fractional bits is used for the fixed point representation. This means if we want to achieve a significant reduction in computational and storage requirements we want to use stochastic rounding. Which results in the following quantisation scheme:

Given a chosen fixed point representation $\langle I, F \rangle$ with I being the number of integer bits and F being the number of fractional bits they define:

$$Round(x, \langle I, F \rangle) = \begin{cases} \lfloor x \rfloor & \text{with probability } 1 - \frac{x - \lfloor x \rfloor}{\epsilon} \\ \lfloor x \rfloor + \epsilon & \text{with probability } \frac{x - \lfloor x \rfloor}{\epsilon} \end{cases}$$

where $\epsilon = 2^{-F}$ is the smallest difference between two values in the chosen fixed point representation.

They then use *Round* for the quantisation:

$$w_{\langle I,F \rangle} = Round(clip(w, 2^{I-1} - 2^{-F}, -2^{I-1}), \langle I, F \rangle)$$

Where $w_{\langle I,F \rangle}$ is the quantized weight, w the original weight and *clip* the clipping function (1).

5 Comparison with other techniques

When trying to compare the different quantisation techniques we need to compare their effectiveness both in reducing computation costs and memory requirements as well as maintaining an accuracy as close as possible to the original non-quantized neural network. For this we first look at the results of prior research ([7],[3],[8]) and then compare it to the results that were found in [1] Results with wide reduced precision Networks. Luckily for us [10] already compares these 3:

Table 2 from [10]: "A qualitative comparison of binary, ternary, integer/fixed point, and mixed precision quantisation schemes"

Quantisation Scheme	**Accuracy Loss**	**Advantages**	**Disadvantages**
Binary	High	**Low cost.** All arithmetic done via binary operations. 32× size compression rate.	**High accuracy loss.** Binary networks often incur around 10% accuracy reductions
Ternary	Low - Moderate	**High compression rate.** Multiplications done via binary operations or capped at two multiplications per activation if using asymmetric scaling factors. 16× compression rate.	**Floating point arithmetic.** For negligible accuracy loss, ternary networks use asymmetric floating point scaling factors, so they need to perform two floating point multiplications per activation
Integer/Fixed Point	Low	**Integer arithmetic.** All arithmetic done via integer arithmetic, which is much cheaper than floating point arithmetic.	**Uniform precision.** For minimal accuracy loss, the networks are limited to the bit width of most sensitive layer, which is often 8 bits, so the compression rates are at most 4×.
Mixed Precision	Low	**Custom precision.** Quantisation scheme for each layer or even row of weights is tailored to their precision sensitivity, reaping the benefits of binary, ternary, and integer quantisation.	**Large search space.** The search space for which quantisation scheme to use for each layer or weight row is exponential in the number of layers or weight rows, respectively

[1] finds that WRPN with double the amount of filter maps based on AlexNet and the ILSVRC-12 dataset "with 4-bits weights and 2-bits activations [exhibit] accuracy at-par with full-precision networks. Operating with 4-bits weights and [more then] 4-bits [for] activations surpasses the baseline accuracy"[1]. The paper also finds that "with 4b operands for weights and activations [...] reduced-precision AlexNet is just 49% of the total compute cost of the full-precision baseline"[1].

[1] also includes results of applying WRPN techniques to ResNet. They found that when doubling the amount of filter maps 4bit activations and 2 bit weights were sufficient to be on-par with the baseline non-quantized network, while using more than 2 bit weights outperforms the baseline network and using 2 bit weights and activations only caused a 0.2% loss in accuracy compared to baseline.

This means that when using enough filter layers the accuracy loss of WRPN is small to negligible. So in terms of accuracy loss WRPN seem superior to techniques that apply quantisation just to the weights of a neural network. Also WRPN are computationally cheaper like traditional quantisation techniques when compared to conventional neural networks. However the paper also finds that this reduction in computational cost is only possible when balancing the precision reduction of weights and activations with the increase in filter maps. Failing to do so properly actually results in an increased computational cost.

6 Conclusion

As we have seen WRPN can be an effective solution to the main issue of quantisation, the Tradeoff between accuracy and meaningful reduction in resource requirements. Common previous techniques either achieved a high resource reduction or a minimal loss in accuracy - not both. This advantage of WRPN however comes at the disadvantage that WRPN now have another trade off instead which needs to be optimised for each dataset: The optimal precision reductions (bit-lengths of the weights and activations) and scaling factor of filter maps. If not optimised properly in the worst case the performance of the WRPN can be even worse then the non-quantized network while having higher resource requirements as well. This means WRPN require a lot of experimental data on a specific data set to be applied to it effectively.

Nevertheless in our opinion the fact that these additional experiments can be seen as part of the "training" or the quantisation process means that WRPN can be a great solution to reducing resource requirements of running a convolutional neural network after training.

References

1. Asit Mishra, Eriko Nurvitadhi, J.J.C.D.M.: Wrpn: Wide reduced-precision networks (Sep 2017), https://arxiv.org/abs/1709.01134v1
2. Damelin, S.B., Miller, W.: The Mathematics of Signal Processing. Cambridge University Press (2011)
3. Fengfu Li, Bin Liu, X.W.B.Z.J.Y.: Ternary weight networks (Nov 2022), https://arxiv.org/abs/1605.04711
4. Hardesty, L.: Explained: Neural networks (2017), https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (Dec 2015), https://arxiv.org/abs/1512.03385
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012), https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
7. Matthieu Courbariaux, Yoshua Bengio, J.P.D.: Binaryconnect: Training deep neural networks with binary weights during propagations (Apr 2016), https://arxiv.org/abs/1511.00363
8. Suyog Gupta, Ankur Agrawal, K.G.P.N.: Deep learning with limited numerical precision (Feb 2015), https://arxiv.org/abs/1502.02551
9. Valueva, M., Nagornov, N., Lyakhov, P., Valuev, G., Chervyakov, N.: Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Mathematics and Computers in Simulation **177**, 232–243 (2020). https://doi.org/https://doi.org/10.1016/j.matcom.2020.04.031, https://www.sciencedirect.com/science/article/pii/S0378475420301580
10. Weng, O.: Neural network quantization for efficient inference: A survey (Jan 2023), https://arxiv.org/abs/2112.06126
11. Yoshua Bengio, Nicholas Léonard, A.C.: Estimating or propagating gradients through stochastic neurons for conditional computation (Aug 2013), https://arxiv.org/abs/1308.3432

Quantization of Neural Networks: An Exploration of Techniques and Trade-offs

Tjaard Pfitzner

Karlsruher Institut für Technologie, 76131 Karlsruhe, Germany
uzgwm@student.kit.edu

Abstract. With the recent advances in artificial intelligence, neural networks are becoming increasingly important. However, their widespread deployment on resource-constrained devices such as mobile phones and embedded systems is limited by their high memory and computational requirements. Neural network quantization allows for reducing the size and computational complexity of neural networks at the cost of precision and performance. This paper provides an exploration of the different techniques used for neural network quantization, with an emphasis on Quantization-Aware Training, a method that incorporates quantized parameters during the training process, where the network learns to accommodate the reduced precision and maintain performance. We will then look at Post-Training Quantization, which quantizes the network after the training process, therefore offering a fast and resource-saving method with the trade-off of lower accuracy.

Keywords: neural network, quantization-aware training

1 Introduction

Neural networks have been shown to be highly effective in various tasks, ranging from natural language processing [10] to computer vision [13]. This effectiveness, however, comes with a trade-off in complexity. According to [17], as neural networks increase in complexity, caused by the number of parameters, new challenges arise when it comes to deploying them on resource-constrained devices, such as mobile phones and embedded systems. The high memory and computational requirements for large neural networks limit their usability in domains, where efficiency and low power consumption are important.

A promising solution to address these challenges lies in neural network quantization. Part of neural network quantization is reducing the precision of weights and activations in a network from floating point values (typically 32-bit) to lower bit-width representation, such as 8-bit integer values. By the quantization of the network's parameters, a significant reduction in memory usage and computational complexity can be achieved [17].

Despite its benefits, quantization has a major downside. According to [20] and [14], it introduces a trade-off between complexity and accuracy. This is no surprise because the parameters are mapped to a lower bit-width representation during quantization and therefore have a lower precision, leading to a loss in model accuracy. This adaptation necessitates developing and using algorithms and techniques that can reduce the negative effects of quantization while making sure that the model retains the speed and size enhancements suggested by the overarching quantization concept [9].

In this paper, we will shine light on some of the techniques used to mitigate the negative effects of quantization. We will mainly emphasize the so-called Quantization-Aware Training, which involves training neural networks with an awareness of the quantization process, allowing the models to adapt and optimize their performance under reduced precision representations. In Section 2, we will give a brief overview of any prerequisite knowledge, while emphasizing neural networks. Section 3 will provide inside on the process of quantizing and dequantizing values to be used in the techniques discussed in Section 4, where we take a look at Quantization-Aware Training and compare it to another method used for neural network quantization called Post-Training Quantization. Finally, we will provide a general summary and conclusion.

2 Prerequisite Knowledge

In this section, we will 1) give an overview of neurons, 2) see how we can connect multiple neurons to form large neural networks and 3) explore how we use the back-propagation algorithm to train these neural networks.

