Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Energy consumption forecasting in energy management systems

Tung Dinh

Course of Study:	MSc Information Technology
Examiner:	Prof. Dr. Marco Aiello
Supervisor:	Dr. Ilche Georgievski
Commenced:	July 01, 2020
Completed:	January 04, 2021

Acknowledgement

It was an honor of mine to work on my master thesis with the Department for Service Computing. I would like to thank my supervisor Dr. Ilche Georgievski and Prof. Dr. Marco Aiello for accepting my proposal and for giving me this opportunity to work on this topic. To my supervisor, I would like to express my wholehearted appreciation for your decisive advice and constant support. I also want to thank my family, my girlfriend for their time and encouragement when needed during my study and master thesis at the University of Stuttgart. And to my friends, the ones who were willing to support me in the fulfillment of this thesis regardless of the time, thank you very much.

Abstract

Energy Management System (EMS) is emerging as a solution for the power grid management, which is indicated by the profit, stability, and reliability to its end users. Load forecasting is a component of an EMS, and it is vital to foresee the future demand to generate a proper schedule for a microgrid. A load forecasting study should cover all types of loads: commercial buildings and households, which has been incompletely studied in other papers. Since 2018, with the improvement in computing power, Deep Learning (DL) models have caught attention in many studies, and shown their potential over the traditional method Auto Regression Moving Average (ARIMA) in solving forecasting problems. In the DL branch, Recurrent Neural Network (RNN) is the type of model having memory, which is designed to estimate future values on history data. This thesis studies the application of RNN in forecasting energy consumption of both load types and implements a service that forecasts demand focusing on commercial buildings. We experiment with six different RNN-based models on two sets of data, commercial buildings in 2004 in San Francisco and households between 2011 to 2014 in London. The performance of the RNN family significantly exceeds ARIMA in accuracy and processing time. In two study cases, the reported error of the best performer in RNN-based models is 48% and 23% lower than ARIMA. The Forecasting service for commercial buildings is then implemented as a restful Application Programming Interface (API) for further deployment in an EMS.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ACF Auto Correlation Function. 44

ADAM Adaptive Moment Estimation. 29

AI Artificial Intelligence. 15

API Application Programming Interface. 39

AR Auto Regression. 32

ARIMA Auto Regression Moving Average. 11

BPTT Back-Propagation Through Time. 27

CEDs Controllable Electrical Devices. 35

CNN Convolutional Neural Network. 17

DERs Distributed Energy Resources. 22

DESs Distributed Energy Storages. 22

DL Deep Learning. 11

DNN Deep Neural Network. 17

EMS Energy Management System. 15

I Integration. 32

LSTM Long Short Term Memory. 17

MA Moving Average. 32

MAE Mean Absolute Error. 17

MAPE Mean Absolute Percentage Error. 17

ML Machine Learning. 15

NN Neural Network. 25

PACF Partial Auto-Correlation Function. 44

ReLU Rectified Linear Unit. 34

RNN Recurrent Neural Network. 15

SOA Service Oriented Architecture. 16

SOC Service Oriented Computing. 17

UDs Uncontrollable Devices. 39

1 Introduction

The energy sector has witnessed the growth in energy consumption, the contribution of renewable energy, and the invention and application of smart devices, which demands a revolution in power switching system of the power grid. The traditional power system shows the inability in managing a system with daily-growing in the use of renewable energy generators, such as wind turbines and solar systems; specifically, in 2018, it accounted for a total of 11% in primary energy consumption globally, and the number peaked in 2020 for the total of 28% [IEA] [Wikb]. To adapt to the challenge, many researchers have studied the design and simulated a fully functioning controlled grid [J L15] [SPL17] [LG16]. These studies are based on microgrids - small scale power grid, they have the capability and potential to reshape the communication, energy transportation, and transaction with the macrogrid while maintaining sustainability and reliability with the control of the Energy Management Systems (EMSs). With the availability of modern software, strategy, smart utilities, and renewable energy generators, a microgrid can optimize asset utilization, minimize the cost of operations and maintenance, and reduce carbon dioxide generation. In such microgrids, buildings are monitored and powered autonomously with pre-planned schedules by reliable energy providers with the optimal energy price.

Forecasting energy consumption is a crucial task of an Energy Management System (EMS) [GFA20]; however, little attention has been paid to study the energy consumption of a microgrid entirely. The emergence of Artificial Intelligence (AI) proposes many techniques in machine learning for this problem, unfortunately, it shows the inefficiency in generating forecast in comparison to the traditional technique Auto Regression Moving Average (ARIMA). Typically, the competition M3 [MH00] and many publications before 2017 claimed to gather poor results using Machine Learning (ML)-based models, such as linear regression, multiple linear regression, and support vector machine and can not be compared with ARIMA. This type of behavior can be understood as a reason for a limitation in applied ML models, where only linear relationships are recognized and real-life data is mostly non-linear. Aside from ML, Deep Learning (DL) promises to produce better results than both ML and ARIMA. In recent years, numerous of studies have investigated in forecasting energy demand [SXl17] [SOS18] [NSW19] [AJS19] [BT19a] and in other sectors [AN18]. Their results indicate that Recurrent Neural Network (RNN)-based models are dominant for the issue of time-series forecasting. Although the achievement in demand forecasting of these studies is high, they either consider a small selected dataset mainly from households or use simple DL models. Additionally, they show no clear reason why should DL models replace ARIMA.

Additionally, in the implementation of an EMS [GFA20], there is no module to forecast energy consumption; it is suggested that a module to forecast future demand should be studied. This thesis researches a wide range of RNN-based models to forecast energy consumption for both households and commercial buildings. With the focus on the models and the small scope of this thesis, we aim to implement one Forecasting service for commercial buildings to prove the feasibility of such a

service. Adopting the new service, the EMS can depict the more realistic scenario of a micro-grid, with consumption pattern and peak value, and consequently enhance the efficiency and correctness in quality of scheduling loads.

1.1 Problem statement

There are two main problems this thesis sorts out and focuses on (1) take advantage of recent DL models to forecast energy consumption and (2) enhance the implementation of an EMS with a forecasting service. It is generally known that there exist two popular approaches to tackle the forecasting situation, ARIMA and AI. Empirically AI methods have been proven to produce a profitable result in the scope of time series forecasting, specifically, RNN in DL branch. This type of model allows future values to be deduced from past values. However, in term of efficiency and cost, the traditional one ARIMA, which was proposed in 1970 by George Box and Gwilym Jenkins [Geo15], is surprisingly reported to overperform the modern techniques in some specific situations, where acquired data is small and requested forecast is in near-future [NSW19]. Noticing the power of DL in time series forecasting applications, which has gained outstanding results in many fields, we study the architecture and applications of recent DL models and compare them with ARIMA. However, the use in the energy sector is still insufficient, the majority of publications only focus on household datatype and using an old-fashion style of DL architecture design where only one typical model is experimented. The comparison with other statistical models barely appears, not to mention that ARIMA family is still highly accessible in modern software due to its simple properties and easy deployment. In this thesis, we propose an approach to solve the problem of forecasting energy consumption in a more general view. The experiments are executed on both types of buildings commercial and household, and use many DL models from other fields to support the demand forecasting. Then we attempt to prove the feasibility of a forecasting service in the Service Oriented Architecture (SOA)-EMS by implementing the Forecasting service for commercial buildings.

Inspired by the implementation [GFA20], in which, the authors simulate a functional microgrid and the management of an EMS on it. Despite having enough components for an EMS: generation service, weather service, demand side, pricing unit, optimization, and user interface and is capable of balance the demand and supply while minimizing the billings to end-users, its demand side is modeled on static data. Using static data, there are only constant history values used instead of forecast ones. Therefore, it is incompetent to work in real life, which proposes a challenge this thesis aims to achieve: forecasting energy demand. The experimental environment for this thesis is assumed that all required data for the new service is available and reliable.

1.2 Research questions

With the mentioned issues, this thesis explores the combinations of DL models in answering these questions:

- RQ1: Can we use RNN-based models to forecast the energy consumption of commercial buildings and households?

- RQ2: Does the performance of the RNN family beat ARIMA, considering both accuracy and execution time?
- RQ3: Can we improve the current EMS implementation with a new forecasting service for commercial buildings?

To address RQ1, we first examine the data of weather and holidays along with the energy consumption with the objective to unveil the dependency of energy on other factors and its pattern. We then propose to use some RNN-based models: RNN-Deep Neural Network (DNN), Long Short Term Memory (LSTM)-DNN, Convolutional Neural Network (CNN)-LSTM-DNN in forecast generation. For the second question, this thesis experiments with ARIMA model, and compare its results with the achievement of the DL models on metrics: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and execution time. Then the most efficient RNN-based model is deployed for building a forecasting service in question three. Since we study the energy consumption, and for that purpose, it is required the data of both types of buildings in a specific city for at least one year, the local weather and holidays. All experiments are conducted on a system with processor AMD FX6300 six-core and a graphic card Nvidia GTX970.

1.3 Contribution

By answering three research questions, this thesis puts an upgrade to the current implementation [GFA20] and fulfills the roles of an EMS. Its contributions are indicated as follows:

- Develop models to tackle the challenge of forecasting energy consumption.
- Develop the strategy to use weather and holiday data to support forecasting energy consumption.
- Forecast the energy consumption of both households and commercial buildings.
- By comparing with ARIMA, it shows the advantage of DL models over the traditional method.
- Develop the Forecasting service for commercial buildings.

This thesis is mainly inspired by the work of re-designing the architecture of an EMS [GFA20], hence we aims to develop the Forecasting service as a standalone module, which could benefit any EMS, that has Service Oriented Computing (SOC) design. Not only support the current implementation, but these contributions also play a considerably big role in supporting the management of an EMS and other related studies with the varieties of models and forecast values in days ahead.

1.4 Outline

Chapter two presents the insight into the technologies and background knowledge required to digest the terms and concepts written in this thesis. The third chapter refers to the related work in the field of energy management and a diversity of forecasting techniques to deal with grid-related and unrelated problems. The design of the Forecasting service and methodology to develop forecasting models and their workflow is discussed in chapter four. Chapter five provides the analysis of the data and the estimated performance of the algorithms; it investigates the features and correlations

inside data and ensemble methods, tactics, and components into the final concept. The comparison of the DL models with the benchmark model ARIMA are carried out in chapter 6, while a short discussion is provided for further understanding of the challenge. Finally, we summarize the whole report and provide tactics to further improve the service and forecasting method.

2 Background Knowledge

Chapter two delivers the necessary background of the whole thesis, starting by introducing to the power grid and the transition to be smart. Then, the small version of the power grid, which is called a microgrid, is explored and the importance of an EMS in a microgrid is reasoned, its current design and implementation [GFA20]. In the implementation, the limitation is pointed out and we state the need for developing a new Forecasting service. This chapter continues by presenting the forecasting algorithms comprehensively with their simplified mathematics operations.

2.1 Microgrids and energy management systems

The evolution of electricity was sparked in the late 1880s, which was the cornerstone for the development of other industries. A power grid is defined as a highly interconnected network, which delivers electricity from the producers to end consumers; it consists of generating stations, electrical substations, high voltage transmission lines, and distribution lines [Kap11]. With the advance in technology, the traditional electrical grid has been empowered with observing, managing, and communicating devices, transforming it into a smart grid. It inherits the building blocks and properties of an electrical grid, but with a wider set of utilities and devices including smart appliances, smart meters, and renewable energy resources. It has advantages in power production, power conditioning, and control distribution.

2.1.1 Microgrids

A smart grid represents an unprecedented opportunity to move the energy industry into a new era of reliability, availability, and efficiency that contributes to economic and environmental health. It provides users with the benefits of a more efficient transmission, quicker restoration after power disturbances, reduced management and operation costs, lower power costs for end-consumers, reduced peak demand, increased integration of large-scale renewable energy systems, better integration of customer-owner power generation systems, and improved security [Sma] [GG11]. Different from a traditional grid, where energy flows in one direction from generators to customers, a smart grid adopts the concept of a distributed energy system, where it has two-way energy and communication flows. It encourages customers to become active participants and it can also self-heal thanks to the rise of microgrids, microgeneration and renewable energy sources [PA13] [PA12] [FA19].

A microgrid is a crucial component, which enhances a smart grid [YOM+17]; it models a small-scaled power grid with independent control capability and aims to operate in a small area. According to Ton and Smith [TS12], a microgrid is “a group of interconnected loads and distributed energy resources within clearly defined electrical boundaries that acts as a single controllable entity with

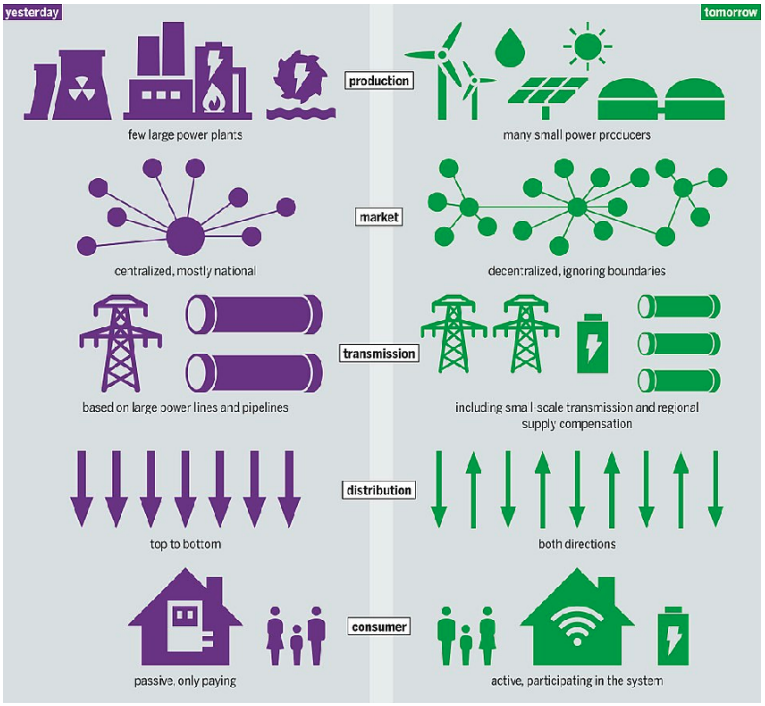

Figure 2.1: Traditional energy system versus smart grid [Wika]

respect to the grid. A microgrid can connect and disconnect from the grid to enable it to operate in both grid-connected or islanded-mode". Its electricity sources are distributed generators, batteries, and renewable energy sources like solar panels and wind turbines. It has the potential to run in island mode, which is a self-sufficient energy system and powers industry application, local energy network, and flexible electrical equipment. The benefits of a microgrid [YOM+17] is (1) reduces the greenhouse gas emissions by deploying more zero-emission electricity sources (2) operate locally and seasonably in case of a power outage (3) decrease the cost of new power generation by local management of power supply and demand and (4) under extreme weather conditions or cyberattacks, it aids to the resilience of a grid [Nye] [C2E].

Microgrids exposes the big capability to reshape the traditional power grid while boosting the concept of a smart grid. Therefore, they catch the attention of of both researchers and investors. This has become a hot topic in digitizing cities, and the software which schedules the operations of microgrids.

2.1.2 Energy management systems

An EMS is a decision-making unit, playing the heart of a microgrid; it coordinates the demand and supply in energy between the dispatchable generators and the loads and fulfills the economic and environmental objectives. It creates a market platform where energy and flexibility can be exchanged between suppliers and consumers while aiming to leverage low-cost energy storage, ensure fair reward for flexibility, and enable the use of more volatile renewable energy. It promotes the fulfillment of environmental and economic objectives.

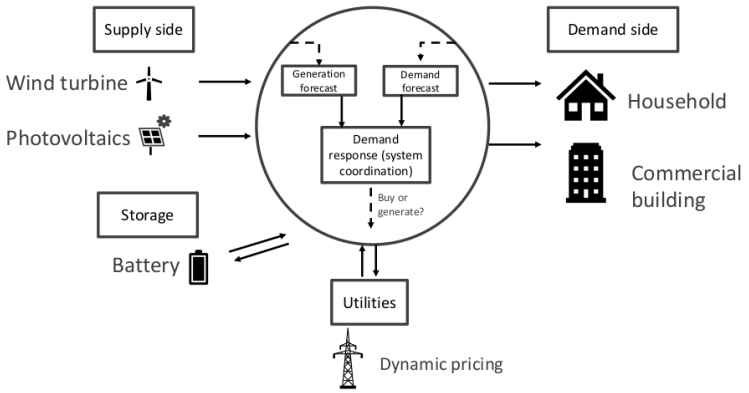

Figure 2.2: A microgrid [Ilc19]

There are two approaches to design and implement an EMS functionally, one is the traditional way, which is monolithic and another is the modern SOA. In a microgrid, the functions of an EMS are specifically:

- Play as the centralized coordinator
- Control flexible loads
- Use locally generated electricity
- Store electricity
- Decide on buy or sell eletricity with the main grid - macrogrid

Traditional energy management systems

Early work on EMSs falls into the monolithic design, in which all functions of an EMS are combined, implemented, and deployed as one block of the program. Since it is single-tiered, this style of software is highly coupled, and poses several limitations in understanding, upgrading, and maintaining. Shin et al. [SPL17] propose implementation of microgrid web services, in which their EMS followed the idea of loosely coupling, however, the functionality of the EMS was not modularized; similarly, Lee E.-K. and Gadh R.and Kim [LG16] introduced an EMS in a microgrid satisfied all required roles of an EMS and it can be an application in small scale for smart homes; and a management software was developed to cope with the energy flow in a house using PLCs [HCP+14]. They all perform well and report to have positive results, however, as their EMSs are cohesively designed, which hinders the opportunity for them to be scaled both vertically and horizontally.

With the traditional EMS, its functionality is restrictedly guaranteed within its application. As the need for expanding a microgrid, adapting more technologies, concepts, and devices become utterly arduous. Under programming views, the monolithic design is not designed to be reused and continuously modified. Being single of truth, one small bug could badly damage the EMS, and disable the microgrid. Moreover, adding components and/or functions will require a full understanding of the software and its dependencies, and facing the problem of scaling. Since more

and more research related to microgrids, latest technologies, regulations, utilities, and devices have migrated into the EMSs. An action of inserting generators or loads may affect the whole EMS and cannot be fast handled. Its ability to adapt to these evolutions is bounded due to its complexity and tight coupling, an EMS cannot be slightly modified to match with the requirements for other microgrids, once being deployed, it remains practically unchanged. Despite certain achievement in the field, it imposes the urge to re-design the architecture of an EMS, so that it can cope with the constantly changing technologies, governance, and environment.

2.1.3 Service-oriented energy management systems

With the increasing complexity and demand for a highly scalable and robust EMSs, the traditional EMS design is not an ideal choice to be implemented. Adopting the idea of SOC [PG03], the new implementation of an EMS must have a fully decentralized architecture without a complex and expensive communication infrastructure. It improves system availability, customer data protection, and provide an interface for all grid users [GFA20]. The suitable software implementation style for an EMS [GFA20] is the SOA, instead of a tightly-coupled module previously suggested. The EMS of a microgrid consisting of Distributed Energy Resources (DERs), Distributed Energy Storages (DESs), load profile, weather service, and price service. It is the software operating, monitoring, maintaining, and optimizing the load, supply, and resources.

SOA is a software design that creates a network of services, and they can communicate with each other across platforms and languages; it enables the interoperation between heterogeneous software and technologies [ope]. Unlike the traditional design of an EMS, SOA introduces the implementation concept of loosely coupling, in which, the dependence of components is limited to the least practically. A service has four properties [Mic]:

- Represents logically a business activity with a specified outcome.
- Is Self-contained.
- Is a black box to customers, whose inner operations are not required to be understood.
- Possibly consists of other underlying services.

SOA can be beneficial in many aspects, it enhances the scalability of software to meet the desires of the business, boosts the development and integration of software while reduces the cost in an upgrade, modification, and maintenance. The main advantages of SOA are [Sar]:

- Reusable code or service reusability: software can be essentially a group of existing services on their functions; it hence enhances the diversification in creating software while cutting down the expense of development.
- Platform/language independence: its ability to combine many services into one complicated software, allowing interaction between services, which are written in different languages and run on different platforms.
- Easy maintenance: as each service is developed independently, it is cheap and simple to update and modify without affecting others.
- Scalability: services should be ideally stateless, hence it can easily run on any servers and make software to grow.

- Reliability: being modularized, each service can be separately tested and debugged, hence, the software is more reliable.

Taking the advantages of SOA, the monolithic EMS is functionally divided into smaller software. It is then designed as a set of loosely coupling and independent services, each one encapsulates one of its capabilities. SOA ensures easy maintenance, by that, services can evolve individually while the EMS is kept functioning uninterruptedly. And the EMS is not designed for any specific microgrid, inheritance, modification, insertion, and integration of services to facilitate another environment is promised and highly encouraged. The proposed architecture of the EMS [GFA20] is conceptualized and implemented as seven services:

- Weather service: collects the current and forecast weather data, then stores them into a database. It is implemented as a service using data from weatherbit.io [wea], and it provides the current and the forecast next 24 hours data.
- Price service: collects the lastest day-ahead prices from the energy market, then stores them into a database. It provides pricing for the current hour, current day, and the next 24 hours, and its data is collected from entsoe.eu [ent].
- DER service: models DERs, such as PVs and wind turbines, and batteries. Due to the weather dependency, it requires communication with the weather service to output the estimated energy production for the next 24 hours based on the pre-configured data of wind turbines and solar farms.
- Building service: model buildings and their loads. This service allowed users to input building and its belonging devices' parameters and modify them.
- Planning service: is the complete representation of our system model and intelligently manages the balance and demand-response plans. It is implemented as the core component of an EMS, and it performs the profit optimization task of the system.
- Coordination service: implements the business logic, which keeps the execution of services in the right order and time. It triggers all other services depending on the business logic and users' queries.
- Middleware service: as the name implied, it serves as an intermediary in communication and data management between the microgrid services and other participated components.

The current implementation of this centralized EMS following Figure 2.3, in which the main idea of loosely coupling of service is generalized in their software architecture. The limitation of this implementation locates in its Building service, instead of forecasting the energy consumption by component buildings, it uses constant data for simulation. It hence fails to implement a complete EMS as proposed by Georgievski et al. [GFA20], and it points out a demand for developing another service, which can predict the consumption of buildings, and that service is named Forecasting service.

