Optimizing AutoML for Tiny Edge Systems: A Baldwin-effect Inspired Genetic Algorithm

1st Yiran Huang, 2nd Yexu Zhou, 3rd Haibin Zhao, 4th Till Riedel, 5th Michael Beigl
Pervasive Computing Systems-TECO
Karlsruhe Institute of Technology
Karlsruhe, Germany
{yhuang, zhou, hzhao, riedel, beigl}@teco.edu

***Abstract*—Tiny edge systems used in IoT devices, wearables or smart textiles are characterized by the need of processing complex sensor data streams under various device constraints. Due to the high number of constraints and the complexity of the optimization of the hyper-parameter space for machine learning based processing, genetic algorithms (GAs) seem to be a perfect fit to enable AutoML for those embedded devices. However, due to aspects such as the high interdependence between optimization parameters, the simultaneous existence of multiple conflicting objectives and complex effects of embedded feature engineering, we made the experience that GA approaches fail to converge within this high dimensional design space. We introduce a novel Genetic Algorithm (GA) customized for AutoML tasks, addressing the unique challenges posed by highly embedded machine learning domains. The proposed approach addresses parameter interdependencies through utilizing the Baldwin-effect in biological evolution, enhances resource utilization by early elimination of less promising individuals, and augments the insufficient capabilities of existing machine learning features via the integration of carefully designed neural network features. Empirical evaluations conducted on two benchmark datasets support the superiority of our proposed method over conventional genetic algorithms. Furthermore, we demonstrate the effect of the different components introduced by our algorithms through an ablation study.**

***Index Terms*—wearable human activity recognition, genetic algorithm, Baldwin effect**

I. Introduction

Tiny edge systems with Artificial Intelligence (AI) capabilities are invisibly driving digital transformation in a wide range of industries. As these systems evolve, particularly IoT devices, wearables and smart textiles, they show significant trends across multiple sectors. The application of machine learning (ML) powered sensors and actuator systems seems without limits. However, edge devices are typically battery powered, and challenges are encountered when running multiple applications with limited resources concurrently, e.g., device endurance. Additionally, the need for real-time services requires that models on these devices to be low-latency. Currently, optimizing these models relies on specialized knowledge and manual tuning, such as hyperparameter adjustment, which can be time-consuming and often inefficient.

Automatic Machine Learning (AutoML) represents a method for automating the ML process, particularly effective in enhancing the model's objective under constraints. For tiny edge systems, AutoML means automatically selecting features, model and model parameters that suit both software and hardware limitations of the edge system, such as computation core, latency, and memory capacity. These techniques are varied and primarily fall into two categories: heuristic and gradient based search strategies. Gradient based systems like Neural Architecture Search (NAS) typically incorporate hardware factors into a loss function and refines the constrained model architecture using methods such as gradient descent, as demonstrated in approaches like NEAT [1]. In contrast, heuristic searches are based on interactive candidate suggestions and evaluation. As it is often complicated to formulate the valid optimization space (such as complex resource constraints on the target hardware) and complex objective functions in NAS. Our research focuses on the heavily extensible methods based on Genetic Algorithms (GA). GAs have the additional advantage that parallelism can be easily exploited on a any training hardware [2].

Despite extensive research, the target domain of tiny embedded systems prevents the direct application of existing GA methods. These challenges include:

- Multimodal data streams often result in large feature sets. For example, in human activity recognition applications, data typically includes inputs from various sensors across different body parts. Tools such as TsFel [3] can extract up to 198 features from each single channel. Considering a standard benchmark dataset like PAMAP2 [4], TsFel can extract as many as 3564 different features for a given data sample. The huge feature set can result in two major issues when using conventional GA methods: *(i)* an overemphasis on feature selection at the expense of other parameters, and *(ii)* increased evaluation time and evaluation time variance among different candidates. This phenomenon is especially noticeable when comparing K Nearest Neighbor (KNN) with more complex models like stochastic gradient descent. The latter takes about 25 seconds for one evaluation on the PAMAP2 dataset. It is about 100 times that of the former.
- Optimizing for a target hardware involves balancing multiple, often conflicting objectives, e.g., model performance and latency. This often leads to a proliferation of non-dominated solutions, reducing the search efficiency of the conventional GA.