2.1 Artificial Neurons

An artificial neuron is the smallest unit of a neural network [18]. Artificial neurons are modelled in a way that reassembles biological neurons. Multiple inputs get weighted and summed up in a linear combination. This combination is then passed through an activation function to calculate the final output. Figure 1 shows a graphical representation of a single neuron, indexed by j. Each input value x_i gets multiplied by its respective weight w_{ij}. All products are then summed up using the transfer function. Usually, we add a bias value b to the sum. The output of this linear combination is called the net input net_j. The net input gets passed through an activation function φ and if the value surpasses a certain threshold θ_j, then the neuron fires and we get an output value o_j. [23] gives a mathematical definition of the net input via:

$$net_j = \sum_{i=1}^{n} x_i \cdot w_{ij} + b.$$

The output of the neuron is defined as:

$$o_j = \varphi(net_j).$$

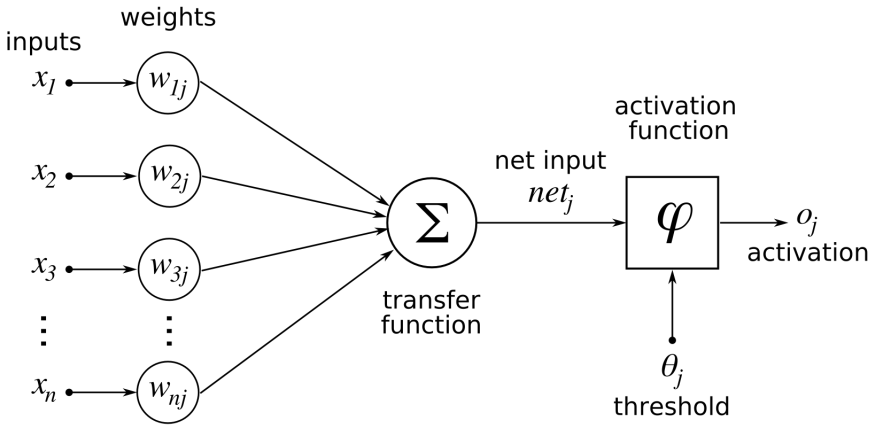


Fig. 1. *Graphical representation of one artificial neuron. Each input $x_1, ..., x_n$ is multiplied with its corresponding weight $w_{1j}, ..., w_{nj}$. The sum over all products is then passed through an activation function φ to calculate the final output o_j. Adapted from [6].*

There are multiple possible activation functions. Common choices are the binary step function, the sigmoid function or the Rectified Linear Unit (ReLU) function. Further information about activation functions can be found in [19].

2.2 Artificial Neural Networks

We can combine multiple artificial neurons to create a neural network [8]. A prevalent and easy neural network approach is the Multi-Layer Perceptron (MLP). In an MLP, neurons are grouped into multiple layers. The input layer, possibly multiple hidden layers, and one output layer. Each neuron in one layer is connected to every neuron in the following layer, therefore the output of a neuron acts as the input to every neuron of the consecutive layer. That way, the MLP forms a directed acyclic graph (see Figure 2) and falls into the category of feed-forward neural networks. With an MLP, and neural networks in general, it is possible to model and learn any linear or nonlinear function, therefore making it attractive for many application domains [1].

2.3 Training a Neural Network with Back-propagation

Neural Networks have the ability to learn through training [8]. A popular learning method is Supervised Learning (SL). In SL, many training data samples, consisting of an input and an expected output, are used to train a neural network. To solve SL tasks, we can use a method called backpropagation [21]. Backpropagation consists of multiple phases: During the so-called forward

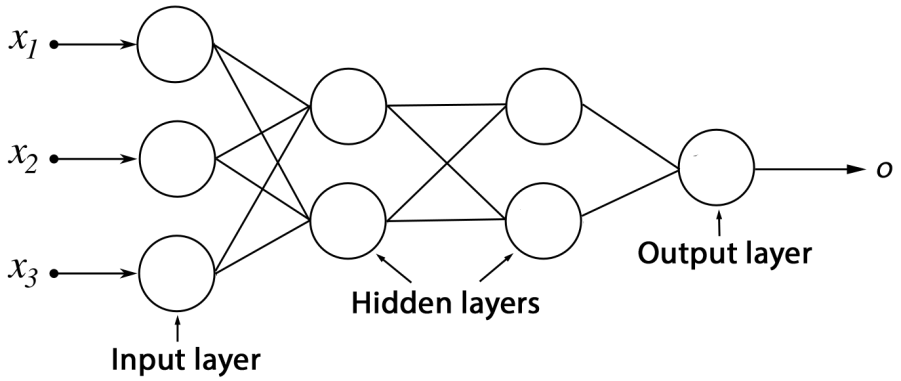


Fig. 2. *Graphical representation of a Multi-Layer Perceptron with the input x_1, x_2, x_3 and output o. The network has one input layer, two hidden layers and one output layer.*

pass, the inputs x_i from these data samples are passed through the network and the network's actual outputs o_j are calculated. Each calculated output is then compared to the expected output y_j of its data sample. A loss is computed through the difference between the desired and actual output. We most commonly use the mean squared error function to calculate the loss, which is defined as:

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_j - o_j)^2.$$

Training a neural net involves minimizing the loss function by adjusting the network's parameters. This is commonly done in the so-called backward pass, using a procedure known as gradient descent. In every iteration, the algorithm calculates the gradient of the loss function L with respect to each parameter. Each parameter is then iteratively updated in the scaled negative gradient direction. The scaling factor μ determines the step size by which a parameter is updated. μ is commonly called the learning rate. Choosing the right learning rate is important when training a neural network. We will not go into any more detail about how to choose μ, however, [12] provides further information. An exemplary iteration of the gradient descent algorithm for each weight is given as:

$$w_{ij,t+1} = w_{ij,t} - \mu\frac{\delta L}{\delta w_{ij,t}},$$

where $w_{ij,t+1}$ depicts the new weight at iteration $t+1$ and $w_{ij,t}$ is the old weight at iteration t. The parameters will be updated until the maximum amount of iterations is reached, or the loss function converges to a minimum.

2.4 Summary (Prerequisite Knowledge)

Neural networks have a lot of potential in many application domains. They consist of neurons, that are combined in multiple layers. A network has an input and an output. They can learn by minimizing the difference between the expected output and calculated output through the adjustment of their parameters.

3 Introduction to Quantization

In this section, we will explore basic concepts of quantization. We will 1) take a look at how to quantize and dequantize values, 2) discuss the difference between uniform and non-uniform quantization and 3) distinguish between symmetric and asymmetric quantization.

3.1 Quantizing Values

To quantize values such as weights and activations used in neural networks, we need to define a function that maps a floating point value $x \in (\alpha, \beta)$ to its lower precision integer representation. A popular quantization function, described in [20] is the following:

$$Q(x) = Int(\frac{x}{S}) - Z, \tag{1}$$

where Q is the quantization operator, S is a floating point scaling factor and Z is the integer value that represents 0 in the quantization scheme. The $Int(\cdot)$ function maps the floating point value x to some integer through rounding. An example of the rounding function is the round-to-nearest, but other functions are also applicable, which can be seen in [16]. The scaling factor S is the same for all real values x, therefore Equation 1 is an example of uniform quantization. This means the distance between quantized values is equally spaced and thereby uniform (see Figure 3, left). It is also possible to vary the distance between each quantized value (see Figure 3, right). This is called non-uniform quantization, however, in this paper, we will only explore uniform quantization.

3.2 Symmetric and Asymmetric Quantization

The scaling factor S has been introduced in Equation 1 and is an important parameter when choosing the right quantization scheme. In [20], the scaling factor is defined as:

$$S = \frac{\beta - \alpha}{2^b - 1},$$

where b is the quantization bit width, α and β define the so-called clipping range of the real values, with α being the lower bound and β being the upper bound of the clipping range. A term used for determining the clipping range in [20] is calibration. [22] states that in practice, values x can fall outside the clipping range (α, β). In this case, the resulting quantized value would fall outside

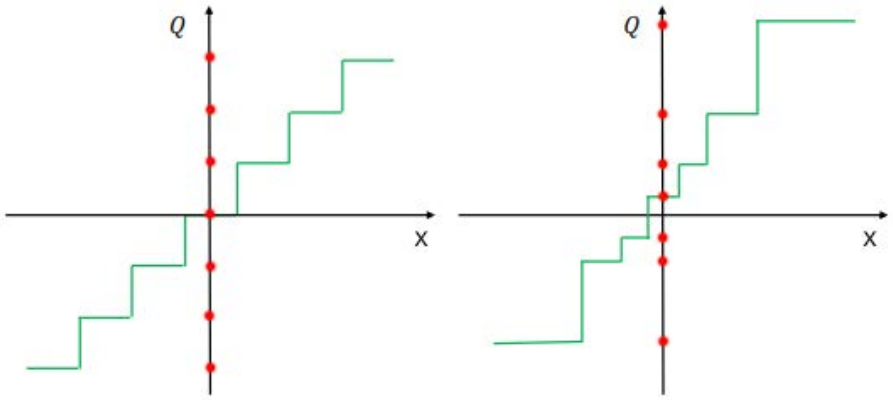


Fig. 3. *Comparison between uniform quantization (left) and non-uniform quantization (right). Real values in the continuous domain x are mapped into discrete, lower precision values in the quantized domain Q, which are marked with red bullets. Note that the distances between the quantized values are the same in uniform quantization, whereas they can vary in non-uniform quantization. Adapted from [9].*