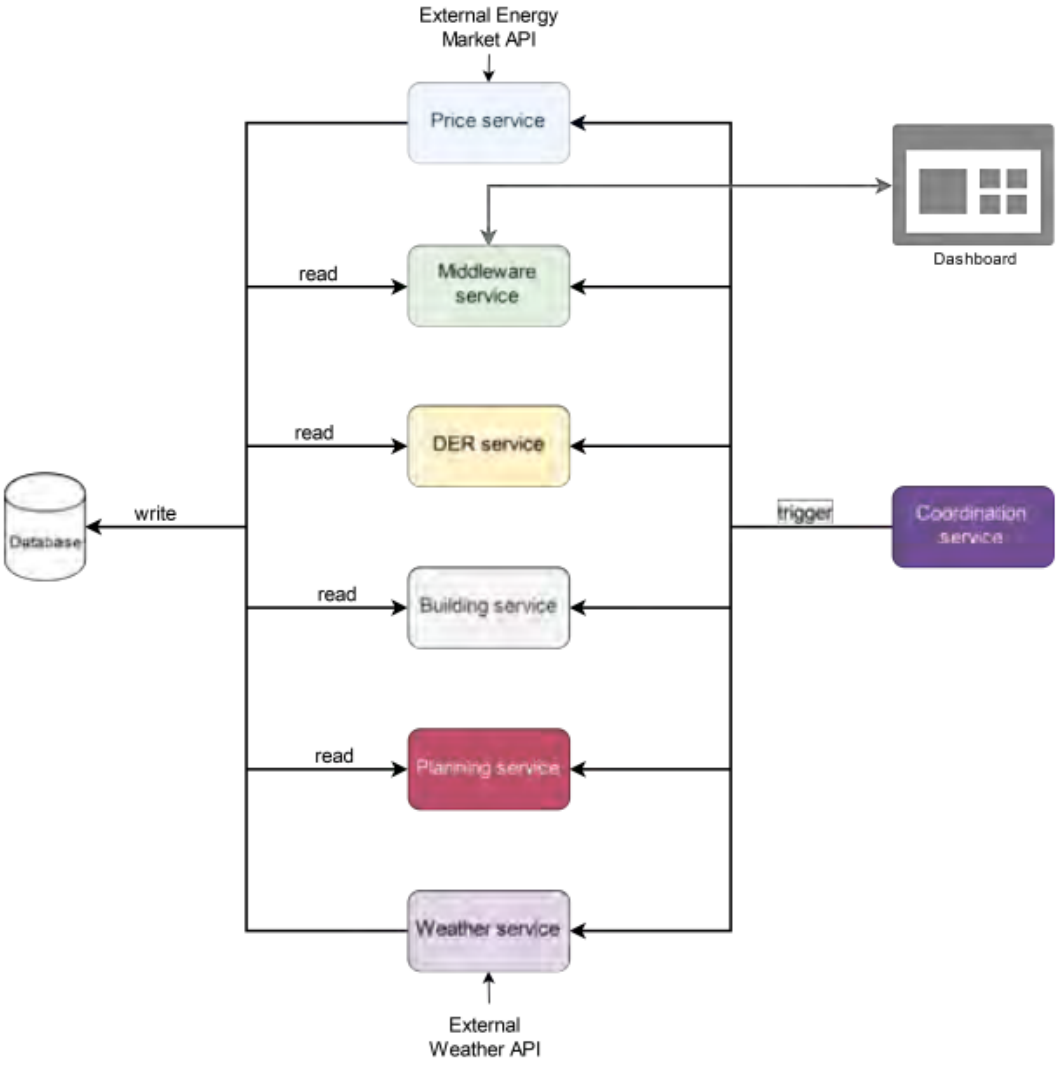

Figure 2.3: Overview of the service-oriented architecture of the EMS [GFA20]

2.2 Neural networks

Throughout history, human muscle work gradually gets replaced, since 1970, the industrial revolution has started with machines. Nowadays, AI targets to use machines for human intelligence job. AI appeared long ago since classical philosophers attempt to describe human brain function by symbolic system. However, until the late 1950s, it was formally named ärtificial intelligenceät a conference at Dartmouth College, in Hanover [MMP97]. Many scientists were positive about the potential of AI, Marvin Minsky, a MIT scientist wrote "the problem of creating 'artificial intelligence' will substantially be solved" [Mon14]. It was in reality never easy for AI to be fully developed, the field experienced 2 major winters from the 1970s to the mid-1990s, where the human race saw the fall of the market and research funding drop dramatically. Since then, the field has been expanding rapidly in diversity and practice. Moreover, the growing influence of AI on other fields besides computer science also grows in numbers of researches and applications.

2.2.1 Overview

Artificial neural network or Neural Network (NN) is the field of computing, which studies imitated neural networks inspired by the operation of a biological one. It is a set of algorithms, designed to recognize patterns for the task of prediction and classification. It is the foundation of countless important applications anti-spam email, speech recognition, object detection, and more.

In order for humans to perform cognitive processes such as memorization, attention, sensation, etc, they base decisions on the brain, which is a network of neurons or nerve cells. The nerve cells are linked to collect and pass electrical impulses from one neuron to the other, like the excitation to run the actions. The dendrites receive an impulse from the terminal button or a synapse of an adjoining neuron. Dendrites carry the impulse to the nucleus of the nerve cell which is also called soma. Here, the electrical impulse is processed and then passed on to the axon. The axon is the longer branch among the dendrites which carries the impulse from the soma to the synapse. The synapse then passes the impulse to the dendrites of the second neuron. Thus, a complex network of neurons is created in the human brain. A human neuron is presented in Figure 2.4.

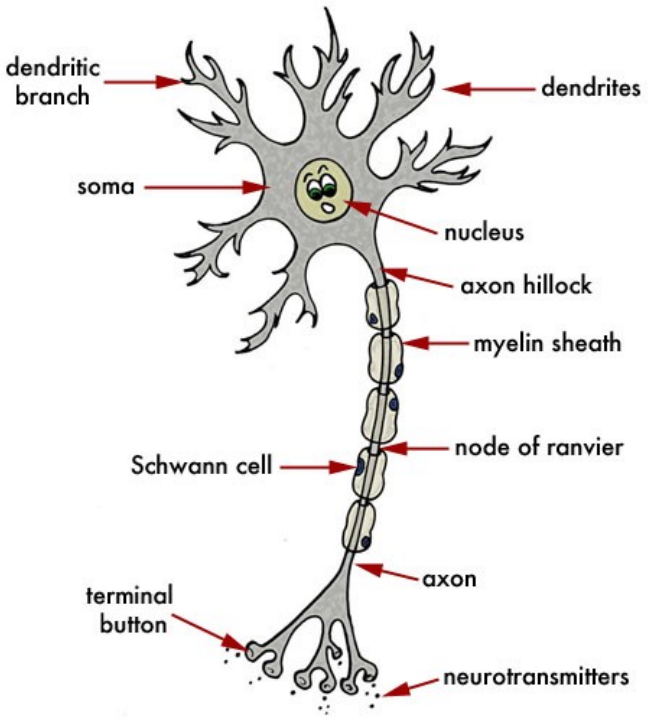

Figure 2.4: A human neuron [Shu18]

Consequently, a computational NN contains neurons, which are the calculation units, and edges, corresponding to biological axon-synapse-dendrite connections. The working mechanism of a neuron is similar to a human neuron, each neuron has inputs and a output, which can be sent to other neurons; in Figure 2.4, signals x_1, x_2 are transmitted by other two neurons, multiplied by weights w_1, w_2 and the final signal is:

$$y = f(x_1 w_1 + x_2 w_2$$

The weight determines the influencing strength from one neuron on another and adjusts the learning process. The output number y is non-linearly calculated on the sum of its inputs, which is the reason to make NN stand out from other methods: using a non-linear function, a NN can recognize the non-linear pattern in a signal.

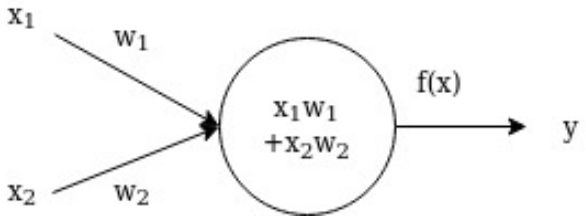

Figure 2.5: A neuron

Neurons can be connected in various patterns, allowing the output of some neurons to become the input of others, and have a wide range of application. Usually, neurons join to make a layer, and many layers stack into a directed and weighted network. Figure 2.6 takes an example of a neuron network containing one input layer, one hidden layer, and one output layer. Conventionally, it is called one-layer NN.

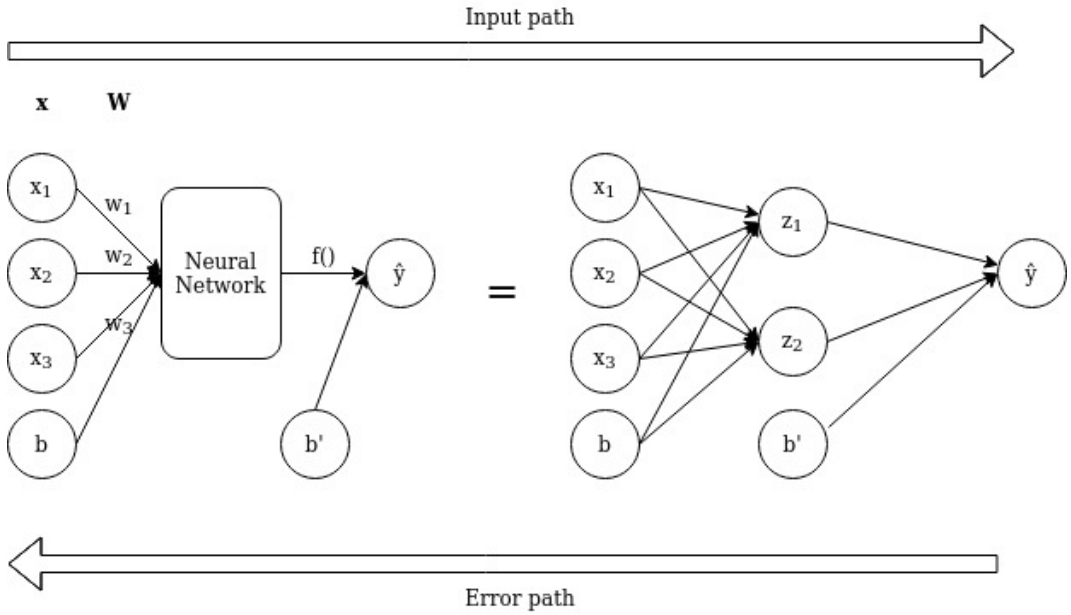

Figure 2.6: A neural network

The output of a NN is computed as:

$$\mathbf{z} = f(\mathbf{W}^T \mathbf{x} + b)$$

$$\hat{y} = f'(\mathbf{W}^T \mathbf{z} + b')$$

where f and f' are the non-linear activation functions, determining the output from the combination of input vector $\mathbf{x}$, weight matrix $\mathbf{W}$ and constants named bias b and b'. The extension work on the explanation for the activation functions is covered in Appendix A.1. NN has performed mostly all

tasks that conventional algorithms had little success with. Expectedly, it raises the awareness in NN application, reoriented research, and job market, while bearing positive empirical results. Training steps for a NN can be listed as:

- Give the NN some inputs to calculate some outputs - forward propagation
- Measure the loss function for gradient
- Perform Back-Propagation Through Time (BPTT)
- Update Weight matrices
- Repeat for other inputs

2.2.2 Gradient descent

As can be seen from the output Section 2.2, the weight matrix plays a key role in output decision; using $\mathbf{W}$, a NN can know the contribution of each input, to generate output. Gradient descent is the most common algorithm in optimizing a NN, which means to follow the direction of the gradient. Its target is to minimize the loss function $J(\mathbf{W})$, and it is formulated as $\mathbf{W} = \mathbf{W} - \alpha\nabla_{\mathbf{W}}J(\mathbf{W})$; the loss function can be understood as the difference between the actual and the prediction [And20]. Analyzing the above formula, α is the learning rate, which is used to regulate the amount of weight needed to be updated. To be the optimized one, a model has to repeatedly run through its training dataset, and update its weights with gradient descent; epoch is the number of loops a model goes through. In each epoch, the gradient is calculated as the first-order derivative of function $J(\mathbf{W})$ with respect to its weights.

To visualize the gradient, Figure 2.7 shows function $J(\mathbf{W})$ as a second-order polynomial of weights $\mathbf{W}$ $J(\mathbf{W}) = \mathbf{W}^2$, the gradient of $J(\mathbf{W})$ is $\nabla_{\mathbf{W}}J(\mathbf{W})$. Supposedly, there are two red points on the curve, to get to the global minimum, they have to go in the direction of $\nabla_{\mathbf{W}}J(\mathbf{W})$, and the learning rate α decides how big a step should they take. When $\nabla_{\mathbf{W}}J(\mathbf{W}) < 0$, gradient heads towards the positive side of $\mathbf{W}$, and to negative side of $\mathbf{W}$ when $\nabla_w J > 0$. Learning rate is a constant, multiplied with the derivative $\nabla_{\mathbf{W}}J(\mathbf{W})$ in each epoch, to determine the size of the step to reach the local minimum. A small learning rate results in slow convergence while a big one causes overshoot [And20].

Learning is the network adaptation to better predict the outcome from observed samples, it involves the adjustment of the weights to improve the accuracy of the result, by minimizing the prediction errors. There are two iterative steps in the training process of a NN: feed-forward and back-propagation. The feed-forward path follows the input path, from the input layer to the output layer, where each neuron is activated by a non-linear activation function f on the summation of its previous neurons' outputs and bias. The product of feed-forward propagation is the prediction, written as $\hat{y}$, then the error results in the difference with the actual y. The back-propagation path is used to adjust the weights $\mathbf{W}$ by performing gradient descent of propagated error on weights, the weights are after that get updated. Learning is considered complete when the model examines additional observations, which are called test samples, and does not usefully reduce the error rate. After learning, The error rate typically does not reach 0, however, the model expects to see constant declines error in the learning process; this behavior is named convergence. In this thesis, we use Mean Absolute Error (MAE) = $\frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n}$ and Mean Absolute Percentage Error

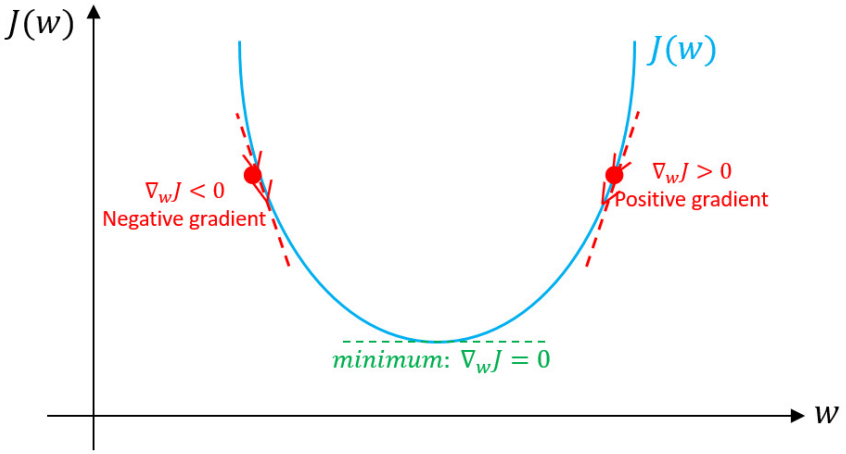

Figure 2.7: Gradient descent

(MAPE) = $\frac{1}{n}\sum_{i=1}^{n} \mid \frac{y_i - \hat{y}_i}{y_i} \mid$ to calculate the accuracy of the forecasting models. The work on process explanation with mathematics proof is provided in chapter two of the book "Fundamentals of the New Artificial Intelligence"by Munakata [Mun08].

Gradient descent variants

Being the most popular optimization technique in a NN, gradient descent is easy to use and to train. It guarantees to converge to the global minimum in case of the convex loss function and a local minimum in the non-convex one. However, it still has two main disadvantages (1) the training time is huge due to all training samples are trained at the same time, and (2) some training samples may be redundant [Z2L].

Depends on the number of training samples used to compute the loss function, there are three variations of gradient descent; they are batch, stochastic, and mini-batch gradient descent. Batch gradient descent aka gradient descent uses the total training samples in one epoch to update the weights. Stochastic gradient descent upgrades the weights for each training sample. Assuming to have m training samples, stochastic gradient descent runs m times in one epoch, therefore the training time is significantly smaller but the gradient will fluctuate drastically due to the separate update of each sample. And the last one mini-batch gradient descent, it trains faster than batch gradient descent and fluctuates much lesser than stochastic gradient descent because of inherited advantages of the previous two.

Optimizing gradient descent

In the following, two algorithms for gradient descent optimization used in the thesis are outlined, gradient descent with momentum and Adaptive Moment Estimation (ADAM). Momentum introduces another force helping to speed up the convergence of gradient and overcome the trap of local minima [Unk]. ADAM is one of the best optimizers and is widely implemented in numerous pieces of research. Unlike other approaches, ADAM changes its learning rate α adaptively with respect to its parameters during training. With its adaptive scheme, it is reported to "computationally efficient, has little memory requirement, invariant to the diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters"[KB14].

2.3 Recurrent neural networks

"Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence." - Christopher Olah

RNN is a member in the NN family, unlike a vanilla NN, the basic NN, which is introduce in Section 2.2.1. In a vanilla NN, the inputs and outputs are independent and identically distributed, RNNs can model the temporal behavior and forecast output on the past inputs. As a part of RNNs, while learning the parameters, they remember the previous state and uses it as input for the next cell. In real life, we encounter countless problems related to sequential events, where the decision process requires both learned parameters and previous inputs. Here, memory is what makes a RNN extraordinary to a vanilla NN or a CNN in solving these problems of forecasting time series data, such as energy consumption, sales, and natural language processing.

In 1982, a formulation of a recurrent-liked network was presented by John Hopfield on the context of neuroscience. This places the cornerstone for the birth of RNNs in the work of David Rumelhart in 1986 in order to learn the underlying associative memory in a network [Hop82]. Up to now, their work is still important in neuroscience. The term "Recurrent" stands for the same structure in the model, which means a recurrent cell uses the shared parameters **V**, **U**, and **W**, the whole network operates on different inputs, however, its weights are the same. One main advantage is the model has fewer parameters to learn in comparison to a vanilla NN. Referring to the unfold architecture of RNN in Figure 2.8 for the convenience of formula expression.

The Figure 2.8 shows the architecture of a RNN and its unfold form, with parameter abbreviations:

- T: number of layers
- **x**: input vector and **x_t** is the input at time step t
- **h**: hidden vector and **h_t** is the hidden state at time step t, and is seen as the memory for the next cell
- **y**: ground truth output vector, and **y_t** is the ground truth at time step t
- $\hat{\mathbf{y}}$: predicted output vector, and $\hat{\mathbf{y}}$**_t** is the output at time step t
- **V, W, U** are the weight matrices

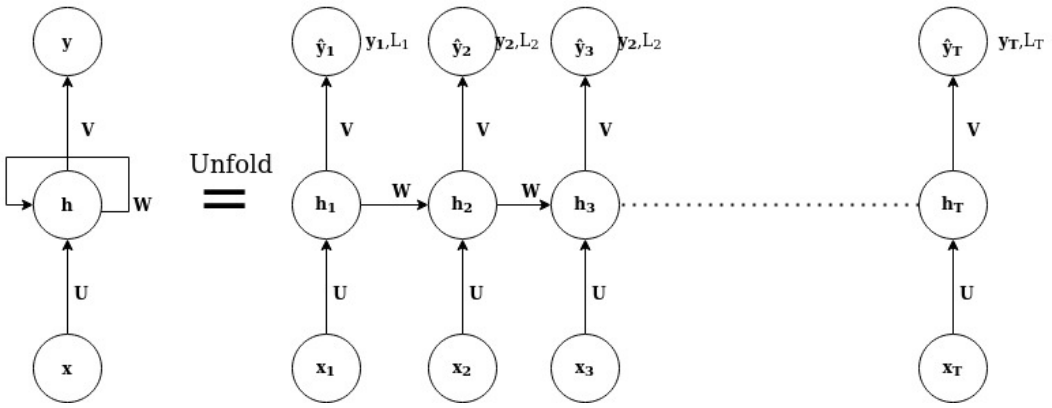

Figure 2.8: Fold and unfold architecture of a Recurrent Neural Network

In the Figure 2.8, it is shown that, recursively, a RNN computes the current output at time t y_t by the past hidden state and current input h_{t-1} and x_t. It explains the capability of a RNN in estimating future point by history. A RNN is trained through a process called BPTT, in which the weight matrices is learned with gradient descent technique in Section 2.2.2, hidden state and output at time t are defined as:

$$\mathbf{h_t} = g(\mathbf{W h_{t-1}} + \mathbf{U x_t})$$

$$\mathbf{y_t} = g^*(\mathbf{V h_t})$$

In which g and $g*$ are two activation functions, and are not necessarily the same. The comprehensive mathematics presentation is covered in the Appendix A.2. Related to the task of forecasting numerical values, the basic linear regression method would work well on; it is a basic algorithm, yet a common and powerful method. However, linear regression is a mapping function of inputs to output following a linear fashion, and it is mostly simple and small in scale. RNN is a more complicated method, which can identify the nonlinear relationship with the help of nonlinear activation functions, but it is costly to train and requires large and clean dataset. The activation is detailed in Appendix A.1.

2.4 Long short term memory

Practising RNN is utterly problematic due to its vanishing and exploding gradient [Raz13] [Yos94], the fact that, the weights of a RNN are learned by the recursive gradient, which is multiplied recursively in training. As a result, its gradient is exponentially proportional to time, which means it can only learn short-term dependency and suffer in long term. An alternative architecture is therefore required in the applications. From the derivative function of the hidden state Section 2.3, in case of gradient less than 1.0, the further it goes backward, the smaller it becomes or the larger it becomes if the gradient is bigger 1.0. This situation causes difficulty for the network to memorize far nodes in the sequence and hence predictions are solely based on only the most recent ones. LSTM is a variation of RNN, while its field of applications stay the same, LSTM however introduces

"gates"to manipulate memory inside a cell, making it easier to remember past data. Gates allow which data to output, which data to hold for the current state and can work with in-predetermined time lags.

In a LSTM cell, there are three gates: input, forget, and output gates, and their values are in the range $[0, 1]$. LSTM was first proposed by Sepp Hochreiter and Jürgen Schmidhuber in 1997 by introducing Constant Error Carousel (CEC) units, and to deals with the vanishing gradient problem in RNN [HS97][HS96]. Initially, a LSTM cell includes input and output gates, and later in 1999 Felix Gers, Jürgen Schmidhuber, and Fred Cummins introduced the forget gate to LSTM [GSC00]. A LSTM is depicted as in Figure 2.9.

