RESEARCH REPORT

A Scalable Machine Learning Algorithm Utilizing Cloud Computing for Demand Forecasting in Smart Grids

submitted to the
Austrian Marshall Plan Foundation

by

Eva-Maria Friedl, BSc

Marshallplan-Jubiläumsstiftung
Austrian Marshall Plan Foundation
Fostering Transatlantic Excellence

USC University of Southern California

FH Salzburg

Supervisor FHS:	FH-Prof. Priv.-Doz. DI Mag. Dr. Dominik Engel
Supervisor USC:	Prof. Viktor K. Prasanna

Affidavit

Herewith I, Eva-Maria Friedl, BSc, born on the 11.06.1990 in Salzburg, declare that I have written the present diploma thesis fully on my own and that I have not used any other sources apart from those given.

Los Angeles and Salzburg, September 2018

1610581006

Eva-Maria Friedl, BSc

Registration Number

Details

First and Last Name:	Eva-Maria Friedl, BSc
University:	Salzburg University of Applied Sciences
Degree Program:	Information Technology & Systems Management
Thesis Title:	A Scalable Machine Learning Algorithm Utilizing Cloud Computing for Demand Forecasting in Smart Grids
Keywords:	SMART GRID, MACHINE LEARNING, DEMAND FORECASTING, LSTM , CLOUD COMPUTING, ELECTRIC VEHICLES
Academic Supervisor:	FH-Prof. Priv.-Doz. DI Mag. Dr. Dominik Engel

Abstract

A large-scale utilization of electric vehicles results in numerous challenges for the Smart Grid. The increased load due to Electric Vehicles (EVs) in certain areas can have a negative impact on the electricity infrastructure. Demand Forecasting of EV load for various locations on a city wide level could provide utility companies with information about where to update their infrastructure. The existing work on EV demand forecasting is mostly limited to a certain location due to specifically created test data from a real world simulation or statistical models. Furthermore, most models developed for EV demand forecasting utilize Autoregressive Integrated Moving Average (ARIMA). However, recent research on direct comparison of ARIMA and Long Short-Term Memory (LSTM) suggests that LSTM models perform better on time series prediction. Therefore, this thesis utilizes a Recurrent Neural Network (RNN) based approach to forecast the EV demand. Also, publicly available datasets are used for the input of the model which leads to significant cost and time reductions. The results show that LSTM is able to make a reasonable forecast. However, it is found that the model is still underfitted and more training data is needed to create an optimal model. Different feature sets showed that a simple model with less features performs best. Tests on different hyperparameter settings revealed that, while parameters like the number of epochs, batch size and the number of neurons showed no significant improvement, an increase in the number of time steps leads to lower error rates.

Table of Contents

List of Abbreviations

ADALINE	Adaptive Linear Element
ANN	Artificial Neural Network
API	Application Programming Interface
ARIMA	Autoregressive Integrated Moving Average
BEV	Battery Electric Vehicle
CEC	Constant Error Carousel
CPU	Central Processing Unit
EV	Electric Vehicle
fARIMA	Fractionally Autoregressive Integrated Moving Average
GPU	Graphics Processing Unit
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
NIST	National Institute of Standards and Technology
NumPy	Numerical Python
PHEV	Plug-in Hybrid Electric Vehicle
PV	Photovoltaic
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SciPy	Scientific Python
SoC	State-of-Charge
TPU	Tensorflow Processing Unit

V2G	Vehicle-to-Grid

List of Figures

List of Tables

Chapter 1

Innovation

The internet transformed the telecommunication sector and provided room for many innovations such as Facebook, Amazon, Paypal – only to name a few. The utility and transportation sector are undergoing a major transformation right now. The traditional electricity grid is evolving into a complex Smart Grid. The transportation sector is moving away from fossil fuels towards electrically driven vehicles. As with other areas like the Internet, new infrastructures - an innovation itself - provide room for many other innovations and bring also many yet still invisible problems, waiting to be solved.

One such problem is the additional load that electric vehicles cause in the grid. The traditional electrical grid was not initially designed for load patterns that differ from regular homes or corporate buildings. When electric vehicle usage rises, also the stress on the electricity grid and its transformers increases, especially when fast charging methods are used that demand an extremely large amount of energy in a very short time. The resulting damages on the grid and possible power outages can cause immense chaos and costs. Therefore, it is of great importance to have predictive models that can estimate the future additional demand due to electric vehicles, ideally before electric vehicles are actually deployed on a larger scale.

There are already existing models that try to forecast the energy demand of electric vehicles. However, most of them use legacy models to do so. The latest technological advances paved the way for powerful machine learning models. Furthermore, the used data in existing research is often generated through costly and time-consuming real-world testbeds. The aim of this work is to generate a model that works by using only publicly available or otherwise easily accessible data. This leads ideally to zero costs. One of the questions for innovation by Peter Thiel is the engineering question. It states that a true innovation should be 10x better than something that is currently available. For example, it costs about £100 million to send a rocket into space and Elon Musk aims to do it for only £10 million – a 10x advantage [1]. Minimizing the costs for electric vehicle demand forecasting to zero by still receiving comparable or better results can be also an improvement on such a scale. The following paragraphs aim to give an overview about the different types and degrees of innovation.

Types of Innovation Innovations can be classified in terms of what is being innovated. Innovation can happen anywhere, whether it is in a profit-oriented organization or a non-profit organization [2]. The following points from [2] provide an overview of the different types of innovation:

- Product innovation: products can be material products or intangible services that meet customer needs. By innovating a product, a company earns money and tries to differentiate itself from competitors.
- Service innovation: there are two types of service innovations, the ones that are sold directly to the customer like insurance or consulting and services that are not actively sold but necessary for a company. An example would be a manufacturing company that still needs to provide services to companies like sales advice, complaints or logistics.
- Business model innovation: A business model innovation – how a company makes works and makes money – encompasses innovations in strategy, marketing, supply chains, value creation, pricing or cost structures.
- Process and technology innovation: Technological innovations can be the creation of a product or service. They can also be process innovations which include production processes or IT technologies for example. Product innovations, quality improvements or cost savings often imply process and technology innovations.
- Others: Organizational innovation, social innovation, environmental innovation

It is noted that an innovation can simultaneously affect more than one of the above-mentioned categories. For example, the Smart Grid can be seen as a technological innovation but also as an environmental innovation. The developed forecasting model in this thesis can be classified as a process or technology innovation.

Degree of Innovation Innovations can also be classified in terms of their novelty or degree of innovation. A widespread notation distinguishes between incremental and radical innovations.

Besides the two extremes also different intermediate forms are used in practise. The following description from [3] helps to classify an innovation into one or the other category:

- Incremental innovations: are an optimization and further development of already existing products, services or processes. The purpose here is to optimize customer experience, costs, achieve a new market position or an adaption to be able to compete in new markets or with new laws and regulations.
- Radical innovations: are new products, services or processes and comprise a fundamental change and novelty. This leads to a higher impact which can lead to completely new markets.

This section aimed to give an overview about the topic of innovation and tried to point out the innovative aspects of this thesis. To summarize it can be said that radical innovations are basic innovations or revolutions and incremental innovations are improvement or adaptive innovations and therefore evolutions. Considering the points above it can be inferred that an innovation does not necessarily have to be something completely new. Also, an improvement of an existing product, service or process can be called an innovation.

Chapter 2

Related Work and Segregation

This chapter aims to give an overview about the findings of the literature research. Existing work in the field of EV demand forecasting is described as well as research that has been done with the allocation of EV charging stations and the generation of test data sets for EVs applications. This leads to a clear distinction of the research topic and existing work, which is given at the end of this chapter, followed by possible use cases.

2.1 Electric Vehicle Demand Forecasting

Statistical approaches The researchers in [4] developed a stochastic model based on queuing theory for consumer charging behaviour of Battery Electric Vehicles (BEVs) and Plug-in Hybrid Electric Vehicles (PHEVs) on a substation or charging station level. They use real PHEVs charging data from [5] to observe questions like: (1) when do vehicles arrive where a charger is available? (2) how often do customers request charging when the vehicle is parked? (3) how much energy is required for each charge event? (4) how much flexibility accompanies each charging event? The model presumes a large population of electric vehicles, theoretically infinite. Queuing theory is used to analyse the effects of customers randomly arriving and being served at a charging station. The outcome of this study provides short-term load forecasting under the condition that real-time sub-metering data is available. This can be used for designing demand response and dynamic pricing schemes. However, the model is not suitable to capture the BEV/PHEV load at a feeder for a few buildings. It also does not include the geographical information meaning the charging demand in a certain area of the grid [4]. ARIMA was also used by the researchers in [6] to develop a method for demand forecasting of conventional electrical load and charging demand of EVs in a parking lot. The model determines the charging load profiles in a parking area. The used parking lot has a capacity of 100 EVs and two cases with different charging rates were modeled. The input to the model are the daily driving patterns and distances. However, a detailed description of the input data is not given. The output of the model is used to formulate a chance-constrained day-ahead scheduling problem to show potential cost saving benefits for the parking lot [6]. The research group in [7] developed a cellphone application algorithm to predict the energy consumption of EV charging stations at the University of Los Angeles. They utilize data from 15 charging stations. They use a k-nearest neighbor based prediction algorithm. Two applications have been developed, one that predicts the expected available energy at the station and the other one predicts the expected finish time of the charging event. The granularity of the prediction is one hour and the prediction horizon is 24 hours [7]. The authors in [8] deal with the modelling and prediction of power loads due to fast charging stations for EVs. They try to simulate the behaviour of a fast charging event by exploiting empirical data to characterize EV user behaviour and obtained a time series with those properties. The utilized a Fractionally Autoregressive Integrated Moving Average (fARIMA) model to generate short-term load forecasts. The researchers in [9] analyse common types of charging modes and their characteristics and effect on the grid. They established a statistical model of charging stations to simulate a regional power grid. This model is used to forecast the daily load curve using Monte Carlo simulation. The output shows the negative effect of uncoordinated charging of EVs.

Parking lot and charging station allocation Related to EV demand forecasting is the allocation of parking lots and/or charging stations, which is also a topic in this thesis. The research group in [10] discusses the allocation of EV parking lots in the distribution systems. The outcome is a two-stage model. First, the behaviour regarding market interactions is optimized to provide profit to the parking lot owner. Second, the parking lot allocation problem is solved considering different network constraints. The parking lot allocation is solely based on an optimization of the distribution system in order to minimize power losses, voltage deviations and ensuring network reliability. The traffic information is not considered in this research [10]. In [11] a mathematical model is proposed to help with the decision of charging station allocation. A game theoretical approach is used to determine the interactions among the availability of EV charging opportunities, destination and route choices as well as the price of electricity.

Generation of test datasets Several works try to create data or a model for electric vehicle usage. Some are bottom-up approaches, like [12] others are top-down such as [13]. The authors in [13] have developed a publicly available test data set for electric vehicle applications. They analysed the trips of 536 GPS-equipped taxi vehicles in San Francisco. This data is then combined with features of different PHEV brands. It consists of information about the vehicle's State-of-Charge (SoC), traces of charging loads at different charging stations, information on SoC and charging deadline when the PHEV is parked at a charging stations and information about the potential of PHEVs for Vehicle-to-Grid (V2G) applications. The monitored data shows three hotspots, namely the airport, the taxi headquarters and downtown. However, this dataset is more suited to show where to place EV charging stations when all taxis would be replaced by PHEVs but not for general trips. For this work NYC taxi data is chosen because it is assumed that in NYC taxis are more frequently used by people on a regular basis instead of taking their own car. So the assumption can be made that NYC taxi trip data reassembles private car trips as taxis have long been a more ubiquitous and integral part of the city's transportation network [14]. In [12] a Markov-chain approach is used to model the behaviour of individuals. The developed activity patterns should represent the time-sequence of activities performed by typical US drivers. The generated driving patterns for different individuals can be used in performing statistical analysis, evaluation and comparison of different vehicles. With the model the total primary energy consumption for personal transportation in the US can be computed. The generated datasets are often used for statistical implementations in demand forecasting.

The objective of the following section is to introduce the topic of time series forecasting. It should give an overview about the topic in general and provide an analysis of existing work. Moreover, the different algorithms should be compared and discussed.

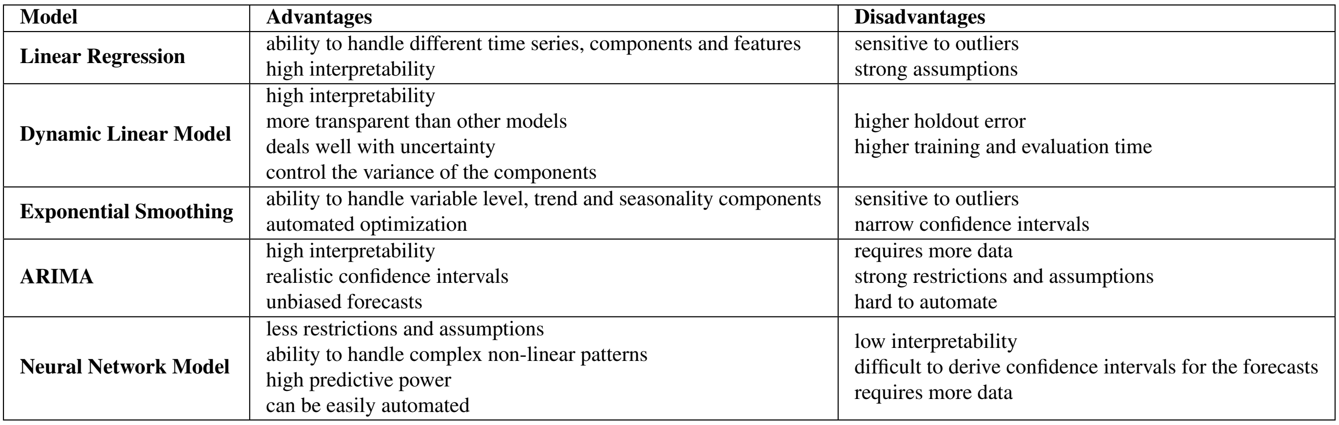
2.2 Time Series Forecasting

An important characteristic of a time series forecasting algorithm is the ability to extrapolate patterns outside of the training data as the objective is to predict and project into the future [15]. Such forecasting methods can be broadly classified into univariate methods, where the forecast depends only on past values of a single time series, and multivariate methods, where forecasts depend, at least partially, on a time series with one or more additional variables [16].

One of the most widely used linear models for time series forecasting is the ARIMA model. It is popular among researchers due to its statistical properties and the well-known Box Jenkins methodology [17]. However, recent research suggests that non-linear models like neural networks can be a promising alternative to classic linear models [17]. Early research on this topic in [18] compared the performance of neural network models to the traditional Box Jenkins methodology, which is used in ARIMA models. The results show that neural networks indeed are a promising alternative approach for time series forecasting. While the Box Jenkins model performs slightly better for short term forecasting, neural networks proved to be better in long term forecasts. Also, the results showed that neural networks appear to be superior to the Box Jenkins model for short term memory series [18]. Neural network and ARIMA models can also be combined, as the researchers in [17] propose, which can result in an improved forecasting accuracy. The hybrid model can be an advantage in cases where it is difficult to determine whether a time series is generated from a linear or non-linear underlying process [17]. With recent advancements in computational power and more advanced machine learning approaches like deep learning, novel algorithms are developed to forecast time series data [19]. As later described in Subsection 3.7.4, RNNs can be an improvement to classical neural networks due to their ability to store contextual information. The researchers in [20] developed a boosting algorithm with RNNs for time series forecasting. Their results performed well, even when long-range dependencies were present in the data. Even better results for time series forecasting can be achieved by using LSTM networks, which are an improved version of RNNs. A recent study in [19] compared the financial time series forecasting ability of ARIMA and LSTM. The results show that deep learning-based algorithms such as LSTM can outperform traditional algorithms like ARIMA.

Comparison of Different Models for Time Series Forecasting As already mentioned above there are different algorithmic approaches for the purpose of time series forecasting, which are summarized in Table 2.1. Despite simple and more complex statistical models there are also neural network models that can be applied to time series [15]. Depending on the underlying data and objective the right model has to be chosen.

Table 2.1 Overview of common algorithms for time series forecasting [15]



Model	Advantages	Disadvantages
Linear Regression	ability to handle different time series, components and features high interpretability	sensitive to outliers strong assumptions
Dynamic Linear Model	high interpretability more transparent than other models deals well with uncertainty control the variance of the components	higher holdout error higher training and evaluation time
Exponential Smoothing	ability to handle variable level, trend and seasonality components automated optimization	sensitive to outliers narrow confidence intervals
ARIMA	high interpretability realistic confidence intervals unbiased forecasts	requires more data strong restrictions and assumptions hard to automate
Neural Network Model	less restrictions and assumptions ability to handle complex non-linear patterns high predictive power can be easily automated	low interpretability difficult to derive confidence intervals for the forecasts requires more data

Linear regression models are used to model simple linear relationships between target and one or more predictors [21]. Dynamic linear models define a general class of non-stationary time series models. The main objectives are short-term forecasting, intervention analysis and monitoring [22]. Exponential smoothing refers to a technique where an exponentially weighted moving average is used to smooth the time series. The new time series is then a smoothed version of the actual time series [23]. As stated in Table 2.1, this technique is sensitive to outliers, which is not suitable for this research.

The most common advanced methods for time series forecasting are ARIMA and neural network models. The ARIMA methodology is well suited for long and stable series. Also, ARIMA models simply approximate historical patterns and are therefore not able to examine the structure of the data [24]. As described in [19], it is difficult for ARIMA models to model non-linear relationships between variables. Furthermore, ARIMA models assume a constant standard deviation in error which may not be satisfied in practice [19]. Also, classical linear methods can be difficult to adapt to multivariate input forecasting problems [25]. That is why ARIMA might not be well suited for this research objective since the input features are assumed to have also non-linear relationships. For example, the same amount of cars in a location might not always lead to the same increase in energy demand. This can depend on various other factors like time of day or other reasons that can lead to an increased value of the total energy. RNNs like LSTMs are well suited for finding patterns in input data when it spans over long sequences. The architecture of LSTMs enables them to manipulate the internal memory state which is ideal for such problems. Furthermore, LSTMs are able to model problems with multiple input variables. This is an advantage in time series forecasting, where classical linear methods can be difficult to adapt to multivariate inputs. The downsides of such networks should also be noted. As listed in Table 2.1, a lot of data is needed when working with neural networks such as LSTMs. Furthermore, multiple hyper-parameters need to be tuned [26].

After the completed literature research on time series forecasting and the comparison of different models LSTM is selected to be the most suitable model for the research objective due to its ability to identify structure and patterns of data such as non-linearity and complexity in time series forecasting. The following points summarize the advantages of LSTM and why it is chosen in this work:

- The ability to handle multivariate inputs.
- The ability to store information over a longer period of time.
- The ability to handle non-linearity and complexity.

The next section should summarize how this work distinguishes from other existing work on electric vehicle demand forecasting. Furthermore, a summary of the defined requirements is given.