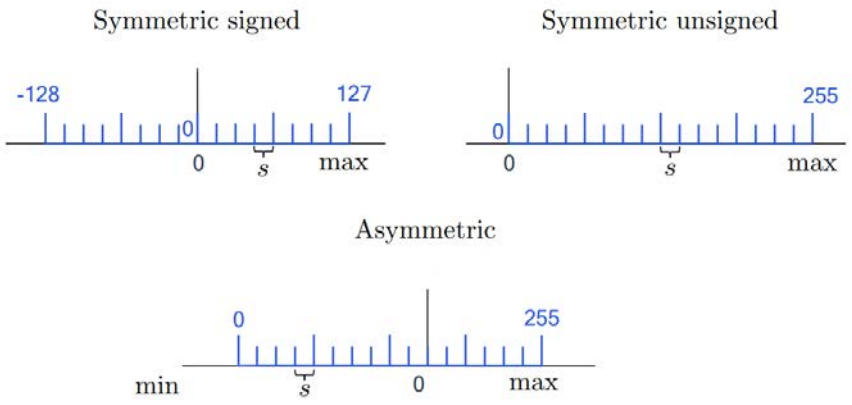


Fig. 4. *Illustration of two symmetric quantization grids (signed/unsigned) and an asymmetric quantization grid. s is the scaling factor. The floating point grid is depicted in black and the integer quantized grid is in blue. Adapted from [17].*

the integer grid range (α_q, β_q). Therefore, it is necessary to clip the quantized values. [17] proposes that, if we use signed integers, the range is defined as $(\alpha_q, \beta_q) = (-2^{b-1}, 2^{b-1} - 1)$. With unsigned integers, the range is defined as $(\alpha_q, \beta_q) = (0, 2^b - 1)$. To include the clipping function in our quantization scheme, we modify Equation 1 into:

$$Q(x) = clip\left(Int(\frac{x}{S}) - Z; \alpha_q, \beta_q\right), \tag{2}$$

where $clip(\cdot)$ is defined as:

$$clip(x; a, c) = \begin{cases} a, & x < a, \\ x, & a \leq x \leq c, \\ c, & x > c. \end{cases}$$

According to [9], symmetric quantization implies $\alpha = -\beta$, which can be achieved by choosing α and β with respect to the minimum and maximum values of the input: $-\alpha = \beta = \max(|x_{max}|, |x_{min}|)$ (see Figure 4, top). By using symmetric quantization, the zero-point Z is mapped to 0, which is computationally less expensive at inference time, according to [20]. Using the mapping $Z = 0$ simplifies the quantization function in Equation 2:

$$Q(x) = clip\left(Int(\frac{x}{S}); \alpha_q, \beta_q\right).$$

In neural networks, where the target weights or activations are imbalanced, e.g., the activation after ReLU that always has non-negative values, asymmetric quantization provides a more accurate approach. Using asymmetric quantization, the clipping range is not symmetric with respect to the origin (see Figure 4, bottom). By choosing the minimum and maximum of the input values as the clipping range, i.e., $\alpha = x_{min}$, and $\beta = x_{max}$, this asymmetric quantization can be achieved, given $-\alpha \neq \beta$ [9].

3.3 Dequantizing Values

By defining a quantization function, it is also necessary to define its inverse function, which calculates the original real value from the quantized value. [20] defines this dequantization function as:

$$\hat{x} = S \cdot (Q(x) + Z). \tag{3}$$

Using symmetric quantization, Equation 1 can again be simplified to:

$$\hat{x} = S \cdot Q(x).$$

As said in [17], it is essential to note that the dequantized value $\hat{x}$ might not equal the original real value x. This is due to the rounding function $Int(\cdot)$

introducing a bias and thereby some error that cannot be recovered by the dequantization function.

We can combine Equations 2 and 3 to get a general definition for the quantization function as:

$$\hat{x} = S \cdot \Big[clip\Big(Int(\frac{x}{S}) - Z; \alpha_q, \beta_q \Big) + Z \Big]. \tag{4}$$

3.4 Summary (Introduction to Quantization)

We introduced a quantization scheme, that can quantize real values into a smaller bit representation. We can quantize values uniformly or non-uniformly. This refers to the distance between the quantized values. We can also choose between symmetric or asymmetric quantization. This sets the clipping range of the quantized values, with symmetric quantization mapping the zero-point of the real values to 0. We can dequantize values to go from a low bit width representation to the original value, however, this introduces some bias, due to a rounding operation in the quantization process.

4 Quantization Methods

In this section, we will explore two main methods used for neural network quantization. We will 1) give an overview of Quantization-Aware Training (QAT) and 2) discuss another approach called Post-Training Quantization (PTQ). For both methods, [20] suggests using a pre-trained model, to be more effective at minimizing accuracy loss when quantizing a neural network. The key difference between QAT and PTQ is, that QAT requires retraining the network, to learn to minimize the quantization error introduced during the quantization process. PTQ can be used without retraining the model, which is beneficial when there is not enough training data available.

4.1 Quantization-Aware Training

QAT is a method for neural network quantization that involves retraining a pre-trained neural network with respect to quantized parameters such as weights and activations. QAT simulates the quantized parameters during the forward and backward pass, by injecting quantization nodes into the computation graph, to introduce the quantization error and learn to minimize it. Figure 5(b) shows the computation graph with quantization nodes. These nodes are removed during inference (see Figure 5(a)). During the backward pass, we use floating point values to calculate the gradients but run into a problem due to the quantization function, defined in Equation 4, being non-differentiable. The quantization function uses a rounding function $Int(\cdot)$ which has a gradient that is zero almost everywhere. This necessitates an approximation of the gradient. [4] proposes an

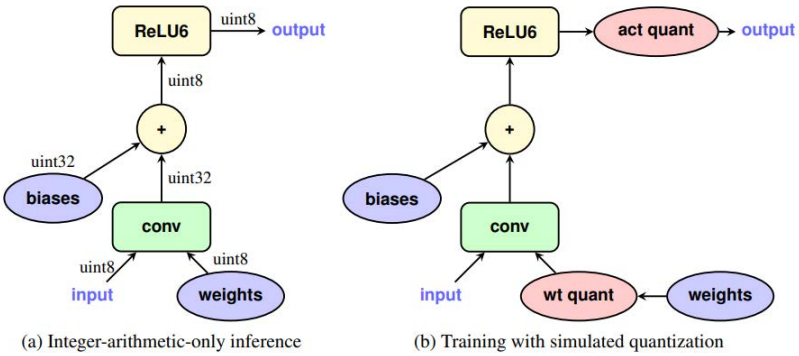


Fig. 5. *(a) shows a computation graph for integer-arithmetic-only inference of a quantized neural network. The network has been trained using simulated quantization, shown in (b). During training, quantization nodes are injected into the computation graph following the weight and activation nodes. These quantization nodes get removed during inference. Adapted from [11].*

estimator called the Straight-Through-Estimator (STE), which approximates the gradient of the rounding function as:

$$\frac{\partial}{\partial y} Int(y) = 1, \ \forall y \in \mathbb{R}. \tag{5}$$

According to [17], Equation 5 enables calculating the gradient of the quantization function 4. Note that we are using symmetric quantization, so $Z = 0$, x is the input and (α, β), (α_q, β_q) determine the clipping ranges for the float and integer values. We can now calculate the gradient of the quantization function 4 with respect to the input x as:

$$\begin{aligned}
\frac{\partial \hat{x}}{\partial x} &= \frac{\partial}{\partial x}\Big(S \cdot clip\big(Int(\frac{x}{S}); \alpha_q, \beta_q\big)\Big) \\
&= S \cdot \frac{\partial}{\partial x}\Big(clip\big(Int(\frac{x}{S}); \alpha_q, \beta_q\big)\Big) \\
&= \begin{cases} S \cdot \frac{\partial Int(x/S)}{\partial (x/S)} \cdot \frac{\partial (x/S)}{\partial x} & if \ \alpha \leq x \leq \beta, \\ S \cdot \frac{\partial}{\partial x} Int(\alpha_q / S) & if \ x < \alpha, \\ S \cdot \frac{\partial}{\partial x} Int(\beta_q / S) & if \ x > \beta, \end{cases} \\
&= \begin{cases} 1 & if \ \alpha \leq x \leq \beta, \\ 0 & otherwise. \end{cases}
\end{aligned}$$

The STE enables us to use the quantization function during the forward pass and then, as the name suggests, we can estimate straight through the quantization function and are therefore able to calculate the gradient during the backward pass (see Figure 6). [17] proposes, that we can use the STE to also calculate the gradient with respect to the other quantization parameters, Z and S. By making these quantization parameters learnable, we can achieve better accuracy [5], [7]. We calculate the gradient with respect to the scaling factor S as:

$$\begin{aligned}\frac{\partial \hat{x}}{\partial S} &= \frac{\partial}{\partial S}\Big(S \cdot clip\big(Int(\frac{x}{S}); \alpha_q, \beta_q\big)\Big) \\ &= \begin{cases} -\frac{x}{S} + Int(\frac{x}{S}) & if\ \alpha \leq x \leq \beta, \\ \alpha_q & if\ x < \alpha, \\ \beta_q & if\ x > \beta. \end{cases}\end{aligned}$$

In Section 3.1, we introduced the zero-point value Z as an integer. However, to make this parameter learnable, we first have to convert it into a real number and then apply the integer rounding operator. That way, we again get a modified quantization function as:

$$\hat{x} = S \cdot \Big[clip\Big(Int(\frac{x}{S}) - Int(Z); \alpha_q, \beta_q\Big) + Int(Z)\Big].$$

The gradient with respect to Z can now be calculated using the STE:

$$\begin{aligned}\frac{\partial \hat{x}}{\partial Z} &= \frac{\partial}{\partial Z}\Big(S \cdot \Big[clip\Big(Int(\frac{x}{S}) - Int(Z); \alpha_q, \beta_q\Big) + Int(Z)\Big]\Big) \\ &= \begin{cases} S \cdot \Big[\frac{\partial Int(x/S)}{\partial Z} - \frac{\partial Z}{\partial Z} + \frac{\partial Z}{\partial Z}\Big] & if\ \alpha \leq x \leq \beta, \\ S \cdot \Big[\frac{\partial \alpha_q}{\partial Z} + \frac{\partial Z}{\partial Z}\Big] & if\ x < \alpha, \\ S \cdot \Big[\frac{\partial \beta_q}{\partial Z} + \frac{\partial Z}{\partial Z}\Big] & if\ x > \beta, \end{cases} \\ &= \begin{cases} 0 & if\ \alpha \leq x \leq \beta, \\ S & otherwise. \end{cases}\end{aligned}$$

