Probabilistic Prediction of Energy Demand and Driving Range for Electric Vehicles with Federated Learning

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)

angenommene

DISSERTATION

von

M.Sc. Adam Thor Thorgeirsson

Tag der mündlichen Prüfung:	25.04.2024
Hauptreferent:	Prof. Dr. rer. nat. Frank Gauterin
Korreferent:	Prof. Dr.-Ing. Eric Sax

Kurzfassung

Probabilistische Vorhersage des Energiebedarfs und der Reichweite von Elektrofahrzeugen mittels föderierten Lernens

Vernetzte Fahrzeuge und Backend-Infrastrukturen bilden ein verteiltes System, in dem große Mengen an Fahrzeugdaten erzeugt werden. Maschinelle Lernalgorithmen können diese Daten nutzen, um Vorhersagen und Prognosen künftiger Ereignisse zu verbessern. Die dezentrale Datenerhebung stellt jedoch eine Herausforderung dar. Fahrer von batterieelektrischen Fahrzeugen müssen mit einer begrenzten Reichweite und einer limitierten Ladeinfrastruktur zurechtkommen. Eine genaue Vorhersage des Energiebedarfs und der Reichweite ist daher wichtig und ermöglicht zuverlässige Routenberechnung und Ladeplanung.

Maschinelle Lernalgorithmen verwenden große Datenmengen und stellen hohe Anforderungen an die Rechenleistung. Eine herkömmliche Platzierung der entsprechenden Software in einem Steuergerät (ECU) eines Fahrzeugs kann zu hohen Latenzzeiten führen und damit die Benutzerfreundlichkeit beeinträchtigen. Durch eine intelligente Verteilung der Software-Module über die Fahrzeugflotte und das Backend, können hohe Latenzzeiten vermieden werden. Ein verteiltes System sollte die Unsicherheit von Daten und Vorhersagen berücksichtigen. Diese Unsicherheiten können direkt durch den Einsatz probabilistischer Lernalgorithmen berücksichtigt werden.

In dieser Dissertation werden verteilte und probabilistische Vorhersagealgorithmen sowie Systemarchitekturen für verteilte Systeme untersucht. Verschiedene Systemarchitekturen und Modulplatzierungen werden im Hinblick auf Latenzzeiten, Netzwerknutzung, Energieverbrauch und Kosten analysiert. In einem verteilten System können föderierte Lernalgorithmen wie *Federated Averaging* angewandt werden. Diese Algorithmen berücksichtigen jedoch keine Unsicherheiten der Vorhersagen. In dieser Dissertation wird daher eine Erweiterung von *Federated Averaging*, FedAvg-Gaussian, vorgestellt, die probabilistische neuronale Netze und lineare Regressionsmodelle kommunikationseffizient und unter Wahrung des Datenschutzes trainieren kann. Die Vorhersagealgorithmen werden anhand von realen Fahrdaten validiert.

Die Ergebnisse der Arbeit zeigen, dass ein zwischen dem Fahrzeug und der Backend-Infrastruktur verteiltes System die Ende-zu-Ende-Latenzzeiten im Vergleich zu einer herkömmlichen fahrzeugbasierten Platzierung um bis zu Faktor Zehn reduziert. Ebenso wird die Netzwerknutzung im gleichen Maße reduziert. Die Vorteile von probabilistischen Vorhersagealgorithmen gegenüber deterministischen Vorhersagealgorithmen werden mit *Proper Scoring Rules* demonstriert. Föderiertes Lernen kann das konventionelle, fahrerindividuelle Lernen verbessern und mit probabilistischen Vorhersagen können variable Energiereserven auf Basis der Ankommenswahrscheinlichkeit angewandt werden. In einer virtuellen Testumgebung mit der realen Straßen- und Ladeinfrastruktur werden die Wechselwirkungen zwischen Energiebedarfsvorhersage, Routenberechnung und Ladeplanung untersucht. Die somit ermöglichten genauen Vorhersagen mit Unsicherheitsbetrachtung führen zu einer erhöhten erlebbaren Reichweite und zu einer reduzierten Reisedauer für Fahrer von batterieelektrischen Fahrzeugen.

Abstract

Connected vehicles and backend infrastructures comprise a distributed system, in which large amounts of data are generated. Machine learning algorithms can use this data to improve predictions and forecasts of future events. However, the distribution of the data is a challenge. Today's drivers of battery electric vehicles must deal with limited driving range in a sparse charging infrastructure. An accurate prediction of energy demand and driving range is therefore important and enables reliable routing and charge planning applications.

Machine learning algorithms use a large amount of data and have high computational requirements. A traditional placement of the software within a vehicle's electronic control unit (ECU) could lead to high latencies and thus detrimental to user experience. High latencies can be prevented with intelligent distribution of the algorithm parts over the vehicle fleet and backend. A distributed system should take the uncertainty of data and predictions into account. Predictive uncertainty can be regarded directly with the use of probabilistic machine learning algorithms.

In this dissertation, distributed and probabilistic prediction algorithms as well as system architectures for distributed systems are investigated. Different system architectures and module placements are analyzed in terms of latency, network usage, energy usage and cost. Federated learning algorithms such as FedAvg can be applied in this distributed setting, but predictive uncertainty is typically not considered. In this dissertation, an extension of the federated averaging algorithm, FedAvg-Gaussian, is applied to train probabilistic neural networks and linear regression models in a communication-efficient and privacy-preserving manner. The prediction algorithms are validated using real driving data.

The results show that a system distributed between the vehicle and the cloud reduces the end-to-end latency by up to a factor of 10, when compared to a traditional vehicle-based placement. Likewise, the network usage is decreased to the same degree. The performance advantage of probabilistic prediction models over deterministic prediction models is demonstrated using proper scoring rules and it is shown that federated learning can improve the standard, driver-individual learning. Using probabilistic predictions, routing and charge planning based on destination attainability can be applied. Using a simulation framework with real road and charging infrastructure, it is shown that accurate probabilistic predictions lead to increased effective driving range and reduced travel time.

Preface

The research conducted for this dissertation was a collaborative effort between the KIT Institute of Vehicle Technology and the Department of Energy Management at Dr. Ing. h.c. F. Porsche AG. I would like to express my appreciation to all those who played a pivotal role in making this endeavor possible.

First and foremost, I am immensely grateful to my advisor, Prof. Dr. rer. nat. Frank Gauterin. Your dedication to excellence and your ability to inspire have made this journey not only educational but also immensely rewarding. Furthermore, your feedback and expert insights proved to be extremely valuable.

I extend my heartfelt thanks to my second examiner, Prof. Dr.-Ing. Eric Sax. Your feedback, rigorous examination, and thought-provoking questions have significantly enriched this thesis and my understanding of the subject matter.

I would like to thank my research group leader, Dr.-Ing. Michael Frey. Your support, guidance, and helpful discussions have been very important for my academic progress.

I want to express my sincere appreciation to my supervisor at Porsche, Dr.-Ing. Marc Albrecht. Your trust and leadership fostered a highly conducive environment for my research.

I am indebted to my colleagues and mentors Dr.-Ing. Moritz Vaillant, Dr.-Ing. Stefan Scheubner, Dr.-Ing. Sebastian Fünfgeld, and Dr.-Ing. Péter Megyesi at Porsche. Your camaraderie, encouragement, and willingness to collaborate made the experience both enjoyable and intellectually stimulating. The countless discussions, brainstorming sessions, and shared research adventures have been invaluable.

I wish to convey my thanks to Patrick Petersen from the FZI Research Center for Information Technology. The collaboration with you, along with our discussions, was both enjoyable and highly productive.

I'd like to offer my heartfelt thanks to the dedicated students, Yanglou Yue, Thomas Fischbach, Carsten Binz, Marcel Brödel, and Pascal Nießner, who supported me on this research journey. Your enthusiasm for learning and commitment to our projects have been truly commendable.

In addition to expressing my gratitude to my academic and research mentors and colleagues, I would also like to take a moment to extend my deepest thanks to my family and, in particular, my loving wife Elvira. I want to acknowledge that your contributions to my personal and academic growth have been immeasurable. I promise to continue working diligently to make you proud. Thank you, from the bottom of my heart, for being my source of strength, love, and inspiration.

Karlsruhe, September 2023 *Adam Þór Þorgeirsson*

Contents

Acronyms and symbols

Acronyms

API	application programming interface
B	battery
BDL	Bayesian deep learning
BEV	battery electric vehicle
BLR	Bayesian linear regression
BNN	Bayesian neural network
CAN	controller area network
CART	classification and regression trees
CISC	complex instruction set computer
CLT	central limit theorem
CP	charging point
CRPS	continuous ranked probability score
DE	deep ensembles
DSRC	dedicated short range communication
DSVGD	distributed Stein variational gradient descent
DS	determinant sharpness
ECE	expected calibration error

ECU	electronic control unit
EDP	energy demand prediction
EM	electric motor
ES	energy score
FedAvg	federated averaging
FedAvgC	FedAvg-Clustering
FedAG	FedAvg-Gaussian
FedAGC	FedAG-Clustering
FedBE	federated Bayesian ensemble
FedPA	federated posterior averaging
FLOPS	floating point operations per second
FL	federated learning
GB	gearbox
GD	gradient descent
HMC	Hamiltonian Monte Carlo
HTTP	Hypertext Transfer Protocol
HVAC	heating, ventilation and air conditioning
ICEV	internal combustion engine vehicle
IID	independent and identically distributed
IoT	internet of things
LR	linear regression
LSTM	long short-term memory
MAE	mean absolute error

MCE	maximum calibration error
MCMC	Markov chain Monte Carlo
MC	Monte Carlo
MIPS	million instructions per second
ML	machine learning
NLL	negative log-likelihood
NN	neural network
OEM	original equipment manufacturer
OMC	ordinary Monte Carlo
PBP	probabilistic backpropagation
PE	power electronics
PVI	partitioned variational inference
QoS	quality of service
ReLU	rectified linear unit
RISC	reduced instruction set computer
RMSCE	root mean square calibration error
RMSE	root mean square error
SGD	stochastic gradient descent
SoC	state of charge
SoE	state of energy
SVM	support-vector machine
SWAG	stochastic weight averaging Gaussian
TLS	transport layer security

TMC	traffic message channel
TRDB	traffic and routing database
UI	user interface
UX	user experience
V2I	vehicle-to-infrastructure
V2V	vehicle-to-vehicle
V2X	vehicle-to-everything
VI	variational inference

Latin symbols and variables

A	frontal area
a	attainable
B	batch size
b	batch
C	fraction of devices in federated learning
c	latency of communication
c_d	aerodynamic drag coefficient
$\mathcal{D}$	data
$\mathcal{D}_k$	data on device k
D	destination
d	target variable dimension
$\mathbb{E}$	expected value

E	edge of graph
E_b	battery energy
E_c	energy consumption
E_p	number of epochs
$\mathcal{F}$	variational free energy
F_r	driving resistance
f_r	coefficient of rolling resistance
$\mathcal{G}$	road network graph
H	Heaviside function
h	hidden unit
$\mathcal{I}$	number of iterations
$\mathbf{I}$	identity matrix
I	bias
i	index variable
j	index variable
K	total number of devices
k	index variable
$\mathcal{L}$	logarithm score
L	loss function
l	route length
M	number of Monte Carlo simulations
m	mass

$\mathcal{N}$	normal distribution
N	number of...
n	number of...
P	power
P_t	true distribution
P_i	i-th percentile
p	probability
Q	battery capacity
Q_p	well defined distribution
q	battery state of charge
R	range
R_E	earth radius
$\mathcal{S}$	driver cluster
S	response size
S_k	road segment
S_R	scoring rule
S_t	set of clients in round t
$\mathcal{T}_{\mathcal{I}}$	time for iteration
T	time
t	communication round
u	traffic speed
V	vertex of graph

v	velocity
w	model weight
x	input variable
y	target variable
$\mathbf{z}$	road segment features

Greek symbols and variables

α	road slope
β	noise precision parameter
Δ	safety margin
δ	random variable
ϵ	error
ζ	risk threshold
η	efficiency
η_l	learning rate
Θ	traffic phase
θ	precision parameter for prior distribution
γ	random variable
κ	road curvature
Λ	street class
λ	longitude
μ	mean value

ξ	calibration accuracy
π	route
ρ_d	speed distribution percentile of driver d
Σ	covariance matrix
σ	standard deviation
τ	charging time
Φ	inverse cumulative distribution function
ϕ	latitude
Ψ	population
ω	unordered combination

General deep indexes

air	air-line
aux	auxiliary
dyn	dynamic
iter	iterations
kin	kinetic
lim	limit
nom	nominal
req	required
seg	segment
stat	static

vars	variables

1 Introduction

Modern vehicles are connected with each other, as well as to backend infrastructures in the cloud. The connected vehicles and backend infrastructures constitute a distributed system, in which large amounts of data are generated and processed. The abundance of data sources in this vehicular network enables the widespread, commercial application of machine learning (ML) algorithms. In particular, predictions and forecasts can greatly profit from the rich database resulting from the connectivity. However, the distributed nature of the data and resources poses a challenge. The data are distributed over a large group of end devices and are commonly transferred to a central server, where the learning of the ML models can be performed using the entire dataset. A ML algorithm utilizing data from this network must guarantee the privacy of the users and be able to function without excessive computation and communication overhead. In federated learning (FL), each end device learns from local data, and a centralized server creates a global model by aggregating the model weights received from the devices at regular intervals. The global model is then sent back to the devices where the learning continues [156]. FL algorithms, such as federated averaging (FedAvg), are typically applied when a large dataset is desired, but sharing data between devices is not possible or too expensive. In a distributed setting, data may not be independent and identically distributed (IID) and a robust model should take uncertainty into account. In many applications, uncertainty of estimations or predictions can be significant. However, FL is not commonly applied to probabilistic models. Bayesian deep learning (BDL) is typically applied to account for uncertainty in neural networks (NNs) [166]. However, BDL methods are computationally expensive in comparison to non-Bayesian methods and hardware may as well be a limiting factor [134]. The inclusion of predictive uncertainty in distributed settings should therefore be addressed. The dissertation presents FedAvg-Gaussian (FedAG), where uncertainty is introduced in the aggregation step of the FedAvg algorithm by treating the set of local weights as a posterior distribution for the weights of the global model.

The prediction of the energy demand and driving range of battery electric vehicles (BEVs) is a prediction problem frequently solved with ML algorithms [58]. An energy demand prediction (EDP) algorithm predicts the energy demand for a given, planned route. Based on the EDP, the remaining driving range can be computed. The uncertainty of the predictions is important and the advantage of a large database can be considerable. The EDP algorithms rely on data distributed between vehicles and backend infrastructures. In the vehicle, information such as vehicle speed and acceleration are relevant. In backend infrastructures, information such as live traffic speed and road slope are important features for the EDP. Unifying and integrating these sources is an issue that currently hinders further development of range prediction and charge planning software [202]. By applying ML algorithms, many different features can be included in the range prediction, without the need of exact mathematical or physical modeling of their influences on the energy consumption. Furthermore, an online ML algorithm can automatically adapt to changes in system behavior, resulting in a robust model. The EDP and driving range prediction rests upon information about the driver, vehicle, route, traffic, and other environmental factors. Because of the high number of influence factors, large amounts of predictive data are required for an accurate prediction. Few researchers have addressed the issue of uncertainty of these predictions [5]. By applying FedAG, the uncertainty is directly taken into account.

In current literature about energy demand and driving range prediction algorithms, system architecture and the practicability of the proposed methods is rarely investigated. Researchers frequently propose ML algorithms that rely on big data and distributed computing, yet the analysis from a systems aspect is neglected. Not all devices and system architectures are ready for using ML software. This is a problem adressed in the *Machine Learning and Systems (MLSys)* whitepaper [188], where the authors discuss problems regarding the widespread use of ML systems in commercial applications, such as:

- *How can ML algorithms and systems be designed for device constraints such as power, latency, and memory limits?*
- *How should distributed systems be designed to support ML training and serving?*

These and similar problems have been analyzed generally in the context of the internet of things (IoT) paradigm and how IoT with its cloud, edge, and fog computing can support the increasing amount of data that is transferred and used, e.g., for ML-based systems [49]. An important aspect of system design is the user experience (UX) and how applications can be deployed to ensure high quality of service (QoS) [154]. Distributed computing in cloud, edge, and fog systems is highly dependent on communication and can exhibit high system complexity [43]. Even with the emergence of 5G, lean communications are still essential and resources need to be used efficiently [64]. Despite these challenges, connected vehicles can achieve significant improvements in efficiency, performance and QoS with ML and IoT applications. The evaluation of system architecture is therefore included in the dissertation.

1.1 Motivation and Significance

The call for low or zero emissions vehicles, along with improved battery technology, makes the BEV a serious candidate for the replacement of internal combustion engine vehicles (ICEVs). Despite the advantages of such vehicles, they have not gained significant popularity among the general public. BEVs have the potential to solve future problems regarding greenhouse gas emissions from passenger and commercial vehicles and establish independence from depleting fossil fuel resources. Due to limited charging-infrastructure and the inevitably shorter driving range, BEV drivers may experience range anxiety, which is the fear that the energy storage will run out before reaching the destination [70]. To increase driving range of BEVs, choosing a larger battery capacity is one option. However, a large battery corresponds to high vehicle weight, costs, and demand for rare minerals. Therefore, a solution where existing driving range can be fully utilized is better. In order to eliminate range anxiety and increase the usability of BEVs, there is a need for applications that help drivers in arriving safely at their destinations without excessive time or cost. The primary goals of such applications are to maximize the effective driving range and to accurately predict this range. Drivers tend to reserve up to 20 % of the battery capacity as a safety margin [82], i.e., the utilization of the available battery energy is poor. The utilization of the battery strongly depends on the calibration of the driving range prediction [2]. A central challenge in this context is the prediction of future energy demand. The EDP is not only used to display remaining driving range [243], but also for other purposes such as the prediction of a destination's attainability [5]. If the destination is not attainable, charge planning is needed. Charge planning suggests suitable charging points along the route, considering battery capacity and the predicted energy demand for the planned trip [163, 209]. Additionally, EDP have been applied in energy optimal control [232, 237], BEV fleet management systems [81] and charging infrastructure planning [233]. An accurate EDP can thus contribute in many ways to improve BEVs and their acceptance.

1.2 Dissertation Goals and Contributions

This dissertation presents research and development of methods and techniques for the advancement of the topic of energy demand and driving range prediction. The advantages and benefits of the application of new methods to practical problems in the field are shown and discussed. In contrast to related literature, the focus is not only on the evaluation of the accuracy of the predictions, but additionally on the quantification of the significance of this work in the applications of the prediction results in routing and charge planning.

The main goals of the dissertation are:

1. Evaluate the effect of system architecture and module placement of distributed driving range prediction algorithms on QoS metrics.

2. Develop a probabilistic learning scheme for distributed systems where data from a fleet of vehicles are used to compute probabilistic predictions for energy demand and driving range.
3. Investigate the application of probabilistic predictions in routing and charge planning software and evaluate the effect of the prediction performance on daily driving.

Consequently, the central contributions of the dissertation are the following:

1. It is shown that performance and QoS of range prediction and charge planning software is highly dependent on system architecture. Therefore, software concepts should be evaluated with respect to system architecture.
2. An extension of the FedAvg algorithm, FedAvg-Gaussian, is proposed, where predictive uncertainty is introduced in the aggregation step. Thereby, probabilistic ML models can be learned in an efficient manner.
3. The advantages of probabilistic predictions, computed with FedAG, are demonstrated using open-source datasets as well as in the prediction of energy demand and driving range. Proper scoring rules are used to compare the prediction results to benchmark algorithms.
4. The probabilistic EDP is used to compute destination attainability levels based on the available battery energy. Both initial predictions and the evolution of the predictions during the trips are analyzed.
5. The effects of EDP on routing and charge planning and the everyday usability of BEVs are investigated. A positive effect of predictive performance on travel time is shown and the inter-dependencies between destination attainability and travel time are illustrated.

1.3 Dissertation Organization and Previous Publications

In this section, the organization of the dissertation is described. Several portions of this dissertation are based on the author's previous publications, which are mentioned in the following overview. These publications have served as essential foundations for the chapters within this thesis. While certain sections have been directly reproduced, care has been taken to integrate them seamlessly into the broader narrative of this dissertation. [1]

Chapter 2 presents an overview of related literature on the prediction of energy demand and driving range. Relevant influence parameters on energy demand are put in context with the available data and candidate prediction algorithms are discussed.

Chapter 3 introduces a thorough analysis of the digital eco-system, in which the energy demand and driving range prediction operates. Possible system architectures and software module placements are analyzed with respect to performance indicators such as end-to-end latency and network usage. The material and results in the chapter are based on the author's journal article "Evaluating System Architectures for Driving Range Estimation and Charge Planning for Electric Vehicles", which appeared in *Software: Practice and Experience* in 2021 [4].

Chapter 4 presents FedAG, a novel machine learning algorithm for learning probabilistic models in distributed settings. The algorithm extends the FedAvg algorithm to include predictive uncertainty. FedAG is validated with empirical data from open-source datasets and compared to benchmark algorithms. The chapter is based on the author's journal article "Probabilistic Predictions with Federated

[1] It is important to note that all material used in this thesis, including the verbatim text from the mentioned publications, constitutes original contributions by the author. The specific content utilized in this thesis is solely the author's work and does not include contributions from co-authors.

Learning", which appeared in *Entropy* in 2020 [1] and on the poster presentation "Federated Learning with Predictive Uncertainty" [7].

Chapter 5 shows the application of FedAG to the EDP problem. The predictions are evaluated with proper scoring rules. Destination attainability levels are computed and their calibration is shown. The findings in the chapter are based on the author's journal article "Probabilistic Prediction of Energy Demand and Driving Range for Electric Vehicles with Federated Learning", which appeared in the *IEEE Open Journal of Vehicular Technology* in 2021 [3].

Chapter 6 presents an investigation into the effects of the EDP on routing, charge planning, and long-distance driving with BEVs. The chapter builds upon material and methods presented by the author in the following sources: the journal article "An Investigation into Key Influence Factors for the Everyday Usability of Electric Vehicles", which appeared in the *IEEE Open Journal of Vehicular Technology* in 2021 [2], as well as in the conference presentations "Analysis of the Impact of Range Estimation Errors on Long-Distance Electric Vehicle Trips" [6] and "Probabilistic Energy Demand Prediction, Routing, and Charge Planning for Electric Vehicles" [8].

In Chapter 7, concluding remarks summarize the findings of the dissertation. Additionally, possible directions for further work are discussed.

2 Prediction of Energy Demand and Driving Range

In this chapter, the fundamental background of the prediction of energy demand and driving range is introduced. The central task in driving range prediction is to first predict energy demand based on the available predictive data for the planned route. Because of complex relationship between driver behavior, traffic situations and vehicle energy consumption patterns, ML algorithms are applied. The focus of this work lies in the prediction of tractive energy demand, as this makes up the largest part of the total energy demand. Under normal conditions, i.e., mild temperatures and moderate to high driving speed, auxiliary consumers make up approximately 5 to $10\,\%$ of the total energy demand. Furthermore, many auxiliary consumers such as infotainment systems and electronic control units (ECUs) account for a relatively constant load and do not have to be predicted separately. In certain conditions, such as extreme heat or very low ambient temperatures, the energy demand of heating, ventilation and air conditioning (HVAC) systems is significantly higher than under normal conditions [149, 118, 240]. In that case, a separete prediction of auxiliary energy demand is feasible. The prediction of thermal management and HVAC energy demand falls out of the scope of this dissertation, but was discussed by Engel et al. [72], Valentina et al. [215], and Enthaler et al. [73].

The chapter is organized as follows. Section 2.1 presents a survey of related literature on energy demand and driving range prediction. Section 2.2 gives an overview of the parameters influencing the energy demand of BEVs and how these parameters can be included in the prediction algorithm. Section 2.3 briefly introduces ML algorithms for prediction problems and the challenges in the application of ML algorithms in the EDP. Finally, Section 2.4 puts the EDP into context with routing and charge planning.

2.1 Literature Survey

Current practice in energy demand prediction (EDP) is to use information from the vehicle, such as driving speed, acceleration, and historic energy consumption together with predictive information about the planned route from a traffic and routing database (TRDB). TRDB information comprises static map data, e.g., road slope, legal speed limit, and dynamic data such as live traffic. Thus, a range prediction algorithm shall use both data from the ego vehicle and from other connected vehicles via the cloud. Live route and traffic information from a TRDB is already available in some production vehicles. Traditionally, this is transmitted in a low resolution over the traffic message channel (TMC). Another standard, OpenLR, allows more flexibility and a higher resolution. An overview of live traffic related data formats and protocols was given by Henrickson et al. [109]. The prediction of future energy consumption is frequently performed using ML algorithms [46, 58, 84, 205]. The ML algorithm is trained to find the relation between the available predictive information and the resulting tractive energy consumption. The main advantage of ML algorithms is that an exact modeling of the mathematical relation between a feature and the target variable is not necessary, or rather, the ML algorithm automatically creates this model. As an alternative to ML algorithms, future energy consumption can be estimated using mechanistic models based on physical principles and equations [163, 234, 101, 122, 133]. In addition, a hybrid model combining a mechanistic model and ML can as well be applied [209, 5].

Traditionally, automotive software is implemented on ECUs within the vehicle and uses mostly data from the vehicle itself. This is also true for range prediction and charge planning. Some research articles, however, have proposed distributed systems with software placed partially or completely in the cloud. Table 2.1 shows a summary of these related works. The articles are classified by the method used for range prediction and by the system architecture which corresponds to the proposed concepts. The range prediction methods can be divided into ML models, mechanistic models, and hybrid models combining the two. Regarding system architecture, it can be observed whether the software modules are placed mainly in the cloud, in the vehicle or divided uniformly between the two in a hybrid manner.

Table 2.1: Summary of related works on driving range prediction in distributed systems.

Work	**Prediction Method**	**Architecture**
Thibault et al. [209]	hybrid model	hybrid
Fukushima et al. [84]	ML model	cloud-based
Yi et al. [234]	mechanistic model	vehicle-based
Grubwinkler et al. [101, 100]	mechanistic model	cloud-based
Jayakumar et al. [122]	mechanistic model	hybrid
Scheubner et al. [5]	hybrid model	vehicle-based
Ferreira et al. [77]	ML model	vehicle-based
Lee et al. [136, 137]	ML model	cloud-based

In the articles, the system architectures are only vaguely described and the feasibility of the proposed concepts regarding system performance is not investigated. In further research articles, proposed algorithms rely on live-information such as traffic or weather, which is normally not available in the vehicle without some sort of connectivity [205, 46, 244, 212, 211, 186]. However, system architecture and performance is not investigated.

2.2 Influence Parameters and Available Data

The EDP computes predictions for future energy demand based on information about parameters that influence energy consumption. A review of parameters influencing the energy demand and driving range of BEVs was presented by Smuts et al. [201]. The parameters can be divided into four groups [141]:

1. vehicle,
2. driver,
3. static environment,
4. dynamic environment.

The parameters in the first group are all vehicle-specific parameters that influence the energy consumption. These can be summarized as the driving resistances of the vehicle, i.e., mass, aerodynamic drag area, and rolling resistance coefficient [191]. Furthermore, the efficiency and losses in components of the vehicle's powertrain influence the consumption significantly [20]. The second group represents the driver, which has a significant influence on the energy consumption through his driving style, as the speed and acceleration behavior are important influence factors [37]. The third group, static environment, constitutes the parts of the environment that remain relatively constant, such as the road and charging network [151]. The topology of the route, especially road grade, road curvature, and speed limits are important influence factors [238, 146]. Furthermore, the charging network has an effect on the course of the route and can even influence driver behavior [66]. The fourth and last group is the dynamic part of the environment, i.e., environmental variables that change over time. Traffic is an important factor that directly influences speed and acceleration behavior [5]. Ambient temperature can affect the efficiency and losses in the vehicle [238]. Solar radiation [72], wind speed [235], and road surface wetness [71], can have an impact on the energy consumption, but it is difficult to obtain reliable predictive data for these factors. In practice, a significant impact on tractive energy consumption caused by variations of these factors is seldom, but may become more frequent with climate change [123].

Using the vehicle's sensors, information about the vehicle and the driver can be gathered. Furthermore, the surroundings of the vehicle can be observed, e.g., with cameras and radar. For environmental data beyond the reach of the sensors, information is gathered from the TRDB. In the EDP, relevant parameters can either act as input variables in the prediction model or as an inherent part of the model. The decision whether and how a parameter is used in the EDP is based on three factors:

- parameter importance,
- parameter variance,
- predictive parameter availability.

If a variable has an insignificant influence on energy demand, it does not have to be directly considered in the prediction model. Some parameters may have an important influence but exhibit a small variance. Such parameters can be modeled as an inherent part of the model. An example is the aerodynamic drag area of the vehicle, which is typically a constant. The third factor is the availability of predictive information. Some parameters have both high importance and high variance but predictive information about these parameters is severely limited or non-existent. An example is the coefficient of rolling resistance between the tires and the road, which has high influence on the energy consumption of the vehicle. As the coefficient is dependent on numerous factors such as temperature, road surface, and road surface wetness, there is generally no information available on the variability of the coefficient for a given route [61].

If the effects of a parameter, or the variability of the effects, cannot be included in the prediction model, they can be seen as parts of the predictive uncertainty [162]. Here, aleatoric and epistemic predictive uncertainty may be distinguished. Aleatoric uncertainty is also known as statistical uncertainty, and epistemic uncertainty is also known as systematic uncertainty. The aleatoric part is irreducible and is the result of intrinsic randomness of a physical relation. The epistemic part originates from lack of knowledge about the modelled relation [112]. Due to the lack of predictive information about influence parameters, it is necessary to model the epistemic uncertainty of the EDP. Epistemic uncertainty is a central part of Bayesian inference, which are discussed in Chapter 4 [130].

2.3 Online Machine Learning Algorithms

Depending on the ML algorithm and the amount of data, the prediction algorithm can demand a lot of resources. Smart system architectures are required to deal with distributed data sources and to minimize end-to-end latencies. The data observed in the vehicle is *streaming data*, which arrives sequentially. For streaming training data, online learning algorithms are a good choice [29]. With these, the learning can be done with a single pass over the observed data, which are then discarded. Optionally, the data may be saved, which leads to a set of training data, enabling the application of batch learning algorithms in an offline learning setting. Usually, the data cannot be stored completely and conventional batch learning is not applicable. Therefore, the ML algorithm must be able to run online by learning iteratively from single samples or from small batches, such as in the form of a sliding window of samples. A significant benefit of online learning algorithms is that the range prediction model is maintained live and can improve the prediction on-the-go. The algorithms can therefore flexibly react to potential *concept drifts*, such as changes in driver behavior or vehicle properties [223]. Fig. 2.1 shows a schematic overview of the data stream structure. In the single pass setting of online learning, the model is updated based on a single observation $\{\mathbf{x}, y\}_i$, where $\mathbf{x}$ is a vector of input variables and y is the target variable. The model can also be updated online based on a small batch of observations $\{\mathbf{x}, y\}_{i-2\ldots i}$ in a sliding window.

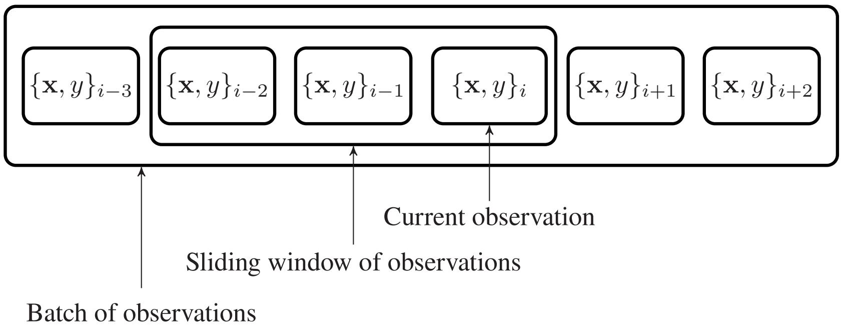

$\{\mathbf{x}, y\}_{i-3}$
$\{\mathbf{x}, y\}_{i-2}$
$\{\mathbf{x}, y\}_{i-1}$
$\{\mathbf{x}, y\}_{i}$
$\{\mathbf{x}, y\}_{i+1}$
$\{\mathbf{x}, y\}_{i+2}$
Current observation
Sliding window of observations
Batch of observations

Figure 2.1: Data structure of the learning schemes.

Not all ML algorithms are equally well applicable to online learning from data streams. Tree-based learning algorithms, such as classification and regression trees (CART), random forests, and gradient boosting, are typically applied to complete datasets [30]. To apply such algorithms in an online learning setting, special extensions are required, such as *Streaming Gradient Boosting* [114] or *Streaming Random Forests* [11]. Support-vector machines (SVMs) also require special online learning algorithms such as LASVM to function properly with streaming data [34]. Neural networks (NNs) are naturally suited for the application of online learning from data streams. Deep learning of NNs uses the backpropagation algorithm to adapt internal parameters, weights $\mathbf{w}$, in multiple (hidden) layers to optimally represent observed data. Based on an objective function or a loss function $L(\mathbf{w}, y, \mathbf{x})$, a gradient vector is computed, which indicates how the weights $\mathbf{w}$ should be adjusted [135]. The loss function and gradient vector can be computed with the whole dataset, e.g., with batch gradient descent (GD), using small batches of observations, e.g., with mini-batch GD, or using single observations, e.g., with stochastic gradient descent (SGD) [31]. The two latter methods, SGD and mini-batch GD, can be applied in online learning. Linear regression (LR), which can be seen as a NN with zero hidden layers, is also easily implemented in the online learning setting with SGD. SGD exists in many variants and extensions, which are widely used learning algorithms [35]. Table 2.2 shows a summary of different machine learning algorithms and their applicability in online learning.