2.3 Segregation of Existing Work

To further define the scope of this thesis a few aspects are discussed. A characteristic that distinguishes this work from others is that the electricity forecast should be made as easy and inexpensive as possible. By that it is meant that in order to forecast the electricity demand of electric vehicles no explicit testbed has to be set up beforehand where for example a fleet of vehicles is equipped with GPS sensors for a certain time as in [13]. This is not only time consuming but can also be expensive and might limit the algorithm to a certain geographical area. The purpose of this thesis is not to create data itself but to use existing publicly available data to create a reasonable and up-to-date forecast. Another requirement is that the used algorithm should be able to handle multivariate inputs well. This is because the algorithm should be able to predict the electric vehicle demand from receiving the total electrical load as well as the amount of cars in the selected location as the input. It is expected that every location or future customer has information about their total load which makes this an easy feature for the algorithm. The number of cars in that location can be derived from publicly available trip data or from the customer itself since any building with a parking lot usually is able to monitor entering and leaving vehicles through sensors. Furthermore, the algorithm should not be tied to a specific location or simulated test. It should further be easy to feed new publicly available data to the algorithm in order to forecast the demand in other cities as well by just using the vehicle trip data and total load information. The resulting forecasts would be inexpensive and easily expandable. As a result from the comparison of different models above, a neural network seems to be the best fit for the research objective. Also, to the best of the author's knowledge there has been no research so far on how LSTMs perform for electric vehicle time series forecasting in particular.

The following points summarize the found specifications for the practical implementation and provide a segregation from the existing work described above:

1. The data should be publicly available or otherwise easily accessible.
2. The used algorithm should be able to handle multivariate inputs.

3. The algorithm should be designed to forecast the electric vehicle electricity demand for various geographical locations.
4. The forecast should be performed using a recurrent neural network.

The next section summarizes possible applications that could benefit from the outcome of this thesis. The mentioned use cases reflect the above described specifications and provide specific ideas on how the outcome of this work can be applied in practise.

2.4 Use Cases

Two major applications have been identified that can be improved with the development of an EV demand forecasting algorithm for various geographical locations. On the one hand, the knowledge about the electricity demand that is due to EVs in different geographical locations can be used to optimize the electricity consumption in a city so that the EV charging demand is as balanced as possible. For example, incentives can be given to customers if they drive to a different charging station in another area close by that has currently lower demand. This can decrease stress on distribution transformers and in turn save money to the utility company by increasing the lifetime of the distribution transformers. Furthermore, with the information about the future EV demand utility companies are able to update their infrastructure beforehand to cope with the additional demand in certain areas.

On the other hand, companies who are in locations with high EV demand could deploy Photovoltaic (PV) plus storage solutions. Usually buildings with large parking lots like supermarkets have enough square feet to be able to deploy PV systems plus storage. An option for them would be to charge their storage during the day with solar and/or buy energy during the night when prices are cheaper to load their batteries and generate value by selling this energy to vehicles during the day. The algorithm can help to identify areas in which such a solution would be feasible as well as the right amount of storage to buy. Furthermore, there might be incentives for such solutions in the future in order to facilitate a faster change in the transportation system from fossil fuel to electricity driven vehicles.

Chapter 3

Terms and Definitions

This chapter aims to give a thorough description of concepts and notions relevant for the practical part of this thesis. The following sections form the basic knowledge and the described concepts and definitions will be used repeatedly throughout this work.

3.1 Smart Grid

Historically, the electrical grid has been a few-to-many distribution where a few central power generators provide all of the electricity for a region or a whole country. The electricity is then broadcasted over a large network of transmission lines and transformers. While this legacy system has served well for the duration of about a century, there is a growing need for an update in terms of the aging infrastructure as well as because of new environmental and societal challenges [27]. Emerging from the convergence of energy with telecommunications, transportation, internet and electronic commerce, the Smart Grid can be seen as a new infrastructure for the demand and supply of electricity [28]. Those energy networks are able to automatically monitor energy flows and adjust to changes in supply and demand. As a cyber-physical system, they encompass generation, transmission, distribution as well as monitoring of electrical power [29]. According to the researchers in [30] the Smart Grid brings the following benefits:

- Better situational awareness and operator assistance
- Autonomous control actions to enhance reliability
- Efficiency enhancement by maximizing asset utilization
- Improved resiliency against malicious attacks

- Integration of renewable resources
- Integration of all types of energy storage and other resources
- Two-way communication between the consumer and utility
- Improved market efficiency
- Higher quality of service to power an increasingly digital economy

Smart Grid Architecture Figure 3.1 from [31] visualizes the Smart Grid and its stakeholders. The big picture displays the electricity network (grey line) as well as the IT-infrastructure (green line) which is an enabler for the Smart Grid. The little white arrows from and to the center visualize the two-way power and communication flow. This enables consumers like office buildings and houses, depicted on the top left, to also act as prosumers and supply excess energy generated from renewable energy sources back to the grid. On the top right different energy storage solutions are visualized such as batteries and EVs. Classic energy generators such as coal power plants as well as new distributed renewable energy generators such as PV systems are visualized on the bottom left and right respectively.

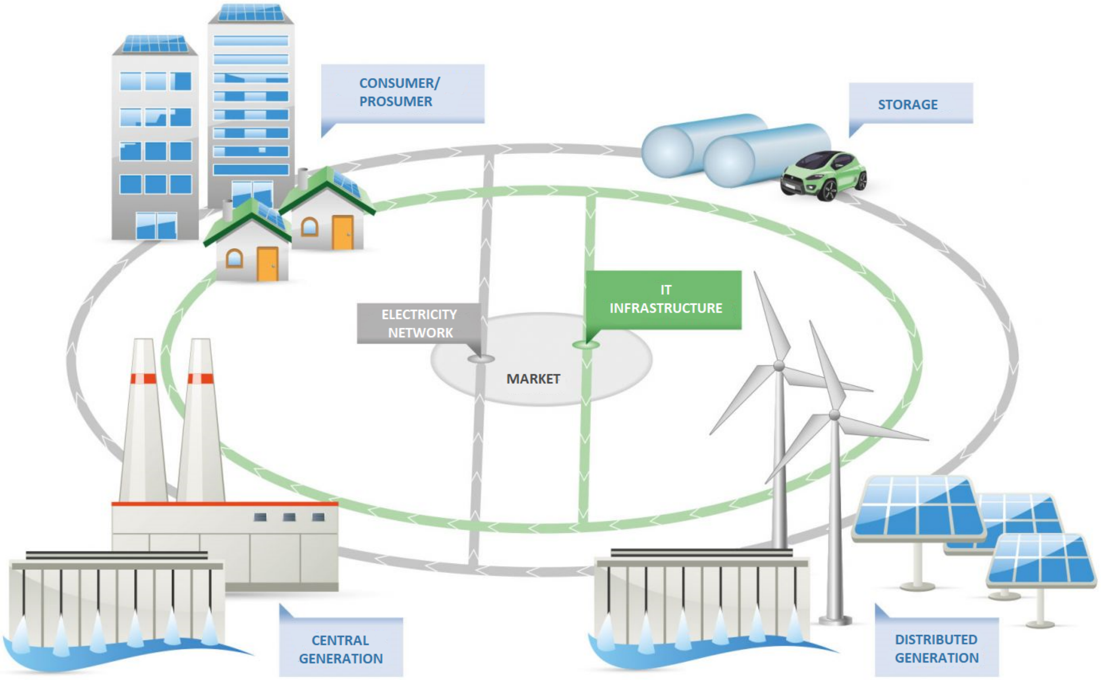


Fig. 3.1 The Smart Grid and its stakeholders [31]

An important role in the realisability of the Smart Grid plays the implementation of Smart Meters which are computational devices that collect energy consumption data and enable the data-flow between consumers and suppliers. The IT infrastructure enables the communication to and from the Smart Meters which are installed at the consumer/prosumer side. Smart Meters provide the utility with information on real-time energy consumption and usage patterns, usually in 15-minute time intervals [32]. Due to the use of those devices, consumers are also able to adapt their energy usage to different energy prices throughout the day. This can be achieved either by shifting a task to a time when energy demand is low (e.g., to turn on the washing machine in the night) or to lower the consumption when demand for energy is high (e.g., to increase the air conditioning temperature on a hot summer day). This is also known as demand response as discussed in [33].

On the utility side, Smart Meters are an enabler for new business models that encourage demand response scenarios over buying new energy power plants. On the one hand, Smart Grids are an enabler to move away from fossil fuels to renewable energy resources. On the other hand, it is a grand challenge to ensure grid stability. This is due to the increasing number of renewable energy resources, customers who might also act as an energy supplier (e.g. through rooftop solar panels) and electric vehicles that need to be supplied with energy as well. The challenge here is to process and analyze this huge amount of information on energy consumption from Smart Meters (petabytes of data) to perform a reliable forecast of the future demand. By combining the gathered data with weather forecasts, grid operators are able to plan the integration of renewable energy systems [32]. Thus, a large amount of energy consumption data needs to be processed in a reliable and fast manner as the prediction of future energy demand plays an important role for the energy management system to ensure grid stability [34].

Smart Grid Trends The recent developments in the power system sector have not only revived the interest in research and development but also resulted in significant socio-economic benefits for communities. People are more aware of the environmental impact of different energy sources which leads to a growth in renewable and alternative energy sources [35]. Power system deregulation also influences the direction of electric power technology. The authors in [30] have identified that the main initiatives for the Smart Grid can be categorized into five trends, namely:

- Reliability
- Renewable resources

- Demand Response
- Electric Storage devices
- Electric transportation

This thesis focuses on the electric transportation sector and its reliability in terms of trying to accurately forecast the electricity demand of EVs. Subsequently demand response scenarios and electric storage solutions can be derived from it as well. Therefore, the next section focuses on demand forecasting in the Smart Grid.

3.2 Demand Forecasting in the Smart Grid

Forecasting the future demand of electrical energy plays a vital role in the feasibility and security of the Smart Grid. It also supports efficient energy management and better planning of the power system infrastructure. A requirement of energy demand forecasts is a high accuracy. Depending on the use case the predictions are done for multiple time horizons that are relevant in the power grid. Usually the time frames for load forecasting range from short-term forecasts (1 hour to 1 day or 1 week ahead) to medium-term forecasts (1 month to 1 year ahead) and long-term forecasts (1 year to 10 years ahead) [34].

An over- or underestimation of the demand can have different effects depending on the time horizon that is forecasted. In long-term forecasts, an overestimation of the demand would lead to higher investments in energy generation and transmission expansion whereas an underestimation could lead to a deficiency in electricity supply. For short-term forecasts overestimates could result in an increased number of committed units to serve load, which means higher system operation cost. Underestimates could on the other hand cause a deficiency in electricity supply, which can have a catastrophic impact on the grid due to real-time imbalances between generation and demand [36].

Achieving a high forecasting accuracy can be difficult since energy markets are facing many different global challenges. According to the researchers in [37] the challenges are:

- Global deregulation
- Distributed energy resources or distributed generation
- Imminent massive irruption of electric vehicles
- Consumer involvement

Especially due to the global deregulation of the power industry, demand forecasting has gained more and more attention in energy market operation and planning. Another challenge mentioned above is the irruption of EVs which has become a highly active research field in the recent past. This is due to the fact that a large-scale usage of EVs has a significant impact on the overall load profile. Usually, demand forecasting is designed for different load patterns induced by the different seasons. However, a large number of EVs in the grid can decrease the accuracy of traditional forecasting algorithms [6].

Demand forecasting plays a crucial role in many fields in the Smart Grid. It applies to several methods such as demand-side management, renewable energy integration, energy storage planning and scheduling. The increasing number of the above mentioned challenges that contribute to demand forecasting requires a revision of present forecasting frameworks for processing data [36]. Different demand forecasting needs exist in the Smart Grid. Traditional demand forecasting of electricity consumption on a household level and overall electrical load is already well established. Overall load forecasting is also strongly related to price forecasting which is in turn utilized for demand response purposes [38]. However, with the emerging number of EVs and the predicted massive increase of electrically powered vehicles in the near future the current research focus lies on predicting the additional load that those vehicles cause in the grid. As mentioned above by the researchers in [6], traditional forecasting algorithms lack in accuracy for this kind of purpose.

Therefore, this thesis focuses on developing an algorithm for EV load forecasting. The next section focuses on electric vehicles. First, the history of EVs is described, followed by the role EVs play in the Smart Grid. Also, the charging infrastructure is discussed and technical details about different EV types and charging levels are given.

3.3 Electric Vehicles

In the beginning of the 19th century the marketplace for motorized transportation was divided into steam-powered, gasoline-powered internal combustion engines and electric vehicles. In the early years, EVs held a competitive share of the market against the popular steam-powered and the faster internal combustion engine competitors. By 1904, one third of all powered vehicles in New York, Chicago and Boston were electric. However, the advent of improved roads and the expansion of the road system in the second decade of the 20th century changed that. Vehicles with a longer range, higher speed as well as the emergence of a refueling infrastructure for gasoline cars led to the dominance of combustion engines [39].

In the late 20th century the world energy crisis brought back the interest to the automobile sector since the transportation sector is one of the top contributors in green house gas emissions globally [40]. Nowadays, the market conditions and technological developments are more favorable for EVs. Battery development has moved EVs forward with lithium-ion batteries with increased storage capacity and less weight [39]. Prices for energy storage and batteries, which used to be very high, are now at an all time low. As shown in Figure 3.2 the prices for lithium-ion batteries have dropped from 1000 USD per kWh to 273 USD per kWh in only six years. This enormous decrease in price is an enabler for more efficient and affordable electric vehicles. Furthermore, battery prices are expected to decrease by more than 30% by 2020 as mentioned in [41].

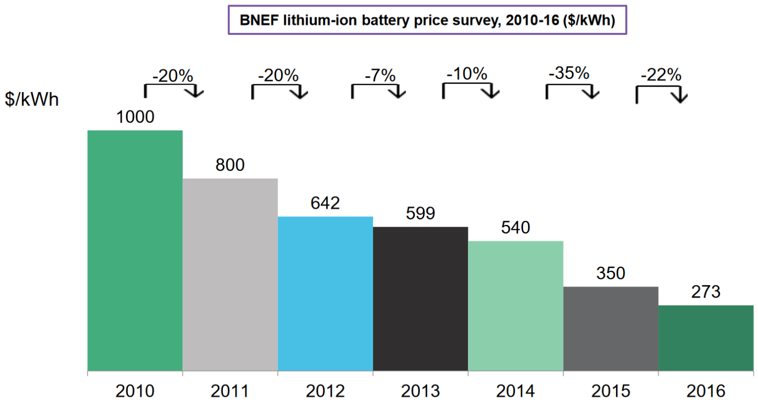


Fig. 3.2 Pricedrop for lithium-ion batteries from 2010 to 2016 [42]

3.3.1 Role of Electric Vehicles in the Smart Grid

The EV is one of the solutions to reduce global green house gas emissions. Besides being a cleaner and quieter alternative, electric cars are also reducing operating costs dramatically. In the United States it costs about $1.10 per eGallon to charge an electric car while the average price for a gallon of gasoline is $2.50 [43]. Furthermore, a lot of time is wasted for stopping at a gas station to fill up the car which adds up over the course of a year. According to the researchers in [40], people could save almost 15 hours per year by owning an EV and charging the car mostly in their home at night. Furthermore, EVs have an integration flexibility advantage that promises a better performance. Energy generators such as fuel cells, solar panels, regenerative braking and any other generators can be integrated into the EV making it even more energy efficient [40].

According to a study initiated from the German government, the most common use of charging for EV owners is their home, which account for 48% of all participants in the study. About 20% use charging at the workplace. Public charging stations are used by 70% of all participants at least once per month. Overall, one third of all charging events is taking place in public, semi-public or on manufacturer owned infrastructure [44]. It is expected due to government incentives and the public awareness of climate change, but also due to the above-mentioned battery technology improvements and therefore longer ranges, that the annual sales of EVs increase by 400% by 2023. This will have substantial effects on the way people use electricity and on the peak demand in the distribution grid [45]. However, EV technology can also provide grid support through peak power shaving, spinning reserve and voltage and frequency regulations when needed [46].

3.3.2 Electric Vehicle Charging Infrastructure

It can be challenging for utilities to develop a charging infrastructure for EVs in the Smart Grid. Especially due to the high costs associated with charging stations. Generally, the authors in [47] distinguish between the following three installation locations:

1. In-home charging: EV charging at home favorably takes place during the night. The additional peak load in the evenings needs to be considered. Certain utility incentives like demand response try to convince customers to charge later when demand is low. Typically the load ranges from 3.7 kW to 11 kW. The advantage of in-home charging is the already existing electricity infrastructure.
2. Workplace charging: During work the car can be charged for a few hours. Through actions like load-shedding any additional peak load during noon can be prevented.
3. Public charging: Typically those areas include shopping and leisure centers, cinemas and other locations just like where regular gas stations are located. The charging time is assumed to be short in those areas especially on streets or highways where people are expected to demand only fast charging so they can continue their trip as fast as possible. There is a high capital expenditure associated with those charging stations due to the higher power that is required, the needed battery cooling and the external loading unit for the battery.

3.3.3 Electric Vehicle Types

There are two different types of EVs available, PHEVs and BEVs. While the first one uses both gasoline and electrical power the latter is pure battery electric [48]. PHEVs are a solution to replace some part of the energy used for transportation with electricity until full electrification of vehicles becomes mature. They can further eliminate concerns about the charging time of EVs and their limited range. Full-electric vehicles on the other hand convert electrical energy that is stored in a battery into mechanical energy used to move the vehicle. Pure EVs are associated with a limited range - around 100 miles and long recharging times such as up to 20 hours using a standard home outlet [49]. Table 3.1, adapted from [48], gives an overview about their differences and characteristics.

Table 3.1 Characteristics of electric vehicle types [48]

Characteristics	**BEV**	**PHEV**
Propulsion	Electric motor/battery only	Electric motor / battery plus gasoline engine
Refueling	Electricity	Electricity OR gasoline
Range	70 - 100 miles	15 - 35 miles electric and 300+ in gasoline-electric hybrid mode
Charging Time	about 4-6 hours with 220V	about 1 hour with 220V and 3 hours with 120V
Charger Type	L1: 120V, L2: 220V	L1 usually preferred since there is no cost for charging equipment and time to charge is minimal
Charging Cost	about $1 per gallon equivalent	depending on ratio of electric to gasoline miles driven
Battery	Lithium-ion battery between 24kWh-36kWh	Lithium-ion battery between 5-12kWh
Emissions	zero emissions	low emissions but depending upon electric to gasoline ratio

3.3.4 Electric Vehicle Charging Levels

There are several charging levels for EVs that affect the charging duration. The loading time of the battery depends on the power used for charging. Table 3.2 visualizes the different levels. The AC Level 1 charging is the standard home charging and is practically realized whereas AC Level 2 is suitable for public and commercial areas like work, shopping malls or restaurants. EVs can be charged with AC Level 1 and AC Level 2 using the on-board charger of the EV. For DC Level 1 and 2 charging an off-board charger is needed. Furthermore, for DC-fast charging an infrastructure with high power capability is needed. Due to the time efficiency this type of

EV charging will be the most prominent one in the next decade. EV charging stations can then be used equivalently as gasoline charging stations today in terms of the time needed to recharge or refuel the vehicle [46].

Table 3.2 Charging levels for electric vehicles per SAE J1772 standard [46]

Power level type	**Voltage level [V]**	**Current Capacity [A]**	**Power Capacity [kW]**	**Charging duration**
AC Level 1	120	16	1.9	BEV: 17h PHEV: 7h
AC Level 2	240	up to 80	19.2	BEV: 7h (3.3kW charger) PHEV: 3h (3.3kW charger) BEV: 3.5h (7kW charger) PEV: 1.5h (7kW charger)
DC Level 1	200-500	< 80	up to 40	BEV: 1.2h PHEV: 22min
DC Level 2	200-500	< 200	up to 100	BEV: 20min PHEV: 10min

In order to enable a stable smart grid with a functioning infrastructure for electric vehicles an underlying working IT infrastructure is needed. In order to adapt to the increasing demand of consumption data in the smart grid cloud computing is needed to guarantee accurate forecasts in a short time. Therefore, the next section discussed the topic of cloud computing and how it can benefit the smart grid.