- The complexity of edge tasks necessitate intricate feature sets. For example, in daily life activity recognition task, models are expected to distinguish actions like brushing teeth and washing dishes. These activities are characterized by complex patterns, large subject variations, and extended duration. This compels the inclusion of features from diverse domains such as temporal, spatial, and frequent. However, traditional time-series feature extraction methods like TsFel are inadequate for extracting such multidimensional features.
- Optimizing embedded ML architectures involves a large number of parameters of different types and interdependencies, e.g., the specific model parameters depend on the chosen model. In addition, while most model parameters are numerical, feature and model selection are categorical. Different types of parameters require different optimization approaches to be addressed effectively.

Based on this, this paper introduces an innovative adaptation of genetic algorithms, demonstrating enhanced performance in solving AutoML problems within tiny edge systems. This advanced method recognizes the interdependence of parameters, treating the optimization of partial parameters as akin to the growth of individuals in a generation, reflecting the Baldwin effect of evolutionary development. It conserves resources by early elimination of suboptimal individuals. In addition, it uses process pooling and early selection to minimize the impact of extended fitness evaluation times, thus reducing excessive waiting. Moreover, the proposed approach harmonizes conflicts between multiple objectives by integrating them into unified optimization goals with different focus. To address the inherent limitations of feature extraction in ML, we customize a neural network. This network is adept at extracting features from the spatial, temporal, and frequency domains, thus expanding the range of potential features available for selection.

II. Related Work

GA are designed and developed using the principles of natural biological evolution. Their primary goal is to find optimal solutions by mimicking natural evolutionary processes. A conventional GA algorithm consists of key stages: coding, population initialisation, fitness evaluation, selection, crossover and mutation. Since GA cannot interact directly with the parameters of the problem space, it represents potential solutions as individuals in a genetic space. For example, in feature selection scenarios, individual features can be encoded as Boolean values, indicating their inclusion or exclusion in model training.

Innovative applications have been proposed in recent years. Magdum *et al.* [5] use an advanced GA to improve Artificial Neural Network (ANN) performance in Optimal Power Flow (OPF) applications, modifying ANN training to select optimal weights and biases, thereby reducing error rates and costs. Ali *et al.* [6] use a hybrid filter-GA feature selection approach to improve cancer classification accuracy in high-dimensional micro array datasets. This method uses filter feature selection

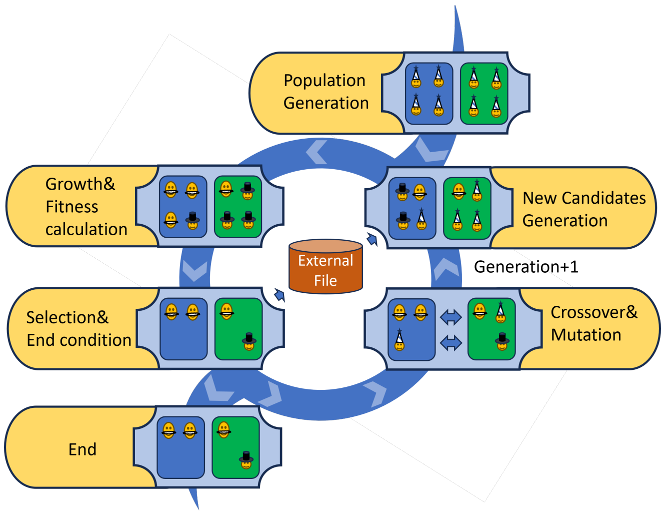

Fig. 1. The pipeline of the proposed algorithm.

techniques, such as information gain and chi-squared, to identify significant features, followed by a GA to further refine these features and improve machine learning performance metrics. However, these methods are highly task specific and may not be easily adaptable to other domains.