Fig. 2.2 shows a neural network with three input variables x_i, one output variable $\tilde{y}_1$ and two hidden layers with seven hidden neurons $h_i^{(j)}$, where j denotes the hidden layer index and i denotes the hidden neuron index within layer j. The $\tilde{}$ denotes that $\tilde{y}$ is a prediction variable. For regression problems, the neurons in the hidden layers usually have a rectified linear unit (ReLU) activation function $y = \max(0, x)$. The neuron in the last layer has a linear activation function $y = x$, allowing the NN to predict both positive and negative continuous values [96]. According to the *universal approximation theorem*, deep NNs with at least one hidden layer can approximate any Borel measurable function, provided the

Table 2.2: Machine Learning Algorithms and Online Learning Applicability

Algorithm	**Description**	**Online Learning**
LR	Fits a linear model	Yes
Decision Trees	Builds a tree-like model	No
Random Forest	Ensemble of decision trees	No
Gradient Boosting	Ensemble with boosting	No
SVM	Uses support vectors	No
NN with Batch GD	Uses all the training data in each step	No
NN with SGD	Updates model incrementally	Yes
NN with Mini-Batch GD	Mini-batch updates	Yes

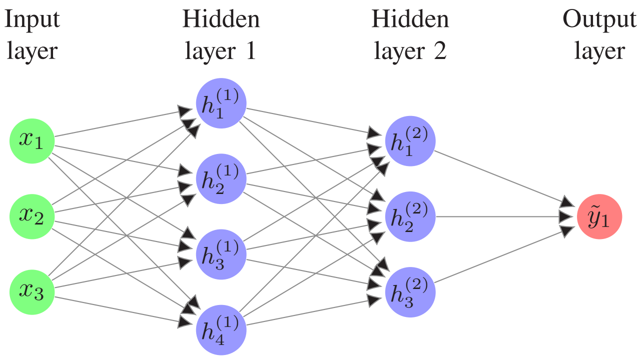
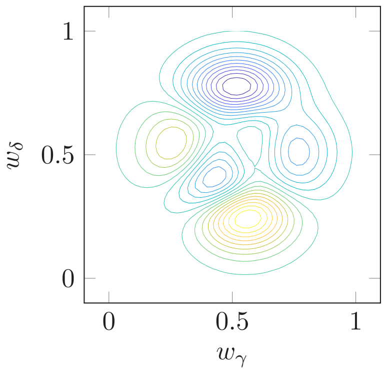


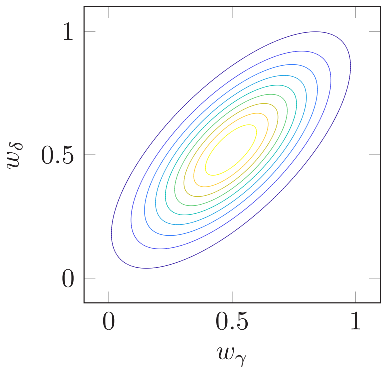
Figure 2.2: A NN with three inputs x_i, one output $\tilde{y}_1$ and seven hidden units $h_i^{(j)}$ in two hidden layers.

number of hidden neurons is high enough [55, 113]. This shows the capacity of NNs to learn complex and non-linear patterns.

A squared loss function L of a NN with ReLUs activation functions can be expressed as a polynomial function of the weights in the network, whose degree is the number of layers, and whose number of monomials is the number of paths from inputs to output [51]. The loss function is thus typically a non-convex function with multiple local minima. An exception is LR, where the loss function is the square of a linear combination of the observations, which is obliviously a convex function. Fig. 2.3a shows an exemplary loss surface for a deep NN.



(a) NN loss surface.



(b) LR loss surface.

Figure 2.3: Exemplary loss surfaces of a NN and a LR model.

The loss function exhibits several local minima and saddle points. Fig. 2.3b shows an exemplary loss surface for a LR, where a single, global minimum is present. The SGD algorithm typically requires more iterations in the learning of NNs than in LR. LR models can therefore easily be implemented in an online, single pass setting, whereas NNs may require more iterations on a sliding window of observations [126]. Nevertheless, it has been shown that the learning can be implemented in linear time, i.e., the learning time increases at most linearly with the number of observations [83]. In the online learning setting, where new data are constantly being observed, a strongly decreasing learning rate η is not sensible. In this dissertation, SGD with a momentum term and a constant learning rate is applied, so that the algorithm can react appropriately to new data [187]. SGD is applied in many probabilistic learning algorithms, in approximations of Bayesian learning algorithms, and in federated learning (FL), all of which are further introduced in Chapter 4.

2.4 Electric Vehicle Routing Process

The energy demand and driving range prediction algorithm is essential in the planning of long-distance trips, where the driving range is shorter than the distance to the destination and charging stops are necessary. The software requirements are specified using the following description. The principle of most range prediction algorithms is to first estimate the energy required for the given route. By comparing the required energy with the battery's estimated state of energy (SoE), the driving range and destination attainability can be determined. Fig. 2.4 shows the process of routing, range prediction and charge planning as a flowchart.

First, all relevant information, such as the road map and the live traffic feed is pre-processed. Routing algorithms are then used to calculate the fastest route from the starting point to the destination using corresponding graphs. Commercial navigation software suppliers do not publish the exact details of their proprietary methods. A standard routing algorithm is Dijkstra's algorithm [62]. Dijkstra's algorithm is typically quite slow, whereas the A-star algorithm is a relatively fast routing algorithm [104]. The A-star algorithm can be implemented with different heuristics that speed up the algorithm, such as highway hierarchies, where highway connections are preferred [59]. The route calculation can be extended to include energy efficiency (eco-routing). In this case, edge weights representing energy demand could be negative, since BEVs are able to regenerate energy when decelerating or driving downhill. The Bellman-Ford algorithm [28, 80] can handle negative edge weights and has been used in eco-routing [209, 47]. Based on the calculated route, the energy demand is predicted. If the energy demand is greater than the available battery energy, a new route with charging stops needs to be planned. The charge planning algorithm is presented in Chapter 6. The route and charging stop information, along with the route-based driving range, is then displayed in the vehicle's infotainment system. The routing, driving range prediction, and charge planning process is analyzed in terms of QoS metrics in Chapter 3, in terms of predictive performance in Chapter 5, and finally in terms of everyday usability of BEVs in Chapter 6.

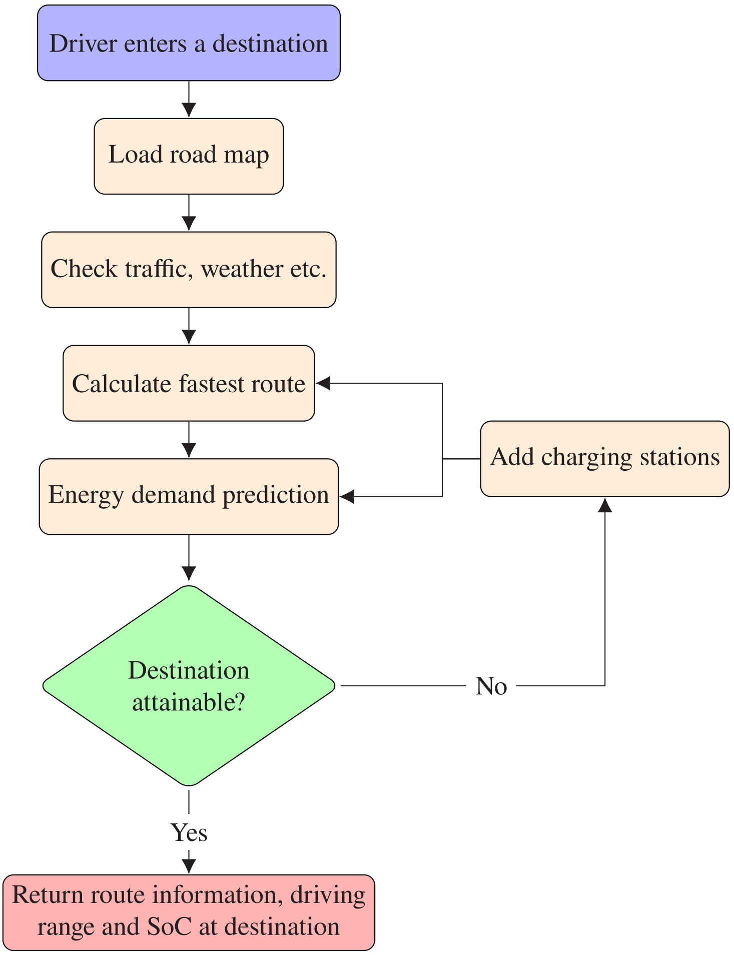

Driver enters a destination
Load road map
Check traffic, weather etc.
Calculate fastest route
Energy demand prediction
Add charging stations
Destination attainable?
No
Yes
Return route information, driving range and SoC at destination

Figure 2.4: A simplified flowchart for routing, range prediction, and charge planning.

3 System Architecture Evaluation

In this chapter, systems and software for energy demand prediction (EDP), routing, and charge planning are analyzed and their performance is evaluated in terms of quality of service (QoS). Thereby, the question of how distributed systems should be designed to support ML training and inference is addressed. This chapter analyzes system architectures corresponding to modern commercial vehicles, as well as to concepts proposed in related works. Furthermore, promising alternatives to existing system architecture concepts are proposed. The system analysis considers both learning and inference of machine learning (ML) based energy demand and driving range prediction models. The task of performing prediction or estimation with a learned model is understood under the term inference. Most research focuses on the learning procedure, whereas in terms of user experience, inference is just as important [52]. Thereby, the focus of this work is on model inference and the performance of the system as a whole. The approach models and simulates the software in a distributed computing setting and enables the evaluation of different system architectures. It is proposed to estimate resource requirements based on route length, algorithm time complexity and the number of instructions in the compiled code. In that way, the software can be tested for different use cases to ensure a sufficient QoS for routes of varying length.

The chapter is organized as follows. Section 3.1 gives an overview of related literature on distributed systems. In Section 3.2, the necessary hardware and software is described and modeled to enable system simulation. In Section 3.3, system architecture variants are presented and their performance is evaluated with simulations. Section 3.4 concludes the chapter and provides a summary of the findings. The methods and results in the chapter were previously published in the author's journal article "Evaluating System Architectures for Driving Range Estimation and Charge Planning for Electric Vehicles", which appeared in *Software: Practice and Experience* in 2021 [4].

3.1 Literature Survey

In connected vehicles, vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication is possible. Mobile internet (4G, 5G) and dedicated short range communication (DSRC) are two competing communication technologies used in vehicle-to-everything (V2X) settings, which can be seen in a survey conducted by Abboud et al. [10]. Different computing principles are possible in connected vehicles, such as cloud, edge and fog computing [239]. In the vehicles, resources such as computing performance and storage are limited. In turn, cloud resources are ample and a synergistic distribution in a system with vehicles and clouds is reasonable. Deploying software in such systems is a challenge, as the system architecture has a significant impact on their performance and latency. Distributed and cloud-based vehicle functions have been classified into four different models: *only cloud*, *fall-back method*, *duplicate function* and *elastic application*. This chapter focuses on the class *elastic application*, where the software consists of several modules distributed between the vehicle and cloud [160]. The challenge of the system architecture design is thus finding optimal placements of the software modules.

Brogi et al. give an overview of existing methodologies for optimally solving module placement problems in fog infrastructures [40]. The vehicle communication platform CloudThink was developed to securely enable software functionality divided between vehicle and cloud. Its system architecture consists of vehicle ECUs as well as data and gateway servers that communicate over wireless connections [224]. One of the elastic application use cases identified by Milani & Beidl includes predictive functions relying on predictive cloud information, and the EDP operates precisely in such a fashion [160]. Further examples for elastic applications are functions using computation offloading. In such applications, computational or data intensive tasks, such as image processing and ML, are offloaded to the cloud. This offloading can either be permanent or adaptive, but in both cases the module placement is intelligently performed to minimize parameters such as latency, energy and cost [21]. There are many examples in the literature where cloud or fog computing is used in vehicular technology. Siegel et al. give an overview of the state of the art of connected vehicles and their applications [199]. Lee et al. use ML to analyze driving behavior in the cloud using data from connected BEVs [137]. Wu et al. use cloud computing in electric vehicle charging control and dispatch optimization [227]. Ozatay et al. implemented a velocity profile optimization with dynamic programming where computationally intensive calculations were performed in the cloud [175]. Saini et al. propose a middleware for vehicular infotainment systems where computation tasks are carried out in the cloud and only relevant content is forwarded to the vehicle [189]. Yaseen et al. perform cloud-based video analytics using convolutional neural networks [231]. Siegel et al. studied the feasibility of different connected vehicle applications and even identified driving range prediction as an application that could be deployed in connected vehicles [198]. In recent years, fog and cloud based ML algorithms have received increased attention. Due to the diversity of ML algorithms and their applications, corresponding systems are analyzed individually. Tuli et al. analyzed a system for a fog-cloud based object detection with deep learning [213], as well as a deep learning based smart healthcare system [214]. Lin et al. proposed and analyzed a deep learning framework for smart manufacturing inspection systems based on fog computing [143].

For optimal module placement and system architecture design, performance evaluation is important. Ghosal et al. defined and proposed diverse QoS metrics for the evaluation of system architectures. These can be divided into *non-functional requirements*, *degree to accommodate changes*, *customer requirements* and *compatibility to legacy designs* [90]. In this chapter, the most important metrics regard the systems performance in terms of latency, network usage, reliability, availability, flexibility, scalability, expandability, security, energy efficiency, and cost [4]. For the user of the system, the latency or response time is a central criteria [169]. In the early stages of development, measurements of system performance are rarely possible, but simulations can be performed to approximately evaluate performance of different system architectures and module placements. However, simulating the performance of distributed systems is a challenging task. Cloud, edge and fog computing concepts introduce increased complexity with high diversity of devices and high numbers of possible system architecture variants. Traditional tools for the simulation of embedded systems, such as *chronSIM* [16] and *SymTA/S* [108], as well as tools for the simulation of wireless sensor networks, such as *OMNeT++* [216] and *TOSSIM* [140] are typically not applicable in fog and edge computing scenarios. An overview of simulation scenarios in fog and edge computing was given by Svorobej et al. In addition, their review includes a brief comparison of simulation tools [206]. *Cloudsim* is a widely used toolkit for the modeling and simulation of cloud computing environments [45]. In an analysis of cloud and fog simulation tools, three of six tools were extensions of *Cloudsim* [12]. The tool with the most options and flexibility, as well as the most used tool according to a citation count, is *iFogSim*, which enables the simulation of cloud, edge and fog computing settings to evaluate the impact of different module placements and resource management techniques on different QoS metrics [103]. Latency, energy usage, network usage, and cost of cloud execution are all measurable with simulations in *iFogSim* and monetary cost of network communication can be derived from network usage, if the network provider's conditions are known. *iFogSim* fulfills all requirements, allows for variable modeling of resource requirements, evaluates the correct metrics and is well-established in the scientific community. A more detailed overview of *iFogSim* can be seen in [103].

3.2 Modeling of Software and Hardware Systems

To investigate performance through simulations, the EDP software and the corresponding hardware must be modeled to construct the simulation framework. A simulation framework for a distributed system comprising a cloud infrastructure, a fleet of vehicles, and a mobile network has several key requirements. Firstly, the simulation should accurately represent the interactions between these components, capturing the dynamic nature of their connections and dependencies. It should also incorporate realistic communication models for the vehicles and mobile network to mimic real-world scenarios effectively. Scalability and resilience should be considered to evaluate the system's performance under varying conditions. In the following sections, the modeling of the software and hardware for the simulation framework is introduced.

3.2.1 Hardware and Connectivity

In hardware modeling, memory, storage and bandwidth properties are trivial to determine and model. Modeling the computational performance is not as trivial and several different approaches and metrics exist. One option is to measure processor capacity and load or the number of cores or virtual cores [105]. Alternatively, generic and traditional metrics such as floating point operations per second (FLOPS) and million instructions per second (MIPS) can be used [120]. Several variants and derivative metrics have been defined and an overview was given by Wang et al. [221]. *iFogSim* defines hardware units with the following indicators:

- processor performance [MIPS],
- RAM [e.g., GB],
- cost rate per MIPS used [e.g., $],
- busy and idle power [W].

Each hardware unit is modeled with these parameters. MIPS is not a perfect performance indicator, its main defect is that it is architecture-dependent. However, for a reduced instruction set computer (RISC), it is an acceptable performance indicator [128].

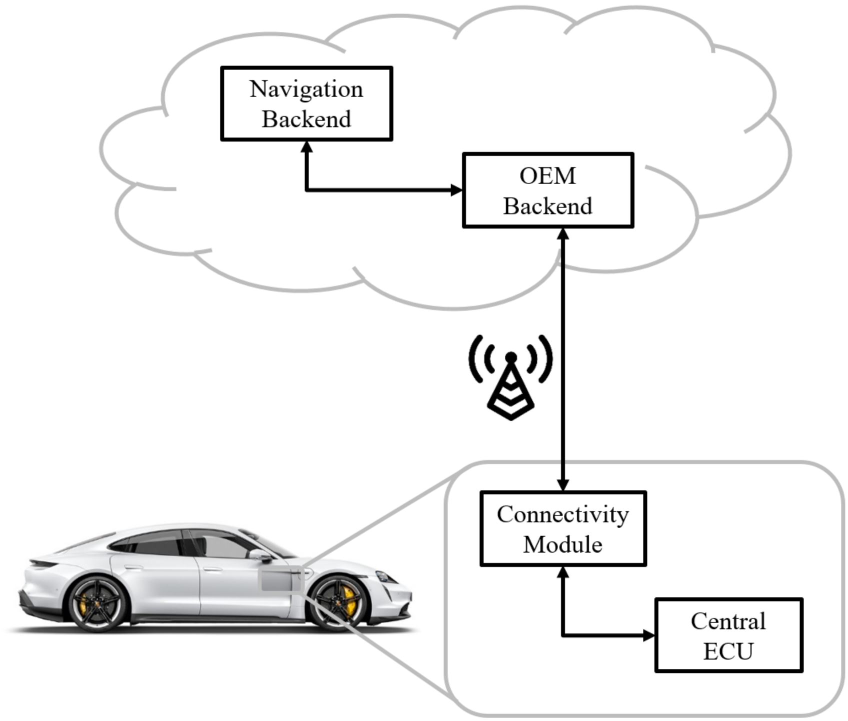


Figure 3.1: Cloud-Vehicle hardware topology.

For the EDP and charge planning, several dedicated hardware units are needed. Fig. 3.1 shows an overview of the hardware units and their topology. The vehicle's central ECU is the main processing unit, whereas a connectivity module establishes a connection with the backend and handles data transfer over wireless or mobile internet. In the cloud, an original equipment manufacturer (OEM) backend is used for all processing except the route calculation, which is performed in a 3rd party backend from the navigation provider. Table 3.1 shows these devices and their specified memory size, processor speed in MIPS, power usage, and instruction set of the processor. In contrast to a dedicated cloud server, an ECU needs to perform multiple tasks simultaneously, i.e., a single function does not generally have full access to the processor's performance and memory. Therefore, an available processor speed in the magnitude of 10^3 MIPS is assumed for the ECUs [128]. The backend units are significantly more powerful than the ECUs, therefore an estimated processor speed of 10^5 MIPS is assumed [45]. One difference between processors of a cloud and a vehicle ECU is the architecture and instruction set. Usually, RISC architecture chips are used in ECUs, whereas complex instruction set computer (CISC) architecture processors are more commonly used in servers and backend units. This means that different compilers are required, which can affect the number of instructions of a software module and is therefore considered in the module placement analysis [176]. The power usage of different hardware units is estimated based on commercial device information [181, 38]. The cost of using cloud instances is estimated to be 1 $ per instance per hour, based on commercial cloud services [45, 38].

Table 3.1: Specifications of the hardware units considered.

Device	OEM Backend	Navigation Backend	Connectivity Module	Central ECU
Location	Cloud	Cloud	Vehicle	Vehicle
RAM [GB]	200	400	0.2	0.5
CPU Speed [MIPS]	1×10^5	2×10^5	1×10^3	5×10^3
Active Power [W]	1×10^4	2×10^4	45	90
Idle Power [W]	8×10^3	1.6×10^4	20	80
Instruction Set	CISC	CISC	RISC	RISC

Wired and wireless connections between hardware units in *iFogSim* are defined by bandwidth and average network communication latency. The vehicle is connected to the cloud via 4G mobile internet with common upload and download transfer rates. The vehicle ECUs can be connected internally via controller area network (CAN), ethernet, or similarly [152]. It is assumed that the cloud units have a high-speed and low-latency wired connection. In Table 3.2, the estimated connection speeds and communication latencies are shown. It is assumed that up to 100 vehicles will simultaneously use the service.

Table 3.2: Data transfer rate in $[\mathrm{Mbit\,s^{-1}}]$ and communication latency in $[\mathrm{ms}]$ from (row) to (column).

Device	OEM Backend	Navigation Backend	Connectivity Module	Central ECU
OEM Backend	-	$100\,\mathrm{Mbit\,s^{-1}}$ $50\,\mathrm{ms}$	$15\,\mathrm{Mbit\,s^{-1}}$ $100\,\mathrm{ms}$	N/A
Navigation Backend	$100\,\mathrm{Mbit\,s^{-1}}$ $50\,\mathrm{ms}$	-	N/A	N/A
Connectivity Module	$10\,\mathrm{Mbit\,s^{-1}}$ $100\,\mathrm{ms}$	N/A	-	$10\,\mathrm{Mbit\,s^{-1}}$ $1\,\mathrm{ms}$
Central ECU	N/A	N/A	$10\,\mathrm{Mbit\,s^{-1}}$ $1\,\mathrm{ms}$	-

3.2.2 Software Modules and Resource Requirements

In this section, the modules of the proposed EDP and charge planning software are introduced. For the realization of the routing, EDP, and charge planning process shown in Fig. 2.4, different software modules are required. A generic software that fulfills the requirements discussed in Section 2.4 may consist of a user-interface, route calculation, EDP, charging point (CP) database, and a vehicle configuration module. Additionally, a controller module is required to coordinate the different computing modules and the user-interface. Information is transferred between the modules through application programming interfaces (APIs). Fig. 3.2 shows a software block diagram, where the modules and their connections are visualized.

The software modules are modeled by the number of processor instructions required for the computational tasks and by the size of the response or data generated by the module. The number of instructions is measured in million instructions (MI) and the response size is measured in bytes. The instructions are dependent on the processor architecture, such as RISC or CISC, and the number of instructions for a software module is therefore dependent on the processor. Simple instructions running in a single clock cycle is a typical characteristic of RISC architecture, whereas many instructions in CISC architectures run over several clock cycles

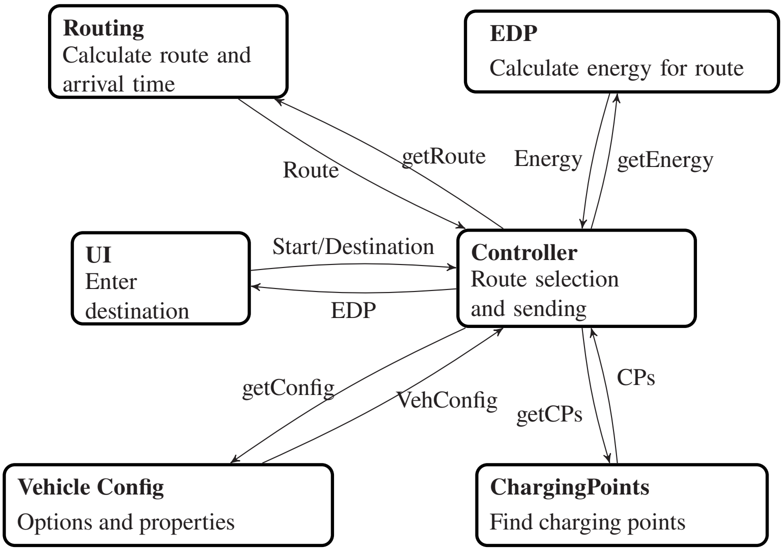


Figure 3.2: Software block diagram for routing, EDP, and charge planning.

[121]. The time required for a processor instruction is dependent on the type of instruction, i.e., performing some instruction may take a shorter or longer time than another instruction. To determine the number of instructions, one option is to measure execution time on a certain processor with a known speed in MIPS and then determine the number of instructions in MI. Here, the suggestion is, however, to count the instructions directly in the compiled assembly code. In the following, each step of the routing, EDP, and charge planning is described. For each module, the requirements are estimated in terms of number of instructions and response size. As the software is dependent on route length, the algorithm's time complexity is used to describe the number of instructions as a function of route length.

3.2.2.1 User Interface

In this module, the driver chooses the destination in the user interface (UI) and the destination is sent to the route calculation module. The drivers input is a string with the destination address or name. Before the communication is initiated, the vehicle must be authenticated through a transport layer security (TLS) handshake. The vehicle and backend exchange messages to establish a secure connection. The duration of the TLS handshake comprises time for crypto-processing, network latency and other delays due to message parsing [180].

3.2.2.2 Routing

After receiving starting position, destination, and routing options, this module calculates the fastest possible route, with regard to actual traffic information. The performance of a routing algorithm is strongly dependent on the graph that represents the road network. The complexity is dependent on the number of edges $|E|$ and vertices $|V|$ in the graph, as well as its sparsity and the branching factor. For the A-star algorithm, the time complexity is between $\mathcal{O}\left(|V|\right)$ and $\mathcal{O}(|E| + |V| \log |V|)$, depending on heuristic and graph type [195]. In this work, the time complexity is estimated to be $\mathcal{O}(|E| + |V|)$. Thereby, the number of instructions needed for a route calculation in a sub-graph with $|E|$ edges and $|V|$ vertices is linearly dependent on the sum $|E| + |V|$. Based on real graphs for USA [63] and Germany [98], an assumption is made that for a unit length route and a given heuristic, a sub-graph with $V(l) = 1,600$ vertices and $E(l) = 4,200$ edges is required. In this work, the number of instructions for one iteration of the A-star algorithm is $345\,\mathrm{I}$. In addition, a baseline computation of $100\,\mathrm{MI}$ is assumed. The total number of instructions is thus

$$\begin{aligned}\mathrm{CPU}_{\mathrm{req}}^{\mathrm{Routing}}(l) &= 100\mathrm{MI} + 345\ \mathrm{I}\ \left(E(l) + V(l)\right) \\ &= 100\mathrm{MI} + 345\frac{\mathrm{I}}{\mathrm{iter}} \cdot (4200 + 1600)\frac{\mathrm{iter}}{\mathrm{km}} \cdot l\ \mathrm{km} \\ &= (100 + 2 \cdot l)\ \ \mathrm{MI}\,, \end{aligned} \tag{3.1}$$

where l is the route length in km. For a calculated route, its attributes for each route segment are sent to the EDP. On average, the length of a segment is approximately $200\mathrm{m}$, i.e., there are $N_S = 5$ segments per $l = 1\mathrm{km}$ [110]. For each segment, 20 attributes are stored as doubles ($8\,\mathrm{B}$). Additionally, a constant $2000\,\mathrm{B}$ is used to account for Hypertext Transfer Protocol (HTTP) header and other data and overhead not dependent on route length. The total response size for a route is therefore

$$\begin{aligned} S^{\text{Routing}}(l) &= 2000\mathrm{B} + 20 \cdot N_S(l) \cdot 8\mathrm{B} \\ &= 2000\mathrm{B} + 20 \cdot 5l \cdot 8\mathrm{B} \\ &= (2000 + 800 \cdot l)\ \mathrm{B}\,. \end{aligned} \tag{3.2}$$

3.2.2.3 Energy Demand Prediction

For the calculated route, the route specific EDP is computed with the ML algorithm using attributes for the planned route as input parameters. In this regression problem, a NN is used. The learning of the NN is a non-event based process which runs in the background and is not considered in the control loop, but rather analyzed separately in Section 3.3.3. The time complexity of the inference is $\mathcal{O}(|N_S|)$, where N_S is the number of segments in the route. In this work, inference with NNs can be efficiently performed with $200\,\mathrm{I}$.

$$\begin{aligned} \text{CPU}^{\text{EDP}}_{\text{req}}(l) &= 100\mathrm{MI} + 200\frac{\mathrm{I}}{\mathrm{seg}} \cdot N_S(l) \\ &= 100\mathrm{MI} + 200\frac{\mathrm{I}}{\mathrm{seg}} \cdot 5\frac{\mathrm{seg}}{\mathrm{km}} \cdot l\ \mathrm{km} \\ &= (100 + 10^{-3} \cdot l)\ \mathrm{MI}\,. \end{aligned} \tag{3.3}$$

By comparing the required energy with the battery's state of energy (SoE), the destination attainability can be determined. If the current SoE is sufficient to reach the destination, route and energy information can be sent back to the driver. If the destination is not attainable, a charging point search and a charge planning is triggered. The response includes the energy consumption for each route segment, represented by $2 \cdot N_S$ variables. Thereby, the response size is

$$\begin{aligned} S^{\mathrm{EDP}}(l) &= 2000\mathrm{B} + 2 \cdot N_S(l) \cdot 8\mathrm{B} \\ &= 2000\mathrm{B} + 2 \cdot l \cdot 5 \cdot 8\mathrm{B} \\ &= (2000 + 80 \cdot l)\ \mathrm{B}\,, \end{aligned} \tag{3.4}$$

where the $2000\,\mathrm{B}$ represent HTTP header and other overhead.

3.2.2.4 Charging Point Search and Planner

When triggered, the search finds applicable charging points (CPs) in a geographical corridor along the route. It is assumed that there is always a free and functioning plug at each CP. These CPs are then sent to the routing module. With an analysis of the charging infrastructure in western Europe [95], it was established that the mean distance between fast chargers ($P \geq 100\mathrm{kW}$) along major routes is approximately $50\,\mathrm{km}$, i.e., the number of CPs along a route is

$$n_{\mathrm{CPs}}(l) = 0.02 \cdot l\,. \tag{3.5}$$

The complexity of the CP search is dependent on the number of possible CPs, i.e., it is dependent on route length. When implemented with linear search, the time complexity is $\mathcal{O}(l)$ and the resource requirements are

$$\mathrm{CPU}^{\mathrm{CPs}}_{\mathrm{req}}(l) = 100\mathrm{MI} + 0.02\tfrac{\mathrm{CPs}}{\mathrm{km}} \cdot l\mathrm{km} \cdot 500\mathrm{I} \tag{3.6}$$

$$= (100 + 10^{-5} \cdot l)\ \mathrm{MI}\,, \tag{3.7}$$

where it is assumed that one search iteration can be performed with $500\mathrm{I}$. The CPs' GPS coordinates are given by two variables and the response size is therefore

$$\begin{aligned} S^{\mathrm{CPs}}(l) &= 2000\mathrm{B} + 2\tfrac{\mathrm{vars}}{\mathrm{CP}} \cdot 0.02\tfrac{\mathrm{CPs}}{\mathrm{km}} \cdot l\mathrm{km} \cdot 8\mathrm{B} \\ &= (2000 + 0.32 \cdot l)\ \mathrm{B}\,, \end{aligned} \tag{3.8}$$

where the $2000\,\mathrm{B}$ represents HTTP header and other overhead.

Through all reasonable combinations of these CPs, the fastest route is calculated. Depending on battery SoE and maximum driving range, the number of reasonable CP combinations for a route of length l can be up to

$$n_{\text{CP-Comb.}}(l) = 2^{n_{\text{CPs}}(l)} = 2^{0.02 \cdot l}\,. \tag{3.9}$$

Subsequently, the energy for all route segments between the CPs is estimated and charging times at each charger are determined. Thereby, the total travel time is calculated. The charge planner sends multiple requests to the routing and EDP, dependent on the number of possible and reasonable routes. Ideally, the routes and waypoints overlap to a certain extent, which means that a previously calculated route and energy consumption can be partially used for another route. In the worst case, all possible routes are different, which means that each route and its energy consumption is calculated individually. The number of possible routes is given by (3.9). The resulting worst-case resource requirements are

$$\begin{aligned} \text{CPU}_{\text{req}}^{\text{Routing, Mult.}}(l) &= n_{\text{CP-Comb.}}(l) \cdot \text{CPU}_{\text{req}}^{\text{Routing}}(l) \\ &= \left(100 + 2^{0.02 \cdot l + 1} \cdot l\right)\ \text{MI} \end{aligned} \tag{3.10}$$

and

$$\begin{aligned} \text{CPU}_{\text{req}}^{\text{EC, Mult.}}(l) &= n_{\text{CP-Comb.}}(l) \cdot \text{CPU}_{\text{req}}^{\text{EC}}(l) \\ &= (100 + 2^{0.02 \cdot l} \cdot 10^{-3} \cdot l)\ \text{MI}\,. \end{aligned} \tag{3.11}$$

3.2.2.5 Route Selection and Sending

The final step is to choose the fastest route and charge plan, and to deliver this information to the UI. Finding the fastest route means searching in the list of calculated routes for the shortest travel time and can be achieved through linear search. The time complexity of linear search is $\mathcal{O}(n)$ where n is the number of routes. Each comparison in the linear search requires approximately $10\,\mathrm{I}$, so the resource requirements are

$$\begin{aligned}\mathrm{CPU}^{\mathrm{Rate}}_{\mathrm{req}}(l) &= 100\mathrm{MI} + 10 \cdot 10^{-6} \cdot n_{\text{CP-Comb.}}(l)\ \mathrm{MI} \\ &= 100\mathrm{MI} + 10 \cdot 10^{-6} \cdot 2^{0.02 \cdot l}\ \mathrm{MI} \\ &= (100 + 10^{-5} \cdot 2^{0.02 \cdot l})\ \mathrm{MI} \end{aligned} \tag{3.12}$$

and the response size is assumed to be $1\,\mathrm{kB}$. Sending the final response, i.e., the route, EDP and charge plan, to the UI requires $50\,\mathrm{MI}$. The route is represented by one $8\,\mathrm{B}$ variable for each route segment, so the response size is

$$\begin{aligned}S^{\mathrm{Resp.}}(l) &= 2000\mathrm{B} + N_S(l) \cdot 8\mathrm{B} \\ &= 2000\mathrm{B} + 5 \cdot l \cdot 8\mathrm{B} \\ &= (2000 + 40 \cdot l)\ \mathrm{B}\,, \end{aligned} \tag{3.13}$$

where the $2000\,\mathrm{B}$ represent HTTP header and the display values for EDP and charge plan, which are independent on route length.