The above equations all use the STE, it is however possible to use different estimators. [15] proposes an alternative optimization technique, termed alpha-blending, which quantizes neural networks to low precision using

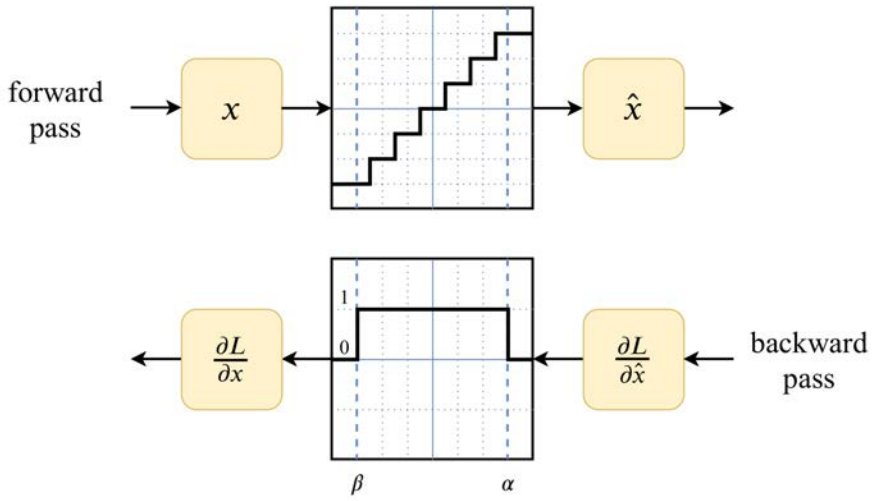


Fig. 6. *Forward and backward pass using the Straight-Through-Estimator. During the forward pass, the input x is quantized, which involves applying the non-differentiable quantization function. During the backward pass, the Straight-Through-Estimator is used to approximate the gradient of the quantization function as the identity function in the clipping range (α, β) and 0 everywhere else. Adapted from [22].*

stochastic gradient descent. Alpha-blending avoids STE approximation by replacing the quantized weight w_q in the loss function with the affine combination $(1 - \alpha)w + \alpha w_q$ of the quantized weight and the corresponding full-precision weight w with non-trainable scalar coefficient α and $(1 - \alpha)$. During training, α is gradually increased from 0 to 1. Weights are updated through the affine combination's full precision term, $(1 - \alpha)w$. That way, the model is converted from full precision to low precision progressively.

QAT offers high accuracy, however, according to [20] it comes with a trade-off. Due to the necessity to retrain the model throughout many epochs, we get high training times with correspondingly high computational retraining costs. A sufficient amount of training data is also essential to prevent the net from overfitting. In environments, where such a quantized network is deployed for long periods of time, it has been proven, that the hardware and energy efficiency gains make up for the trade-offs, that come with using QAT. [24] conducted a case study to examine resource cost and processing speed when using quantized parameters (INT8) instead of parameters in the FP32 format during training (see Table 1). Results reveal that using quantized parameters can effectively alleviate system overhead while not downgrading the model quality.

Table 1. *System performance using quantized INT8 parameters and FP32 parameters. Adapted from [24].*

	Forward Pass (ms)	Backward Pass (ms)	Per-iteration Time (ms)	Parameter Memory (MB)	Model Accuracy
FP32	95.85	140.03	240.06	18.51	97.6%
INT8	54.57	67.66	126.41	9.42	95.2%
Comparison	1.86×	2.07×	1.89×	1.96×	−2.39%

4.2 Post-Training Quantization

A different method used for quantizing neural networks is PTQ [20]. PTQ is often used when there is not enough training data available or the training data is unlabeled [9]. Unlike QAT, PTQ quantizes the network without the need to retrain it. This makes PTQ a swift method without any significant overhead. According to [20], PTQ requires to first calibrate the quantization parameters based on any available calibration data and then to quantize the model using the quantization scheme explained in Section 3. However, networks quantized using PTQ have a lower accuracy and precision than networks quantized with QAT (see Table 2). To mitigate this accuracy loss, many approaches have been explored. For example, [2] proposes a bias correction, due to an inherent bias in the mean and variance of the weight values following their quantization. Analytical Clipping for Integer Quantization is a method that analytically approximates the optimal clipping range [3]. More approaches can be found in [9].

Table 2. *A qualitative comparison of the two main quantization procedures. Adapted from [20].*

Quantization Procedure	Accuracy Loss	Quantization Time	Minimum Achievable Precision
Quantization-Aware Training	Negligible	High	≥ 1 bit [32]
Post-Training Quantization	Moderate	Low	≥ 4 bits [30]

4.3 Summary (Quantization Methods)

We explored two methods of quantization: QAT starts with a pre-trained model, which then gets retrained by injecting quantization nodes into the training process. This simulates quantized parameters, thereby enabling the network to learn the quantization error and minimize it. QAT commonly uses the Straight-Through Estimator, which is needed to perform back-propagation on

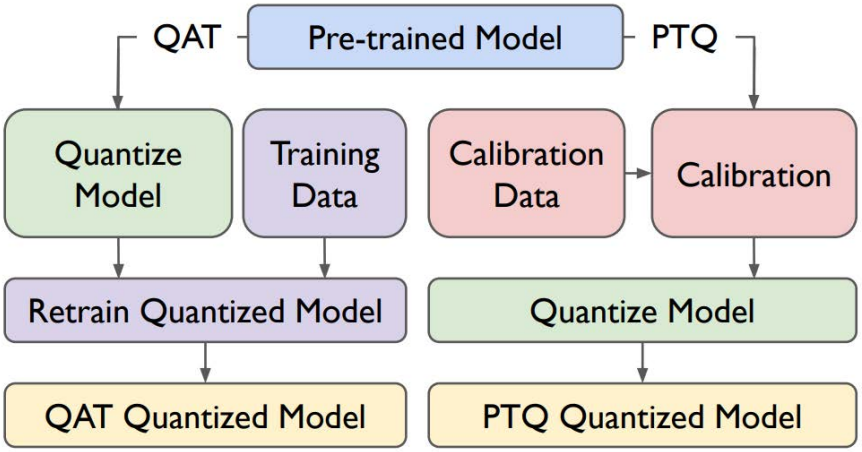


Fig. 7. *Different pipelines using QAT or PTQ for neural network quantization. The calibration data can be either a subset of the training data or a small set of unlabeled input data. Adapted from [9].*

the quantization scheme by estimating the gradient of the non-differentiable function. PTQ involves calibrating and quantizing a network without retraining it. That way, we save a lot of time, however, with a trade-off of lower precision due to the quantization error.

5 General Summary and Conclusion

In conclusion, neural network quantization, along with techniques such as QAT and PTQ, has shown to be promising for optimizing and deploying deep learning models in resource-constrained environments. Through the process of quantization, which involves reducing the precision of weights and activations, neural networks can achieve significant reductions in memory, computational requirements, and energy consumption, while maintaining acceptable levels of accuracy.

QAT has proven to be an effective approach for training neural networks to mitigate the loss of precision during quantization. By simulating the quantization process during training, the network learns to adapt to lower precision, ensuring that the quantization error is minimized. That way, we can achieve a balance between accuracy and resource efficiency.

PTQ, on the other hand, involves quantizing a pre-trained neural network without retraining. This approach provides a convenient way to apply quantization to existing models, avoiding the need for extensive retraining.

As research and development in this field continues, we can expect quantization

to play a significant role in bringing the possibilities of neural networks to devices with limited resources.

References

1. Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A., Arshad, H.: State-of-the-art in artificial neural network applications: A survey. Heliyon **4**(11), e00938 (2018). https://doi.org/https://doi.org/10.1016/j.heliyon.2018.e00938, https://www.sciencedirect.com/science/article/pii/S2405844018332067
2. Banner, R., Nahshan, Y., Hoffer, E., Soudry, D.: ACIQ: analytical clipping for integer quantization of neural networks. CoRR **abs/1810.05723** (2018), http://arxiv.org/abs/1810.05723
3. Banner, R., Nahshan, Y., Hoffer, E., Soudry, D.: ACIQ: Analytical clipping for integer quantization of neural networks (2019), https://openreview.net/forum?id=B1x33sC9KQ
4. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation (2013)
5. Bhalgat, Y., Lee, J., Nagel, M., Blankevoort, T., Kwak, N.: LSQ+: improving low-bit quantization through learnable offsets and better initialization. CoRR **abs/2004.09576** (2020), https://arxiv.org/abs/2004.09576
6. Camuñas-Mesa, L., Linares-Barranco, B., Serrano-Gotarredona, T.: Neuromorphic spiking neural networks and their memristor-cmos hardware implementations. Materials **12**, 2745 (08 2019). https://doi.org/10.3390/ma12172745
7. Esser, S.K., McKinstry, J.L., Bablani, D., Appuswamy, R., Modha, D.S.: Learned step size quantization. CoRR **abs/1902.08153** (2019), http://arxiv.org/abs/1902.08153
8. Gardner, M., Dorling, S.: Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric Environment **32**(14), 2627–2636 (1998). https://doi.org/https://doi.org/10.1016/S1352-2310(97)00447-0, https://www.sciencedirect.com/science/article/pii/S1352231097004470
9. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K.: A survey of quantization methods for efficient neural network inference (2021)
10. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine **29**(6), 82–97 (2012). https://doi.org/10.1109/MSP.2012.2205597
11. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
12. Jacobs, R.A.: Increased rates of convergence through learning rate adaptation. Neural Networks **1**(4), 295–307 (1988). https://doi.org/https://doi.org/10.1016/0893-6080(88)90003-2, https://www.sciencedirect.com/science/article/pii/0893608088900032
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger,

K. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012), https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
14. Liu, Z., Cheng, K., Huang, D., Xing, E.P., Shen, Z.: Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. CoRR **abs/2111.14826** (2021), https://arxiv.org/abs/2111.14826
15. Liu, Z.G., Mattina, M.: Learning low-precision neural networks without straight-through estimator(ste) (2019)
16. Nagel, M., Amjad, R.A., van Baalen, M., Louizos, C., Blankevoort, T.: Up or down? adaptive rounding for post-training quantization (2020)
17. Nagel, M., Fournarakis, M., Amjad, R.A., Bondarenko, Y., van Baalen, M., Blankevoort, T.: A white paper on neural network quantization (2021)
18. Schmidhuber, J.: Deep learning in neural networks: An overview. CoRR **abs/1404.7828** (2014), http://arxiv.org/abs/1404.7828
19. Siddharth Sharma, Simone Sharma, A.A.: Activation functions in neural networks. International Journal of Engineering Applied Sciences and Technology **4**(12), 310–316 (2020)
20. Weng, O.: Neural network quantization for efficient inference: A survey (2023)
21. Werbos, P.: Backpropagation through time: what it does and how to do it. Proceedings of the IEEE **78**(10), 1550–1560 (1990). https://doi.org/10.1109/5.58337
22. Wu, H., Judd, P., Zhang, X., Isaev, M., Micikevicius, P.: Integer quantization for deep learning inference: Principles and empirical evaluation (2020)
23. Zhang, Z.: Artificial Neural Network, pp. 1–35. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-67340-0$_1$, https://doi.org/10.1007/978-3-319-67340-0_1
24. Zhou, Q., Guo, S., Qu, Z., Guo, J., Xu, Z., Zhang, J., Guo, T., Luo, B., Zhou, J.: Octo: INT8 training with loss-aware compensation and backward quantization for tiny on-device learning. In: 2021 USENIX Annual Technical Conference (USENIX ATC 21). pp. 177–191. USENIX Association (Jul 2021), https://www.usenix.org/conference/atc21/presentation/zhou-qihua

An Introduction: Neural Network Quantization and Parameterized Clipping Activation

Marc Thieme
Supervisor: Haibin Zhao

Karlsruher Institut für Technologie, 76131 Karlsruhe, Germany
usldt@student.kit.edu

Abstract. Quantization of neural networks is currently one of the most promising and well researched approaches to neural network compression and acceleration we have. It is a diverse research field tending in the direction of customizing and adapting ever more and creative parameters. The Parameterized Clipping Activation (PACT) quantization scheme was a contribution to this trend in that it opened up the clipping and quantized range of activations to be adapted during training.
In this paper, we give an overview over the quantization research field and detail the PACT algorithm as an example. The goal is to present materials in an approachable way and act as an introduction to the field with an in-depth example constituted by the PACT algorithm.

1 Introduction

Deep Neural Networks have revolutionized the fields of image, audio and video processing [14]. As phrased in [14], "Deep learning [has been] making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years". However, the size of deep neural networks has increased to the point where they can be not only hard to train, but also hard to deploy, specifically concerning their memory footprint, execution speed and training requirements. [28, 13] This hinders their use in resource-constrained environments, such as "electronic devices and services, from smartphones, smart glasses and home appliances, to drones, robots and self-driving cars" [19]. Extensive research is directed towards studying several approaches to mitigate this problem [6]. Examples consist of "channel pruning, [which] directly reduces feature map width [...] [and] shrinks a network into a thinner one" [12]. In Channel Pruning, redundant channels are eliminated by using sparsity-constraints, selecting channels to remove in a way which minimizes the error [33]. Similarily, network sparsification "attempts to reduce the number of operands [neurons and weights]", e.g. making a "heuristic search of unimportant elements" in the network [7]. Other approaches for model compression are "Knowledge distillation [, which] is a popular technique for training a small student network to emulate a larger teacher model" [25], neural architecture search as "the process of automating architecture engineering" [9], or lastoly dedicated architectures such as SqueezeNet [13].

1.1 Introduction to Neural Network Quantization

One promising and well-researched approach to network compression is quantization of neural networks [28]. Here, the overarching idea is to replace computations using 32-bit floating point numbers with computations using numbers of smaller bit-width, usually in a different representation. This reduces the memory footprint and the computational cost of the network [28]. Conceptionally, the "Quantization of weights is equivalent to discretizing the hypothesis space [the mappings from the input space to the output space [2]] of the loss function with respect to the weight variables" [6]. Therefore, the approach trades "accuracy for lower precision to achieve smaller models" [28], i.e. it trades computational efficiency for the effectiveness (and applicability) of the gradient descent during backpropagation. There exist numerous approaches to quantization largely differing in bit-widths they require and number representations they use. Inherently, they exhibit different capabilities in compression and inference speed increases on one hand and inference accuracy on the other hand. Therefore, the goal for research in the field is generally minimizing the accuracy loss that comes from quantizing and pushing the field to make those approaches enabling the largest compression potential and computational efficiency viable [28].

1.2 Overview of the Pact Algorithm

In works previous to [6], it was difficult incorporating the Linear Rectified Unit (ReLU) activation function (see section 2.2) effectively in quantization. The problem was that when "the output of the ReLU function is unbounded, the quantization after ReLU requires a high dynamic range" [6] and when the output was clipped to become unbounded, approaches would "not fully utilize the strength of back-propagation to optimally learn the clipping level because all the quantization parameters are determined offline and remain fixed throughout the training process" [6][3]. Other research did not use ReLU for the activation function in favour of leaving the threshold function unspecified but requiring its output range in [0; 1), which enables straightforward quantization [32]. However ReLU has proven very effective in Convolutional Neural Networks (CNN) (see section 2.1) because it leaks the gradient in the positive range through to previous layers during back propagation [21].

In order to preserve the effectiveness of ReLU, the PACT algorithm introduces a similar activation function which clips the output based on a parameter learned during training. As such, it introduces a clipping range dynamically per layer, which is meant to prevail the benefits of ReLU, while enabling effective quantization. More specifically, it identifies the need for varying quantization scales in ReLU due to large variations in the activation amplitude and consequently large variations in the optimal quantization ranges across the network [6].

1.3 Objectives and Audience

This paper gives an overview over quantization ideas and approaches. Afterwards, we first introduce concepts relevant to the PACT algorithm as presented in [6] and subsequently explain the algorithm in a more elaborate and approachable way than the original paper. It is intended for readers with a basic understanding of neural networks but no previous experience with quantization. The paper is structured as follows: Section 2 provides an overview of quantization and branches out to techniques neighbouring those applied in [6]. Section 3 introduces the concepts relevant for understanding and motivates the PACT algorithm. Section 4 provides a detailed explanation of the algorithm. Section 5 discusses extensions and variations of the algorithm published since its release.

2 Background of Neural Network Quantization

The following section constitutes a recollection of basic concepts and terminology. It is intended to be a primer for readers already familiar with the concepts. It is not intended to contain comprehensive explanations of the topics.

2.1 Overview of Convolutional Neural Networks

A neural network is typically organized in layers, with each layer containing a set of nodes that process information from the previous layer. When executed, the first layer (input layer) of the network assumes the values of the input of the network while the values of the last layer (output layer) are interpreted as the output. Neural networks are trained to minimize a loss function with respect to the actual and the desired output of the network. During so-called backpropagation and gradient descent, the gradient of the network's parameters with respect to the loss function is calculated and then used to update the network's parameters in the direction of minimal loss. By repeating this algorithm, ideally, the network finds a minimum of the loss function which corresponds to a satisfying predication performance of the network [23].

Convolutional neural networks (CNNs) are a type of neural network successfully used in image recognition tasks [11]. A CNN contains convolutional layers which convolve a set of learnable kernels across each channel of their input. Ideally after training, each kernel detects specific features such as edges, corners, or textures of the input image. By sequencing multiple convolutional layers, CNNs can learn complex and abstract features [31].

In a fully connected layer, all n nodes of one are connected to all m nodes of the previous layer. The output is calculated according to a weight matrix $W \in \mathbb{R}^{m \times n}$ and a bias vector $b \in \mathbb{R}^n$. Both are parameters of the network. The output is then given by $Wx + b$, where x is the vector of activations of the nodes in the previous layer. The output is then fed through an activation function introducing non-linear properties to the networks transformations, allowing it to learn complex data patterns [24]. For the activation functions, we will only discuss ReLU in this article.

2.2 Loss function and Gradient descent

The loss function in Neural networks is used as a quality measure for the predictions a neural network makes. It takes the networks' and the desired predictions of a network as inputs and calculates a measure for their disparity. By offering a quantifiable measure of how closely the network matches the desired predictions, it presents an objective whose minimization by gradient descent represents aligning the network's transformations to the desired ones [27].

Backpropagation with gradient descent is the process of adjusting the models weights to minimize the loss function. It calculates the gradient of the loss function with respect to each parameter by recursively calculating previous layers' based on the gradients calculated for successive layers. Along the way, it adjusts the parameters to decrease the loss function based on the calculated gradient [24].

ReLU Activation Function ReLU is an activation function which is used in neural networks. It is defined as

$$h(x) = max(0, x) \tag{1}$$

[21] and performs particularly well because it allows the gradient to propagate through layers of the neural network [11]. [6] use a variant of ReLU which is bounded by a parameter α learned during training. As such, their contribution can be boiled down to enabling ReLU activation function to be used effectively in a quantization context.

2.3 Introduction to Quantization Techniques

The representation of numbers within computers provides the basis for understanding quantization. Often, neural networks "use 32-bit floating point data types to represent their parameters as well as all of the computations involved with inference, meaning they require expensive floating point units to run" [28]. When applied to neural networks, quantization encompasses the reduction of these numerical representations' precision, shifting from n-bit precision to smaller m-bit precision. This transition can lead to a notable reduction in memory usage and computational requirements, a critical consideration for implementing neural networks on resource-limited devices [28, 19].