In terms of algorithmic advances, Blanchard *et al.* [7] propose the use of masked language models to automate mutation generation in GAs, optimising molecular string representations for drug similarity and synthesizability. This method identifies common subsequences within a population to create a vocabulary that tokenises each genome. A masked language model trained on this data generates potential genome rearrangements, overcoming the limitations of traditional pointwise mutation methods. While it is effective at the molecular level, its application to time series is hampered by a lack of consistent patterns and is resource-intensive, requiring further exploration. GAs demonstrate significant task dependency, primarily due to variations in the coding of individuals. This necessitates the creation of specialized designs focused on tiny edge systems.

III. Method

The proposed algorithm represents a specialized adaptation of the GA tailored for tiny edge systems, adhering to the conventional GA process. Fig. 1 delineates the algorithm workflow. In this section, we will systematically unveil the method, aligning with the sequential stages of the GA's workflow.

A. Individual Coding Design

The proposed algorithm aims to enhance the efficacy of models utilized in tiny edge systems through optimizing the training pipeline. It encompasses the refinement of the selection of features, models, and model parameters.

In the conventional GA, each individual is depicted as a sequence with constituent elements mirroring the values of parameters in problem space. However, when applied to typical tasks such as wearable human activity recognition

(WHAR), this approach encounters two significant hurdles: the variable length of sequences contingent upon model selection and an extensive search space for feature selection attributed to the multimodal data.

In the proposed method, we distinguish the optimization of model parameters from the other parameters. Individuals in the method are represented with a Boolean array for feature selection and an integer to indicate the model choice. On this basis, each individual maintains a function that fine-tunes the parameters of the selected model and yields the fitness score, reflecting the trained model's performance relative to its objective. It is worth highlighting that the selection of the model parameters does not act as part of an individual chromosome. It is not involved in the crossover and mutation stage of the genetic algorithm. Instead, it is a reflection of the individual's growth. As the algorithm iterates, the model parameters are gradually optimized. Additionally, each individual possesses an extra attribute: a fitness table named 'result' with two columns that log the number of times model parameters are fine-tuned within the individual and the respective optimal fitness scores achieved.

B. Populations Generation

The optimization of tiny edge models presents inherent complexity, given their multi-objective nature that often involves conflicting goals. To mitigate this, one strategy is to normalize and combine these sub-objectives. By drawing a parallel with the division of academic tracks into arts and sciences in China, we suggest the establishment of distinct population groups, each guided by a blend of sub-objectives with varying emphasisThis approach permits certain populations to focus on specific sub-objectives while preserving a comprehensive equilibrium. In addition, by distinguishing populations during their generation and encouraging interactions during the crossover phase, we aim to create a synergistic effect that produces offspring with improved characteristics, while also actively preserving diversity within the population.

C. Individual Growth and Fitness Calculation

Unlike the general GA where an individual's fitness score is fixed upon creation, the proposed method allows for the optimization (growth) of individuals, as reflected by the optimization of model parameters. Given that these parameters are predominantly numerical, we utilize Bayesian Optimization (BO) for fine-tuning, with the goal of enhancing the overall fitness of the corresponding individual. BO operates on prior probability distribution to emulate the behavior of the objective function. This distribution is iteratively refined through the evaluation of candidate points in the search space, increasingly aligning the distribution with the objective function's real distribution.

Additionally, we have introduced the concept of 'age' to track the number of evaluations conducted by BO. All individuals initially commence with an age of zero. In each iteration of the proposed method, those that are younger than a set threshold undergo the optimization (growth) process.

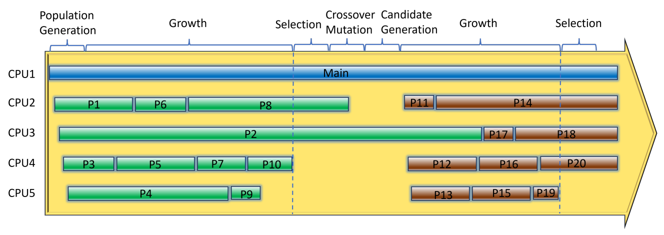

Fig. 2. The processes executed in process pool.