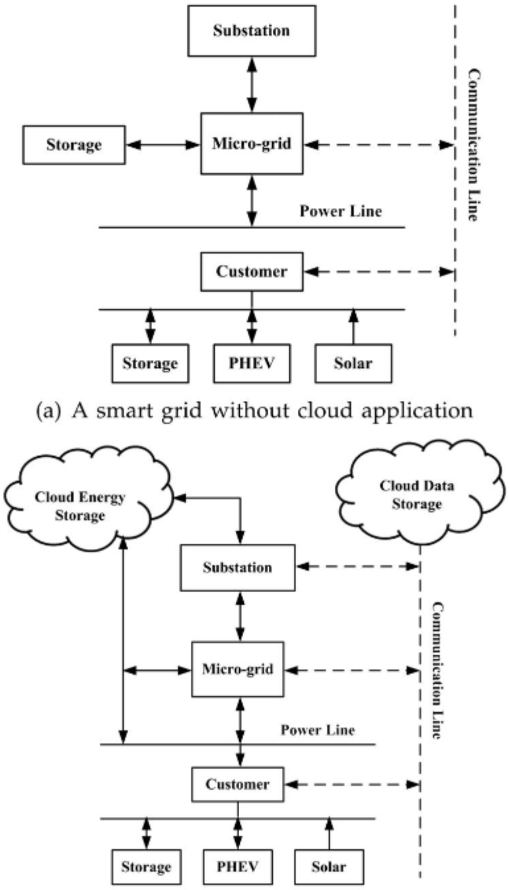
3.4 Cloud Computing

Cloud Computing is defined by the National Institute of Standards and Technology (NIST) as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [50]. A key benefit of implementing cloud computing is that computing resources can be allocated on demand. As with the roll-out of smart meters in the next years, a huge increase in processing data is expected this can be highly effective. With the ever-increasing number of devices and resources connected to the grid such as electric vehicles and renewable energy resources, cloud computing can be a highly beneficial tool for energy providers to not have to deal with expanding and administrating a data center to be able to process power information at a high speed. For economic reasons, it is also profitable as this is a pay-for-use scenario.

3.4.1 Cloud Computing in the Smart Grid

The researchers in [51] identified several problems with existing smart grid applications without utilizing cloud computing. Figure 3.3 visualizes a conventional smart grid without implemented cloud and a smart grid where cloud computing is integrated into the architecture. In the conventional solution customers are served by micro-grids which have self-generating units like solar and wind. The communication in this solution is done via the communication network. On the bottom half of the figure a smart grid architecture with integrated cloud solution is presented where cloud applications can be provided as virtual energy storage and data storage devices. In this case the communication flow is via the cloud instead of direct communication between the components which enables taking decisions for energy management [51]. The problems in a conventional solutions are the following as identified in [51]:

- Extensive exposure to cyber attacks
- Single failure in master-slave achitecture
- Limited capacity to serve customers
- Management of micro-grids
- Difficult real-time implementation



(a) A smart grid without cloud application

(b) A smart grid with cloud application

Fig. 3.3 Conventional smart grid without cloud (a) and with cloud (b) [51]

The above mentioned issues describe why a conventional solution might not be adequate for the modern smart grid. This is especially the case when an increasing number of electric vehicles and battery storage solutions are deployed. To guarantee a safe and reliable smart grid with a functioning infrastructure for electric vehicles an adequate infrastructure is inevitable.

3.4.2 Cloud Computing and Data Analytics

Cloud computing is especially important in domains where a lot of data needs to be processed such as in the smart grid. The researchers in [52] discuss the relationship between "big data" and cloud computing. To analyse large amounts of data typically distributed queries across multiple dataset are performed which return resultant sets in a short time. Cloud computing can provide the underlying engine through the use of hadoop which is a type of distributed data-processing platform. Figure 3.4 visualizes the use of cloud computing in big data applications. As depicted, large data sources from the cloud and internet are stored in a distributed database. The data is processed by a programming model for large datasets with a parallel algorithm. The main purpose of visualization of data is to get insights into analytical results by presenting them visually through different graphs which help in the decision making process [52].

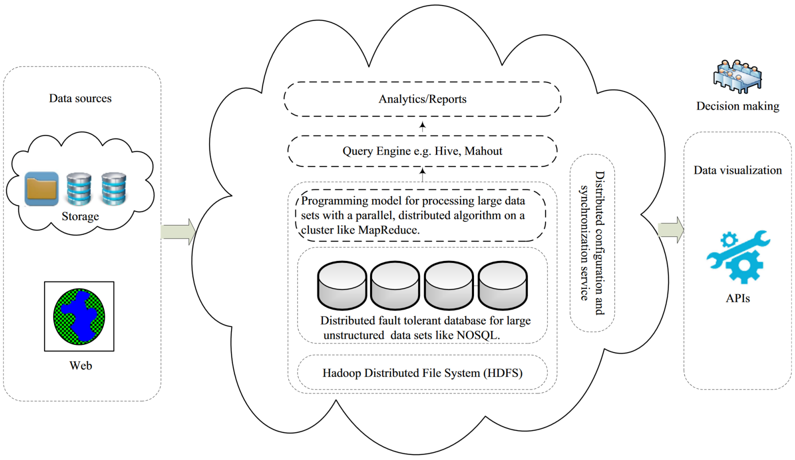
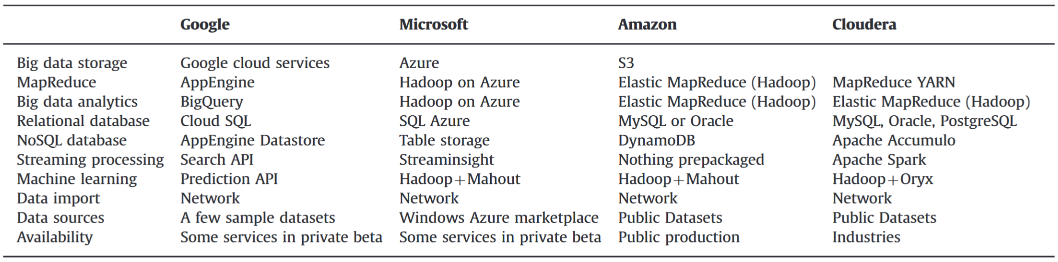

Data sources
Storage
Web
Analytics/Reports
Query Engine e.g. Hive, Mahout
Programming model for processing large data sets with a parallel, distributed algorithm on a cluster like MapReduce.
Distributed fault tolerant database for large unstructured data sets like NOSQL.
Hadoop Distributed File System (HDFS)
Distributed configuration and synchronization service
Decision making
Data visualization
APIs

Fig. 3.4 Cloud computing usage in data analytics) [52]

3.4.3 Cloud Computing Solutions

Cloud computing has many advantages compared to conventional IT infrastructures as discussed in [53]. First, there are the huge cost reductions by eliminating the expense to buy hard- and software. Furthermore, also the costs in setting up racks of servers and maintaining them can be eliminated. Also, cloud computing services provide service on demand, so even vast amount of resources can be accessed and used in minutes giving businesses a lot of flexibility and eliminating the need for capacity planning. Cloud Computing also enables to scale globally meaning to deliver IT resources when it is needed from the needed geographic location. In terms of productivity cloud solutions enables IT teams to spend more time on achieving more important business goals than setting up and maintaining a server infrastructure. Last, cloud computing offers performance improvements and increased security in helping to protect infrastructure from potential threats [53].

There are different types of cloud deployments, namely public, private and hybrid systems. Public clouds are owned and managed by a third party service providers which provide computing power and resources like servers and storage via the internet. A private cloud refers to cloud computing resources which are owned exclusively by a single business or organisation. It can be for example located in the company's own datacenter. A hybrid cloud solution provides a combination of public and private clouds [53]. The most common and easiest form of cloud computing are public clouds because they completely eliminate acqusition and maintenance costs as already discussed above. Figure 3.5 from [52] depicts and compares different public big data cloud platforms. The big players in this field are Google, Microsoft, Amazon and Cloudera. The figure visualizes the different techniques they use for database, machine learning, data storage and other important cloud services. The table should provide an orientation for different use cases and support the decision making process.



	Google	Microsoft	Amazon	Cloudera
Big data storage	Google cloud services	Azure	S3	
MapReduce	AppEngine	Hadoop on Azure	Elastic MapReduce (Hadoop)	MapReduce YARN
Big data analytics	BigQuery	Hadoop on Azure	Elastic MapReduce (Hadoop)	Elastic MapReduce (Hadoop)
Relational database	Cloud SQL	SQL Azure	MySQL or Oracle	MySQL, Oracle, PostgreSQL
NoSQL database	AppEngine Datastore	Table storage	DynamoDB	Apache Accumulo
Streaming processing	Search API	Streaminsight	Nothing prepackaged	Apache Spark
Machine learning	Prediction API	Hadoop+Mahout	Hadoop+Mahout	Hadoop+Oryx
Data import	Network	Network	Network	Network
Data sources	A few sample datasets	Windows Azure marketplace	Public Datasets	Public Datasets
Availability	Some services in private beta	Some services in private beta	Public production	Industries

Fig. 3.5 Overview about different public cloud solutions) [52]

In order to create a more reliable infrastructure for EV charging as well as a faster adoption of EVs proper planning is necessary. Existing data can be used for techniques like machine learning to provide utilities with a forecast about the additional electricity demand of electric vehicles. Therefore, the next section describes the topic of data analytics.

3.5 Data Analytics

Nowadays, organizations collect massive amounts of data. The applications of data mining and statistics cover a wide spectrum. In almost any industry whether it is research, economics, prevention of terrorists, agriculture, entertainment or others the anaylsis of data can be utilized to make better decisions. The most relevant fields are those where a large amount of data needs to be analysed, sometimes with the aim of fast decision making [54]. Machine learning and other techniques can be applied to generate insights based on the used data [55]. The graphic below adapted from [55] illustrates this process.

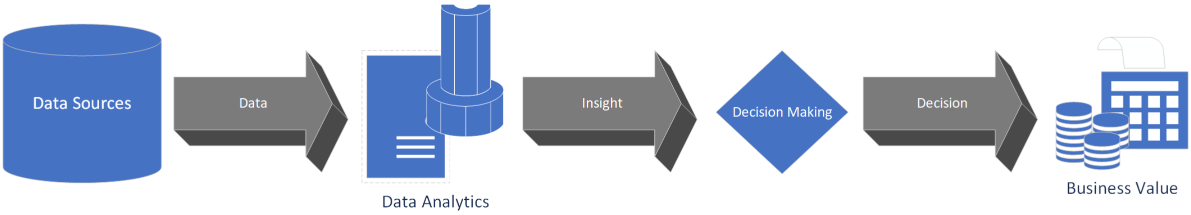


Fig. 3.6 The process of data analytics

The use of this data – in particular personal data – for data mining can have serious ethical implications. When applied to people, data mining is often used to discriminate. For example – who gets the loan, who gets the special offer and so on. Also, surprising things can emerge from data mining. For example, researchers found that people with red cars are more likely to default on their car loans. However, data mining is just a tool in the whole process. The people who take the result, along with other knowledge, are in the end deciding which actions to apply [56]. When looking at the topic of this research, it is up to the utility what to do with this information. They can use the results to save costs and place their charging stations in those areas to optimize the grid. However, they could also use the predicted EV consumption data to increase prices in areas where the demand is predicted to be higher at a certain time.

Machine learning is a powerful tool to create insights from huge amounts of data. It plays a vital role in this thesis as it is used to forecast the future electricity demand of electric vehicles based on historical data. Therefore, the next section gives a more detailed overview about machine learning and how it can be applied to datasets.

3.6 Machine Learning

Machine Learning is a subfield of Artificial Intelligence concerned with intelligent systems that are able to learn. Often, learning is viewed as the most fundamental aspect of intelligence since it enables an agent to be independent of its teacher [57]. Advances in computer systems paved the way for this technology to be applied practically and it serves as a powerful tool to program computers to optimize a performance criterion or make accurate predictions using historical data. It enables a computer to solve problems that are not explicitly programmable but can be solved by utilizing example data or past experience [58]. Figure 3.7 from [59] visualizes the difference between classical programming and machine learning. In classical programming, humans input rules in form of a program and data to be processed according to the rules. The output are answers. In machine learning humans input data as well as the answers expected from this data (in a supervised setting) and the outcomes are the rules. Those rules can be used to apply to unseen data and predict future values [59].

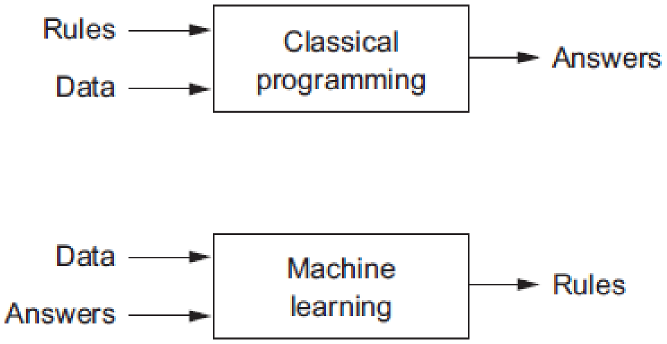
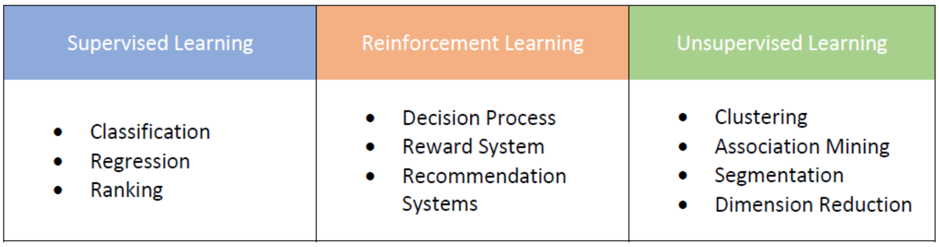


Fig. 3.7 Classical programming and machine learning [59]

The entire process of learning requires the following datasets:

1. Training set: The data set that is used to teach the machine learning algorithm and fit the model [60].
2. Testing set: The test set is used to evaluate the performance of the model on previously unseen data. It is important that the model is tested with a different data set than that used for training in order to avoid a biased score [61].

Depending on the problem and the available data, different learning paradigms of machine learning exist. One type of machine learning models are supervised learning algorithms that enable the computer to learn from a comparably small training set and perform the prediction objective on test data. Then there is reinforcement learning where the feedback during training is not the answer itself but whether it is correct or not. Last, in an unsupervised setting there is no training data at all [62]. A more detailed description about the different machine learning paradigms is given in Subsection 3.6.1, 3.6.2 and 3.6.3. Figure 3.8, adapted from [60], summarizes the three types of learning paradigms and the related problems they are typically used for.



Supervised Learning	Reinforcement Learning	Unsupervised Learning
• Classification • Regression • Ranking	• Decision Process • Reward System • Recommendation Systems	• Clustering • Association Mining • Segmentation • Dimension Reduction

Fig. 3.8 Types of machine learning and related objectives

3.6.1 Supervised Learning

Supervised learning is commonly used to make predictions about the future [63]. In a supervised setting, the computer is provided with a labeled training set which consists of the input variables and their corresponding correct output value [64]. The authors in [65] refer to it as learning with a teacher. Figure 3.9 visualizes the main goal in supervised learning which is to create a predictive model from labelled training data. The algorithm receives the training data along with the correct labels. The output is a predictive model that is able to take new data as the input and predict the output values [63].

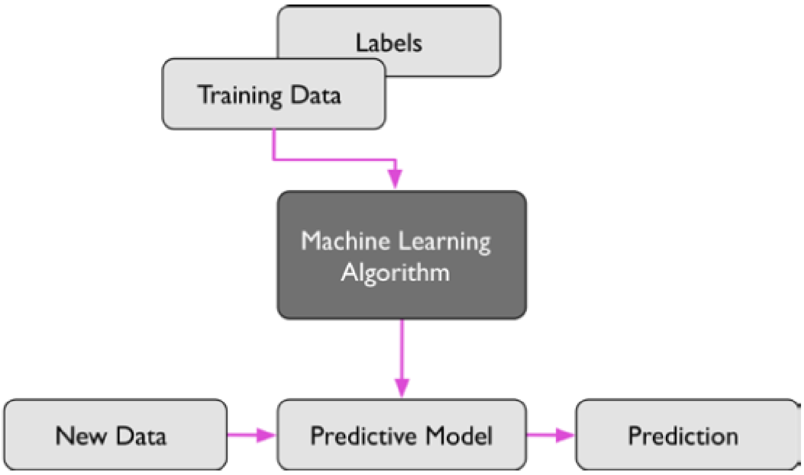


Fig. 3.9 Visualization of supervised learning [63]

Most of the supervised learning algorithms have one thing in common: the training is performed by minimizing a particular loss or cost function which represents the output error provided by the system with respect to the desired output which is known from the training set. The system then changes the weights to minimize this error function. How good the model performs is evaluated with the test set by evaluating the percentage of correctly classified examples and the percentage of misclassified examples [60]. Typically, the following learning types are used in supervised learning algorithms [63]:

- Classification: In classification learning the goal is to look at a data-point with its features and assign it to a certain class. For example, if there are two classes "play" and "don't play" and depending on weather features like wind, humidity and temperature it is assigned one of the above mentioned classes. The result is a category [56].
- Regression: In numeric prediction or regression problems the attributes and the outcome are numeric. For example, if one tries to determine the Central Processing Unit (CPU) performance from a data-point with attributes like cycle time, main memory and cache, a regression equation can be used o determine the outcome [56].

3.6.2 Reinforcement Learning

Reinforcement learning is an approach that affirms the learning of the system through environment interactions. The system parameters in the learning phase are adapted based on feedback received from the environment. The feedback on the made decision is binary in the form of correct or incorrect. For example, a system modelling a chess player using the result of a step to improve the performance is a system that learns with reinforcement [60]. The authors in [65] refer to reinforcement learning as a special case of supervised learning since it can be also considered as learning with a teacher where the only feedback is whether an output is correct or incorrect but not what the correct answer is.

3.6.3 Unsupervised Learning

In unsupervised learning the learning goal is not defined in terms of specific correct examples. The input data is either unlabeled or has an unknown structure [63]. During the training phase a set of inputs is presented to the system however not labeled with the related belonging class. This type of learning is in that sense important because it is probably far more common in the human brain than supervised learning [60]. The only available information to the algorithm lies in the correlations of the input data[65]. The goal is to explore the structure of the data to extract useful information without the guidance of a teacher using unsupervised learning techniques [63].

Same as in a supervised setting their certain learning types associated with unsupervised learning, namely [63]:

- Clustering: In clustering, the goal is to find groups of data that belong together and understand and show their relationships [56].
- Dimensionality Reduction

3.6.4 Terminology

Below a short description of important terminology used in machine learning adapted from [66] unless stated otherwise:

- Instance: Individual, independent example of the concept to be learned [56].
- Examples: Instances of data used for learning or evaluation. The collection of time series containing the number of cars, the load and the total load would be an example.
- Features: The set of attributes, often represented as a vector. For example, the time series containing the number of cars at a particular location is a used feature in this work.
- Labels: The corresponding value assigned to an item of an example. In regression, each item is assigned a real-valued label.
- Training sample: Examples used to train the machine learning algorithm. Usually, 70% of the training data is used for training and 30% for testing.
- Test sample: Examples used to evaluate the performance of the machine learning algorithm. This test dataset is separated from the training and validation set and is not presented to the algorithm during learning. In this case the test sample would be composed of the same columns as the training sample except that the column which should be predicted is removed.
- Loss function: A function that measures the difference between a predicted label and the ground truth.