Table 3.3: Resource requirements of EDP modules for a single trip.

Component	**Complexity**	**Instructions** $[\mathrm{MI}]$	**Response size** $[\mathrm{kB}]$
enterDest()	$\mathcal{O}(1)$	11	1.4
getConfig()	$\mathcal{O}(1)$	20	1
getRoute()	$\mathcal{O}(l)$	$100 + 2 \cdot l$	$2 + 0.8 \cdot l$
getEnergy()	$\mathcal{O}(l)$	$100 + 10^{-3} \cdot l$	$1 + 8 \cdot l \cdot 10^{-2}$
checkAttainability()	$\mathcal{O}(1)$	100	1
getCPs()	$\mathcal{O}(l)$	$100 + l \cdot 10^{-5}$	$1 + 0.32 \cdot 10^{-3} \cdot l$
rateRoutes()	$\mathcal{O}(l)$	$100 + 10^{-5} \cdot 2^{0.02 \cdot l}$	1
writeResults()	$\mathcal{O}(1)$	50	$2 + 4 \cdot 10^{-2} \cdot l$

In Table 3.3, the resource requirements of each module are summarized. The complete process of the EDP and charge planning is shown in Fig. 3.3. The figure shows a sequence diagram describing the process beginning from when the driver enters a destination ending with the display of the route, driving range and charge plan.

In Fig. 3.4, a further visualization of the resource requirements of the software modules is shown for different route lengths. Fig. 3.4a shows the response size and Fig. 3.4b shows the number of instructions of the route-length-dependent software modules as a function of route length. As can be seen in Fig. 3.4, the routing algorithm and the EDP output the most data and also require the most instructions. This gives a hint that the data output from these modules should not be transmitted over a mobile internet connection and that placing these in the cloud would result in lower end-to-end latencies.

3.3 Performance Evaluation

To evaluate the performance of the systems described in Section 3.2, simulations are performed with *iFogSim*. The control loop shown in Fig. 3.3 is analyzed with different route lengths and module placements. For the control loop, following performance indicators are measured:

- latency,
- network usage,
- cost of cloud execution,
- energy usage in cloud and vehicle.

An optimal module placement minimizes all performance indicators for all route lengths. In this work, the number of possible and reasonable combinations is low. Therefore, all of these combinations can be analyzed to determine the optimal module placement. For each module placement variation, the performance indicators are calculated. Fig. 3.5 shows four module placement variants between the vehicle and cloud. The placement in Fig. 3.5a is the classical vehicle-based

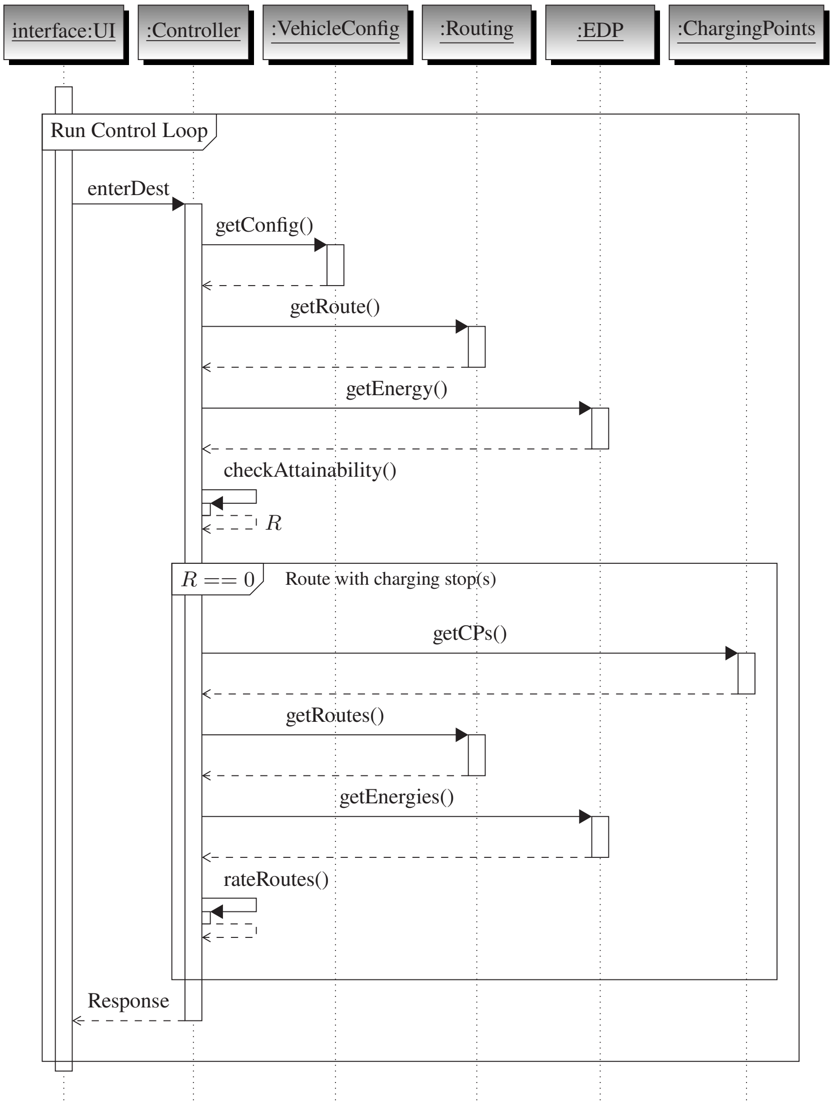
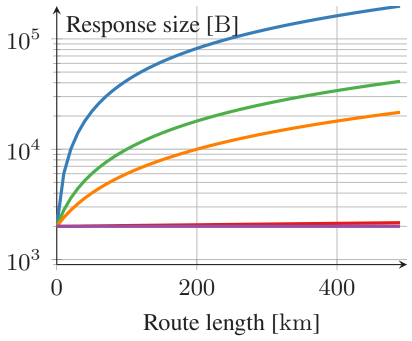
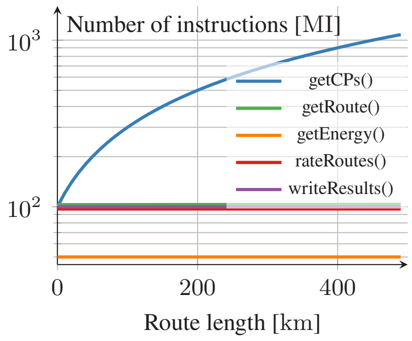


Figure 3.3: Sequence diagram for routing, EDP, and optional charge planning.



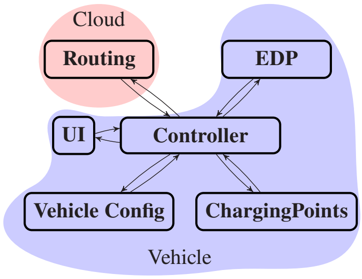
(a) Amount of data generated by the software modules depending on route length.



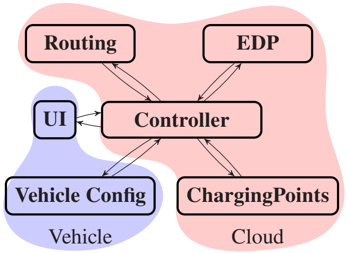
(b) Number of instructions of the software modules depending on route length.

Figure 3.4: Resource requirements of route planning, EDP and charge planning.

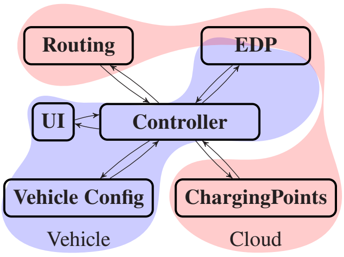
placement used as a baseline for the evaluation. The vehicle-based placement is commonly applied in current day BEVs. Furthermore, it was suggested by Yi et al. [234], Scheubner et al. [5], and Ferreira et al. [77]. In that placement, all software modules are placed in the vehicle except for the routing algorithm, which is based in the cloud. The routing algorithm relies on real time information on traffic and road conditions and can be seen as an external service. In the cloud-based placement shown in Fig. 3.5b, all but the UI and vehicle configuration module are placed in the cloud. In terms of inference, this placement corresponds to the systems suggested by Fukushima et al. [84], Grubwinkler et al. [101, 100], and Lee et al. [137]. The placement shown in Fig. 3.5c is a hybrid between the cloud- and vehicle-based placements, where the routing algorithm and the CP search are implemented in the cloud and the EDP module is placed dynamically both in the cloud and in the vehicle. The second hybrid placement shown in Fig. 3.5d is the same as hybrid 1, except the EDP module is only implemented in the cloud. This placement corresponds to the systems suggested by Thibault et al. [209] and Jayakumar et al. [122]. In the following, the simulations and their results are discussed. In Section 3.3.1, the setup of the experiments in *iFogSim* is presented. In Section 3.3.2, inference with the EDP algorithm is analyzed. In Section 3.3.3, the learning of the eEDP is examined.



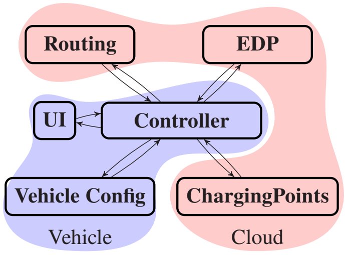
(a) Vehicle-based placement



(b) Cloud-based placement



(c) Hybrid 1



(d) Hybrid 2

Figure 3.5: Software component diagram showing the four module placement variants.

3.3.1 Experiment Setup

To analyze the EDP and charge planning software, *iFogSim* is configured for the simulation of the systems presented in Section 3.2. In the following, the classes of *iFogSim* applied in the experiments are described. A *FogDevice* is created for each of the hardware units in Table 3.1 with the given specifications. According to the topology shown in Fig. 3.1, a direct hierarchy of *FogDevices* is defined. The parent-child pair communication in the hierarchy is configured according to the specification in Table 3.2. In addition, appropriate *Sensors* are configured in the vehicle to measure velocity, energy consumption etc. Finally, a *Sensor* and an *Actuator* are configured to represent the UI. The latency between a *Sensor*/*Actuator* and the vehicle's central ECU is estimated to be 5 ms [217]. An *AppModule* is created for each of the modules in the EDP and charge planning software shown in Fig. 3.2. For each of the edges between the software blocks in Fig. 3.2, an *AppEdge* is created. Each edge carries a *Tuple* that defines the function of the edge. The processing requirements of the edges' tuples are specified according to Table 3.3, where the variable l is used to specify the length of the route. The edges are event-based and their function is triggered by an incoming tuple from a source software module. To monitor and measure the end-to-end latency of the control loop shown in Fig. 3.3, an *AppLoop* is specified according to the sequence diagram. With the class *ModulePlacement*, the mapping of the *AppModules* on the *FogDevices* is defined. Thereby, the module placement variants shown in Fig. 3.5 can be configured for the simulations. The results of the simulations include the end-to-end latency of the control loop, as well as the network usage. Furthermore, each *FogDevice* measures the energy used during the simulation and the cost of using cloud instances is calculated. In the following section, an analysis of these results is presented.

3.3.2 Inference

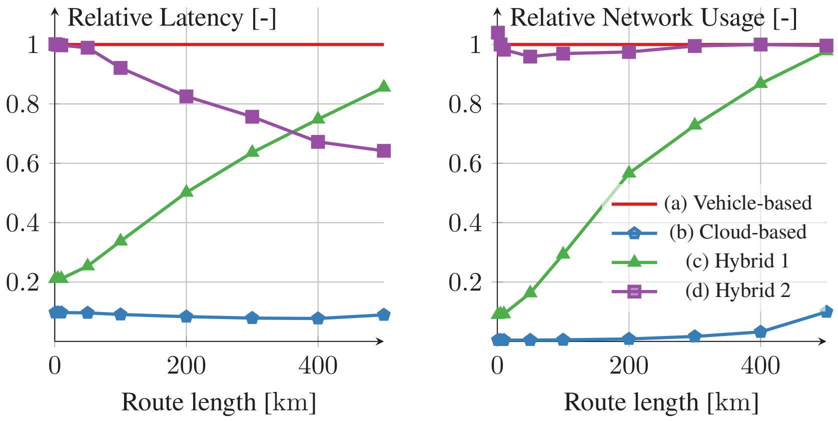
For the EDP and charge planning software, the four different module placement variants shown in Fig. 3.5 are simulated, each of which for different route lengths. The route lengths simulated are $10, 50, 100, 200, 300, 400, 500\,\mathrm{km}$. The mean results based on the simulations are shown in Table 3.4. The table shows the performance indicators control loop latency, cost of cloud execution, energy usage in the cloud, energy usage in the vehicle and total network usage, in proportion to the baseline vehicle-based placement (a). As the simulation includes uncertainty, the results are shown in proportion to the baseline, not as the absolute values. In that way, the absolute system specifications are of lower importance compared to the ratio of the specifications of different system architectures. For all performance indicators, the cloud-based placement (b) is the best. The control loop latency, cost of execution in cloud and total network usage are significantly lower in the cloud-based placement than in the other three placements. The energy usage, both in the cloud and in the vehicle, is similar for all four module placements.

The first hybrid placement (c) shows significantly better results than the vehicle-based placement (a), but still fails to attain the performance of the cloud based placement, which achieves more than tenfold improvement in latency, cost and network usage.

Table 3.4: Performance indicators of module placements in proportion to the baseline, vehicle-based, placement.

Placement	(a) Vehicle-based	(b) Cloud-based	(c) Hybrid 1	(d) Hybrid 2
Latency [-]	1	0.09	0.56	0.80
Cost [-]	1	0.03	0.68	1.07
Energy Cloud [-]	1	0.92	0.98	1.00
Energy ECU [-]	1	0.99	1.00	1.00
Network usage [-]	1	0.03	0.60	0.98

Fig. 3.6 shows the control loop latency and network usage of different module placements in proportion to the vehicle-based placement (a) for different route lengths. For the driver, these performance indicators are most important in terms of user experience. Fig. 3.6a shows the latency of the control loop and Fig. 3.6b shows the network usage. For both latency and network usage, the cloud-based placement (b) is consistently better than the vehicle-based placement (a). The improvement in latency is approximately 10-fold for all route lengths. The network usage of the cloud-based placement (b) is approximately 10 times lower than that of the vehicle-based placement (a) for a route length of $500\,\mathrm{km}$ and even lower for shorter route lengths. The Hybrid 1 placement's (c) performance is similar to the cloud-based placement (b) for shorter routes, but with increasing route length the performance worsens and becomes similar to that of the vehicle-based placement (a). The second hybrid placement's (d) performance is similar to that of the vehicle-based placement (a), apart from an improvement in relative latency with increasing route length. The cloud-based placement (b) achieves clearly the highest over-all performance.



(a) Latency of the control loop in different module placements for different route lengths in proportion to the latency of the vehicle-based placement (a).

(b) Network usage of the module placements for different route lengths in proportion to the network usage of the vehicle-based placement.

Figure 3.6: Relative latency and network usage of EDP of a route with charging stops.

If the causes for the differences are analyzed, the most important factor is the time used down- or uploading data needed for the EDP and charge planning. In the cloud-based placement (b), the EDP uses the route, road and traffic information directly within the cloud and only transmits the final display values to the vehicle, which are considerably smaller in size. Furthermore, computation in the cloud is faster, which also improves the control loop's latency. Of the four module placement variants analyzed, cloud-based inference is clearly superior. The proposed system architecture can enable energy demand and driving range prediction concepts, such as those of Fukushima et al. [84], Grubwinkler et al. [101, 100], and Lee et al. [136, 137] to perform efficiently.

3.3.3 Learning

In the previous section, it was observed that inference can be done efficiently when the software modules are distributed intelligently between vehicle and cloud. However, the learning of models has yet to be considered. To ensure sufficient performance of the software, the correct placement of the learning module must be decided. This section compares the latency and network usage of two different strategies: vehicle-based (on-device) learning and cloud-based learning, utilizing online learning algorithms [36]. In related works on cloud-based energy demand and driving range prediction, cloud-based learning has to date been the preferred choice [84, 136, 137].

The training data in the EDP is streaming data that arrives sequentially with a frequency of up to 10 Hz. As discussed in Section 2.2, some of the input parameters for the EDP algorithms are traffic and road topology information from a TRDB. This information is also necessary in the learning of the model. However, only a description of the current situation is needed, which can be observed in the vehicle directly. For example, the temperature can be measured with the appropriate sensor and the traffic can be observed with on-board cameras and other sensors [5].

Consider the number of iterations $\mathcal{I}$ needed for an algorithm, such as SGD, to converge to an acceptable level. The time needed for each iteration is $\mathcal{T}_{\mathcal{I}}$ and therefore the total time for the learning of an prediction model in a non-distributed setting is

$$T = \mathcal{I} \cdot \mathcal{T}_{\mathcal{I}} \,. \tag{3.14}$$

In a distributed setting, the communication between vehicle and backend is also important. If c is the latency of the communication, the total time needed for a distributed learning of the model is

$$T = \mathcal{I} \cdot (c + \mathcal{T}_{\mathcal{I}}) \,, \tag{3.15}$$

if communication in each iteration is assumed. Since algorithms, such as SGD, require many fast iterations, even fast communications result in poor performance, as $c \gg \mathcal{T}_{\mathcal{I}}$ [132]. The learning algorithm can be deployed in a vehicle ECU as well as in the cloud. The communication latency c is estimated based on the data size of the training data for cloud-based learning, compared with the model size for vehicle-based learning. Fig. 3.7 shows the estimation of the communication cost in terms of data size. In the figure, the size of the 10 Hz training data stream is shown dependent on driving time in minutes. Additionally, the estimated model size of a LR model and a NN with 100 hidden units in 2 hidden layers is visualized. The

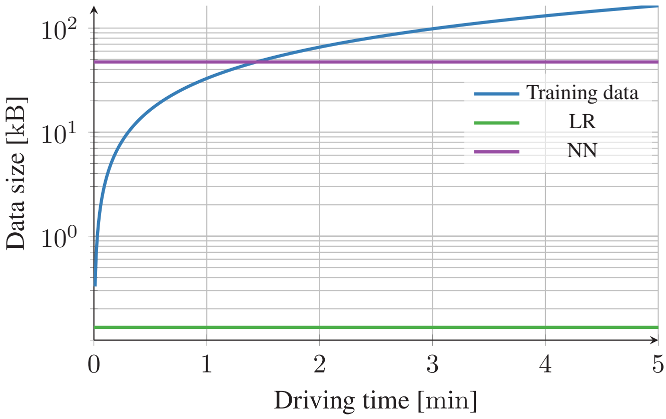


Figure 3.7: Size of the 10 Hz training data stream and estimated model size of a LR model and a NN in kB, dependent on driving time in minutes.

preferred placement of the learning module is dependent on the required frequency of updates to the EDP in terms of driving time. The threshold values for the update intervals can be observed as the points of intersection of the graphs in Fig. 3.7. For the cloud-based learning setting, the communication cost is higher than in the vehicle-based setting if the time interval between updates is greater than 0.24 s for the LR, or 86 s for the NN. To minimize total network usage and communication cost c, the update frequency should be as low as possible and as high as necessary for the optimal user experience. Choosing update intervals greater than 0.24 s for a LR or 86 s for a NN and placing the learning algorithm in the vehicle reduces the communication cost, as the model size is then effectively smaller than the size of the training data. In terms of communication and computation cost, vehicle-based learning can therefore be significantly better than cloud-based learning. In this respect, the vehicle-based ML concepts for energy demand and driving range prediction by Scheubner et al. [5] and Ferreira et al. [77] can therefore be implemented with the proposed system architecture in an efficient way. Related works such as by Fukushima et al. [84] and Lee et al. [136, 137], which have favored cloud-based learning, could benefit from placing the learning modules directly in the vehicles. Furthermore, by keeping the training data locally in the vehicle, the privacy of the users remains protected.

3.4 Summary and Conclusions

In this chapter, the performance of an electric vehicle routing software was evaluated in different system architectures. The evaluation enables the comparison of different module placements regarding latency, network usage, energy usage, and cost. By modeling the software and the hardware, simulations with *iFogSim* were performed. The results show that for the inference process, a cloud-based module placement is superior to other investigated placements. The cloud-based module placement is significantly better than the current day baseline, the vehicle-based placement. It is estimated that the end-to-end latency from the input of the destination to the display of the route and driving range can be improved by a factor of 10, which improves the user experience significantly. Furthermore, network usage can be reduced at least by a factor of 10. Additionally, different settings for the learning process of the model were analyzed and a vehicle-based learning setting was found to be a more feasible choice than cloud-based learning, which was favored in related works. The result obtained with the analysis can be used in early stages of the development to compare possible solutions and identify which are promising and which are not practicable.

The investigation of the criteria reliability, availability, flexibility, scalability, expandability, and security falls out of the scope of the simulations. From a subjective point of view, the proposed cloud-based module placement offers more flexibility, scalability and expandability than a traditional vehicle-based placement, as software updates and changes are simpler in the cloud than in the vehicle. As with all connected vehicle functions, the proposed system is dependent on mobile connectivity, which may have negative impact on availability and thus, reliability. With the extension of mobile networks and the deployment of 5G connectivity, availability and reliability continues to improve.

4 Probabilistic Predictions with Federated Learning

In Chapter 3, two problems were identified: transferring large amounts of data between the vehicle and a backend may lead to high communication costs and the privacy of the users may be compromised [219]. To counter these problems, machine learning (ML) is performed on-device, so that the data are kept localized on the device and are not uploaded to a central server. The most prominent on-device ML methods are distributed learning [245, 119], gossip learning [173], and federated learning (FL), e.g., federated averaging (FedAvg) [156].

This chapter focuses on the application of FL to generate a probabilistic model. Inspired by related work on probabilistic predictions with NNs, the learning of a probabilistic model through FL is proposed. The proposed algorithm, FedAvg-Gaussian (FedAG), introduces weight uncertainty in the aggregation step of the algorithm. In that way, the end devices can calculate probabilistic predictions but only have to learn conventional, deterministic models. To accentuate the capability of FedAG, the algorithm is validated using numerous open-source datasets, before the algorithm is applied to the EDP in Chapter 5. The open-source datasets originate from different domains, which demonstrates the generalizability of FedAG. This chapter is organized as follows: In Section 4.1, probabilistic predictions with ML are discussed and an overview of related work is given. In Section 4.2, FedAG is presented. In Section 4.3, a summary of proper scoring rules is presented, which are necessary for the evaluation of probabilistic predictions. In Section 4.4, open-source datasets are used to evaluate the performance of the method and the results are compared to benchmarks from related literature. In Section 4.5, the computational complexity of the presented algorithms is discussed. Finally, Section 4.6

gives concluding remarks and a summary. The results in this chapter were previously presented in the author's journal article "Probabilistic Predictions with Federated Learning", which appeared in *Entropy* in 2020 [1], as well as in the poster presentation "Federated Learning with Predictive Uncertainty" [7] and are directly reproduced in this chapter.

4.1 Literature Survey

A probabilistic prediction (or stochastic prediction) is when the prediction takes the form of a probability distribution, instead of a scalar value [93]. The application of ML in this topic is of significant relevance. Probabilistic predictions are commonly used in geology [138], electricity markets [41], urban water consumption [54], wind power [230], driver behavior [115], and vehicle dynamics [85]. Two prominent probabilistic prediction methods are Bayesian deep learning (BDL) and ensemble methods, on both of which a summary was given by Ashuka et al. [23]. In BDL, the model parameters, e.g., weights $\mathbf{w}$, are random variables represented by probability distributions $p(\mathbf{w})$. With a dataset $\mathcal{D}$, consisting of features $\mathbf{x}$ and target variable y, the posterior distribution for the model weights can be derived using the Bayes' rule, which states that the posterior distribution is proportional to a prior probability $p(\mathbf{w}|\theta)$ multiplied with likelihood $p(\mathcal{D}|\mathbf{w},\beta)$

$$p\left(\mathbf{w}|\mathcal{D},\theta,\beta\right) \propto p(\mathbf{w}|\theta)p(\mathcal{D}|\mathbf{w},\beta)\ , \tag{4.1}$$

where θ is a precision parameter for the prior distributions on weights $\mathbf{w}$ and β is a noise precision parameter. For simplicity, the weight posterior distribution is written as $p(\mathbf{w}|\mathcal{D})$. To make predictions for new, unseen data, the predictive distribution is obtained with

$$p(y|\mathbf{x},\mathcal{D}) = \int p(y|\mathbf{x},\mathcal{D},\mathbf{w})p(\mathbf{w}|\mathcal{D})\mathrm{d}\mathbf{w}\ . \tag{4.2}$$

The exact computation of (4.1) and (4.2) is usually intractable due to the non-linearity of NNs [31]. The integration over the posterior is commonly approximated with Monte Carlo (MC) methods, such as Markov chain Monte Carlo (MCMC) or Hamiltonian Monte Carlo (HMC) [166]. Alternative approximation methods are extended variational inference [60] and cubature rules based on the unscented transformation [222].

A recent survey on BDL was given by Wang & Yeung [220]. Traditional Bayesian neural networks (BNNs) do not scale well to large models and the posterior $p(\mathbf{w}|\mathcal{D})$ is usually difficult to calculate and sample from, but various approximation approaches have succeeded in creating probabilistic NNs. In variational inference (VI), the posterior $p(\mathbf{w}|\mathcal{D})$ is approximated with a well-defined distribution $Q_p(\mathbf{w}|\mathcal{D})$ and the variational free energy $\mathcal{F}$ is minimized to minimize divergence between $p(\mathbf{w}|\mathcal{D})$ and $Q_p(\mathbf{w}|\mathcal{D})$ [97]. In Bayes by Backprop, the variational free energy is not minimized naively but approximately using gradient descent [33]. In probabilistic backpropagation (PBP), the posterior is determined with a calculation of a forward propagation of probabilities followed by a backwards calculation of gradients [111]. Gal & Ghahramani used dropout to achieve a mathematical equivalent of a Bayesian approximation without probabilistic weights [86]. Maddox et al. proposed stochastic weight averaging Gaussian (SWAG), where an approximate posterior distribution over NN weights is determined by observing the SGD trajectory during the learning process [153]. Farquhar et al. showed that a deep NN with mean-field weight distributions, e.g., Gaussian, can approximate any posterior distribution over predictive functions [75].

An established alternative to BDL is the use of ensembles to generate a probabilistic prediction. Thereby, multiple scalar predictions are combined to infer a probabilistic prediction. The predictions are either calculated with several different models or with a single model with varying initial conditions or input data. In a statistical post-processing of the ensemble predictions, a single probability density is derived [39]. A simple method is fitting a probability distribution to the predictions, e.g., a normal distribution $\mathcal{N}(\mu, \sigma^2)$, by setting μ equal to the ensemble mean and σ to the ensemble standard deviation [225]. Further techniques exist, such as the ensemble model output statistics (EMOS) method [24], which is common in the atmospheric sciences [139]. Numerical models, mechanistic models and ML algorithms can all be used as individual predictors in the ensemble, but in this work, the focus is set on the application of ML algorithms.

Deep ensembles (DE) are ensembles of NNs where each of the NNs predicts the parameters of a predictive distribution, e.g., μ and σ, and the ensemble prediction is then a mixture of Gaussians [134]. Snapshot ensembles are generated by taking snapshots of NN weights at local minima during the training process, thus achieving an ensemble of NNs by training a single NN [116]. Fast geometric ensembling also trains a single NN and explores the weight space to find a set of diverse weight samples with minimal loss, thereby generating an ensemble of NNs [87]. Depth uncertainty networks are ensembles of sub-networks of increasing depth which share weights, thus needing only a single forward pass [17]. Ensembles of other ML algorithms also exist, e.g., gradient boosting ensembles [155]. Out of these ensemble methods, DE have recently shown quite promising results in terms of prediction performance. The nature of DE has a certain resemblance to distributed methods, i.e., independent and parallel training of multiple NNs.

The learning of probabilistic ML models in a distributed and federated setting is the central challenge of this work. In partitioned variational inference (PVI), federated approximate learning of BNNs is presented [44]. Sharma et al. presented an extension of PVI including differential privacy [196]. However, probabilistic predictions of continuous variables are not implemented and these methods can therefore not be used as benchmarks. Concurrent to this work, a number of articles on probabilistic FL were published. Kassab & Simeone introduced distributed Stein variational gradient descent (DSVGD), where non-random and interacting particles represent the model global posterior. Iterative updates of the particles are performed on the devices by minimizing the global free energy [129]. Al-Shedivat et al. proposed federated posterior averaging (FedPA), where the clients use MCMC to infer approximations of the local posteriors, and the server computes an estimate of the model global posterior [14]. Zhang et al. used FedAvg with differential privacy to learn a Bayesian long short-term memory (LSTM) network, where Monte Carlo dropout is applied to compute probabilistic forecasts of solar irradiation [242]. Finally, Chen & Chao proposed federated Bayesian ensemble (FedBE), where the local models are aggregated via Bayesian model ensemble. There, a probability distribution is fitted to the local models but with the assumption that the server has access to some training data [50]. In the next section, an alternative method for the application of FL to probabilistic ML models is proposed.

4.2 Federated Learning with Predictive Uncertainty

The proposed method, FedAvg-Gaussian (FedAG), builds on the FedAvg algorithm [156]. In FedAvg, clients perform quick local updates on the weights, which are then aggregated in a central server. In turn, the aggregated weights are then returned to the clients for further learning. FedAvg does not consider predictive uncertainty. However, before the weights are aggregated, information on their distribution over the clients is known. Xiao et al. showed that during FL, client weights become increasingly correlated but not closer to each other in terms of distance metrics [228]. This fact may be a sign that the client weights are a good, approximate Bayesian marginalization, i.e., the weights represent multiple *basins of attraction* in the posterior [226]. In the presented algorithm, this information is used to introduce weight uncertainty in the aggregation step of the FedAvg algorithm. Thereby, a probabilistic model is approximated by treating the set of local weights of the clients as an empirical posterior distribution for the weights of the global model. This approach is somewhat similar to SWAG, where instead of the SGD trajectory, local weights of the federated model are used. Using the probabilistic model, inference is performed by calculating predictive distributions for new, unseen data.

A pseudo-code for FedAG is shown in Algorithm 1. In the aggregation step, a probability distribution is fitted to the set of client weights. The choice of this distribution is arbitrary, but for simplicity, normal distributions are considered in this work. Hence, the posterior distributions are found by calculating the mean value $\mu_{\mathbf{w}}$ and variance $\sigma^2_{\mathbf{w}}$ of weights $\mathbf{w}^{(k)}$. In turn, the posterior distributions $p(\mathbf{w}|\mathcal{D})$ are returned to the clients. The clients use the expected value, i.e., the mean value $\mu_{\mathbf{w}}$ of the weight posterior distributions to further iterate local updates to the global model using their own data, but calculate probabilistic predictions with $p(\mathbf{w}|\mathcal{D})$.

Algorithm 1: FedAvg-Gaussian (FedAG). C is the fraction of devices used in each round, K is the total number of devices, $\mathcal{D}_k$ is the data observed by device k, B is the batch size, E_p is the number of local epochs, η_l is the learning rate and L is the squared loss function.

```
1  Server executes:
2  initialize $\mathbf{w}_0$
3  for each round $t = 1, 2, ...$ do
4      $g \leftarrow \max(C \cdot K, 1)$
5      $S_t \leftarrow$ (random set of $g$ clients)
6      for each client $k \in S_t$ do
7          $\mathbf{w}^{(k)}_{t+1} \leftarrow \text{ClientUpdate}(k, p(\mathbf{w}_t|\mathcal{D}))$
8      end
9      $p(\mathbf{w}_{t+1}|\mathcal{D}) \leftarrow \mathcal{N}(\mu_{\mathcal{D}}(\mathbf{w}^{(k)}_{t+1}), \sigma^2_{\mathcal{D}}(\mathbf{w}^{(k)}_{t+1}))$
10     return $p(\mathbf{w}_{t+1}|\mathcal{D})$ to clients
11 end
12
13 ClientUpdate$(k, p(\mathbf{w}|\mathcal{D}))$:  // Run on client k
14 $\mathcal{B} \leftarrow$ (split $\mathcal{D}_k$ into batches of size $B$)
15 $\mathbf{w} \leftarrow \mathbb{E}(p(\mathbf{w}_t|\mathcal{D}))$
16 for each local epoch $i = 1$ to $E_p$ do
17     for batch $b \in \mathcal{B}$ do
18         $\mathbf{w} \leftarrow \mathbf{w} - \eta_l \nabla L(\mathbf{w}; b)$
19     end
20 end
21 return $\mathbf{w}$ to server
```

As in FedAvg, the clients minimize the squared loss function. The client updates are therefore fast and do not require extensions in order to learn a probabilistic model. FedAG is therefore significantly less complicated than PVI, DSVGD and FedPA, which is beneficial when resources such as computing performance and storage are limited. Additionally, the only target variable during training is μ, so that the amount of operations is smaller in comparison to DE. FedAG and FedBE exhibit strong similarities, except that the server in FedAG does not require access to training data. Fig. 4.1 shows an overview of the training process where a network of end devices learns a probabilistic model. For w_i, the clients return their local updates, to which a normal distribution is fitted to generate a posterior probability distribution $p(w_i|\mathcal{D})$. The distributions $p(w_i|\mathcal{D})$ constitute the weights of the NN with input variable x, hidden units h_i, bias I and target variable $\tilde{y}$. FedAG does not require prior probabilities on the weights.