The fitness scores and corresponding ages are subsequently documented in the fitness table 'results'.

Due to the large number of samples in edge device tasks and the considerable feature number per sample, there is an extended growth duration (running time of the BO optimization) for each individual. To address this issue, we utilize process pool to parallelize individual growth. A process pool is created with its size matching the number of CPUs present on the device. As depicted in Fig. 2, individuals are enqueued into the process pool, and as processes become available, the subsequent individual is granted the opportunity to grow. The queuing order of the individuals is determined by the sequence in which they are introduced to the pool.

It is important to note that not all individuals undergo the selection process at each iteration. Significant variance in growth duration among individuals implies that waiting for all to complete the growth process before proceeding to the selection stage would be inefficient. Hence, we apply 'early selection' technique for process pool. Concretely, we set a threshold: once the number of individuals that have completed the growth process exceeds this threshold, they proceed directly to the selection stage. Those still in the pool bypass the current selection stage and move to the next cycle of individual selection.

D. Individual Selection

Each individual possesses a fitness table 'results', which varies in the number of rows, reflecting the differing experiences of each individual. To equitably rank matrices of varying sizes, we address this by giving preference to younger individuals who have attained superior scores and to relatively older individuals who have more experience. Consistent with this idea, we employ Algorithm 1 for ranking purposes. After the ranking, lower-ranked individuals are systematically archived, with all pertinent data, including feature and model selections, optimal parameters, and peak fitness scores, meticulously recorded in an external file for future reference. This wealth of information is instrumental in the generation of new individuals.

E. Crossover and Mutation

We adopt a dual-tier crossover strategy that includes both intra-population and inter-population information exchanges, as outlined in Algorithm 2. The selection of candidates within each population group is randomized and the exchange of characteristic and model values is carried out according to a

Algorithm 1 Individual Selection

```
Require: $results$: A dictionary with DataFrames, $ratio$: Ratio of Individual to keep
Ensure: $li\_keep$: list containing individual ids to keep
1: $all\_ages \leftarrow$ sorted unique ages from $results$
2: Initialize $rank\_df$ as a DataFrame with index as $all\_ages$
3: for each $age$ in $all\_ages$ do
4:     Initialize lists: $scores$, $ids$
5:     for each $id, df$ in $results$ do
6:         if $age$ is in $df['age']$ then
7:             $score \leftarrow$ score for the given $age$ in $df$
8:             Append $score$ to $scores$ and $id$ to $ids$
9:         end if
10:    end for
11:    $ranks \leftarrow$ rank $scores$ in descending order
12:    for each $id, rank$ in $ranks$ do
13:        Assign $rank$ to $rank\_df$ for the corresponding $age$ and $id$
14:    end for
15: end for
16: $average\_ranks \leftarrow$ mean of $rank\_df$
17: $li\_keep \leftarrow$ top items of $average\_ranks$ based on $ratio$
18: return $li\_keep$
```

predefined probability. We favor this stochastic approach over a deterministic half-swapping due to the pattern of extracted features observed in TsFel, which tends to cluster correlated features. This clustering can concentrate high-quality features within certain areas of the feature array, potentially diminishing the effectiveness of simple half-swapping.

Algorithm 2 Crossover for Feature

```
Require: $individual1, individual2$
Ensure: $child1, child2$: Two new feature arrays derived from the parents
1: $parent1 \leftarrow individual1.features$
2: $parent2 \leftarrow individual2.features$
3: Initialize $child1$ and $child2$ as zero arrays of the same shape as $parent1$ and $parent2$
4: Generate a $mask$ bool array with random value of the same length as $parent1$
5: for each index $i$ in the range of length of $parent1$ do
6:     if $mask[i]$ is 1 then
7:         $child1[i] \leftarrow parent2[i]$
8:         $child2[i] \leftarrow parent1[i]$
9:     else
10:        $child1[i] \leftarrow parent1[i]$
11:        $child2[i] \leftarrow parent2[i]$
12:    end if
13: end for
14: return $child1, child2$
```

The mutation process is aligned with the crossover strategy, based on a predetermined probability to determine the feature and model mutations. Upon activation, the mutation causes the selected feature or model values to deviate from their established state, thus injecting new variation into the GA search process.