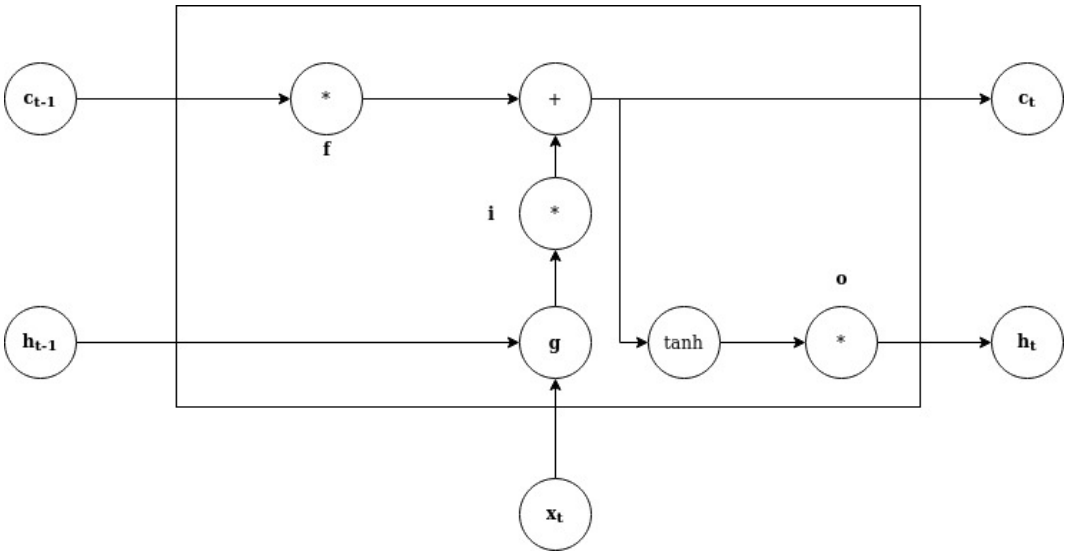

Figure 2.9: LSTM cell

The presented functions below are simplified, to be specific: they are filtered of bias term, for the ease of explanation and compatible with the presentation of RNN in Section 2.3, in which c_t is the output or memory of the cell at time t, h_t is the hidden state at time t.

$$\mathbf{c_t} = \mathbf{f} \otimes \mathbf{c_{t-1}} + \mathbf{i} \otimes \mathbf{g}$$

$$\mathbf{h_t} = \mathbf{o} \otimes tanh(\mathbf{c_t})$$

Input gate: determines which input value should be kept as the current memory. Sigmoid activation function decides values range to be $[0, 1]$. There is also a transition state g, and its value ranging from -1 to 1, deciding the weightage of input x_t and hidden state h_{t-1}

$$\mathbf{i} = \sigma(\mathbf{z_i})$$

$$\mathbf{z_i} = \mathbf{W_i h_{t-1}} + \mathbf{U_i x_t}$$

$$\mathbf{g} = tanh(\mathbf{z_g})$$

$$\mathbf{z_g} = \mathbf{W_g h_{t-1}} + \mathbf{U_g x_t}$$

Forget gate: determines which detail of the previous inputs to be discarded from memory, and its decision is also made by Sigmoid function. Forget gate takes consideration of the the previous state h_{t-1} and the current input x_t, and outputs a number between 0 and 1 for the cell state c_{t-1}

$$\mathbf{f} = \sigma(\mathbf{z_f})$$

$$\mathbf{z_f} = \mathbf{W_f h_{t-1}} + \mathbf{U_f x_t}$$

Output gate: decides the output on the current input and previous state. With Sigmoid function, gate value is in the range $[0, 1]$.

$$\mathbf{o} = \sigma(\mathbf{z_o})$$

$$\mathbf{z_o} = \mathbf{W_o h_{t-1}} + \mathbf{U_o x_t}$$

It can be seen, that LSTM is more complicated than RNN, with eight weight matrices $\mathbf{W_i}, \mathbf{U_i}, \mathbf{W_g}, \mathbf{U_g}, \mathbf{W_f}, \mathbf{U_f}$ and $\mathbf{W_o}, \mathbf{U_o}$ to be trained, while two in RNN. This section pictures the simplied LSTM, in order to show the key behind. The details on LSTM can be found in [JB14] and in [Chr15].

2.5 Auto regression moving average

ARIMA (p, d, q) is the method of forecasting value, is a simple stochastic time series model, which can train and forecast future time values. It can capture complex relationships by taking lagged term Auto Regression (AR)(p), differentiation step Integration (I) (d), and error terms Moving Average (MA)(q) to generate output. ARIMA was first proposed as Box-Jenkins Model in the publication "Time Series Analysis: Forecasting and Control 5th Edition"(2015) [Geo15], by using the idea of regression as forecasting methodology. By words, ARIMA is formulated as: $futurevalue = constant + LinearcombinationLagsofY$ (upto p lags) $+LinearCombinationofLaggedforecasterrors$ (upto q lags) [Sel20].

I(d) determines d times the data should be differentiated to be stationary, as it is the first step in processing data. Normally, the input data is not mean-variance stationery, which creates difficulty examining and forecasting. And it is described as:

$$y_{t+1} - y_t = b'_0 + b_1(y - y_{t-1}) + b_2(y_{t-1} - y_{t-2}) + \dots$$

in which, b_0, b_1.. are the weights betas generated in training, and y, y_{t-n} are the output and lagged output at time t. The integral component implies, that the future values are the linear function of the changes in past values. By applying differentiation on the data, the model gets more robust, as statistical properties do not vary with the time stamp samples are taken.

AR(p) determines the target value based on the regression of its own p past values, it is written as:

$$y_t = \alpha + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_p y_{t-p} + \epsilon_t$$

As AR is modeled as a linear function, it is necessary to have uncorrelated and independent data for the predictor to best perform.

MA(q) decides the forecast values on the lagged error of the q past values. Similar to AR, MA is a linear function, representing the deviation between the forecast and the ground truth values.

$$y_t = \mu + \phi_1 e_{t-1} + \phi_2 e_{t-2} + \cdots + \phi_q e_{t-q}$$

MA is used for the compensation of the forecast value. Since the model is linear, it exhibits error in predicting new value, which needs to be balanced.

The overall formula for ARIMA is:

$$y_t = c + \sum_{i=1}^{p} \beta_i y_{t-i} + \epsilon_t + \sum_{i=0}^{q} \phi_i \epsilon_{t-i}$$

where c is the total constant between AR and MA functions.

Training ARIMA steps are itemized as:

- Choose a dataset, and analyse whether the data is stationary with time
- Differentiation until data is stationary
- Estimate model parameters by Autocorrelation and Partial autocorrelation
- For each (p', d', q') in the combination of (p, d, q), loop to train:
  - Fit model ARIMA(p', d', q')
  - Generate prediction
  - Update history
- Calculate Loss

2.6 Challenges with neural networks

High-Dimentional

Bellman first introduced the term "The curse of dimensionalityïn 1961, which means "the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with respect to the number of input variables of the function". In simple words, a typical NN has many layers, each contains many neurons, resulting in the growth in the number of parameters [GBC16]. It then becomes extremely high-dimensional, compared to other traditional ML algorithms.

Gradient vanishing and exploding

These phenomenons appear in the training process of RNN, specifically, in BPTT, the RNN tries to learn from the mistaken predictions using the gradient descent technique. The goal of this optimization is to bring the next prediction near to its ground truth by adding adjustment to the weight matrices. Based on the chain rule in performing BPTT, the gradient is multiplied in propagating from one neuron to another, therefore if the gradient is smaller than 1.0, it will shrink in long term multiplications and if the gradient is greater than 1.0, it will explode to infinitive. The consequence of gradient vanishing inhibits RNN to learn from far neurons, it can only learn from the nearest ones when the gradient is near 0 [Raz13].

The solution to exploding gradient is simple and effective such as having a predefined threshold. However, the uncertainty of vanishing gradients is difficult to handle; instead of reducing the weight matrices, a more preferred solution is to use Rectified Linear Unit (ReLU) instead of tanh or Sigmoid activation functions. The ReLU derivative is a constant of either 0 or 1, so it is unlikely to suffer from vanishing gradients. Another popular solution is to use alternative architecture LSTM instead of RNN.

2.7 Other terms

Time series data represents many data points in time order, which are successively collected. It shows the changes and pattern of data over time. It hence has wide applications, such as: finance, sale, COVID infections and energy consumption. The two main properties of a time series are trend and seasonality [Ail].

DNN is a term to describe a vanilla NN, which has multiple fully connected layers. It is the basic model in DL, and it appears in many applications, from basic to complicated ones.

CNN is one type of model in DL, it was first introduce in the 1980s by Bengio and Lecun [BL97]. Since then, it has become one of the most popular and powerful NN. It is designed to work specifically with images, as the name implied, CNN performs convolution with its filters to extract typical feature on an image.

3 Related work

The thesis is inspired by various work of other researchers in the architecture of EMSs, forecasting techniques, and the applications of forecasting in other fields, for examples: weather and stock forecast. This chapter presents some pieces of related work, their findings, limitations, and contributions to the idea of this thesis.

The balance in supply-demand has greatly changed due to the rise in energy consumption, the adaptation of renewable energy sources, and the pressure of the environment and economic issues. On the smaller scale, a microgrid is the standard for the power grid simulation, and it is run by an EMS. Design of an EMS has been studied and implemented in many publications [SPL17] [M A15].Generally, they described and designed a functional EMS, and identified essential components of it, such as DERs, energy storage, and Controllable Electrical Devices (CEDs), however, they failed to solve the non-functional side of an EMS due to the single-tiered architecture which prevents it from scaling. Other studies proposed the SOA approach to feature an SOA-EMS [GFA20] [LG16] [HCP+14]. Based on SOC, Han et al. [HCP+14] divide an EMS into four chunks: web services, data collecting and monitoring, data management and analysis and optimal operation, while Lee E.-K. and Gadh R.and Kim [LG16] demonstrate another SOA-based version of an EMS, which has three services: grid side, microgrid platform and customer side. Both take into account the functional requirement of an EMS and engineering challenge in expanding the software, however, the functions of an EMS are still tightly coupled. Noticing that challenge, another study tries to decentralize an EMS into single functional services; the authors redefine the previous designs of an EMS, while ensure the capability of a microgrid [GFA20]. The new architecture promotes flexibility in development and maintenance; here, each function of an EMS is modeled by a single service. They implement a prototype featuring a microgrid in Germany; it has a variety of energy sources, non-residential buildings, and optimization service which is responsible to manipulate the CEDs and optimize the economic concerns. However, they do not fully introduce a forecasting service, which can foresee the future demand of a microgrid, and another building type households is ignored.

Time series forecasting is an attractive research area in many applications, including energy consumption. Generally, ARIMA is the to-go model for such problem, in which future values are estimated on the causality with the past ones. However, there is a limitation within ARIMA, it could only perfectly fit with linear data, which means it suffers the non-linearity and randomness, which is a big trouble, especially in finance [SN18] [STS18] [KBKK96]. The situation in energy consumption is quite relaxed since the distribution depends on fewer factors than the stock market, which makes it is more stable. Therefore, ARIMA and its variations are still powerful in these applications. Since 2017, the revolution in computing hardware directly has pushed the research and development of AI. Numerous of studies have shown the possibility in forecasting with DL, a group of algorithms which can model nonlinear relationship while require less work to process and feature data. And the most suitable ones are RNN-based.

Nugaliyadde et al. [NSW19] attempt to forecast energy consumption using DNN, RNN, LSTM and ARIMA. These models are used to predict demand for a single building and a block of buildings from one day up to to 13 months ahead. The results show that there are only two cases that ARIMA is only slightly better than RNN and LSTM and it is one-day ahead forecasting for a single building and a block of buildings. In other situations, both RNN and LSTM outperform ARIMA. The authors present the possibility to replace ARIMA by other DL models, however, there are several limitations in their approach (1) only a selected set of buildings are considered in the experiments (2) only a block of houses in London is involved in tests and (3) the used models are the vanilla version of them, which means they are used as stand-alone.

Al Khafaf et al. [AJS19] apply LSTM in solving the task of forecasting energy demand in 3 to 15 days ahead. The authors use the data of 609 houses from 09.03.2015 to 08.03.2016 in Victoria, Australia. They propose to cluster energy consumption into four groups by temperature, which reduces the amount of work for forecasting. The practice of clustering data brings good results with an error of only 3.15%. But their experiments do not cover a wide range of data, as they use only one-year data, which might not express the true behavior of the buildings. And they only focus on using a vanilla LSTM model for this task. Similarly, Bedi and Toshniwal [BT19a] also cluster energy consumption data into three groups. However, the clusters are based on the similarity between consumption patterns between months and seasons. Using LSTM to forecast total annual energy consumption of the city UT Chandigarh, India in 2013, it is proven that can achieve high accuracy for non-linear complex data. Since this publication uses the total energy consumption of a city, the understanding of the energy demand of both households and non-residential buildings might not be entirely true. Besides, it shares a similar limitation in the length of data as in the previous one.

Bedi and Toshniwal [BT19b] present another way to use LSTM to forecast day total energy consumption in Korea in 2017-2018, they stack LSTM on DNN. By observing, the authors point out that, the temperature is also a great influence on electricity usage. Typically, people tend to consume more electricity in too hot or too cold weather. Using temperature as a supplementary factor for forecasting and the model LSTM-DNN return positive results. In conclusion, the authors claim that the model LSTM-DNN surpasses a single DNN. Because the Korean total energy consumption is involved in experiments, the energy consumption patterns and accuracy of forecast demand for commercial buildings or households might not be seen accurately.

Next, another study in an effort to forecast energy consumption of one residential customer in India from 2006 to 2010 [MAM16]. The authors target to forecast building energy consumption in a resolution of one minute and one hour. The experiments report that one-layer LSTM fails to obtain accurate predictions, but the situation improves greatly with two-layer LSTM. They prove the feasibility of two-layer LSTM in forecasting energy demand. Despite the low error rate of approximately 6.5%, the models need to be tested on different types of buildings and a bigger dataset. And here, the forecasting models are used as the vanilla ones.

To the contribution of CNN in forecasting, Kim and Cho [KC18] propose the use of the hybrid model CNN-LSTM-DNN in predicting household power consumption. The authors use an individual household dataset from 2006 to 2010. The results claim that the hybrid model achieves quicker and more accurate results than other tested models, including the model LSTM-DNN. However, using a private dataset might be biased in the application for other datasets. Additionally, Benidis et al. [BRF+20] suggest a transition from the old model structure to the new one. In the old structure, the majority of applied models are implemented as vanilla ones while in the new style, they are

combined with each other, for example, CNN, RNN, DNN and so on. With highly accurate results in retails and crude oil price forecasting, further research and applications could be beneficial with the new structure.

To cope with the issue of forecasting energy consumption, this thesis inherits the idea from the previously mentioned work. We attempt to predict energy demand in a more general view, which is on public datasets and various types of DL models. The used datasets consist of both kinds of buildings, commercial buildings, and households. Learning from others' results, it could be deduced that the RNN-based models are useful for our problem. We experiment with RNN and its variations while applying the strategy in clustering data into a smaller group sharing the same weather condition. Besides the weather, energy consumption might be affected by public holidays, we hence account them into forecasting. In order to show the performance of the RNN-based models, we compare the results with the benchmark ARIMA. Then, the RNN-based model, which best performs is deployed for implementation in the Forecasting service in an EMS. As mentioned before, while we experiment with forecasting on both types of buildings, our Forecasting service could predict only commercial buildings

4 Design of solution

The novel architecture of an EMS and its current implementation are in need of a forecasting service [GFA20]. Chapter four represents in-depth the component needed to be implemented and its location in the overall architecture of the EMS. The targets, difficulties, and implementation strategy are sketched in three sections: the architecture of the EMS - the SOA-EMS, Forecasting service and Forecasting models. The demand side of a microgrid contains two types of customers: household and commercial buildings with CEDs and Uncontrollable Devices (UDs).

4.1 Architecture of the service-oriented energy management system

The theoretical architecture of an EMS in a microgrid contains three main services: generation forecast, load forecast, and system coordination [GFA20]. The generation forecast is responsible to estimate the energy production within a microgrid; it is produced by distributed energy suppliers, ie. wind turbines and solar panels. The numbers are calculated based on the weather conditions, light intensity, wind speed, etc, provided by a third party Application Programming Interface (API). In the Building service, there are two types of consumers: households and commercial buildings. The Coordination service operates a microgrid by interacting with other services and decides on the action of the grid in terms of buying or selling electricity and turning on or off devices.

Its implementation is recommended with seven services to ensure the modularity in SOA software, as it is proved to be simply and independently developed, maintained, and delivered. The overall architecture of an EMS must be three-tier, including data warehouse, logic, and user interface [GFA20]. Each service has its API for communication, in which, the Coordination service posses a final API of the EMS, which triggers the actions of all other services. This implementation poses a limitation in the load forecasting service in using predetermined consumption data for both households and commercial buildings.

In the current implementation, the demand side only simulates non-residential buildings. There is no service to forecast demand, and the load profile contains very limited data. Hence, after this point, we relate the Forecasting service as the new service proposed by this thesis. It could be seen as a separate service or supplementary service for the Building service. The required data for the Forecasting service are energy consumption data, local temperature, and holidays. The new architecture of the current EMS can be seen as in Figure 4.1. The temperature could be provided by the Weather service while the history of energy consumption could be extracted from the Building service. Holiday data could be collected externally by the Forecasting service, or manually since the holiday data type is not constantly changed during time. With these required data, the Forecasting service's job is to output the energy consumption in the future, which can be presented as the consumption pattern, and peak value can be spotted. On one hand, when the Forecasting service supports the Building service, it poses two minor additional functions in the API of the Building

service, one triggers forecast values, and the other imports holiday data from another module or a third party. On the other hand, the Forecasting service could be implemented as a separate module in any EMS, which requires its own database, a logic component, and an API.

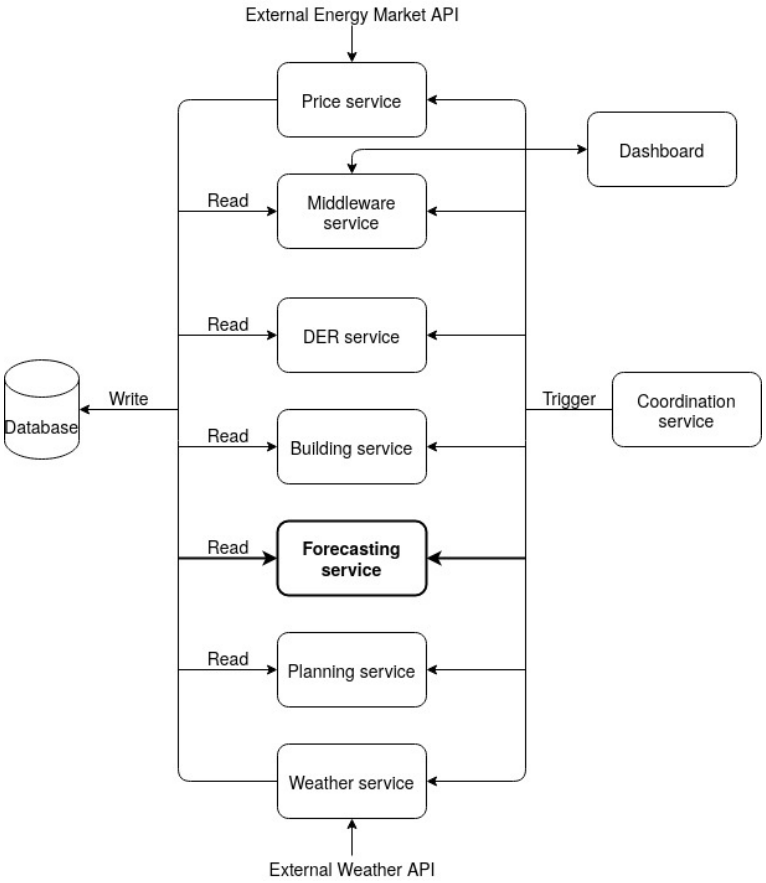

Figure 4.1: New architecture

In the thesis, for the rest of the design and implementation, we focus to develop the Forecasting service as a standalone module. It has the benefits of wide applications on any SOA-EMSs and elimination of unwanted communication between existing services in the current implementation

Building service

The demand side is one of three main components of an EMS. It contains households and commercial buildings and provides their information on energy consumption. Although in this thesis, the Forecasting service is a standalone module, there is an option to connect it to the Building service for the convenience of the building's consumption forecasting. Each building has two types of load UDs and CEDs. UDs are usually critical loads in a system, which operate mostly all the time and make up the base of energy demand They are not interrupted, ie. lighting, computers, refrigerators, etc. CEDs are non-essential electric loads, which can be deferred in a period of time without affecting the stability of the system. They model various kinds of devices, such as water heater

service, clothing, dryer service, cloth washer service, dishwasher service, printer service, etc. The hourly energy demand for a building is formulated as: $D(t) = D^{UD}(t) + D^{CED}(t)$ where $D^{UD}(t)$ is the electrical power demand of UDs at time t, and $D^{CED}(t)$ is the electrical power demand of CEDs at time t. In the proposed EMS Figure 4.1, when working together with the Building service, the Forecasting service can generate estimated values while its correctness is examined with the information imported from the Building service.

4.2 Forecasting service

Figure 4.2 is the pipeline, describing the steps to design and develop our Forecasting service. The raw data energy consumption of a region, local weather, and holidays are collected. To facilitate the analysis, undefined and missing values are either filled or disposed and to different study cases, a different approach is chosen. The Data cleaner deals with the erroneous values in each dataset, and they are fed to the Forecasting models for analysis and model selection. The suitable model features eventually the forecasting method in the corresponding API. In each study case, it is necessary to visualize and capture the trend and seasonality within the datasets themselves. These achieved properties are beneficial in creating training-testing data and choosing a suitable model.

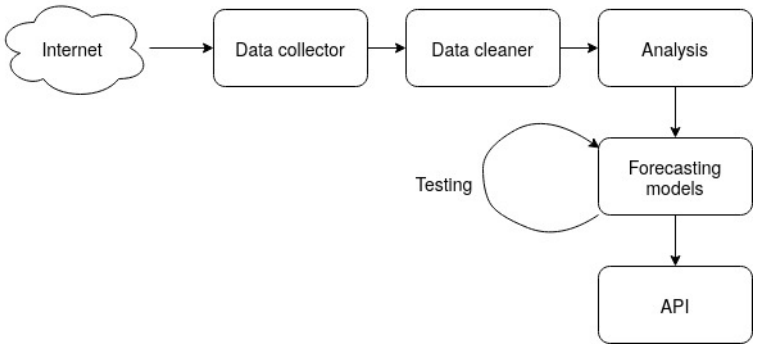

Figure 4.2: Pipeline for forecasting

We use a variety of DL models for implementation and evaluation. For each model, it is important to have the proper parameters, they are, the number of lags, number of weather features, number of layers, number of neurons on each layer, optimizer, learning rate, and so on; among them, the optimizer and learning rate play the key role in training a model. The outcome of a forecasting model is indicated by its accuracy or error after testing. To have the desired outcome of a model, we iteratively test for the suitable set of hyperparameters. The final step in deploying a service is constructing a restful API, for utilization and communication between modules. By triggering the provided method, the Forecasting service returns the inquired values. The mechanism for forecasting future values used in developing the Forecasting service is rolling forecast, which means we constantly forecast and store the future values for the next step in forecasting Algorithm 4.1, here the number of lags is presented as window_size.