Different numerical representations can be used during quantization:

1. Floating Point Representation: Numbers are represented using scientific notation, e.g. according to IEEE floating point standard using 32 bits. They provide a broad dynamic range and higher precision for values closer to 0. However, multiplication is more complicated and costly than alternatives. [28].

2. Fixed Point Representation: Numbers are represented with a fixed number of digits before and after the decimal point. Multiplying two numbers is efficient because we can first multiply the representations as if they were integers and then shift the bits to align the decimal point [28].
3. Dynamic Fixed Point Representation: "Dynamic fixed point attempts to meet floating point and fixed point in the middle. [...]. Dynamic fixed point employs several scaling factors that are shared among a group of values, and these scaling factors are dynamically updated as the statistics of the group changes" [28].
4. Integer Representation: Only integer values are represented, providing straightforward calculations but naturally only integer precision, which can hamper accuracy [28].
5. Binary and Ternary Representation: Numbers are represented using only two (0 and 1) or three (-1, 0, and 1) values respectively. This yields very good compression ratio and enables hardware optimizations/simplifications [28, 16].

Additionally, the process of quantization can adopt either a uniform or non-uniform approach and can be symmetric or asymmetric. Uniform quantization maintains equal intervals between each quantization level, while non-uniform quantization permits variable intervals [28, 19, 15]. In symmetric quantization, 0 is quantized to zero. For asymmetric, also called affine quantization, this restriction is lifted [19].

[6] use uniform and symmetric quantization. It is prefaced with a clipping function (resembling ReLU) and quantizes from the clipping range $[0, \alpha]$ in turn to the quantized range $[0, \alpha]$ and use fixed-point representation with constant k bits precision.

2.4 Neural Network Quantization: Goals, Benefits and Limitations

When applying quantization to a neural network, the goal is generally to empower running on resource-constrained environments [28]. The problem without network compression often lies in network size, inference latency and energy consumption [28]. Also, quantization can enable hardware optimizations for some specific number representations which would not be possible conventionally [32, 16].

By network size or memory footprint of a network, we refer to the amount of data required to store the model parameters. As models can have millions of parameters filling about 100MB [5] their deployment can be a challenge on devices with restricted memory capabilities, e.g. on the edge. Also, a large memory footprint can me a major hindrance in applications where the model is transmitted across a network, e.g. image recognition models in cars [13]. Quantization addresses this problem by reducing the size of the model parameters thereby decreasing the overall memory footprint of the entire model. [28]

As quantization reduces bit-widths and ideally simplifies the number representation, operating on them naturally becomes cheaper. Also, the quantized representations can enable special optimizations, e.g. for binary quantization, dot product operations can be implemented using only bitshifts [32] or for "ternary weights[,] one can get rid of the multipliers and use adders instead" [16]. For the same reason, inference on the quantized networks save energy [28]. For instance, an adder component on a circuit board for 32-bit floating point numbers consumes 30 times more energy than an 8-bit adder component [5].

2.5 Limitations and Challenges in Conventional Quantization Techniques

Quantization techniques, while beneficial in minimizing the memory footprint and computational costs of neural networks, also present their own set of challenges. The most prominent among these are accuracy loss and a discrete search space which needs to be worked around and generally results in slower converges during training.

At the time of writing this, quantizing a network can still lead to 20% to 30% decrease in accuracy even for the best choices in the quantization scheme [28]. Usually, using fewer bits sacrifices accuracy. This trade-off between model accuracy and computational efficiency is a fundamental challenge in neural network quantization [28].

When quantizing a neural network, the parameters are mapped to a discrete set of values. As a result, the gradient becomes trivial almost everywhere and the traditional gradient descent would not progress. To combat this, alternative techniques to the analytical gradient descent are introduced, often leading to more complex optimization landscapes (e.g. the search space grows exponentially with the number of layers when using mixed precision quantization [8]). This makes the process of finding a good local minimum more challenging and ultimately results in slower convergence and lies at the root of the accuracy degradation [28].

2.6 Overview of existing Neural Network Quantization Techniques

Neural network quantization methodologies primarily categorize into two types: Quantization Aware Training (QAT) and Post-Training Quantization (PTQ). QAT integrates quantization into the training phase, thereby enabling the network to adapt to the quantization errors during training. On the contrary, PTQ applies quantization after the training process has been completed. This method is more effective during training since the search space remains continuous, albeit the sudden introduction of quantization can lead to significant performance loss, as the model has not been equipped to adjust to the quantization errors during the training phase [28].

Both QAT and PTQ employ a range of quantization strategies that largely match the general quantization techniques discussed in section 2.3. Uniform

quantization refers to evenly spaced quantization levels, while non-uniform quantization adjusts the levels based on the distribution of weights, potentially providing higher precision only where it is needed. However, this non-uniformity makes operations more complex as it generally carries along an additional normalization step before numbers can be operated together [15]. Another promising approach is mixed precision quantization [28]. It varies and determines custom bit-widths where needed: With mixed precision quantization, "each layer has a tailored bit width and precision because mixed precision recognizes that some layers may benefit from more bits whereas others can afford to use fewer bits" [28]. [8] demonstrate one example for employing mixed precision quantization effectively.

As the search space becomes discrete when using quantization, the standard gradient descent cannot be employed meaningfully. Two solutions for this problem are introducing a proxy value for the analytical derivation. These are the popular [29] Straight-Through-Estimator (STE) or the backpropagation approach in Monte-Carlo-Quantization (MCQ). The former approximates the discrete quantization function with a similar differentiable function and delegates the gradient of the former to the gradient of the latter [1, 29]. In contrast, Monte Carlo Quantization (MCQ) introduces a stochastic component to the gradient estimation process, approximating the gradient using the Monte Carlo method [17].

3 Overview over the PACT Algorithm

3.1 Summary

The PACT algorithm consists of an activation function $PACT(\cdot)$ based on ReLU and the subsequent quantization operator. This operator is similar to $\mathbf{quantize}_k$ proposed by [32] but scales the quantization range by a parameter α. This parameter distinguishes the approach from other quantization schemes [32, 16] since it allows dynamically adapting the quantization scale to the problem domain and the locale in the neural network [6]. α is a parameter of the search space and is optimized during training.

In this section, we will first introduce prerequisite knowledge which is not original in [6]. Afterwards, we will present the algorithm in detail and discuss nuances and empirical recommendations presented by the authors.

3.2 Technical Concepts used in the Pact Algorithm

Regularization Regularization is a technique used to prevent overfitting [30]. Overfitting, in turn, is a "fundamental issue in supervised machine learning which prevents us from perfectly generalizing the models to well fit observed data on training data [sic], as well as unseen data on testing sets". The problem entails that "the model performs perfectly on [the] training set, while fitting poorly on testing sets" [30]. Generally, this phenomenon is caused either by the

model learning noise in the training data or by the model's parameter set being complex enough to fit the training data so accurately that it does not generalize meaningfully to data not in the training set [30].

Generally, the goal of regularization is to "minimize the weights of the features which have little influence on the final classification" [30]. In practice, this is accomplished by augmenting the loss function by an additional term which penalizes large weights. For L1 and L2 regularization respectively, the regularized loss function $\widetilde{c}(\omega, X, y)$ becomes

$$\textbf{L1:} \qquad \widetilde{c}(\omega, X, y) = c(\omega, X, y) + \lambda \sum_{i=1}^{n} |\omega_i| \tag{2}$$

$$\textbf{L2:} \qquad \widetilde{c}(\omega, X, y) = c(\omega, X, y) + \underbrace{\lambda \sum_{i=1}^{n} \omega_i^2}_{\text{regularization term}} \tag{3}$$

where $\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix}$ are the parameters of the network, X is the training set, y is the labelled data and λ is the regularization coefficient dictating the strength of the regularization effect. Since the parameter updates during backpropagation always follow the gradient of the loss function in the decreasing direction, the regularization term "provides incentive" for the parameters not to grow extremely large. This can prevent features from being overrepresented in the model due to the parameters of the feature contributing excessively to the output of the model [30].

Backpropagation on discrete functions: Straight Through Estimator As mentioned in section 2 the gradient of a discrete function is everywhere either zero or undefined [29, 19]. This is a problem in neural network quantization [1] because the functions used to quantize the weights, activations or gradients is discrete by definition.

However, the gradient is necessary to update the parameters (e.g. the weights) of the neural network during training. In particular, if the incoming gradient for a neuron or a convolution kernel is zero, the parameters would not change no matter the learning rate. Ideally, you would want the parameters to move closer in the direction whose next quantization level decreases the loss function. PACT uses the Straight Through Estimator (STE) to approximate this judgement [6], which, for uniform quantization, generally approximates the derivative of the quantization with the derivative of the identity function [1, 29, 19].

The Quantize Operator [32] present an operator which implements a quantization function for the forward pass and an STE for the backward pass.

This operator is similar to the following operator, which [6] append to their custom activation function. Adhering to their notation, [32] define the $\textbf{quantize}_k$ operator as follows:

$$\textbf{FORWARD:} \qquad x_q = \frac{1}{2^k - 1} round\left(x \cdot \left(2^k - 1\right)\right) \tag{4}$$

$$\textbf{BACKWARD:} \qquad \frac{\partial c}{\partial x} = \frac{\partial c}{\partial x_q} \tag{5}$$

where x is the input to the quantization function, x_q is the quantized output, k is the number of bits used to represent the quantized value and $c(\cdot)$ is the loss function.

We first observe that, during the forward pass, this operator implements a symmetric uniform quantization operation over the clipped and also quantized range $x, x_q \in [0, 1]$. Second, we observe the application of an STE for the backward pass. By rewriting the equation to

$$\frac{\partial c}{\partial x_q}\frac{\partial x_q}{\partial x} = \frac{\partial c}{\partial x} = \frac{\partial c}{\partial x_q} \tag{6}$$

and deriving:

$$\frac{\partial x_q}{\partial x} = 1 \tag{7}$$

we see the resemblance of the STE in its canonical form for uniform quantization [29].