F. New Individual Generation

Due to the vast search space, exploration relying solely on GA crossover and mutation is inefficient. To overcome this, we utilize heuristic methods, guided by the data retained from the selection process, to facilitate the creation of new individuals, thus enhancing the efficiency.

In developing specialized methods for different components of an individual based on their unique characteristics, we have implemented separate strategies for feature and model candidate generation. For the feature candidate, inspired by [8], we train two distinct multilayer perceptron (MLP) networks: one as a generator and the other as an evaluator. The generator network uses Gaussian noise to produce a one-dimensional float array with values ranging within the [0,1] interval. By applying a predefined threshold to this output, we generate a candidate feature array. In contrast, the evaluator network accepts a feature array and a model selection value as input, and outputs a single float value to estimate the fitness score of the given feature-model combination.

We train the evaluator network using the stored external file and a mean squared error (MSE) loss function. Once the evaluator network is trained, we fix its weights and integrate it with the generator network. This integration allows the generator output, paired with a random model value, to serve as the input for the evaluator. We then train the generator network by maximizing the evaluator's output, employing the straight-through estimator for effective gradient descent between the two networks. The injection of noise as input to the generator guarantees the diversity of the generated feature candidates.

For model selection, we apply the Upper Confidence Bound strategy, as documented in [9]. The strategy aggregates the total number of trials v, the number of times the model i was visited v_i and their corresponding mean fitness scores f from the external file, and ultimately calculates a score for each model i to guide the selection of the model:

$$score_i = f + \gamma\sqrt{\frac{v_i}{v}},$$

where γ is the weight of exploration. The model with the highest score is selected. This established approach strikes a balance between exploitation and exploration, thus optimizing the identification of the most viable models.

The proposed method iterates the above processes, growth, selection, crossover, mutation, and individual generation, until a predefined condition is reached.

G. Neural Network Feature Extraction

The multimodal data handled by tiny edge systems is often highly complex, placing high demands on the features extracted from the data. To this end, we designed a simple neural network. The structure of the network is shown in

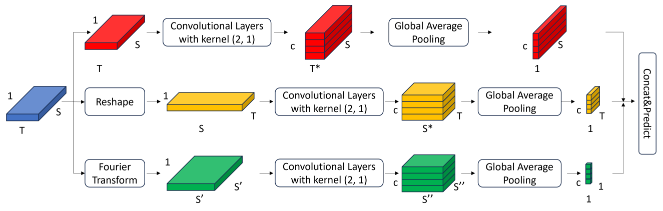

Fig. 3. The neural network for feature extraction.

Fig 3. It consists of three branches. Each branch contains a two-layers convolutional layer with output channel size 16. The network extracts features from multimodal data in terms of temporal, spatial, and frequent, respectively, and combines these features for target prediction. In the figure, T denotes the duration of time for each prediction sample, S denotes the number of sensor channels, and c denotes the number of channels generated by layers of the Convolutional Neural Network (CNN). After model training, the features generated in each branch are collected and, together with the features extracted with the ML method, serve as candidate features for target model training.

IV. Experiment

In this section, we design an experiment to validate the efficacy of our proposed algorithm. With the experiment, we aim to first compare the performance of our method with state-of-the-art genetic algorithms, and secondly examine the impact of each constituent component of the proposed algorithm, thereby isolating and understanding the contribution of individual elements to the overall effectiveness.