3.6.5 Applications

A use case that can be solved with machine learning can be for example a problem that changes over time or depends on a special environment. A preferred solution would be to have a general system that can adapt to different circumstances as opposed to writing a program for each individual case. An example described in [58] considers routing packets over a computer network. The path to maximize the quality of service changes at every instance because the network traffic changes over time [58]. So can the electricity consumption of electric vehicles in the Smart Grid. In particular, the electricity demand of non-stationary objects such as EVs changes during the time of the day and can be different from one location to another. The electricity demand might also be different with higher electric vehicle adoption rates and additional smart electronic devices that need to be charged or constantly supplied with electricity. The machine learning algorithm can be used to forecast the electricity demand by optimizing a function such that for each data point (e.g., historical values of electric load, weather or other data types) the prediction is as close as possible to the corresponding target value from the training data set [62].

3.6.6 Overfitting and Underfitting

A common problem in machine learning is overfitting, in simple terms this means that the algorithm is learning the data by heart. On the contrary, underfitting occurs at the beginning of the training process where the network has not learned all relevant patterns yet. In order to prevent the model to learn misleading or irrelevant patterns found in the training data is is important to have a large enough training data set. There is not a general number how big the dataset has to be to get a good model because it depends on several factors like the complexity of the machine learning algorithm as well as on the data and the amount of features for example. A model that is trained on more data will be able to generalize better [59]. Figure 3.10 visualizes overfitting, optimum fit and underfitting as explained above.

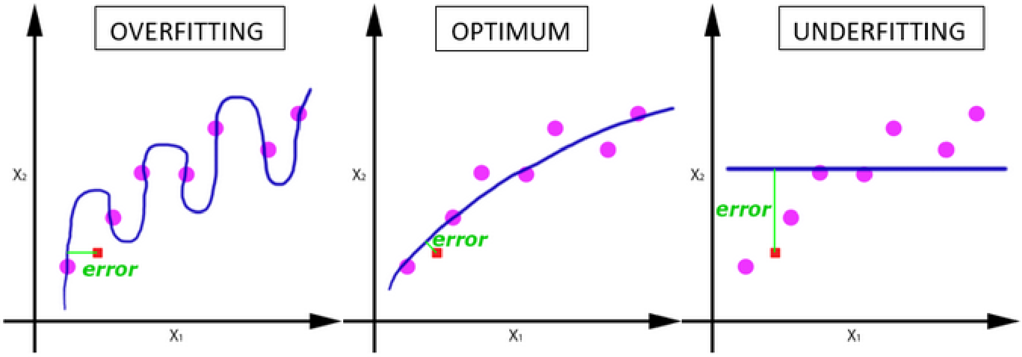


Fig. 3.10 Visualization of overfitting, optimum fit and underfitting [67]

The underlying algorithms for machine learning are often neural networks which can be used to solve complex problems. As this thesis deals with complex time series forecasting the use of neural networks is necessary. Therefore, the next section describes this topic in detail.

3.7 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a group of algorithms used for machine learning that model the data using artificial neurons. Neural network models are inspired from the cell structures in the human brain with the aim to build useful machines. An ANN is made out of millions of simple cells called neurons [58]. A simple cell can have a random number of inputs and one output [68]. The most simple form of an ANN – a single layer network with a threshold activation – is called a perceptron [69]. It is referred to as the basic processing element and is mostly used in theoretical applications of neural networks [70]. An example of a single layer perceptron is visualized in Figure 3.11 from [68].

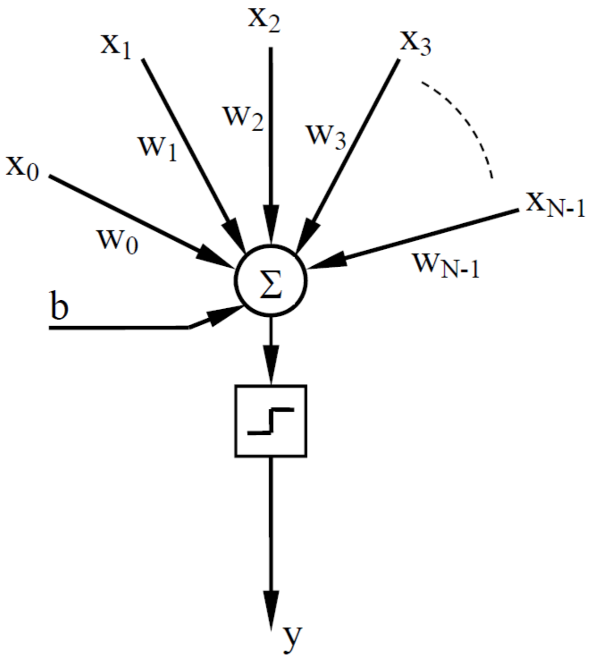


Fig. 3.11 Concept of a single layer perceptron [68]

The inputs denoted as x_n come from the environment or may be the outputs of other perceptrons. Associated with each input is a connection weight denoted by w_n [58]. Weights are also described in [68] as the strength of the ties between the cells, or the strength of how the input affects the cell. The input value is multiplied with its corresponding weight and is then summed up at the node. The weights are typically chosen randomly at the beginning and are then optimized during the learning process [68]. The result of the simple network is the output of the threshold activation function [69]. A simple perceptron as depicted in 3.11 is able to classify input patterns in two classes (output is 0 or 1). In that case a threshold value of 0 is used to determine the output y to be 0 or 1 as shown in Equation 3.1 [68].

$$y = \begin{cases} 1 \ if \ \sum_{i=0}^{N-1} w_i x_i + b \geq 0 \\ 0 \ if \ \sum_{i=0}^{N-1} w_i x_i + b < 0 \end{cases} \tag{3.1}$$

3.7.1 Activation Functions

The activation function in the in 3.11 depicted case that leads to this binary output is a step function. One version of the perceptron is the so called Adaptive Linear Element (ADALINE). The only difference to the perceptron is the output function which is linear in this case. The learning process differs from the perceptron in terms of the correction of the weights. They are not only corrected in terms of correct or wrong but also in considering the size of the error. This leads to an output that can have all values instead of only 0 or 1 [68]. This behaviour is usually not desired in neural networks since it doesn't help with the complexity [71]. In general, the activation function that leads to a certain output value depends on the use case [70]. However, in order to limit the effect of seldom large values on the learning process, the output of a neural network unit is commonly limited through a non-linear function [68]. The non-linearity helps the model to generalize or adapt with the variety of the data and to differentiate between the outputs [71].

Figures 3.12 and 3.13 visualize commonly used activation functions. Practically, activation functions behave similar to the perceptron activation function for outliers – meaning for very small or large values – but have a smooth transition between the two binary numbers. The standard sigmoidal function (Equation 3.2) depicted in Figure 3.12 is commonly used for this type [70]. Furthermore, it is especially useful for models that have to predict a probability as an output because the function exists between 0 and 1 [71].

$$S(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

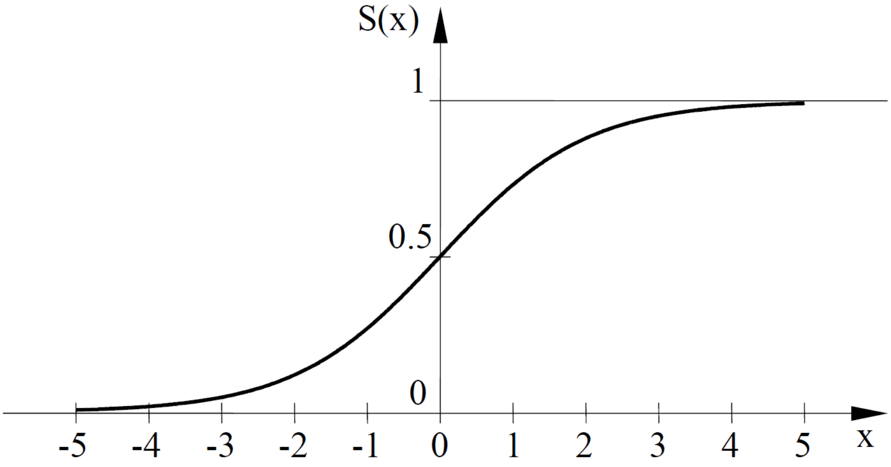


Fig. 3.12 Logarithmic sigmoid activation function

The hyperbolic tangent function (Equation 3.3), depicted in Figure 3.13, has the advantage that negative inputs will be mapped negative and that zero inputs will be also close to zero in the output. It is mainly used for classification between two classes.

$$S(x) = \frac{2}{1+e^{-2x}} - 1 \tag{3.3}$$

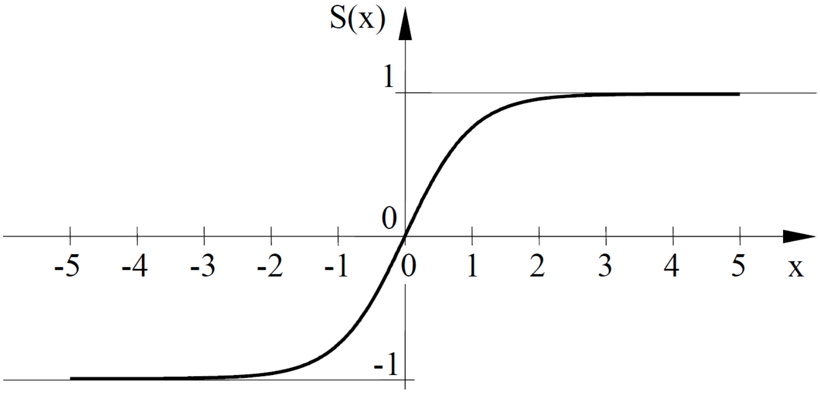


Fig. 3.13 Hyperbolic tangent sigmoid activation function

3.7.2 Neural Network Architecture

A neural network consists of at least three layers: 1.) an input layer, 2.) hidden layers and 3.) an output layer [19]. A typical neural network structure with multiple layers is shown in Figure 3.14 from [68] which consists of several input units one hidden layer and an output layer [72]. The number of inputs to a network depends on the system. Generally speaking it should be avoided to have too many inputs because the complexity of the network increases with the number of input features which leads to a more difficult learning phase. The number of output nodes depends on the number of classes the inputs should be classified [68].

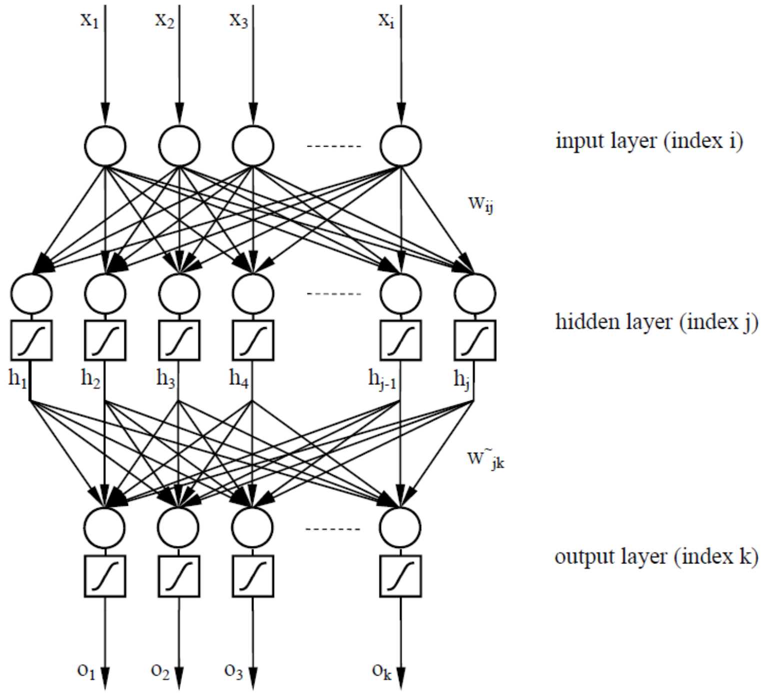


Fig. 3.14 Neural network architecture [68]

The fundamental data structure in neural network applications is the layer. It can be seen as the data-processing module. It takes as input one or more tensors which leads to an output of one or more tensors. Some layers can be stateless but typically they have a state. The weights of a layer learned with stochastic gradient descent contain the knowledge of the network [59]. Different layers exist for different tensor formats and different types of data processing [59]:

- Densely connected layers: for simple vector data stored in 2D tensors of shape (samples, features)

- Recurrent layers: for sequence data stored in 3D tensors of shape (samples, time steps, features), typically processed by recurrent layers such as LSTM
- Convolutional layers: for image data stored in 4D tensors

3.7.3 Learning Process

The fundamental trick in the learning process of a neural network is to use the error score as a feedback signal to adjust the value of the weights in each iteration a little bit in the right direction so the error score becomes smaller. This process is visualized in Figure 3.15. The weight adjustment is performed by an optimizer, which implements the so called backpropagation algorithm [59]. In the beginning the weights are initialized randomly. Then the input is put on the network and the output is determined [68]. If the output is not the desired one the weights are corrected as mentioned above. The correction of the weights works because the output functions are deriverable which gives the algorithm the direction in which the weight needs to be corrected which is also referred to as gradient descent[71]. In a whole network there are different weights on every layer. Usually the weights of the output layer are corrected first and then those of the hidden layer above [68].

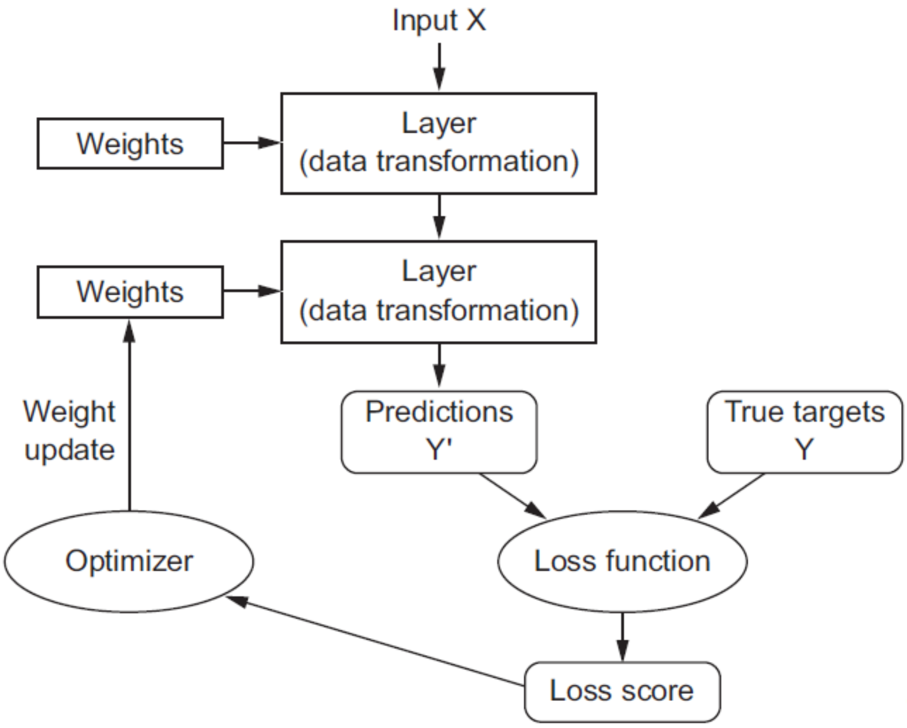


Fig. 3.15 Neural network learning process [59]

The process of correcting the weights is called optimization. The goal is to adjust the weights in such a way to get the best possible performance on the training data. The goal is to have a model that performs well on new unseen data without learning the data by heart meaning the model can generalize well [59]. Different optimizers can be used to correct the weights. The selection of an opitimization algorithm for a machine learning model can mean a difference between good results in minutes, hours or days [73]. Typically, stochastic gradient descend algorithms are used for this purpose [74]. The stochastic gradient descent has a broad adoption for deep learning applications in computer vision and natural language processing [73]. In the gradient descent approach the weights are initialized randomly [75]. The estimates are then iteratively updated with the formula shown in Equation 3.4 from [75] where α_k is the stepsize parameter and w are the weights at different iterations k. The weights are updated based on the choice of the step size sequence α_k.

$$w_{k+1} \leftarrow w_k - \alpha_k \nabla F(w_k)) \tag{3.4}$$

Computationally, the gradient descent algorithm can be expensive due to the fact that each computation of F requires a pass over the entire dataset which is especially the case when the dataset is large. The step size is also known as the learning rate and is fixed and sufficiently small. The stochastic gradient method is a variation to solve stochastic equation systems with the advantage that the per-iteration cost is less than with gradient descent. This is due to the fact that in each iteration of the stochastic gradient method an unbiased estimator of the true gradient is computed. This estimator can be computed at a low cost [75]. The stochastic gradient-based optimization algorithm is of great importance in science and engineering [76].

Many problems can be seen as an optimization of some scalar parametrized optimization function requiring maximization or minimization regarding its parameters [76]. The Adam optimizer is an extension to the stochastic gradient descent [73]. It is a method for efficient stochastic optimiztation that only needs first-order gradients with little memory requirement [76]. The developers of Adam describe it as a combination between two other extensions of stochastic gradient decent namely, Adaptive Gradient Algorithm and Root Mean Square Propagation [73]. Figure 3.16 from [76] shows the performance of the Adam algorithm compared to other optimizers. It can be seen that the training cost is significantly less, especially for a large number of iterations. For this reason, the Adam optimizer is used in the practical part of this thesis.

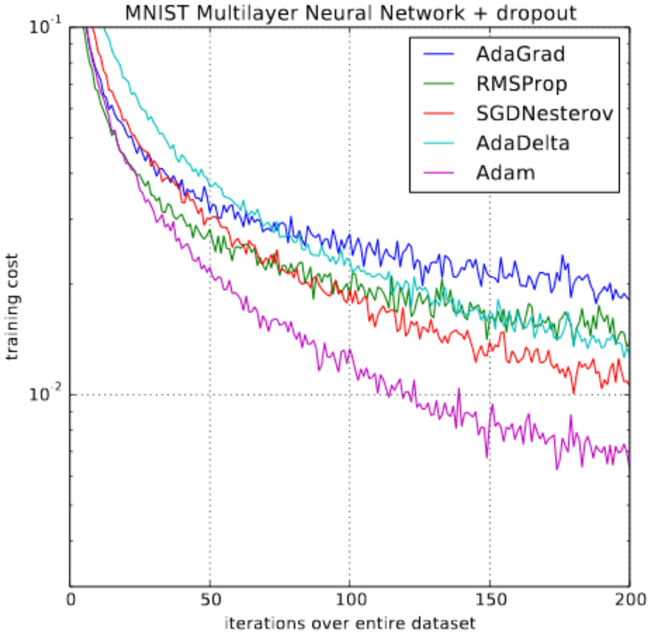


Fig. 3.16 Multilayer neural network optimizer cost comparison [76]

3.7.4 Recurrent Neural Networks

RNNs are a special form of ANNs. Their objective is to predict the next step in a sequence of observations with respect to the previous steps in the sequence [19]. They are able to use contextual information when mapping between input and output sequences [72]. This enables them to make use of sequential observations and learn from earlier stages to forecast future trends [19]. However, the range of context that can be used with RNNs is limited. That is because of the influence a given input can have on the hidden layer and in turn on the output of the network. The result either decays or blows up exponentially as it cycles around in the recurrent connections of the network [72]. Therefore, RNNs can use their feedback connections to store recent input events in the form of activations but are unable to keep a long-term memory which would be embodied by slowly changing weights [77].