The abstract architecture of the Forecasting service is designed based on Figure 4.2, in which, the whole pipeline is modeled as four main components and their corresponding database. The six components of the forecasting service:

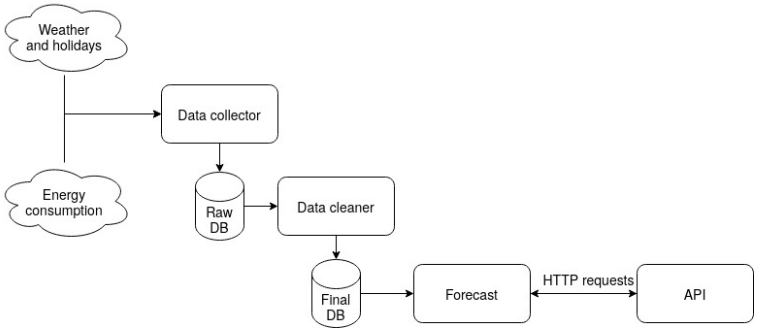

Figure 4.3: Abstract architecture of forecasting service

Algorithm 4.1 Pseudocode for rolling forecast

```
procedure FORECAST(days)
    series = data[-window_size:]
    results = LIST
    for all d ∈ days do
      _series = series[-window_size:]
      _r = model.PREDICT( _series)
      results.APPEND(_r)
      series.APPEND(_r)
  end for
end procedure
```

- Data collector: collects energy consumption of a specific city, its weather, and holidays from the internet, and store them in the Raw DB.
- Data cleaner: operates on the Raw DB, detects and corrects the missing and invalid values. It also calibrates and merges data on the same date-time; its output is exported to the Final DB. Final DB is the main storage, and is exploited for analysis, model testing, and producing predicted values when being called by API.
- Forecast: after passing through the process of building, training, and testing models, the one with the lowest error is set as the final model. After training, the output of the chosen model is stored. Using data in the Final DB and its trained results, future values are calculated and returned as requested by the API.
- API: behaves as a gate for other services to communicate and get forecast values from forecasting service by a procedure call. To keep the method syntax minimal, yet comfortable to be deployed, it is designed as a single method forecast(). The service accepts the command through HTTP POST and returns the forecast values for a defined number of days ahead with HTTP GET.

Algorithm 4.2 Pseudocode for trainng DL models

```
procedure TRAIN_MODEL(train_x, train_y, test_x, test_y)
    while stop_criteria is not None do
        for all i ∈ train_epochs do
            history = model.FIT(train_input, train_output)
        end for
        yhat_test = model.PREDICT(test_x)
        accuracy = LOSS(yhat_test, test_y)
        if PLOT(loss) is divergent then
            adjust model's hyperparameters
        end if
    end while
end procedure
```

Algorithm 4.3 Pseudocode for testing DL models

```
procedure TEST_MODEL(test_x, test_y)
    yhat_test = model.PREDICT(test_x)
    accuracy = LOSS(yhat_test, test_y)
end procedure
```

4.3 Forecasting models

The Forecast component locates at the rear of the process Figure 4.3, and can be initiated only after clean data is acquired in the Final DB. The cleaned data contains energy consumption per day per household/commercial building, weather, and holidays in the region of the household/commercial building. At this point, for each type of building, we analyze the data to discover any pattern between energy consumption and features of weather such as temperature, humidity, wind speed, etc. These relationships can be observed with the correlation matrix and described as line plots. It is desired to have other features besides the main data, which can contribute to the forecasting process by their influence on the main data; in this situation, the main data is the data of energy consumption. We choose the features that show the clear positive or negative trend against energy consumption and eliminate the ones having no or minor impact on it. By doing so, the training process speeds up by the reduced data set, and the result would be more accurate due to the absence of noise caused by unprofitable features.

We use a variety of models, from the simplest DNN, which is not supposed to work with time-series data due to its lack of causal deduction, to another form of models designed for this issue RNN and LSTM because of their property to memorize past data. Another model, which is widely used for image classification, CNN, was also brought to experiment. From the next chapter, where the implementation of these models are outlined, we refer one-layer RNN-DNN, two-layer RNN-DNN, one-layer LSTM-DNN and two-layer LSTM-DNN as one-layer RNN, two-layer RNN, one-layer LSTM and two-layer LSTM for short; in this thesis, single RNN and LSTM are not used. The process of training and testing a DL model are as the following Algorithm 4.2 and Algorithm 4.3

Algorithm 4.4 Pseudocode for training ARIMA model

```
procedure TRAIN_ARIMA
    Analyse dataset with statistical test: Dickey-Fuller Hypothesis
    while PLOT(dataset) is not stationary do
        Differencing dataset, pick hyperparameter d
    end while
    PLOT(ACF,PACF) and estimate possible hyperparameters p and q
    aic := inf
    for all (p,d,q) ∈ hyperparameter_list do
     model := ARIMA(train_set, oder = (p, d, q))        model.FIT
        aic_cur = COMPUTE_AIC(model)
        if aic_cur < aic then
            hyperparameter_opt = (p,d,q)
            aic = aic_cur
        end if
    end for
  end procedure
```

Algorithm 4.5 Pseudocode for testing ARIMA

```
procedure TEST_ARIMA
    predictions = LIST
    history := train_set
    for all sample ∈ test_set do
     model = ARIMA(history,oder = (p,d,q))
     model_fit = model.FIT(disp=0)
     output = model_fit.FORECAST
     yhat = output[0]
     predictions.APPEND(yhat)
     history.APPEND(sample)
  end for
end procedure
```

To fairly compare two types of models, RNN family and ARIMA, it is compulsory to have the correct set of (d, p, q) for the best performance of our ARIMA. There are several tests needed to be done: Auto Correlation Function (ACF), Partial Auto-Correlation Function (PACF) and Dickey-Fuller. The range of (d, p, q) is determined after the tests, we then use grid search to spot the combination that output the minimal loss as in the Algorithm 4.4 and Algorithm 4.5. Using that model on test data, the accuracy is estimated and compared with other models.

5 Realisation of solution

With the target to estimate future energy consumption, we analyze two study cases: commercial buildings and households. Chapter five explains the implementation of the forecasting service and its models. To the models' implementation, it gives details about the hyperparameters tuning, model training, and testing, and service developing.

5.1 Forecasting service for commercial buildings

The Forecasting service implementation is developed with Python; its Restful API is implemented with Flask. To optimize the training and testing process, the implementation of the Forecasting models is powered by Tensorflow and Keras libraries instead of self-built DNN, RNN, LSTM and CNN. Having a restful API, The Forecasting service meets the constraint of SOA in language independence. Due to modularization, it benefits the forecasting service on implementation and simple maintenance. In Figure 4.3, there are four blocks: Data collector, Data cleaner, Forecast and App, and their functions are listed in Figure 5.1. Because the feature names are different and complicated in energy consumption and weather data, block Data cleaner is divided into two parts for the ease of supervision and maintenance.

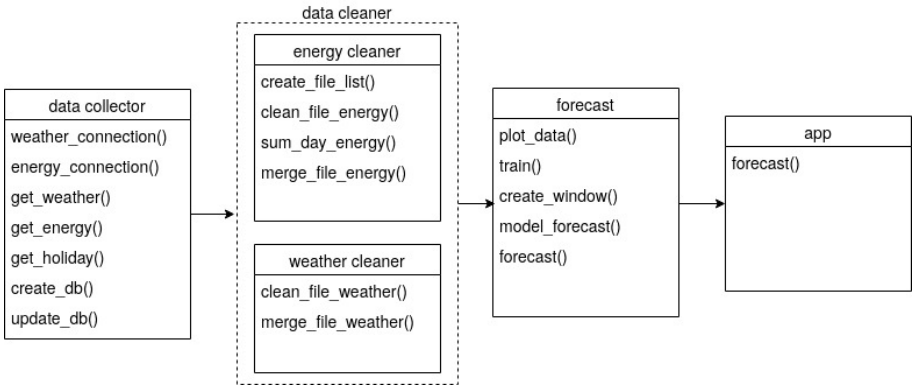

Figure 5.1: Function diagram

Table 5.1: Functions of data collector

Name	Role
'get_weather()'	gets weather data from a given URL in config file
'get_energy()'	gets energy consumption data from a given URL in config file
'get_holiday'	gets holidays from a given URL in config file
'create_db'	triggers get_weather(), get_energy() and get_holiday() and stores their results in the raw DB

Table 5.2: Functions of data cleaner

Name	Role
'clean_file_energy()'	fills empty values, calculates the sum of energy consumption of that file and returns it as hourly value
'sum_day_energy()'	calculates the sum of energy consumption per day
'merge_file_energy()'	merges all consumption files, triggers clean_file_energy() and sum_day_energy(), calculates the average consumption per building and exports result into final DB
'create_file_list()'	creates a list of files in a location for easy access
'clean_file_weather()'	eliminates redundant features, fills empty values and adjusts date
'merge_file_weather()'	merges all weather files into one, and calculates the average per year

Table 5.3: Functions of forecast

Name	Role
'plot_data()'	represents the loss and hyperparameters as line plots.
'create_window()'	prepares input data for forecasting.
'train()'	runs training process.
'model_forecast()'	supplement function for forecast(), it forecasts a single point.
'forecast()'	runs as a loop to forecast a series of points.

Table 5.4: Functions of API

Name	Role
'forecast()'	generates forecast values for a range of time

5.2 Data collection and analysis

The first prerequisite in the workflow Figure 4.2 is collecting required data, and it should be at least one-year long. With that length, it can show partially the seasonality and trend, so that applied models can form forecasting equations. The data comes from different sources for two study cases, due to the hardship in locating the proper datasets. The constraint of the collected dataset is: it must be a public dataset. And to the MIT license, there is no dataset that we could found. We then use the datasets having CC or CC0 license for our experiments.

5.2.1 Study case: commercial buildings

The first study case is about commercial buildings, where it is required to have a dataset of at least one-year long in one city. The amount of published commercial building data is scarce in comparison to household datasets. The only possible option is to use data from openei.org [Ope] , it has a CC0 license, which meets the requirement for this thesis. However, it is an old and small dataset, it was recorded in 2004 in the US, and being 2004, it poses another difficulty in collecting the weather data.

There are ten zip files in the energy consumption dataset, having 160 buildings in the US. For the purpose of forecasting the energy consumption of commercial buildings in one city, this section explored the data by 16 commercial buildings in San Francisco, California in 2004; San Francisco is an arbitrary choice at the time of experiments. They are restaurants, hospital, hotels, offices, an apartment, a patient building, schools, a retail building, a supermarket, a mall, and a warehouse.

```
['RefBldgFullServiceRestaurantNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgPrimarySchoolNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgHospitalNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgWarehouseNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgOutPatientNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgStand-aloneRetailNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgLargeOfficeNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgSmallOfficeNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgStripMallNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgMidriseApartmentNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgSuperMarketNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgQuickServiceRestaurantNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgSmallHotelNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgLargeHotelNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgMediumOfficeNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv',
 'RefBldgSecondarySchoolNew2004_7.1_5.0_3C_USA_CA_SAN_FRANCISCO.csv']
```

Figure 5.2: List of buildings [Ope]

In each building, there are two sources of energy, electricity, and gas. The first seven columns are accumulated under function CLEAN_FILE_ENERGY, calculates the total consumption during one hour for one building. Because the gas data is inconsistent with 27 missing values, neither dropping them nor filling them is possible. Therefore, we only consider the energy consumption from electricity in this study case Figure 5.8.

After summing all electricity demand in one building, the the data frame has two columns: Time and Total_Electricity_[KW] Figure 5.3. All files in the Raw DB are processed as above, then their values in hour granularity are merged into one file with MERGE_FILE_ENERGY. This function calculates the sum in one day and average value among 16 buildings, then converts the them from KW to MW. Since 2004 is a leap year, this dataset, unfortunately, lacks date 29.02.2004 Figure 5.4.

	Time	Total_Electricity_KW
0	01/01 01:00:00	38.6929
1	01/01 02:00:00	23.78
2	01/01 03:00:00	23.8029
3	01/01 04:00:00	23.8105
4	01/01 05:00:00	23.9487

Figure 5.3: Sample electricity consumption file [Ope]

There are two common solutions for missing data, filling with window data, or drop undefined data. Based on the observation of data distribution, we use an average window with the size of two, in order to keep the weekly pattern and the trend.

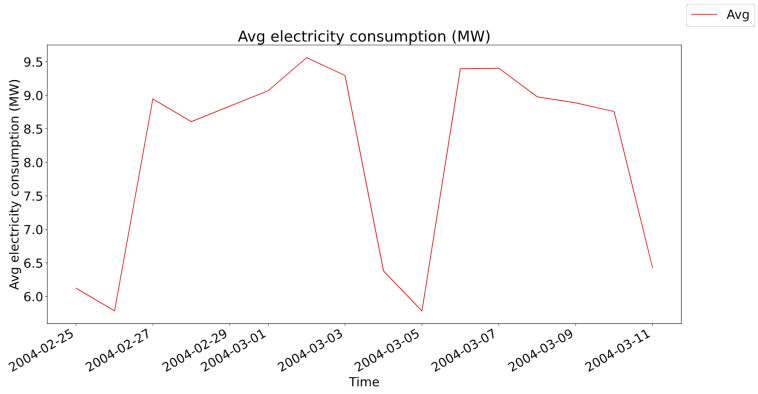

Figure 5.4: Zoom in electricity consumption in February and March

After filling 29.02.2004 with average value from 27.02.2004 to 02.03.2004, the new plot is Figure 5.5:

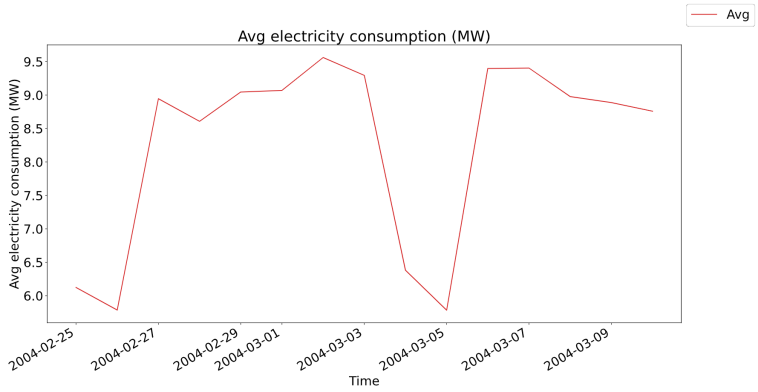

Figure 5.5: Zoom in electricity consumption in February and March after filling

The function MERGE_FILE_ENERGY then writes the return data frame as one file final energy consumption to the Final DB. It has three columns, and its first five lines and plot are below Figure 5.6 Figure 5.7:

	Time	Total_Electricity_[MW]	Avg_Electricity_[MW]
0	2004-01-01	91.5441	5.7215
1	2004-01-02	139.309	8.70682
2	2004-01-03	142.005	8.87533
3	2004-01-04	136.463	8.52896
4	2004-01-05	136.404	8.52528

Figure 5.6: Electricity consumption per building sample head

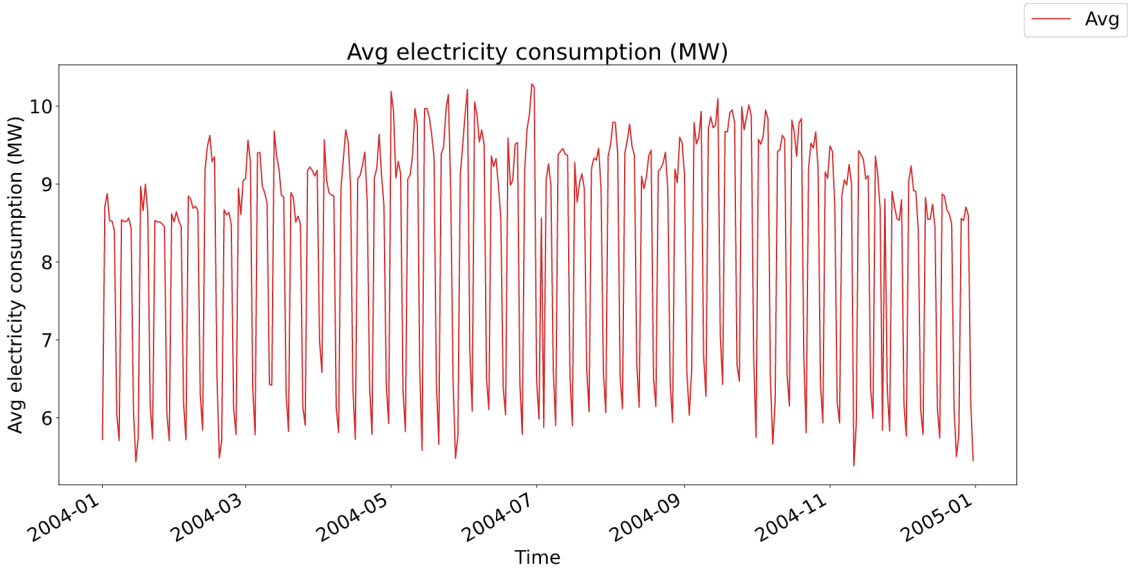

Figure 5.7: Electricity consumption per building in 2004

To the weather data, the providers since late 2020 have not been publicly open to access. They either block public access to historical data or change their policies. We tried to access weatherbit.io, weather underground, weather.com, openweathermap.org, and darksky.net. Specifically, darksky has not provided a public API since August 1st, 2020 Dark Sky [Dar]. A similar situation happens to www.wunderground.com, where it is forbidden to crawl data from. Instead of the initial plan to direct crawl data, we extract it manually and save them as csv files in the Raw DB. And as 2004 is a timestamp that is hidden by the providers, our solution is to use the data in the recent four years. We obtain weather from 2016 to 2019 in San Francisco, then take the average of them, and use the average value as the weather in 2004. The weather data is collected from San Francisco International airport station on wunderground.com. The raw weather history is then stored in the Raw DB, including temperature, dewpoint, humidity, wind speed, pressure, and precipitation Figure 5.9. For the date 29.02.2004, it has the value of 29.02.2016.

	Date/Time	Electricity:Facility [kW](Hourly)	Fans:Electricity [kW](Hourly)	...	Heating:Gas [kW](Hourly)	InteriorEquipment:Gas [kW](Hourly)	Water Heater:WaterSystems:Gas [kW](Hourly)
0	01/01 01:00:00	22.239192	3.674591	...	52.195467	3.33988	6.349174
1	01/01 02:00:00	14.759857	0.000000	...	0.000000	3.33988	0.020000
2	01/01 03:00:00	14.782705	0.000000	...	0.000000	3.33988	0.592050
3	01/01 04:00:00	14.790355	0.000000	...	0.000000	3.33988	0.020000
4	01/01 05:00:00	14.928563	0.000000	...	0.000000	3.33988	0.592056

Figure 5.8: Sample electricity consumption head [Ope]

	Time	Max_temp_[F]	Min_temp_[F]	...	Max_pressure_[in]	Min_pressure_[in]	Precipitation_[in]
0	2004-01-01	53.00	43.50	...	30.175	30.050	0.0000
1	2004-01-02	53.50	44.00	...	30.175	30.075	0.0125
2	2004-01-03	54.75	46.00	...	30.100	29.925	0.0650
3	2004-01-04	60.25	47.00	...	30.000	29.825	0.2750
4	2004-01-05	56.25	48.75	...	30.000	29.825	0.2025

Figure 5.9: Weather data sample

We examine the collected data in the Final DB to capture their positive and negative relationships in trend. Each weather factor is featured with energy consumption in 2004, and their correlation matrix is shown at Figure 5.17:

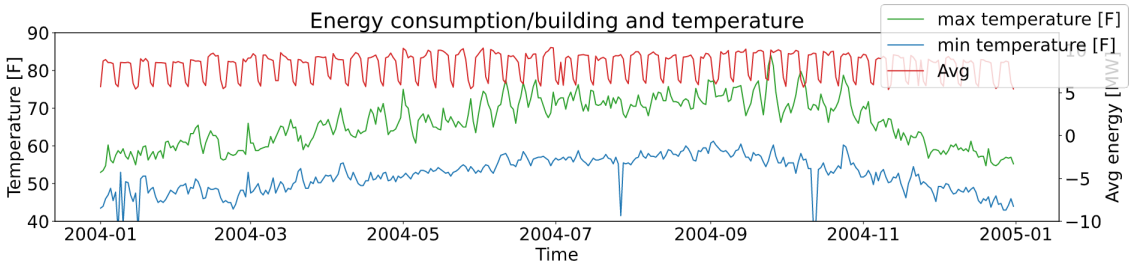

Figure 5.10: Electricity consumption per building and temperature

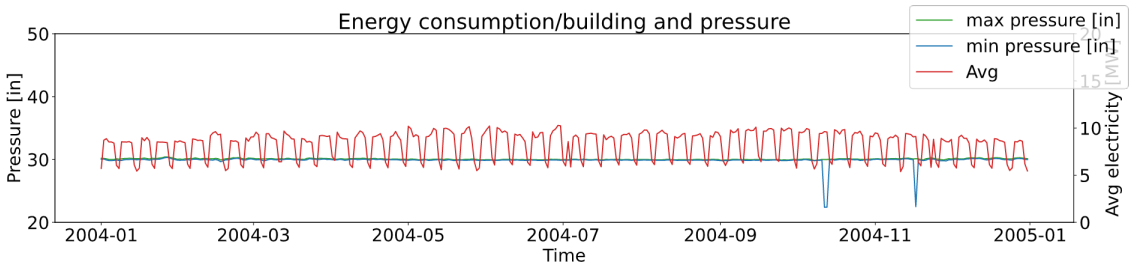

Figure 5.11: Electricity consumption per building and pressure

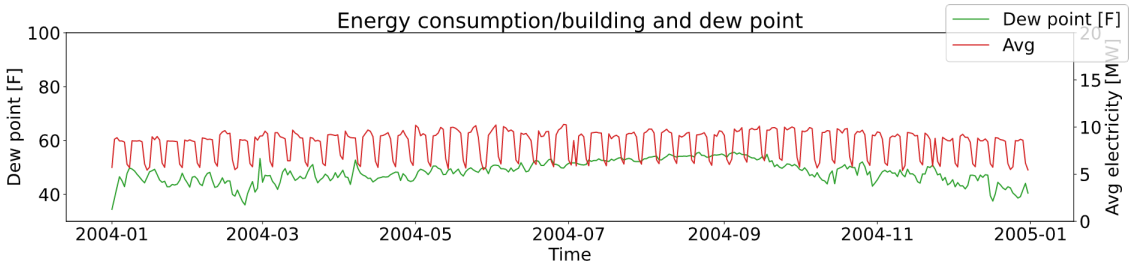

Figure 5.12: Electricity consumption per building and dew point

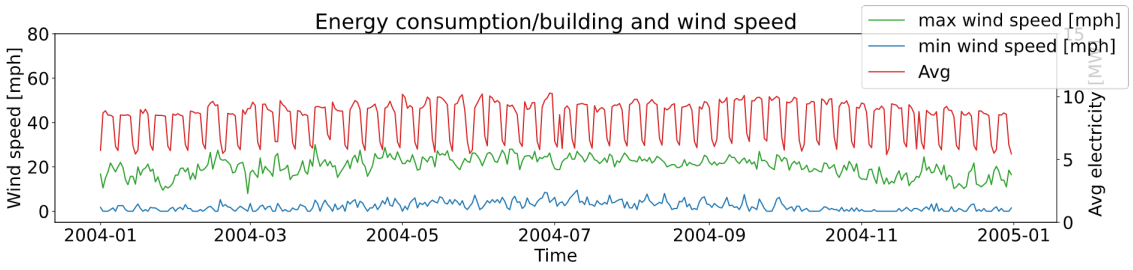

Figure 5.13: Electricity consumption per building and wind speed

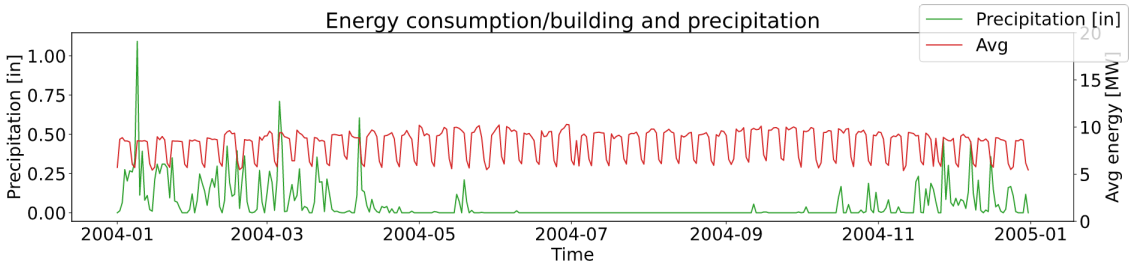

Figure 5.14: Electricity consumption per building and precipitation

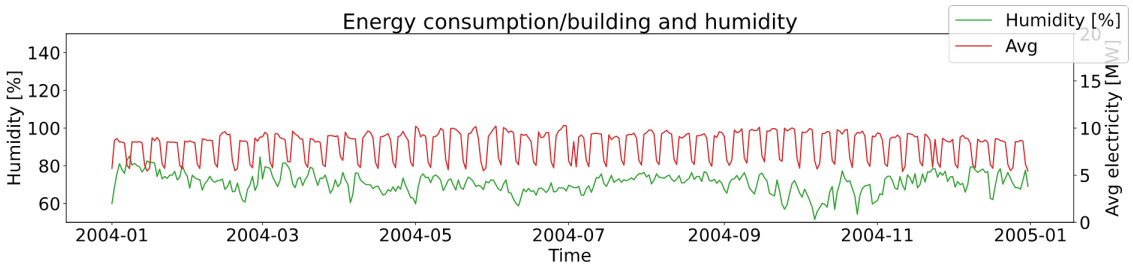

Figure 5.15: Electricity consumption per building and humidity

Essentially, we consider holidays as a related feature of energy consumption. In 2004, there were 14 holidays in San Francisco, this data is collected from google.com. In these holidays, Figure 5.16, there is merely any signals showing energy consumption rises or sinks.