Having defined the behavior of $\textbf{quantize}_k$ for the backward pass, we can use it in training a model. However, the quantization range is fixed to $[0, 1]$ and the quantization step size is fixed to $\frac{1}{2^k-1}$. Consequently, it assumes a bounded activation function $h(\cdot)$ which ensures activations fall in this range [32]. This is a problem because the quantization range and step size are not adapted to the problem domain. For example, if the input to the quantization function is always in the range $[-0.5, 0.5]$, the quantization range $[-1, 1]$ is unnecessarily large. This results in an unnecessary loss of precision. When ReLU is used, the range of activation is $[0, \text{inf}]$ [6]. Attempting to use this as the quantization range for a uniform quantization yields an infinitely large step size, making the operation impossible in practice. This problem will be solved by the PACT activation function, which uses a threshold function akin to ReLU but bounded by a parameter α which is learned during training.

3.3 Principles and Motivation of the Pact Algorithm

Those are the techniques and topics relevant to the PACT algorithm. In the following section, they will be used to construct a quantization scheme which enables the use of an activation function akin to ReLU and enables a model during training to adjust the quantization scale used in quantizing the activations according to a specific model's and problem domain's requirements.

4 Algorithm Details

4.1 The PACT Activation Function

As stressed in the previous section, the PACT algorithm is based on the idea of encorparating ReLU into a framework which quantizes activations. [6] propose the PACT activation function to serve this purpose. They define $PACT\colon \mathbb{R} \to [0, \alpha]$ as

$$\text{PACT}(x) = \begin{cases} 0 & \text{if } x \in (-\inf, 0) \\ x & \text{if } x \in [0, \alpha) \\ \alpha & \text{if } x \in [\alpha, \inf) \end{cases} \tag{8}$$

where $\alpha \in (0, \inf)$ is a learnable parameter. Note that the PACT function is equal to ReLU appended to another clipping function whose clipping range is $[0, \alpha]$.

As previously stated, the idea is to leverage the training capabilities of the model to adjust α in such a way that the clipping range is the value range which carries the most "meaning" and does not disturb the inference capabilities of the network [6].

4.2 The PACT Function in the Backward Pass

Before covering the quantization-aspect of the PACT algorithm, we first look at the PACT function itself in more detail.

While its use in the forward pass is expressed by its definition, we need to derive its gradient in order to use it in the backward pass. Without quantization, using the PACT function in the backward pass is trivial. Note that we need the gradient of the loss function with respect to x but that we can calculate this gradient $\frac{\partial c}{\partial x} = \frac{\partial c}{\partial PACT(x)} \cdot \frac{\partial PACT(x)}{\partial x}$ by using the chain rule and the derivative of $PACT(x)$ with respect to x.

Whilst not being differentiable in $x = 0$ and $x = \alpha$, we can calculate the derivative of PACT based on its piecewise derivatives.

$$\frac{\partial \text{PACT}(x)}{\partial x} = \begin{cases} 0 & \text{if } x \in (-\inf, 0) \\ 1 & \text{if } x \in [0, \alpha) \\ 0 & \text{if } x \in [\alpha, \inf) \end{cases} \tag{9}$$

As such, the derivative of PACT behaves akin to the derivative of ReLU for $x \in (-\inf, \alpha]$, only for $x \geq \alpha$ the derivative is zero instead of one. Note that ReLU's supposed property of being able to pass the gradient through to previous layers is conserved for $x \in (-\inf, \alpha)$. Note further that conceptually, this range $(-\inf, \alpha)$ is meant to be the "relevant" value range for values passed into the function so ideally, the deviation from ReLU in the range $x \geq \alpha$ does not hamper the models training and inference capabilities significantly.

4.3 Training the parameter α

The parameter α is trained in the same way the other network parameters are learned by minimizing the loss function and by using gradient descent. This implies that we need to be able to calculate the derivative $\frac{\partial c}{\partial \alpha}$ of the loss function with respect to α. Recap that after obtaining this derivative, we can adjust α in the opposite direction to decrease the loss function further. Also recap that the derivative of the PACT function with respect to α is enough to infer the derivative of the loss function according to the chain rule. Without quantization, this derivative is calculated similarly to the gradient with respect to the input vector x above [22]:

$$\frac{\partial \text{PCACT}}{\partial \alpha}(x) = \begin{cases} 0 & \text{if } x \in (-\infty, 0) \\ 1 & \text{if } x \in [0, \alpha) \\ 0 & \text{if } x \in [\alpha, \infty) \end{cases} \tag{10}$$

When writing the PACT and the ReLU function as

FORWARD: $$\text{ReLU}(x) = \max(0, x) \tag{11}$$

BACKWARD: $$\frac{\partial \text{ReLU}}{\partial x}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{12}$$

FORWARD: $$\text{PACT}(x) = \max(0, \min(\alpha, x)) \tag{13}$$

BACKWARD: $$\frac{\partial \text{PACT}}{\partial x}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \text{ and } x \leq \alpha \\ 0 & \text{if } x > \alpha \end{cases} \tag{14}$$

their resemblance becomes apparent.

As we can see, the approach and calculations for incorporating the PACT function into a model without quantization is trivial. Following, we will look at how [6] encorparates this quantization into their algorithm.

4.4 Quantizing after PACT

[6] contribution is an improved approach to quantizing activations in CNNs. In the last section, we looked at the general activation function PACT. In this

section, we will look at the quantization mechanism they combine with this activation function.

PACT assumes a constant bit-width k which is used for the fixed-point number representation in their model [6].

Let y be the activation after applying the PACT function. Let y_q be the output after quantization, then y_q is calculated using

$$y_q = \text{round}(y \cdot \frac{2^k - 1}{\alpha}) \cdot \frac{\alpha}{2^k - 1} \tag{15}$$

Therefore, the activation y is quantized to the nearest value in the range $[0, \alpha]$ which is representable using k bits. The quantization is done by first scaling the input y to the range $[0, 2^k - 1]$ and then rounding it to the nearest integer. The result is then scaled back to the range $[0, \alpha]$. As discussed previously, this represents a uniform and symmetric quantization akin to the $\textbf{quantize}_k$ operator in [32]. The difference is that $\textbf{quantize}_k$ quantizes values $x \in [0, 1]$ while the above operation quantizes values in the range $x \in [0, \alpha]$ [6, 32]. As this range depends on the trained parameter α, the same bit vector b almost always encodes different real numbers in different training epochs because α changes.

4.5 Analyzing the quantization

Briefly, we want to demonstrate how to interpret the quantized values. Ideally, we could assign each quantization level to one possible configuration of our k-bit number. E.g, if $(b_1, \ldots, b_k) \in \{0, 1\}^k$ is an arbitrary bit-vector, we would match the quantized value $y_q = \frac{\alpha}{2^k-1} \cdot \sum_{i=1}^{k} \left(2^{i-1} \cdot b_i\right) \in \{0, \frac{\alpha}{2^k-1}, \frac{2\alpha}{2^k-1}, \frac{3\alpha}{2^k-1} \cdots, \alpha\}$ to it. Note the scaling factor $\frac{\alpha}{2^k-1}$ to the integer value.

[6] mention they use fixed-point multiply-accumulate (MAC) [19] for their experiments. However, fixed-point numbers require the scaling factor to be a power of 2 [28]. However, the scaling factor $\frac{\alpha}{2^k-1}$ can be arbitrary. It remains unclear to us how the authors implemented their calculations exactly.

They do, however, recommend to share the value of α per layer. Amongst yielding good accuracy [6] and working well, "this choice also reduces hardware complexity because α needs to be multiplied only once after all MAC operations in reduced-precision in a layer are completed" [6]. Also, they recommend to initialize α larger than expected as reducing α in combination with a regularizing it worked better than the other way around as per the observations by the authors. [6].

4.6 Using the PACT Function in Backward Pass

So far, we have discussed the application of the PACT function in the forward pass, specifically in combination with quantization. We have also discussed its behavior in the backward pass when values are not quantized. Lastly, we look at their combination in the context of the backward pass and backpropagation. Our objective is to compute the derivative of the loss function. Using the chain

rule, the derivative of the PACT function itself with respect to α suffices for that.

[6] use an STE for that. Similar to its application for the $\mathbf{quantize}_k$ operator in [32], the STE approximates the PACT function as the uniform distribution and approximates the derivative to be equal to the derivative of the identity function. As such, we start with the transformation $\frac{\partial y_q}{\partial \alpha} = \frac{\partial y_q}{\partial y}\frac{\partial y}{\partial \alpha}$ using the chain rule, we approximate $\frac{\partial y_q}{\partial y} \approx 1$ according to the STE and we plug in our result from equation 10 to obtain

$$\frac{\partial y_q}{\partial \alpha} \approx \frac{\partial y}{\partial \alpha} = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{16}$$

which gives the α component of the gradient and can be used in backpropagation.

4.7 Regularization

[6] also propose a regularization term for α to be added to the loss function. They use L2 regularization. Consequently, the term added per parameter (when using one value for α per layer, this means one additional regularization term per layer) equates to

$$\widetilde{c}(\omega, X, y) = c(\omega, X, y) + \lambda_\alpha \cdot \alpha^2 \tag{17}$$

or for the whole model

$$\widetilde{c}(\Omega, X, y) = c(\Omega, X, y) + \sum_{i=1}^{n} \left(\lambda_{\alpha_i} \cdot \alpha_i^2\right) \tag{18}$$

The authors' idea behind this regularizer is to depress large values of α which equate to large clipping ranges and therefore low precision in these ranges [6].