LSTM is a special form of RNNs with additional features to be able to memorize longer sequences of data [19]. It is described in more detail in the next section.

3.8 Long Short-Term Memory

LSTMs are used to learn to store information over extended time intervals which would take a long time using recurrent backpropagation networks, mostly because of insufficient, decaying error backflow [77]. The authors in [78] also name LSTM as an effective and scalable model for several learning problems linked to sequential data. Earlier solutions to solve problems with sequential data were not able to scale to long-term dependencies.

An LSTM is a special form of RNN where the summation units in the hidden layer are replaced by memory blocks [72]. A multiplicative input gate unit is used to protect the stored memory contents from disruption by irrelevant inputs. Further, a multiplicative output gate unit is introduced to protect other units from disruption by currently stored irrelevant memory contents [77]. LSTM neural networks are able to almost seamlessly model problems with multiple input variables. Especially in time series forecasting this is a great benefit over classical linear methods since it can be difficult for them to adapt to multiple input forecasting problems [25].

3.8.1 LSTM Architecture

Figure 3.17 from [78] visualizes an LSTM memory block as used in the hidden layers of the recurrent neural network. As depicted, an LSTM block has three gates: an input gate, an output gate and a forget gate. It also features the block input and output at the bottom and the top, a single cell in the center also known as the Constant Error Carousel (CEC), an output activation function and peephole connections depicted in blue. The block output is recurrently connected back to the block input and all other gates as visualized by the dashed lines [78].

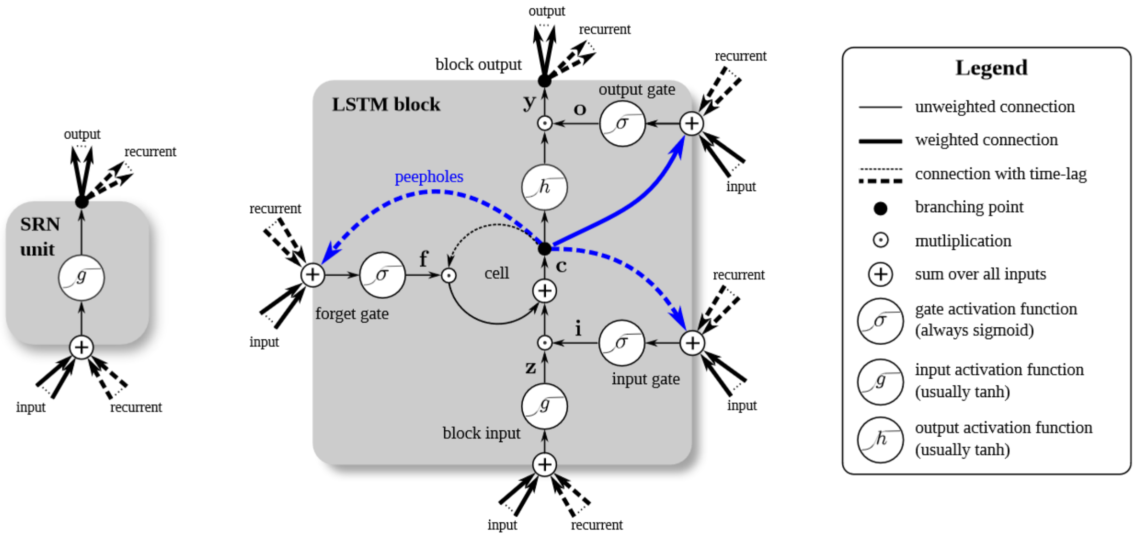


Fig. 3.17 Simple Recurrent Network unit (left) and a Long Short-Term Memory block (right) [78]

The central idea behind the LSTM architecture in Figure 3.17 is the memory block with one or more memeory cells which are able to maintain its state over time as well as non-linear gates which regulate the information flow into and out of the cell [78]. Each memory cell contains a self-connected linear unit called the CEC whose activation is called the cell state. The Carousel enforces constant error flow and overcomes the above mentioned problem. The point-wise non-linear activation functions are depicted as σ, g and h. The logistic sigmoid is used as gate activation function σ whereas the hyperbolic tangent is usually used as the block input and output activation function (g, h) [78]. The forget gate enables the LSTM to reset its own state which allows learning continual tasks. However, even tough each gate receives connections from input and output units of all cells there is no direct connection from the CEC it is supposed to control. The solution are so called peephole connections from the CEC to the gates of the same memory block. The gates learn to protect the CEC from unwanted inputs or unwanted error signals [79]. The input vector for the LSTM network is a three dimensional array that consists of the number of samples, the number of time steps and the number of features [26]. The selection of hyperparameters often makes a difference between mediocre results and state-of-the-art performance [80]. Therefore, the next subsection describes different hyperparameters of the LSTM model.

3.8.2 Hyperparameters

Number of epochs Terminologies like epochs and batch size (which is described in the next paragraph) are needed when the data is too big to pass to the algorithm at once which is usually the case in all machine learning models. An epoch of one means that the entire dataset is passed through the neural network only once. Passing the dataset through the algorithm only once might be not enough as the gradient descent algorithm is an iterative process. Updating the weights with one epoch is therefore not enough and can lead to underfitting. As the number of epochs increases the weights are changed more often and the model can go to optimal fit or even overfitting as described in Section 3.6.6. There is not a general number of epochs to use because it depends on the complexity of the dataset. Usually, the more diverse the data is the more epochs are needed [67].

Batch size Since one epoch is too big to pass to the algorithm it is divided into smaller batches [67]. The batch size represents the number of samples from the data which are shown to the machine learning algorithm before optimization or updating the weights respectively. The batch size controls how many pradictions the algorithm must make at a time. For cases where the same number of predictions should be made as used for the batch size this is not a problem. In some cases it does become a problem when fewer predictions than the batch size should be made. On sequence or some time series prediction problems it may be desirable to set a large batch size for training and a smaller batch size or a batch size of one when making predictions for the next step in the sequence. A possible approach can be to select different batch sizes for training and predicting. Usually, the internal LSTM state is cleared at the end of each batch in Keras [81]. Stochastic gradient descent, the optimization technique used in this work, is often performed with small batches. The gradients are computed from a small set of training samples rather than individual samples which is supposed to smooth out the gradient [82].

Time steps The number of time steps refers to the number of lag observations [83]. In this case, since the time series has a 15-minute interval, one time step equals to a 15-minute time frame. If the parameter is set to 10 for example it means that the algorithm looks back ten lag observations.

Neurons The number of neurons denotes the number cells in the LSTM network. It is important to select the number of neurons on the basis of the training set. To ensure a network that is able to generalize well to new data is is important to limit the number of connections. Unnecessary complexity can decrease the performance of the neural network [84].

3.8.3 Evaluation of Forecast Accuracy

Typically there are two categories of forecast accuracy measures, scale-dependent errors and percentage errors. The forecast error is simply formulated as

$$e_i = y_i - \hat{y}_i \tag{3.5}$$

where y_i denotes the ith observation and $\hat{y}_i$ denotes the forecast of y_i. The error measure is on the same scale as the data. Accuracy measures that are based on this measure are therefore scale-dependent which makes them useful for error measures with different algorithms on the same dataset but unusable for comparison between different datasets. The two most commonly used scale-dependent error measures are the Mean Absolute Error (MAE) shown in Equation 3.6 and the Root Mean Squared Error (RMSE) shown in Equation 3.7 [85]. The RMSE is a measure that is often used for assessing the accuracy of predictions made by a model. It measures the residuals between actual and predicted values [19].

$$MAE = mean(|e_i|) \tag{3.6}$$

$$RMSE = \sqrt{mean(e_i^2)} \tag{3.7}$$

Percentage errors on the other hand are given by

$$p_i = 100e_i/y_i \tag{3.8}$$

The advantage of percentage errors is that they are scale-independent and can therefore be used to compare the forecast performance among different data sets. The most commonly used measure is:

$$MAPE = mean(|p_i|) \tag{3.9}$$

However, percentage errors have the disadvantage of being infinite or undefined if $y_i = 0$ for any i and have extrem values for y_i being close to zero [85].

This section concludes the theoretical and mathematical part of this thesis. The next section describes the tools that are used to apply the above described techniques and algorithms to the data.

3.9 Frameworks

In order to create long-term autonomous systems, efficient learning methods and adaptive models are needed. The researchers in [86] give an overview about prominent machine learning frameworks with a focus on open source solutions. Figure 3.18 shows the most common frameworks according to their popularity obtained from GitHub. The rating is calculated as the sum of the GitHub contributionsx30, Issuesx20, Forksx3 and the Stars, scaled to 100% defined by the most sophisticated framework TensorFlow as a benchmark, which is the core of most machine learning and deep learning projects [86]. In this work Keras [87], the second most sophisticated framework, shown in Figure 3.18, is used with a TensorFlow backend [88]. The frameworks are described in Subsections 3.9.2 and 3.9.1 respectively. It is followed by a description of the used programming language Python along with needed libraries in Subsection 3.9.3. The chapter concludes with Subsection 3.9.4 about the used open source software platform Jupyter Notebook.

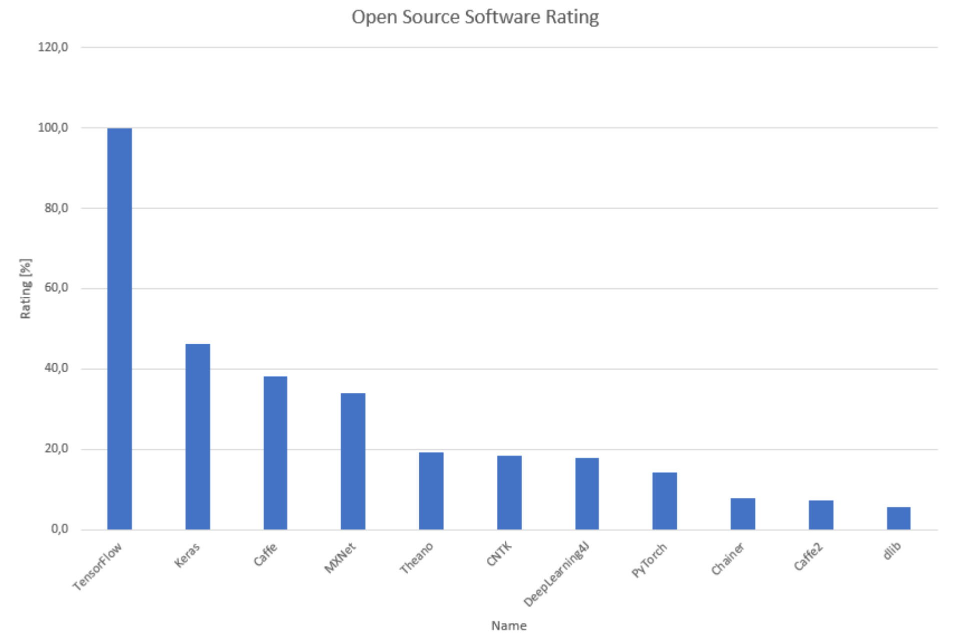


Fig. 3.18 Open source software rating based on GitHub metrics [86]

3.9.1 TensorFlow

TensorFlow is an open source software library for high performance computation. It can be deployed across a variety of platforms like CPUs, Graphics Processing Units (GPUs) and Tensorflow Processing Units (TPUs) and on desktops, clusters of servers as well as on mobile devices. The framework is originally developed by researchers from Google's AI organization. It therefore comes with strong support for machine learning and deep learning [88]. TensorFlow uses dataflow graphs to represent computation, shared state as well as the operations that modify the state. The nodes of a dataflow graph are being mapped across many machines in a cluster. TensorFlow also supports advanced machine learning algorithms that contain conditional and iterative control flow such as an LSTM network to generate predictions from sequential data [89].

3.9.2 Keras

Keras is a high-level neural networks Application Programming Interface (API) written in Python. It is capable of running on top of other frameworks like TensorFlow, CNTK or Theano. It was developed to reduce the development time compared to frameworks like TensorFlow. The focus lies on minimizing the delay from an idea to the result. It therefore allows for easy and fast prototyping, supports convolutional and recurrent networks and runs seamlessly on CPU and GPU. It is compatible with current Python versions 2.7–3.6. [87].

The guiding principles of Keras from [87] are:

- User friendliness: consistent and simple APIs, minimal number of user actions required for common use cases, clear and actionable feedback upon user error
- Modularity: neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes are standalone modules which can be combined to create new models
- Easy extensibility: new modules can be added simply (as new classes and functions)
- Works with Python: models are described in compact, easier to debug python code which can be easily extended

3.9.3 Python

Python is a general-purpose, high-level programming language focusing on code readability. The syntax allows programmers to express concepts in fewer lines of code as C for example. Python also supports multiple programming paradigms, namely object-oriented, imperative and functional programming. It features a fully dynamic type system as well as automatic memory management and comes with a large and comprehensive standard library [90]. There are two Python versions available for download that is Python 2 or Python 3. Python 2 is considered legacy unless some current Linux distributions and Macs still use Python 2 as default. However, Python 3 is now definitely mature and ready to use as long as Python 3.x is installed on the user's computers. Since Python 3 was released in 2008 most major packages have already been ported to Python 3 [91].

The easiest way to install Python libraries is via Anaconda [92], which is the open source distribution for Python and R [93]. Anaconda enables the user to easily install over 1000 data science packages and also allows to manage packages, dependencies and environments. It comes for various platforms including Windows, Linux and macOS [92]. Below is a short description of all libraries that are used in this work.

Pandas Pandas is an open source library for the Python programming language. It provides high-performance, easy-to-use high-level data structures and data analysis tools. Pandas was developed to solve the lack of support in Python for data analysis and modeling [93]. Pandas is built on top of Numerical Python (NumPy) and makes it easy to use in NumPy applications. The most important thing in pandas are its data structures: Series and Dataframe. They provide a solid basis for most applications. A Series is a one-dimensional array-like object and an contains an associated array of labels, called its index. A Dataframe represents a tabular data structure containing an ordered collection of columns. Each column can be a different datatype (numeric, string, boolean, etc.). The Dataframe has a row as well as a column index [94].

Here are some library characteristics described in [93]:

- DataFrame object for fast and efficient data manipulation and indexing.
- Tools for reading and writing data between in-memory data structures in various formats such as CSV and text files, Excel sheets, SQL databases and the HDF5 format.
- Flexible reshaping and pivoting of the data.

- High performance merging and joining of datasets.
- Time series-functionality: date range generation and frequency conversion
- Highly optimized performance
- Columns can be inserted and deleted from data structures for size mutability.
- Aggregating or transforming data and allowing for split-apply-combine operations on data sets.

NumPy NumPy is a Python package required for high performance scientific computing and data analysis. Some of the tools it provides are [94]:

- A fast and space-efficient multidimensional array providing vectorized arithmetic operations
- Standard mathematical functions for fast operations on entire arrays of data without the necessity of loops
- Tools for reading and writing array data to disk and working with memory-mapped files
- Linear algebra, random number generation and fourier transform skills
- Tools for integrating code written in C, C++ and Fortran

Matplotlib Matplotlib is an open source 2D plotting library for Python which produces figures in publication quality in various formats. Matplotlib can be used in Python scripts as well as in Python shells, web application servers, for graphical user interfase toolkits as well as within Jupyter notebook which is practical for testing. The library allows to easily generate plots, histograms, power spectra, bar charts, scatterplots, etc. Furthermore, the pyplot module provides a MATLAB-like interface. The current stable version is 2.2.2 [95].

Scikit-learn Scikit-learn is an open source library which provides simple and efficient tools for data mining and data analysis. It is built on NumPy, Scientific Python (SciPy) and matplotlib. It contains scientific toolboxes built around SciPy [96].

3.9.4 Jupyter

Jupyter is a non-profit, open-source project. Since 2014 it has evolved into supporting interactive data science and scientific computing across all programming languages. It has four different user interfaces namely JupyterLab, JupyterConsole, Qt Console and Jupyter Notebook. The latter is used in this work for simple and fast prototyping and testing along with the other above mentioned frameworks. The Jupyter Notebook is a web-based application suitable for capturing the whole computation process: developing, documenting, executing code and documenting results. It combines the two components:

- A Web Application: browser-based tool for interactive generation of documents which combine text, mathematics, computations and media output.
- Notebook Documents: representation of all contents including inputs and outputs of computations, explanatory text, mathematics, images and media representation of objects [97].

Chapter 4

Practical Part

This chapter covers the practical realization of the proposed approach, namely, the implementation of a machine learning algorithm that forecasts the demand due to EV for various selected locations in the grid. Furthermore, it comprises of a description of the used data and the preprocessing thereof, different implementations of the model and hyperparameter tuning. Finally, the results are presented and evaluated.

4.1 Data Description

In areas like machine learning the used data plays a vital role in the success of the algorithm. To perform the research objective two different datasets are used as the input for the algorithm. One dataset contains taxi trips which should model EV traffic assuming all gasoline vehicles are replaced by electric cars. The other dataset includes electricity consumption data. Subsection 4.1.1 describes the taxi trip data in detail whereas Subsection 4.1.2 describes the electricity consumption dataset.

4.1.1 Vehicle Trip Data

The used dataset contains taxi trips in NYC for a whole year and is publicly available from [98]. The data from 2017 is used because this is the first year where they used location specific pick-up and drop-off information which is a necessary information. The historical data from the years before is also available but the information about the drop-off and pick-up location is recorded as latitude and longitude values which would be more complicated to use. Table

4.1 visualizes selected rows from the dataset in its original downloaded form. As depicted, the trips are recorded in detail along with the time stamp of pick-up and drop-off as well as the location specific information.