	Bank holidays	Type
0	2004-01-01	New Year's Day
1	2004-01-19	Martin Luther King Jr. Day
2	2004-02-12	Lincoln's Birthday
3	2004-02-16	Washington's Birthday
4	2004-03-31	Cesar Chavez Day
5	2004-05-31	Memorial Day
6	2004-07-05	Independence Day
7	2004-09-06	Labor Day
8	2004-09-24	Native American Day
9	2004-10-11	Columbus Day
10	2004-11-11	Veterans Day
11	2004-11-25	Thanksgiving
12	2004-12-24	Christmas Day
13	2004-12-31	New Year's Day

Figure 5.16: Holidays in San Francisco in 2004

From the plots above and the correlation matrix in Figure 5.17, there exists a non-obvious trend of consumption towards the temperature, humidity, pressure, precipitation, dew point, and holidays. It is observed to have a strong seasonality in the time step of every week. In this scenario, the investigation exposes that there are two possible hypotheses, (1) is commercial buildings' operations are not directly influenced by the weather and holidays, and (2) the analyzed dataset (2004) is too small for trend observation. In the case of a longer history, approximately four years, for instance, the relationship pattern might display properties contributing to viewing and forecasting energy consumption. Consequently, we use energy consumption alone with lags from seven and up.

	temp	dew	humid	wind	pressure	precip	Avg
Max_temp_[F]	1.000000	0.757700	-0.471664	0.467264	-0.675029	-0.445370	0.178807
Dewpoint_[F]	0.757700	1.000000	0.167889	0.414828	-0.641487	-0.289548	0.156563
Humidity_[%]	-0.471664	0.167889	1.000000	-0.227020	0.214089	0.275344	-0.082364
Max_windspeed_[mph]	0.467264	0.414828	-0.227020	1.000000	-0.610217	-0.165257	0.088482
Max_pressure_[in]	-0.675029	-0.641487	0.214089	-0.610217	1.000000	0.250736	-0.081044
Precipitation_[in]	-0.445370	-0.289548	0.275344	-0.165257	0.250736	1.000000	-0.024892
Avg_energy_[MW]	0.178807	0.156563	-0.082364	0.088482	-0.081044	-0.024892	1.000000

Figure 5.17: Correlation matrix of commercial buildings

5.2.2 Study case: households

In the second study case, there are several available datasets, however, given the match in our license requirement and the accessibility of the them, we use the energy consumption from "London Households"[UK]. It was collected by UK Power Network in the project of Low Carbon London between November 2011 to February 2014. It contains 5567 London households, and it has a granularity of haft-hour, hour, and daily consumption. Along with the hourly energy consumption dataset, it is needed to have the weather and UK bank holiday dataset, this study case uses the ones provided by Jean-Michel D [Jea] in 2017.

For energy consumption, we use the daily dataset, instead of half-hour or hour ones for the unification among study cases. The raw data is unbiased, as it can be seen, in the early of the project 2011, there are approximately more than 50 households, and the number goes up sharply in mid-2012 and remain stable from the end of 2012 until the end of 2014 with more than 5000 households. We

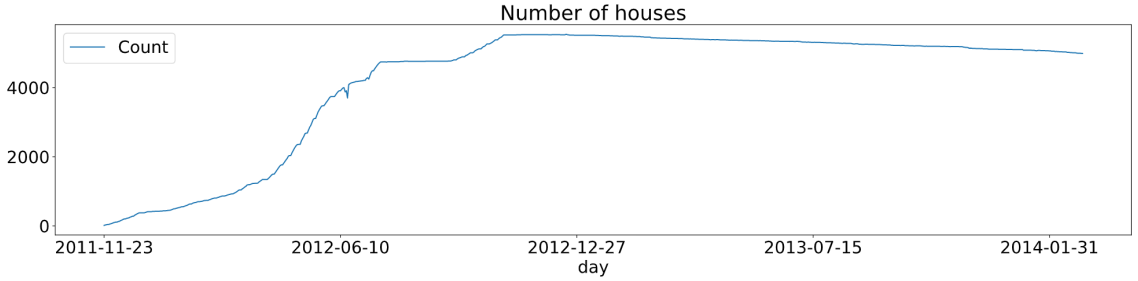

Figure 5.18: Distribution of households

collect the total electricity consumption per day of all available households and use the average value per household for the experiment. The household dataset contains three small problems (1) the unit for electricity consumption and the weather datasets are vague; (2) it does not has the date 01.01.2014 and (3) the holiday dataset does not have the holidays in late 2011. Using a different technique to tackle undefined/missing values, we dropped them in this experiment. The raw electricity consumption dataset includes 112 files. After merging them into one dataframe, we calculate the daily average consumption per household and save it as merged_energy.csv in the Final DB.

	day	energy_sum	Count	avg_energy
0	2011-11-23	90.385000	13	6.952692
1	2011-11-24	213.412000	25	8.536480
2	2011-11-25	303.993000	32	9.499781
3	2011-11-26	420.976000	41	10.267707
4	2011-11-27	444.883001	41	10.850805

Figure 5.19: Electricity consumption per household

We analyze the relationship between household consumption and the outside weather in London. By inspecting the trend and dependency, there are some weather conditions adding to the variation in electricity consumption patterns. They are shown by the below plots and their correlation matrix Figure 5.24:

- Temperature has a high negative correlation with electricity consumption while humidity and dew point has a high positive correlation
- Wind speed shows a low correlation with electricity consumption
- Max and min temperatures have the same tendency, we used only max temperature as the temperature

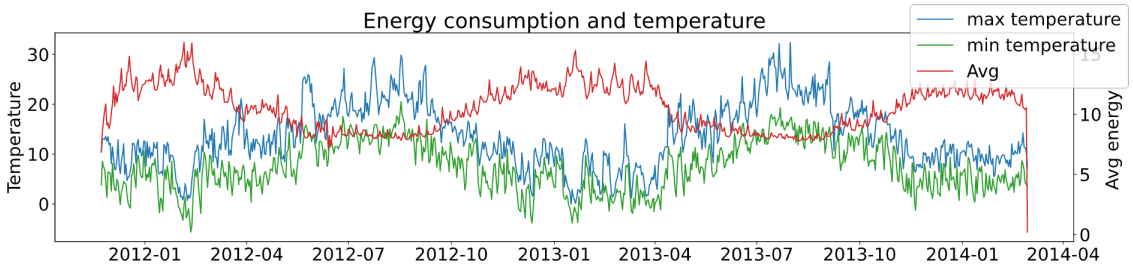

Figure 5.20: Electricity consumption per house and temperature

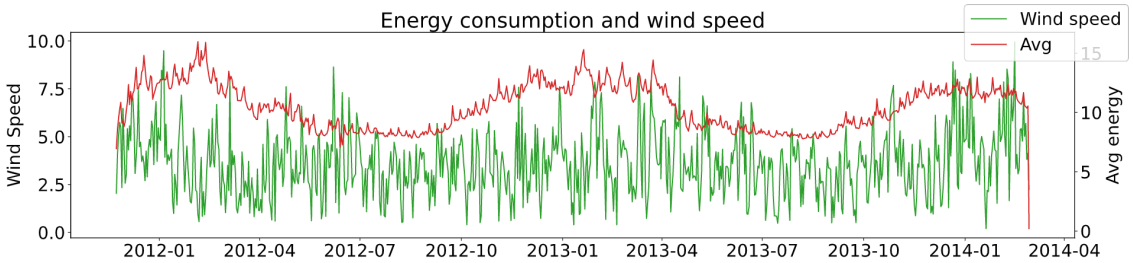

Figure 5.21: Electricity consumption per house and wind speed

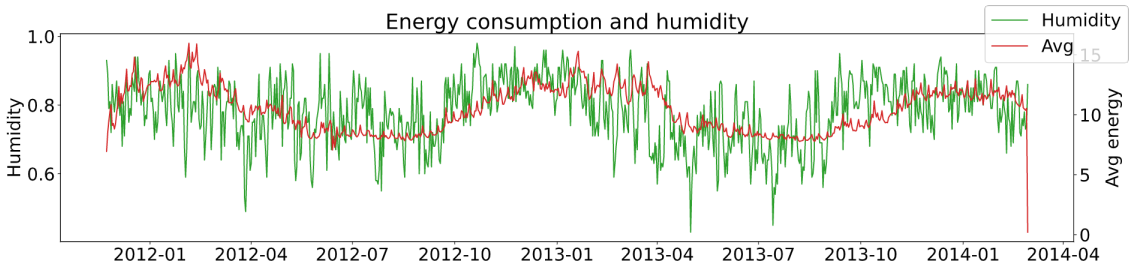

Figure 5.22: Electricity consumption per house and humidity

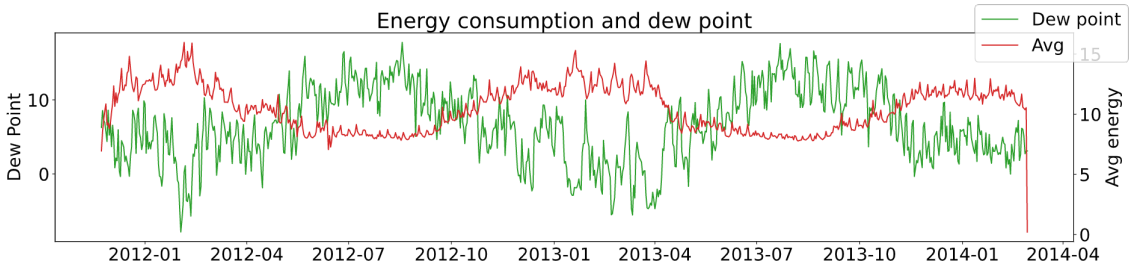

Figure 5.23: Electricity consumption per house and dew point

	avg_energy	temperatureMax	dewPoint	cloudCover	windSpeed	pressure	humidity	uvIndex	moonPhase
avg_energy	1.000000	-0.846965	-0.755901	0.241779	0.149624	-0.028851	0.361237	-0.733171	-0.031716
temperatureMax	-0.846965	1.000000	0.865038	-0.333409	-0.153602	0.118933	-0.404899	0.696497	0.003636
dewPoint	-0.755901	0.865038	1.000000	-0.025207	-0.092212	-0.028121	0.055514	0.486692	-0.008239
cloudCover	0.241779	-0.333409	-0.025207	1.000000	0.170235	-0.101079	0.480056	-0.248695	-0.062126
windSpeed	0.149624	-0.153602	-0.092212	0.170235	1.000000	-0.344354	-0.042391	-0.152634	-0.023273
pressure	-0.028851	0.118933	-0.028121	-0.101079	-0.344354	1.000000	-0.250941	0.100774	0.038462
humidity	0.361237	-0.404899	0.055514	0.480056	-0.042391	-0.250941	1.000000	-0.533919	-0.013997
uvIndex	-0.733171	0.696497	0.486692	-0.248695	-0.152634	0.100774	-0.533919	1.000000	0.012833
moonPhase	-0.031716	0.003636	-0.008239	-0.062126	-0.023273	0.038462	-0.013997	0.012833	1.000000

Figure 5.24: Correlation matrix of households

Another problem with the consumption dataset is its last value, as in the description Figure 5.25, it is the min value 0.211766, and it is abnormally small, therefore it is eliminated as an error Figure 5.27.

	energy_sum	Count	avg_energy
count	829.000000	829.000000	829.000000
mean	43535.325676	4234.539204	10.491862
std	20550.594031	1789.994799	1.902513
min	90.385000	13.000000	0.211766
25%	34665.436003	4084.000000	8.676955
50%	46641.160997	5138.000000	10.516983
75%	59755.616996	5369.000000	12.000690
max	84156.135002	5541.000000	15.964434

Figure 5.25: Household description

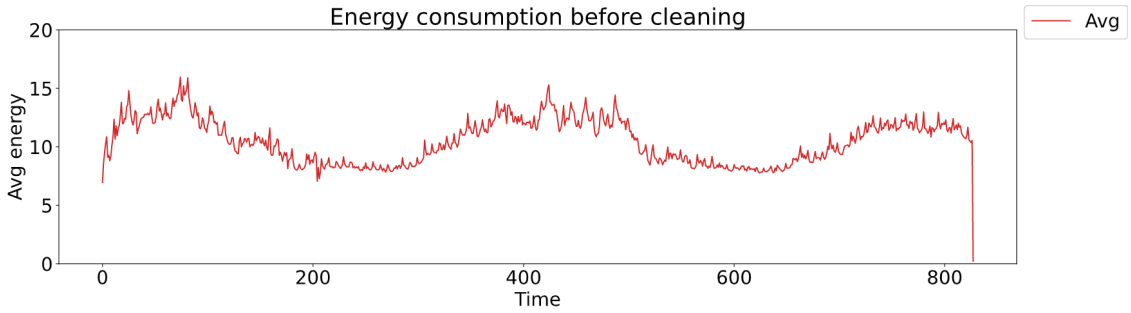

Figure 5.26: Avg energy consumption before cleaning

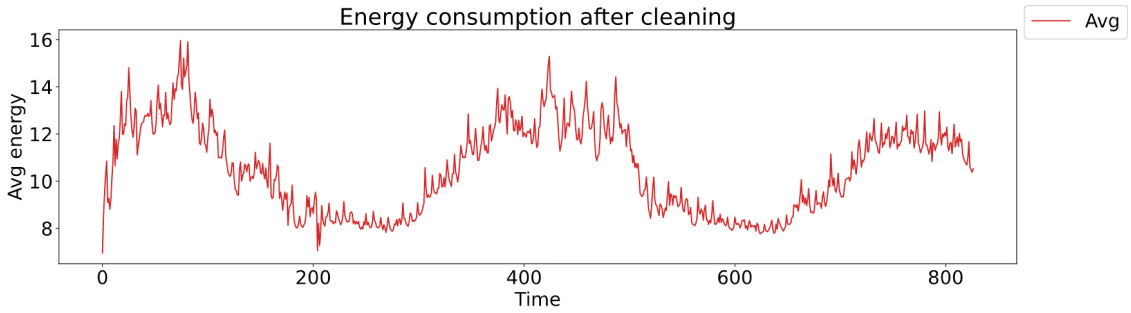

Figure 5.27: Avg energy consumption after cleaning

A similar set of models are applied to the household study case; on the contrary to the commercial buildings one, there are other strategies we applied here due to its complexity: clustering the weather and labeling holidays. Clustering the weather into groups depends on the maximum temperature, humidity, and wind speed. To form clusters, we apply K-mean clustering to identify the number of groups suitable for the current weather, Figure 5.28.

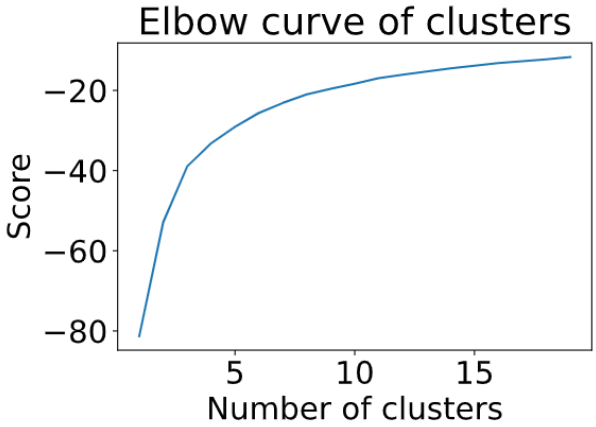

Figure 5.28: Number of clusters

In clustering, the number of clusters is the one at the turning point; there are two possible values on the Figure 5.28, which are three and four. This experiment separates the study case into three clusters. They are then visualized with Figures 5.29 and 5.30.

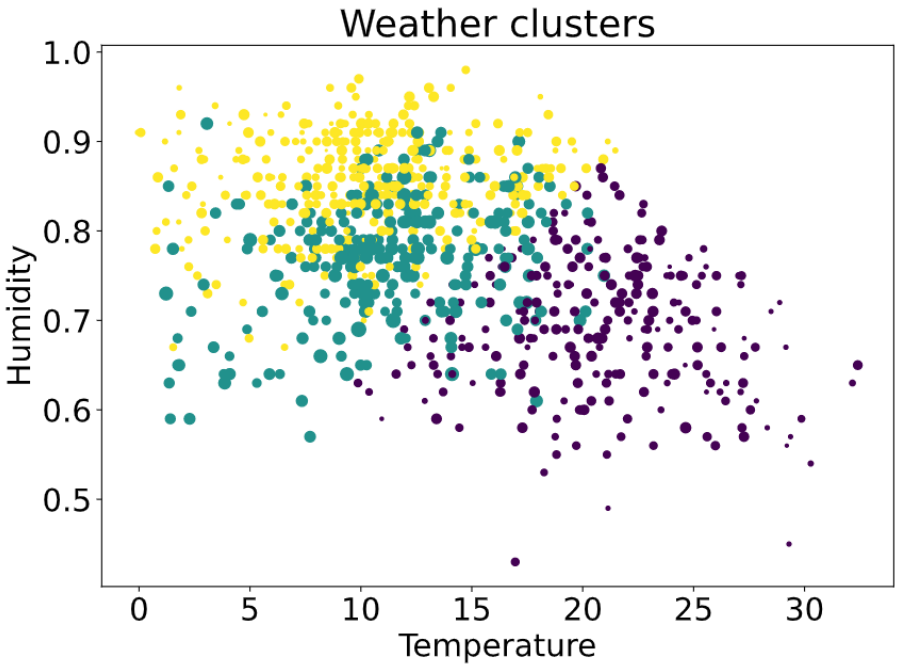

Figure 5.29: Household cluster plot

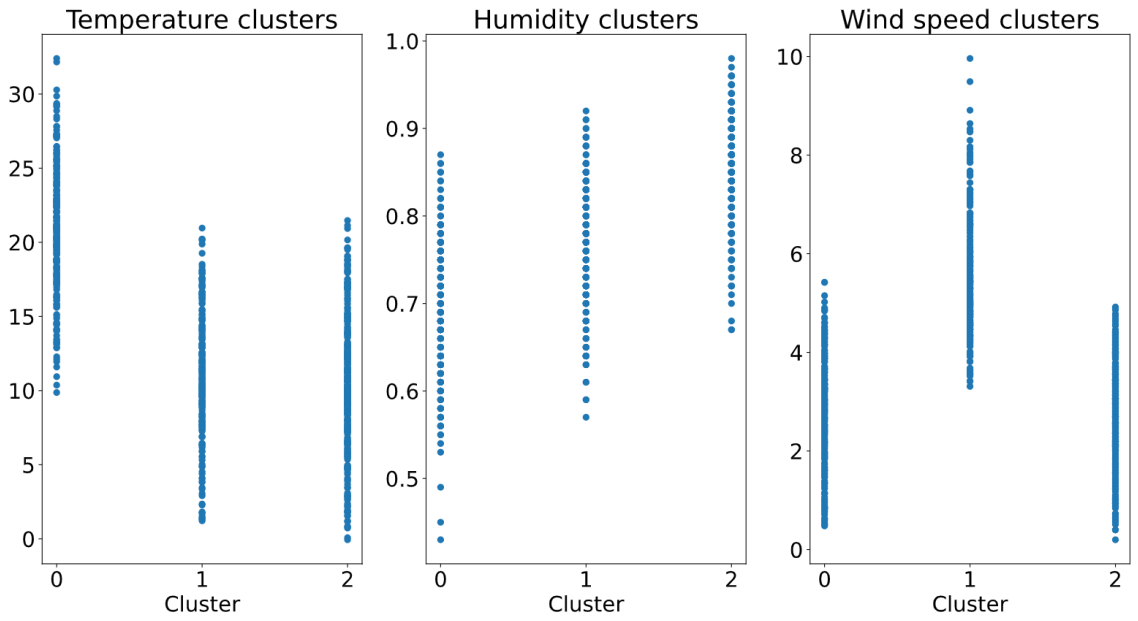

Figure 5.30: Household cluster analysis

The holiday dataset contains UK Bank holidays Figure 5.31 in the range 2012 to 2014, we label the holiday as 1 and the normal day is 0. It is then merged with the average electricity consumption and weather cluster into the final dataframe for the models.