5 Applications and Future Directions

The neural network quantization field is being actively researched [28]. As the PACT algorithm was published in 2018, novel ideas have come up since surpassing PACT in terms of accuracy [8, 4, 26], hardware utilization [4], or proposing novel complementary [10] and enabling promising results with alternative approaches such as PTQ [18, 20]. In this section, we will discuss two of these algorithms to give examples of developments since PACT.

5.1 Extensions and Variations of the Pact Algorithm

Two examples for developments succeeding the PACT algorithm will be highlighted, both employing alternative quantization mechanisms. The former will be using mixed-precision quantization, intelligently varying the bit-width used for each layer. The latter is employing non-uniform quantization.

HAWQ The Hessian Aware Quantization (HAWQ) algorithm uses mixed-precision quantization, seeking to pinpoint those layers volatile to perturbation and quantization noise. They employ higher bit-widths and precision for those layers they deem to be more volatile based on the second-order derivative of the loss function. Those layers where this measure is highest are estimated to be more volatile than others [8, 28]. They use "the Hessian matrix[,] (a matrix of the second derivatives), to determine how sensitive the weights and activations are. [...] The key observation is that layers with higher Hessian spectrum (larger eigenvalues) have a more volatile loss. These layers are prone to more fluctuations in the loss when even a small amount of quantization noise is introduced (e.g., by rounding errors). [...] The idea is that a flat loss magnifies noise, e.g., quantization noise, significantly less than a region with sharper curvature in their loss. Based on this Hessian information, they manually select the bit widths for each layer. Another key insight from HAWQ is that the order in which layers are quantized is important and affects accuracy loss. [...] They argue that [...] [t]he less sensitive layers adjust well to the introduction of quantization noise, so it is better to "lock in" the quantized values of the more sensitive layers first, allowing the less sensitive layers to recalibrate during this time." [28]

Non-Uniformity Generally, non-uniform quantization of neural networks is apt "to better fit underlying distributions and mitigate quantization errors [...] by adjusting the quantization resolution according to the density of real-valued distribution." However, as new projection steps are needed to operate on non-uniformly quantized numbers, the performance overhead becomes large enough that it is usually worth maintaining look-up-tables for the operations, which still incurs significantly worse performance than the fixed-point or integer multiplication variants. [15]

[28] propose a method "maintaining the [same] hardware projection simplicity as uniform quantization". In the process, they propose a variation on the STE which they call Generalized Straight Through Estimator (GSTE). It looks at each quantization level separately and fits a differentiable slope through the interval in a way such that "the influence of the threshold parameter α_1 to the network is decently encoded in the backward approximation function". α_1 refers to the length of the segment, and therefore dictates the end of the clipping range.

As for the quantization itself, they introduce the "nonuniform-to-uniform-quantizer (N2UQ) with its forward pass formulated as:

$$x^q = \begin{cases} 0, & x^r \leq T_1 \\ 1, & T_1 \leq x^r < T_2 \\ \dots & \dots \\ 2^n - 1 & x^r \geq T_{2^n-1} \end{cases} \tag{19}$$

[15] where n is the number of bits, T represents learnable thresholds, and x^r and x^q are the real and quantized values, respectively." Using this

power-of-two quantizer [19] yielding fixed quantization levels in their design, they can achieve learnable non-uniformity through the adaptable segmentation, "while output[ting] uniformly quantized weights and activations to accommodate fast bitwise operations without the post-processing step between quantization and matrix multiplication." [15]

References

1. Bengio, Y., Léonard, N., Courville, A.: Estimating or Propagating Gradients Through Stochastic neurons for Conditional Computation (Aug 2013), arXiv:1308.3432 [cs]
2. Blockeel, H.: Hypothesis space (2011)
3. Cai, Z., He, X., Sun, J., Vasconcelos, N.: Deep Learning With Low Precision by Half-Wave Gaussian Quantization. pp. 5918–5926 (2017)
4. Chang, S.E., Li, Y., Sun, M., Shi, R., So, H.K.H., Qian, X., Wang, Y., Lin, X.: Mix and Match: A Novel FPGA-Centric Deep Neural Network Quantization Framework. In: 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). pp. 208–220 (Feb 2021). https://doi.org/10.1109/HPCA51647.2021.00027, iSSN: 2378-203X
5. Chin, H.H., Tsay, R.S., Wu, H.I.: A High-Performance Adaptive Quantization Approach for Edge CNN Applications
6. Choi, J., Wang, Z., Venkataramani, S., Chuang, P.I.J., Srinivasan, V., Gopalakrishnan, K.: PACT: Parameterized Clipping Activation for Quantized Neural Networks (Jul 2018), arXiv:1805.06085 [cs]
7. Deng, L., Li, G., Han, S., Shi, L., Xie, Y.: Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey. Proceedings of the IEEE **108**(4), 485–532 (Apr 2020). https://doi.org/10.1109/JPROC.2020.2976475, conference Name: Proceedings of the IEEE
8. Dong, Z., Yao, Z., Gholami, A., Mahoney, M., Keutzer, K.: HAWQ: Hessian AWare Quantization of Neural Networks With Mixed-Precision. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 293–302. IEEE, Seoul, Korea (South) (Oct 2019). https://doi.org/10.1109/ICCV.2019.00038
9. Elsken, T., Metzen, J.H., Hutter, F.: Neural Architecture Search: A Survey
10. Fan, A., Stock, P., Graham, B., Grave, E., Gribonval, R., Jegou, H., Joulin, A.: Training with Quantization Noise for Extreme Model Compression (Feb 2021). https://doi.org/10.48550/arXiv.2004.07320, arXiv:2004.07320 [cs, stat]
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. pp. 770–778 (2016)
12. He, Y., Zhang, X., Sun, J.: Channel Pruning for Accelerating Very Deep Neural Networks. pp. 1389–1397 (2017)
13. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size (Nov 2016), arXiv:1602.07360 [cs]
14. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (May 2015). https://doi.org/10.1038/nature14539
15. Liu, Z., Cheng, K.T., Huang, D., Xing, E., Shen, Z.: Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation (Apr 2022), arXiv:2111.14826 [cs]

16. Mishra, A., Nurvitadhi, E., Cook, J.J., Marr, D.: WRPN: Wide Reduced-Precision Networks (Sep 2017). https://doi.org/10.48550/arXiv.1709.01134, arXiv:1709.01134 [cs]
17. Mordido, G., Van Keirsbilck, M., Keller, A.: Monte Carlo Gradient Quantization. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 3087–3095. IEEE, Seattle, WA, USA (Jun 2020). https://doi.org/10.1109/CVPRW50498.2020.00367
18. Nagel, M., Amjad, R.A., van Baalen, M., Louizos, C., Blankevoort, T.: Up or Down? Adaptive Rounding for Post-Training Quantization (Jun 2020). https://doi.org/10.48550/arXiv.2004.10568, arXiv:2004.10568 [cs, stat]
19. Nagel, M., Fournarakis, M., Amjad, R.A., Bondarenko, Y., van Baalen, M., Blankevoort, T.: A White Paper on Neural Network Quantization (Jun 2021), arXiv:2106.08295 [cs]
20. Nahshan, Y., Chmiel, B., Baskin, C., Zheltonozhskii, E., Banner, R., Bronstein, A.M., Mendelson, A.: Loss Aware Post-training Quantization (Mar 2020). https://doi.org/10.48550/arXiv.1911.07190, arXiv:1911.07190 [cs]
21. Nair, V., Hinton, G.E.: Rectified Linear Units Improve Restricted Boltzmann Machines
22. Park, E., Kim, D., Yoo, S., Vajda, P.: Precision Highway for Ultra Low-Precision Quantization (Dec 2018), arXiv:1812.09818 [cs]
23. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural Networks **61**, 85–117 (Jan 2015). https://doi.org/10.1016/j.neunet.2014.09.003
24. Sharma, S., Sharma, S., Athaiya, A.: ACTIVATION FUNCTIONS IN NEURAL NETWORKS. International Journal of Engineering Applied Sciences and Technology **04**(12), 310–316 (May 2020). https://doi.org/10.33564/IJEAST.2020.v04i12.054
25. Stanton, S., Izmailov, P., Kirichenko, P., Alemi, A.A., Wilson, A.G.: Does Knowledge Distillation Really Work? In: Advances in Neural Information Processing Systems. vol. 34, pp. 6906–6919. Curran Associates, Inc. (2021)
26. Uhlich, S., Mauch, L., Cardinaux, F., Yoshiyama, K., Garcia, J.A., Tiedemann, S., Kemp, T., Nakamura, A.: Mixed Precision DNNs: All you need is a good parametrization (May 2020). https://doi.org/10.48550/arXiv.1905.11452, arXiv:1905.11452 [cs, stat]
27. Wang, Q., Ma, Y., Zhao, K., Tian, Y.: A Comprehensive Survey of Loss Functions in Machine Learning. Annals of Data Science **9**(2), 187–212 (Apr 2022). https://doi.org/10.1007/s40745-020-00253-5
28. Weng, O.: Neural Network Quantization for Efficient Inference: A Survey (Jan 2023), arXiv:2112.06126 [cs]
29. Yin, P., Lyu, J., Zhang, S., Osher, S., Qi, Y., Xin, J.: Understanding Straight-Through Estimator in Training Activation Quantized Neural Nets (Sep 2019), arXiv:1903.05662 [cs, math, stat]
30. Ying, X.: An Overview of Overfitting and its Solutions. Journal of Physics: Conference Series **1168**, 022022 (Feb 2019). https://doi.org/10.1088/1742-6596/1168/2/022022
31. Zeiler, M.D., Fergus, R.: Visualizing and Understanding Convolutional Networks (Nov 2013). https://doi.org/10.48550/arXiv.1311.2901, arXiv:1311.2901 [cs]
32. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients (Feb 2018), arXiv:1606.06160 [cs]

33. Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., Zhu, J.: Discrimination-aware Channel Pruning for Deep Neural Networks. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018)