Table 4.1 Raw vehicle trip data

Vendor ID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger _count	trip _distance	Ratecode ID	store_and _fwd_flag	PU LocationID	DO LocationID	payment _type	fare _amount	extra	mta _tax	tip _amount	tolls _amount	improvement _surcharge	total _amount
1	2017-01-01 06:53:19	2017-01-01 06:59:50	1	1.60	1	N	162	186	1	7.5	0	0.5	1.65	0	0.3	9.95
2	2017-01-01 06:53:19	2017-01-01 07:02:24	1	1.73	1	N	181	25	2	9	0	0.5	0	0	0.3	9.8
2	2017-01-01 06:53:20	2017-01-01 07:07:53	1	3.55	1	N	113	141	2	13.5	0	0.5	0	0	0.3	14.3
2	2017-01-01 06:53:20	2017-01-01 07:01:09	2	1.27	1	N	232	211	2	7.5	0	0.5	0	0	0.3	8.3
2	2017-01-01 06:53:21	2017-01-01 06:59:46	1	1.89	1	N	7	260	1	8.5	0	0.5	1.86	0	0.3	11.16
2	2017-01-01 06:53:22	2017-01-01 07:07:57	2	7.14	1	N	258	191	2	21.5	0	0.5	0	0	0.3	22.3
1	2017-01-01 06:53:23	2017-01-01 07:09:29	1	5.40	1	N	107	112	1	18	0	0.5	3.75	0	0.3	22.55
2	2017-01-01 06:53:23	2017-01-01 07:09:47	3	8.07	1	N	226	231	1	24.5	0	0.5	5.06	0	0.3	30.36
2	2017-01-01 06:53:23	2017-01-01 06:53:28	2	.00	5	N	264	42	1	65	0	0.5	0	0	0	65.5
1	2017-01-01 06:53:24	2017-01-01 06:56:12	1	.80	1	N	107	113	1	4.5	0	0.5	1.05	0	0.3	6.35
2	2017-01-01 06:53:25	2017-01-01 06:58:54	1	1.64	1	N	152	24	1	7.5	0	0.5	1.66	0	0.3	9.96
2	2017-01-01 06:53:25	2017-01-01 07:04:41	2	3.21	1	N	164	230	2	11.5	0	0.5	0	0	0.3	12.3
2	2017-01-01 06:53:26	2017-01-01 07:01:00	1	1.18	1	N	152	116	2	7	0	0.5	0	0	0.3	7.8
2	2017-01-01 06:53:26	2017-01-01 07:34:38	6	17.98	2	N	230	132	2	52	0	0.5	0	0	0.3	52.8
2	2017-01-01 06:53:26	2017-01-01 06:53:37	0	.00	5	N	48	48	1	5.5	0	0.5	1.26	0	0.3	7.56
2	2017-01-01 06:53:27	2017-01-01 06:59:24	6	1.45	1	N	249	164	1	7	0	0.5	1.56	0	0.3	9.36

4.1.2 Electricity Consumption Data

The electricity consumption data is obtained from the Pecan Street organization [99]. Pecan Street is a research network which was launched in 2009. It is the only real-world electricity-gas-water testbed in the world and is located in Austin, Texas. It consists of 750 homes and businesses, 200 solar homes and 75 EV owners. The data is downloaded from [100] with a University of Southern California academic license. Table 4.2 visualizes the raw data as it is downloaded. It includes, as visualized, the electricity consumption data in 15-minute intervals for a whole year for 200 different homes. The column ’dataid’ refers to one of the 200 households. The ’use’ column lists the used electricity in 15-minute intervals. The column ’gen’ would be the energy that is generated from solar panels. The ’grid’ column displays the actual used energy as seen from the grid and is the same as ’use’ if no solar is being used which is always the case in this dataset. The entity of the electricity consumption values is kW.

Table 4.2 Raw Pecan Street dataset

local_15min	dataid	use	gen	grid
2015-01-01 00:00:00	22	0.072000	NaN	0.072000
2015-01-01 00:15:00	22	0.071867	NaN	0.071867
2015-01-01 00:30:00	22	0.130933	NaN	0.130933
...	...	...	...	...

The above visualized raw data needs to be processed in order to serve as an input to the algorithm. Therefore, the next section describes how the datasets are prepared to do so.

4.2 Data Preprocessing

The accuracy of the forecast greatly depends upon the input for the machine learning algorithm as mentioned in [34]. Thus, a preprocessing of the data set is vital before the algorithm is applied to ensure efficient processing and better results of the forecast model.

4.2.1 Vehicle Trip Data

A city is sectioned into areas. Using any historical traffic data that contains the Start Time, End Time, Start Location and End Location of each trip a time series is generated. Table 1 shows an example entry of the dataset from [98] that fulfills the described criteria. The historical data set from 2017 contains all trips of the NYC taxi fleet for a whole year. Table 4.2 visualizes the format of the used NYC dataset. The selected values are the pickup time, the drop-off time, the pickup location and the drop-off location. The remaining columns are dropped because they are not relevant for this research purpose and unnecessary features would only downgrade the performance of the model. The location ID corresponds to a certain area in NYC which can be determined in a downloadable lookup table from [98]. For the purpose of simplicity, only one location is selected to demonstrate the EV demand prediction. It is noted that the algorithm can be used for all NYC locations available in the dataset. In a second use-case, that is described later in Section 4.2.3, another location is selected as a reference area as an additional input for the machine learning algorithm.

The first step is to extract the needed information from the large dataset described in Subsection 4.1.1. There is a separate file for every month, each being about 1-2 GB in size. The files are combined to have one file for the whole year. The resulting file contains trip data for all NYC locations. In this case location 162 and location 163, both in mid-town Manhattan are used meaning just trips to and/or from that location. In more detail, it is meant that in the case of the dataset for location 162 only the rows are selected that contain 162 either as pick-up and/or the drop-off location. However, any other location can be selected here as well by simply changing the location parameter in the code. The Python code to do so is shown in Listing 4.1. The result is a single csv file that contains only the data about one location for a whole year.

Listing 4.1 Preprocessing: location selection

```

import pandas as pd
import numpy as np
from pandas import read_csv
from pandas import DataFrame

#PARAMETERS
YEAR="2017"
LOCATION=162

appended_data = pd.DataFrame()

#read yellow taxidata for 12 months
for x in range(1,13):
    y = pd.read_csv("yellow_tripdata_" + YEAR + "-" + str(x) + ".csv",header=0,
    usecols=["tpep_pickup_datetime", "tpep_dropoff_datetime", "PULocationID", "DOLocationID"])
    y_pu=y[y.PULocationID == LOCATION]
    y_do=y[y.DOLocationID == LOCATION]
    y_total=y_pu.append(y_do)
    appended_data=appended_data.append(y_total)

#save data to csv
appended_data.to_csv("timeseries_1year_" + str(LOCATION) + ".csv", index=False)
```

The file from Table 4.1 should now look like Table 4.3.

Table 4.3 Example data from the NYC data set [98]

tpep_pickup_datetime	tpep_dropoff_datetime	PULocationID	DOLocationID
2017-01-01 06:53:19	2017-01-01 06:59:50	162	186
2017-01-01 06:53:20	2017-01-01 07:07:53	113	141

From this data a 15 minute interval time series is generated for a selected location, in this case location 162 is used which refers to Midtown East in Manhattan. For the reference case, location 163 is used which refers to Midtown North also located in Manhattan. Listing 4.2 contains the Python code used to generate the time series data. Table 4.4 visualizes the generated time series for a selected location, for example as above mentioned Midtown Center in Manhattan.

Listing 4.2 Preprocessing: vehicle trip data time series generation

```

import pandas as pd
import numpy as np
from numpy import concatenate
from matplotlib import pyplot
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from pandas import Grouper
import datetime

#PARAMETERS
LOCATION=164
START_TIME="2017-01-01 00:00:00"
END_TIME="2017-12-31 23:45:00"
FILENAME="timeseries_load_1year_"+ str(LOCATION) +".csv"
```

```
FILENAME_OPEN="timeseries_1year_" + str(LOCATION) + ".csv"


#read file
y = pd.read_csv(FILENAME_OPEN, header=0,
    usecols=["tpep_pickup_datetime", "tpep_dropoff_datetime", "PULocationID", "DOLocationID"])

#define format
y['tpep_pickup_datetime'] = pd.to_datetime(y['tpep_pickup_datetime'], format='%Y-%m-%d %H:%M:%S', utc=False)
y['tpep_dropoff_datetime'] = pd.to_datetime(y['tpep_dropoff_datetime'], format='%Y-%m-%d %H:%M:%S', utc=False)

#round original date to 15 minute intervals
y['tpep_pickup_datetime']=y['tpep_pickup_datetime'].dt.round('15min')
y['tpep_dropoff_datetime'] =y['tpep_dropoff_datetime'].dt.round('15min')

#generate time series intervals
timeseries = pd.date_range(START_TIME, END_TIME, freq="15min")
df = pd.DataFrame(timeseries)

#create column in dataframe containing the number of cars
df["number_of_cars"]=[0]*len(timeseries)

#rename columns
df.columns = ['date', 'number_of_cars']

#fill car column with the correct values for each time stamp
sum_location=0;
for x in range(0,len(df.index)):
    #case: x is pickup location, and dropoff is not x
    pu_minus_x = y[(y['tpep_pickup_datetime'] == df['date'][x]) & (y['PULocationID'] == LOCATION) & (y['DOLocationID'] != LOCATION)]
    pu_minus = len(pu_minus_x.index)

    #case: x is pickup location, and dropoff is x
    pu_do_plus_x = y[(y['tpep_pickup_datetime'] == df['date'][x]) & (y['PULocationID'] == LOCATION) & (y['DOLocationID'] == LOCATION)]
    pu_do_plus = len(pu_do_plus_x.index)

    #case: x is dropoff location and pickup is not x
    do_plus_x = y[(y['tpep_dropoff_datetime'] == df['date'][x]) & (y['DOLocationID'] == LOCATION) & (y['PULocationID'] != LOCATION)]
    do_plus = len(do_plus_x.index)

    sum_location = do_plus + pu_do_plus - pu_minus
    if(sum_location < 0):
        sum_location=0

    df.loc[x,'number_of_cars'] = sum_location
    print('sum_location:',x,sum_location)
    sum_location=0

#save file
df.to_csv(FILENAME, index=False)
```

Table 4.4 Generated time series for a selected location

Time	Number of Cars
2017-01-01 00:00	0
...	...
2017-01-01 11:30	0
2017-01-01 11:45	2
2017-01-01 12:00	1
...	...
2017-12-31 23:45	0

Figure 4.1 visualizes the hotspot identification in a city. The vehicles leave one area and enter another at certain times. This information is extracted from the above described NYC taxi data. The depicted scenario leads to the generated time series from Table 4.4. The red square is the observed location 162 but any other location can be observed as well. Ideally, when all fields in the grid are observed, the hotspots in a city can be identified. In the depicted simplified case are two cars, C1 in green and C2 in red at different timestamps. For example, Car C2 enters the observed area at 11:31. Car C1 enters the same area at 11:36. Therefore in the time frame 11:45 which covers the time from 11:31 until 11:45 the car count in this area is 2. When C1 leaves the area at 11:46 there is only one car left in the area. It is assumed in this work that if a car resides in one area longer than 15 minutes it can be subject to charging.

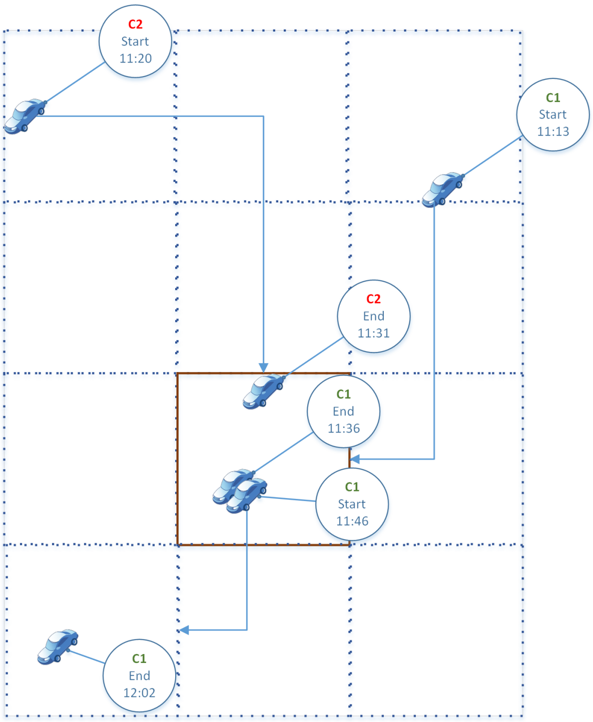


Fig. 4.1 Visualization of hotspot identification

The taxi data is now ready to be combined with the electricity consumption data to form a single dataset that serves as the input to the LSTM forecasting model.

4.2.2 Electricity Consumption Data

The electricity consumption data is first filtered for a specific user. This is because it was found that not every user in the dataset has a continuous time series for the whole year. In this case user ’86’ is used. Table 4.5 depicts the structure of the dataset after applying a filter for user ’86’. For the later in Subsection 4.2.3 described case of a reference location also the data of user ’59’ is used - both having continuous consumption values for the year. Along with the time stamp as index the column ’use’ is selected, all other columns are dropped. Furthermore, the ’use’ values are multiplied by a factor of 1000 to model a denser household area that better fits the amount of the taxi data. Since there is currently no publicly available dataset with the needed granularity on a city level, the Pecan Street Consumption data is used for the research serving as a proof-of-concept. It is noted that for better performance more accurate data is needed that matches better with the vehicle trip data.

Table 4.5 Example of the Pecan Street dataset for a specific household

local_15min	**dataid**	**use**	**gen**	**grid**
2015-01-01 00:00:00	86	0.738133	NaN	0.738133
2015-01-01 00:15:00	86	0.738933	NaN	0.738933
2015-01-01 00:30:00	86	0.739000	NaN	0.739000
...				

For further processing and combination with the other dataset only the ’use’ column is selected. The next subsection describes how the final dataset that is later presented to the machine learning algorithm.

4.2.3 Combined Data and Preparation for LSTM

From the described datasets in Subsection 4.2.1 and Subsection 4.2.2 a single dataframe is generated. Since the Pecan Street data does not contain EV charging, the following formula in Equation 4.1 is used to create an additional column consisting of the approximate EV load. In the formula an approximate EV load is calculated for every time step i. Where $cars_i$ is the number of cars at time i, the constant $ALPHA$ is a parameter for charging meaning how many percent of EVs that are currently in the location are actually charging and $CHARGINGL1$ is the selected charging level as described in 3.3.4 which is constant throughout the simulation. Besides the EV load also the total load is added as a column to the dataset which is simply a summation of the electricity consumption at time i and the previously described calculated EV load. The final dataset which is used for the algorithm is shown in Table 4.6. Figure 4.2 displays the histograms of the three features over the timespan of a year.

$$evload_i = cars_i * ALPHA * (0.25 * CHARGINGL1) \tag{4.1}$$

Table 4.6 Combined final dataset

local_15min	**cars_in_location**	**total_load**	**ev_load**
2015-01-01 00:00:00	0	738.13	0
2015-01-01 00:15:00	0	738.93	0
2015-01-01 00:30:00	43	997	258
2015-01-01 00:45:00	13	1018.67	78
2015-01-01 01:00:00	0	689.47	0
...			

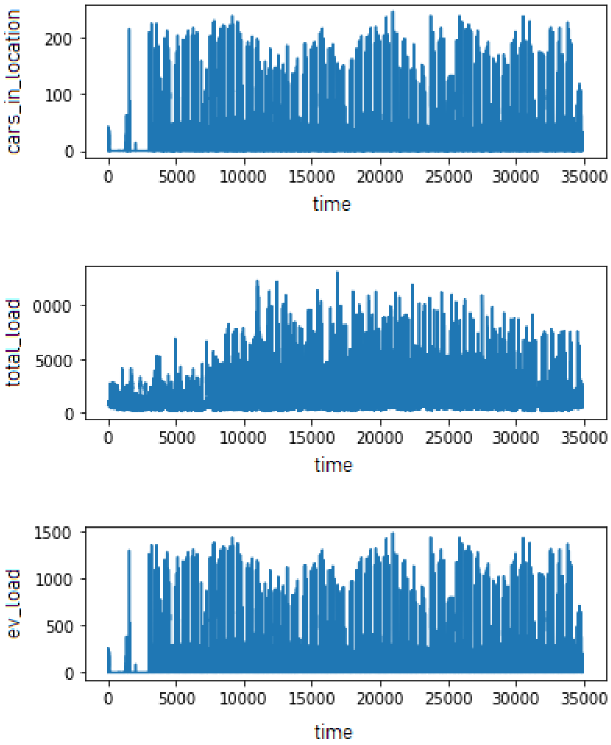


Fig. 4.2 Histogram of the input features

After the data is prepared, it can be used for the LSTM algorithm. First, the conversion of the data to a supervised learning problem is described. After that, the implementation of the LSTM algorithm is given.

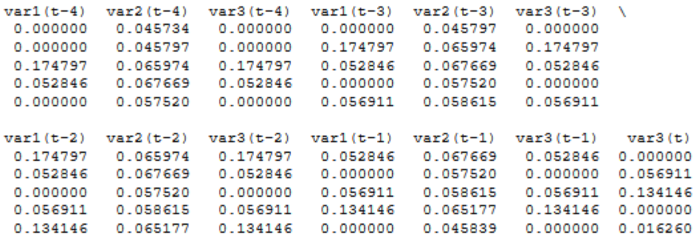
4.3 Multivariate LSTM Forecast Model

According to [25], recurrent neural networks are able to model problems with multiple input features. Therefore, an LSTM is used in this work to forecast the electricity demand of EVs in a selected area. The code for the machine learning algorithm is based on the tutorial in [25]. The next step is to frame the dataset as a supervised learning problem and to normalize the input features since not all of them are on the same scale. The problem is framed as predicting the energy demand due to EVs for future time steps using the total load and traffic information from previous time steps. The values in the dataset are normalized to the range of 0 and 1. The dataframe is converted to a supervised learning problem using the function shown in Listing 4.3 from [101]. The function creates pairs of input and output sequences from the given input sequence. After the code of the function, the values are normalized and columns that should not be predicted are dropped. In this case the total load and the number of cars should not be predicted, only the load due to EVs.

Listing 4.3 Preprocessing: convert series to supervised learning

```

# function: convert series to supervised learning
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = pd.DataFrame(data)
    cols, names = list(), list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j + 1, i)) for j in range(n_vars)]
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j + 1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j + 1, i)) for j in range(n_vars)]
    # put it all together
    agg = pd.concat(cols, axis=1)
    agg.columns = names
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg

# dataset
values = dataset.values

# ensure all data is float
values = values.astype('float32')

# normalize features
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)

# frame as supervised learning
reframed = series_to_supervised(scaled, n_15min, 1)

# drop columns we don't want to predict
reframed.drop(reframed.columns[[n_15min*n_features,n_15min*n_features+1]], axis=1, inplace=True)
print(reframed.head())
```

The data is now prepared for a machine learning algorithm. Since only the load due to EVs should be predicted this column is removed. After that, the first five values of each column containing the input features at different time steps are displayed in Figure 4.3. Where *var*1 refers to the cars in a location, *var*2 refers to the total load and *var*3 is just the EV load. Since the prediction should only be made for the EV load, the other columns are dropped for time *t*.



```
var1(t-4)  var2(t-4)  var3(t-4)  var1(t-3)  var2(t-3)  var3(t-3)  \
 0.000000   0.045734   0.000000   0.000000   0.045797   0.000000
 0.000000   0.045797   0.000000   0.174797   0.065974   0.174797
 0.174797   0.065974   0.174797   0.052846   0.067669   0.052846
 0.052846   0.067669   0.052846   0.000000   0.057520   0.000000
 0.000000   0.057520   0.000000   0.056911   0.058615   0.056911

var1(t-2)  var2(t-2)  var3(t-2)  var1(t-1)  var2(t-1)  var3(t-1)   var3(t)
 0.174797   0.065974   0.174797   0.052846   0.067669   0.052846  0.000000
 0.052846   0.067669   0.052846   0.000000   0.057520   0.000000  0.056911
 0.000000   0.057520   0.000000   0.056911   0.058615   0.056911  0.134146
 0.056911   0.058615   0.056911   0.134146   0.065177   0.134146  0.000000
 0.134146   0.065177   0.134146   0.000000   0.045839   0.000000  0.016260
```

Fig. 4.3 Reframed supervised learning data

The next step is to split the dataset into train and test sets. A common value is to use 70% of the data for training and 30% for testing. The code in Listing 4.3 is used to do so.

```

# split into train and test sets
values = reframed.values
numel = len(values)
#define percentage for training

n_train_intervals = math.floor(train * numel)

#split into test and training data
train = values[:n_train_intervals, :]
test = values[n_train_intervals:, :]

# split into input and outputs
n_obs = n_15min * n_features
train_X, train_y = train[:, :n_obs], train[:, -1]
test_X, test_y = test[:, :n_obs], test[:, -1]

train_X.shape[0]

#reshape input to be 3D [samples, time steps, features]
train_X = train_X.reshape((train_X.shape[0], n_15min, n_features))
test_X = test_X.reshape((test_X.shape[0], n_15min, n_features))
```

After splitting the data into train and test sets it can be used as an input for the LSTM algorithm. The code is displayed in Listing 4.4. The meaning of hyperparameters like number of neurons, epochs and batch size is described in Section 3.8.2. The values used in the listing form the basic setup and are tuned according to the results obtained from different hyperparameter variations which are described in Section 4.4.