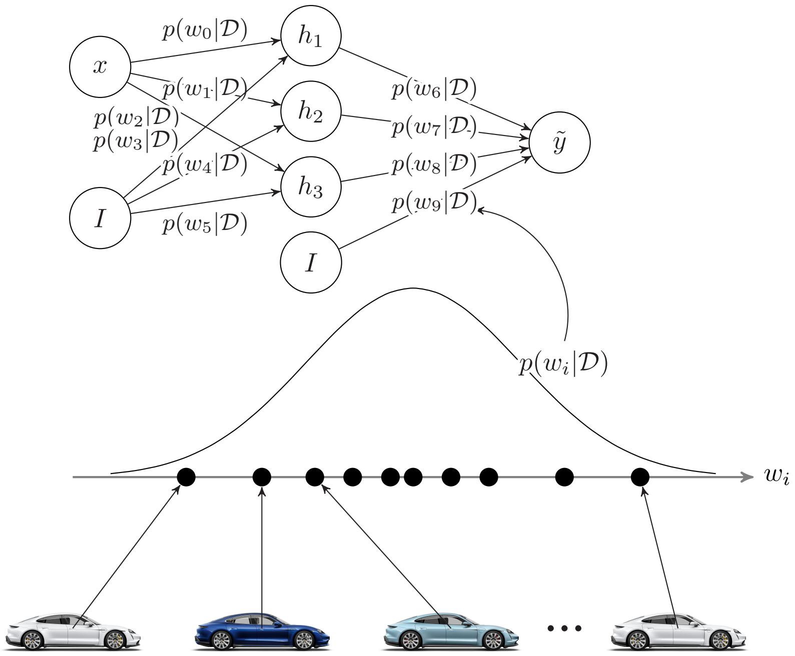


Figure 4.1: A network of end devices learns a probabilistic model.

As mentioned in Section 4.1, an exact calculation of the integral in (4.2) for the predictive distribution is generally intractable and some approximation is needed. Two variations for the algorithm are proposed: ordinary Monte Carlo (OMC) and non-parametric bootstrapping. In OMC, M sets of the weights are drawn from the posterior distributions to calculate M scalar predictions $\tilde{y}_k$ for the target variable y [89]. It may seem strange to draw sets of sample weights from a distribution created by aggregating sets of sample weights. An alternative would be to use the sample weights from the clients directly to calculate the predictions. In a sense, this resembles non-parametric bootstrapping to create an ensemble [161]. In that way, $\tilde{y}_k$ are calculated directly from the client updates. In this work, the latter sampling method is used. The predictive distribution is approximated with a normal distribution $\mathcal{N}(\tilde{\mu}_y, \tilde{\sigma}_y^2)$

$$p(y|\mathbf{x}, \mathcal{D}) = \int p(y|\mathbf{x}, \mathcal{D}, \mathbf{w})p(\mathbf{w}|\mathcal{D})\mathrm{d}\mathbf{w} := \mathcal{N}(\tilde{\mu}_y, \tilde{\sigma}_y^2) \tag{4.3}$$

$$\tilde{\mu}_y \approx \frac{1}{M}\sum_{k=1}^{M} \tilde{y}(\mathbf{x}, \mathbf{w}^{(k)}) \tag{4.4}$$

$$\tilde{\sigma}_y^2 \approx \left(\frac{1}{M}\sum_{k=1}^{M} \left[\tilde{y}(\mathbf{x}, \mathbf{w}^{(k)})\right]^2\right) - \tilde{\mu}_y^2\,, \tag{4.5}$$

where $\tilde{y}(\mathbf{x}, \mathbf{w}^{(k)})$ is a prediction calculated with features $\mathbf{x}$ and weights $\mathbf{w}^{(k)}$. In the case of a linear model, the predictive distribution takes the form

$$p(y|\mathbf{x}, \mathcal{D}, \beta) = \mathcal{N}\left(\mu_{\mathbf{w}}^{\mathrm{T}}\mathbf{x}, \beta^{-1} + \mathbf{x}\left[\sigma_{\mathbf{w}}^2\mathbf{I}\right]\mathbf{x}^{\mathrm{T}}\right)\,, \tag{4.6}$$

where β is a noise precision parameter for data $\mathcal{D}$ and is considered to be independent of the distribution of the weights $\mathbf{w}$, $\mathbf{I}$ is the identity matrix, $\mu_{\mathbf{w}}$ and $\sigma_{\mathbf{w}}^2$ are the mean and variance of the weight posterior distribution $p(\mathbf{w}|\mathcal{D})$ [31].

The communication complexity of FedAG is different from that of FedAvg. FedAG learns a posterior distribution for each weight of the model. Therefore, its communication complexity is somewhat higher than that of FedAvg. If a Gaussian posterior is assumed, each distribution is defined by its mean and standard deviation. Compared to FedAvg, the global model has twice the amount of parameters. The communication complexity of sending the global model to the clients in FedAG can thus be up to two times higher than in FedAvg, depending on the communication overhead. However, the client updates only include scalar weights $\mathbf{w}$, so the upload communication complexity in FedAG is the same as in FedAvg.

4.3 Proper Scoring Rules

To evaluate the performance of the prediction algorithms, proper scoring rules are required. Scoring rules assess the quality of probabilistic predictions by comparing the predictive distribution and the true observation [125]. A scoring rule S_R is proper if the expected score is optimized by issuing the true distribution of observations as the prediction. In this work, scores are regarded as negatively oriented, i.e., a better prediction leads to a lower score. The requirement for a scoring rule S_R to be proper is thus

$$\mathbb{E}_{y \sim P_t}\left[S_R(P_t, y)\right] \leq \mathbb{E}_{y \sim P_t}\left[S_R(Q_p, y)\right] , \tag{4.7}$$

where y is the true observation of the target variable, P_t is the true distribution of y and Q_p is a predictive distribution. The scoring rule is strictly proper if the equality in (4.7) only holds when $P_t = Q_p$ [92].

Popular scoring rules for the prediction of continuous variables are the logarithmic score $\mathcal{L}(F, y)$, the continuous ranked probability score (CRPS) and its generalization, the energy score (ES) [94]. In related work, negative log-likelihood (NLL) has been favored as a performance indicator

$$\mathrm{NLL} = \frac{1}{2}\log(2\pi\sigma_{Q_p}^2)) + \frac{(y - \mu_{Q_p})^2}{2\sigma_{Q_p}^2} , \tag{4.8}$$

where μ_{Q_p} and $\sigma_{Q_p}^2$ are the mean value and variance of the predictive distribution. NLL is equal to the negative logarithmic score and is therefore also a proper scoring rule. Furthermore, NLL is unitless which is advantageous when evaluating a model's performance on different datasets.

A good prediction is well calibrated and sharp. Calibration is the statistical consistency between the predictive distribution and the observation of the target variable. Sharpness measures the concentration of the predictive distribution. NLL measures both calibration and sharpness whereas root mean square error (RMSE) only measures calibration. RMSE is not a proper scoring rule, but due to its wide acceptance, it is included in this work to allow a direct comparison to related works. Separate measures for calibration and sharpness allow a more detailed comparison. The width of a central prediction interval, e.g., $50\,\%$, was suggested by Gneiting & Raftery as a measure for sharpness [94]. As all candidate algorithms in this work calculate a prediction in the form of a normal distribution, the standard deviation appropriately measures the sharpness by indicating the width of the central $68\,\%$ prediction interval. This is also called determinant sharpness (DS):

$$\mathrm{DS} = \det(\Sigma)^{1/2d}\,, \tag{4.9}$$

where $\Sigma \in \mathbf{R}^{d\times d}$ is the covariance matrix of the predictive distribution and d is the dimension of the target variable. In the evaluation using open-source datasets, the proper scoring rule NLL is used, as well as RMSE and DS.

4.4 Experimental Evaluation

As commonly done in the field of ML, the proposed method is validated with openly accessible empirical data. In this section, the experiments are described and the results are analyzed. In Section 4.4.1, FedAG is applied to toy regression data. Section 4.4.2 shows the setup of the empirical validations and in Section 4.4.3, the results are presented.

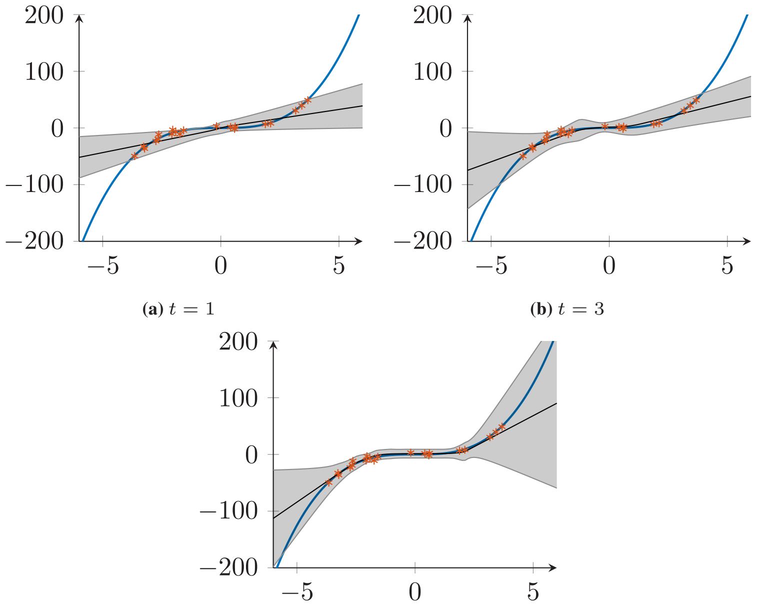
4.4.1 Regression with Toy Data

To analyze the performance of the proposed method on simple data, a one-dimensional toy dataset is generated as suggested by Hernández-Lobato & Adams [111]. In the analysis, 10 clients draw 16 independent examples from $y = x^3 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 3^2)$. The clients collaborate in learning a probabilistic NN with a single hidden layer with 100 hidden units from these data according to Algorithm 1. The data are sampled in the interval $[-4, 4]$ but predictions are calculated for the interval $[-6, 6]$. Fig. 4.2 shows the resulting probabilistic predictions after 1, 3 and 5 communication rounds. The results after $t = 5$ rounds show that FedAG can calculate accurate probabilistic predictions with low but appropriate uncertainty for input data close to the observed training data. For input data farther away from observed data, the uncertainty is high. The prediction interval thus includes the ground truth despite the scarce training data, making the prediction superior to those calculated after rounds $t = 1$ and $t = 3$. The results after $t = 5$ rounds are similar to those reported by Hernández-Lobato & Adams [111] and Lakshminarayanan et al. [134].

4.4.2 Experiment Setup

For the empirical validation, FedAG is implemented with two models: a NN with a single hidden layer and a linear regression model, denoted NN-FedAG and LR-FedAG, respectively. The experiment setup is the same as described by Hernández-Lobato & Adams [111], which was also used by Lakshminarayanan et al. [134] and Gal & Ghahramani [86]. There, 10 datasets from the UCI Machine Learning Repository are used [68]. Table 4.1 shows a summary of the corresponding datasets.

The performance of FedAG is compared to three benchmarks: Bayesian linear regression (BLR) [31], variational inference (VI) [97], and deep ensembles (DE) [134], all of which are implemented in a non-distributed setting. Gaussian posteriors are used in VI and the DE consists of 5 networks. The NNs trained using VI,



(a) $t = 1$ **(b)** $t = 3$ **(c)** $t = 5$

Figure 4.2: Results on regression for toy data after 1, 3 and 5 rounds. The blue line represents the ground truth, the orange points are exemplary observed noisy training data, the black line is the mean value of the predictive distribution and the grey area demarcates a prediction interval containing ± 3 standard deviations.

DE, and FedAG all have the same architecture with 50 hidden units with ReLU activation functions in a single hidden layer. For the *Protein Structure* and *Year Prediction MSD* datasets, 100 hidden units are used. A 20-fold cross validation is performed to evaluate test performance, where $E_p = 40$ passes over the available training data are done. For the *Protein Structure* dataset, a 5-fold cross validation is performed and for the *Year Prediction MSD* dataset, the specified split is used. The linear models, LR-FedAG and BLR, are validated in the same manner. In the federated setting, $K = 10$ devices are simulated with $C = 1$ and batch size $B = 1$. K and C are chosen so that the amount of data per device is maximized, subject to the condition that the number of devices is sufficiently large to enable an accurate approximation of the posterior distribution in the aggregation step.

Table 4.1: Summary of the UCI datasets for regression.

Datasets	**Observations**	**Features**
Boston Housing	506	13
Concrete Compression Strength	1030	8
Energy Efficiency	768	8
Kin8nm	8192	8
Naval Propulsion Power Plant	$11,934$	16
Combined Cycle Power Plant	9568	4
Protein Structure	$45,730$	9
Red Wine Quality	1599	11
Yacht Hydrodynamics	308	6
Year Prediction MSD	$515,345$	90

The training data are randomly divided into K equally large shards, each of which is assigned to a simulated device. Hence, each observation is uniquely assigned to one device.

The training of a linear model is a convex optimization and it can be expected that LR-FedAG should need no more than $t = 1$ rounds to converge. On the contrary, the training of a NN is usually a non-convex optimization and $t > 1$ rounds are therefore required for convergence of NN-FedAG in this setting. When each device has a limited amount of data, such as in small datasets or when the number of devices is increased, an even higher number of communication rounds might be required. Strong baselines in BDL are important and this work tries to generate a fair basis for the comparison of FedAG and the benchmarks [164]. For BLR and LR-FedAG, appropriate precision parameters for the variance of the target variable are estimated using the variance of the training data. In addition, conjugate priors given by unit Gaussians are used for the weight posterior distributions $p(\mathbf{w}|\mathcal{D})$ in BLR.

4.4.3 Results

With the experiment setup and proper scoring rules, the performance of FedAG and the benchmarks can be evaluated. In the following, the results of the validation are presented. Table 4.2 shows the mean NLL and standard error for the algorithms on all datasets and Table 4.3 shows the RMSE and standard error. The results for VI and DE are reported by Hernández-Lobato & Adams [111] and Lakshminarayanan et al. [134], respectively. The entries in bold denote the best performing model(s), where the performance is considered similar if the standard error intervals overlap.

Table 4.2: Mean NLL and standard error of the predictions.

Dataset	BLR	VI	LR-FedAG$^{(t=1)}$	NN-FedAG$^{(t=5)}$	DE
Boston	3.07 ± 0.03	2.90 ± 0.07	3.02 ± 0.03	$\mathbf{2.58 \pm 0.06}$	$\mathbf{2.41 \pm 0.25}$
Concrete	3.78 ± 0.02	3.39 ± 0.02	3.76 ± 0.03	$\mathbf{3.21 \pm 0.04}$	$\mathbf{3.06 \pm 0.18}$
Energy	5.12 ± 0.05	2.39 ± 0.03	5.31 ± 0.06	2.07 ± 0.04	$\mathbf{1.38 \pm 0.22}$
Kin8nm	1.17 ± 0.04	-0.90 ± 0.01	1.03 ± 0.04	-0.87 ± 0.01	$\mathbf{-1.20 \pm 0.02}$
Naval	-3.55 ± 0.02	-3.73 ± 0.12	-3.45 ± 0.01	-3.21 ± 0.01	$\mathbf{-5.63 \pm 0.05}$
Power	2.97 ± 0.01	2.89 ± 0.01	2.94 ± 0.01	2.92 ± 0.01	$\mathbf{2.79 \pm 0.04}$
Protein	3.07 ± 0.00	2.99 ± 0.01	3.08 ± 0.00	2.95 ± 0.00	$\mathbf{2.83 \pm 0.04}$
Wine	1.50 ± 0.07	$\mathbf{0.98 \pm 0.01}$	$\mathbf{1.01 \pm 0.03}$	$\mathbf{0.99 \pm 0.02}$	$\mathbf{0.94 \pm 0.12}$
Yacht	3.63 ± 0.05	3.44 ± 0.16	4.02 ± 0.07	1.92 ± 0.06	$\mathbf{1.18 \pm 0.21}$
Year	$3.73\pm$ NA	$3.86\pm$ NA	$3.72\pm$ NA	$3.66\pm$ NA	$\mathbf{3.35\pm}$ **NA**

The performance of the two linear models, BLR and LR-FedAG is similar. In 8 out of 10 datasets, LR-FedAG$^{(t=1)}$ performs similarly or slightly better than BLR in terms of NLL. BLR significantly outperforms LR-FedAG$^{(t=1)}$ only in two datasets, the *Energy Efficiency* and *Yacht Hydrodynamics* datasets, which are also two of the smallest datasets. Hence, each device only has access to a small amount of data. In 9 out of 10 datasets, the performance of LR-FedAG$^{(t=1)}$ and BLR is almost identical in terms of RMSE.

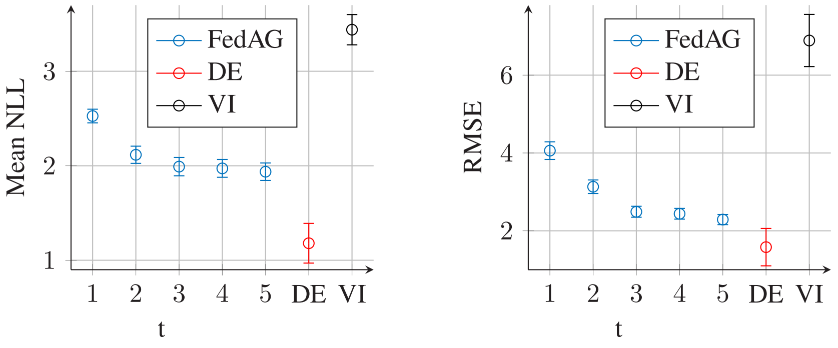
In the results for NNs, the difference between the three algorithms, VI, FedAG, and DE is somewhat significant. DE achieve the best results, followed by FedAG

Table 4.3: RMSE and standard error of the predictions.

Dataset	BLR	VI	LR-FedAG$^{(t=1)}$	NN-FedAG$^{(t=5)}$	DE
Boston	4.87 ± 0.22	$\mathbf{4.32 \pm 0.29}$	4.96 ± 0.22	$\mathbf{4.07 \pm 0.18}$	$\mathbf{3.28 \pm 1.00}$
Concrete	10.58 ± 0.33	7.13 ± 0.12	10.52 ± 0.33	$\mathbf{6.50 \pm 0.20}$	$\mathbf{6.03 \pm 0.58}$
Energy	4.35 ± 0.14	2.65 ± 0.08	4.36 ± 0.14	$\mathbf{2.02 \pm 0.07}$	$\mathbf{2.09 \pm 0.29}$
Kin8nm	0.20 ± 0.00	0.10 ± 0.00	0.20 ± 0.00	0.10 ± 0.00	$\mathbf{0.09 \pm 0.00}$
Naval	0.01 ± 0.00	$\mathbf{0.00 \pm 0.00}$	0.01 ± 0.00	0.01 ± 0.00	$\mathbf{0.00 \pm 0.00}$
Power	4.74 ± 0.05	4.33 ± 0.04	4.56 ± 0.05	4.45 ± 0.05	$\mathbf{4.11 \pm 0.17}$
Protein	5.18 ± 0.02	4.84 ± 0.03	5.18 ± 0.02	$\mathbf{4.63 \pm 0.02}$	$\mathbf{4.71 \pm 0.06}$
Wine	$\mathbf{0.65 \pm 0.02}$	$\mathbf{0.65 \pm 0.01}$	$\mathbf{0.65 \pm 0.02}$	$\mathbf{0.65 \pm 0.02}$	$\mathbf{0.64 \pm 0.04}$
Yacht	9.12 ± 0.52	6.89 ± 0.67	9.12 ± 0.52	2.29 ± 0.15	$\mathbf{1.58 \pm 0.48}$
Year	$9.51\pm$ NA	$9.03\pm$ NA	$9.51\pm$ NA	$9.35\pm$ NA	$\mathbf{8.89}\pm$ **NA**

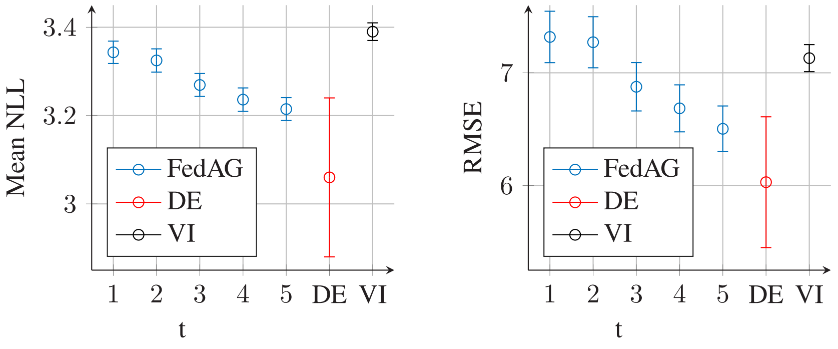
and VI. In 8 out of 10 datasets, FedAG outperforms VI in terms of NLL and in 3 out of 10 datasets, the performance of FedAG approaches that of DE. In terms of RMSE, the performance of FedAG and DE is similar in 5 out of 10 datasets. Further rounds ($t > 5$) do not improve the results of NN-FedAG significantly.

To further compare the performance of VI, DE and NN-FedAG over the course of the communication rounds, the dataset *Concrete Compression Strength* is investigated, where the performance of the methods is similar, and the dataset *Yacht Hydrodynamics* where DE show a significant advantage in terms of NLL. Fig. 4.3 shows the prediction performance (NLL and RMSE) of the algorithms on these two datasets. In Fig. 4.3a and Fig. 4.3b, NN-FedAG outperforms VI already after $t = 1$ rounds, both in terms of NLL and RMSE. However, NN-FedAG reaches a certain saturation and cannot match the performance of DE, despite a significant improvement in NLL and RMSE after $t = 5$ rounds. In Fig. 4.3c and Fig. 4.3d, NN-FedAG and VI show similar performance after $t = 1$ rounds. With increasing number of communication rounds t, NLL and RMSE of FedAG improve. After $t = 5$ rounds, the results of NN-FedAG and DE overlap, i.e., the algorithms achieve similar performance, though DE still retains a slight advantage. In the initial round of FedAG, different devices might find weights corresponding to different minima of the NN's loss function, so that the global model's initial weight posterior distributions are not optimal. As shown in



(a) Yacht Hydrodynamics NLL

(b) Yacht Hydrodynamics RMSE



(c) Concrete Compression Strength NLL

(d) Concrete Compression Strength RMSE

Figure 4.3: Prediction performance of NNs trained with FedAG, DE, and VI in terms of NLL and RMSE on the datasets *Yacht Hydrodynamics* and *Concrete Compression Strength*.

Section 2.3, the degree of the loss function is dependent on the number of hidden layers [51]. It can therefore be expected that loss functions of small NNs have few local minima, and that the global minimum can be found within relatively few communication rounds in FedAG. Accordingly, larger NNs might require more communication rounds. On the two datasets in Fig. 4.3, it is observed how the performance of NN-FedAG gradually improves with increased rounds t. This can also be observed on other datasets.

Another important property of the predictive distributions is their sharpness, which is measured with DS. In Table 4.4, the mean DS of the predictive distributions calculated with NN-FedAG$^{(t=5)}$ and DE are shown. Of the datasets that exhibit similar performance in terms of NLL, *Boston Housing* and *Concrete Compression Strength* can be predicted with greater sharpness by FedAG than DE, whereas DE's predictions of *Red Wine Quality* are sharper on average. In 7 out of 10 datasets, FedAG predicts on average a sharper distribution than DE.

Table 4.4: Mean determinant sharpness (DS) of the predictive distributions calculated with NN-FedAG$^{(t=5)}$ and DE.

Dataset	NN-FedAG$^{(t=5)}$	DE
Boston	4.05	4.79
Concrete	6.37	7.07
Energy	2.16	2.67
Kin8nm	0.10	0.14
Naval Propulsion	0.01	0.02
Power Plant	4.30	5.48
Protein	3.88	4.59
Red Wine	0.79	0.69
Yacht	2.85	0.94
Year Prediction	11.12	7.85

4.5 Computational Complexity

In addition to the predictive performance of the algorithms, their computational complexity is of significant importance. The candidate algorithms have different computational complexity at training time and at testing time. On the one hand, the two linear models BLR and LR-FedAG, have the same structure and the same amount of parameters. The predictive distributions can be computed analytically with (4.6) and no sampling is required. On the other hand, the NNs trained using VI, DE, and FedAG are more complex. VI and FedAG learn probabilistic NNs with Gaussian posterior, whereas DE are ensembles consisting of deterministic NNs with scalar weights, but with two output variables. VI maximizes a lower bound on the marginal likelihood of the NN. First, a Monte Carlo approximation for the lower bound is computed, which is then optimized using SGD. The computational complexity at training time is therefore higher in VI than in DE and FedAG, where SGD is applied directly. VI and FedAG approximate predictive distributions using Monte Carlo sampling from the posterior distributions. Contrarily, DE only have to analytically compute the two output variables of the 5 networks in the ensemble. Subsequently, the predictive distribution is approximated as a mixture of the individually computed normal distributions. The computational complexity at testing time is therefore higher in VI and FedAG than in DE. Nevertheless, the complexity is within the assumptions made in Section 3.2.

4.6 Summary and Conclusions

In this chapter, the important problem of predictive uncertainty in distributed and federated machine learning algorithms was addressed. FedAvg-Gaussian (FedAG) was presented as an efficient method for the learning of probabilistic models in a distributed, federated setting. FedAG extends FedAvg to include predictive uncertainty, by treating the set of local weights as a posterior distribution for the weights of the global model. Thereby, predictive uncertainty can be represented in a computation- and communication-efficient way, so that probabilistic on-device machine learning is realized.

FedAG was validated using open-source datasets from different fields to emphasize the capability of the algorithm. FedAG was used to learn two different models, a linear regression and a feed-forward neural network with a single hidden layer. The performance of the proposed method was evaluated on UCI regression datasets and compared to benchmark methods using proper scoring rules. When implemented with a linear regression model, FedAG's performance is similar to that of a BLR. FedAG with a neural network can after $t = 5$ communication rounds outperform VI on most datasets and its performance approaches that of DE on several datasets.

As each of the clients in FedAG only has access to a fraction of the dataset, it cannot be expected to out-perform the benchmarks BLR, VI and DE, which simultaneously have access to complete datasets. Nevertheless, the linear models BLR and LR-FedAG attain almost identical performance. Consequently, LR-FedAG can be applied as an alternative to BLR in federated, distributed settings. In the case of a non-linear model, the performance of NN-FedAG can generally compete with that of VI and approaches the performance of DE on some datasets. Additionally, the sharpness of the predictive distributions calculated with FedAG and DE is comparable. Hence, FedAG can be used as a probabilistic model in a federated setting, achieving predictive performance comparable with state-of-the-art non-distributed methods. Further advantages of FedAG are the retained privacy and communication efficiency [19]. Thereby, FedAG offers prediction performance comparable with state-of-the art probabilistic ML algorithms in an efficient and privacy-preserving manner.

5 Probabilistic Energy Demand and Driving Range Prediction

In Chapter 4, an extension of FedAvg with predictive uncertainty, FedAvg-Gaussian (FedAG), was presented. A network of connected BEVs and backend infrastructure in the cloud constitute a distributed system with various sources of information relevant for the energy demand prediction (EDP). By applying federated learning (FL) and computing a probabilistic prediction, the uncertainty of the distributed data is considered in a communication efficient and privacy preserving manner. This chapter presents the application of FedAvg and FedAG to the EDP of a BEV on a planned route. An efficient way to learn probabilistic EDP models is shown and the advantages of probabilistic EDPs are evaluated and accentuated. The chapter is organized as follows: An overview of related literature is given in Section 5.1. In Section 5.2, the prediction system and predictive data are presented. Section 5.3 shows the velocity prediction processing. The EDP algorithms and federated learning schemes are described in Section 5.4. The validation of the prediction is shown in Section 5.5 before the chapter is concluded in Section 5.6. The findings presented in this chapter were previously published in the author's journal article "Probabilistic Prediction of Energy Demand and Driving Range for Electric Vehicles with Federated Learning", which appeared in the *IEEE Open Journal of Vehicular Technology* in 2021 [3].

5.1 Literature Survey

In the context of energy demand and driving range prediction and BEV routing, few articles have addressed predictive uncertainty. Oliva et al. describe remaining driving range as a random variable, where the remaining battery energy is estimated with an unscented Kalman filter and the driving profile is predicted with a Markov chain. With that, a probability density function for the remaining driving range is computed [170]. Ondruska & Posner trained linear regression models to describe the mean and the variance of the energy consumption based on road segment features. Thereby, two deterministic models are used to calculate the parameters of a normal distribution for the prediction of energy consumption [172]. Scheubner et al. used a linear regression model to compute a stochastic velocity prediction, which is then used to predict a probability distribution for the energy consumption using a physical model and a sequential Monte Carlo simulation [5].

Data-driven predictions such as with ML algorithms benefit from a rich training dataset [69]. A few articles have proposed sharing data between vehicles and the cloud, so that a user can benefit from the experience of other users, ultimately leading to more accurate predictions. Grubwinkler et al. proposed an energetic road map created through crowd-sourcing by collecting information on energy consumption of BEVs while driving a road segment [99]. Tseng & Chau applied the concept of *participatory sensing* to gather crowd-sourced data for the prediction of vehicle energy demand [211]. Straub et al. presented another approach for creating an energetic road map, by collecting crowd-sourced driving profiles where the gaps in data coverage were eliminated using ML methods [204].

By applying FedAG to the EDP problem, the advantages of crowd-sourcing can be extended to probabilistic models in an efficient and privacy preserving manner. Recent publications showed the application of FL in vehicle-to-vehicle (V2V) communications [190], in autonomous driving [184], and in traffic flow prediction [147]. To the best of the author's knowledge, the first application of FL in EDP for BEVs was presented in [3].

5.2 Probabilistic Prediction System

The digital ecosystem in which the EDP operates is a distributed system of connected vehicles and backend infrastructures in the cloud. In this distributed system, large amounts of data can be used to learn ML models, which typically have high computational requirements. The central challenge is to make use of information in the distributed system to enable accurate and robust probabilistic predictions, while considering aspects such as privacy protection and lean communications.

In Chapter 3, the importance of system architecture and module placement was demonstrated for the performance and QoS of driving range prediction and charge planning software. By placing the prediction algorithm parts intelligently across the vehicle and cloud, the performance can be increased. The learning of the models is performed in the vehicle, so that training data remains in the vehicle. Thereby, the communication between the vehicle and the cloud covers only the transfer of the model weights. Furthermore, the predictions are computed in the cloud, so that the transmission of predictive data from the cloud to the vehicle is reduced to the final predictions. In that way, the amount of data transferred between the vehicles and the cloud is minimized. Keeping the training data within the vehicle and to generally minimize the data transfer is beneficial in terms of security and data protection. Fig. 5.1 shows an overview of the distributed system. The ego vehicle and the vehicle fleet share their model weights $\mathcal{W}$ in a central backend in the cloud, where a probabilistic NN is built. When a destination D is entered in the ego vehicle's navigation system, the route and predictive information is queried in the traffic and routing database (TRDB) and a probabilistic EDP $\widetilde{E}_c$ is computed with the probabilistic NN.

5.2.1 Measurement Drives and Powertrain Model

In this work, a dataset first presented by Scheubner et al. is used [5]. The dataset includes 20 real world measurement drives performed by 10 different drivers. All relevant data is logged in the vehicle with a sampling rate of 10 Hz. To generate

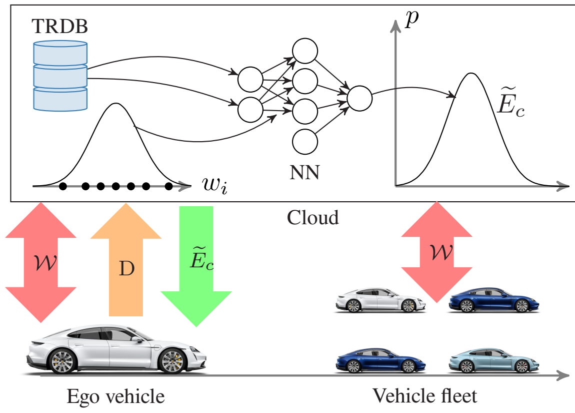


Figure 5.1: Schematic overview of the distributed system.

unified driving data from the pool of measurements with different vehicles, a simulation model for the powertrain of an electric vehicle is used. The simulation model calculates the power P and energy E_c drawn from the battery based on velocity v and driving resistance F_r. Fig. 5.2 shows a schematic overview of the powertrain model. Based on efficiency maps for components such as the gearbox (GB), electric motor (EM), and power electronics (PE), component losses are computed. These losses are denoted by red arrows in Fig. 5.2. Losses in the battery (B) are not only dependent on velocity v and driving resistance F_r, but also on parameters such as battery temperature, voltage, and internal resistance [207, 182]. The electrical and thermal modeling of the battery and the calculation of its losses fall out of the scope of this dissertation. For a complete description of the model, the reader is referred to [5]. An overview of the test drive data is shown in Table 5.1.