	Bank holidays	Type
0	2012-12-26	Boxing Day
1	2012-12-25	Christmas Day
2	2012-08-27	Summer bank holiday
3	2012-05-06	Queen?s Diamond Jubilee (extra bank holiday)
4	2012-04-06	Spring bank holiday (substitute day)
5	2012-07-05	Early May bank holiday
6	2012-09-04	Easter Monday
7	2012-06-04	Good Friday
8	2012-02-01	New Year?s Day (substitute day)

Figure 5.31: Holidays in London 2011-2014

5.3 Forecasting models

There are six DL models used to perform forecasting task of electricity consumption, they are DNN, one-layer RNN, two-layer RNN, one-layer LSTM, two-layer LSTM and the merged model of CNN-LSTM-DNN. For the DL models, the cleaned dataset is first imported as a csv file from the

database of cleaner. It is then split in the ratio 70:30 for training and testing. Next, the model is built, tuned, trained, and tested as in Algorithm 4.2 and Algorithm 4.3. The statistic model ARIMA acquires a similar process, in which, the dataset is imported, correlatively analyzed, the model is tuned for the best hyperparameter combination (p, d, q) and with the same ratio of train-test split as in Algorithm 4.4 and Algorithm 4.5. The implementation and execution of both model types locate in the folder Model and is presented as notebooks for analysis; for each study case, there are a set of notebooks:

- analysis.ipynb: analysis of the dataset and examine the relationships between data features.
- dnn.ipynb: notebook for only DNN model.
- rnn.ipynb: notebook for the family of RNN, containing one-layer RNN, two-layer RNN, one-layer LSTM, two-layer LSTM, CNN-LSTM-DNN.
- arima.ipynb: notebook for analysis and execution of ARIMA model.

The model building process applies for DL models, starting with the selection of layer type, the number of layers, and the number of neurons per layer. Another hyperparameter is the learning rate, which needs to be correctly tuned in the learning rate schedule with callback API in Keras. Tuning depicts a series of actions in order to find the combination of parameters that will output the least loss; in the experiment, we use mean absolute error loss for training and mean squared error for testing. Models are trained with two optimizers, gradient descent with momentum (SGD) and its variation, named ADAM. Sharing the same sets of models, the hyperparameters in two study cases are slightly different, due to the tuning process. By this, the models can best fit each behavior of data.

The layer names are defined in Tensorflow as:

- The DNN model consists of three fully connected layers, which means that all neurons in one layer are linked to those in the next layer. The model is then presented as three Dense layers.
- RNN-DNN: RNN is placed on top of the DNN model, one layer of RNN is defined as one SimpleRNN layer. The RNN-DNN stands for one SimpleRNN layer on three Dense layers.
- LSTM-DNN: similar to the RNN-DNN, the model swap SimpleRNN by LSTM.
- CNN-LSTM-DNN: The CNN layer in the experiments is defined as Conv1D, the model describes one layer Conv1D on one layer LSTM and on three Dense layers.

The last model CNN-LSTM-DNN is the combination of three types of models. Functionally, it has all the benefits, feature extraction of a CNN, and sequence dependency observation of LSTM-DNN. It is hence more robust to complex data [LPP20] [KC18] [BRF+20]. A CNN is very famous to handle the job of images, ie: object classification, image segmentation,...due to its powerful filters [GBC16]. These filters can compute dilation between cells, allowing LSTM to better understand the relationship of different data points. Therefore, in this experiment, we attempt to use it for the question of forecasting energy consumption. Instead of 2D-CNN in image handling tasks, time series uses 1D-CNN. The reason behind this is the image has its pixels ranging in two dimensions, while time series has only one dimension time.

Before training the DL models, one more step relating the processing data must be done, which is normalization. This is a regular practise in any ML or DL model, here, it is even more important to normalize data into a common range (0,1). In the previous section, the range of energy consumption

is (0,11)[MW] in commercial buildings and (0,16) in households. By bringing the consumption to the range (0,1), we can avoid the situation of gradient vanishing and exploding to RNN, which hinder the learning process of our models. The mathematical reason is in the architecture of RNN and LSTM, where it has the activation; tanh activation can only output predictions in range (-1,1), resulting in the huge calculation of $loss = actual - prediction$, leading to either exploding or vanishing gradient. We use a tool in the sklearn library named minmaxscaler to further process data before feeding it to forecasting models. Since the data is consistent in scale and distribution, the forecasting models run faster and perform extremely better. The output of the models, hence, must be converted back to its normal scale by calling MINMAXSCALER.INVERSE_TRANSFORM.

Study case: commercial buildings

Except for DNN and ARIMA, the RNN family requires windowed dataset as input. Function CREATE_WINDOW generates windowed dataset, in this case, data at time points $[t-window\_size, t-1]$ are used to forecast data at time points $[t - window\_size - 1, t]$, because there is a relationship between data points in the past and the ones in the future. The construction of models is listed in Table 5.5. Here, we use ReLU as the activation function for models, and Appendix A.1 covers in details about the activation functions.

Table 5.5: Models in study case commercial buildings

Model	Definition	Optimizer	Others
DNN	Dense(30, ReLU activation) Dense(15, ReLU activation) Dense(1)	SGD learning rate =$1e-3$ momentum = 0.9	batch_size = 64 window_size = 7 epochs = 400
One-layer RNN	SimpleRNN(32) Dense(32, ReLU activation) Dense(16, ReLU activation) Dense(1)	ADAM	batch_size = 100 window_size = 7 epochs = 400
Two-layer RNN	SimpleRNN(32) One-layer RNN	ADAM	batch_size = 64 window_size = 7 epochs = 400
One-layer LSTM	LSTM(64) Dense(64, ReLU activation) Dense(32, ReLU activation) Dense(1)	ADAM	batch_size = 120 window_size = 7 epochs = 400
Two-layer LSTM	LSTM(64) One-layer LSTM	ADAM	batch_size = 100 window_size = 7 epochs = 400
CNN-LSTM-DNN	Conv1D() LSTM(32) Dense(32, ReLU activation) Dense(16, ReLU activation) Dense(1)	ADAM	batch_size = 100 window_size = 7 epochs = 400

The Conv1D() layer in the last model comprises of 32 filters, kernel size 5, strides 1, causal padding, and ReLU activations. To the ARIMA model, after analysis as in Algorithm 4.4, we perform grid search on and obtain the optimal combination of $(p, d, q) = (10, 1, 0)$.

Study case: households

Similar to the previous study case, the RNN family requires windowed dataset as input. Function CREATE_WINDOW generates windowed dataset, in this case, data at time points $[t-window\_size, t-1]$ are used to forecast data at time points $[t]$, because we observe and find out that this type of window

is suitable for this study case. Although the processed inputs and the hyperparameters of models in two cases are slightly different, the architecture of models are identical. The construction of models is listed in Table 5.6.

Table 5.6: Models in study case households

Model	Definition	Optimizer	Others
DNN	Dense(30, ReLU activation) Dense(15, ReLU activation) Dense(1)	SGD learning rate =$3e-5$ momentum = 0.9	batch_size = 10 window_size = 30 epochs = 500
One-layer RNN	SimpleRNN(30) Dense(30, ReLU activation) Dense(15, ReLU activation) Dense(1)	ADAM	batch_size = 100 window_size = 7 epochs = 400
Two-layer RNN	SimpleRNN(32) One-layer RNN	ADAM	batch_size = 100 window_size = 7 epochs = 400
One-layer LSTM	LSTM(30) Dense(30, ReLU activation) Dense(15, ReLU activation) Dense(1)	ADAM	batch_size = 64 window_size = 7 epochs = 400
Two-layer LSTM	LSTM(30) One-layer LSTM	ADAM	batch_size = 100 window_size = 7 epochs = 400
CNN-LSTM-DNN	Conv1D() One-layer LSTM	ADAM	batch_size = 100 window_size = 7 epochs = 400

The Conv1D() layer in the last model comprises of 30 filters, kernel size 5, strides 1, causal padding, and relu activations. To the ARIMA model, after analysis as in Algorithm 4.4, we perform grid search and obtain the optimal combination of $(p, d, q) = (7, 1, 1)$.

5.4 Challenges

To demonstrate the experiments, we encounter two challenges (1) vanishing gradient and (2) dataset collection. -based models are one of the most challenging types of models to train in DL. In the training process, its task is to find a set of weights in the network through an iterative process

until it can prove to have high accuracy on prediction, in which the weights are adjusted in small steps by gradient descent technique, which results in the change in performance of the model after each iteration. In training a DL model, it is crucial to find the suitable optimizer and normalize the input data, otherwise, the model can unlikely to learn and the forecast values are hence highly incorrect. We use minmaxscaler from the sklearn library to normalize data and two optimizers, gradient descent with momentum and ADAM. To gradient descent with momentum, choosing the right learning rate is decisive in generating a correct result, while ADAM is more comfortable to use and faster.

The second problem is with the datasets, as partially mentioned in Section 5.2, the required data sources prohibit access via their API and HTTP crawling. This problem directly disables the functions of the module data collector, which we implemented before the prohibition. The only possible workaround is to collect necessary data from sources manually and store them in the raw DB as the first step. Following the idea, we collect energy consumption, weather, and holidays other modules can be afterwards initialized.

6 Evaluation

The testing dataset (test_x, test_y) is used to validates the performance of models, in which, accuracy was derived by TENSORLOW.KERAS.METRICS.MEAN_ABSOLUTE_ERROR(test_y, predictions), and predictions = KERAS.MODELS.SEQUENTIAL.PREDICT(test_x). The function to calculate MAPE is programmed as the given mathematics formula in Chapter 2. Chapter six reports the results of all the models used to predict electric consumption of the commercial buildings, and compare their results with the baseline ARIMA. They are measured with MAE, MAPE and execution time. They are DNN, one-layer RNN, two-layer RNN , one-layer LSTM, two-layer LSTM, CNN-LSTM-DNN and finally ARIMA. In practice, more than two layers of both RNN and LSTM can be stacked [SCKV15]. However, in our experiments, from three layers of them breeds poor results. Focusing on this behavior is out of the scope of this thesis, here, their results are not shown.

6.1 Study case: commercial buildings

All models are tested on 30% of the dataset, which is 110 days. Figure 6.1 plots the forecast values versus the actual one between one-layer LSTM and ARIMA. Because one-layer LSTM is the best performer in the RNN family, it is used for comparison with the benchmark. The plots of other models are in Appendix A.4.1.

ARIMA has been long studied and known for its decent performance on time series forecasting problem, and the experiments show no controversial result. It can be seen from Table 6.1, ARIMA functions excellent, with MAE only 0.4387, meaning 5.82% different to the correct value or 94.18% accuracy. As the benchmark, we aim to develop DL models to outperform it. Starting from a non-memorizable model DNN, we witness the improvement in results, slightly from MAE 0.4387 to 0.4073. With more complicated models, it is expected to have an advantage over traditional ones, One-layer RNN performs better than DNN, with MAE of 0.4029 and accuracy of 95.08%, on the other hand, the result declines in two-layer RNN. A more complex model is not necessarily equal to better result besides the longer execution time; and it is a good sign of overfitting, where a model remembers too much of the training data leading to worsening the generality of the weight matrices and results in bad forecast; this problem is out of the scope of the thesis, a trustworthy explanation is supported by [Jas]. As presented in Chapter 2, LSTM have some advantages over RNN, which is impressively proven with the dramatic decrease in MAE, from 0.4387 of ARIMA to 0.3386. Two-layer LSTM and the most sophisticated model CNN-LSTM-DNN also share the same common problem to two-layer RNN, they all show marginal enhancement, approximately 1% accuracy.

To the training and testing time, ARIMA shows the superior in training time, however, in testing, it is extremely slow 1641.85s. The reason might lay in its mechanism in forecasting Algorithm 4.5, while in each step, it muss retrain then forecast. And because of this, the longer the forecasting sequence, the slower it takes to generate forecast values. DL models are preferable in testing, as

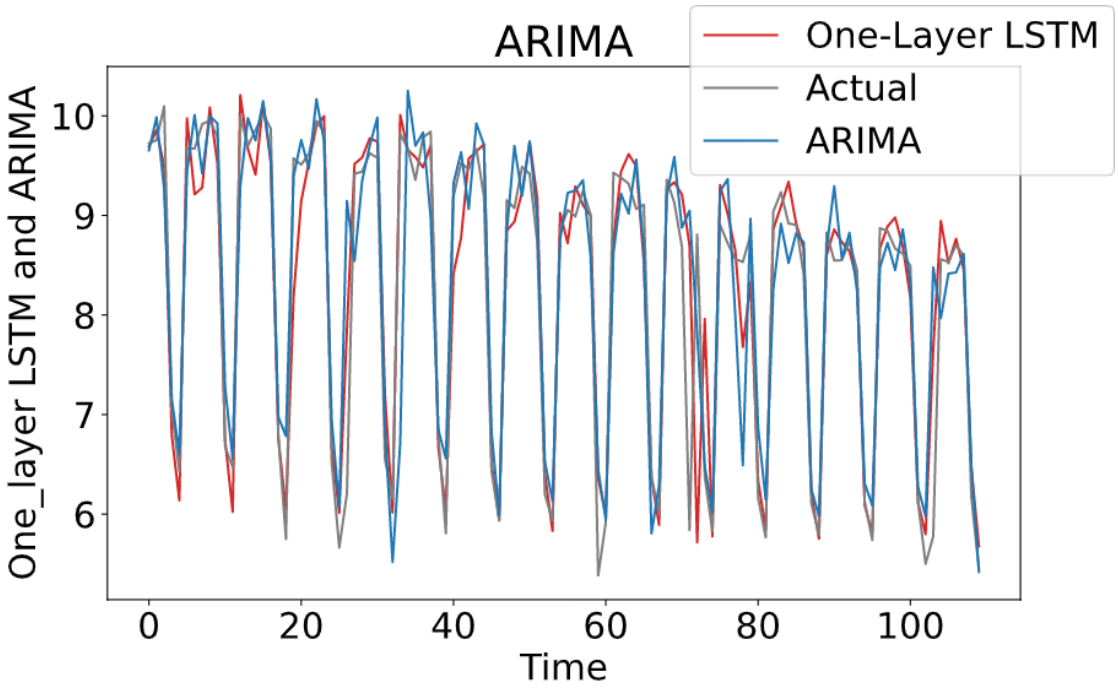

Figure 6.1: One-layer LSTM and ARIMA

their mechanism is matrix multiplication. They only have to train once, then the testing inputs are multiplied with their training output, which is the weight matrices; performing vectorization under matrix multiplication is reported to be notably faster than iteration [Jal].

Table 6.1: Metrics of study case commercial buildings

Model	**MAE**	**MAPE (%)**	**training time (s)**	**testing time (s)**
DNN	0.4073	5.56	27.70	13.1
One-layer RNN	0.4029	4.92	33.33	0.31
Two-layer RNN	0.4094	4.95	37.49	0.4
One-layer LSTM	**0.3386**	**4.30**	**43.75**	**0.16**
Two-layer LSTM	0.3972	4.96	59.36	0.9
CNN-LSTM-DNN	0.3982	4.90	40.81	0.54
ARIMA	**0.4387**	**5.82**	**13.05**	**1641.85**

With the reasoning above and the number on the Table 6.1, we use model one-layer LSTM for the Forecasting service deployment. The model outputs the weight matrices, which are stored after training and are imported when FORECAST(days) is triggered. They are then used to forecast values outside of both training and testing sets, using rolling forecast tactics Algorithm 4.3, the model generates future values as being called.

6.2 Study case: households

The second study case with the dataset from 5567 households in London from 2011 to 2014. On one hand, we equivalently use the same set of models as in the previous study case, starting with six DL models and then ARIMA for comparison. On the other hand, due to the complexity in the dataset itself, different method to assemble energy consumption, weather and holidays is applied. There exist a clear correlation between temperature, humidity, wind speed, and energy consumption. Hence, weather features are clustered in three while holidays are labeled for the convenience of picturing and separating samples, and forecasting demand. All models are tested on 30% of the dataset, in this study case, they are 249 days. Figure 6.2 plots the forecast values versus the actual one between CNN-LSTM-DNN and ARIMA, because CNN-LSTM-DNN is the best performer in the RNN family. The plots of other models are in Appendix A.4.2.

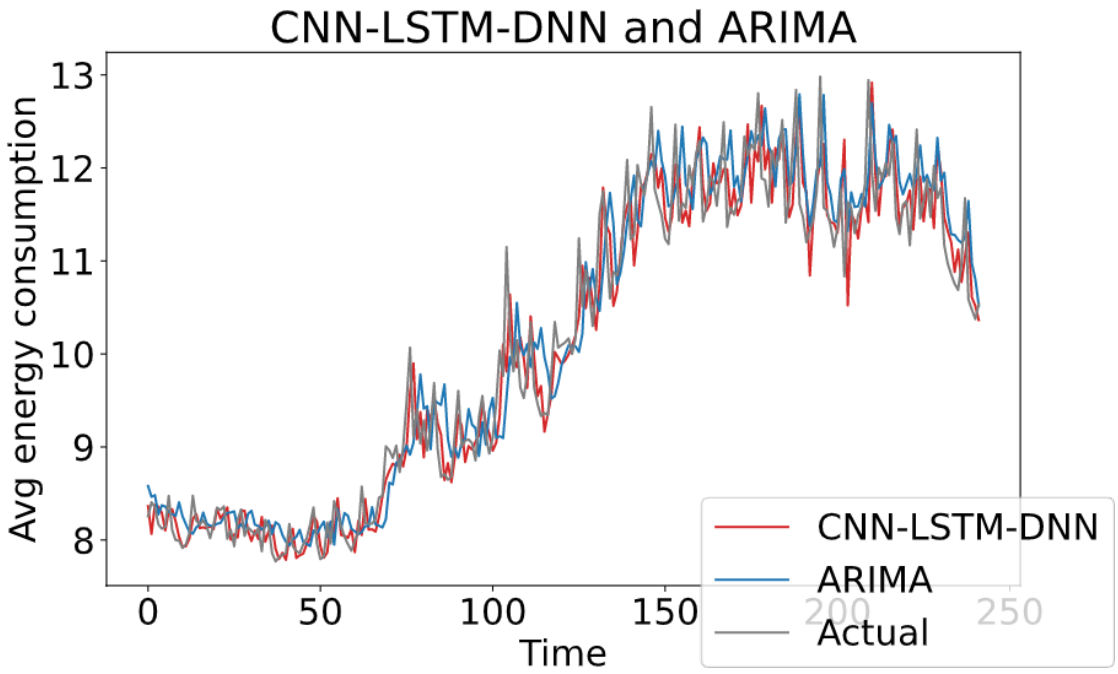

Figure 6.2: CNN-LSTM-DNN and ARIMA

Although being the benchmark, and was created in 1900s, ARIMA still shows the potential in solving time series problem. On Figure 6.2, the forecast values show a minor difference in comparison with the actual one. Its impressive results are MAE 0.272, and 3.22s in training time Table 6.2. DNN outperformed ARIMA by 0.0071 in MAE, it lifts the accuracy up, however, by a trivial amount. Switching to memorizable DL models, remarkable improvement is seen Table 6.2, specifically, the accuracy is as twice as better than ARIMA, from 0.2720 to roughly 0.14 achieved by two-layer RNN, one-layer LSTM, two-layer LSTM and CNN-LSTM-DNN. It is noticed that, stacking two layers of RNN or LSTM in this study case decays an immense amount of MAE, from 0.1778 to 0.1447 in RNN case and 0.1512 to 0.1487 in LSTM case. And here, the most complicated model CNN-LSTM-DNN beats all other DL models, significantly lower the error to 0.1416, much smaller than two-layer LSTM 0.1487.

To complicated dataset, DL models hardly get overfitting, because there is no apparent pattern to be extracted from. Hence, stacking two layer of RNN and LSTM show better achievement than with only one layer; and CNN-LSTM-DNN performs the best, with features extraction by CNN, and time series forecasting by one-layer LSTM. As features are exclusively formed and extracted by a CNN, the required time for training and testing with one-layer LSTM are substantially less than others, almost four times less than two-layer LSTM in training time. This can be proven with the fact, that the one-layer LSTM in CNN-LSTM-DNN is identical to one-layer LSTM, but with only half the training time, indicating the great contribution of CNN in the whole model.

Table 6.2: Metrics of study case households

Model	**MAE**	**MAPE (%)**	**training time (s)**	**testing time (s)**
DNN	0.2649	2.61	83.78	30.92
One-layer RNN	0.1778	3.07	9.32	0.22
Two-layer RNN	0.1447	2.71	11.00	0.26
One-layer LSTM	0.1512	2.70	17.92	0.43
Two-layer LSTM	0.1487	2.71	19.95	0.74
CNN-LSTM-DNN	**0.1416**	**2.62**	**5.598**	**0.42**
ARIMA	**0.2720**	**2.61**	**3.22**	**348**

6.3 Forecasting service of commercial buildings

The API is the last component of the structure presented in Figure 4.3, where a method FORECAST(int:days) is provided. When it is triggered, several values in days ahead are forecast and returned in json format. The default value is one day, however, an arbitrary number of days can be input; the task of forecasting one week or one month ahead is solved. The forecast consumption would greatly benefit the in preparation for one day ahead, and with a pattern in the long future, it has an advantage in managing the process of generating, storing, buying, and selling energy within a microgrid.

The core of the API is built on the selected model from the evaluation sections. In the study case of commercial buildings, Table 6.1 reveals evidently the out-performance of model one-layer LSTM, with 0.3386 in MAE, greatly lesser than other model in the experiments. The model is trained, and its output is saved for further usage when method FORECAST(int:days) in the API is being called. Loading the saved model, the API estimates unseen values with the help of the rolling forecast technique. Lastly is the minmaxscaler, since we used it to normalize the training and testing data, the forecasting one hence needs to reverse the scale from (0,1) to have normal values to return. The sample returned forecast values are captured in the following Figure 6.3 for default value and Figure 6.4 for one month ahead.

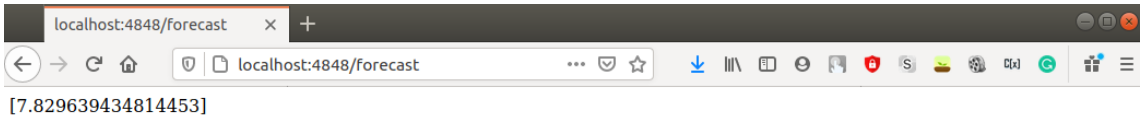

Figure 6.3: Forecasting API - default value

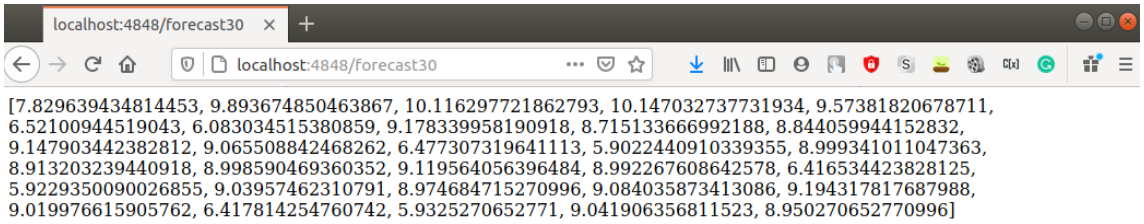

Figure 6.4: Forecasting API - one month ahead

7 Conclusion and Outlook

To apply recent DL methods in forecasting energy consumption, two study cases show the advantages of them over the statistic model ARIMA. Their advantages are mainly from the prominent accuracy, MAE of 0.1416 versus 0.2720 and 0.3386 versus 0.4387 for the next 249 and 110 days ahead in households and commercial buildings study cases respectively; although training them is slower than ARIMA, testing and forecasting are extremely fast, less than 1 second, while testing ARIMA took 348s and over 1641.85s in both cases.