Listing 4.4 LSTM implementation

```

# design network
model = Sequential()
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')

#fit network
history = model.fit(train_X, train_y, epochs=50, batch_size=72, validation_data=(test_X, test_y), verbose=2, shuffle=False)
```

The results from the in this section described LSTM algorithm are presented in the next section. Besides the results for the above presented use case with one location the model is also trained with more features in form of a reference location as well as with a two year dataset which is increased artificially.

4.4 Results

The following tables display the basic setup for the simulation. In Section 4.4.1 the effect of different values for selected parameters is tested to obtain an optimal parameter set.

Table 4.7 Parameter values for basic setup

Basic Setup	
Parameter	**Value**
ALPHA	0.3
CHARGING_L1	80
n_15min	4
n_features	3
train	0.7

Table 4.8 Parameter values for LSTM setup

LSTM Settings	
Parameter	**Value**
Neurons	50
Layers	1
loss	MAE
Optimizer	Adam
Batch Size	72
Epochs	50
verbose	2
shuffle	False

4.4.1 Forecasting Model with One Location

The simplest case is where only the time series for one year and for one location is used. This means that the input parameters for the algorithm are the number of cars in the location, the total electricity load in that location as well as just the load due to EVs. The duration of the time series is for this case one year. The model is first trained with the basic parameters shown in Table 4.7 and 4.8. The result is visualized in Figure 4.4. The y-axis displays the error whereas the x-axis values are the different epochs. It can be seen that the test error stays higher while the train error goes down. This is typical for an underfit model which means that more data is needed.

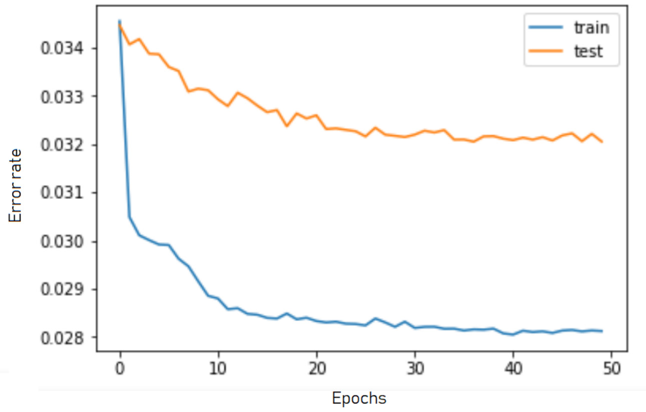


Fig. 4.4 Forecasting model with one location train/test error

Table 4.9 shows the result for the parameters in Table 4.8. The result is presented in terms of the RMSE. The mean, min and max values are also displayed to give a more meaningful result.

Table 4.9 Results for the forecasting model with one location for one year

RMSE	15.004
Mean	17.695
Min	0
Max	238

Despite the underfitted result, the model is also tested with different parameter settings to tune the model. Different values for LSTM hyperparameters are tested.

Table 4.10 Results for the forecasting model with one location for one year with varying number of epochs

Test: Number of Epochs	
Epochs	**RMSE**
10	15.250
30	15.009
50	15.004
70	14.967
90	14.907
100	15.002
150	14.937
200	14.963

Table 4.11 Results for the forecasting model with one location for one year with varying batch size

Test: Batch Size	
Batch Size	**RMSE**
8	14.886
48	15.019
72	15.004
96	14.934
192	15.007
384	15.225
1024	15.491

Table 4.12 Results for the forecasting model with one location for one year with varying time steps

Test: Time steps (15-min intervals)	
Time steps	**RMSE**
1	16.331
4	15.004
10	13.653
26	13.713
42	13.833
74	13.750
96	13.833

Table 4.13 Results for the forecasting model with one location for one year with varying neurons

Test: Neurons	
Number of Neurons	**RMSE**
10	14.977
30	14.938
70	14.949
90	14.993
100	15.042
150	14.938
500	14.954

The final optimal parameter set is obtained by selecting the parameters that obtained the lowest RMSE value from each hyperparameter test. Table 4.14 summarizes the resulting parameters. Figure 4.5 visualizes the train and test error obtained with this configuration. Table 4.15 shows the resulting RMSE value along with the mean, min and max values.

Table 4.14 Optimal parameters for one location one year forecasting model

Epochs	90
Neurons	30
Batch Size	8
Time steps	10

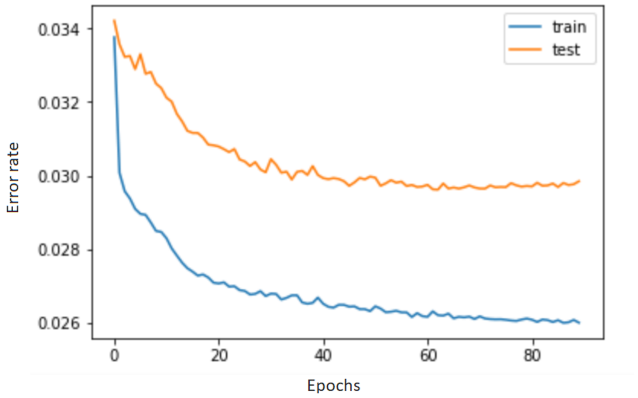


Fig. 4.5 Forecasting model with one location train/test error with optimal parameters

Table 4.15 Results: optimal parameters, 1 location, 1 year

RMSE	13.810
Mean	17.698
Min	0
Max	238

4.4.2 Forecasting Model with a Reference Location

In this approach the input features are increased by three. The number of cars, the total load as well as the load due to EVs from a reference location, in this case location 163 is used, are added. The total number of features in this test is six. The duration of the data is one year. Table 4.16 visualizes the results in terms of the RMSE. Figure 4.6 shows the corresponding train/test error plot.

Table 4.16 Results: 1 year with reference location

RMSE	16.170
Mean	16.424
Min	0
Max	199.974

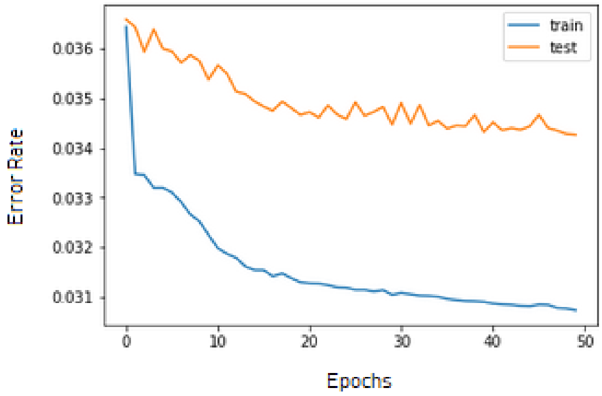


Fig. 4.6 Forecasting model with reference location train/test error

4.4.3 Forecasting Model with an Artificially Increased Dataset and One Location

The model is extended with another year of data. For the load data more data is available from Pecan Street but the traffic data needs to be generated for the second year. The second year of data is generated using the code in Listing 4.5. The histogram plot of the original data visualized in Figure 4.7 and the histogram of the generated data visualized in Figure 4.8 are similar. The parameters as well as the LSTM settings are the same as in the basic setup listed in Section 4.4.1. Table 4.17 visualizes the results in terms of the RMSE. Figure 4.9 shows the corresponding train/test error plot.

Listing 4.5 Generation of second year car data

```
mu, sigma = 0,14 #mean and standard deviation
s=np.random.normal(mu,sigma,len(car_data))
np.around(s) #round to integer
car_data_year2 = car_data.copy(deep=True) #make copy of car_data

for x in range(len_car):
    if(s[x]==0):
        car_data_year2[x]=0
    if(s[x]>0 and car_data_year2[x] > 0):
        car_data_year2[x]= car_data_year2[x]-s[x]
        if(car_data_year2[x]<0):
            car_data_year2[x]=0
    if(s[x]>0 and car_data_year2[x] < 0):
        car_data_year2[x]=s[x]
    if(s[x]>0 and car_data_year2[x] == 0):
        car_data_year2[x]=s[x]
    if(s[x]<0 and car_data_year2[x]==0):
        car_data_year2[x]=s[x]
    if(s[x]<0 and car_data_year2[x]>0):
        car_data_year2[x]=s[x]

car_data_year2 = car_data_year2.abs()
```

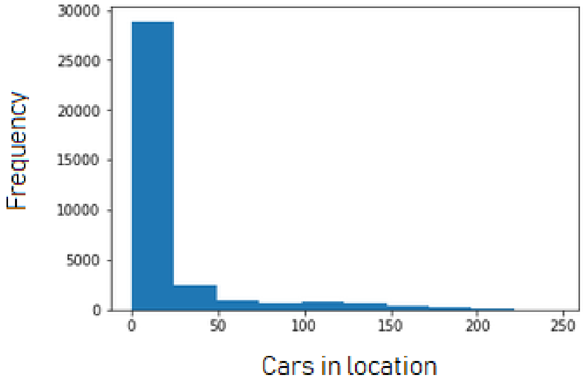


Fig. 4.7 Histogram of original data for one year

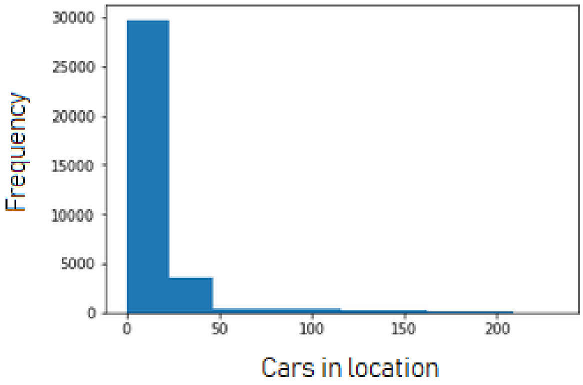


Fig. 4.8 Histogram of generated data for second year

Table 4.17 Results: 1 location, 2 years of data

RMSE	24.420
Mean	9.1823
Min	0
Max	235

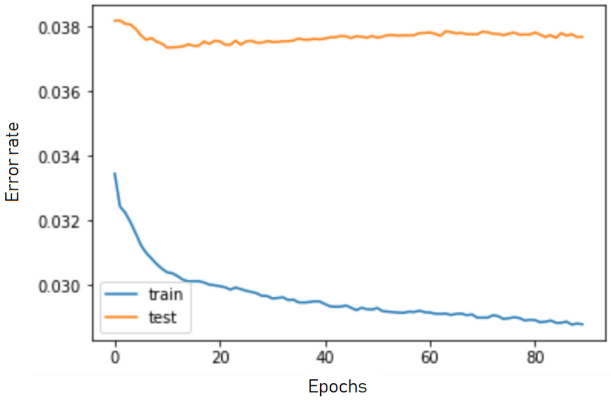


Fig. 4.9 Forecasting model with one location and two year data train/test error

After the implementation part the results can be evaluated. Therefore, the next section covers the discussion of the obtained results.

4.5 Evaluation

The results of the first setup with the basic parameter settings visualized in Table 4.7 and 4.8 are summarized in Table 4.9. The input features are the three time series, namely the number of cars, the total load and the load due to EVs from a selected location. The duration of the data is one year. Figure 4.4 shows the graphical representation of the obtained train and test error graphs. It can be seen that the train error is significantly lower than the test error. This behaviour is typical for underfitting as shown in [102]. Different experiments were performed to improve the result. The following subsections evaluate the results from the different hyperparameter settings and test cases.

4.5.1 Evaluation of Different Hyperparameter Settings

Hyperparameters are parameters which are adjusted before the machine learning algorithm. Therefore, different settings have been simulated to show their effect on the performance of the algorithm to obtain the best possible parameter set. The meaning of the different hyperparameters is described in Subsection 3.8.2.

Number of epochs The outcome in terms of the RMSE of different epoch values is shown in Table 4.10. Epoch values between 10 and 200 are tested. The upper value of 200 is chosen to keep runtimes reasonable. However, the outcome shows that there is no significant difference or reduced error rate between different epoch values. At first, the error goes down and is the lowest at 90 epochs. With a further increase of epochs, the error seems to get up and then down again. The random behavior of different epoch values has also been observed by researchers in [19] who declared that the number of epochs has no significant effect on the outcome of the algorithm which has also been found in this work.

Batch size Table 4.11 shows the values for the batch size and their outcome in terms of the RMSE. Batch size values between 8 and 1024 are tested. The results show a similar behaviour than the number of epochs. It can be concluded that also this parameter has, at least in this setup, no significant effect on the model performance.

Time steps Table 4.12 shows the tested values for the number of time steps and their outcome in terms of the RMSE. The tested values range from 1 to 96. The results show that the error rate is significantly reduced from one time step to 10 time steps, then the error rate starts to slowly increase with a higher number of time steps. It can be concluded that for this case 10 time steps would be the best value as it results in the lowest error rate.

Number of neurons Table 4.13 shows the tested values for the number of neurons and their outcome in terms of the RMSE. The tested values range from 10 to 500. The best value is received when choosing 30 neurons. However, no significant change was observed for different values.

4.5.2 Evaluation of the Different Test Cases

The algorithm is trained with different input features. First, the algorithm is trained with 1 year of data were only one location is used. Next, the model is trained with one year of data but an additional location is added as a new input feature to serve as a reference location. Last, the training is performed with two years of data for one location. The results in terms of the RMSE values of the different settings are summarized in Table 4.18. The mean, min and max values are also provided to be able to compare the results better since each dataset is slightly different. However, it can still be seen that the first case with one location for one year performs best. It is assumed that an increase of the dataset or an extension of the features would improve the forecasting model but this is not the case as can be seen from the resulting error rates. This can be due to the fact that an increased number of features adds more complexity to the model which in turn leads to a more difficult learning phase. Moreover, when the number of features is higher, more training data is needed for an optimal adjusting of the weights. The two year dataset has the worst performance with an error rate of 24.42. Usually, an increase in data is expected to significantly improve the model. However, in this case the reason is possibly that for the vehicle trip data no second year was available. The second year was therefore generated base on the data of the first year. Because of the already existing mismatch in the data a second year might add more confusion instead of clarification. It is expected that it would be beneficial to have input features that are from the same location and actually reflect each other in terms of year, weather and location.

Table 4.18 Summary of the results for the different cases

	RMSE	**Mean**	**Min**	**Max**
1 year 1 location	13.810	17.698	0	238
1 year 2 locations	16.170	16.424	0	199.974
2 year 1 location	24.420	9.1823	0	235

4.5.3 General Problems

The following problems have been identified to impact the performance of the algorithm:

- different seasonal pattern of EV demand and general electricity load
- sparse vehicle dataset
- different data sources
- amount of training data

A reason for the higher forecasting error might be the seasonally changing load patterns of general electricity consumption. As the researchers in [6] mention, conventional electricity load and charging demand of electric vehicles might have different seasonal patterns. This would explain why it is hard for the algorithm to learn the electric vehicle demand from the total load that is seasonally changing but has not the same effect on the EV demand. Furthermore, the vehicle dataset contains a lot of zero values which stand for the times when there is no car in that location at a particular 15-minute interval. Those zeros cannot simply be dropped because they have a meaning, namely that no car is in that location which can potentially be requesting charging. This leads to the conclusion that data is needed that would represent a more realistic and frequent car appearance to be able to train an optimal model. With the rise of ride-share companies like Uber and Lyft it might be wrong to still assume that taxi traffic in NYC still models normal traffic behaviour. Also, the different data sources can be problematic for the results. The different datasets, the traffic data from NYC and the consumption data from Texas mismatch in year, location and resulting other factors like weather conditions. Last, it has been identified that the amount of training data is too less to fit such a complex model. As already mentioned in Chapter 2, a downside of neural network based models is that a lot of training data is needed. It is expected that if the amount of training data is increased that the model would provide better results.

Chapter 5

Conclusion and Future Work

The purpose of this thesis was to develop a machine learning model that is able to predict the energy demand that is due to electric vehicles from the given time series of the total load and the traffic information for various locations available in the data. The first chapter introduced the research topic and the objective of this work. First, an overview about current developments in the Smart Grid is given to introduce the reader to the domain and point out associated challenges, especially due to the deployment of electric vehicles. To address the seriousness of the problem government initiatives to counteract climate change are presented as well. The outcome of this chapter is a description of the research objective as well as the developed research questions which define the scope of this thesis. The questions are answered throughout this work.

In Chapter 2, the findings of the literature research are presented. Related work in the field of electric vehicle electricity demand forecasting and connected topics like allocation of charging stations and test data development are discussed. Furthermore, the state-of-the-art techniques for time series forecasting are compared and analysed. Different forecasting algorithms in the domain of time series forecasting are compared and discussed. It was found that a RNN approach like LSTM would be the best fit for this research objective. LSTM was chosen because of its ability to handle multivariate inputs, its capacity to store information over a longer period of time as well as its ability to handle non-linearity. Furthermore, it was found in a recent study where a direct comparison between ARIMA and LSTM for time series forecasting was conducted, that the LSTM model outperforms the ARIMA model in terms of having significantly smaller error rates. This is another reason why LSTM was chosen over the up-to-date most widely used demand forecasting algorithm ARIMA. Finally, the scope of the thesis is defined in more detail along with a clear distinction from the described existing work. The solution is a flexible and costly inexpensive model that aims to work using publicly available datasets in a form that would enable its application for various locations. Also, the algorithm is able to handle multivariate inputs well.

Chapter 3 covered the theoretical aspects of the thesis by means of the topics which are relevant for understanding the domain as well as definitions and standards which are used in the practical part of the thesis. First, a detailed overview about the Smart Grid is given which aims to follow up with the short description already given in the introduction. The next section focuses on demand forecasting in the Smart Grid and further addresses the objectives and challenges. This is followed by a detailed section about electric vehicles. The role of the EV is addressed as well as technical details about the charging infrastructure, different EV types and charging levels are given. The next sections are built constructively and aim to introduce the reader into the topic of neural networks. First, the area of data analytics is discussed followed by an introduction into machine learning. Different techniques like supervised, unsupervised and reinforcement learning are presented. Furthermore, common problems like overfitting and underfitting are described as well. The next section about neural networks describes their functionality along with the learning process and architecture in detail. Also, RNNs are described in order to understand the used algorithm LSTM which falls into this category. Next, LSTM is explained in more detail to understand the practical implementation of the algorithm. Last, an overview about the used frameworks is given. Python was selected as the programming language due to its usability and its compatibility with machine learning. The used libraries for the algorithm and graphical representation are pandas, numPy, matplotlib and scikit-learn. The used framework Keras which runs on Tensorflow was chosen because of its simple API's which are built on top of TensorFlow. The outcome of this chapter is a thorough description of all the topics which are relevant for the practical implementation of this thesis.