5.2.2 Map and Traffic Data

To complement the driving data measured in the vehicles, map and traffic data are acquired to match the driven routes. Using the GPS traces from the measurement

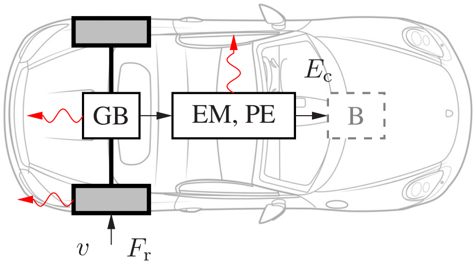


Figure 5.2: Powertrain model with input variables v, F_r and output variable E_c. The red arrows indicate simulated component losses.

Table 5.1: Test drive data

Number of drivers	10
Number of drives	20
Number of segments	1×10^4
Total distance	1896 km
Total duration	22.2 h
Mean velocity	$23.7\,\mathrm{m\,s^{-1}}$

drives, the measured data can be matched to a map. Using the IDs of the road segments that form the driven route, the TRDB can be queried to obtain static map data as well as real-time traffic information for the exact date and time of the test drive. The TRDB includes a list of properties such as road slope α, street class Λ, mean traffic speed $\overline{u}$, road curvature κ, legal speed limit v_lim, segment length l etc. Contrary to the measured driving data, map and traffic have a much lower spatial resolution, where a typical segment length is 200 m. The TRDB does not only report the mean traffic speed but also information on its distribution, such as standard deviation σ_u and percentile values $P_i(u)$ in steps of $5\,\%$. A further aspect of traffic is the traffic phase. The three-phase traffic theory divides traffic into *free flow*, *synchronized flow*, and *wide moving jam* [131]. Scheubner et al. presented a method to classify the traffic phase Θ directly in the vehicle using data from on-board cameras and radar [5]. Using this method, the estimated traffic phase is included in the dataset.

5.3 Velocity Percentile Estimation

An important factor in the energy consumption pattern is the driving speed. In this work, the velocity prediction reported by the TRDB is used. As different drivers may exhibit different driving styles and cruise at different speeds in free flowing traffic, the velocity predictions in this work are individualized. To this end, it is observed to which percentile of the velocity distribution the driver belongs on a complete trip. By minimizing the squared error between ego vehicle speed and percentile values of the traffic speed distribution, the best matching percentile can be found:

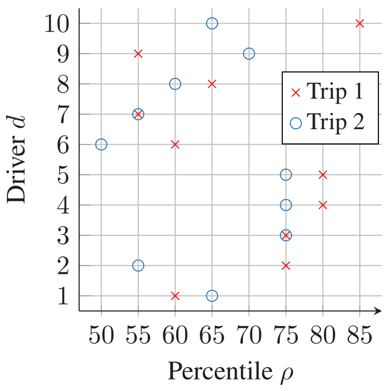
$$\rho_d = \underset{i}{\operatorname{argmin}} \left(v - P_i(u)\right)^2 \ , \tag{5.1}$$

where ρ_d is the percentile that best matches driver d, v is the speed of the ego vehicle, and $P_i(u)$ is the i-th percentile of the traffic speed distribution u. As the traffic speed distribution is very narrow in the case of a traffic jam, only synchronized flow and free flow are regarded to determine the best fitting percentile.

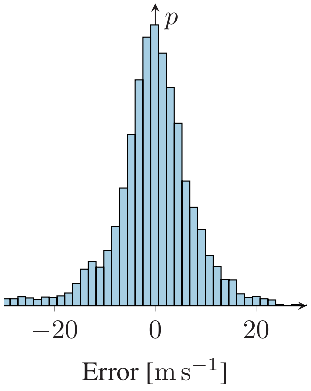
For each of the drives, (5.1) is used to find the best fitting percentile. Fig. 5.3 presents the results of the velocity prediction. Fig. 5.3a shows the observed velocity percentiles ρ_d for all 10 drivers. The drivers tend to drive faster than the median traffic speed. Most drivers tend to drive consistently, i.e., the velocity percentiles of trips 1 (×) and 2 (○) are close to each other. However, drivers 2 and 10 have significant inconsistencies between trips 1 and 2. Fig. 5.3b shows a histogram of the velocity prediction error $\epsilon = v - P_\rho(u)$. The mean value of the error distribution is $0\,\mathrm{m\,s^{-1}}$ and the prediction is therefore unbiased.

5.4 Energy Demand Prediction Algorithm

The task of the EDP algorithm is to predict the energy demand for a planned route from start to destination. The route consists of multiple road segments and for each of the segments, the energy demand is predicted based on the features



(a) Observed velocity percentiles of 10 drivers during trip 1 and trip 2.



(b) Velocity prediction error for all drivers.

Figure 5.3: Velocity percentile observation and velocity prediction error.

corresponding to the segment. In the probabilistic approach, the EDP algorithm computes a probability density for each of the segments. The total EDP is the sum of the EDPs for the individual segments. The sum of random variables γ and δ is equal to the convolution of their probability density functions:

$$p_{\gamma+\delta}(x) = \int_{-\infty}^{\infty} f_\gamma(y) f_\delta(x-y)\mathrm{d}y = (f_\gamma * f_\delta)(x)\,. \tag{5.2}$$

For a route with segments $S_1, S_2, ..., S_N$ and predictions $\widetilde{E}_{c,1}, \widetilde{E}_{c,2}, ..., \widetilde{E}_{c,N}$ the probability density for the total EDP is

$$p_{\widetilde{E}_c}(x) = (p_{\widetilde{E}_{c,1}} * p_{\widetilde{E}_{c,2}} * ... * p_{\widetilde{E}_{c,N}})(x)\,. \tag{5.3}$$

According to the central limit theorem (CLT), the sum of independent random variables tends toward a normal distribution and the total EDP is

$$p_{\widetilde{E}_c}(x) = \mathcal{N}(\mu_{\widetilde{E}_c}, \sigma^2_{\widetilde{E}_c})\,, \tag{5.4}$$

where $\mu_{\widetilde{E}_c}$ is the mean value and $\sigma^2_{\widetilde{E}_c}$ is the variance of the normal distribution [5]. To describe the energy demand on a road segment as a function of the available data, two types of regression models are applied: a linear regression (LR) and a neural network (NN). Both models can be used as probabilistic models with random weights $p(\mathbf{w})$.

The length of the road segments is not uniform. Furthermore, the training data measured in the vehicle is measured with a relatively high sampling frequency (10 Hz). Therefore, the data exhibit certain irregularities. To make the most out of the available data, a learning scheme operating on two scales is proposed in this work. One part of the model is updated with the sampling frequency of the vehicle measurement data while a second part is updated in accordance to the lower, event based frequency of road segment changes. In the following, the two-scale method and the application of FedAG are presented.

5.4.1 Two-Scale Regression Model

To optimally learn the regression model using unstructured data, two regression models are applied. The first model (M_1) is learned continuously with a data stream (10 Hz) to describe the current energy consumption. The second model (M_2) is learned based on the road segments and tries to correct the prediction of the first model. Fig. 5.4 displays a block diagram of the ML process. $E_{c,i}$ is

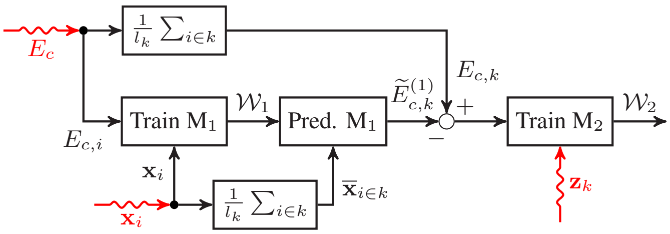


Figure 5.4: Block diagram showing a schematic overview of the training process of the two-scale regression model.

the vehicle's measured energy consumption at time i and is the target variable for M_1. Feature vector x_i includes the variables measured by the vehicle at time

i. Thereby, model weights $\mathcal{W}_1$ are learned. Simultaneously, the mean values of features $\mathbf{x}_i$ on segment k are calculated

$$\overline{\mathbf{x}}_{i\in k} = \frac{1}{l_k} \sum_{i\in k} \mathbf{x}_i l_i \,, \tag{5.5}$$

where l_k is the length of segment k and l_i is the distance driven from time $i-1$ to time i. Using the updated weights $\mathcal{W}_1$ and features $\overline{\mathbf{x}}_{i\in k}$, M_1's estimation of the energy consumption on segment k, $\widetilde{E}_{c,k}$ is computed. The difference of the true energy consumption $E_{c,k}$ and the prediction $\widetilde{E}_{c,k}$ delivers the target variable for M_2. Based on the feature vector $\mathbf{z}_k$, weights $\mathcal{W}_2$ are learned. The first model's features $\mathbf{x}_i$ are:

- v vehicle speed,
- α road slope,
- κ road curvature,
- Θ traffic phase,
- $\overline{u}_h$ historic mean traffic speed,
- $\overline{u}_c$ current mean traffic speed.

The second model's features $\mathbf{z}_k$ are $\overline{\mathbf{x}}_{i\in k}$ and additionally:

- $\overline{v}_k - \overline{v}_{k-1}$ segment speed difference,
- $\sigma_v^{(k)}$ segment speed standard deviation,
- l_k segment length.

The predictions step is limited to the road segments, as the predictive data is only reported on that scale. The final EDP is the sum of the predictions computed with M_1 and M_2:

$$\widetilde{E}_{c,k} = \widetilde{E}_{c,k}^{(1)} + \widetilde{E}_{c,k}^{(2)} \,. \tag{5.6}$$

5.4.2 Federated Learning

To learn the proposed regression models including predictive uncertainty, FedAG is applied as shown in Algorithm 1. The central part of the algorithm is the aggregation step, where a Gaussian is fitted to the set of client weights $\mathbf{w}$. In this work, the posterior distributions are found by calculating the mean value $\mu_{\mathbf{w}}$ and variance $\sigma^2_{\mathbf{w}}$ of weights $\mathbf{w}^{(k)}$. Subsequently, the posterior distributions for the weights $p(\mathbf{w}|\mathcal{D})$ are returned to the clients. The clients use the expected value $\mu_{\mathbf{w}}$ of the weight posterior distributions for further training, but the predictive distributions are computed with (4.2). FedAG learns a probabilistic prediction model in an efficient manner, which benefits from a rich database of a vehicle fleet, while minimizing communication overhead and preserving the privacy of the users. In case of an unstable internet connection, a client cannot send and receive updates from the server until a stable connection is restored, i.e., the federated learning becomes *asynchronous* [203]. In this work, a stable connection between the vehicles and the server is assumed at all times.

Not all drivers and vehicles exhibit the same driving behavior and energy consumption patterns. Therefore, a single, global model might not be the best choice for the EDP. An alternative is to generate several models, each of which acts as a global model for a subset of drivers. A cluster analysis can be executed to divide the set of drivers into subsets. Drivers can then be assigned to these subsets by observing their driving behavior and properties of their vehicles. In this work, aggregated data from the drivers is used to create two driver clusters with *k-means clustering* [148]. The features used in the clustering are:

- observed velocity percentile ρ_d,
- relative positive acceleration [74],
- relative velocity in free flowing traffic v/v_{lim},
- distribution of observed traffic phases Θ.

The following driver subsets are generated by the cluster analysis:

$$\mathcal{S}_1 = \{2, 4, 7, 8, 9\} \ ,$$
$$\mathcal{S}_2 = \{1, 3, 5, 6, 10\} \ .$$

The drivers in $\mathcal{S}_1$ cooperate in learning one model and the drivers in $\mathcal{S}_2$ learn a separate model. FedAG with clustering is denoted by FedAG-Clustering (FedAGC) with FedAvg-Clustering (FedAvgC) as the deterministic counterpart. With the availability of a larger dataset with more variety, additional features, e.g., the type of vehicle, geographical region, or the distribution of observed temperature, could be included.

5.5 Prediction Validation

To validate the prediction algorithms presented in Section 5.4, the data presented in Section 5.2 is used. A leave-one-out cross validation where the scheme depends on the learning algorithm is applied [9]. FL algorithms effectively have access to training data from the entire vehicle fleet, whereas conventional ML algorithms, e.g., SGD, can typically only access data observed by the respective vehicle. In the following, the learning algorithms FedAG, FedAGC, FedAvg, and conventional driver-individual SGD are validated and compared. The algorithms are applied to a LR and a NN. FedAGC is not applied to the NN, as a NN is able to learn more sophisticated dependencies than a linear model and benefits from a larger database. The NN has two hidden layers, each containing 50 hidden units. $E_p = 40$ passes over the available training data are done. In FedAG, $K = 10$ devices denote the 10 drivers, each of which with $C = 1$ and batch size $B = 1$. The training of a NN is a non-convex optimization and $t > 1$ rounds are usually required to ensure convergence. The results are reported after $t = 5$ rounds, but further rounds do not improve the results significantly. The training of the LR is a convex optimization and no more than $t = 1$ rounds are needed for the training to converge.

The continuous ranked probability score (CRPS) is a proper scoring rule for density predictions of continuous variables

$$\mathrm{CRPS}(Q_p, y) = \int_{-\infty}^{\infty} (Q_p(x) - H(x-y))^2 \mathrm{d}x\,, \tag{5.7}$$

where H is the Heaviside step function. The CRPS can be directly compared with the mean absolute error (MAE) of deterministic predictions. Thereby, CRPS can be used to evaluate both probabilistic and deterministic predictions, whereas NLL can only be applied to probabilistic predictions. Futhermore, CRPS is expressed in the unit of the target variable, e.g., $[\mathrm{kW\,h}]$. In the following, CRPS is used as the main performance indicator in the evaluation of the prediction algorithms.

5.5.1 Prediction Performance Evaluation

Table 5.2 shows the mean CRPS (MCRPS), RMSE, and standard error for all algorithms on all drives. The entries in bold denote the best performing models, where the performance is considered similar if the standard error intervals overlap. Boxplots for distribution of the CRPS on all drives for the algorithms are shown in Fig. 5.5. The performance of the algorithms in terms of CRPS and RMSE increases with increasing algorithm complexity and the NN trained using FedAG achieves the best performance. For the LR, FedAGC slightly improves the results of FedAG. Generally, the application of FL increases the performance significantly. Finally yet importantly, the probabilistic prediction algorithms achieve a much smaller CRPS than their deterministic counterparts.

A further visualization of the results of the two best performing algorithms is shown in Fig. 5.6. The figure shows the mean values and $95\,\%$ confidence intervals of the predictions computed with a NN and a LR trained using FedAG and FedAGC, respectively. The predictions are normalized with the true energy consumption of the respective drive. The observed energy consumption rarely matches the mean value exactly, but falls within the confidence intervals in all drives. It can be observed that the predictions computed with the NNs tend to be

Table 5.2: Performance evaluation of all algorithms on all drives with MCRPS, RMSE, and standard error.

	MCRPS [kW h]		RMSE [kW h]	
	LR	NN	LR	NN
SGD	1.2019 ± 0.2690	1.2078 ± 0.2059	1.6791 ± 0.2690	1.5048 ± 0.2059
FedAvg	0.9594 ± 0.1645	0.8088 ± 0.1274	$\mathbf{1.1978 \pm 0.1645}$	$\mathbf{0.9811 \pm 0.1274}$
FedAvgC	0.9333 ± 0.1572	-	$\mathbf{1.1578 \pm 0.1572}$	-
FedAG	$\mathbf{0.6750 \pm 0.1111}$	$\mathbf{0.5628 \pm 0.0905}$	$\mathbf{1.1978 \pm 0.1645}$	$\mathbf{0.9811 \pm 0.1274}$
FedAGC	$\mathbf{0.6594 \pm 0.0977}$	-	$\mathbf{1.1578 \pm 0.1572}$	-

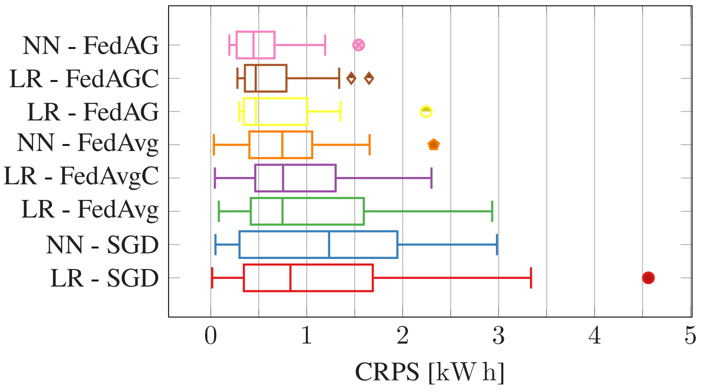


Figure 5.5: Boxplots showing the distribution of the CRPS on all test drives for the prediction algorithms.

closer to the observed value and the confidence intervals are narrower than those computed with LR.

The prediction for an exemplary drive (Nr. 18) using the best algorithm, NN-FedAG is shown in Fig. 5.7. The green band represents a 95 % confidence interval for the accumulated energy consumption at each point in the drive. The measured energy consumption is shown in purple. Additionally, the predicted traffic speed percentile value $\tilde{u}$ is shown in yellow and the observed driving speed v is shown in blue. In Fig. 5.3a, the driver (Nr. 9) displayed a moderate inconsistency in driving speed (55th and 70th percentiles). In Fig. 5.7, the velocity prediction fails

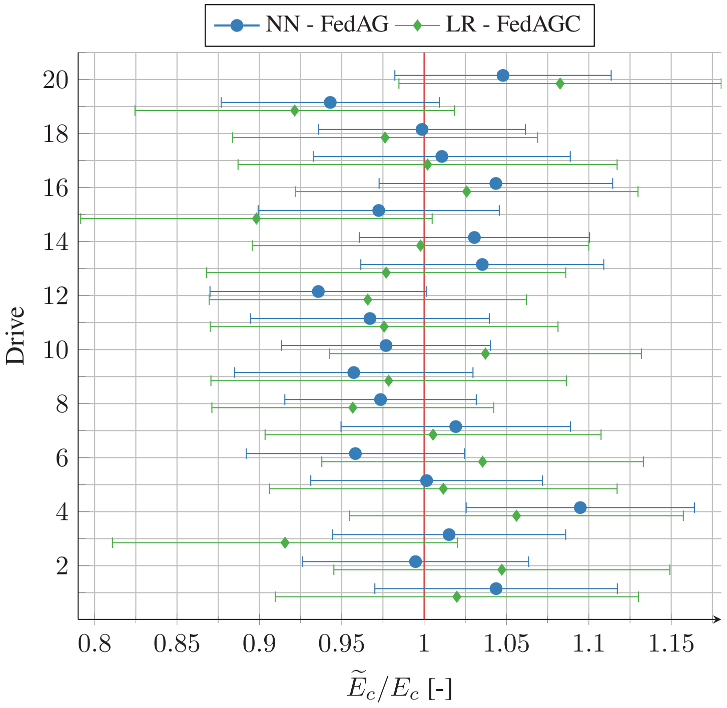


Figure 5.6: Mean values and 95 % confidence intervals of the predictions $\widetilde{E}_c$ computed with a NN and a LR trained using FedAG and FedAGC, respectively, normalized by the true energy consumption E_c.

to predict high driving speed of up to more than $50\,\mathrm{m\,s^{-1}}$ at around $80\,\mathrm{km}$. Apart from this, the observed velocity is close to the prediction and the observed energy consumption always lies within the 95 % confidence interval of the prediction.

The sharpness of a prediction is a measure for the concentration of the predictive distribution. The EDP is a univariate distribution and the determinant sharpness (DS) therefore reduces to the standard deviation of the predictive distribution. Fig. 5.8 shows boxplots displaying the distribution of the DS of the predictions on all drives for the three probabilistic algorithms. The NN computes significantly sharper predictive distributions than the LRs in all drives. The clustering in

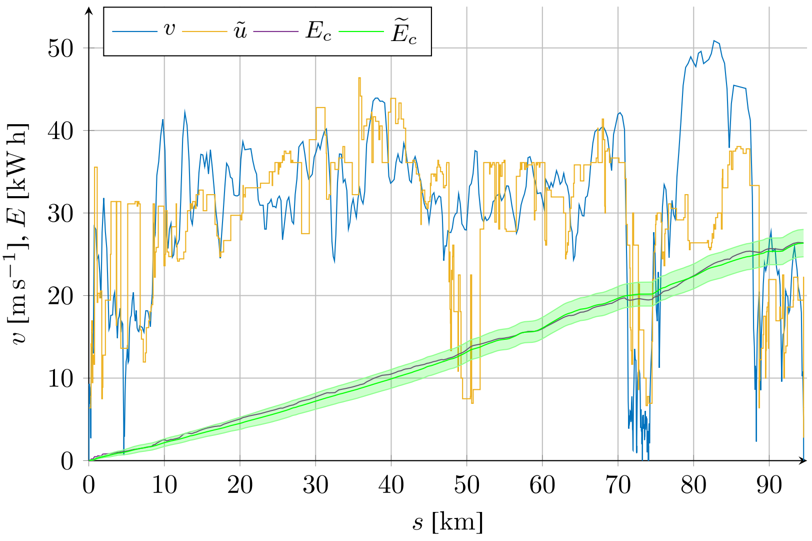


Figure 5.7: Predicted $\tilde{u}$ and observed velocity profile v and predicted $\widetilde{E}_c$ and observed accumulated energy consumption E_c of drive 18, computed with NN-FedAG.

FedAGC brings a marginally significant benefit in DS compared to FedAG when tested with a two-sample Kolmogorov-Smirnov test [127].

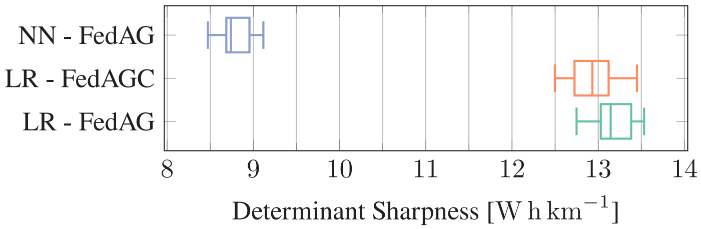


Figure 5.8: Boxplots showing the distribution of the DS of the probabilistic EDP algorithms. A smaller value represents a sharper and better prediction.

5.5.2 Destination Attainability

With a probabilistic EDP and a known available battery energy, the probability of reaching a destination, i.e., destination attainability $p(a)$, can be calculated [5]. However, this is not possible with a determinstic EDP. The available battery energy is a variable that cannot be measured directly, but is estimated with some uncertainty [150]. The attainability can thus be calculated with

$$p(a) = p(\widehat{E}_b \geq \widetilde{E}_c) = p(\widehat{E}_b - \widetilde{E}_c \geq 0)\,, \tag{5.8}$$

where $\widehat{E}_b$ is the estimated available battery energy. The $\widehat{\ }$ denotes that $\widehat{E}_b$ is an estimated value. Additionally, the amount of energy needed to achieve $p(a) = 0.99$ can be calculated using the inverse of the normal cumulative distribution function Φ

$$\widetilde{E}_{c,p} = \mu_{\widetilde{E}_c} + \sigma_{\widetilde{E}_c}\Phi^{-1}(p)\,. \tag{5.9}$$

With (5.9), the amount of energy to be charged in order to reach a destination with probability p can be computed. An important feature of the prediction and attainability estimation is that the destination is ultimately reached. To analyze this, the energy needed for $p(a) = 0.99$ is computed with (5.9) for each drive, the initial battery energy $\widehat{E}_b$ is to this value, and the attainability $p(a)$ is observed during the trip. Fig. 5.9 shows the progression of the destination attainability over the course of all drives. In some drives, the attainability exhibits fluctuation, e.g., in drives 12 and 19, and $p(a)$ is significantly lower than 0.99 at times. The gradient of a sharp prediction's cumulative distribution is proportionally large, so that a single maneuver, e.g., strong acceleration during overtaking, can have a significant impact on the attainability. However, the attainability converges to 1 when the destination is approached and the destination is reached in all drives. The linear models trained using FedAG and FedAGC are also able to accurately estimate the attainability.

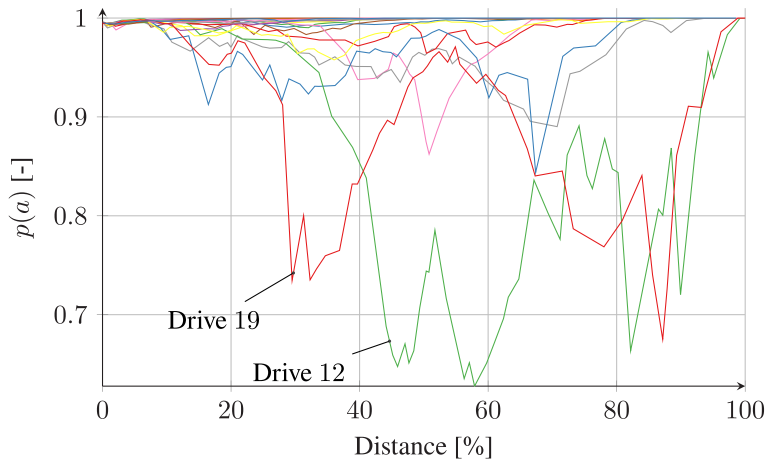


Figure 5.9: Destination attainability $p(a)$ over the course of all drives based on the EDP computed with a NN trained using FedAG.

5.5.3 Attainability Calibration

The value $p(a)$ can also be called the confidence of the attainability estimation and the observed ratio of drives in which the destination is reached can be denoted as accuracy. If the confidence always matches the accuracy, the prediction is well calibrated [102]. A measure for the calibration of the attainability decision is the difference in expectation between confidence and accuracy

$$\mathbb{E}\left[\left|\mathbb{P}\left(\widetilde{Y}=Y|\widehat{P}=p\right)-p\right|\right], \tag{5.10}$$

where the accuracy term $\mathbb{P}\left(\widetilde{Y}=Y|\widehat{P}=p\right)$ is the probability of the prediction $\widetilde{Y}$ being equal to observation Y given the estimated confidence $\widehat{P}=p$ of the predictor. A perfect calibration, although impossible, is when the expected difference is zero [165]. Using (5.9) and the observed energy consumption, the accuracy for different p-values can be computed. In this work, accuracy ξ is the empirical frequency of successful trips given EDP $\widetilde{E}_{c,p}$ and confidence p

$$\xi(p) = \frac{1}{N_D} \sum_{j=1}^{N_D} \mathbb{1}\left(\widetilde{E}_c^{(j)}(p) \geq E_c^{(j)}\right) , \tag{5.11}$$

where N_D is the total number of drives. The expected calibration error (ECE) is defined as mean difference between accuracy and confidence

$$\mathrm{ECE} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left|\xi(p_i) - p_i\right| , \tag{5.12}$$

where N_p is the number of confidence levels p tested. The maximum calibration error (MCE) is the maximum difference

$$\mathrm{MCE} = \max_i \left|\xi(p_i) - p_i\right| . \tag{5.13}$$

Finally, the idealized root mean square calibration error (RMSCE) is defined as

$$\mathrm{RMSCE} = \sqrt{\frac{1}{N_p} \sum_{i=1}^{N_p} \left(\left|\xi(p_i) - p_i\right|^2\right)} . \tag{5.14}$$

Table 5.3: Calibration error measures for the destination attainability with probabilistic EDP algorithms.

	ECE	MCE	RMSCE
LR-FedAG	0.0426	0.1323	0.0520
LR-FedAGC	0.0322	0.0965	0.0396
NN-FedAG	0.0446	0.1576	0.0576

Table 5.3 shows the ECE, MCE and RMSCE values for the probabilistic prediction algorithms. The LR trained with FedAGC has the lowest calibration errors,

followed by the LR and NN trained with FedAG. The ranking of the algorithms is thus not the same as according to the prediction performance in terms of CRPS and RMSE. Fig. 5.10 shows a reliability diagram visualizing the expected sample accuracy ξ of the attainability estimation as a function of the confidence p of the prediction. The black, straight line with slope 1 is the ideal calibration. NN-FedAG tends to be slightly under-confident for $p < 0.5$ but slightly over-confident for $p > 0.5$. Guo et al. discovered that modern NNs are often poorly calibrated [102]. A poorly calibrated prediction can not only lead to a driver being stranded with an empty battery, but an under-confident prediction may lead to the planning of unnecessary charging stops. Nonetheless, all three probabilistic EDP algorithms exhibit a sufficient calibration.

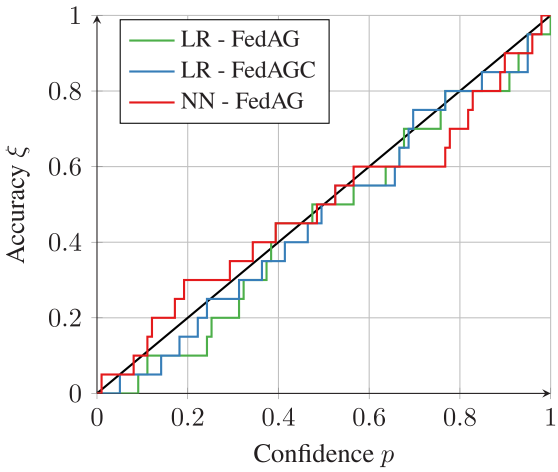


Figure 5.10: Reliability diagram for the destination attainability estimation using the probabilistic EDP algorithms.

5.6 Summary and Conclusions

In the chapter, a probabilistic energy demand prediction was implemented with federated learning algorithms. With a multi-scale regression, the prediction models can be trained using data measured in the vehicles while the predictions are computed directly in the cloud with data from a traffic and routing database. The energy demand predictions were validated with real driving data and the performance was measured with proper scoring rules. The results show that the performance of probabilistic predictions is superior to that of conventional, deterministic predictions. Furthermore, a non-linear model (NN) achieves higher performance in terms of CRPS and RMSE than a linear model (LR). A probabilistic prediction allows the estimation of destination attainability, i.e., the probability of reaching a destination using the available battery energy. The destination attainability estimation is well calibrated and the error between accuracy and confidence is low for all algorithms.

6 Energy Demand Prediction and Everyday Usability of Electric Vehicles

When planning a trip with a battery electric vehicle (BEV), energy demand prediction (EDP) is used to enable the planning of optimal charging stops by predicting the energy demand $\widetilde{E}_\mathrm{c}$ on a chosen route. Comparing $\widetilde{E}_\mathrm{c}$ with the battery's SoE $\widehat{E}_\mathrm{b}$ results in the attainability of a specific destination. Most drivers feel that their range prediction is not reliable and reserve a part of the capacity as safety margin, e.g., $20\,\%$ in [82]. Energy demand and range prediction accuracy receives relatively little attention in related research on everyday usability of BEVs. However, this factor is of high importance, as charge planning relies on the EDP. Consequently, high errors in prediction accuracy could have a considerable effect on the everyday usability of BEVs. With more accurate and reliable prediction algorithms, drivers will use more of the installed capacity and therefore, more of their range between charging stops. Furthermore, accurate prediction algorithms can optimize charge planning and reduce total charging time. Higher utilization of capacity translates into higher everyday usability of the vehicle, a key issue for electric vehicles. Here, the goal is to investigate *how accurate* a range prediction algorithm has to be to ensure everyday usability of BEVs.

This chapter introduces a detailed simulation framework to analyze the interaction between energy demand and driving range prediction, routing, charge planning and everyday usability of BEVs. The framework includes current road and charging infrastructure, in which a realistic range prediction, charge planning as well as vehicle simulation is carried out. The simulated vehicle is based on a powertrain model and can be adapted to different vehicle concepts. One simulation in the framework consists of a single trip which comprises driving and possible charging stops. Thereby, a multitude of realistic scenarios can be calculated by random trip generation, where total travel time is the measure for the everyday usability.

The chapter is organized as follows: Section 6.1 shows a survey of relevant literature. Section 6.2 introduces the routing and charge planning algorithm. Section 6.3 presents the actual trip simulation framework and the analysis of the key factors concerning everyday usability of BEVs. Subsequently, the results of the simulations are shown in Section 6.4 before the chapter is concluded in Section 6.5. The methodology employed in this chapter was previously introduced by the author in the following sources: the journal article "An Investigation into Key Influence Factors for the Everyday Usability of Electric Vehicles", which appeared in the *IEEE Open Journal of Vehicular Technology* in 2021 [2], as well as in the conference presentations "Analysis of the Impact of Range Estimation Errors on Long-Distance Electric Vehicle Trips" [6] and "Probabilistic Energy Demand Prediction, Routing, and Charge Planning for Electric Vehicles" [8]. This methodology serves as the foundation for the current investigation.

6.1 Literature Survey

A primary goal of mobility is transporting people and goods as quickly as possible. A key performance indicator for everyday usability is therefore the time spent charging and driving the vehicle on each trip [106, 194]. BEVs typically have shorter driving ranges and longer charging times than conventional vehicles, which, in turn, may limit everyday usability so that BEVs have not gained significant popularity. Previous work on this topic has been done with limited granularity. When looking at the everyday usability, the usual approach is to compare the total ranges of BEVs with trip statistics from mobility surveys or with driving profiles collected with GPS trackers [67, 159]. A significant drawback of these approaches is that the variability in energy consumption and driving range between different scenarios is not included. Moreover, Pasaoglu et al. found that mobility surveys alone are not enough to evaluate feasibility and the everyday usability [177]. Driving profiles provide a basis for the evaluation of everyday usability and a review of different studies using driving profiles was recently given by Meinrenken et al. [157].

In the majority of published results, charging stops are either neglected or severely simplified with assumptions about the availability of charging points (CPs) [241, 65, 158, 200]. Researchers rather tend to assume that no charging should be necessary to make BEVs feasible [144, 88, 167, 178, 197, 142]. Driving profiles as well as simulations have been used to determine optimal CP positioning [124, 229, 233], CP power demand [22, 18, 42, 91, 107, 171] as well as for battery lifetime analysis [168].