A. Benchmark Models

To demonstrate the performance of the proposed model and the function of each component, we compare the performance of the following algorithms: *(i)* the conventional genetic algorithm (baseline); *(ii)* the neural network proposed in the algorithm with three branches (nn); *(iii)* the proposed algorithm without using the neural network suggestion (noSuggestion); *(iv)* the proposed algorithm without individual growth (noGrowth); *(v)* the proposed algorithm without using the features generated by the designed neural network (noNN); *(vi)* the proposed algorithm with running the fitness calculation until all jobs in the pool finish (noEarlySel); *(vii)* the proposed algorithm (proposed); The experiment is run on Intel Xeon Gold 6230 processor with 96GB of memory.

B. Benchmark Datasets

In order to test the proposed method in various scenarios and at the same time maintain experimental consistency with other work, we adopt two benchmark datasets that are widely used in Wearable Human Activity Recognition (WHAR), namely HAPT [10] and PAMAP2 [4].

C. Experiment Setup

We apply the Raspberry Pi Zero W with 512Mb memory size as the potential running device. Throughout the experimental phase, for the proposed method, the maximum generation is set at 30. The selection and crossover ratios are set to 0.1 and the mutation ratio is set to 0.01. The number of evaluations applied in each growth process is set to 5. We set three sub-objectives, that is, maximizing prediction accuracy, f1 score and minimizing inference time. We set a hard constraint according to the memory size. When the memory utilized larger than 100Mb, we set the fitness value to -1. To calculate the utilized memory, we apply the 'getsize' function of 'os' package after saving the trained model with 'joblib'. Considering that subtle differences in inference time are difficult to detect in application, we use $(\text{latency}\%0.02)\lambda$ to calculate the latency fitness, where λ denotes the weights. We created two population groups, each contains 50 individuals. The objective of each group is the combination of the above three objectives with weight $[0.7, 0.2, 0.1]$ and $[0.2, 0.7, 0.1]$. We use 5-fold cross-validation to compute the fitness score to improve algorithm stability. To demonstrate the generalizability of the model, we divide the dataset into a training set and a test set with the leave-one-subject-out method. We use the training set to find the optimal parameter settings and report its performance on the test set. For the conventional genetic algorithm, we set the maximum generation to 150 to keep the number of evaluations the same as the proposed model. We keep the other parameters of the genetic algorithm the same as those of the proposed algorithm.

According to [11], to train the generator, evaluator, and feature extraction networks, we use Adam optimizer [12] with an initial learning rate of 10^{-3}. Batch training was implemented with a designating a batch size of 16. The cross-entropy loss [13] was used as an objective function for the feature extraction network. Neither early stopping nor learning rate adjustment was employed. The maximum training epoch was set to 50.

D. Discussion

The experimental results are summarized in TABLE I. It is evident from the data that the proposed model surpasses the baseline in both scenarios, thereby underscoring the efficacy of the proposed algorithm. Additionally, it was observed that the synergistic approach, combining neural network features with machine learning features, yields enhancements in performance compared to 'noNN' and 'nn' algorithms. A notable decline in performance was detected in the 'noSuggestion' algorithm, attributable primarily to model overfitting to the training dataset. Comparatively, the performance of the 'noEarlySel' and the 'proposed' algorithms are closely aligned. However, a significant difference was observed in their computational efficiency. The proposed algorithm completed its run in approximately 21 hours, whereas the 'noEarlySel' algorithm required 29 hours to achieve a similar outcome. This distinction highlights the improved computational efficiency of the proposed algorithm. In addition, the reference time for all models after optimization is less than 0.02s.

TABLE I
ACCURACY PERFORMANCE OF THE PROPOSED ALGORITHM

	baseline	nn	noSuggestion	noGrowth	noNN	noEarlyCal	proposed
HAPT	94.2	95.1	95.1	95.4	95.2	95.8	95.7
PAMAP2	85.3	80.1	73.3	85.1	83.5	85.8	85.6

V. CONCLUSION

Tiny edge systems, characterized by constraints on energy consumption, inference time, and model complexity, require specialized optimization strategies. This paper presents a novel GA variant tailored to optimize AutoML tasks in tiny edge systems, addressing the unique challenges of these environments. It includes accommodating large parameter search spaces, interdependencies between parameters, longer computation times for fitness evaluation, significant computational variances between individuals, and the inherent complexity of tiny edge data.