In Chapter 4 the practical part of this work is presented. The used data is on the one hand vehicle trip data from the NYC taxi fleet which is publicly available and on the other hand electricity consumption data from a Smart Grid testbed in Austin, Texas. Unfortunately, no data was found from the same location. The vehicle trip data is a time series with taxi trips in NYC for a whole year. The raw data from both sources was preprocessed to have the right format. Unnecessary features were dropped. The two datasets are then combined into a single table where the time series column is in the granularity of 15 minutes. The dataset is extended with an artificially created column for the load due to EVs which is used for training in the supervised learning process. The load is calculated using a developed formula that takes the charging level into account as well as a parameter which stands for how many EVs that are in a location at a given interval are actually charging. The prepared data is converted into a supervised learning problem and split into train and test data sets. 70% are used for training and the remaining 30% for testing. The train data is then fed into the LSTM algorithm. The inputs for training are the total load, the EV load as well as the number of cars in the location. The goal for the trained algorithm is to predict the load due to EVs by just receiving the total load and the number of cars.

The outcome of the practical part is a demand forecasting model that is able to make reasonable forecasts. However, the resulting error plots show that the model is still underfitted. As found in the literature research more training data is needed to optimize the model. Furthermore, different hyperparameter settings have been tested on the model. The results show that while the number of epochs, the batch size and the number of neurons have no significant effect on the system performance an increase in the number of time steps shows a better performance. Besides the tests on hyperparameters also different input features have been tested. The model is trained with three variations. At first, one year of data and one location is used which results in three input features. Next, the model is trained with one year of data but additionally a reference location is added which results in six total features. Finally, the model is trained with two years of data for one location of which the second year of the vehicle data is created artificially due to lack of data. The results showed that the lowest error rates can be achieved with the first case. It is concluded that more data is needed to train an optimal fit model. Furthermore, the data should match in year and location to create a more reliable model.

Future Work The implementation is limited by the current publicly available data. In order to extend the range of this thesis for different geographical locations, it is expected that more vehicle trip data is made available online in the future such as the one from the New York City taxi fleet. Uber is already beginning to upload anonymous trip data from selected cities via Uber Movement [103]. Furthermore, with the continuing deployment of Smart Meters across the globe, anonymous consumption data is also expected to be made available for more geographical locations in the future to enhance the research process. Electricity consumption data and vehicle data from the same location would help to provide stronger results.

Furthermore, the second year of vehicle data was artificially generated. In the future the model can be updated using real data for the second year as well. One way to do this is to include the 2018 NYC taxi data once it is available from [98]. Currently real data is only used for 2017 because the data for 2018 is not yet published. As soon as the year is over the second-year data can be replaced by the real dataset. However, since the sparseness of the taxi data set it might not be the best solution. Another way can be to generate EV traffic data using the stochastic approach presented by the researchers in [4].

Bibliography

[1] T. Woods. (2015). Peter Thiel's 7 Questions for Product Innovation, [Online]. Available: https://blog.hypeinnovation.com/peter-thiels-7-questions-for-product-innovation (visited on 08/21/2018).

[2] D. Zapfl. (2016). Welche Innovationsarten gibt es? [Online]. Available: http://www.lead-innovation.com/blog/innovationsarten (visited on 08/21/2018).

[3] F. Emprechtinger. (2016). Was ist der Innovationsgrad? [Online]. Available: http://www.lead-innovation.com/blog/innovationsgrad (visited on 08/21/2018).

[4] M. Alizadeh, A. Scaglione, J. Davies, and K. S. Kurani, "A scalable stochastic model for the electricity demand of electric and plug-in hybrid vehicles," *IEEE Transactions on Smart Grid*, vol. 5, no. 2, pp. 848–860, 2014.

[5] K. S. Kurani, J. Axsen, N. Caperello, J. Davies, and T. Stillwater, "Learning from consumers: Plug-in hybrid electric vehicle (phev) demonstration and consumer education, outreach, and market research program," 2009.

[6] M. H. Amini, A. Kargarian, and O. Karabasoglu, "Arima-based decoupled time series forecasting of electric vehicle charging demand for stochastic power system operation," *Electric Power Systems Research*, vol. 140, pp. 378–390, 2016.

[7] M. Majidpour, C. Qiu, P. Chu, R. Gadh, and H. R. Pota, "Fast prediction for sparse time series: Demand forecast of ev charging stations for cell phone applications," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 1, pp. 242–250, 2015.

[8] N. Korolko, Z. Sahinoglu, and D. Nikovski, "Modeling and forecasting self-similar power load due to ev fast chargers," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1620–1629, 2016.

[9] J. Wang, K. H. Wu, F. Wang, Z. H. Li, Q. S. Niu, and Z. Z. Liu, "Electric vehicle charging station load forecasting and impact of the load curve," in *Applied Mechanics and Materials*, Trans Tech Publ, vol. 229, 2012, pp. 853–858.

[10] N. Neyestani, M. Y. Damavandi, M. Shafie-Khah, J. Contreras, and J. P. Catalão, "Allocation of plug-in vehicles' parking lots in distribution systems considering network-constrained objectives," *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2643–2656, 2015.

[11] F. He, D. Wu, Y. Yin, and Y. Guan, "Optimal deployment of public charging stations for plug-in hybrid electric vehicles," *Transportation Research Part B: Methodological*, vol. 47, pp. 87–101, 2013.

[12] M. Muratori, M. J. Moran, E. Serra, and G. Rizzoni, "Highly-resolved modeling of personal transportation energy consumption in the united states," *Energy*, vol. 58, pp. 168–177, 2013.

[13] H. Akhavan-Hejazi, H. Mohsenian-Rad, and A. Nejat, "Developing a test data set for electric vehicle applications in smart grid research," in *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*, IEEE, 2014, pp. 1–6.

[14] L. Rayle. (2017). The taxi market in NYC vs SF: inferring trip purpose using LDA, [Online]. Available: http://lisarayle.com/comparing-the-taxi-market-in-new-york-city-and-san-francisco/ (visited on 07/02/2018).

[15] R. Josue. (2018). 7 Ways Time-Series Forecasting Differs from Machine Learning, [Online]. Available: https://www.datascience.com/blog/time-series-forecasting-machine-learning-differences (visited on 08/21/2018).

[16] C. Chatfield, *Time-series forecasting*. Chapman and Hall/CRC, 2000.

[17] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[18] Z. Tang, C. De Almeida, and P. A. Fishwick, "Time series forecasting using neural networks vs. box-jenkins methodology," *Simulation*, vol. 57, no. 5, pp. 303–310, 1991.

[19] S. Siami-Namini and A. S. Namin, "Forecasting economics and financial time series: Arima vs. lstm," *arXiv preprint arXiv:1803.06386*, 2018.

[20] M. Assaad, R. Bone, and H. Cardot, "A new boosting algorithm for improved time-series forecasting with recurrent neural networks," *Information Fusion*, vol. 9, no. 1, pp. 41–55, 2008.

[21] S. Swaminathan. (2018). Linear Regression-Detailed View, [Online]. Available: https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86 (visited on 08/21/2018).

[22] G. Huerta. (2017). Introduction to Dynamic Linear Models, [Online]. Available: http://www.math.unm.edu/ghuerta/tseries/dlmch2.pdf (visited on 08/21/2018).

[23] P. Jinka. (2017). Exponential Smoothing for Time Series Forecasting, [Online]. Available: https://www.vividcortex.com/blog/exponential-smoothing-for-time-series-forecasting (visited on 08/21/2018).

[24] R. Dalinina. (2017). Introduction to Forecasting with ARIMA in R, [Online]. Available: https://www.datascience.com/blog/introduction-to-forecasting-with-arima-in-r-learn-data-science-tutorials (visited on 08/21/2018).

[25] J. Brownlee. (2017). Multivariate Time Series Forecasting with LSTMs in Keras, [Online]. Available: https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/ (visited on 05/16/2018).

[26] R. Kompella. (2018). Using LSTMs to forecast time-series, [Online]. Available: https://towardsdatascience.com/using-lstms-to-forecast-time-series-4ab688386b1f (visited on 08/21/2018).

[27] Z. Fan, P. Kulkarni, S. Gormus, C. Efthymiou, G. Kalogridis, M. Sooriyabandara, Z. Zhu, S. Lambotharan, and W. H. Chin, "Smart grid communications: Overview of research challenges, solutions, and standardization activities," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 21–38, 2013.

[28] S. M. Amin and B. F. Wollenberg, "Toward a smart grid: Power delivery for the 21st century," *IEEE power and energy magazine*, vol. 3, no. 5, pp. 34–41, 2005.

[29] E. Commission. (2017). Smart grids and meters, [Online]. Available: https://ec.europa.eu/energy/en/topics/markets-and-consumers/smart-grids-and-meters (visited on 05/30/2018).

[30] K. Moslehi and R. Kumar, "A reliability perspective of the smart grid," *IEEE Transactions on Smart Grid*, vol. 1, no. 1, pp. 57–64, 2010.

[31] S. Austria. (2016). Was sind Smart Grids? [Online]. Available: https://www.smartgrids.at/smart-grids.html (visited on 08/21/2018).

[32] E. Commission. (2017). Smart Grids and Meters, [Online]. Available: https://ec.europa.eu/energy/en/topics/markets-and-consumers/smart-grids-and-meters (visited on 07/16/2018).

[33] F. Shariatzadeh, P. Mandal, and A. K. Srivastava, "Demand response for sustainable energy systems: A review, application and implementation strategy," *Renewable and Sustainable Energy Reviews*, vol. 45, pp. 343–350, 2015.

[34] M. Q. Raza and A. Khosravi, “A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings,” *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 1352–1372, 2015.

[35] M. E. El-Hawary, “The smart grid—state-of-the-art and future trends,” *Electric Power Components and Systems*, vol. 42, no. 3-4, pp. 239–250, 2014.

[36] M. E. Khodayar and H. Wu, “Demand forecasting in the smart grid paradigm: Features and challenges,” *The Electricity Journal*, vol. 28, no. 6, pp. 51–62, 2015.

[37] L. Hernández, C. Baladron, J. M. Aguiar, B. Carro, A. Sanchez-Esguevillas, J. Lloret, D. Chinarro, J. J. Gomez-Sanz, and D. Cook, “A multi-agent system architecture for smart grid management and forecasting of energy demand in virtual power plants,” *IEEE Communications Magazine*, vol. 51, no. 1, pp. 106–113, 2013.

[38] A. Motamedi, H. Zareipour, W. D. Rosehart, *et al.*, “Electricity price and demand forecasting in smart grids,” *IEEE Trans. Smart Grid*, vol. 3, no. 2, pp. 664–674, 2012.

[39] C. D. Anderson and J. Anderson, *Electric and hybrid cars: A history*. McFarland, 2010.

[40] S. F. Tie and C. W. Tan, “A review of energy sources and energy management system in electric vehicles,” *Renewable and Sustainable Energy Reviews*, vol. 20, pp. 82–102, 2013.

[41] P. Chakraborty, E. Baeyens, K. Poolla, P. P. Khargonekar, and P. Varaiya, “Sharing storage in a smart grid: A coalitional game approach,” *arXiv preprint arXiv:1712.02909*, 2017.

[42] Bloomberg. (2017). Lithium-ion Battery Costs: Squeezed Margins and New Business Models, [Online]. Available: https://about.bnef.com/blog/lithium-ion-battery-costs-squeezed-margins-new-business-models/ (visited on 06/22/2018).

[43] C. Douris. (2017). The Bottom Line On Electric Cars: They’re Cheaper To Own, [Online]. Available: https://www.forbes.com/sites/constancedouris/2017/10/24/the-bottom-line-on-electric-cars-theyre-cheaper-to-own (visited on 08/21/2018).

[44] G. Government. (2017). Ergebnispapier der Begleit- und Wirkungsforschung: Bedarfsorientierte Ladeinfrastruktur aus Kundensicht, [Online]. Available: http://schaufenster-elektromobilitaet.org/media/media/documents/dokumente_der_begleit__und_wirkungsforschung/EP35_Studie_LIS_online.pdf (visited on 07/03/2018).

[45] F. Bizzarri, F. Bizzozero, A. Brambilla, G. Gruosso, and G. S. Gajani, “Electric vehicles state of charge and spatial distribution forecasting: A high-resolution model,” in *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, IEEE, 2016, pp. 3942–3947.

[46] F. Mwasilu, J. J. Justo, E.-K. Kim, T. D. Do, and J.-W. Jung, "Electric vehicles and smart grid interaction: A review on vehicle to grid and renewable energy sources integration," *Renewable and Sustainable Energy Reviews*, vol. 34, pp. 501–516, 2014.

[47] M. Zoglauer, G. Nischler, C. Gutschi, W. Süßenbacher, S. Otzasek, and H. Stigler, "Auswirkungen zukünftiger elektromobilität auf die österreichische elektrizitätswirtschaft," in *Alte Ziele-Neue Wege*, Verlag der Technischen Universität Graz, 2010.

[48] C. A. R. Board. (2017). PEV Types, [Online]. Available: https://www.driveclean.ca.gov/pev/Plug-in_Electric_Vehicles/PEV_Types.php (visited on 06/18/2018).

[49] O. Veneri, *Technologies and applications for smart charging of electric and plug-in hybrid vehicles*. Springer, 2017.

[50] P. Mell, T. Grance, *et al.*, "The nist definition of cloud computing," 2011.

[51] S. Bera, S. Misra, and J. J. Rodrigues, "Cloud computing applications for smart grid: A survey," *IEEE Transactions on Parallel & Distributed Systems*, no. 5, pp. 1477–1494, 2015.

[52] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98–115, 2015.

[53] Microsoft. (2018). What is Cloud Computing? [Online]. Available: https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/ (visited on 08/21/2018).

[54] S. Tuffery, *Data mining and statistics for decision making*. Wiley Chichester, 2011, vol. 2.

[55] J. D. Kelleher, B. Mac Namee, and A. D'Arcy, *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT Press, 2015.

[56] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[57] M. A. Boden, *Artificial Intelligence*. Academic Press, 1996, ISBN: 1-78646-066-1.

[58] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.

[59] F. Chollet, *Deep learning with python*. Manning Publications Co., 2017.

[60] G. Zaccone, M. R. Karim, and A. Menshawy, *Deep Learning with TensorFlow*. Packt Publishing Ltd, 2017.

[61] J. Brownlee. (2017). What is the Difference Between Test and Validation Datasets? [Online]. Available: https://machinelearningmastery.com/difference-test-validation-datasets/ (visited on 05/16/2018).

[62] P. Mirowski, S. Chen, T. Kam Ho, and C.-N. Yu, "Demand forecasting in smart grids," *Bell Labs technical journal*, vol. 18, no. 4, pp. 135–158, 2014.

[63] S. Raschka, *Python machine learning*. Packt Publishing Ltd, 2015.

[64] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[65] J. A. Hertz, *Introduction to the theory of neural computation*. CRC Press, 2018.

[66] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2012.

[67] S. Sharma. (2017). Epoch vs Batch Size vs Iterations, [Online]. Available: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9 (visited on 07/27/2018).

[68] G. Joechtl, *Digital Signal Processing - Script for lecture DSP1/2*. University of Applied Sciences Salzburg, 2012.

[69] C. Bishop, C. M. Bishop, *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[70] B. Hammer, *Learning with recurrent neural networks*. Springer, 2007, vol. 254.

[71] S. Sharma. (2017). Activation Functions: Neural Networks, [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (visited on 07/07/2018).

[72] A. Graves, "Supervised sequence labelling," in *Supervised sequence labelling with recurrent neural networks*, Springer, 2012, pp. 5–13.

[73] J. Brownlee. (2017). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (visited on 07/16/2018).

[74] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, Omnipress, 2011, pp. 265–272.

[75] F. E. Curtis and K. Scheinberg, "Optimization methods for supervised machine learning: From linear models to deep learning," in *Leading Developments from INFORMS Communities*, INFORMS, 2017, pp. 89–114.

[76] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[77] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[78] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, “Lstm: A search space odyssey,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[79] F. A. Gers and J. Schmidhuber, “Recurrent nets that time and count,” in *ijcnn*, IEEE, 2000, p. 3189.

[80] N. Reimers and I. Gurevych, “Optimal hyperparameters for deep lstm-networks for sequence labeling tasks,” *arXiv preprint arXiv:1707.06799*, 2017.

[81] J. Brownlee. (2017). How to use Different Batch Sizes when Training and Predicting with LSTMs, [Online]. Available: https://machinelearningmastery.com/use-different-batch-sizes-training-predicting-python-keras/ (visited on 07/23/2018).

[82] T. M. Breuel, “Benchmarking of lstm networks,” *arXiv preprint arXiv:1508.02774*, 2015.

[83] J. Brownlee. (2017). How to Use Timesteps in LSTM Networks for Time Series Forecasting, [Online]. Available: https://machinelearningmastery.com/use-timesteps-lstm-networks-time-series-forecasting/ (visited on 08/21/2018).

[84] J. Mira and F. Sandoval, *From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks, Malaga-Torremolinos, Spain, June 7-9, 1995: Proceedings*. Springer Science & Business Media, 1995, vol. 930.

[85] H. Rob and A. George. (2017). Evaluating forecast accuracy, [Online]. Available: https://www.otexts.org/fpp/2/5 (visited on 07/27/2018).

[86] J. Zacharias, M. Barz, and D. Sonntag, “A survey on deep learning toolkits and libraries for intelligent user interfaces,” *arXiv preprint arXiv:1803.04818*, 2018.

[87] Keras. (2018). Keras: The Python Deep Learning library, [Online]. Available: https://keras.io/ (visited on 06/01/2018).

[88] Tensorflow. (2018). About TensorFlow, [Online]. Available: https://www.tensorflow.org/ (visited on 06/01/2018).

[89] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, “Tensorflow: A system for large-scale machine learning.,” in *OSDI*, vol. 16, 2016, pp. 265–283.

[90] G. Van Rossum *et al.*, "Python programming language.," in *USENIX Annual Technical Conference*, vol. 41, 2007, p. 36.

[91] M. Wichmann. (2018). Python Documentation, [Online]. Available: https://wiki.python.org (visited on 06/16/2018).

[92] Anaconda. (2018). Anaconda Cloud, [Online]. Available: https://www.anaconda.org (visited on 06/16/2018).

[93] pydata.org. (2018). Python Data Analysis Library, [Online]. Available: https://pandas.pydata.org/ (visited on 05/16/2018).

[94] W. McKinney, *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.", 2012.

[95] H. John, D. Darren, and F. Eric. (2018). Matplotlib Documentation, [Online]. Available: https://matplotlib.org/ (visited on 07/03/2018).

[96] scikitdevteam. (2018). scikit-learn documentation, [Online]. Available: http://scikit-learn.org/stable/ (visited on 07/03/2018).

[97] Jupyter. (2018). Jupyter documentation, [Online]. Available: http://jupyter.org/ (visited on 07/03/2018).

[98] nyc.gov. (2017). TLC Trip Data, [Online]. Available: http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml (visited on 07/02/2018).

[99] pecanstreet.org. (2017). Pecan Street, [Online]. Available: https://www.pecanstreet.org (visited on 07/16/2018).

[100] dataport. (2017). Pecan Street Data, [Online]. Available: https://dataport.cloud/ (visited on 07/16/2018).

[101] J. Brownlee. (2017). How to Convert a Time Series to a Supervised Learning Problem in Python, [Online]. Available: https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/ (visited on 07/27/2018).

[102] J. Brownlee. (2017). How to Diagnose Overfitting and Underfitting of LSTM Models, [Online]. Available: https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/ (visited on 08/21/2018).

[103] Uber. (2018). Uber Movement, [Online]. Available: https://movement.uber.com (visited on 08/26/2018).