One of the most important applications of EDP is, in addition to the display of remaining driving range, BEV routing and charge planning. The objective of routing and charge planning algorithms is to minimize total travel time, subject to certain constraints such as driving range and charging capacity. Total travel time is the sum of driving time, charging time, and time overhead for authentication and handling at a CP. BEV routing and charge planning is a NP-hard (non-deterministic polynomial-time hard) optimization problem that requires substantial computation effort [13]. Charge planning has been studied extensively to enable time or energy-optimal routing for electric vehicles [15, 26, 193, 192]. Furthermore, the uncertainty of the EDP has been considered in BEV routing applications [78, 236, 179, 117, 25]. By including velocity trajectory planning and auxiliary consumer control in the optimization of route and charging stops, the probability of finding a globally optimal strategy is increased. Small adjustments to the velocity profile or auxiliary consumers may increase the range of the vehicle, which in turn may lead to faster routes if a charging stop can be skipped or optimized [27, 185, 53, 133]. The velocity trajectory planning significantly increases the complexity of the optimization and falls out of the scope of this dissertation.

6.2 Electric Vehicle Routing and Charge Planning

Conventional routing algorithms using deterministic EDPs cannot consider the destination attainability or the estimated risk of a given route and charge plan. Such routing algorithms can only determine a binary attainability based on a constant safety margin Δb_E. Therefore, a destination such as a CP is deemed attainable if

$$\widetilde{E}_\mathrm{c} \leq (1 - \Delta b_E)\, \widehat{E}_\mathrm{b}\,. \tag{6.1}$$

In Section 5.5.2, the calculation of a destination attainability was presented, which is the probability of reaching a destination given the available battery energy. This is an important property which allows probabilistic predictions to express their confidence. With a probabilistic EDP, the destination attainability is a continuous scalar variable between 0 and 1. The task of probabilistic routing and charge planning is thus to find a route π including necessary charging stops, which minimizes travel time T subject to a destination attainability level $p(a)$

$$\min T(\pi)\ \mathrm{s.t.}\, p(a) \geq \zeta\,, \tag{6.2}$$

where ζ is a threshold value for an acceptable attainability $p(a)$ [174]. The opposite value $1 - p(a)$ denotes the risk of the planned route. Fig. 6.1 shows EDPs for three hypothetical routes. The left axis shows the amount of energy needed to charge at a given charging stop as a function of the risk and the right axis shows the corresponding charging time in minutes. A constant charging power and equal driving times for the three route alternatives are assumed. The three predictions have different expected values and different standard deviations. When looking at the expected value of the predictions (at $p(a) = 0.5$), route 3 will have the shortest travel time. With a slightly higher risk, route 3 might possibly be driven without a charging stop. With decreasing risk, the energy demand increases according to the sharpness of the predictions. The prediction for route 3 is not as sharp as for the other two routes, so that the energy demand increases faster with decreasing risk. For low risks, route 2 becomes the optimal route. With such phenomena, there exist circumstances where travel times and the route choice is highly dependent on the destination attainability $p(a)$ or risk $1 - p(a)$.

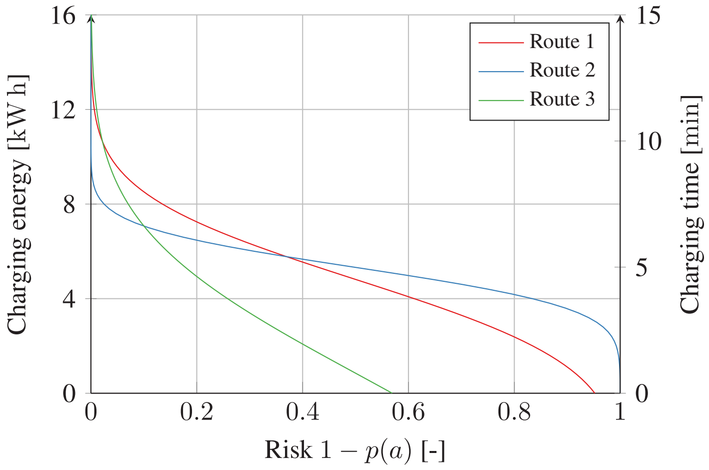


Figure 6.1: Charging energy in kW h and charging time in min as functions of risk $1 - p(a)$.

The charge planning algorithm in this dissertation has no performance requirements except that the fastest route should be found. Therefore, a brute-force search of all possible charging stop combinations can be performed. A maximum number of charging stops N_{CPs} can be defined to reduce the number of combinations. The algorithm then explores all unordered combinations of $1 ... N_{\text{CPs}}$ nearby CPs. An upper bound for the number of reasonable combinations can be determined using the binomial coefficient

$$\sum_{i=0}^{N_{\text{CPs}}} \binom{N_V}{i} \leq \binom{N_V}{N_{\text{CPs}}} \frac{N_V - (N_{\text{CPs}} - 1)}{N_V - (2N_{\text{CPs}} - 1)}, \tag{6.3}$$

where N_V is the total number of candidate CPs in the geographical region between the start and destination positions. To speed up the computation, candidate CPs can be sorted by charging power into a priority queue V_P with a stopping criteria so that combinations with slow CPs are only explored if no solution using only high-performance CPs is found. Combinations, in which a single leg is greater than the range of the vehicle, can be discarded. The fastest path from the start position to the destination with stops at the given CPs can be calculated with A-Star or Dijkstra's algorithm [104, 62].

For each charging stop, the amount of electrical energy needed to charge is determined. If the vehicle's characteristic charging curve decreases monotonically, charging only until a CP with a greater charging power can be reached will lead to time-optimal strategies. The corresponding charging time τ is calculated using the relation

$$\tau = \int_{\widehat{E}_{\mathrm{b},start}}^{\widehat{E}_{\mathrm{b},end}} \frac{\mathrm{dE}}{\min(P(E), P_V)} \,. \tag{6.4}$$

Here, $\widehat{E}_{\mathrm{b},start}$ and $\widehat{E}_{\mathrm{b},end}$ are the battery SoEs at beginning and end of charging, respectively, P_V is the available CP power at vertex V, and $P(E)$ is the possible charging power at SoE $\widehat{E}_{\mathrm{b}} = q\,Q$, where q is the state of charge (SoC) and Q is the battery capacity. It is assumed that the CP is always free and based on measurements for this work, the time overhead at a CP is $2.5\,\mathrm{min}$ on average. The charging times τ are then used to update the travel time T and the route with the least travel time is selected. The procedure is shown in Algorithm 2.

6.3 Stochastic Framework for Trip Simulation

In this work, the approach for trip simulation consists of a stochastic framework, whose individual steps are shown in Algorithm 3. First, vehicle parameters are defined. Subsequently, the desired length of the trip is set, random start and endpoints on the map are selected and the fastest route between them is calculated. $\widetilde{E}_{\mathrm{c}}$ is predicted analog to the EDP model presented in the previous chapter and the destination attainability indicates whether charging is required

Algorithm 2: Charge planning algorithm. $\widetilde{E}_{c,k}$ is the predicted energy demand, V are nearby CPs, Q is the battery capacity, q is start SoC, l is route length, T is travel time, ΔE_b is charging energy, τ is charging time and π is route.

```
1  get maximum reasonable number of charging stops $N_{\text{CPs}} = f(q, Q, l, \widetilde{E}_c)$
2  get priority queue with CPs $V$
3  for $n = 1, 2, ..., N_{\text{CPs}}$ do
4      for each unordered combination of $n$ CPs $V_\omega$ do
5          calculate fastest path $\pi_\omega$ from start through $V_\omega$ to destination
6          get travel time $T_\omega$
7          for each $V_\omega$ in $\pi_\omega$ do
8              calculate necessary charging energy $\Delta E_b(V, \widetilde{E}_c)$
9              calculate charging time $\tau(\Delta E_b)$
10         $T'_\omega = T_\omega + \tau$
11 choose $\pi'$ with fastest travel time
   Result: fastest route $\pi'$ including necessary charging stops
```

during the trip. If necessary, charging stops are added to the trip. Lastly, the trip is simulated, generating virtual test drive data. To obtain insights into key influences for everyday usability, a multitude of trips is simulated with different EDP algorithms and algorithm properties. In the remainder of this section, the individual steps in the simulation framework are explained in detail.

6.3.1 Vehicle Properties

The proposed framework includes a virtual vehicle with the powertrain topology shown in Fig. 5.2. Different characteristic parameters can be adjusted, leading to different energy consumption behavior. These parameters are vehicle mass m, aerodynamic drag area $c_d A$, and tire rolling resistance f_r. Another important specification is the charging power P, which is dependent on various factors such as charging strategy and thermal conditions [210]. Modern BEVs are equipped with systems that pre-condition the battery before charging, to maximize the

Algorithm 3: Simulation process. π is the fastest route, $\widetilde{E}_\mathrm{c}$ is the EDP, Δb_E is safety margin, and $\widehat{E}_\mathrm{b}$ is battery capacity.

```
1 choose vehicle parameters
2 draw route start and destination points
3 get fastest route $\pi$
4 predict energy consumption $\widetilde{E}_\mathrm{c}$
5 if $\widetilde{E}_\mathrm{c} > (1 - \Delta b_E)\,\widehat{E}_\mathrm{b}$ then
6     plan charging stops with Algorithm 2
7 simulate trip with BEV powertrain model
  Result: trip data
```

possible charging power at a charging stop [208]. Dependent on driving profile and ambient temperature, optimal conditions may not be reached at all times. Furthermore, the SoC at the beginning of the charging session is important. Based on data published by Tesla drivers [145] and experiences with Porsche Boxster and Porsche Taycan BEVs, three characteristic charging power curves are defined for different levels of pre-conditioning and initial SoC. Fig. 6.2 shows the characteristic curves for the charging power P dependent on SoC q. Each of the charging curves includes an uncertainty margin, to account for variability in driving profile, environment conditions and CP performance. The three levels of pre-conditioning are optimal, near-optimal, and sub-optimal. The optimal charging curve can be assumed if the initial SoC is low and battery temperature is within optimal range. In case of higher initial SoC or battery temperatures below the optimal range, near-optimal or sub-optimal pre-conditioning levels can be assumed. In the simulation framework, a random variable following a multinomial distribution indicates the level of pre-condition attained at a given charging stop. For a higher SoC at the beginning of the charging, the probability of a less-optimal charging curve is higher. Nevertheless, if a CP's available charging power is lower than the characteristic charging curve, then the CP is the limiting factor. The most powerful public high-performance CPs can deliver a charging power of up to $350\,\mathrm{kW}$ [218]. Nonetheless, the vast majority of CPs can only charge with $50\,\mathrm{kW}$ or less [95]. The maximum charging power is dependent on

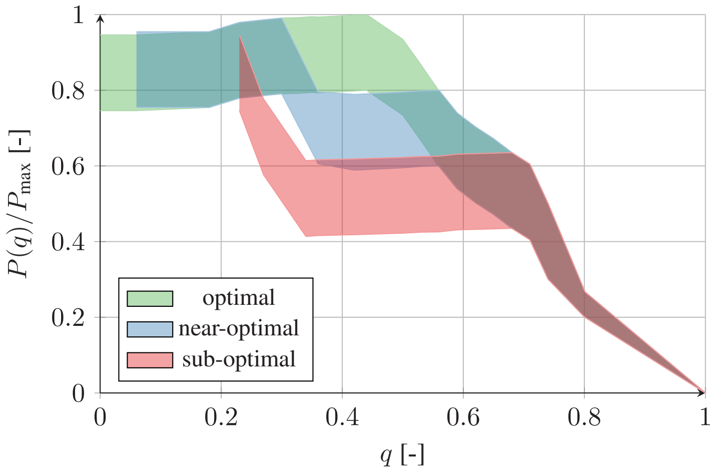


Figure 6.2: Characteristic battery charging power curves as a function of battery state of charge q, for three different levels of pre-conditioning.

the electrical current flowing in each battery module. A battery's peak C-*Rate* is defined as the ratio:

$$\text{C-}\mathit{Rate} = \frac{P_{\max}}{Q} , \tag{6.5}$$

where $P_{\max}$ is peak charging power. The Porsche Taycan has a peak C-*Rate* of about $270\,\mathrm{kW}/93.4\,\mathrm{kW\,h} = 2.89\,\mathrm{h}^{-1}$ [218], and the Tesla Model 3 $250\,\mathrm{kW}/75\,\mathrm{kW\,h} = 3.33\,\mathrm{h}^{-1}$ [208].

6.3.2 Trip Sampling

To achieve an accurate analysis of BEV driving, the trips should be chosen so that they resemble typical mobility patterns and driver behavior. The trips in this work are restricted to the region in which the test-drives were carried out. The region is defined by parallels 47°N and 54.5°N and meridians 4°E and 14°E, as shown in Fig. 6.3. The region includes parts of Germany, the Netherlands, Belgium, Luxembourg, France, Switzerland, Liechtenstein, Austria and the Czech Republic and shows diversity in road and charging infrastructure as well as in population density. To generate realistic mobility patterns, the empirical probability distributions described in the following are used to draw samples for

- route length,
- time of departure,
- start and destination positions.

In real life, a driver's daily milage is usually shorter than $72\,\mathrm{km}$ [183]. According to the *US National Household Travel Survey*, only $9.8\,\%$ of trips are 21 miles ($33.8\,\mathrm{km}$) or longer [76]. A European survey (France, Germany, Italy) showed that an average personal trip distance is approximately $16\,\mathrm{km}$ and an average business trip approximately $20\,\mathrm{km}$. The average daily driven distance is approximately $55\,\mathrm{km}$ and approximately $70\,\%$ of daily driving does not exceed $50\,\mathrm{km}$ [79].

In this dissertation, one main goal is to analyze the effects of EDP on routing and charge planning and the everyday usability of BEVs. A primary concern regarding BEV usability is their suitability for longer trips. Therefore, a total distribution of trip distances is chosen with a significant bias toward long-distance trips longer than $100\,\mathrm{km}$, when compared to mobility surveys. Fig. 6.4a shows a histogram of the distances of the sampled routes. The time of departure is relevant for the traffic state in the trip data. The traffic state influences the driving speed which has a direct influence on the vehicle's energy consumption. For departure time, a uniform distribution is used for the day of the week and a two modal normal distribution for the time of day, with modes at 8:00 in the morning and 17:00 in the afternoon as shown in Fig. 6.4b.

For each trip, the start and destination point samples are drawn from population data that include estimates of population count for 30 arc-second grid cells [48]. Fig. 6.3 shows a contour plot of the population data in the chosen geographical region. This population count is used as an empirical probability distribution $p\left(\phi, \lambda | \Psi\right)$, where Ψ is population count and ϕ, λ are the geographic coordinates of a grid cell. Thereby, highly traveled roads and routes, where population is high, are favored over routes in remote and less populated regions. The procedure is

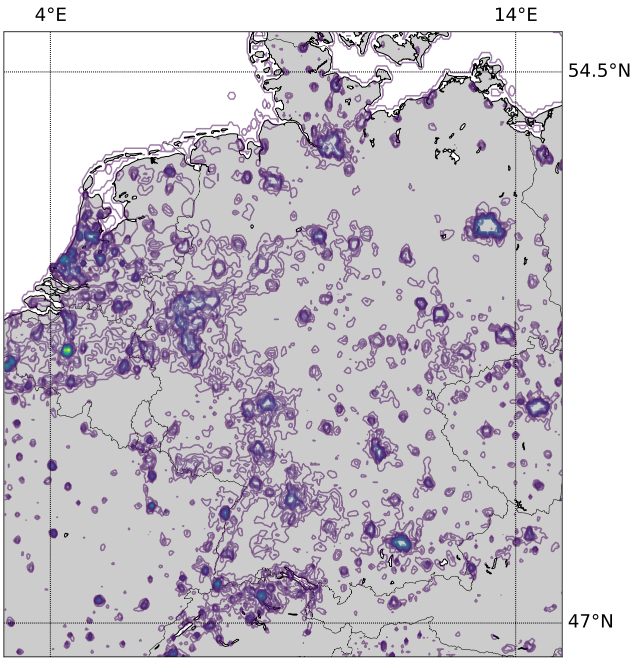
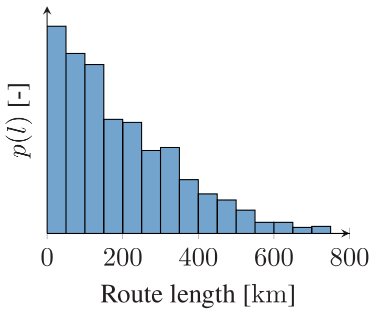
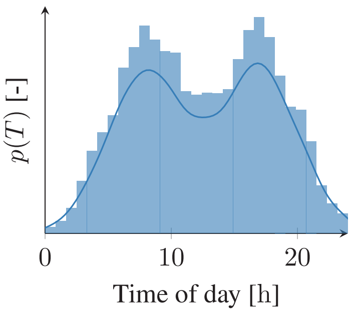


Figure 6.3: Contour plot of the population data used as a probability distribution for start and destination points.



(a) Trip length.



(b) Departure time.

Figure 6.4: Trip length and departure time distributions for the trip sampling.

described in Algorithm 4. In the algorithm, the haversine formula is utilized to calculate the air-line distance l_{air} (as the crow flies) between the start point and a potential destination as a first estimate of the route length [57].

Algorithm 4: Start and destination sampling. l is route length, ϕ and λ are latitude and longitude, respectively, Ψ is population count, R_E is earth radius, and archav is the inverse haversine function.

```
1 draw sample $l_{\text{desired}}$ from $p(l)$
2 draw sample $(\phi_1, \lambda_1)$ from $p(\Psi|\phi, \lambda)$
3 initialize air-line distance $l_{\text{air}} \not\approx l_{\text{desired}}$
4 while $l_{\text{air}} \not\approx l_{\text{desired}}$ do
5     draw sample $(\phi_2, \lambda_2)$ from $p(\Psi|\phi, \lambda)$
6     estimate air-line distance $l_{\text{air}} = R_E \cdot \text{archav}(\phi_2, \lambda_2, \phi_1, \lambda_1)$
  Result: $(\phi_1, \lambda_1)$ and $(\phi_2, \lambda_2)$
```

Using the TRDB, the fastest route π is calculated from the start point to the destination based on the speed profile u derived from mean traffic speed. Using the geographic coordinates of the route (ϕ_k, λ_k), nearby CPs can be found. In this work, an open database is used to collect information such as CP location and charging power of the current day charging infrastructure [95]. The CPs and the start and destination points constitute the vertices of the trip's directed graph $\mathcal{G}$. The weights of the edges between the vertices represent the travel time for each edge, which again is determined using the TRDB. Additionally, for each directed edge, the road segment information is extracted from the TRDB. This information is necessary for the powertrain simulation model. The route sampling is independent from the chosen vehicle, as the routes π and speed profiles u do not exceed the limitations of the vehicle models. A resulting collection of routes with CPs can be seen in Fig. 6.5.

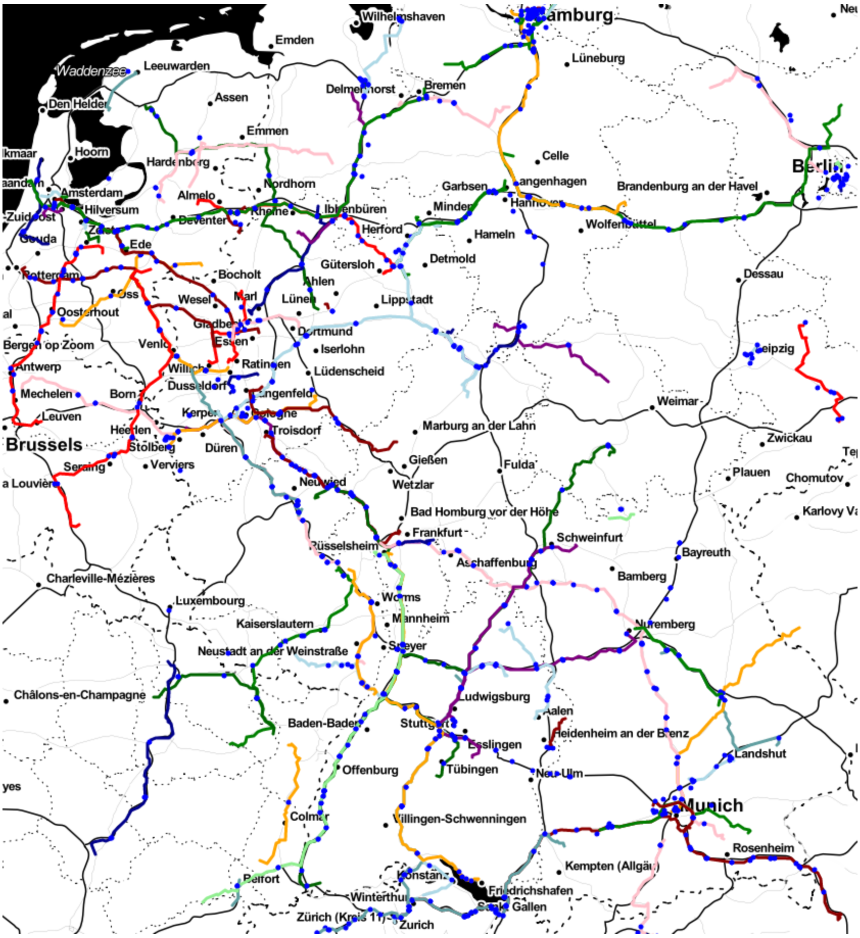


Figure 6.5: A collection of routes (colored lines) and nearby charging points (blue points). Map data (c) OpenStreetMap contributors.

6.3.3 Energy Demand Prediction

For each trip, the energy consumption is simulated with the model of a BEV powertrain introduced in Section 5.2.1. The speed profile u acquired from the TRDB is assumed to be the exact speed of the vehicle. As information about acceleration is missing, the change in kinetic energy between road segments is used to compensate for the missing acceleration data. The energy consumed on segment k is:

$$E_{c,k} = \underbrace{\int \frac{F_{r,k}}{\eta_{\text{stat}}(F_{r,k}, u_k)} \mathrm{d}s}_{\widehat{E}_{c|a=0}} + \underbrace{\frac{m}{2\eta_{\text{dyn}}} \left[(u_{k+1})^2 - (u_k)^2 \right]}_{\Delta E_{\text{kin}}} + \frac{l_k P_{\text{aux},k}}{u_k} \,, \tag{6.6}$$

where F_r is the driving resistance corresponding to the segment's mean traffic speed u, η_{stat} and η_{dyn} are static and dynamic efficiencies based on losses in powertrain components, l_k is segment length, and P_{aux} is power of auxiliary units. The first term describes the energy consumption without acceleration and the second term calculates the change in kinetic energy due to acceleration or deceleration between segments. The dynamic efficiency η_{dyn} is computed separately for acceleration and deceleration. For deceleration, the inverse of the efficiency is used. The third and last term encompasses the additional energy consumed by auxiliary equipment, such as infotainment and HVAC systems, which together draw power P_{aux}. The model was validated in [5] and is applied without changes. The total energy needed for a trip is thus

$$E_c = \sum_{i\in\pi} \sum_{k\in i} E_{c,k} = \sum_{i\in\pi} E_{c,i} \,, \tag{6.7}$$

where π is the given route, consisting of edges i, which consist of segments k. P_{aux} is constant in all simulations and is based on mean temperature in the chosen geographical region. The total power drawn by auxiliary equipment is estimated to be $1.5\,\mathrm{kW}$. In very hot or very cold weather conditions, the power demand of the auxiliaries can be significantly higher, but these corner cases are not considered in the model.

At the beginning of each simulation, an EDP is calculated. Instead of calculating this prediction explicitly as done in Chapter 5, an implicit prediction is done. Using the energies required for the graph's edges $E_{c,i}$ and the error distribution known from the results in Chapter 5, an EDP is implicitly calculated for each edge i of the graph, i.e., for each leg of the route:

$$E_{c,i} = \widetilde{E}_{c,i} + \epsilon \cdot l_i \ , \tag{6.8}$$

where l_i is the edge length and ϵ is the error in the EDP drawn from the known error distribution. Alternatively, ϵ can be drawn from an arbitrary distribution to simulate other EDP algorithms. In this way, the impact of better or worse performing range prediction algorithms can be examined. The predicted total energy consumed on the route is thus:

$$\widetilde{E}_c = \sum_{i \in \pi} \widetilde{E}_{c,i} \ . \tag{6.9}$$

As discussed in Section 6.2, an EDP is always subject to uncertainty. Therefore, a certain part of available battery energy needs to be reserved at CPs and other destinations. This safety margin Δb_E is based on the error distribution of the EDP. If the sum of $\widetilde{E}_\mathrm{c}$ and $\Delta b_E \widehat{E}_\mathrm{b}$ is larger than the available battery energy corresponding to the current SoC, $\widehat{E}_\mathrm{b}$, charging stops need to be planned with Algorithm 2.

6.3.4 Trip Simulation

Ultimately, the trip simulation is carried out as shown in Algorithm 5. The results for the EDP algorithms in Chapter 5 are used in the simulations. In the following sections, the results obtained by using different EDP algorithms and algorithm settings for a collection of trips are analyzed.

Algorithm 5: Trip simulation. π is route, $\widetilde{E}_\mathrm{c}$ is the EDP, Δb_E is safety margin, $\widehat{E}_\mathrm{b}$ is battery energy, V is vertex of route graph, and E_c is energy consumption.

```
1  calculate fastest route $\pi$
2  calculate EDP: $\widetilde{E}_\mathrm{c}$ with (6.9)
3  if $\widetilde{E}_\mathrm{c} > (1 - \Delta b_E)\, \widehat{E}_\mathrm{b}$ then
4      plan charging stops with Algorithm 2
5  start driving at vertex $V = 0$
6  for each edge $i = 1, 2, ...$ in route $\pi$ do
7      for each segment $k = 1, 2, ...$ in edge $i$ do
8          simulate $E_{c,k}$ with (6.6)
9      $E_{c,i} = \sum_{k \in i} E_{c,k}$
10     $\widehat{E}_\mathrm{b} = \widehat{E}_\mathrm{b} - E_{c,i}$
11     re-calculate $\widetilde{E}_{c,i}$ for remaining edges $i$
12     if $\widehat{E}_\mathrm{b} < 0$ then
13         vehicle stranded
14     else if $\widetilde{E}_{c,i+1} > (1 - \Delta b_E)\, \widehat{E}_\mathrm{b}$ then
15         plan additional charging stops with Algorithm 2
16     else if $V$ is planned charging stop then
17         charge battery
18     else
19         continue
```

6.4 Experimental Evaluation

In this section, the results of the simulations are presented. The investigation aims at examining the inter-dependencies of energy demand and driving range prediction, routing, charge planning, and everyday usability of BEVs. The performance indicator travel time is used to represent the results. The driving time is calculated using the speed profile u derived from mean traffic speed. The charging time is calculated with (6.4), where the characteristic charging curves shown in Fig. 6.2 are modeled from real data. Since the terms in the calculation of total travel time are independently valid, it is assumed that the travel time is valid as well. The results are mainly analyzed in relation to a benchmark, and numerous conclusions can be drawn from the characteristics of the results irrespective of their absolute numerical values.

In the simulations, a mid-size vehicle is considered, as the error distributions were determined using real data from this class of vehicles. The mid-size vehicle is simulated with different EDP algorithm variants and corresponding safety margins Δb_E, while other settings remain constant. Table 6.1 shows an overview of the simulation data. Before the simulated test runs start, the SoC at the beginning of each trip is set to $50\,\%$, thus a higher percentage of routes include at least one charging stop. This means that the resulting travel times could be somewhat longer than in the best case, when each trip starts with a fully charged battery. Nonetheless, a direct comparison between the algorithms is presented.

Table 6.1: Simulation data overview

Number of trips	473
Total distance	225 503 km
Total number of charging stops	839
Mean velocity	$27.5\,\mathrm{m\,s^{-1}}$
Net battery capacity	$90\,\mathrm{kW\,h}$
Mean consumption	$214\,\mathrm{W\,h\,km^{-1}}$
Start SoC	$50\,\%$
$c_d A$	$0.62\,\mathrm{m}^2$
m	2200 kg
f_r	0.0090
$P_{\max}$	270 kW

An exemplary route with CPs can be seen in Fig. 6.6. The blue line shows the fastest route π from the Ruhr area to Berlin. The blue points are the possible CPs along this route and the red points are the two high-performance CPs at which

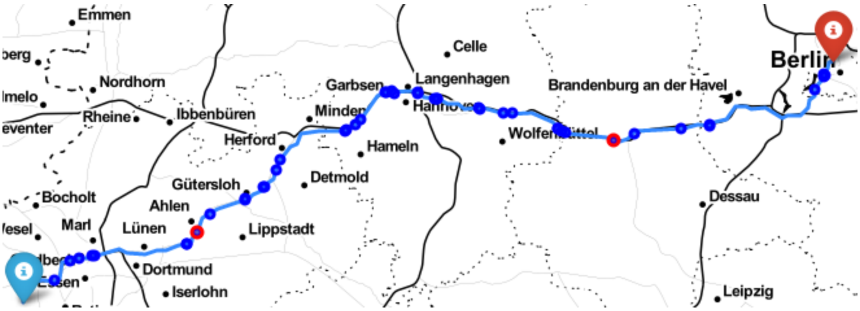


Figure 6.6: Example route from the Ruhr area to Berlin (blue line) with possible charging points (blue points) and planned charging points (red points). Map data (c) OpenStreetMap contributors.

charging stops are planned with Algorithm 2. The results for the simulation of this trip obtained with Algorithm 5 are shown in Table 6.2.

Table 6.2: Exemplary results for the simulation of the trip shown in Fig. 6.6.

1st driving leg	50...16 % SoC over 139 km in 1 h and 28 min
1st charging stop	16...77 % SoC @ 320 kW in 18 min
2nd driving leg	77...3 % SoC over 290 km in 2 h and 34 min
2nd charging stop	3...44 % SoC @ 350 kW in 11 min
3rd driving leg	44...6 % SoC over 161 km in 1 h and 41 min
Driving time	5 h and 43 min
Charging time	29 min
Travel time	6 h and 12 min
Distance	590 km

6.4.1 Probabilistic Safety Margin

As already mentioned, a requirement of the EDP is to predict the energy demand so that a destination can be reached safely without an unnecessary large safety margin Δb_E. A robust EDP should maximize the probability of attaining the destination while minimizing the safety margin, which in turn maximizes the effective driving range of the vehicle. A driver primarily experiences how far he can drive without charging and how fast he can safely travel from A to B. Hence, the user experience is positively influenced by an appropriate safety margin. The safety margin is closely related to the sharpness of the prediction and a sharp prediction leads to a smaller safety margin than a less sharp prediction. In the following, the safety margins resulting from the EDPs are analyzed as well as their impact on travel time.

A deterministic prediction includes no information about the uncertainty of the prediction and a safety margin can not be derived directly. The maximum length of traversable edges is limited by the specific maximum range $R_{\max}$ of the vehicle:

$$R_{\max} = (1 - \Delta b_E)\, \frac{Q}{\mathrm{d}E_c/\mathrm{d}s} \,, \tag{6.10}$$

where $\mathrm{d}E_c/\mathrm{d}s$ is the mean consumption. To determine a suitable Δb_E, the following condition must be satisfied:

$$\epsilon_{\max}\, R_{\max} \leq \Delta b_E\, Q \,, \tag{6.11}$$

with $\epsilon_{\max}$ being the maximum probable error in terms of energy per distance. This leads to

$$\Delta b_E^{(d)} \geq \left(\frac{\mathrm{d}E_c/\mathrm{d}s}{\epsilon_{\max}} + 1 \right)^{-1} \,, \tag{6.12}$$

where the superscripted (d) denotes that the safety margin is based on a deterministic prediction.

With probabilistic predictions, the safety margin can be directly derived from the predictive distribution. The difference between the expected value and the $p = 0.99$ value of the predictive distribution can be seen as a safety margin. Using (5.9), these values can be calculated and the safety margin $\Delta b_E^{(p)}$ is

$$\Delta b_E^{(p)} = 1 - \frac{\widetilde{E}_{c,0.5}}{\widetilde{E}_{c,0.99}} \,, \tag{6.13}$$

where the superscripted (p) denotes that the safety margin is based on a probabilistic prediction. It can be observed that when (6.12) is applied to the expected values of the probabilistic EDPs, a safety margin similar to the maximum value of the probabilistic safety margins is found:

$$\Delta b_E^{(d)} \approx \max\left[\Delta b_E^{(p)}\right] . \tag{6.14}$$

Fig. 6.7 shows empirical cumulative probability distributions for the resulting safety margins of the EDP algorithms. The algorithms trained using FedAG compute probabilistic predictions and the algorithms trained using FedAvg and SGD compute deterministic predictions. The ranking of the algorithms is the same as according to CRPS. Predictions with a NN lead to lower safety margins than with a LR. Using the probabilistic ML algorithm FedAG leads to lower safety margins than with FedAvg and SGD. The figure clearly shows that a constant, deterministic safety margin is frequently too large. An unnecessarily large safety margin reduces the effective driving range and reduces the possibilities for feasible routing and charge planning strategies [117].

6.4.2 Safety Margin and Travel Time

The safety margin determines the amount of reserved battery energy. The smaller the safety margin, the further a BEV can drive before a charging stop needs to be planned. Thereby, a faster CP might be attainable. Additionally, a planned charging stop may be shorter, since with a smaller safety margin, the energy needed for the continuation of the trip may be smaller. Additionally, the driving time may be reduced as well, if a more convenient CP, i.e., closer to the fastest route, is attainable with greater effective driving range. The safety margin has therefore a direct influence on charging and travel time.

Using the stochastic simulation framework, the trips are simulated using different EDP algorithms and corresponding safety margins. Table 6.3 shows the simulated safety margins Δb_E based on Fig. 6.7 and the resulting total driving, charging, and travel times as percentages of a benchmark algorithm LR-SGD. With probabilistic

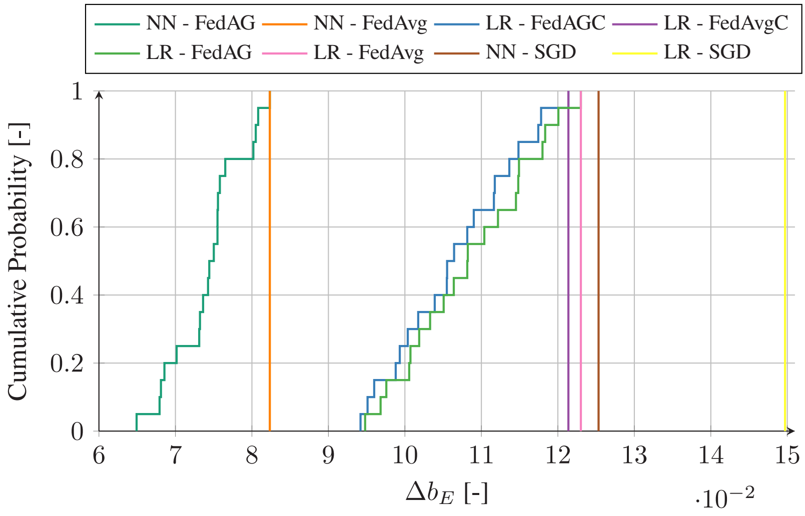


Figure 6.7: Empirical cumulative probability distributions of safety margins Δb_E of the EDP algorithms.

Table 6.3: Safety margin and driving, charging and travel time results.

	Δb_E	Driving time	Charging time	Travel time
LR-SGD	0.1497	100 %	100 %	100 %
NN-SGD	0.1253	99.8 %	95.1 %	99.2 %
LR-FedAvg	0.1230	99.7 %	94.6 %	99.1 %
LR-FedAvgC	0.1214	99.7 %	94.2 %	99.1 %
LR-FedAG	$\mathcal{N}(0.1078, 0.0084^2)$	99.7 %	90.7 %	98.6 %
LR-FedAGC	$\mathcal{N}(0.1066, 0.0080^2)$	99.6 %	90.5 %	98.5 %
NN-FedAvg	0.0823	99.4 %	86.4 %	97.8 %
NN-FedAG	$\mathcal{N}(0.0743, 0.0047^2)$	99.3 %	84.7 %	97.6 %

predictions, the safety margins follow a normal distribution. The results in Table 6.3 show that a decreased safety margin Δb_E leads to a decrease in driving time, charging time, and travel time. The advantage of a probabilistic EDP, such as with FedAG, over a deterministic EDP can be seen as well. The driving time can be slightly reduced with better EDP algorithms. With a better prediction and a lower safety margin, more CPs may be attainable and the probability of a faster route increases. Subsequently, better CPs in terms of charging power may be attainable and by arriving at a CP with a lower SoC, higher charging power can be achieved as shown in Fig. 6.2. In turn, the charging time is reduced significantly. As charging time is generally a smaller part of total travel time than driving time, a moderate reduction in total travel time is observed. A further analysis of the charging time benefit of probabilistic EDPs can be seen in Fig. 6.8, where the empirical probability distribution of difference in charging time over the complete route collective is shown. In the case of LR, the mean reduction in

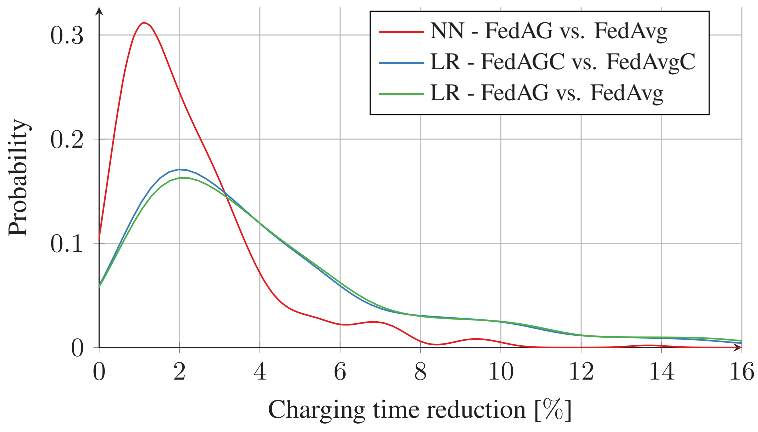


Figure 6.8: Empirical probability distribution of proportional charging time reduction between probabilistic and deterministic EDP algorithms.

charging time is approximately $4.7\,\%$ when predictive uncertainty is considered. For a NN, including predictive uncertainty leads to a mean charging time reduction of $2.3\,\%$. Predictions computed with NNs are generally significantly sharper than those computed with LRs. This means that the variance in sharpness and thus Δb_E is smaller. This leads to the somewhat smaller charging time reduction. Thus, considering predictive uncertainty explicitly improves driving, charging, and travel time, especially in regions with sparse charging infrastructure.

6.4.3 Destination Attainability and Travel Time

As stated in Section 6.2, the routing and charge planning algorithms minimize travel time subject to the destination attainability $p(a)$ or risk $1 - p(a)$. In Fig. 6.1, it can be observed that with increasing risk, travel and charging time may decrease. Conversely, minimizing risk could lead to increased travel and charging times. A desired quality of an EDP is that the risk can be minimized sufficiently without a significant increase in travel time. To analyze the connection between destination attainability, risk, and travel time, the stochastic simulation framework is used. Using the setup and trips shown in Table 6.1, the simulations are performed with different minimum destination attainability thresholds $p(a)$. Fig. 6.9 shows the resulting travel and charging times in proportion to the greatest travel and charging time, computed with LR and $p(a) = 0.9999$, as functions of risk $1 - p(a)$. Furthermore, the proportion of failed trips, i.e., the accuracy of the destination attainability estimation, is shown on the right axis of Fig. 6.9. The figure only displays the results for LR-FedAGC and NN-FedAG. The plots of charging and travel time form Pareto fronts, where charging and travel time reduces with increased risk. However, the total travel time decreases only minimally with increased risk. The calibration of the NN is close to the ideal calibration and the maximum calibration error (MCE) is 0.030. On the other hand, the LR is somewhat over-confident and exhibits a worse calibration with MCE of 0.103.

In Section 5.5.2, it was observed that all 20 measurement drives could be safely driven with $p(a) = 0.99$. With the larger set of drives in the simulation framework, $0.42\,\%$ of the routing and charge plans using $p(a) = 0.99$ lead to failures. In sparse charging infrastructures, the worst case scenario is that the driver will end up being stranded with an empty battery. That means that in some cases,

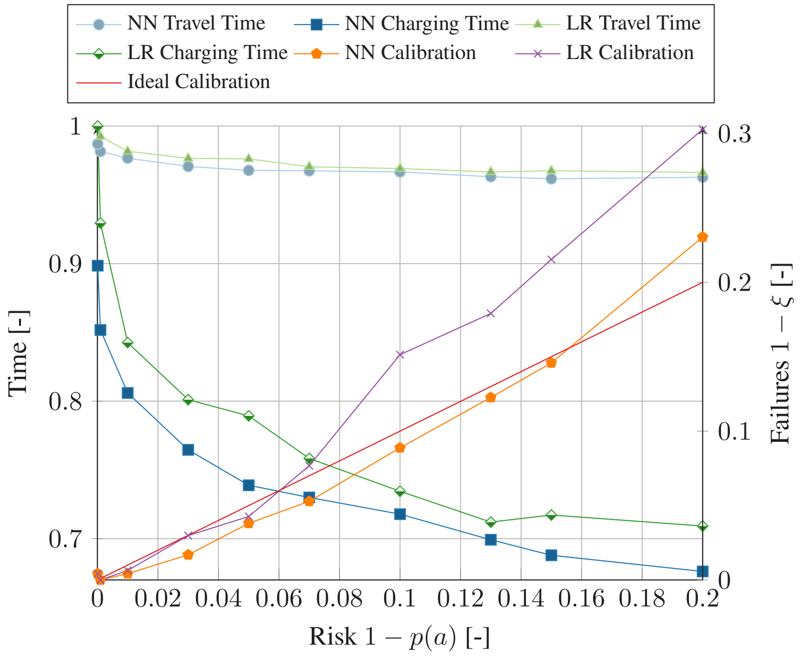


Figure 6.9: Proportional travel time, charging time (left axis) and trip failure quota (right axis) for the EDP algorithms LR-FedAGC and NN-FedAG as functions of risk $1 - p(a)$.

a new route and charge plan may need to be calculated during the trip. When minimizing the risk, it can seen in Fig. 6.9 that the travel time is not significantly increased, e.g., the proportional travel time increases from 0.96 at $p(a) = 0.8$ to 0.97 at $p(a) = 0.99$ for the NN and from 0.97 at $p(a) = 0.8$ to 0.98 at $p(a) = 0.99$ for the LR. Therefore, both EDP algorithms enable robust routing and charge planning where risk can be minimized without a significant increase in travel time. The change in charging and travel time with a change in risk is smaller with a NN than with a LR, since a NN with FedAG computes sharper predictive distributions than a LR. A sharper prediction therefore reduces the cost of minimizing risk.

6.5 Summary and Conclusions

In this chapter, the inter-dependencies between energy demand prediction, routing, charge planning, and the everyday usability of battery electric vehicles was investigated. The total travel time was chosen as a mathematical equivalent to the everyday usability, as this encompasses the ultimate goal of mobility: bringing people from A to B as quickly as possible. The difference between electric vehicles compared to conventional ones, lies in the limited driving range and charging time. Therefore, this work was concentrated on these aspects when executing a stochastic trip simulation. Random trips were drew according to probability distributions from mobility patterns and performed simulations with different EDP algorithms, safety margins, and attainability levels to investigate the influence of the EDP on everyday usability of BEVs.

A central advantage of an accurate, probabilistic energy demand prediction is the variable safety margin. This leads to a better utilization of the battery energy, where the usable battery capacity is effectively increased as the safety margin is smaller. In turn, this increases the effective driving range. Additionally, this translates into shorter driving, charging, and travel times on long distance trips. This is a more cost efficient choice than increasing total battery capacity.

The destination attainability level of a probabilistic EDP algorithm influences the accuracy and failure quota of the simulated trips. With decreasing attainability, the risk of failure increases. Furthermore, charging and travel time is influenced by the attainability. With decreasing attainability, i.e., increasing risk, charging and travel time is reduced. Nevertheless, the EDP algorithms compute sharp predictive distributions and travel time increases insignificantly when risk is minimized.

7 Conclusions and Outlook

This dissertation presents an investigation of the application of probabilistic machine learning algorithms to the prediction of energy demand and driving range in distributed systems of electric vehicles. The primary goal of this work was to develop a prediction system for a fleet of connected vehicles, in which the vehicles cooperate in learning a probabilistic model.

First, the performance of system architectures was analyzed in terms of performance indicators for the quality of service, such as end-to-end latency and network usage. The relevant hardware and software was modeled for the analysis of the control loop from the driver's destination input to the display of the planned route and remaining driving range. The simulation results show that the right system architecture for driving range prediction can improve the user experience significantly, by reducing latency and network usage. The optimized system learns the prediction model from local data within the vehicle, but computes predictions in the cloud.

Secondly, an extension of the FedAvg algorithm was presented, where predictive uncertainty is included. FedAvg-Gaussian (FedAG) treats the distribution of local weights as a posterior distribution for the weights of the global model. Therefore, the global model is learned efficiently through multiple vehicles. With the global model, predictive distributions for the target variable are computed. FedAG was applied to open-source datasets from the UCI Machine Learning Repository and to a dataset with real driving data from 10 different drivers. Using proper scoring rules, the predictive performance was evaluated, e.g., the calibration and sharpness of the predictions. On the UCI regression datasets, FedAG achieves similar performance as benchmark algorithms in non-distributed settings. In the energy demand prediction (EDP), FedAG computes probabilistic predictions for the energy demand of a planned route. Probabilistic predictions computed using FedAG are significantly better than deterministic predictions computed using FedAvg. Moreover, a neural network can more accurately predict the energy demand than a linear regression model. Furthermore, a destination attainability level was computed, enabling the estimation of the confidence of the predictions. In addition to the initial predictions, the evolution of the prediction during the trips was analyzed. In a vast majority of the trips, the predictions are stable and the destination attainability is well calibrated.

Lastly, the inter-dependencies between the EDP, routing, charge planning, and everyday usability of BEVs were investigated. In a stochastic simulation framework, a multitude of trips was simulated in the current, real road and charging infrastructure. A probabilistic EDP enables routing and charge planning with a variable safety margin, whereas with a deterministic EDP a constant safety margin must be chosen. With a smaller safety margin, travel time can be decreased. Additionally, travel time is dependent on the destination attainability level and increased attainability leads to an increase in travel time. Nevertheless, the EDP algorithms compute sharp predictions and the attainability can be maximized, i.e., risk can be minimized, without a significant increase in travel time.

FedAG can be implemented in a distributed system architecture to learn probabilistic ML models in an efficient manner. By keeping data locally within the devices, the privacy of the users is protected. The probabilistic EDP computed with FedAG are well calibrated and sharp. Hopefully, the improved quality of service, optimized travel time, and ensured safety through probabilistic EDPs will drive the acceptance of BEVs.

The research presented in this dissertation opens up interesting avenues for further work. To confirm the conclusions drawn from this dissertation, the performance of different system architectures can be tested using a fleet of connected vehicles. In addition, a strategy for deploying the proposed system in large scale can be devised. Moreover, an ever-expanding database of real driving data can be used to confirm the presented advantages of FedAvg-Gaussian and probabilistic models for BEV driving. An important question is how global, federated models can be personalized for the participating drivers. For the personalization of the local models, different aggregation and initialization methods can be applied. Evaluating such concepts in an asynchronous federated learning environment might prove an important area for future research. FedAG could benefit from an aggregation step more robust to outliers and adversarial attacks, such as in the methods Krum [32] and Aggregathor [56]. Furthermore, larger neural networks and other deep learning architectures can be applied and analyzed. Furthermore, future research should aim to benchmark FedAG against other novel federated learning concepts, such as partitioned variational inference (PVI) [44], distributed Stein variational gradient descent (DSVGD) [129], federated posterior averaging (FedPA) [14], federated Bayesian ensemble (FedBE) [50], and the combination of FedAvg and Monte Carlo dropout [242].

Developing probabilistic charge planning algorithms with stricter performance requirements for a deployment in production vehicles might prove an important area for future research. With increasing popularity of BEVs, methods from queuing theory can be used to account for waiting times at occupied charging points. The availability of charging at home or at a workplace is an interesting aspect that could be analyzed and integrated in the framework. Future research should certainly examine the aspect of velocity trajectory planning as well as auxiliary consumer control within probabilistic routing and charge planning. In the estimation of destination attainability, accounting for increasing uncertainty in the battery's state of charge estimation over the lifetime of a vehicle is an issue for future research to explore. Moreover, considering battery degradation as an additional criteria in the charge planning algorithm may extend the lifetime of BEVs. Lastly, including different weather conditions and environmental effects such as temperature, wind, and precipitation in the prediction would enable the analysis of more detailed scenarios with an increased variance in tractive and auxiliary energy consumption.

List of Figures

List of Tables

List of Publications

Journal articles

[1] Adam Thor Thorgeirsson and Frank Gauterin. Probabilistic predictions with federated learning. *Entropy*, 23(1):41, Dec 2020.

[2] Adam Thor Thorgeirsson, Stefan Scheubner, Sebastian Fünfgeld, and Frank Gauterin. An investigation into key influence factors for the everyday usability of electric vehicles. *IEEE Open Journal of Vehicular Technology*, 1:348–361, 2020.

[3] Adam Thor Thorgeirsson, Stefan Scheubner, Sebastian Fünfgeld, and Frank Gauterin. Probabilistic prediction of energy demand and driving range for electric vehicles with federated learning. *IEEE Open Journal of Vehicular Technology*, 2:151–161, 2021.

[4] Adam Thor Thorgeirsson, Moritz Vaillant, Stefan Scheubner, and Frank Gauterin. Evaluating system architectures for driving range estimation and charge planning for electric vehicles. *Software: Practice and Experience*, 51(1):72–90, 2021.

[5] Stefan Scheubner, Adam Thor Thorgeirsson, Moritz Vaillant, and Frank Gauterin. A stochastic range estimation algorithm for electric vehicles using traffic phase classification. *IEEE Transactions on Vehicular Technology*, 68(7):6414–6428, July 2019.

Conference contributions

[6] Adam Thor Thorgeirsson and Frank Gauterin. Analysis of the impact of range estimation errors on long-distance electric vehicle trips. Presented at the WKM Symposium, Karlsruhe, Germany, July 2019.

[7] Adam Thor Thorgeirsson and Frank Gauterin. Federated learning with predictive uncertainty. Poster presented at AI CON, Renningen, Germany, October 2019.

[8] Adam Thor Thorgeirsson and Frank Gauterin. Probabilistic energy demand prediction, routing, and charge planning for electric vehicles. Presented at the WKM Symposium, Stuttgart, Germany, June 2021.

[9] Patrick Petersen, Adam Thor Thorgeirsson, Stefan Scheubner, Stefan Otten, Frank Gauterin, and Eric Sax. Training and validation methodology for range estimation algorithms. In *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems - Volume 1: VEHITS*, pages 434–443, Heraklion, Crete, Greece, 2019. INSTICC, SciTePress.

Bibliography

[10] K. Abboud, H. A. Omar, and W. Zhuang. Interworking of DSRC and cellular network technologies for V2X communications: A survey. *IEEE Transactions on Vehicular Technology*, 65(12):9457–9470, December 2016.

[11] H. Abdulsalam, D. B. Skillicorn, and P. Martin. Streaming random forests. In *11th International Database Engineering and Applications Symposium (IDEAS 2007)*, pages 225–232, Banff, Alta., Canada, 2007. IEEE.

[12] David Perez Abreu, Karima Velasquez, Marilia Curado, and Edmundo Monteiro. A comparative analysis of simulators for the cloud to fog continuum. *Simulation Modelling Practice and Theory*, 101:102029, May 2020.

[13] Anagnostopoulou Afroditi, Maria Boile, Sotirios Theofanis, Eleftherios Sdoukopoulos, and Dimitrios Margaritis. Electric vehicle routing problem with industry constraints: Trends and insights for future research. *Transportation Research Procedia*, 3:452 – 459, 2014. 17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain.

[14] Maruan Al-Shedivat, Jennifer Gillenwater, Eric Xing, and Afshin Rostamizadeh. Federated learning via posterior averaging: A new perspective and practical algorithms. In *International Conference on Learning Representations*, pages 1–23, Amherst, MA, USA, 04–08 May 2021. OpenReview.net.

[15] Mahnoosh Alizadeh, Hoi-To Wai, Anna Scaglione, Andrea Goldsmith, Yue Yue Fan, and Tara Javidi. Optimized path planning for electric vehicle

routing and charging. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 25–32, Monticello, IL, USA, September 2014. IEEE.

[16] Saoussen Anssi, Karsten Albers, Matthias Dörfel, and Sébastien Gérard. chronVAL/chronSIM: A Tool Suite for Timing Verification of Automotive Applications. In *Embedded Real Time Software and Systems (ERTS2012)*, Toulouse, France, February 2012.

[17] Javier Antorán, James Urquhart Allingham, and José Miguel Hernández-Lobato. Depth uncertainty in neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10620–10634, Red Hook, NY, USA, 06–12 Dec 2020. Curran Associates, Inc.

[18] Mariz B. Arias, Myungchin Kim, and Sungwoo Bae. Prediction of electric vehicle charging-power demand in realistic urban traffic networks. *Applied Energy*, 195:738–753, June 2017.

[19] Muhammad Asad, Ahmed Moustafa, and Takayuki Ito. Fedopt: Towards communication efficiency and privacy preservation in federated learning. *Applied Sciences*, 10(8):2864, April 2020.

[20] Johannes Asamer, Anita Graser, Bernhard Heilmann, and Mario Ruthmair. Sensitivity analysis for energy demand estimation of electric vehicles. *Transportation Research Part D: Transport and Environment*, 46:182–199, 2016.

[21] Ashwin Ashok, Peter Steenkiste, and Fan Bai. Enabling vehicular applications using cloud services through adaptive computation offloading. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, MCS '15, pages 1–7, New York, NY, USA, 2015. ACM.

[22] Ali Ashtari, Eric Bibeau, Soheil Shahidinejad, and Tom Molinski. PEV charging profile prediction and analysis based on vehicle usage data. *IEEE Transactions on Smart Grid*, 3(1):341–350, March 2012.

[23] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *International Conference on Learning Representations*, pages 1–29, Addis Ababa, Ethiopia, 2020. OpenReview.net.

[24] Sándor Baran and Sebastian Lerch. Combining predictive distributions for the statistical post-processing of ensemble forecasts. *International Journal of Forecasting*, 34(3):477–496, July 2018.

[25] Rafael Basso, Balázs Kulcsár, and Ivan Sanchez-Diaz. Electric vehicle routing problem with machine learning for energy prediction. *Transportation Research Part B: Methodological*, 145:24 – 55, 2021.

[26] Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, and Tobias Zündorf. Shortest feasible paths with charging stops for battery electric vehicles. *Transportation Science*, 53(6):1627–1655, November 2019.

[27] Moritz Baum, Julian Dibbelt, Lorenz Hübschle-Schneider, Thomas Pajor, and Dorothea Wagner. Speed-Consumption Tradeoff for Electric Vehicle Route Planning. In Stefan Funke and Matúš Mihalák, editors, *14th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 42 of *OpenAccess Series in Informatics (OASIcs)*, pages 138–151, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[28] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, April 1958.

[29] András A. Benczúr, Levente Kocsis, and Róbert Pálovics. Online machine learning in big data streams. *CoRR*, abs/1802.05872:1–40, February 2018.

[30] Albert Bifet, Ricard Gavald, Geoff Holmes, and Bernhard Pfahringer. *Machine Learning for Data Streams: With Practical Examples in MOA*. The MIT Press, Cambridge, Massachusetts, 2018.

[31] Christopher Bishop. *Pattern recognition and machine learning*. Springer, New York, NY, USA, 2006.

[32] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 119–129. Curran Associates, Inc., Red Hook, NY, USA, 2017.

[33] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, 07–09 Jul 2015. PMLR.

[34] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(54):1579–1619, 2005.

[35] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, New York, NY, USA, 2012.

[36] Léon Bottou and Yann Le Cun. Large scale online learning. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 217–224, Cambridge, MA, USA, 2003. MIT Press.

[37] Andreas Braun and Wolfgang Rid. The influence of driving patterns on energy consumption in electric car driving and the role of regenerative braking. *Transportation Research Procedia*, 22:174 – 182, 2017. 19th EURO Working Group on Transportation Meeting, EWGT2016, 5-7 September 2016, Istanbul, Turkey.

[38] Paul Brebner and Anna Liu. Modeling cloud cost and performance. In *Proceedings of the International Conference on Cloud Computing & Virtualization 2010 CCV 2010*, pages 79–86, Singapore, 2010. Global Science and Technology Forum.

[39] Jochen Bröcker and Leonard A. Smith. From ensemble forecasts to predictive distribution functions. *Tellus A*, 60(4):663–678, August 2008.

[40] Antonio Brogi, Stefano Forti, Carlos Guerrero, and Isaac Lera. How to place your apps in the fog: State of the art and open challenges. *Software: Practice and Experience*, 50(5):719–740, May 2020.

[41] Alessandro Brusaferri, Matteo Matteucci, Pietro Portolani, and Andrea Vitali. Bayesian deep learning based method for probabilistic forecast of day-ahead electricity prices. *Applied Energy*, 250:1158–1175, September 2019.

[42] Thomas S. Bryden, George Hilton, Andrew Cruden, and Tim Holton. Electric vehicle fast charging station usage and power requirements. *Energy*, 152:322–332, June 2018.

[43] Christian Buckl, Alexander Camek, Gerd Kainz, Carsten Simon, Ljubo Mercep, Hauke Stähle, and Alois Knoll. The software car: Building ict architectures for future electric vehicles. In *2012 IEEE International Electric Vehicle Conference*, pages 1–8, Greenville, SC, USA, 2012. IEEE.

[44] Thang D. Bui, Cuong V. Nguyen, Siddharth Swaroop, and Richard E. Turner. Partitioned Variational Inference: A unified framework encompassing federated and continual learning. *arXiv e-prints*, abs/1811.11206:1–76, November 2018.

[45] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, August 2010.

[46] Cedric De Cauwer, Wouter Verbeke, Thierry Coosemans, Saphir Faid, and Joeri Van Mierlo. A data-driven method for energy consumption prediction and energy-efficient routing of electric vehicles in real-world conditions. *Energies*, 10(5):608–625, May 2017.

[47] Cedric De Cauwer, Wouter Verbeke, Joeri Van Mierlo, and Thierry Coosemans. A model for range estimation and energy-efficient routing of electric vehicles in real-world conditions. *IEEE Transactions on Intelligent Transportation Systems*, 21(7):2787–2800, 2020.

[48] Center For International Earth Science Information Network-CIESIN-Columbia University. Gridded population of the world, version 4 (GPWv4): Population count adjusted to match 2015 revision of UN WPP country totals, revision 11. `http://sedac.ciesin.columbia.edu/data/set/gpw-v4-population-count-adjusted-to-2015-unwpp-country-totals-rev11`, 2018. [Online; accessed 19-July-2020].

[49] G. S. S. Chalapathi, Vinay Chamola, Aabhaas Vaish, and Rajkumar Buyya. Industrial internet of things (iiot) applications of edge and fog computing: A review and future directions. In Wei Chang and Jie Wu, editors, *Fog/Edge Computing For Security, Privacy, and Applications*, pages 293–325. Springer International Publishing, Cham, Switzerland, 2021.

[50] Hong-You Chen and Wei-Lun Chao. FedBE: Making Bayesian model ensemble applicable to federated learning. In *International Conference on Learning Representations*, pages 1–21, Amherst, MA, USA, 04–08 May 2021. OpenReview.net.

[51] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gerard Ben Arous, and Yann LeCun. The Loss Surfaces of Multilayer Networks. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 192–204, San Diego, California, USA, 09–12 May 2015. PMLR.

[52] Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17, pages 613–627, Berkeley, CA, USA, 2017. USENIX Association.

[53] M. Cussigh, C. Löchel, T. Straub, and T. Hamacher. Assessing time-optimal journeys: Combined routing, charging and velocity strategies for electric vehicles. In *2020 Forum on Integrated and Sustainable Transportation Systems (FISTS)*, pages 51–57, Delft, Netherlands, 2020. IEEE.

[54] P. Cutore, A. Campisano, Z. Kapelan, C. Modica, and D. Savic. Probabilistic prediction of urban water consumption using the SCEM-UA algorithm. *Urban Water Journal*, 5(2):125–132, June 2008.

[55] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[56] Georgios Damaskinos, El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, and Sébastien Rouault. Aggregathor: Byzantine machine learning via robust gradient aggregation. In A. Talwalkar, V. Smith, and M. Zaharia, editors, *Proceedings of Machine Learning and Systems 2019*, volume 1, pages 81–106, Stanford, CA, USA, 2019. mlsys.org.

[57] Michael John De Smith, Michael F Goodchild, and Paul Longley. *Geospatial analysis: a comprehensive guide to principles, techniques and software tools*. Troubador Publishing Ltd, Leicester, UK, 2007.

[58] S. Deepak, A. Amarnath, G. Krishnan U., and S. Kochuvila. Survey on range prediction of electric vehicles. In *2019 Innovations in Power and Advanced Computing Technologies (i-PACT)*, volume 1, pages 1–7, Vellore, India, 2019.

[59] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Highway hierarchies star. In *The Shortest Path Problem*, pages 141–174, Providence, Rhode Island, 2006. American Mathematical Society.

[60] D. Dera, G. Rasool, and N. Bouaynaya. Extended variational inference for propagating uncertainty in convolutional neural networks. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, Pittsburgh, PA, USA, 2019. IEEE.

[61] G Descornet. Road-surface influence on tire rolling resistance. In W. Meyer and J. Reichert, editors, *Surface Characteristics of Roadways: International Research and Technologies*, pages 401–415. ASTM International, West Conshohocken, PA, USA, 1990.

[62] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[63] DIMACS. 9th dimacs implementation challenge: USA graph. `http://users.diag.uniroma1.it/challenge9/download.shtml`, 2006. [Online; accessed 19-July-2020].

[64] Jasenka Dizdarević, Francisco Carpio, Admela Jukan, and Xavi Masip-Bruin. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Computing Surveys (CSUR)*, 51(6):116, 2019.

[65] Jing Dong and Zhenhong Lin. Stochastic modeling of battery electric vehicle driver behavior. *Transportation Research Record: Journal of the Transportation Research Board*, 2454(1):61–67, January 2014.

[66] Jing Dong and Zhenhong Lin. Stochastic modeling of battery electric vehicle driver behavior: Impact of charging infrastructure deployment on the feasibility of battery electric vehicles. *Transportation Research Record*, 2454(1):61–67, 2014.

[67] Jing Dong, Xing Wu, Changzheng Liu, Zhenhong Lin, and Liang Hu. The impact of reliable range estimation on battery electric vehicle feasibility. *International Journal of Sustainable Transportation*, 14(11):833–842, November 2019.

[68] Dheeru Dua and Casey Graff. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2017. [Online; accessed 19-July-2020].

[69] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, Hoboken, NJ, USA, 2012.

[70] Matthias Eisel, Ilja Nastjuk, and Lutz M Kolbe. Understanding the influence of in-vehicle information systems on range stress–insights from an electric vehicle field experiment. *Transportation research part F: traffic psychology and behaviour*, 43:199–211, November 2016.

[71] Jerzy Ejsmont, Leif Sjögren, Beata Świeczko-Żurek, and Grzegorz Ronowski. Influence of road wetness on tire-pavement rolling resistance. *Journal of Civil Engineering and Architecture*, 9(11), November 2015.

[72] Peter Engel, Sebastian Meise, Andreas Rausch, and Wilhelm Tegethoff. Improving thermal management of electric vehicles by prediction of thermal disturbance variables. In *ADAPTIVE 2018, The Tenth International Conference on Adaptive and Self-Adaptive Systems and Applications*, pages 75–83, Barcelona, Spain, February 2018. IARIA.

[73] Achim Enthaler, T. Weustenfeld, Frank Gauterin, and J. Koehler. Thermal management consumption and its effect on remaining range estimation of electric vehicles. In *2014 International Conference on Connected Vehicles and Expo (ICCVE)*, pages 170–177, Vienna, Austria, 2014. IEEE.

[74] E. Ericsson. Independent driving pattern factors and their influence on fuel-use and exhaust emission factors. *Transportation Research Part D: Transport and Environment*, 6(5):325–345, 2001.

[75] Sebastian Farquhar, Lewis Smith, and Yarin Gal. Liberty or depth: Deep Bayesian neural nets do not need complex weight posterior approximations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4346–4357, Red Hook, NY, USA, 06–12 Dec 2020. Curran Associates, Inc.

[76] Federal Highway Administration. *2017 National Household Travel Survey*. U.S. Department of Transportation, Washington, DC, Washington, DC, USA, 2017. Available online: `https://nhts.ornl.gov`; accessed 19-July-2020.

[77] Joao C Ferreira, Vítor Duarte Fernandes Monteiro, and João L Afonso. Data mining approach for range prediction of electric vehicle. In *Conference on Future Automotive Technology-Focus Electromobility*, pages 1–15, Garching, Germany, 2012. Springer.

[78] Matthew William Fontana. *Optimal routes for electric vehicles facing uncertainty, congestion, and energy constraints*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2013.

[79] European Commission. Joint Research Centre. Institute for Energy and Transport. *Driving and parking patterns of European car drivers: a mobility survey.* Publications Office, Luxembourg, 2012. Available online: `https://data.europa.eu/doi/10.2790/7028`; accessed 19-July-2020.

[80] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, USA, 1962.

[81] Abbas Fotouhi, Neda Shateri, Dina Shona Laila, and Daniel J. Auger. Electric vehicle energy consumption estimation for a fleet management system. *International Journal of Sustainable Transportation*, 0(0):1–15, 2019.

[82] Thomas Franke, Isabel Neumann, Franziska Bühler, Peter Cocron, and Josef F Krems. Experiencing range in an electric vehicle: Understanding psychological barriers. *Applied Psychology*, 61(3):368–391, 2012.

[83] Roy Frostig, Rong Ge, Sham M. Kakade, and Aaron Sidford. Competing with the empirical risk minimizer in a single pass. In Peter Grünwald, Elad Hazan, and Satyen Kale, editors, *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 728–763, Paris, France, 03–06 Jul 2015. PMLR.

[84] Arika Fukushima, Toru Yano, Shuichiro Imahara, Hideyuki Aisu, Yusuke Shimokawa, and Yasuhiro Shibata. Prediction of energy consumption for new electric vehicle models by machine learning. *IET Intelligent Transport Systems*, 12(9):1174–1180, 2018.

[85] Sebastian Fünfgeld, Marc Holzäpfel, Michael Frey, and Frank Gauterin. Stochastic forecasting of vehicle dynamics using sequential monte carlo simulation. *IEEE Transactions on Intelligent Vehicles*, 2(2):111–122, 2017.

[86] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, NY, USA, 20–22 Jun 2016. PMLR.

[87] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8803–8812, Red Hook, NY, USA, 2018. Curran Associates Inc.

[88] Michele De Gennaro, Elena Paffumi, Giorgio Martini, and Harald Scholz. A pilot study to address the travel behaviour and the usability of electric vehicles in two Italian provinces. *Case Studies on Transport Policy*, 2(3):116–141, December 2014.

[89] Charles Geyer. Introduction to Markov chain Monte Carlo. In *Handbook of Markov Chain Monte Carlo*, pages 3–47. CRC Press, Boca Raton, FL, USA, 2011.

[90] Arkadeb Ghosal, Paolo Giusto, Purnendu Sinha, Massimo Osella, Joseph D'Ambrosio, and Haibo Zeng. Metrics for evaluating electronic control system architecture alternatives. In *SAE 2010 World Congress & Exhibition*, pages 45–55, Detroit, MI, USA, April 2010. SAE International.

[91] Till Gnann, Simon Funke, Niklas Jakobsson, Patrick Plötz, Frances Sprei, and Anders Bennehag. Fast charging infrastructure for electric vehicles: Today's situation and future needs. *Transportation Research Part D: Transport and Environment*, 62:314–329, July 2018.

[92] Tilmann Gneiting, F. Balabdaoui, and Adrian Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

[93] Tilmann Gneiting and Matthias Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1(1):125–151, January 2014.

[94] Tilmann Gneiting and Adrian Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, March 2007.

[95] GoingElectric. Stromtankstellenverzeichnis. `https://www.goingelectric.de/stromtankstellen/`. [Online; accessed 19-July-2020].

[96] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[97] Alex Graves. Practical variational inference for neural networks. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 2348–2356, Red Hook, NY, USA, 2011. Curran Associates Inc.

[98] PTV Group. Ptv-dimacs-europe-graph. `https://i11www.iti.kit.edu/resources/roadgraphs.php`, 2014. [Online; accessed 19-July-2020].

[99] S. Grubwinkler, T. Brunner, and M. Lienkamp. Range prediction for evs via crowd-sourcing. In *2014 IEEE Vehicle Power and Propulsion Conference (VPPC)*, pages 1–6. IEEE, 2014.

[100] S. Grubwinkler, M. Hirschvogel, and M. Lienkamp. Driver- and situation-specific impact factors for the energy prediction of evs based on crowd-sourced speed profiles. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1069–1076, Dearborn, MI, USA, 2014. IEEE.

[101] S. Grubwinkler, M. Kugler, and M. Lienkamp. A system for cloud-based deviation prediction of propulsion energy consumption for EVs. In *Proceedings of 2013 IEEE International Conference on Vehicular Electronics and Safety*, pages 99–104, Dongguan, China, 2013. IEEE.

[102] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330, Sydney, Australia, 06–11 Aug 2017. PMLR.

[103] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, June 2017.

[104] Peter Hart, Nils Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[105] Jonathan Hasenburg, Martin Grambow, Elias Grünewald, Sascha Huk, and David Bermbach. Mockfog: Emulating fog computing infrastructure in the cloud. In *Proceedings of the First IEEE International Conference on Fog Computing*, pages 144–152, Prague, Czech Republic, 2019. IEEE.

[106] Jurjen R. Helmus and R. van den Hoed. Key performance indicators of charging infrastructure. *World Electric Vehicle Journal*, 8(4):733–741, December 2016.

[107] Jurjen R. Helmus, Seyla Wachlin, Igna Vermeulen, and Mike H. Lees. SEVA: A Data driven model of Electric Vehicle Charging Behavior. *arXiv e-prints*, abs/1904.08748:1–49, April 2019.

[108] Rafik Henia, Arne Hamann, Marek Jersak, Razvan Racu, Kai Richter, and Rolf Ernst. System level performance analysis–the symta/s approach. *IEE*

Proceedings - Computers and Digital Techniques, 152:148–166, March 2005.

[109] Kristian Henrickson, Filipe Rodrigues, and Francisco Camara Pereira. Data preparation. In *Mobility Patterns, Big Data and Transport Analytics*, pages 73–106, Amsterdam, Netherlands, 2019. Elsevier.

[110] HERE Global B.V. Traffic API Developer's Guide. `https://developer.here.com/rest-apis/documentation/traffic/topics/overview.html`. [Online; accessed 19-July-2020].

[111] Jose Miguel Hernandez-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1861–1869, Lille, France, 07–09 Jul 2015. PMLR.

[112] Stephen C. Hora. Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management. *Reliability Engineering & System Safety*, 54(2):217 – 223, 1996.

[113] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[114] Hanzhang Hu, Wen Sun, Arun Venkatraman, Martial Hebert, and Andrew Bagnell. Gradient Boosting on Stochastic Data Streams. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 595–603, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.

[115] Y. Hu, W. Zhan, and M. Tomizuka. Probabilistic prediction of vehicle semantic intention and motion. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 307–313, Changshu, China, 2018. IEEE.

[116] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get M for free. *CoRR*, abs/1704.00109:1–14, 2017.

[117] G. Huber, K. Bogenberger, and H. van Lint. Optimization of charging strategies for battery electric vehicles under uncertainty. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–17, 2020.

[118] Paolo Iora and Laura Tribioli. Effect of ambient temperature on electric vehicles' energy consumption and range: Model definition and sensitivity analysis based on nissan leaf data. *World Electric Vehicle Journal*, 10(1):2, January 2019.

[119] Martin Jaggi, Virginia Smith, Martin Takac, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3068–3076. Curran Associates, Inc., Red Hook, NY, USA, 2014.

[120] Raj Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, Hoboken, NJ, USA, 1990.

[121] T. Jamil. RISC versus CISC. *IEEE Potentials*, 14(3):13–16, Aug/Sep 1995.

[122] A. Jayakumar, F. Ingrosso, G. Rizzoni, J. Meyer, and J. Doering. Crowd sourced energy estimation in connected vehicles. In *2014 IEEE International Electric Vehicle Conference (IEVC)*, pages 1–8, Florence, Italy, December 2014. IEEE.

[123] Hocheol Jeon. The impact of climate change on passenger vehicle fuel consumption: Evidence from u.s. panel data. *Energies*, 12(23):4460, November 2019.

[124] Patrick Jochem, Carsten Brendel, Melanie Reuter-Oppermann, Wolf Fichtner, and Stefan Nickel. Optimizing the allocation of fast charging infrastructure along the German Autobahn. *Journal of Business Economics*, 86(5):513–535, October 2015.

[125] Alexander Jordan, Fabian Krüger, and Sebastian Lerch. Evaluating probabilistic forecasts with scoringRules. *Journal of Statistical Software*, 90(12), 2019.

[126] Ellango Jothimurugesan, Ashraf Tahmasbi, Phillip Gibbons, and Srikanta Tirthapura. Variance-reduced stochastic gradient descent on streaming data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9906–9915. Curran Associates, Inc., RedHook, NY, USA, 2018.

[127] Frank J. Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[128] A. Kapsalis, P. Kasnesis, I. S. Venieris, D. I. Kaklamani, and C. Z. Patrikakis. A cooperative fog approach for effective workload balancing. *IEEE Cloud Computing*, 4(2):36–45, March 2017.

[129] Rahif Kassab and Osvaldo Simeone. Federated Generalized Bayesian Learning via Distributed Stein Variational Gradient Descent. *arXiv e-prints*, abs/2009.06419:1–31, September 2020.

[130] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30 of *NIPS'17*, pages 5574–5584, Red Hook, NY, USA, 2017. Curran Associates Inc.

[131] Boris S. Kerner. Three-phase traffic theory and highway capacity. *Physica A: Statistical Mechanics and its Applications*, 333:379 – 440, 2004.

[132] Jakub Konecný. Stochastic, distributed and federated optimization for machine learning. *CoRR*, abs/1707.01155:1–178, July 2017.

[133] Kurt Kruppok. *Analyse der Energieeinsparpotenziale zur bedarfsgerechten Reichweitenerhöhung von Elektrofahrzeugen*. PhD thesis, Karlsruher Institut für Technologie (KIT), Karlsruhe, Germany, 2020.

[134] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc.

[135] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[136] Chung-Hong Lee and Chih-Hung Wu. A novel big data modeling method for improving driving range estimation of EVs. *IEEE Access*, 3:1980–1993, October 2015.

[137] Chung-Hong Lee, Chih-Hung Wu, Chien-Cheng Chou, Xiang-Hong Chung, Pei-Wen Zeng, and Yi-Hsiang Lin. A framework for a connected electric vehicle cloud to learn drivers' behaviors. In *2017 25th International Conference on Systems Engineering (ICSEng)*, pages 405–411, Las Vegas, NV, USA, 2017. IEEE.

[138] E.M. Lee, J.W. Hall, and I.C. Meadowcroft. Coastal cliff recession: the use of probabilistic prediction methods. *Geomorphology*, 40(3-4):253–269, October 2001.

[139] M. Leutbecher and T.N. Palmer. Ensemble forecasting. *Journal of Computational Physics*, 227(7):3515–3539, March 2008.

[140] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*,

SenSys ’03, page 126–137, New York, NY, USA, 2003. Association for Computing Machinery.

[141] Wen Li, Patrick Stanula, Patricia Egede, Sami Kara, and Christoph Herrmann. Determining the main factors influencing the energy consumption of electric vehicles in the usage phase. *Procedia CIRP*, 48:352 – 357, 2016. The 23rd CIRP Conference on Life Cycle Engineering.

[142] Zhiheng Li, Shan Jiang, Jing Dong, Shoufeng Wang, Zhennan Ming, and Li Li. Battery capacity design for electric vehicles considering the diversity of daily vehicles miles traveled. *Transportation Research Part C: Emerging Technologies*, 72:272–282, November 2016.

[143] Shih-Yang Lin, Yun Du, Po-Chang Ko, Tzu-Jung Wu, Ping-Tsan Ho, V. Sivakumar, and Rama Subbareddy. Fog computing based hybrid deep learning framework in effective inspection system for smart manufacturing. *Computer Communications*, 160:636–642, July 2020.

[144] Zhenhong Lin. Optimizing and diversifying electric vehicle driving range for U.S. drivers. *Transportation Science*, 48(4):635–650, November 2014.

[145] Bo Lincoln. Tesla battery charging data from 801 cars. `https://forum.abetterrouteplanner.com/blogs/entry/6-tesla-battery-charging-data-from-801-cars/`, November 2018. [Online; accessed 19-July-2020].

[146] Kai Liu, Jiangbo Wang, Toshiyuki Yamamoto, and Takayuki Morikawa. Modelling the multilevel structure and mixed effects of the factors influencing the energy consumption of electric vehicles. *Applied Energy*, 183:1351 – 1360, 2016.

[147] Y. Liu, J. J. Q. Yu, J. Kang, D. Niyato, and S. Zhang. Privacy-preserving traffic flow prediction: A federated learning approach. *IEEE Internet of Things Journal*, 7(8):7751–7763, 2020.

[148] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[149] Henning Lohse-Busch, Michael Duoba, Eric Rask, Kevin Stutenberg, Vivek Gowri, Lee Slezak, and David Anderson. Ambient temperature (20°f, 72°f and 95°f) impact on fuel and energy consumption for several conventional vehicles, hybrid and plug-in hybrid electric vehicles and battery electric vehicle. In *SAE 2013 World Congress & Exhibition*, pages 1–33, Detroit, MI, USA, April 2013. SAE International.

[150] L. Lu, X. Han, J. Li, J. Hua, and M. Ouyang. A review on the key issues for lithium-ion battery management in electric vehicles. *Journal of Power Sources*, 226(1):272–288, 2013.

[151] B. Luin, S. Petelin, and F. Al Mansour. Modeling the impact of road network configuration on vehicle energy consumption. *Energy*, 137:260 – 271, 2017.

[152] Martin Lukasiewycz, Sebastian Steinhorst, Sidharta Andalam, Florian Sagstetter, Peter Waszecki, Wanli Chang, Matthias Kauer, Philipp Mundhenk, Shreejith Shanker, Suhaib A Fahmy, et al. System architecture and software design for electric vehicles. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, Austin, TX, USA, 2013. IEEE.

[153] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for Bayesian uncertainty in deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 13153–13164, Vancouver, British Columbia, Canada, 2019. Curran Associates, Inc.

[154] Redowan Mahmud, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya. Quality of experience (QoE)-aware placement of applications in fog computing environments. *Journal of Parallel and Distributed Computing*, 132:190–203, October 2019.

[155] Andrey Malinin, Liudmila Prokhorenkova, and Aleksei Ustimenko. Uncertainty in gradient boosting via ensembles. In *International Conference on*

Learning Representations, pages 1–16, Amherst, MA, USA, 04–08 May 2021. OpenReview.net.

[156] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.

[157] Christoph J. Meinrenken, Zhenyu Shou, and Xuan Di. Using GPS-data to determine optimum electric vehicle ranges: A Michigan case study. *Transportation Research Part D: Transport and Environment*, 78:102203, January 2020.

[158] Andrew Meintz, Jiucai Zhang, Ram Vijayagopal, Cory Kreutzer, Shabbir Ahmed, Ira Bloom, Andrew Burnham, Richard B. Carlson, Fernando Dias, Eric J. Dufek, James Francfort, Keith Hardy, Andrew N. Jansen, Matthew Keyser, Anthony Markel, Christopher Michelbacher, Manish Mohanpurkar, Ahmad Pesaran, Don Scoffield, Matthew Shirk, Thomas Stephens, and Tanvir Tanim. Enabling fast charging – vehicle considerations. *Journal of Power Sources*, 367:216–227, November 2017.

[159] Marc A. Melliger, Oscar P.R. van Vliet, and Heikki Liimatainen. Anxiety vs reality – sufficiency of battery electric vehicle range in switzerland and finland. *Transportation Research Part D: Transport and Environment*, 65:101–115, December 2018.

[160] F. Milani and C. Beidl. Cloud-based vehicle functions: Motivation, use-cases and classification. In *2018 IEEE Vehicular Networking Conference (VNC)*, pages 1–4, Taipei, Taiwan, December 2018. IEEE.

[161] Christopher F Mooney, Christopher L Mooney, Christopher Z Mooney, Robert D Duval, and Robert Duvall. *Bootstrapping: A nonparametric approach to statistical inference*. Number 95. Sage, London, UK, 1993.

[162] Millett Granger Morgan, Max Henrion, and Mitchell Small. *Uncertainty: a guide to dealing with uncertainty in quantitative risk and policy analysis*. Cambridge University Press, Cambridge, UK, 1990.

[163] F. Morlock, B. Rolle, M. Bauer, and O. Sawodny. Forecasts of electric vehicle energy consumption based on characteristic speed profiles and real-time traffic data. *IEEE Transactions on Vehicular Technology*, 69(2):1404–1418, 2020.

[164] Jishnu Mukhoti, Pontus Stenetorp, and Yarin Gal. On the importance of strong baselines in Bayesian deep learning. *CoRR*, abs/1811.09385:1–4, 2018.

[165] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2901–2907, Austin, Texas, 2015. AAAI Press.

[166] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, New York, NY, USA, 2012.

[167] Zachary A. Needell, James McNerney, Michael T. Chang, and Jessika E. Trancik. Potential for widespread electrification of personal vehicle travel in the United States. *Nature Energy*, 1(9):16112—16118, August 2016.

[168] Jeremy Neubauer and Eric Wood. The impact of range anxiety and home, workplace, and public charging infrastructure on simulated battery electric vehicle lifetime utility. *Journal of Power Sources*, 257:12–20, July 2014.

[169] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann, Burlington, MA, USA, 1994.

[170] J. A. Oliva, C. Weihrauch, and T. Bertram. Model-based remaining driving range prediction in electric vehicles by using particle filtering and markov chains. In *2013 World Electric Vehicle Symposium and Exhibition (EVS27)*, pages 1–10, Barcelona, Spain, 2013. IEEE.

[171] Pol Olivella-Rosell, Roberto Villafafila-Robles, Andreas Sumper, and Joan Bergas-Jané. Probabilistic agent-based model of electric vehicle charging demand to analyse the impact on distribution networks. *Energies*, 8(5):4160–4187, May 2015.

[172] P. Ondruska and I. Posner. Probabilistic attainability maps: Efficiently predicting driver-specific electric vehicle range. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1169–1174, Dearborn, MI, USA, 2014. IEEE.

[173] Róbert Ormándi, István Hegedüs, and Márk Jelasity. Efficient P2P ensemble learning with linear models on fully distributed data. *CoRR*, abs/1109.1396:1–12, 2011.

[174] Jorge Oyola, Halvard Arntzen, and David L. Woodruff. The stochastic vehicle routing problem, a literature review, part i: models. *EURO Journal on Transportation and Logistics*, 7(3):193–221, 2018.

[175] Engin Ozatay, Simona Onori, James Wollaeger, Umit Ozguner, Giorgio Rizzoni, Dimitar Filev, John Michelini, and Stefano Di Cairano. Cloud-based velocity profile optimization for everyday driving: A dynamic-programming-based solution. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2491–2505, December 2014.

[176] John O'Loughlin and Lee Gillam. Good performance metrics for cloud service brokers. In *The Fifth International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 64–69, Venice, Italy, 2014. Citeseer.

[177] Guzay Pasaoglu, Alyona Zubaryeva, Davide Fiorello, and Christian Thiel. Analysis of European mobility surveys and their potential to support studies on the impact of electric vehicles on energy and infrastructure needs in Europe. *Technological Forecasting and Social Change*, 87:41–50, September 2014.

[178] Nathaniel S. Pearre, Willett Kempton, Randall L. Guensler, and Vetri V. Elango. Electric vehicles: How much range is required for a day's driving?

Transportation Research Part C: Emerging Technologies, 19(6):1171–1184, December 2011.

[179] Samuel Pelletier, Ola Jabali, and Gilbert Laporte. The electric vehicle routing problem with energy consumption uncertainty. *Transportation Research Part B: Methodological*, 126:225–255, August 2019.

[180] C. Peng, Q. Zhang, and C. Tang. Improved tls handshake protocols using identity-based cryptography. In *2009 International Symposium on Information Engineering and Electronic Commerce*, pages 135–139, Ternopil, Ukraine, May 2009. IEEE.

[181] U. Pesovic, Z. Jovanovic, S. Randjic, and D. Markovic. Benchmarking performance and energy efficiency of microprocessors for wireless sensor network applications. In *2012 Proceedings of the 35th International Convention MIPRO*, pages 743–747, Opatija, Croatia, May 2012. IEEE.

[182] Cristina Pitorac. Estimation of the power losses of a li-ion battery by operation in an electric vehicle. In *2016 International Symposium on Fundamentals of Electrical Engineering (ISFEE)*, pages 1–4, Bucharest, Romania, 2016.

[183] Patrick Plötz, Niklas Jakobsson, and Frances Sprei. On the distribution of individual daily driving distances. *Transportation Research Part B: Methodological*, 101:213–227, July 2017.

[184] S. R. Pokhrel and J. Choi. A decentralized federated learning approach for connected autonomous vehicles. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6, Seoul, South Korea, 2020. IEEE.

[185] S. Pourazarm and C. G. Cassandras. Optimal routing of energy-aware vehicles in transportation networks with inhomogeneous charging nodes. *IEEE Transactions on Intelligent Transportation Systems*, 19(8):2515–2527, 2018.

[186] Xuewei Qi, Guoyuan Wu, Kanok Boriboonsomsin, and Matthew J Barth. Data-driven decomposition analysis and estimation of link-level electric vehicle energy consumption under real-world traffic conditions. *Transportation Research Part D: Transport and Environment*, 64:36–52, October 2018.

[187] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145 – 151, 1999.

[188] Alexander Ratner, Dan Alistarh, Gustavo Alonso, David G. Andersen, Peter Bailis, Sarah Bird, Nicholas Carlini, Bryan Catanzaro, Eric Chung, Bill Dally, et al. MLSys: The New Frontier of Machine Learning Systems. *CoRR*, abs/1904.03257:1–4, 2019.

[189] Mukesh Saini, Kazi Masudul Alam, Haolin Guo, Abdulhameed Alelaiwi, and Abdulmotaleb El Saddik. InCloud: a cloud-based middleware for vehicular infotainment systems. *Multimedia Tools and Applications*, 76(9):11621–11649, January 2016.

[190] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah. Federated learning for ultra-reliable low-latency v2v communications. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, Abu Dhabi, United Arab Emirates, 2018. IEEE.

[191] Stefan Sautermeister, Florian Ott, Moritz Vaillant, and Frank Gauterin. Reducing range estimation uncertainty with a hybrid powertrain model and online parameter estimation. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, Yokohama, Japan, October 2017. IEEE.

[192] Maximilian Schiffer and Grit Walther. The electric location routing problem with time windows and partial recharging. *European Journal of Operational Research*, 260(3):995–1013, August 2017.

[193] Sven Schoenberg and Falko Dressler. Planning ahead for EV: Total travel time optimization for electric vehicles. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3068–3075, Auckland, New Zealand, October 2019. IEEE.

[194] Maximilian Schücking, Patrick Jochem, Wolf Fichtner, Olaf Wollersheim, and Kevin Stella. Charging strategies for economic operations of electric vehicles in commercial applications. *Transportation Research Part D: Transport and Environment*, 51:173 – 189, 2017.

[195] Robert Sedgewick and Jeffrey Scott Vitter. Shortest paths in euclidean graphs. *Algorithmica*, 1(1-4):31–48, 1986.

[196] Mrinank Sharma, Michael Hutchinson, Siddharth Swaroop, Antti Honkela, and Richard E. Turner. Differentially Private Federated Variational Inference. *arXiv e-prints*, abs/1911.10563:1–12, November 2019.

[197] Xiao Shi, Jian Pan, Hewu Wang, and Hua Cai. Battery electric vehicles: What is the minimum range required? *Energy*, 166:352–358, January 2019.

[198] J. E. Siegel, D. C. Erb, and S. E. Sarma. Algorithms and architectures: A case study in when, where and how to connect vehicles. *IEEE Intelligent Transportation Systems Magazine*, 10(1):74–87, Spring 2018.

[199] J. E. Siegel, D. C. Erb, and S. E. Sarma. A survey of the connected vehicle landscape—architectures, enabling technologies, applications, and development areas. *IEEE Transactions on Intelligent Transportation Systems*, 19(8):2391–2406, August 2018.

[200] R. Smith, S. Shahidinejad, D. Blair, and E.L. Bibeau. Characterization of urban commuter driving profiles to optimize battery size in light-duty plug-in electric vehicles. *Transportation Research Part D: Transport and Environment*, 16(3):218–224, May 2011.

[201] Martin Smuts, Brenda Scholtz, and Janet Wesson. A critical review of factors influencing the remaining driving range of electric vehicles. In *2017*

1st International Conference on Next Generation Computing Applications (NextComp), pages 196–201, Mauritius, Mauritius, 2017. IEEE.

[202] Martin Smuts, Brenda Scholtz, and Janet Wesson. Issues in implementing a data integration platform for electric vehicles using the internet of things. In *IFIP Advances in Information and Communication Technology*, pages 160–177. Springer International Publishing, New York, NY, USA, 2019.

[203] Michael R. Sprague, Amir Jalalirad, Marco Scavuzzo, Catalin Capota, Moritz Neun, Lyman Do, and Michael Kopp. Asynchronous federated learning for geospatial applications. In Anna Monreale, Carlos Alzate, Michael Kamp, Yamuna Krishnamurthy, Daniel Paurat, Moamar Sayed-Mouchaweh, Albert Bifet, João Gama, and Rita P. Ribeiro, editors, *ECML PKDD 2018 Workshops*, pages 21–28, Dublin, Ireland, 2019. Springer International Publishing.

[204] Tobias Straub, Mandy Nagy, Maxim Sidorov, Leonardo Tonetto, Michael Frey, and Frank Gauterin. Energetic map data imputation: A machine learning approach. *Energies*, 13(4):982, February 2020.

[205] Shuai Sun, Jun Zhang, Jun Bi, and Yongxing Wang. A machine learning method for predicting driving range of battery electric vehicles. *Journal of Advanced Transportation*, 2019:1–14, January 2019.

[206] Sergej Svorobej, Patricia Takako Endo, Malika Bendechache, Christos Filelis-Papadopoulos, Konstantinos Giannoutakis, George Gravvanis, Dimitrios Tzovaras, James Byrne, and Theo Lynn. Simulating fog and edge computing scenarios: An overview and research challenges. *Future Internet*, 11(3):55–69, February 2019.

[207] John Taggart. Ambient temperature impacts on real-world electric vehicle efficiency amp; range. In *2017 IEEE Transportation Electrification Conference and Expo (ITEC)*, pages 186–190, Chicago, IL, USA, 2017.

[208] Tesla Inc. Introducing V3 Supercharging. `https://www.tesla.com/blog/introducing-v3-supercharging`, March 2019. [Online; accessed 19-July-2020].

[209] Laurent Thibault, Giovanni De Nunzio, and Antonio Sciarretta. A unified approach for electric vehicles range maximization via eco-routing, eco-driving, and energy consumption prediction. *IEEE Transactions on Intelligent Vehicles*, 3(4):463–475, December 2018.

[210] Anna Tomaszewska, Zhengyu Chu, Xuning Feng, Simon O'Kane, Xinhua Liu, Jingyi Chen, Chenzhen Ji, Elizabeth Endler, Ruihe Li, Lishuo Liu, Yalun Li, Siqi Zheng, Sebastian Vetterlein, Ming Gao, Jiuyu Du, Michael Parkes, Minggao Ouyang, Monica Marinescu, Gregory Offer, and Billy Wu. Lithium-ion battery fast charging: A review. *eTransportation*, 1:100011, August 2019.

[211] Chien-Ming Tseng and Chi-Kin Chau. Personalized prediction of vehicle energy consumption based on participatory sensing. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3103–3113, November 2017.

[212] Chien-Ming Tseng, Chi-Kin Chau, Sohan Dsouza, and Erik Wilhelm. A participatory sensing approach for personalized distance-to-empty prediction and green telematics. In *Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems - e-Energy 15*, pages 47–56, Bangalore, India, 2015. ACM Press.

[213] S. Tuli, N. Basumatary, and R. Buyya. Edgelens: Deep learning based object detection in integrated iot, fog and cloud computing environments. In *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, pages 496–502, Mathura, India, 2019. IEEE.

[214] Shreshth Tuli, Nipam Basumatary, Sukhpal Singh Gill, Mohsen Kahani, Rajesh Chand Arya, Gurpreet Singh Wander, and Rajkumar Buyya. HealthFog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated IoT and fog computing environments. *Future Generation Computer Systems*, 104:187–200, March 2020.

[215] R. Valentina, A. Viehl, O. Bringmann, and W. Rosenstiel. Hvac system modeling for range prediction of electric vehicles. In *2014 IEEE Intelligent*

Vehicles Symposium Proceedings, pages 1145–1150, Dearborn, MI, USA, 2014. IEEE.

[216] Andras Varga. Omnet++. In Klaus Wehrle, Mesut Güneş, and James Gross, editors, *Modeling and Tools for Network Simulation*, pages 35–59. Springer Berlin Heidelberg, Berlin, Heidelberg, Germany, 2010.

[217] Kizheppatt Vipin, Shanker Shreejith, Suhaib A. Fahmy, and Arvind Easwaran. Mapping time-critical safety-critical cyber physical systems to hybrid fpgas. In *2014 IEEE International Conference on Cyber-Physical Systems, Networks, and Applications*, pages 31–36, Hong Kong, China, 2014. IEEE.

[218] John Voelcker. Porsche's fast-charge power play: The new, all-electric Taycan will come with a mighty thirst. this charging technology will slake it. *IEEE Spectrum*, 56(09):30–37, September 2019.

[219] Mário P. Véstias, Rui Policarpo Duarte, José T. de Sousa, and Horácio C. Neto. Moving deep learning to the edge. *Algorithms*, 13(5):125, May 2020.

[220] Hao Wang and Dit-Yan Yeung. A survey on Bayesian deep learning. *ACM Comput. Surv.*, 53(5), September 2020.

[221] Lei Wang, Wanling Gao, Kaiyong Yang, and Zihan Jiang. Bops, a new computation-centric metric for datacenter computing. In Wanling Gao, Jianfeng Zhan, Geoffrey Fox, Xiaoyi Lu, and Dan Stanzione, editors, *Benchmarking, Measuring, and Optimizing. Bench 2019*, pages 262–277, Denver, CO, USA, 2020. Springer International Publishing.

[222] P. Wang, N. C. Bouaynaya, L. Mihaylova, J. Wang, Q. Zhang, and R. He. Bayesian neural networks uncertainty quantification with cubature rules. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, Glasgow, Scotland, United Kingdom, 2020. IEEE.

[223] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, April 1996.

[224] Erik Wilhelm, Joshua Siegel, Simon Mayer, Leyna Sadamori, Sohan Dsouza, Chi-Kin Chau, and Sanjay Sarma. Cloudthink: A scalable secure platform for mirroring transportation systems in the cloud. *Transport*, 30(3):320–329, 2015.

[225] D. S. Wilks. Smoothing forecast ensembles with fitted probability distributions. *Quarterly Journal of the Royal Meteorological Society*, 128(586):2821–2836, October 2002.

[226] Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4697–4708, Red Hook, NY, USA, 06–12 Dec 2020. Curran Associates, Inc.

[227] Dan Wu, Bo Liu, Zhijian Chen, Wenyan Xie, Xiang Huang, Shuwei Pei, Bin Sheng, and Donghan Feng. Cloud computing in electric vehicles charging control and dispatch optimization. In *2014 IEEE International Conference on Progress in Informatics and Computing*, pages 597–600, Shanghai, China, 2014. IEEE.

[228] Peng Xiao, Samuel Cheng, Vladimir Stankovic, and Dejan Vukobratovic. Averaging is probably not the optimum way of aggregating parameters in federated learning. *Entropy*, 22(3):314, March 2020.

[229] Fei Xie, Changzheng Liu, Shengyin Li, Zhenhong Lin, and Yongxi Huang. Long-term strategic planning of inter-city fast charging infrastructure for battery electric vehicles. *Transportation Research Part E: Logistics and Transportation Review*, 109:261–276, January 2018.

[230] Wei Xie, Pu Zhang, Rong Chen, and Zhi Zhou. A nonparametric Bayesian framework for short-term wind power probabilistic forecast. *IEEE Transactions on Power Systems*, 34(1):371–379, January 2019.

[231] Muhammad Usman Yaseen, Ashiq Anjum, Mohsen Farid, and Nick Antonopoulos. Cloud-based video analytics using convolutional neural

networks. *Software: Practice and Experience*, 49(4):565–583, September 2018.

[232] Zonggen Yi and Peter H. Bauer. Optimal speed profiles for sustainable driving of electric vehicles. In *2015 IEEE Vehicle Power and Propulsion Conference (VPPC)*, pages 1–6, Montreal, QC, Canada, 2015. IEEE.

[233] Zonggen Yi and Peter H. Bauer. Optimization models for placement of an energy-aware electric vehicle charging infrastructure. *Transportation Research Part E: Logistics and Transportation Review*, 91:227–244, July 2016.

[234] Zonggen Yi and Peter H. Bauer. Adaptive multiresolution energy consumption prediction for electric vehicles. *IEEE Transactions on Vehicular Technology*, 66(11):10515–10525, 2017.

[235] Zonggen Yi and Peter H. Bauer. Effects of environmental factors on electric vehicle energy consumption: a sensitivity analysis. *IET Electrical Systems in Transportation*, 7(1):3–13, March 2017.

[236] Zonggen Yi and Peter H. Bauer. Optimal stochastic eco-routing solutions for electric vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 19(12):3807–3817, December 2018.

[237] Zonggen Yi and Peter H. Bauer. Energy aware driving: Optimal electric vehicle speed profiles for sustainability in transportation. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):1137–1148, 2019.

[238] Z. Younes, L. Boudet, F. Suard, M. Gérard, and R. Rioux. Analysis of the main factors influencing the energy consumption of electric vehicles. In *2013 International Electric Machines Drives Conference*, pages 247–253, Chicago, IL, USA, 2013. IEEE.

[239] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P Jue. All one needs to know about fog computing and related edge computing paradigms: A

complete survey. *Journal of Systems Architecture*, 98:289–330, September 2019.

[240] Tugce Yuksel and Jeremy J. Michalek. Effects of regional temperature on electric vehicle efficiency, range, and emissions in the united states. *Environmental Science & Technology*, 49(6):3974–3980, February 2015.

[241] Anpeng Zhang, Jee Eun Kang, and Changhyun Kwon. Multi-day scenario analysis for battery electric vehicle feasibility assessment and charging infrastructure planning. *Transportation Research Part C: Emerging Technologies*, 111:439–457, February 2020.

[242] X. Zhang, F. Fang, and J. Wang. Probabilistic solar irradiation forecasting based on variational Bayesian inference with secure federated learning. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2020.

[243] Y. Zhang, W. Wang, Y. Kobayashi, and K. Shirai. Remaining driving range estimation of electric vehicle. In *2012 IEEE International Electric Vehicle Conference*, pages 1–7, Greenville, SC, USA, 2012. IEEE.

[244] Zhiyun Zhang, Miaohua Huang, Yupu Chen, and Dong Gao. Big-data based online state of charge estimation and energy consumption prediction for electric vehicles. In *SAE 2016 World Congress and Exhibition*, pages 1–16, Detroit, MI, USA, April 2016. SAE International.

[245] Martin A. Zinkevich, Markus Weimer, Alex Smola, and Lihong Li. Parallelized stochastic gradient descent. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, page 2595–2603, Red Hook, NY, USA, 2010. Curran Associates Inc.