In addition, it is important to emphasize that while the algorithm is specifically designed for optimizing tiny edge systems, its applicability extends beyond this domain. In practical scenarios, it can be effectively applied to any optimization problem that exhibits one or more of the aforementioned characteristics.

For future work, first, more comprehensive experiments are needed for the proposed algorithm to assess its strengths and weaknesses. This includes conducting additional comparative studies with state-of-the-art methods. While our current comparisons involve high-quality methods, the range of methods evaluated is limited. Moreover, detailed exploratory experiments are required, particularly in understanding the influence of each component on end performance and determining optimal parameter selection for new datasets. Furthermore, the dataset utilized in our experiments is restricted, focusing primarily on the human activity recognition domain. Expanding the dataset to include other applications relevant to tiny edge systems is essential.

Second, while our proposed method optimizes tiny edge system from an algorithmic perspective, further investigation into the specific limitations of tiny edge systems in practical applications is warranted. It is crucial to recognize that not all methods are feasible for implementation on edge systems, making the definition of hardware-associated search spaces before applying the proposed algorithm particularly significant in this field. In addition, factors such as the duration of feature extraction and model computation, which depend on hardware specifications, must be considered in the optimization objective.

Third, experiments have shown that the simultaneous use of ML and neural network features can improve final performance. However, is this effect specific only to designed networks in the proposed algorithm, or can it be generalized to state-of-the-art networks utilized in the WHAR domain or networks generated through NAS? This is a question that deserves further investigation.

ACKNOWLEDGMENT

This work was partially funded by the German Ministry for Research and Education as part of SDI-S (Grant 01IS22095A) as well as by the Ministry of Science, Research and the Arts Baden-Wuerttemberg as part of the SDSC-BW project and by the Carl-Zeiss-Foundation as part of the JuBot project.

REFERENCES

[1] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[2] S. Mirjalili and S. Mirjalili, "Genetic algorithm," *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pp. 43–55, 2019.

[3] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa, "Tsfel: Time series feature extraction library," *SoftwareX*, vol. 11, p. 100456, 2020.

[4] A. Reiss and D. Stricker, "Introducing A New Benchmarked Dataset for Activity Monitoring," in *2012 16th international symposium on wearable computers*. IEEE, 2012, pp. 108–109.

[5] V. B. Magdum, S. Kumar, and M. Tarambale, "Opf-aga: Optimal power flow using advance genetic algorithm," in *2022 2nd International Conference on Intelligent Technologies (CONIT)*. IEEE, 2022, pp. 1–7.

[6] W. Ali and F. Saeed, "Hybrid filter and genetic algorithm-based feature selection for improving cancer classification in high-dimensional microarray data," *Processes*, vol. 11, no. 2, p. 562, 2023.

[7] A. E. Blanchard, M. C. Shekar, S. Gao, J. Gounley, I. Lyngaas, J. Glaser, and D. Bhowmik, "Automating genetic algorithm mutations for molecules using a masked language model," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 793–799, 2022.

[8] Y. Huang, Y. Zhou, M. Hefenbrock, T. Riedel, L. Fang, and M. Beigl, "Universal distributional decision-based black-box adversarial attack with reinforcement learning," in *International Conference on Neural Information Processing*. Springer, 2022, pp. 206–215.

[9] Y. Huang, N. Schaal, M. Hefenbrock, Y. Zhou, T. Riedel, and M. Beigl, "Mcxai: local model-agnostic explanation as two games," in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 01–08.

[10] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-Aware Human Activity Recognition Using Smartphones," *Neurocomputing*, vol. 171, pp. 754–767, 2016.

[11] Y. Huang, "Standardizing your training process for human activity recognition models : a comprehensive review in the tunable factors," Karlsruhe, [2023]. [Online]. Available: http://dx.doi.org/10.5445/IR/1000163013

[12] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] J. Shore and R. Johnson, "Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy," *IEEE Transactions on information theory*, vol. 26, no. 1, pp. 26–37, 1980.