During the thesis, we propose, research, experiment, and report the answers to our three research questions stated, (1) Can we use RNN to forecast energy consumption of commercial buildings and households? (2) Does the performance of RNN family beat ARIMA, considering both accuracy and execution time? (3) Can we build a forecasting service using RNN model to forecast the demand for commercial buildings? These questions are solved, we implement six DL models, from the simple one DNN to the complicated ones with RNN-DNN, LSTM-DNN and , they all perform better than ARIMA with lower error and faster testing time. The one-layer LSTM is the best one in the commercial building study case and hence is used to develop the Forecasting service, which provides an API and is compatible for further practice. With the current Forecasting service, we prove the feasibility in implementation the idea of using RNN to forecast energy consumption of buildings, in this thesis, they are commercial.

Overall, RNN-based models can be used to generate a forecast with promisingly high accuracy, and can be used independently or combined. In combination, a better result can be achieved with the inherited properties. Based on their distribution, different techniques are applied to different datasets, commercial building data could be processed alone, while the case of households fits better with clustered weather and labeled holidays. Stacking repeated models could be both positive and negative approaches, two study cases show completely opposite behaviors. Despite the dominance in forecasting, RNN family has downsides, they are complicated in structure, difficult to train and they require data to be carefully analyzed and pre-processed.

We argue that RNN-based models perform far better than ARIMA in the forecasting problem of energy consumption; in both cases, when energy consumption is used as the only source, and when being supported by other features from weather and holidays. As the single source, when the experimented data is simple with a clear pattern or more complicated data with multiple sources, they surpass ARIMA. In handling dataset, it is trivial that, combining the weather and holidays with energy consumption increases the accuracy by an enormous amount. Training and testing time are also an important factor in the design and deployment of DL models, small training time and minimal testing time are a plus in service development. The functioning Forecasting service could contribute to the fulfillment of any EMS, and particularly the EMS by [GFA20]. The topic of energy consumption forecasting, the current Forecasting service and their connection with a real-life EMS are of interest for future work.

Although the thesis's achievement and contribution are noticeable, there exist some inevitable limitations. In this thesis, our data sources might be inconsistent and biased; we use data from two separate cities, they are San Francisco and London, and at two different timestamps 2004 and 2011. Despite being planned in the design, it is not possible to acquire datasets from a distinct city at the same timestamp, not to mention the real-time data sources. Having a decent outcome, such old datasets are not able to precisely model current buildings' operations. Although the idea and mechanism of the models are applicable and truly beneficial to any forecasting service, the current one might be less ideal to serve any current working microgrid.

In the first study case, we study the average energy consumption of 16 non-residential buildings and attempt to predict the average future demand. There are 11 categories of buildings: restaurant, hospital, hotel, office, apartment, patient building, school, retail building, supermarket, mall, and warehouse. Although the average energy consumption data behaves in a regular seasonality, the situation of a single type of buildings might be different. The operations of them are not similar and might share the same dependency with holidays and weather; the demand for every single building type might vary.

Since the microgrid contains two types of buildings, it is practical to have two methods for each type, and by that, a forecasting service is proper for real-time usage and compatible for an EMS. The current implementation of our Forecasting service only works for commercial buildings, and not for households. We initially target to develop a service only for commercial buildings due to the scope of focusing on the models. At the same time, it is unnecessary to develop a nearly-identical service for households' demand forecasting, because they share the exact implementation process. Model CNN-LSTM-DNN was reported to have the most powerful outcome in the second study case; more weather data must be further collected, and its weight matrices would be stored after training. Finally, in the method, we would extract the last set of data in energy consumption and weather to continuously forecast future values.

Outlook

The thesis lays the foundation in developing a forecasting service in an EMS with the successful delivery of DL models and comparison with ARIMA, it furthermore implement an API, which is accessible from other services. In which, we develop a fully functional forecasting service on one type of buildings, which is commercial buildings. Toward the complete deployment of an EMS, all sites of a microgrid are necessarily regulated, and to that end, another method on forecasting energy consumption of households is required. Before the expansion to real-life application, the processed data must be real life; there is another requirement to acquire real-time data providers on energy consumption and weather. In this section, we propose several improvement strategies on the forecasting models, the development and application of its service.

The first and critical prerequisite is the data; in this thesis, due to the shortage and strict license in data, we have to use the one provided by separate providers located in different cities. In such a microgrid, there are commercial buildings and households; future studies could analyze them together, and more insights about the operation of a microgrid could be provided. Another aspect of the data is the opportunism and reliability, as the data dated back in 2004 and 2011, which is not suitable to the current social movement, the development, and utilization of technology. The

forecast values therefore are inevitably limited in the meaning and application to the nowadays system because of its correctness. Therefore, it is necessary to have real-time data providers for a specific city.

Besides, the newly required data is the key factor, which could upgrade the forecasting models with their real-time property. At the moment, our forecasting service could predict future values, the one which is not in the original energy consumption dataset, however, it is a big constraint in controlling its accuracy, due to the limitation of our data. Having the real-time data set enables the daily checkup, we could compare the forecast and the actual values, and the Forecasting module would be triggered to retrain with regarding to the new and daily updated dataset. By continuously training the model every time new data is out, the model is up-to-date, and more precise prediction is consequently generated. This aspect should be highly considered in deployment; with the availability of data providers, the Forecasting service can contribute to the realization of an ideal EMS.

Lastly, the limitation of the current forecast service is the lack of its method for households forecasting, as detailed in the conclusion, it is possible to implement this method. With the two above improvements, the practice of forecasting the energy consumption of households is made possible thanks to the real-time data providers. Instead of taking only weather and holidays, other features might support the models, for example, the area of the building, its built year, and the number of people inside it. Or further studies can tackle the forecasting problem by finer housing types, for instance, offices, warehouses and schools are unlikely to be affected by weather, while supermarkets, restaurants, and hospitals possibly fluctuate with the changes in weather conditions. By separately analyzing, it could be guaranteed to have even better results.

A Appendix

A.1 Activation functions

An activation function is an important building component of a NN as it determines the output of the model. As the name implies, it decides if a neuron's output should be activated or not based on the weighted sum of inputs and bias. In a NN, an activation function must be efficient, so the model can scale along the increase in the number of neurons. In a NN, the activation functions are aimed to be non-linear which can model and recognize the non-linear relationship. Theoretically, independent of the number of layers in a NN, a linear activation would become useless, since it behaves like a single neuron layer. inputs directly to output without intermediate layers, which is a linear regression. An activation function in NN must be differentiable, enabling feed-forward and back-propagation performance. Given an example: $z = x1w1 + x2w2$, in which $w1, w2$ are weight and z is weighted sum.

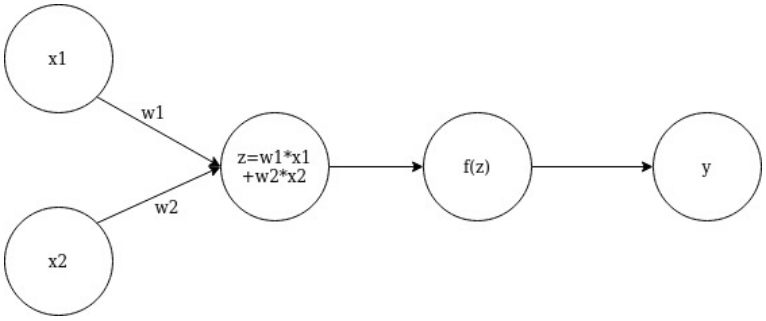

Figure A.1: Sample of a activation function in a neuron

For the illustration of these below activation functions, the images used in this section are extracted from [Sam20][Him19]. These subsections discuss a few popular non-linear activation functions: ReLU, and Tanh.

A.1.1 Rectified Linear Unit

ReLU $relu(x) = max(0, x)$ transfers either the information further or none. Due to the simplicity and non-linearity, it is popular in the implementation a NN. Its derivative is $relu'(x) = \{0|x < 0$ or $1|x > 0\}$ either passes the gradient or not. However, to negative input, the output is filtered, meaning, NN cannot learn from negative values, and is called the dying ReLU problem.

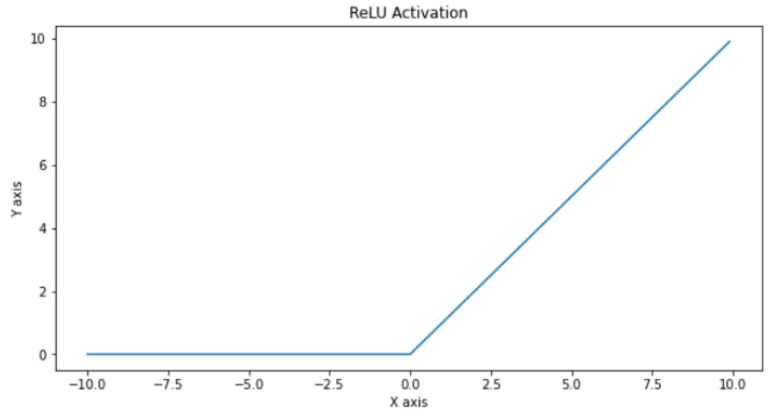

Figure A.2: ReLU function

A.1.2 Tanh

Tanh activation function: $tanh(x) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ and its derivative is $tanh'(x) = 1 - tanh^2(x)$. It behaves like Sigmoid, however, its output ranges from -1 to 1. Besides having Sigmoid advantages, in the work Ëfficient BackProp", Yann LeCun stated that it is 0 center, which is faster in convergence. It is hence widely used and is the best replacement for Sigmoid function [LBOM00]. It is however also suffers from vanishing gradient.

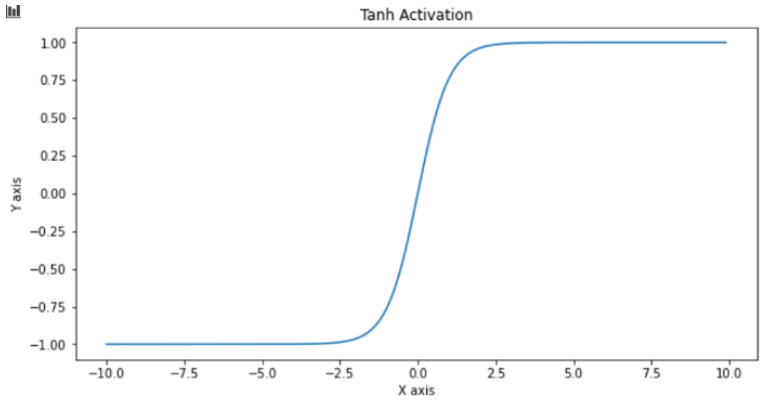

Figure A.3: Tanh function

A.2 Recurrent neural network: extended

Readers might tempt to find the detailed version of RNN, which is fully written in chapter 10 of the textbook "Deep Learning"by [GBC16] and in Chapter 8 in the textbook "Dive to Deep Learning"by D2L [ZLLS20]. This section presents the heuristic approach to train a RNN with its weight matrix, which simplifies the formula while keeping the ideal concept of a RNN; the matrix is denoted as bold uppercase letter while the vector is a bold lowercase letter. The loss function of a RNN

could be calculated as Cross-Entropy or Least Square, depending on the purpose of output. For continuous value number, the loss function is denoted as:

$$L = \sum_{t=1}^{T} L_t$$

$$L_t = \frac{1}{2}(\mathbf{y} - \hat{\mathbf{y}})^2$$

In Figure 2.8, it can be seen that, the output at time t y_t is influenced by h_{t-1} and x_t. A RNN is trained through a process called BPTT, in which the weight matrices are learned with gradient descent technique in Section 2.2.2. Hidden state and output at time t are defined as:

$$\mathbf{h_t} = g(\mathbf{W h_{t-1}} + \mathbf{U x_t})$$

$$\mathbf{y_t} = g^*(\mathbf{V h_t})$$

In the formula, both g and g^* are activation functions, where they must not be the same functions, and is introduced in Appendix A.1. For the ease of explanation, considering a RNN with three layers. In order to perform BPTT, it is mandatory to calculate the following partial derivates $\frac{\partial \mathbf{L_3}}{\partial \mathbf{W}}$, $\frac{\partial \mathbf{L_3}}{\partial \mathbf{V}}$, $\frac{\partial \mathbf{L_3}}{\partial \mathbf{U}}$, and with the ignorance of activation functions assumption. Applying the chain rule, it is deduced as:

$$\frac{\partial \mathbf{L_3}}{\partial \mathbf{V}} = \frac{\partial \mathbf{L_3}}{\partial \hat{\mathbf{y}}_3} \frac{\partial \hat{\mathbf{y}}_3}{\partial \mathbf{V}} = -(\mathbf{y} - \hat{\mathbf{y}}_3)\mathbf{h_3}$$

$$\frac{\partial \mathbf{L_3}}{\partial \mathbf{W}} = \frac{\partial \mathbf{L_3}}{\partial \hat{\mathbf{y}}_3} \frac{\partial \hat{\mathbf{y}}_3}{\partial \mathbf{h_3}} \frac{\partial \mathbf{h_3}}{\partial \mathbf{W}} = -(\mathbf{y} - \hat{\mathbf{y}}_3)\mathbf{V}(1)$$

At point (1):

$$\mathbf{h_3} = g(\mathbf{W h_2} + \mathbf{U x_3}) = g(\mathbf{z_3})$$

$$\frac{\partial \mathbf{h_3}}{\partial \mathbf{W}} = g'(\mathbf{z_3}) \frac{\partial \mathbf{z_3}}{\partial W} = g'(\mathbf{z_3})(\mathbf{h_2} + \mathbf{W} \frac{\partial \mathbf{h_2}}{\partial \mathbf{W}})$$

$$\frac{\partial \mathbf{h_2}}{\partial W} = g'(\mathbf{z_2})(\mathbf{h_1} + \mathbf{W} \frac{\partial \mathbf{h_1}}{\partial \mathbf{W}})$$

$$\frac{\partial \mathbf{L_3}}{\partial \mathbf{U}} = \frac{\partial \mathbf{L_3}}{\partial \hat{\mathbf{y}}_3} \frac{\partial \hat{\mathbf{y}}_3}{\partial \mathbf{h_3}} \frac{\partial \mathbf{h_3}}{\partial \mathbf{U}} = -(\mathbf{y} - \hat{\mathbf{y}}_3)\mathbf{V}(2)$$

At point (2):

$$\begin{aligned}\frac{\partial \mathbf{h_3}}{\partial \mathbf{U}} &= \frac{\partial g(\mathbf{z_3})}{\partial U} \\ &= \frac{\partial g(\mathbf{z_3})}{\mathbf{z_3}} \frac{\partial \mathbf{z_3}}{\partial \mathbf{U}} \\ &= g'(\mathbf{z_3})(\mathbf{z_3} + \mathbf{U}\frac{\partial \mathbf{x_3}}{\partial \mathbf{U}} + \frac{\partial \mathbf{W h_2}}{\partial \mathbf{U}}) \\ &= g'(\mathbf{z_3})(\mathbf{z_3} + \frac{\partial \mathbf{W h_2}}{\partial \mathbf{U}})\end{aligned}$$

while $\mathbf{U}\frac{\partial \mathbf{x_3}}{\partial \mathbf{U}} = 0$, the partial derivative of $\frac{\partial \mathbf{W h_2}}{\partial \mathbf{U}}$ is deduced recursively as:

$$\begin{aligned}\frac{\mathbf{W}\partial \mathbf{h_2}}{\partial \mathbf{U}} &= \mathbf{W}\frac{\partial g(\mathbf{h_2})}{\partial U} + \frac{\partial W h_2}{\partial U}\mathbf{h_2} \\ &= \frac{\partial g(\mathbf{z_3})}{\mathbf{z_3}} \frac{\partial \mathbf{z_3}}{\partial \mathbf{U}} \\ &= g'(\mathbf{z_3})(\mathbf{z_3} + \mathbf{U}\frac{\partial \mathbf{x_3}}{\partial \mathbf{U}} + \frac{\partial \mathbf{W h_2}}{\partial \mathbf{U}}) \\ &= g'(\mathbf{z_3})(\mathbf{z_3} + \frac{\partial \mathbf{W h_2}}{\partial \mathbf{U}}) \\ &= \mathbf{W}\frac{\partial g(\mathbf{z_2})}{\partial \mathbf{U}} \\ &= \mathbf{W} g'(\mathbf{z_2})\frac{\partial \mathbf{z_2}}{\partial \mathbf{U}}\end{aligned}$$

With the above behavior in both forward and backward propagation in training, such NN is referred to as RNN, where in its forward, h_1 is fed to h_2 and h_2 is fed to h_3, and in its backward, the Loss in $t = 3$ is propagated back to its input steps. It is obvious, that RNN is a go-to approach in dealing with the problem of sequential data, it is however very tricky in training, due to the propagation of derivatives [Yos94] [Raz13]. Moreover, theoretically, a RNN can trace back an unlimited amount of neurons, it is in real life, not the case, experiments show very a very limited number before the gradient goes either exploded or vanished.

A.3 Long short term memory: extended

Extending the given presentation of LSTM in Chapter 2, the heuristic way to answer the question "Why is LSTM sufficient to replace RNN?", all the formula is written as scalar value instead of a vector for simple elaboration. Recall in a RNN:

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t}\frac{\partial y_t}{\partial h_t}\cdots\frac{\partial h_2}{\partial h_1}$$

$$\begin{aligned}&h_t = tanh(Wh_{t-1} + Ux_t) \cong tanh(Wh_{t-1}) \cong Wh_{t-1} \\ &\quad\rightarrow h_t \cong W \\ &\quad\rightarrow h_t \cong W^2 h_{t-2} \\ &\quad\rightarrow h_t \cong W^{t-1} h_1\end{aligned}$$

As $W < 1$, the further the gradient goes, the smaller it becomes, and approximately 0 in foreseen short time, and then in case of W > 1, it grows bigger. The gates in LSTM manage to deal with this situation by compensating the gradients the cell, the cell state is the function of previous state and input, and is simplified to $c_t = c_{t-1} + F(x_t) \rightarrow \frac{\partial c_t}{\partial ct-1} = 1$, and the calculation shows the identity relationship between cells; also it is formulated as $c_t = f \otimes h_{t-1} + i \otimes g$, by using forget and input gates, its gradient is balanced out by two gates and can therefore flows freely long before vanishing, it is called Constant Error Flow in [Raz13].

The idea of Constant Error Flow for RNN to create a Constant Error Carousel (CEC) [Sep97] ensures the gradients better behave and do not decay. The cell state at the heart acts like an accumulator, which contains the identity relationship over time.

A.4 Plotting results

A.4.1 Study case: commercial buildings

The tests of the rest five models are shown below:

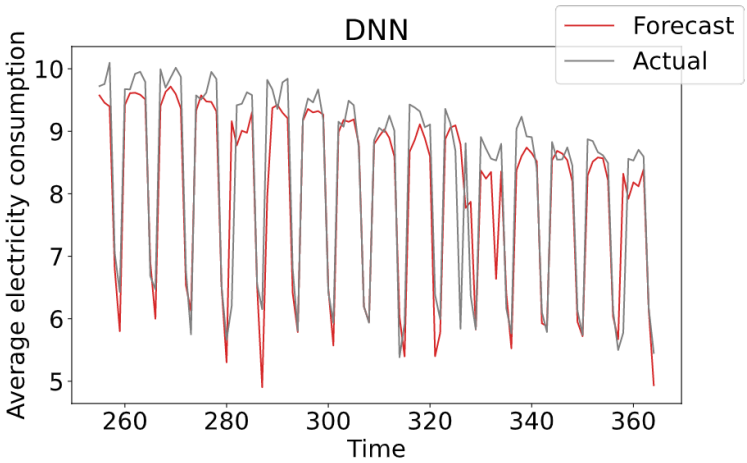

Figure A.4: DNN results

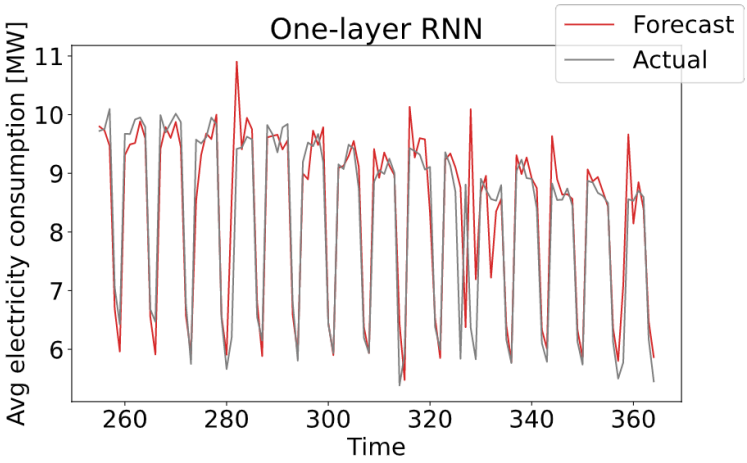

Figure A.5: One-layer RNN results

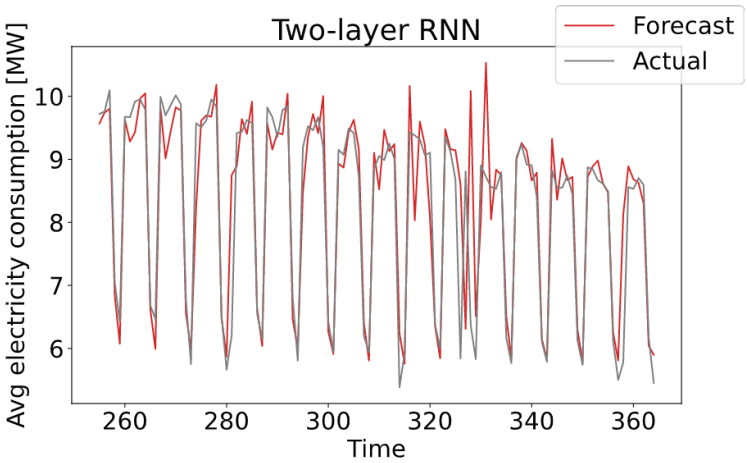

Figure A.6: Two-layer RNN results

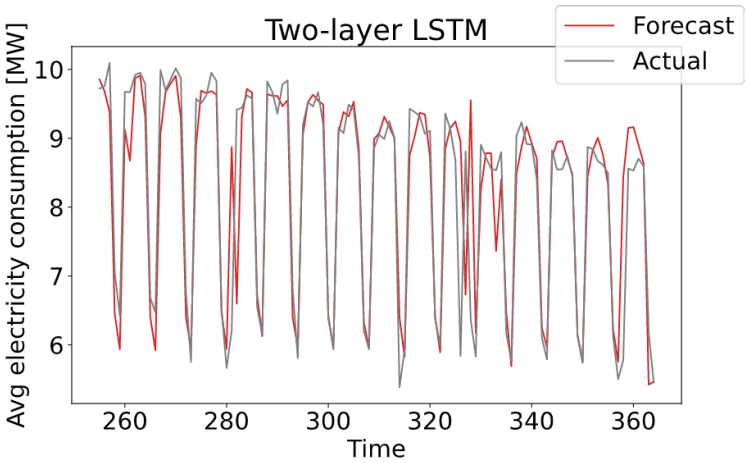

Figure A.7: Two-layer LSTM results

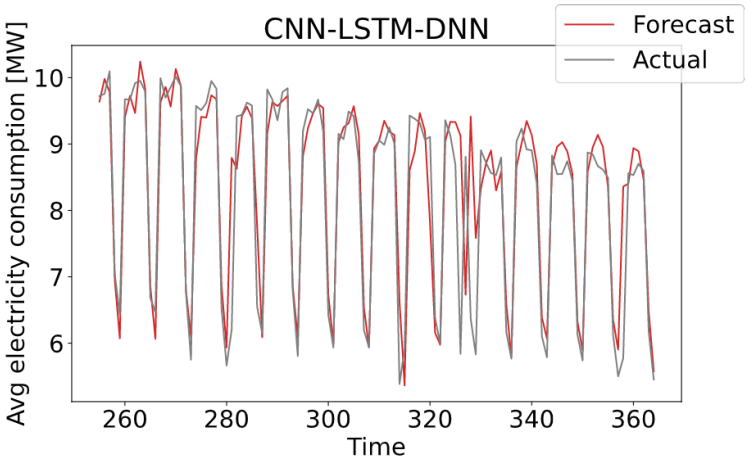

Figure A.8: CNN-LSTM-DNN results

A.4.2 Study case: households

The tests of the rest five models are shown below:

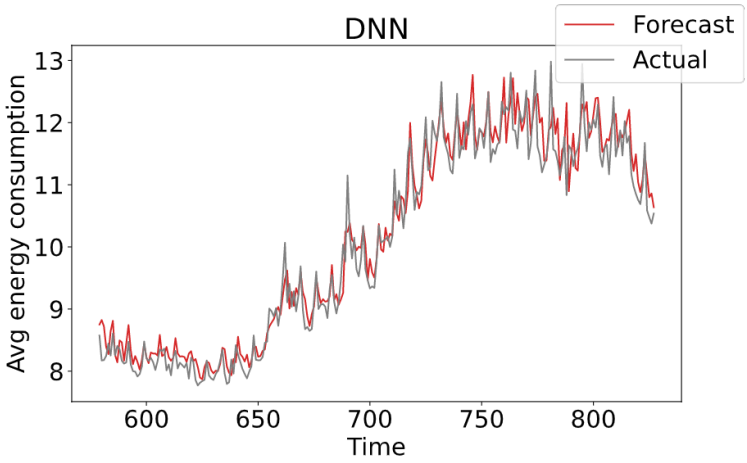

Figure A.9: DNN results

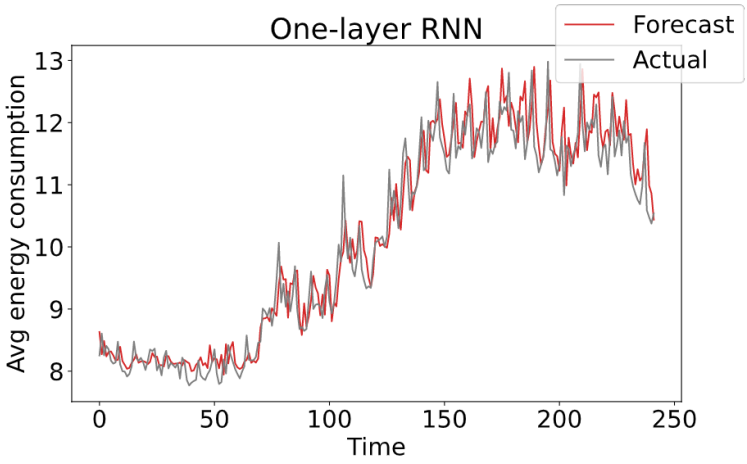

Figure A.10: One-layer RNN results

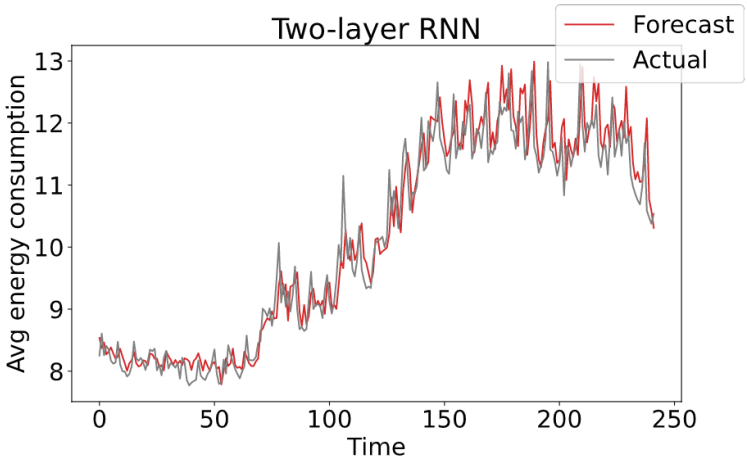

Figure A.11: Two-layer RNN results

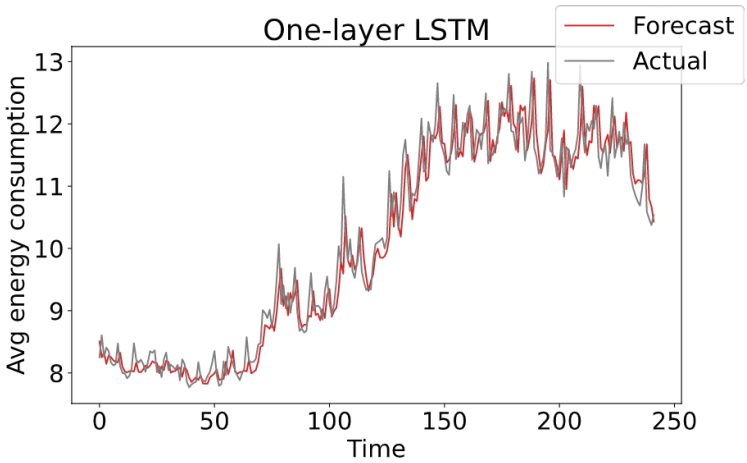

Figure A.12: One-layer LSTM results

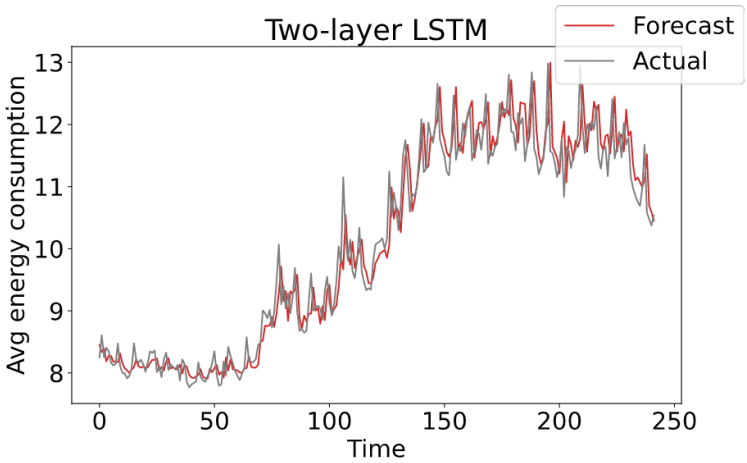

Figure A.13: Two-layer LSTM results

Bibliography

[Ail] Aileen Nielsen. *Practical Time Series Analysis*. https://www.oreilly.com/library/view/practical-time-series/9781492041641/ (cit. on p. 34).

[AJS19] N. Al Khafaf, M. Jalili, P. Sokolowski. "Application of Deep Learning Long Short-Term Memory in Energy Demand Forecasting". In: *Communications in Computer and Information Science* (2019), pp. 31–42. ISSN: 1865-0937. DOI: 10.1007/978-3-030-20257-6_3. URL: http://dx.doi.org/10.1007/978-3-030-20257-6_3 (cit. on pp. 15, 36).

[AN18] M. Abe, H. Nakayama. "Deep Learning for Forecasting Stock Returns in the Cross-Section". In: (June 2018), pp. 273–284. DOI: 10.1007/978-3-319-93034-3_22 (cit. on p. 15).

[And20] Andrew Ng. *CS229*. 2020. URL: http://cs229.stanford.edu/notes/cs229-notes1.pdf (cit. on p. 27).

[BL97] Y. Bengio, Y. Lecun. "Convolutional Networks for Images, Speech, and Time-Series". In: (Nov. 1997) (cit. on p. 34).

[BRF+20] K. Benidis, S. Rangapuram, V. Flunkert, B. Wang, D. Maddix, A. C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, L. Callot, T. Januschowski. "Neural forecasting: Introduction and literature overview". In: (Apr. 2020) (cit. on pp. 36, 59).

[BT19a] J. Bedi, D. Toshniwal. "Deep learning framework to forecast electricity demand". In: *Applied Energy* 238 (Mar. 2019), pp. 1312–1326. DOI: 10.1016/j.apenergy.2019.01.113 (cit. on pp. 15, 36).

[BT19b] J. Bedi, D. Toshniwal. "Deep learning framework to forecast electricity demand". In: *Applied Energy* 238 (Mar. 2019), pp. 1312–1326. DOI: 10.1016/j.apenergy.2019.01.113 (cit. on p. 36).

[C2E] C2ES. *Microgrids*. https://www.c2es.org/content/microgrids/ (cit. on p. 20).

[Chr15] Christopher Olah. *Understanding LSTM Networks*. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (cit. on p. 32).

[Dar] Dark Sky. *Dark Sky has a new home*. https://blog.darksky.net/ (cit. on p. 49).

[ent] entsoe. https://www.entsoe.eu/ (cit. on p. 23).

[FA19] L. Fiorini, M. Aiello. "Energy management for user's thermal and power needs: A survey". In: *Energy Reports* 5 (2019), pp. 1048–1076. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2019.08.003. URL: http://www.sciencedirect.com/science/article/pii/S2352484719300927 (cit. on p. 19).

[GBC16] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 33, 59, 76).

[Geo15] G. M. J. George E. P. Box. *Time Series Analysis: Forecasting and Control 5th*. Wiley, 2015. ISBN: 978-1-118-67502-1 (cit. on pp. 16, 32).

[GFA20] I. Georgievski, L. Fiorini, M. Aiello. "Towards Service-Oriented and Intelligent Microgrids". English. In: (Jan. 2020). The 3rd International Conference on Applications of Intelligent Systems : APPIS 2020, APPIS 2020 ; Conference date: 07-01-2020 Through 09-01-2020, pp. 1–6. DOI: 10.1145/3378184.337821. URL: http://appis.webhosting.rug.nl (cit. on pp. 15–17, 19, 22–24, 35, 39, 71).

[GG11] H. Gharavi, R. Ghafurian. "SCANNING THE ISSUESmart Grid: The ElectricEnergy System of the Future". In: (June 2011) (cit. on p. 19).

[GSC00] F. Gers, J. Schmidhuber, F. Cummins. "Learning to Forget: Continual Prediction with LSTM". In: *Neural computation* 12 (Oct. 2000), pp. 2451–71. DOI: 10.1162/089976600300015015 (cit. on p. 31).

[HCP+14] J. Han, C. Choi, W. Park, I. Lee, S. Kim. "Smart home energy management system including renewable energy based on ZigBee and PLC". In: *IEEE Transactions on Consumer Electronics* 60.2 (2014), pp. 198–202. DOI: 10.1109/TCE.2014.6851994 (cit. on pp. 21, 35).

[Him19] Himanshu Sharma. *Activation Functions : Sigmoid, ReLU, Leaky ReLU and Softmax basics for Neural Networks and Deep Learning*. 2019. URL: https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e (cit. on p. 75).

[Hop82] J. Hopfield. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". In: *Proceedings of the National Academy of Sciences of the United States of America* 79 (May 1982), pp. 2554–8. DOI: 10.1073/pnas.79.8.2554 (cit. on p. 29).

[HS96] S. Hochreiter, J. Schmidhuber. "LSTM can solve hard long time lag problems". In: *Advances in Neural Information Processing Systems* (Jan. 1996), pp. 473–479 (cit. on p. 31).

[HS97] S. Hochreiter, J. Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735 (cit. on p. 31).

[IEA] IEA 2020. *Global energy review 2020*. https://www.iea.org/reports/global-energy-review-2020/renewables (cit. on p. 15).

[Ilc19] M. A. Ilche Georgievski. *Smart Energy Systems Lab course*. 2019. URL: https://www.iaas.uni-stuttgart.de/en/teaching/lectures/2019_ss/ses/ (cit. on p. 21).

[J L15] W. Y. J. Lihu Z. Yongqiang. "Architecture Design for New AC-DC Hybrid Microgrid". In: *IEEE First International Conference on DC Microgrids* (June 2015). DOI: 10.1016/j.rser.2014.11.054 (cit. on p. 15).

[Jal] Jalal Mansoori. *What is Vectorization in Machine Learning?* https://towardsdatascience.com/what-is-vectorization-in-machine-learning-6c7be3e4440a (cit. on p. 66).

[Jas] Jason Brownlee. *Overfitting and Underfitting With Machine Learning Algorithms*. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ (cit. on p. 65).

[JB14] K. C. Junyoung Chung Caglar Gulcehre, Y. Bengio. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: (2014). arXiv:1412.3555 (cit. on p. 32).

[Jea] Jean-Michel D. *Smart meters in London*. https://medium.com/@boitemailjeanmid/smart-meters-in-london-part1-description-and-first-insights-jean-michel-d-db97af2de71b (cit. on p. 53).

[Kap11] S. Kaplan. "Electric power transmission: Background and policy issues". In: (Jan. 2011), pp. 47–85 (cit. on p. 19).

[KB14] D. Kingma, J. Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (Dec. 2014) (cit. on p. 29).

[KBKK96] N. Kohzadi, M. S. Boyd, B. Kermanshahi, I. Kaastra. "A comparison of artificial neural network and time series models for forecasting commodity prices". In: *Neurocomputing* 10.2 (1996). Financial Applications, Part I, pp. 169–181. ISSN: 0925-2312. DOI: https://doi.org/10.1016/0925-2312(95)00020-8 (cit. on p. 35).

[KC18] T.-Y. Kim, S.-B. Cho. "Predicting the Household Power Consumption Using CNN-LSTM Hybrid Networks: 19th International Conference, Madrid, Spain, November 21–23, 2018, Proceedings, Part I". In: (Nov. 2018), pp. 481–490. DOI: 10.1007/978-3-030-03493-1_50 (cit. on pp. 36, 59).

[LBOM00] Y. Lecun, L. Bottou, G. Orr, K.-R. Müller. "Efficient BackProp". In: (Aug. 2000) (cit. on p. 76).

[LG16] S. W. Lee E.-K., W. Gadh R.and Kim. "Design and Implementation of a Microgrid Energy Management System". In: 8.1143 (2016). DOI: https://doi.org/10.3390/su8111143 (cit. on pp. 15, 21, 35).

[LPP20] I. Livieris, E. Pintelas, P. Pintelas. "A CNN-LSTM model for gold price time series forecasting". In: *Neural Computing and Applications* 32 (Dec. 2020). DOI: 10.1007/s00521-020-04867-x (cit. on p. 59).

[M A15] Y.-C. K. M. A. Ahmed Y. C. Kang. "Communication Network Architectures for Smart-House with Renewable Energy Resources". In: *Energies — Open Access Journal* 8 (Aug. 2015), pp. 8716–8735. DOI: 10.3390/en8088716 (cit. on p. 35).

[MAM16] D. L. Marino, K. Amarasinghe, M. Manic. "Building energy load forecasting using Deep Neural Networks". In: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society* (Oct. 2016). DOI: 10.1109/iecon.2016.7793413. URL: http://dx.doi.org/10.1109/IECON.2016.7793413 (cit. on p. 36).

[MH00] S. Makridakis, M. Hibon. "The M3-Competition: Results, Conclusions and Implications". In: *International Journal of Forecasting* 16 (Oct. 2000), pp. 451–476. DOI: 10.1016/S0169-2070(00)00057-1 (cit. on p. 15).

[Mic] Microsoft. *Service-oriented architecture*. https://en.wikipedia.org/wiki/Service-oriented$_{a}rchitecturecite_{n}ote - 2$ (cit. on p. 22).

[MMP97] K. Mehrotra, C. Mohan, S. Preface. "Elements of Artificial Neural Nets". In: (Jan. 1997) (cit. on p. 24).

[Mon14] L. Monostori. "Artificial Intelligence". In: (2014), pp. 47–50. DOI: 10.1007/978-3-642-20617-7_16703 (cit. on p. 24).

[Mun08] T. Munakata. *Fundamentals of the New Artificial Intelligence*. Springer, 2008. ISBN: 978-1-84628-838-8. DOI: 10.1007/978-1-84628-839-51 (cit. on p. 28).

[NSW19] A. Nugaliyadde, U. Somaratne, K. W. Wong. "Predicting Electricity Consumption using Deep Recurrent Neural Networks". In: (2019). arXiv: 1909.08182 [cs.LG] (cit. on pp. 15, 16, 36).

[Nye] Nyenergyweek. *Microgrids – a Solution to Overcoming Large Area Power Blackouts*. http://nyenergyweek.com/microgrids-solution-overcoming-large-area-power-blackouts/ (cit. on p. 20).

[Ope] Openei. *Index of /datasets/files/961/pub*. https://openei.org/datasets/files/961/pub/ (cit. on pp. 47, 48, 50).

[ope] opengroup.org. *Service-Oriented Architecture Standards - The Open Group*. https://publications.opengroup.org/standards/soa (cit. on p. 22).

[PA12] G. A. Pagani, M. Aiello. "Service Orientation and the Smart Grid state and trends". In: *Service Oriented Computing and Applications* 6 (Sept. 2012). DOI: 10.1007/s11761-012-0117-z (cit. on p. 19).

[PA13] G. A. Pagani, M. Aiello. "From the Grid to the Smart Grid, Topologically". In: (2013). arXiv: 1305.0458 [physics.soc-ph] (cit. on p. 19).

[PG03] M. Papazoglou, D. Georgakopoulos. "Introduction: Service-oriented computing". In: *Communications of the ACM* 46 (Oct. 2003), pp. 24–28. DOI: 10.1145/944217.944233 (cit. on p. 22).

[Raz13] Y. B. Razvan Pascanu Tomas Mikolov. "On the difficulty of training recurrent neural networks". In: *Proceedings of the 30thInternational Conference on Ma-chine Learning* 28 (2013). arXiv:1211.5063 (cit. on pp. 30, 34, 78, 79).

[Sam20] Sambit Mahapatra. *Activation Functions in Neural Network*. 2020. URL: https://medium.com/swlh/activation-functions-in-neural-network-eb0ab4bb493 (cit. on p. 75).

[Sar] SarthakGarg. *Service-Oriented Architecture*. https://www.geeksforgeeks.org/service-oriented-architecture/ (cit. on p. 22).

[SCKV15] U. Singh, S. Chauhan, A. Krishnamachari, L. Vig. "Ensemble of deep long short term memory networks for labelling origin of replication sequences". In: (Oct. 2015), pp. 1–7. DOI: 10.1109/DSAA.2015.7344871 (cit. on p. 65).

[Sel20] Selva Prabhakaran. *ARIMA Model – Complete Guide to Time Series Forecasting in Python*. 2020. URL: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/ (cit. on p. 32).

[Sep97] J. S. Sepp Hochreiter. "LONG SHORT-TERM MEMORY". In: *NEURAL COMPUTATION 9* 9 (Nov. 1997), pp. 1735–1780. DOI: https://doi.org/10.1162/neco.1997.9.8.1735 (cit. on p. 79).

[Shu18] Shubham Panchal. *Artificial Neural Networks — Mapping the Human Brain*. 2018. URL: https://medium.com/predict/artificial-neural-networks-mapping-the-human-brain-2e0bd4a93160 (cit. on p. 25).

[Sma] SmartGrid.gov. *The smart grid*. https://www.smartgrid.gov/the$_s mart_g rid/smart_g rid.html$ (cit. on p. 19).

[SN18] S. Siami-Namini, A. S. Namin. “Forecasting Economics and Financial Time Series: ARIMA vs. LSTM”. In: (2018). arXiv: 1803.06386 [cs.LG] (cit. on p. 35).

[SOS18] A. F. Salah Bouktif, A. Ouni, M. A. Serhani. “Optimal Deep Learning LSTM Model for Electric Load Forecasting using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches”. In: *Energies 2018* 11 (2018). DOI: https://doi.org/10.3390/en11071636 (cit. on p. 15).

[SPL17] Y. Shin, W. Park, I. Lee. “Design of microgrid web services for microgrid applications”. In: (2017), pp. 818–822. DOI: 10.1109/ICUFN.2017.7993913 (cit. on pp. 15, 21, 35).

[STS18] S. Siami Namini, N. Tavakoli, A. Siami Namin. “A Comparison of ARIMA and LSTM in Forecasting Time Series”. In: (Dec. 2018), pp. 1394–1401. DOI: 10.1109/ICMLA.2018.00227 (cit. on p. 35).

[SXl17] H. Shi, M. Xu, R. li. “Deep Learning for Household Load Forecasting – A Novel Pooling Deep RNN”. In: *IEEE Transactions on Smart Grid* PP (Mar. 2017), pp. 1–1. DOI: 10.1109/TSG.2017.2686012 (cit. on p. 15).

[TS12] D. Ton, M. Smith. “The U.S. Department of Energy’s Microgrid Initiative”. In: *The Electricity Journal* 25 (Oct. 2012), pp. 84–94. DOI: 10.1016/j.tej.2012.09.013 (cit. on p. 19).

[UK] UK Power Networks. *SmartMeter Energy Consumption Data in London Households*. https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households (cit. on p. 53).

[Unk] Unknown. *Momentum and Learning Rate Adaptation*. https://www.willamette.edu/ gorr/classes/cs449/momrate.html (cit. on p. 29).

[wea] weatherbit. https://www.weatherbit.io/ (cit. on p. 23).

[Wika] Wikipedia. *Smart grid*. https://en.wikipedia.org/wiki/Smart$_{g}rid$ (cit. on p. 20).

[Wikb] Wikipedia. *World energy consumption*. https://en.wikipedia.org/wiki/World$_{e}nergy_{c}onsumption$ (cit. on p. 15).

[YOM+17] Y. Yoldaş, A. Onen, S. M. Muyeen, A. Vasilakos, I. Alan. “Enhancing smart grid with microgrids: Challenges and opportunities”. In: *Renewable and Sustainable Energy Reviews* 72 (May 2017), pp. 205–214. DOI: 10.1016/j.rser.2017.01.064 (cit. on pp. 19, 20).

[Yos94] P. F. Yoshua Bengio Patrice Simard. “Learning Long-Term Dependencies with Gradient Descent is difficult”. In: *IEEE Transcations on Neural Network* 5 (Mar. 1994) (cit. on pp. 30, 78).

[Z2L] Z2Little. *Gradient Descent: Stochastic vs. Mini-batch vs. Batch vs. AdaGrad vs. RMSProp vs. Adam*. https://www.weatherbit.io/ (cit. on p. 28).

[ZLLS20] A. Zhang, Z. C. Lipton, M. Li, A. J. Smola. *Dive into Deep Learning*. https://d2l.ai. 2020 (cit. on p. 76).

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature