Deep Learning-based Time Series Forecasting for Smart Industries

Wen SONG

Graduate School of Information, Production and Systems

Waseda University

A thesis submitted for the degree of

Doctor of Engineering

January 2022

Thesis Supervisor: Professor Shigeru Fujimura

Abstract

With the rapid development of the Internet and big data, the intelligence and informatization of the production process have become an essential part of the process. In order to make the production process more efficient and transparent, many industrialized industries have responded to the concept of "Industry 4.0" by integrating people with computers, production systems, and communication systems through the concept of the Internet of Things to achieve safe, reliable, real-time, coordinated sensing and control of production processes. So far, the use of industrial data remains the most important aspect of industrial informatization. If industrial data can be fully utilized for analysis and scheduling, the production cost of the industry will be greatly reduced in consumption and industrial efficiency will be increased accordingly.

Among the utilization of industrial data, the forecasting of industrial time series has significant value. By accurately predicting the future trend of time series, the resource allocation for industrial production can be effectively deployed in advance to avoid losses. For manufacturers, accurate time series prediction can also help decision makers to make important decisions supported by data due to the complexity of the production environment. Therefore, time series forecasting is a very interesting topic to explore.

In the real world, the model frameworks suitable for forecasting are completely different because the samples for time series forecasting are different and the targets for forecasting are different. Therefore, this thesis investigates to solve several time series forecasting problems in the real world. First, an electricity consumption time series forecasting problem is investigated. This problem is a univariate time series problem and has the special objective that the forecasted value is larger than the ground truth. So this study combines the latest deep learning methods to foreacst the sequence using an attention-based mechanism of the encoder-decoder model and proposes a customized loss function suitable for forecasting values larger than the ground truth. The proposed method has significant advantages over benchmark methods in the same field.

This study then generalizes the type of subject studied from univariate time series forecasting to multivariate time series forecasting, and the data is shifted from power consumption forecasting to more common sensor data prediction. The study has a targeted pre-processing of the data due to the complexity of the data itself. In the study, noise was removed by applying the wavelet transform, delay between sensors was removed using correlation coefficients, and finally, appropriate sensor data was selected using correlation coefficients to form our dataset. Then, we propose a novel deep learning framework that can capture different length of time dependencies well. The experimental results show that our algorithm has better performance on the sensor dataset compared to previous studies.

Finally, we continue to extend our experimental subjects from sensor data to diverse time series forecasting. To capture the impact of unknown factors on the target variable forecasting, we propose a new deep learning framework to solve the MTS forecasting problem. In this framework, we apply several dilation convolution filters to capture the dependencies of all multivariate variables in parallel for different time lengths. In summary, the mixed dependence of long- and short-term factors in multivariate can be well captured by the proposed framework, which can capture both the complex effects between multivariate variables and to some extent the impact of additional factors on the current forecast. The proposed framework's effectiveness was evaluated on several benchmarj dataset and yields competitive results compared to those of several baselines methods.

Keywords: time series forecasting, deep learning, time dependency, recurrent neural network

Acknowledgements

Finally, after four springs and four autumns, I have also come to the final hurdle of my PhD and a major turning point in my life. Only now do I realize that my difficult but challenging PhD career is coming to an end, and here I finally have the opportunity to record the hardships as well as the joys and excitement along the way. Looking back, I am fortunate. During the long years of pursuing my PhD, I had the strong support of my family, the patient guidance of my supervisor, and the gentle care of my love.

Professor Fujimura was my mentor for six years of research, and his wisdom and diligence were a great help and inspiration to me. From the day I took my master's degree at Waseda University, I already dreamed of the day I would do a PhD. I still remember that when I first entered the lab and took the initiative to tell my professor that I wanted to study for a PhD, he just smiled kindly. At that time, I was naive, I didn't understand the persistent efforts required for a PhD, and I didn't understand the courage I needed to have.

Soon, two years of master's studies passed quickly, and I continued to do the research I liked under Professor Fujimura as I had hoped. But the research process was boring and tedious. I still remember my first year of PhD from the beginning of confidence to the doubt of my academic ability. I was confused and worried when I saw the pathetic results of my experiments. In the same year, Mom was diagnosed with depression and became very anxious and depressed. Mom was crying on the phone, wanting me to come home, but it was not yet vacation time and I could not do anything about it while I was studying far away in Japan. I was so worried about her that I could only insist on sending her messages every day and making phone calls now and then to try to help my mom relieve a little of her stress. Fortunately, I had the unwavering support of my family, and although I had nothing to show for my research this year, my family was safe and I still made it through the year.

It is worth mentioning that I met the most important person in my life, Liying, during the first year of my PhD. During my most confusing

time, she was like a ray of light that dispelled the fog inside me. When I was with her, she always prepared my favorite food for me and took the initiative to understand my every mood. I am an introvert myself, but she always opens my heart. She enlightened me when I was lost and encouraged me when I was productive. During that time, we both gained weight, but I wasn't afraid and confused. Because I knew I wasn't alone anymore.

Finally, in the 3rd year of my PhD career, I managed to publish my first journal article. This one paper took me almost a year just to revise. I still remember my excited heart and trembling hands when I received the acceptance email. I read the email over and over again, afraid that I had read it wrong. I am really grateful to my professor Fujimura, I don't know how he put up with reading my inconsistent draft over and over again, and how he patiently guided me to revise the paper little by little. After this paper was accepted, it seems that I slowly understood the right way to do research, and I started to try new models and do experiments.

My PhD career hit a bump in the first half of 2021. This was the year I started to face the problem that every International student faces: finding a job. I initially avoided looking for a job because I was afraid that I would not be able to graduate because of the time taken away from doing my research, and I was even more afraid of leaving Japan and being separated from Liying because I did not find a job. Therefore, I was always looking for excuses to postpone my job search and wanted to spend all my energy on my research. But I knew in my heart that my life was not all about research, and that I would have to take this step one day. Finally, with LIying's persuasion, I tried to allocate my time rationally and do both job hunting and research at the same time. This process was very painful. In the first month, I was able to arrange my time reasonably to complete both tasks, but from the second month onwards, I started to become anxious. I submitted my paper to the journal for review, but no response was forthcoming. During those months, I was checking my email every day, hoping for good news. My anxiety peaked in August, when I was checking the journal website more than a dozen times a day for updates on my dissertation results. During this process, my mother was patient and helped me to share my stress, while my father was confident and encouraged me not to worry. Liying stood behind me as always, encouraging me, guiding me, and taking me on trips. Finally, the acceptance email of my paper by the journal came into my mailbox

on that ordinary and extraordinary morning. I suddenly felt that all the efforts I had made were worthwhile.

I would like to thank professor Shigeru Fujimura, my mentor, for your patience and continued support, without which I would not have been able to do the research I have done today. You have not only guided me in my research, but also cared for me and encouraged me in my life. I have given you so much trouble in the past six years, but you are always smiling and gentle. I hope you will always be healthy and happy.

I also want to thank my family. Studying abroad is very costly, and I have always had you to provide stable financial support for my study career, so that I never worry about living. You have always had confidence in me, believed in every decision I made in my research career, and always supported all my decisions. I know that I am not enough of a son to be with you all the time during my years as an international student, and I dare not ask you about your mental activities when other children are earning money for their families. I can only thank you for the kindness of raising me for so many years.

I would also like to thank Liying. For the four years, everything about you has become a part of my life. You supported me in both my life and studies. In life, you taught me how to deal with every situation. Academically, every scientific progress I made became your greatest joy. You cooked me delicious food and bought my favorite ones, as if you held me in your hands and cared for me. It is a blessing for me to meet you in the crowd. I look forward to spending the rest of my life hand in hand with you in the future.

Finally, I would like to thank my good friends, especially Bo yang, we help each other in life and support each other. We talk about everything in the dormitory. I wish you all the best in the future and success in your career. In fact, there is a lot more I want to say, but it seems pretentious to say it again. That's it. I'll leave it at that.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and motivation

Since entering the 21st century reliance on the internet and industrialization is forming new types of industrial competencies rapidly, the productive capacity of industry was gradually raised to an unprecedented level. This is a new industrial revolution, also known as "Industry 4.0" by the German Academy of Technology (CDTECH) and other institutions [1]. Industry 4.0 is a revolutionary change from the previous three industrial revolutions and is centered on the deep integration of information systems and production systems.

Industry 4.0 is a technological transformation of an industry and an industrial change. The smart manufacturing proposed by Industry 4.0 is oriented to each production cycle of products to achieve highly efficient information-based manufacturing. Smart Industry is based on recent developments in sensing technology, high-speed network transmission and the Internet of Everything. As well as booming of AI, based on perception, the need for human-machine interaction has promoted information intelligence in the factory production process, and has made information technology perfectly integrated and promoted the development of manufacturing technology.

For traditional industries, industrial data is simply provided by machines, while all decisions and planning need to be made by experienced workers on their own. Since it takes time and effort to train an experienced worker, sometimes it is even difficult to give the final result by workers. Compared with traditional industries, smart factories emphasize more on automation and intelligence of industries. More specifically, when workers need to arrange future planning or need to make decisions about a process, machines operate automatically, or provide reliable data reserves or even speculation and opinions intelligently, thus allowing workers to have a more accurate understanding of industrial information and make better

planning and deployment.

With the popularity concepts of "Industry 4.0" and "Smart Industries", a variety of industrialized production is facing a major need to transform to smart manufacturing. With the speedy growth of the Industrial Internet of Things (IoT), time series data plays an important role in the IoT of smart factories. The time series generated by industrial equipment is different from the time series in nature. Due to the complex production process, the large number of sensors and fast sampling frequency, It is easy to stack a large amount of data in a relatively short duration. It mainly presents the characteristics of complex mechanism model, time series arrangement, strong data dependency, high data dimension and a large amount of unlabeled data, etc., which will often produce large economic losses if special working conditions occur. Therefore, the timely and accurate forecasting of time series in the production process will improve the efficiency along the whole production procedure, which will have a greater application value. In the production process, if the time series can be accurately forecasted, it can improve production efficiency, enhance product quality, save cost, etc.

Through the research, the author summarized the research difficulties of industrial time-series data quality management, as follows [2] [3] [4] [5]:

- In addition to the basic characteristics of industrial big data such as large scale, fast speed, miscellaneous types and low quality, it also has new characteristics such as strong correlation and high delay, i.e., multi-modal and variable industrial data, whose data characteristics lead to the traditional data forecasting model cannot be well applied to industrial time-series data.

- Industrial time series data is different from other time series data. For complex industrial time series data, correlation among different variables should be emphasized. At the same time, the noise of industrial data also leads to an increased probability of failure of preparation. Therefore, the pre-preparation work and post-evaluation validation work become more important.

- Industrial time-series data has typical high dimensional characteristics. And most of the current time series models focus on forecasting low-dimension, periodic or simple pattern data, which is difficult to effectively forecast the high-dimension time series data.

In traditional industries, machines can only present industrial time series data to workers but do not contain predictions based on historical data. For workers, they need to use this existing historical data to make reasonable and accurate judgments or even predictions. These judgments and predictions require long hours

of work and training in the factory for workers to learn, which means extremely high time and labor costs.

In smart industries, the human-machine interaction is further increased and machines are able to give detailed analyses and, to a certain extent, predictions based on historical time series data. If prompt and accurate predictions can be made, it will be easier for workers to access the information they need to make future decisions based on the predictions and reduce labor costs. At the same time, due to the speed and stability of the machine, it will give more accurate results in less time and avoid unexpected losses.

In this dissertation, the industrial smartification using time series prediction is discussed in three directions:

- The first is about electricity consumption forecasting. Currently, the electricity consumption in the next day is forecasted by a worker based on information like: the electricity time series data of the previous few days, past long-term trends, and information that affects electricity demand such as weather and events. However, the accumulation of past experiences is very important to make such forecasts. If those experiences can be modeled to automatically and accurately forecast power consumption in a short period of time, then the worker only needs to make a final decision, and he/she can focus on other more intelligent works.

- The second is about reducing the workload of plant operators in chemical plants. In the past, operators had to monitor a great deal of sensor data in order to keep the plant running properly. Such kind of process imposes a heavy workload on the operator. Therefore, in steady-state plant monitoring, many systems are used to notify the operator of a possible abnormal situation by alarming with threshold settings for operator support. However, it is not easy to set the threshold value because it needs to be changed according to the plant conditions. In contrast, the system displays trend of the time series prediction of the sensor data. If the trend shows an abnormal condition in the future, the operator can easily detect it, and the monitoring work can be significantly simplified.

- The third is to make such time series prediction available in other fields. Humans would be able to focus on more human-like works and make even better decisions.

In these ways, we believe that the time series forecasting models proposed in this paper can realize smart industries as various applications, and human work in traditional industries can be more intelligent.

1.2 Univariate and multivariate time series forecasting

Observing data at a series of points in time is a common place activity in smart industries, and industrial production contains a large variety of time-series data. Generally, time series forecasting refers to making forecasting about the possible future values based on the historical data, along with other relevant series that may have an impact on the outcome. There are many real-life time series data forecasting problems, including power consumption forecasting, sensor signal prediction, etc. Its essence is essentially to forecast the value of a time series at a future point in time on the basis of previously observed data.

In terms of overall time series forecasting, we can divide all studies into two categories: univariate time series forecasting and multivariate time series forecasting:

- Univariate Time Series forecasting (UTS).

  Univariate time series forecasting in the traditional sense is a forecasting problem which considers only the effect of a single variable on itself and does not consider the effect of other variables on that variable. However, with the development of research, the concept of univariate time series forecasting gradually leads to the study of forecasting the time series of one variable using multiple variables of relevance [6] [7] [8] [9]. These studies focus more on the effects of other relevant variables on the target variable and ignore the dependence of the target variable on the relevant variable and the variables of interest. The capture of the dependence of the target variable also becomes more precise due to the reference to additional variables of interest, resulting in accurate results.

- Multivariate Time Series forecasting (MTS).

  Multivariate time series do not only depend on time itself. The forecasted values of multiple target variables not only depend on their own historical data, but also receive effects as other variables change. The dimensions of multidimensional time series are highly correlated, and data processing should consider all dimensions together to avoid loss of valid information. In the actual forecasting process, each time step is observed not only for a single variable Y_t, but also for multiple variables $(X_t^1, X_t^2, \cdots, X_t^r)$ at the same time, which requires the analysis of multivariate time series $X_t = (X_t^1, X_t^2, \cdots, X_t^r)^T$. For example, in the analysis of weather preidction it is necessary to consider simultaneously rainfall, temperature and barometric pressure data, not only to study each of their components as a univariate process, but also to study

the relationship and affects between the relevant components, so as to make forecasting and control the time series.

With the advancement of industrialization and the improvement of product quality, industrial production has put forward higher requirements on the monitoring of key process variables. Most of the industrial processes have complex multivariate characteristics, and the traditional monadic time series forecasting methods cannot be adapted to the actual environment and application requirements. The multivariate time series forecasting considers interrelationship of multiple varaibles and has good application prospects for industrial process time series forecasting. In general, multivariate time series forecasting is the most frequent and widely used problem in time series.

1.3 Scope of previous researches

Time series forecasting under smart industry is a relatively new topic. However, with the development of the smart industry and the rise of machine learning research, this area has become rich in research.

For UTS forecasting, Newbold et al [10] first proposed this research. They compare the forecasting performance of Box-Jenkins, Holt-Winters, and stepwise autoregression in the area of economic time series. They also suggested the possibility of combining special variable predictions when forecasting for a single sample. Then, Nogales et al.[11] apply time series to electricity price forecasting as a way to maximize the benefits to electricity producers. They provide two accurate and efficient price forecasting models: dynamic regression and transfer function models, and explain and cross-check these techniques. Hyde et al. [12] propose an automated load forecasting system and present a forecasting model with linear regression of electrical consumption and climate data, and error statistics for the forecasted load for the day ahead. The study can adapt to changing climate factors and achieves forecasting accuracy and robustness. Pai and Hong [13] treated the forecasting electric load as a nonlinear regression sequence problem and applied support vector machine (SVM) to validate its viability of the algorithm for predicting electricity. In addition, they use a heuristic algorithm simulated annealing (SA) to choose the proper parameters for their model. The proposed approach is more accurate compared to ARIMA and general regression neural network (GRNN) models. This is also the first time machine learning has been applied to electricity forecasting. Marino [14] present a deep neural network-based load forecasting method. Their model is based on the architecture of LSTM and employs the first sequence-to-sequence (S2S) architecture. Their results are a significant improvement compared to previous studies. Chandramitasari et al. [6] propose a deep learning framework that combines long short-term memory (LSTM) and feedfor-

ward neural network (FFNN) for power forecasting. In their study, for the first time, FFNNs are applied for processing extra information to improve the accuracy. In this dissertation, the study in Chapter 3 is a further deep dive based on the study of [6].

For MTS forecasting, Jenkins et al. [15] They proposed for the first time the use of a multivariate stochastic model for forecasting, which encompasses a variety of possible phase shifts. With the popularity of artificial neural networks in the 1990s, Chakraborty et al. In [16], a neural network approach for multivariate time series analysis is presented for the first time. In their experiments, feedforward neural networks were designed to simulate flour prices in three U.S. cities over an eight-year period. Their model achieved significant success in price prediction compared to previous studies. In recent years, with the advancement of hardware technology and the rise of deep learning, deep learning-based forecasting methods have gradually become mainstream. Among them, the most important and seminal research is LSTNet [4]. In this research, they propose a new deep learning framework that combines CNNs and RNNs to extract patterns of local dependencies between variables. They also propose a jump connection to handle long-term dependencies. Overall, their study is the first to treat different terms of dependencies separately. In this dissertation, the studies in Chapters 4 and 5 are further in-depth studies based on the study of LSTNet.

Specifically, there are many cases where "prediction" or "forecasting" is used. It is important to note that usage of these 2 words can differ between areas of application. In this dissertation, on the one hand, the ultimate goal of electricity forecasting is to simulate the actual time series data. In the sense of forecasting detailed time series data, the word "forecasting" is used. While on the other hand, prediction in chemical preocess is commonly used for abnormal detection. Comparing with "forecasting", the term "prediction" is used for more general estimates such as whether there would be an error or not. Therefore, the word "prediction" is used for time series forecasting of chemical processes.

In the following chapters, "forecasting" is used in chapter 3, which indicates the forecasting of electricity consumption. The word "predicting" is used in chapter 4 to describe predictions for sensor data in chemical process. In chapter 5, since there are several kinds of data, and the main target is to forecast the detailed future value of these data. Hence, "forecasting" is used to describe detailed forecasting results. Specifically, in other conditions, in the process of forecasting, such as "time series forecasting", "forecasting result", "forecasting models", etc., "forecasting" is used in the conditions which aims to generate detailed values only.

Table 1.1: Research overview

	Chapter 3	Chapter 4	Chapter 5
Forecasting data	Electricity load Data	Sensor and exchange rate data	Various industrial data
Forecasting type	Univariate	Multivariate	Multivariate
Proposed approach	Attention-based Encoder-Decoder model	A deep learning framework with Mixed Length Dilation Blocks	A deep learning framework by capturing combination patterns of long- and short-term dependencies
Publication	- Journal (J2): IEEJ Transactions on Electronics, Information and Systems	- Proceedings (C1): 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) - Proceedings (C2): 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)	- Journal (J3): Neurocomputing

1.4 Research proposal and contributions

This study aims to extend the time series research for the smart industry. A comprehensive research proposal is presented in table 1.4. The proposal presents several frameworks for forecasting time series with different dataset.

- UTS forecasting with attention-based Encoder-Decoder model,
- MTS forecasting with the proposed MLD framework,
- MTS forecasting with the proposed MDT framework.

Chapter3 investigates the deep learning-based power forecasting problem. The problem comes from the study of Sari et al. [6] and is a unitary time series forecasting problem. An attention mechanism-based approach is proposed that separates variables internally and externally and addresses the problem of capturing long-term dependencies. The method is compared with the standard ARIMA and the study of Sari et al. Numerical experiments show that the usage of attentional mechanisms was efficient in capturing long-term dependence. This study established a special objective function based on the actual situation of PPS to achieve the goal that the foreacasting value is greater than ground truth in general. The results of this study were published in the [17] journal.

In Chapter 4, the results obtained from Chapter 3 are generalized to more complex sensor datasets. Since the sensor data is more complex with huge amount of delay and noise, in Chapter 4 we need to preprocess the dataset and select the useful sensor data to compose the dataset. Firstly, wavelet transform is used to denoise the sensor noise, and then the mutual relation number is used to remove the delay between sensors, and finally, pearson and spearman correlation coefficients are used to select the useful sensor data.

In Chapter 4, for the multivariate time series problem, a novel deep learning framework is proposed. By customizing proposed Mixed Length Dilation Blocks, the mixed length of dependencies among related sensor data is captured well. Experiments show that the proposed framework yields opposing results on all benchmark datasets compared to those of several baselines methods. The results of this study were disseminated at a refered conference and published in the [18] Proceedings.

In Chapter 5, we extend the sensor data prediction from Chapter 4 to the entire industrial dataset and further enhance the model's ability to capture the effects of unknown factors. A new deep learning framework named Mixed Dependence Time Series Network (MDTNet) is proposed for the MTS forecasting problem. First, stacked dilation convolution component is first employed, which applies multiple dilation convolution filters to capture all multivariate dependencies of different

time lengths in parallel. Then the dependency combination component, which uses vanilla convolutional filters to deconstruct complex combinations between different dependencies. Finally, recursive component, which applies a recursive layer to capture the variation of different combinations over all time steps. The experimental results show that the proposed framework presents a significantly higher accuracy for most of the data sets, and the future of the model has a promising future. The results of this study were published in the [19] journal.

Here are the details of each point:

1. All models in this study use real data from real-world smart factories. The first of these models corresponds to traditional power load forecasting, the second corresponds to multivariate prediction of sensors in a factory, while the third could be generally applied to data from various of smart factories.

2. In the second model, depending on the complexity of the plant sensors and the actual computational power, workers can customize the number of blocks to select a more appropriate model.

3. In the third model, without setting jump parameters in advance, the model captures the extra-long dependencies, which is suitable for those data with long term dependencies.

4. In terms of model complexity, the complexity of the three models increases sequentially, and the accuracy and robustness of the data forecasting also increases sequentially.

In summary, the main contributions of this dissertation are summarized below.

1. In power forecasting, the problem is the same as in the study of Sari et al. [6] is the same. Based on their study, we overcome the capture problem of long-term dependence and rationalize the influencing factors into internal and external factors to achieve more accurate forecasting.

2. In sensor data prediction, we set out to predict multivariate time series using deep learning methods. Based on previous studies [5] , we proposed a customized block structure and a new framework for sensor data prediction that can capture the mixed length of time dependencies. Extensive experiments are conducted on several benchmarks for comparing our proposed framework with other baselines, and the experimental results prove that our framework performs competitively on all the datasets.

3. In the final intelligent industrial data forecasting, a new deep learning framework called MDTNet is proposed. The mixture dependence of long-term

and short-term factors among multivariate can be well captured by applying stacked dilated convolution with vanilla convolution and recurrent networks. This study builds on [4] to enhance the capturing of blends and to overcome the need to preset the "recurrent jump" length in advance. Experiments show that our proposed framework produces competitive results on all benchmark datasets compared to several benchmark methods.

1.5 Outline of this dissertation

The remaining parts of this deissetation is structured as follows.

Chapter 2 is dedicated to related works on different kinds of time series forecasting research methods. The history and current status of research on time series forecasting is described from traditional methods, time series decomposition, feature engineering, and deep learning methods.

Chapter 3 proposes a deep learning model which applies attention mechanism to forecast electricity consumption. The background of this study is similar to LSTM-FFNN [6], but the model is further improved and a special objective function is built according to the actual demand. Experiments indicate that the proposed model can fit the power consumption more thoroughly in several horizontal comparisons.

Chapter 4 solves the multivariate forecasting problem at sensor values. Due to the complex and noise-filled sensor data, we perform the necessary preprocessing on the raw sensor data. Subsequently, a novel deep learning framework is proposed. This framework can better capture the diverse time related dependencies among sensors. Numerical experiments show that the proposed framework achieves superior results on all three datasets compared to other popular methods.

Chapter 5 builds on Chapter 4 by extending the study to diverse industrial time series. In that study, we not only try to catch the influence of relevant varaibles on target series, but also try to analyze the influence of the unknown variables on the target variables by analyzing the time-dependent mixture patterns of multiple lengths. The evaluation results in the cross-sectional comparison experiments have demonstrated the superior performance of our proposed framework in the large horizon case, but there still remains room for improvement in proposed framework for multivariate sequences of very high dimensionality.

Chapter 6 summarizes all the studies and discusses possible directions for future research.

Chapter 2

Related works

From a mathematical point of view, time series forecasting indicates to use the characteristics of the time of information in the past period to predict how this information will change in the future period. This is a more complex forecasting problem than the regression analysis model. In the time series forecasting problem, the sequence of historical events affects the trend of the time series and never brings different effects on the possible future values.

The study of time series forecasting has a long history. In this chapter, each of the following kinds of time series methods would be introduced separately in each section.

2.1 Traditional time series modeling methods

Among all the methods, the traditional time series models like statistical methods account for a considerable proportion[20][21][22]. One of the most commonly used models is called the autoregressive integrated moving average model (ARIMA) [8], which subsumes autoregression, moving average, and autoregressive moving average [23]. Specifically, the ARIMA model combines three basic approaches[24]:

- Autoregressive (AR) models describe the relationship between the current values and historical values. Forecasts are made using historical time data of the variables as input.
- Moving average (MA) models describe the moving average nature of error term delayed in time.
- Differencing (I-for Integrated) refers to the differencing of time series data. By the differencing operation, the time series will be converted from a non-stational state to a stational state.

The detail of ARIMA is listed as follows:

$$\left(1-\sum_{i=1}^{p'}\alpha_i L^i\right)X_t=\left(1+\sum_{i=1}^{q}\beta_i L^i\right)\gamma_t \tag{2.1}$$

where L indicates the lag operator, the α_i denotes the autoregressive coefficient and and β_i denotes the moving average coefficient, respectively. γ_t denotes the error terms.

The success of the ARIMA model is owing to its robustness against nonstationary data and interpretability of the statistical characteristics. By applying the Box–Jenkins method [25], the best fit of the ARIMA model is calculated based on the historical data. However, owing to its high computational cost and methodology, the ARIMA model is more conventionally used in univariate time series forecasting.

On the other hand, to solve MTS forecasting problems, the VAR model, an autoregression model that extends from a univariate to a vector scale, was proposed[26]. It is extensively used in solving MTS problems owing to its ability to capture the linear interdependencies among different variables. VAR models are one of the easiest models to operate in dealing with the analysis and prediction of multiple variables of interest are performed. And under certain conditions, multivariate MA and ARMA can also be converted into VAR models, so VAR models are recently favored by more and more researchers [27].Since it was proposed, many VAR-based models, including VARMAX[28], elliptical VAR[21], and structured VAR[20], have been constructed. However, in capturing long-term dependencies in the time dimension, VAR-based models perform relatively poorly when encountering high dimensional data. In case of a high dimensional simulation over multivariables, VAR-based models can be easily overfitted. Thus, to better fit such models, some studies applied regularization[22].

However, the research on VAR-based models has a bottleneck owing to the linearity of their methodology. Both the dependencies in the time dimension and the correlation among multivariates are nonlinear[29], which VAR-based models may fail to capture.

2.2 Time series decomposition

Time series could be viewed as a mixture of several different series because they are influenced by several factors. Therefore, when coping with a time series, decompose the time series into multiple series is commonly applied so that the effects from multiple sub-series can be handled separately.

From the perspective of time seires decomposition, a time series typically includes trends, seasonal variations, cyclical fluctuations, and erratic fluctuations.

- Trends refer to a tendency, within a longer time period, for the sequence to continue to move and change.
- Seasonal fluctuations are regular changes in the level of development of a phenomenon due to changes in the seasons.
- Cyclic fluctuations are cyclic continuous changes that are not strictly regular over a certain time period.
- Erratic fluctuations are the effects of numerous contingent influence along time series

The decomposition model is divided into additive and multiplicative models. Additive [30] refers to the fact that the components of the time series are independent of each other and all four components have the same magnitude. Based on the additive models, weighted additive models [31] which used weight sum of the four components is proposed, and STL based models [32], which is insensitive to outliers, also achieves good results.

More specifically,

$$Y_t = T_t + S_T + C_t + I_t \tag{2.2}$$

where Y denotes the forecasting time series, T illustrates the trend sub series, S illustrates seasonal sub series, C denotes the cyclic sub series and I denotes the irregular fluctuations.

There are many time series forecasting approaches that based on additive models. Facebook prophet model [33] is also an additive model. It has many advantages: flexibility: it can be easily adapted to seasonality with multiple periods and allows analysts to make different assumptions about the trend. Missing treatments: Unlike the ARIMA model, the measurements do not need regular intervals and we do not need to insert missing values Fast: The fit is very fast and allows analysts to explore many of the model's properties interactively. More interpretable: the predictive model has parameters that are easy to interpret.

The multiplicative model output component has the same magnitude as the trend term, the seasonal and cyclic terms are scaled numbers, and the irregular variation term is a sequence of independent random variables, which obeys normal distribution. Similar with the additive model,

$$Y_t = T_t \times S_T \times C_t \times I_t \tag{2.3}$$

Due to the lack of fitting ability of simple additive and multiplicative models, some more hybrid models have emerged. However, these models have difficulty in achieving good prediction results due to the difficulty in separating the trend and cyclic terms.

2.3 Feature engineering and machine learning methods

This class of methods usually changes the organization of the data through time sliding windows and uses machine learning methods such as xgboost, svr, etc. for learning. This type of method is often used in the financial field in the past years, but recently it has become particularly popular in the Kaggle competition. [34] proposed a two-stage neural network structure which constructed with SVM and self-organized features prediction in financial field. [9] proposed an approach which transforms time-related data into an infrastructure that can be handled by several machine learning algorithms. Different types of feature selection methods are then applied to the regression task. Subsequently, several machine-learning-based approaches, such as support vector regression (SVR)[35] and neural networks[36][37][16], have been proposed to capture nonlinearity. In SVR[38], forecasting problems are treated as regression problems, and the kernel method is applied to the models to increase their nonlinear processing capabilities. In[36], neural networks were combined with classical ARIMA models to capture nonlinearity.

The advantages and disadvantages of this feature engineering-based approach are clear. In terms of advantages, since each feature is obtained by carefully analyzing the background of the forecasting target, the features are rigorously filtered and the experimental results obtained are usually better. The disadvantage, on the other hand, is that feature engineering cannot be performed when the background of the time series itself is not well understood, or when the input dimension is too large, it is difficult to analyze and filter each dimensional time series.

2.4 Deep learning based methods

The deep learning based approach is the most popular, accurate, universal and robust approach, and is the one used in this research.

Recently, neural networks with deep architecture, also known as deep neural networks (DNN) have become widely discussed in the research community owing to their ability to capture time information and inner relation of data behaviour. Dedinec et al. used the deep belief network (DBN) in electricity load forecasting, which outperformed neural networks of shallow structures for the first time [39]. Qiu et al. proposed a deep learning approach for load demand forecasting [40]. A deep belief network including two restricted Boltzmann machines (RBMs) was used in their model. Ryu et al. proposed a DNN-based load forecasting model, and the obtained results outperformed the double seasonal Holt–Winters (DSHW) model and ARIMA [41].

Considering research works based on RNN, deep learning-based time series forecasting is then divided into UTS forecasting and MTS forecasting.

For UTS forecasting, Kuan et al. proposed a multi-layered self-normalizing gated recurrent unit (MS-GRU) model [42]. Liu et al. proposed a novel approach of long short-term memory (LSTM) for short-term load forecasting [43]. In collaboration with Chai, they also proposed a combination of deep neural network structures including LSTM and convolutional neural network (CNN) to forecast the start time, end time, and average power of fluctuation of electricity consumption [44]. Marino et al. investigated two variants of the LSTM approach: 1) standard LSTM and 2) LSTM-based sequence-to-sequence (S2S) architecture to map sequences of different lengths for load forecasting [14]. Sari et al. proposed an LSTM model embedded in the feed forward neural network (FFNN) for electricity consumption forecasting. This research implied using two distributed models and training them separately for the purposes of electricity consumption forecasting [6]. In these univariate time series studies, although there is bunch of methods, none of the previous studies captured the long-term time series dependence, thus losing the impact of information prior to multiple time steps.

For solving MTS forecasting problems, most deep-learning-based researches can be classified into two types.

The first type of approaches mainly focus on capturing the time dependencies in the entire time window. The most representative and innovative method is LSTNet[4]. In their research, a novel deep learning model specifically for MTS forecasting problems was proposed.

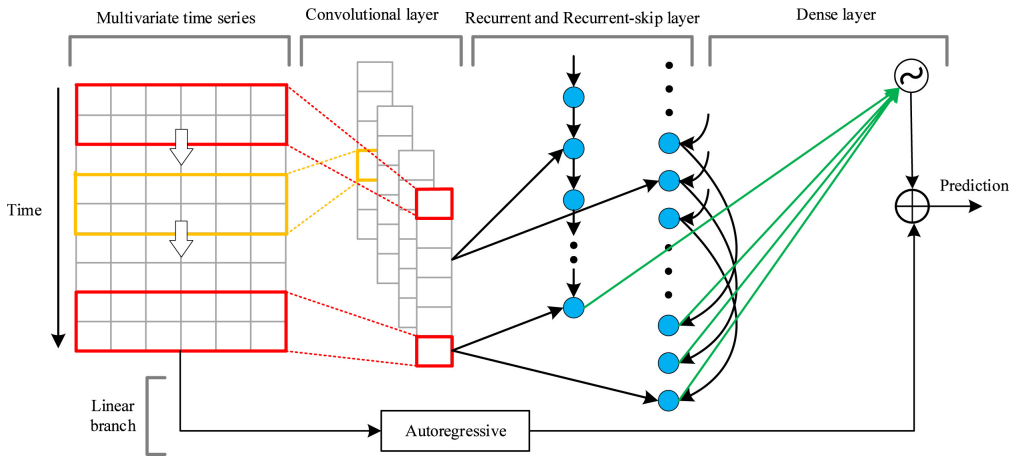

Figure 2.1: The main structure of LSTNet

Convolution neural networks (CNNs)[45] and RNNs[46] have been adopted to capture the short-and long-term dependency patterns among multivariates, respectively. An effective technology named as “recurrent skip” was also proposed

to capture very long-term dependency patterns. Based on evaluation, this method achieved significant performance improvement on four datasets compared to those of several representative baselines. However, one of the major drawbacks of this research is that the length of the "recurrent skip" should be previously defined based on different datasets before training.

The other type of approaches emphasize on capturing the correlation among multivariates at each time step. The most representative approach is the TPA-LSTM[47]. In their research [47], a novel attention mechanism was applied to calculate the attention value for the different variables at each time step. Recently, due to the popularity of graph networks, a number of graph-based multivariate time series forecasting models have emerged. A novel deep learning framework based on transfer entropy graph structure using causal associativity is proposed by TEGNN[48]. Thus far, this approach has reached the state-of-the-art (SOTA) performance comparable with several benchmark algorithms on the same four datasets as used for LSTNet. Nonetheless, both of these researches have a shortcoming compared with our proposed approach: the time-dependency patterns among the multivariates can have different lengths. The time-dependency patterns among the multivariates include both short- and long-term dependencies, and different lengths of dependence often show quite different effects. However, in both LSTNet and TPA-LSTM, only the same-length time dependency patterns of different variables are captured.

2.5 Conclusion

In this chapter, several types of studies that are of high relevance to this dissertation are listed. It starts with traditional time series modeling methods. Then, the time series decomposition methods including additive models and multiplicative models are reviewed. Subsequently, the feature engineering and machine learning methods are reviewed. Finally, this chapter discussed deep learning methods for solving UTS and MTS forecasting, respectively.

Chapter 3

Univariate time series forecasting for electricity forecasting

3.1 Introduction

Electricity is one of the most basic sources of energy in our everyday life and industrial production. As the global economy grows rapidly, electricity consumption around the world is increasing every year. It is most critical for power companies to provide sufficient and stable electricity. For this reason, power companies need to allocate electricity to different consumers according to their production volume. After the liberalization of electricity in 2016, Japan [7], the entire electricity suppliers can be broadly divided into two types: those provided directly by large power companies, and those corresponding to smaller Power Producer and Supplier (also known as PPS). Comparing to large power companies, PPSs provides electricity at lower prices, which saves costs for manufacturers.

Although PPS has the ability to provide power to some manufacturing companies, due to limitations in electrical energy production capacity, PPS can only provide a portion of all energy needs. For the portion that cannot be provided, PPS can only contract with a large electric utility to purchase the default power to fill the portion that is under-provided by power prior to delivering power to the manufacturing company (at least one day). Therefore, in order to provide sufficient power, PPS needs to accurately forecast the manufacturing company's electricity usage in order to purchase power from the large power company and deliver it to the manufacturing company.

In addition to the basic goal of accurate electricity forecasting, this research also worked to solve other problems. On the one hand, when the forecasted electricity consumption is lower than the actual demand of manufacturing companies, PPS will have to buy from large power companies because of insufficient supply and

will be penalized by paying large penalties. On the other hand, if the forecasted electricity consumption is higher than the actual demand, PPS will waste the excess output power. Since the penalties are much more costly than wasting power, PPS prefers power forecasting more than actual demand in order to save costs.

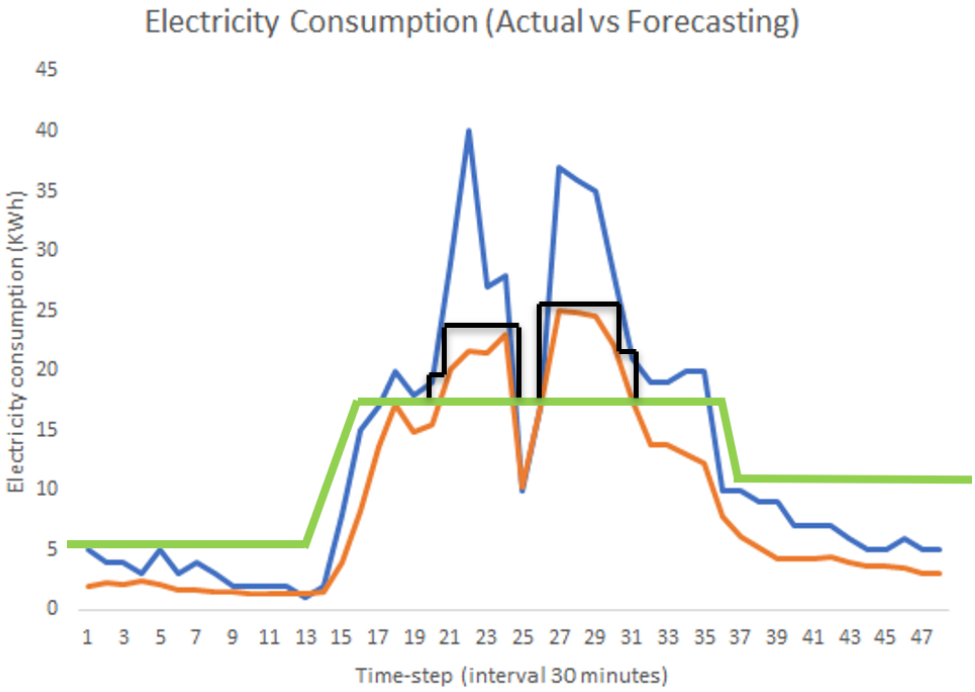

Figure 3.1: An example of the electricity consumption problem

Figure 3.1 shows some details of the difference between actual electricity consumption and forecasting consumption in the study. In this figure, the horizontal axis denotes the time axis in one day, which is separated into 48 halves of hours. The vertical axis denotes the electricity consumption value. The blue line represents the actual electricity consumption of manufacturing companies and the green line represents the basic electricity provided by the PPS. According to this figure, the electricity provided by the PPS is not always sufficient, so additional power needs to be purchased from the large power companies. The orange line represents the forecasted consumption values. The black line is the additional power purchased from the large power company based on the forecasted values. The areas above the black line and below the blue line are underestimates that result in penalties. Conversely, areas above the blue line and below the black line are overestimates that result in wasted electricity. To conclude, this research has mainly 3 targets:

1. The closer the forecasted power consumption is to the actual usage, the better.

2. Reduce underestimation and thus penalty costs.

3. Reduce overestimation and thus wasting electricity.

The three goals here are in decreasing order of importance. Here the subject of this research was provided by a Japanese power company. The research objective is to forecast the electricity consumption of a customer manufacturing factory for one day ahead, with a time interval of half an hour. Since the length of the forecasting is only one day, the study corresponds to a short-term time series forecasting problem.

Since electricity consumption is affected by various internal and external factors, a novel time-series forecasting method that uses past observed electricity consumption data and some possible influencing factors (weather, season, temperature, etc.) to forecast the next day's electricity consumption is proposed. The proposed framework is applied for power forecasting in a real industrial scenario, and the results of comparing the performance of the proposed method with several previous methods show that the proposed method outperforms the previous methods.

3.2 Previous related studies

One of the most related research is proposed by Sari, etc[6]. In this research, they applied the previous days electricity consumption together with other related information, including time, days and season. The model first applied LSTM to pass the continuity along the time axis and used FFNN to capture the impact from the related factors to the electricity consumption[6]. The method shows competitve result comparing with previous traditonal forecasting methods.

However, in their research, there still remains several problems. First, the input of the model concatenate all the electricity conmsumption data with related factors, which denotes a very large models and difficult to trains. Second, the time steps have been add up to $48 \times k$ (Here k denotes the number of days), which is a relatively long time length. Since it is difficult for Long short-term memory (LSTM) to pass the information along such a long time length, it is better to shorten the time steps.

3.3 Long short-term memory and attention mechanism

Long short-term memory (LSTM) is a powerful block structure which was very commonly used in deep learning. It is first proposed by Hochreiter and Schmidhuber which aims to address lost information at longer time lengths in RNN by recursive backpropagation learning. [49]. LSTM is a structurally more complex RNN. In vanilla RNN, the effective information of the time series is recorded within the hidden layer and propagated sequentially through the time step. However, the gradient inside the structure tends to become too small during the backpropagation of the training process, which can easily cause the loss of information before multiple time steps, resulting in the loss of long-range dependence. To solve this problem, LSTM proposes a "gate" mechanism to decide whether the input is important enough to be remembered and whether it can be output, thus selectively control the propagation of gradient information.

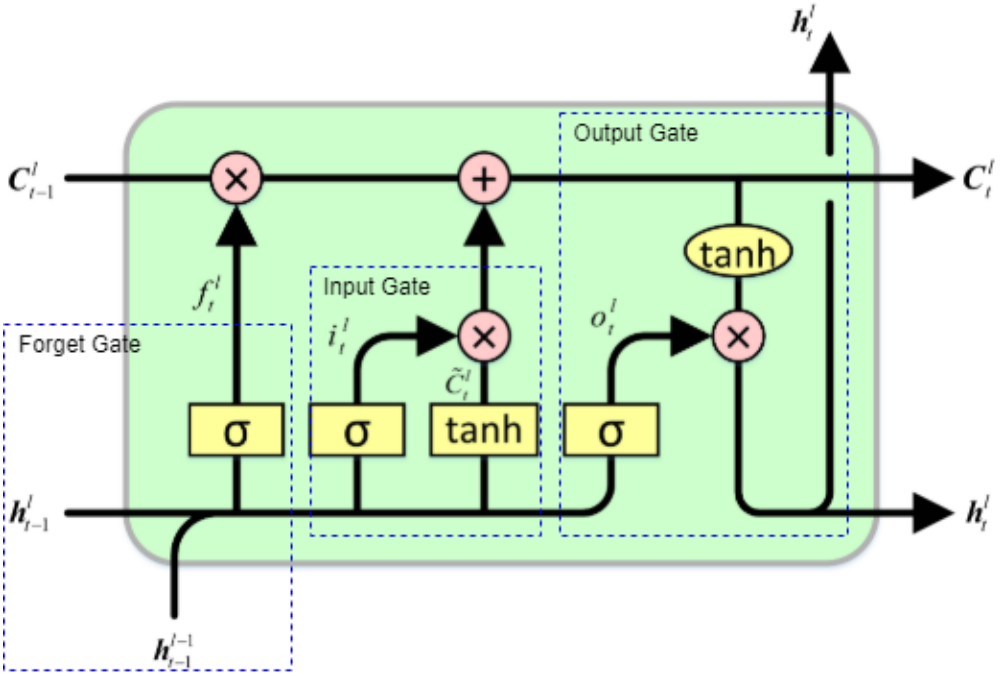

Figure 3.2: The structure of long and short term memory

Specifically according to the figure 3.2, each LSTM block is handled with 3 gates, which are: forget gate, input gate and output gate[49]. Through the forget gate, the LSTM block can selectively forget the unimportant information, then input it through the input gate and participate in the operation, and finally se-

lectively output it to the next LSTM block through the forget gate. On the other hand, when the size of the input series is relatively long, through the training procedure, gradient can easily cause training difficulties during the backpropagation process. However, in the structure of LSTM, gradient can pass directly through the top black line in the graph during the backpropagation process thus without any loss. Therefore, this study uses LSTM blocks as RNN units, and the gradient will pass through the whole backpropagation process smoothly with less loss when information is stored by longer time steps and more memory.

The attention mechanism was first applied to the image field and proposed in the 90's in last century. The essence of the attention mechanism is inspired by the mechanism of visual attention. When we perceive a thing visually, we usually do not look at the same sight from start to end, but tend to observe and pay attention to a specific part according to our needs. And when we find that a scene often appears in a certain part of what we want to observe, we will learn to pay attention to that part when a similar scene appears in the future.

In 2014, google mind team made the attention mechanism start to fire up, and they applied the attention mechanism to an RNN based model for image classification and then obtained very promising performances. Then [50] proposed a new model which applied attention mechanism for simultaneous translation and alignment on a machine translation task. And their research is considered as the first to bring the attention mechanism to the area of NLP. More specifically,

$$S_t = f\left(S_{t-1}, y_{t-1}, c_t\right) \tag{3.1}$$

Here S_t denotes the transfer state at time step t, y_{t-1} denotes the ground truth at time step $t-1$, $f()$ denotes the RNN unit and c_t is computed as:

$$c_t = \sum_{j=1}^{T} a_{tj} h_j \tag{3.2}$$

where h_j is the hidden state output at time step j. Here all h_j from h_1 to h_T has been pre-calculated based on the forward propagation of the encoder. Here a_{tj} is a weight, which is computed as follows:

$$a_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T_x} exp(e_{tk})} \tag{3.3}$$

$$\text{where } e_{tj} = g\left(S_{t-1}, h_j\right) \tag{3.4}$$

Here Tx is the sequence length of the input. a_{tj} is computed similarly to the SoftMax function, which gives the conditional probability of e_{tj}. Here the function g measures the degree of alignment of each state S_{t-1} with the j^{th} input. The

function g used to compute e_{tj} can be a simple dot product or a small multilayer perceptron. Using this function, one obtains c_t as a weighted linear combination of all input vectors for time step t.

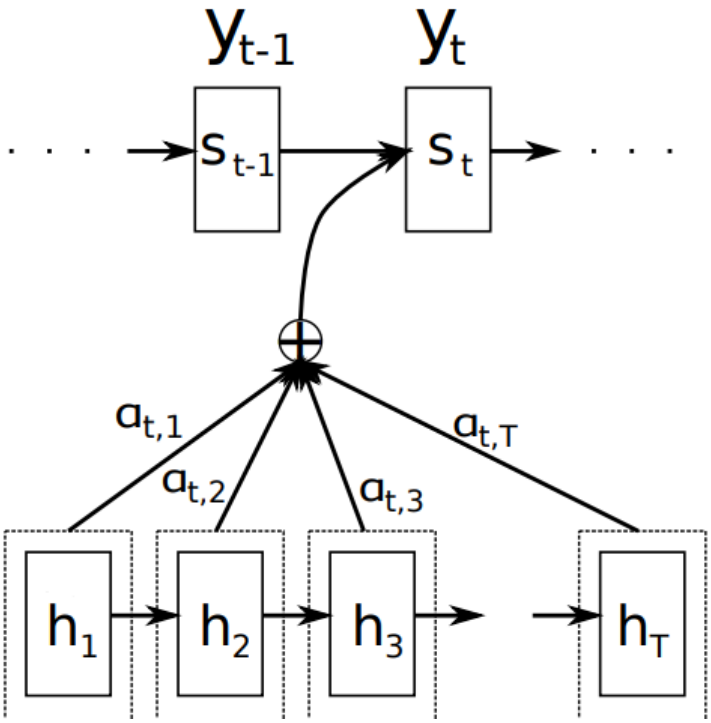

Figure 3.3: The diagram of attentional mechanism

From the perspective of this paper, not every time point in the time series has the same information weight. By applying the attention mechanism, the model can generate the weights of each dimensional feature adaptively based on the implicit state information of the current time step, and use the updated time series with weights as the input to the new encoder. In addition, by using the attention mechanism, the model has the capacity to capture the relationship between each output at any length apart, thus allowing the ability to capture long-distance dependencies to be improved.

3.4 Inside and outside information

The key idea proposed for this research can be summarized as Internal and External Information. Most research works on time series forecasting (especially for electricity consumption forecasting) consider the regularity and periodicity of

the data itself in the time series and try to extract temporal information or factors that tend to influence the forecasting results from the interior of the time series [51]. But it is not only the information in the time series that influences the time series itself, there are also many other factors that affect the time series. The relationship between these factors is very complex and even interacts with each other, which makes it difficult to use these factors to predict.

In this study, I have tried to summarize these studies by dividing them into two types of information: Internal and External Information.

Internal Information Internal information can be artificially generated sequences of the same type as the target sequence (e.g., plant machine temperature sequences, etc.), or it can be a subsequence of the time series we want to forecast. There are a number of approaches to decompose a variable of time series into multiple subseries. Decomposing into multiple subseries allows us to better separate out the useful information hidden in the original time series. In contrast to the external information, the internal information can be influenced by the external information and even interact with other internal information. In contrast, external information can hardly be influenced by internal sequences because of their objective existence. Therefore, separating internal and external information is possible to decrease the computational complexity by eliminating the need to compute the influence of internal information on external information that should not exist.

External Information The external information is usually some objectively existing natural series that may affect the internal information but are not affected by them. In general, external information is generally unaffected by other variables in the system, while can affect the target time series to be predicted. For example, temperature, season, time of day, etc., can be generalized to external information.

3.5 Proposed network structure

In this section, the overall structure of the framework is proposed. In the first section, the general overview of the whole structure is presented. The second section introduces the encoder based on the attention mechanism so as to select the valid input information and avoid the loss of long dependencies. The third section presents the nonlinear corrector by combining external information for correction. Finally, a decoder is presented to fit the forecasting results. The details are presented in the following sections.

3.5.1 Main structure

In the proposed approach, several different types of neural network substructures are combined, which includes an encoder based on an attention mechanism, a nonlinear corrector, and a decoder. An encoder-decoder based structure is applied in this study, where the encoder extracts feature from previous power usage data and the decoder is used to fit the output results. Since the goal is to use previous data to forecast the next day's electricity consumption, I take several days of electricity consumption as input. Each day's data is divided into 48 time steps with a half-hour interval between each two steps. These 48 time steps contain a complex transfer of long-range dependencies. To obtain the input features by applying the attention mechanism, an encoder is used to filter the useful input information, and saving the long-range dependencies at different time steps.

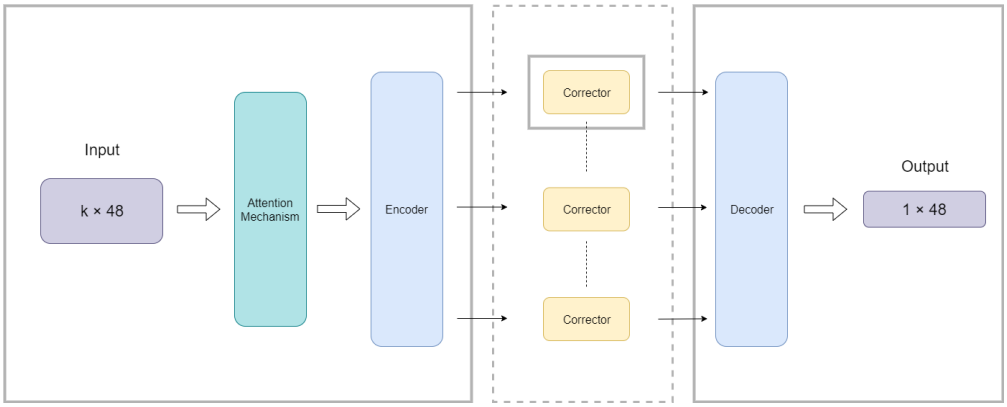

Figure 3.4: The overall structure of the proposed network

In the model structure, the electricity consumption values for k consecutive days prior to the forecast are stitched together in parallel to obtain a $k \times 48$ matrix as an input. This input controls the length of the time steps and preserves the time dependence well during the backpropagation. Using the attention mechanism, we can jump around to calculate the importance of different inputs and calculate the degree of alignment at different time steps. The whole structure of the framework is shown in Figure 5. Its three parts are described next in turn in the following sections.

3.5.2 Attention-based encoder

As shown in the figure 3.5, we used as input the electricity consumption data for k days before the forecasting date. These inputs were combined in parallel into a k $\times$ 48 matrix. At each time step, the degree of alignment between the current

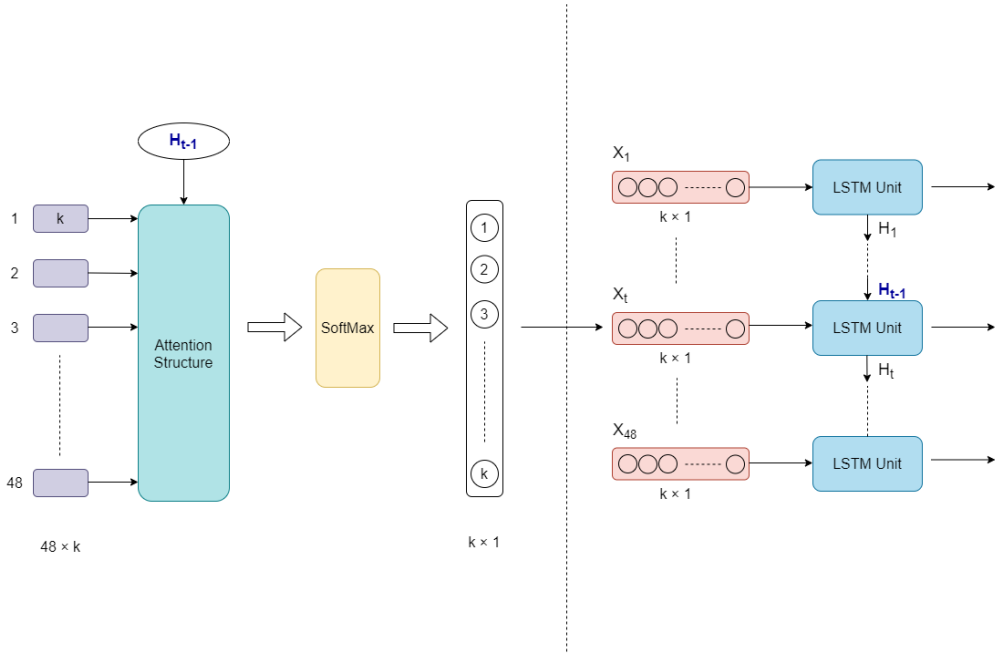

Figure 3.5: The structure of the attention-based encoder

input and the features of the previous hidden layer in the LSTM layer is calculated by means of an attention mechanism. With the attention mechanism, we can then assign different weights to the electricity consumption data for k days at different time steps. The SoftMax function is then used to obtain an encoded vector of size $k \times 1$. This vector is the output of the attention encoder at time step t, which will be transposed to a size of $1 \times k$ as the input of the nonlinear corrector at time step t. At last, we can get 48 matrices of shape $1 \times k$ as the input for the following LSTM blocks.

There are several natural language processing researches [52] [53] which used the attention mechanism in encoders. These research overcome the problems of long time step cases and limited storage of state vectors of fixed size; therefore, we apply attention mechanisms in our studies.

Specifically, the self-attention mechanism, which is commonly used in the field of Natural Language Processing (NLP), is not applied here. In contrast to NLP, the inputs of the proposed framework are not interconnected between different time steps like words. From an NLP perspective, each word is inherently related to other words in a sentence because of its inherent properties (e.g. Subject for Predicate, Object). However, in this study, it is difficult to say that there is an inherent connection between the predicted value at one moment in the same day and the predicted value at another moment. On the contrary, since our goal is to compress

the number of time steps, we are more interested in the connection between the inputs on different days within the same time step. Due to the transferability of time series, we believe that the self-attention mechanism based on position encoding loses the continuity on the time axis, but if the hidden layer in LSTM is used as Query in the attention mechanism, the continuity along time axis can be well preserved.

To summarize the attention-based encoder, since the power consumption in each half hour will have a significant impact on the next half hour, a two-layer LSTM structure is applied to convey this impact. The output of this attention-based encoder is a 48×1 vector, where one of the values is the output of the encoder at each time step. Finally, after several experimental adjustments, we finally set the value of k to 7. At this point, we can obtain the optimal experimental results, which are also in line with the concept of the 7-day cycle of the human week.

3.5.3 Nonlinear corrector

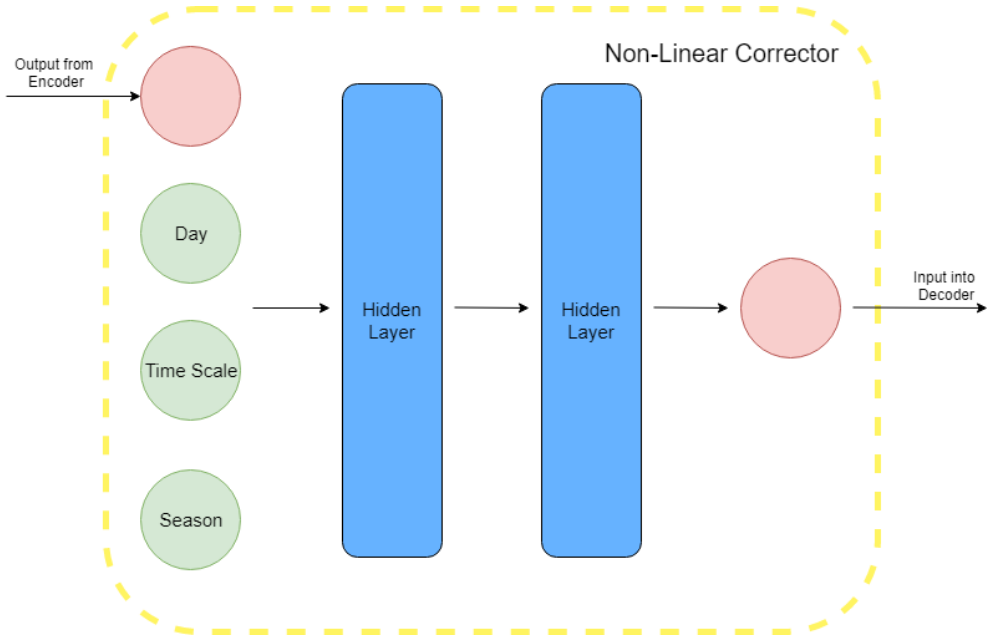

Figure 3.6: The structure of nonlinear corrector

For each output of the encoder in the figure, a nonlinear corrector is used to handle the effect of external information on the input. The input of the corrector at time step t corresponds to the output of the encoder at that point in time. In our study, we argue that after the encoder understands the effect of the internal

information (time series data) propagated through time, we can use the external information to correct the data affected by time. How to deal with the influence of external information on internal information is the key to build a good model. Traditional methods for electricity consumption forecasting, namely ARIMAX [54], NARX-ARMA [55], SARIMAX [56], etc., have tried to use both internal and external information. However, their study only linearly stitched together internal and external information as a larger input [39] [41] [57] or used the two types of information separately in different algorithms and combined the results through integrated learning [58] [59].

However, we believe that the effects of internal and external information on the forecasting series should be predicted separately. First, the influence of external information on internal information is unidirectional; there is no internal information that affects external information in reverse. Second, in some cases, some extreme changes in the external information (e.g., worker holiday changes, sudden temperature drops, etc.) can have a very large effect (sometimes the exact opposite effect) on the internal information. If we treat both internal and external information as being at the same level, we lose the ability of the model to capture the impact between the interiors of these messages. Only when the separate influences of internal and external information are learned separately can the model gain a stronger ability to extract information and thus forecast power consumption more accurately. Therefore, we build such a model that treats the two types of information separately in different layers.

More specifically, each corrector has two hidden layers, as shown in the figure. The correctors are used to adjust the forecasted output of the encoder by the effect of external factors. We added a dropout [60] to reduce the effect of overfitting. We believe that external information can correct the input. Therefore, we combine each data point (output obtained at each time step in the attention-based encoder) with external information on four possible factors: daily information, time-scale information, seasonal information, and holiday information. The output of each nonlinear corrector is one corrected data node. The output of each nonlinear corrector is one corrected data node.

3.5.4 Decoder

For the decoder, the 48 outputs of all correctors are used as inputs. Here, the temporal information of the corrected data nodes may be lost through the correctors, since each corrector corrects the data for each time step separately. For the output of the nonlinear correctors at each time step, we need to repair the time dependence of the previous time steps. Therefore, we add a two-layer LSTM to the decoder to prevent this loss and capture the temporal dependencies between each time step.

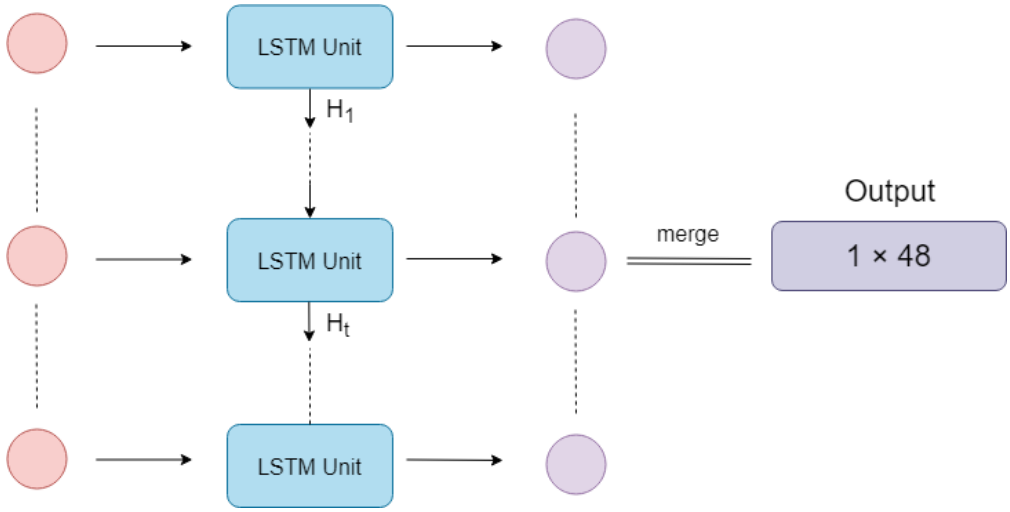

Figure 3.7: The structure of decoder

3.6 Evaluation

3.6.1 Dataset

The experimental dataset was obtained from a small electric utility in Japan. The dataset consists of historical electricity consumption data for one year and four months (from August 2016 to November 2017) with a resolution of 30 minutes. Each day represents 48 time steps of electricity consumption, and each value ranges from 0 to 90. we first rescale the consumption data from [0,90] to [0, 1].

In the proposed approach, the three elements of the external information used as input to the nonlinear corrector are the same as those considered in a previous research work [6]. These three elements consist of discrete data whose values do not have any numerical meaning. Therefore, we binarize the classification input by using a one-hot encoding so that these data can be considered as a vector in Euclidean space. These three elements are related to the following elements provided by the power company:

1. Day information: This information represents what day it is for each time step data in the dataset, ranging from Monday to Sunday.

2. Time-scale information: This information represents the time scale for each time step data in the dataset, ranging from 00:00-23:30.

3. Season information: This information represents the different levels of demand of manufacturing companies. We separate all the levels into three

seasons: the low demand season (November and December), the middle demand season (January – June, September, and October), and the high demand season (July and August). This definition is based on the real situation observed in the power company [6].

4. Holiday information: This information represents whether this day is a holiday or not. Since during holidays, the electricity consumption would decrease to low level, we add this information to improve the forecast accuracy. The detailed holiday information is open source data from the website [61].

3.6.2 Input and output of the model

The purpose of the proposed method is to accurately estimate the next day's electricity consumption. The main inputs are expressed as:

$$x^i = x_1^i, x_2^i, \ldots, x_{48}^i \tag{3.5}$$

Where x_t^i is the actual electricity consumption of the t^{th} half an hour of a day i. The electricity consumption within 48 time steps to be forecasted is expressed as below:

$$\widehat{y} = \widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_{48} \tag{3.6}$$

where $\widehat{y}_t$ is the forecasted consumption for the time step t. The obtained real consumption of the next day is expressed as follows:

$$y = y_1, y_2, \ldots, y_{48} \tag{3.7}$$

where y_t is the ground-truth consumption value obtained on the time step t. The four additional inputs to each data node inside the nonlinear corrector are expressed as follows:

$$D_1, D_2, \ldots, D_7,\ T_1, T_2, \ldots, T_{48},\ S_1, S_2, S_3,\ H_1, H_2 \tag{3.8}$$

where $D_1, D_2, \ldots, D_7$ denote the one-hot code for day information, $T_1, T_2, \ldots, T_{48}$ denote the one-hot code for time-scale information, S_1, S_2, S_3 denote the one-hot code for season information, and H_1, H_2 denote the one-hot code for whether it is a holiday or not.

Here the effect of temperature on electricity forecasting is also considered. However, according to the experimental results, the temperature had almost no effect on the local power forecast (almost all noise). We believe that since the PPS is located in Kitakyushu, industrial power accounts for a large proportion of it. And since industrial electricity does not change dramatically with temperature as

domestic electricity does, we removed the external variable of temperature from the model.

By applying the idea of internal and external information, the parameters in the proposed framework decreased from 3.1×10^5 to 7.4×10^4 comparing to the previous research which concatenate all the related information together. Due to the decrease in the number of parameters and the avoidance of considering the effect of internal factors on external factors, the proposed model is able to converge faster compared to the previous one.

To conduct the evaluation of the performance, we used 2 loss functions:

a. We used the root mean squared error (RMSE) as the first loss function. The reason we did not use MAPE (mean absolute percentage error), which is commonly used in load forecasting, is that a large number of low-value data points close to zero were included in our dataset in our study. On the one hand, for these low values, it makes no sense to use percentages to measure these errors [62]. On the other hand, high values are more important due to the high penalty they cost. Thus, we use RMSE instead of MAPE. The RMSE score is defined as follows:

$$RMSE = \sqrt{\frac{1}{48}\sum_{t=1}^{48}(\hat{y}_t - y_t)^2} \tag{3.9}$$

With respect to the interpretation of the RMSE score, the lower value indicates better performance.

b. We create our own loss function as the second loss function to make our forecasting be more inclined to overestimate instead of underestimate. For the Error part, we consider that: (1) Underestimate should be punished more than overestimate, so we add coefficient m here. (2) Forecasting over maximum PPS value should be punished much more if it is underestimated, so we designed this Error function with these 3 conditions. Our own loss function is defined as follows:

$$Error = \begin{cases} \hat{y}_t - y_t, & \text{when } \hat{y}_t \geq y_t \\ (\hat{y}_t - y_t) \cdot m, & \text{when } \hat{y}_t < \beta < yt \\ \gamma^{\frac{\hat{y}_t - \beta}{\alpha - \beta}} \cdot (\hat{y}_t - y_t) \cdot m, & \text{when } \beta < \hat{y}_t < yt \end{cases} \tag{3.10}$$

$$Regularization = (\hat{y}_t - y_t)^2 \tag{3.11}$$

$$Loss = Error^2 + r \cdot Regularization \tag{3.12}$$

Where $\hat{y}_t$ and y_t are forecasting consumption and real consumption at time step t, respectively. α is the maximum value of PPS supplies. β is a threshold to separat e high and low forecasting. If $\hat{y}_t$ is less than β, we would give lower error since higher value is more important in our task. γ is a predefined base that is larger than 1. If $\hat{y}_t$ is higher than α, which means the PPS does not supply enough, $\gamma^{\frac{\hat{y}_t-\beta}{\alpha-\beta}}$ will become bigger, the error will increase even quicker since we want to punish more for larger forecasting. For the regularization part, we used L2 regularization which is also the MSE (Mean Square Error) to insure the convergence. r here is a flexible coefficient for controlling the importance of regularization. To make the scale to be the same, we square the Error part and add it to Regularization term to be our customized loss function. In our experiment, we set α at 0.25, β at 0.1, γ at 1.5 and m at 2.

3.6.3 Hyper-parameters

To train the whole model, we used mini-batch gradient descent [63] with the momentum of 0.9 and batch size of 4. In this way, stability can be increased to a certain extent; therefore, we can make the model learn faster. Moreover, we can gain the ability to obtain the loss value irrespectively to a local optimum [64].

3.6.4 Experimental result

To evaluate the performance of the proposed method, we construct the training and testing scenarios, which are outlined in Table 3.1, 3.2, 3.3 and 3.4. All the 3 testing scenarios used the same training set in Table 3.1. In the conducted experiments, we calculated the average value of the RMSE score for all days. To discover the different impact extent of all the outside information, we designed 4 kinds of combinations with different outside information as follows:

I. Day and Time-scale information

II. Day, Timescale and Season information

III. Day, Timescale and Holiday information

IV. Day, Timescale, Season and Holiday information

We compared our methods against the traditional LSTM and LSTM-FFNN [6], as well as MA (Moving Average), ARIMA [54] and SARIMA models with the moving step of 7 days; the load scale ranges from 0 to 90 in each scenarios.

For ARIMA and SARIMA, we used the trend order of (48, 1, 1). We set the seasonal order of (1, 1, 1, 7) for SARIMA as we tried to find the regularity of

Table 3.1: The Training Scenarios

Sample	**Training Data**	
	Input	**Output**
1	2016/8/1 ∼ 2016/8/7	2016/8/8
2	2016/8/2 ∼ 2016/8/8	2016/8/9
3	2016/8/3 ∼ 2016/8/9	2016/8/10
...	...	...
390	2017/8/25 ∼ 2017/8/31	2017/9/1

Table 3.2: The First Testing Scenario

Sample	**Testing Data**	
	Input	**Output**
1	2017/8/26 ∼ 2017/9/1	2017/9/2
2	2017/8/27 ∼ 2017/9/2	2017/9/3
3	2017/8/28 ∼ 2017/9/3	2017/9/4
...	...	...
30	2017/9/24 ∼ 2017/9/30	2017/10/1

weekly information. As shown in Table 3.6 and 3.7, the proposed method (I, II, III and IV) demonstrated the best results in comparison to all others for each scenario, respectively. Furthermore, holiday information has a stronger impact compared with season information according to the experiment result. According to our understanding, the previous days' electricity consumption already contains some season information and is input previously in the encoder part. Therefore, the season information in the correctors would not have a very strong impact to the result. In contrast, holiday information cannot be obtained from the previous days' electricity consumption (i.e. It is impossible to know whether the next day is Japanese holiday according to the previous days' electricity consumption) and the outside information of holiday input in the correctors helps to correct the result.

As shown in the table 3.5, the training time (with the unit seconds) of both previous methods and proposed method are listed. By applying the idea of internal and external factors, the training time of the proposed framework was reduced by an average of 51.7% compared to the previous method using SGD with momentum as the optimizer.

As shown in the figure 3.8, our forecast is well fitted to the inflection point of the consumption curve. However, at some parts of the forecast, the red points (actual value) still value higher than blue points (forecasting value), which means the forecasting is underestimated. Thus, in the next experiment, we used our own loss function to prevent this. Since our customized loss function is a little difficult

Table 3.3: The Second Testing Scenario

Sample	**Testing Data**	
	Input	**Output**
1	2017/9/25 ∼ 2017/10/1	2017/10/2
2	2017/9/26 ∼ 2017/10/2	2017/10/3
3	2017/9/27 ∼ 2017/10/3	2017/10/4
...	...	...
30	2017/10/24 ∼ 2017/10/30	2017/10/31

Table 3.4: The Third Testing Scenario

Sample	**Testing Data**	
	Input	**Output**
1	2017/10/25 ∼ 2017/10/31	2017/11/1
2	2017/10/26 ∼ 2017/11/1	2017/11/2
3	2017/10/27 ∼ 2017/11/2	2017/11/3
...	...	...
30	2017/11/23 ∼ 2017/11/29	2017/11/30

to converge, we used 2 training steps with different learning rates and different flexible coefficients r.

For the first training step, we choose a high r value and learning rate to make the model quickly converge and pay more attention to the MSE score, since we need the loss function to converge towards the direction that getting close to the shape of actual time series first. After the model is about to converge, we choose a low r value and low learning rate to let the model carefully find its way to simulate the harsher Error part. And finally, we get the following result.

As shown in Table 3.8 and 3.9, our proposed method demonstrated the best results in comparison of Customized Loss values to all others. All the Customized Loss calculated in experiments in Table 3.8 and 3.9 used the r at 0.5, α at 0.25, β at 0.1, γ at 1.5 and m at 2.

In figure 3.8 we showed 2 days electricity consumption forecasting value which forecasted by the model trained with RMSE as the loss function and showed Customized Loss value to evaluate the result at the top of each graph. Relevantly, in figure 3.9, we showed 2 days electricity consumption forecasting value which forecasted by the model trained and evaluated with Customized Loss function. Here, both of the 2 models used the same training and testing data. We chose 2 corresponding forecasting result of 2 models respectively and used Customized Loss value to compare the result.

According to 3.8 and 3.9, the Customized Loss values in Figure 3.9, are smaller

Table 3.5: The comparison of training time for previous and proposed methods

	I	II	III	IV
Previous	3764	3821	3809	4016
Proposed	1849	1861	1856	1882
Decreased	50.9%	51.3%	51.3%	53.1%

Table 3.6: The comparison of RMSE for other methods

	LSTM	**MA**	**ARIMA**	**SARIMA**	**LSTM-FFNN**
Scenario 1	8.4729	13.415	9.035	8.361	5.365
Scenario 2	6.5476	9.044	7.044	6.258	4.1132
Scenario 3	7.0781	7.962	7.913	6.79	4.3206

than the ones in Figure 3.8 , which means our Customized Loss function helps the model to give overestimation rather than underestimation. Sometimes the forecasted values trained with Customized Loss function are smaller than the forecasted values trained with RMSE.

These values are lower than β in the Customized Loss function (here β is a threshold to separate high and low forecasting), which are not as important as the high values in this research. In fact, the electricity lower than β are the basic electricity supplied by PPS, which PPS do not need to buy from a large power company. In our research, we pay more attention to the values higher than β.

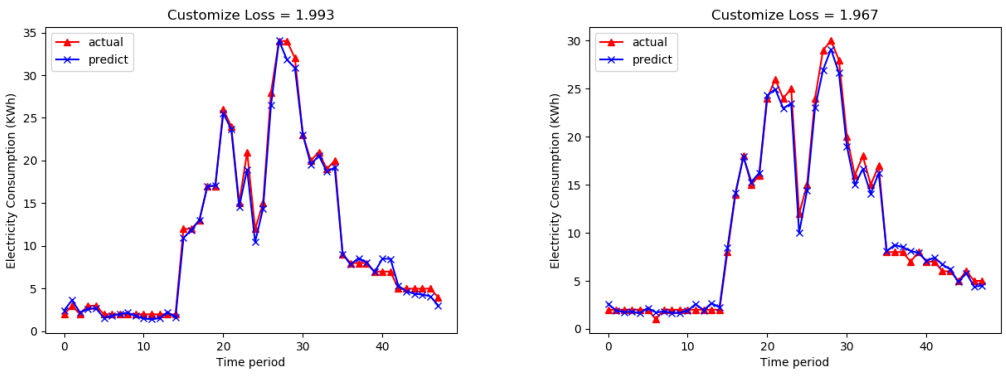

Figure 3.8: Two examples of forecasting using proposed method with RMSE

Table 3.7: The comparison of RMSE for the proposed method

	I	II	III	IV
Scenario 1	3.5817	1.7591	1.2104	1.0026
Scenario 2	2.7964	1.3784	1.0442	0.7881
Scenario 3	3.0211	1.4329	1.1164	0.787

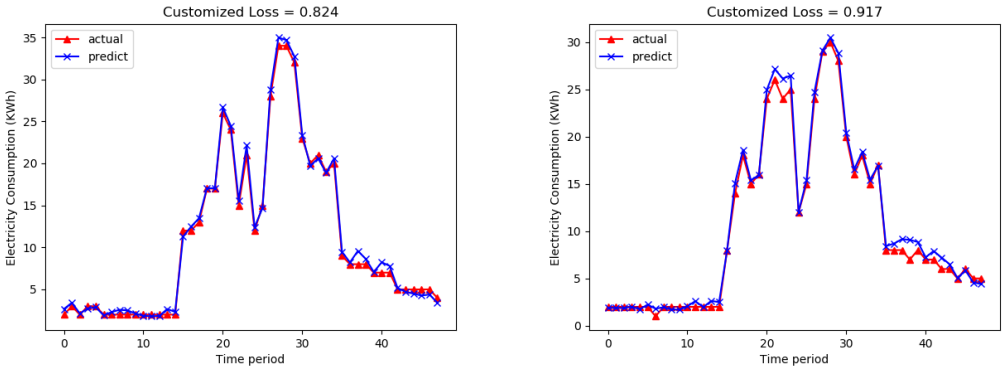

Figure 3.9: Two examples of forecasting using proposed method with customized loss function

3.7 Conclusion

The aim of this research was to forecast the electricity consumption of one day based on the electricity consumption observed for the previous days. However, electricity consumption is affected by several factors that are difficult to predict. And this leads to the difficulty for the power company to keep the balance between keeping a balance between the power demand and supply for customers. To solve this problem, the new method was proposed based on attention mechanism encoder-decoder and several nonlinear correctors.

During this research, we first considered the idea of internal and external information to analyze the previous research works, and thereafter, we proposed a method to combine the time series information with outside related causes. By using the internal and external factors, the number of parameters is decreased and the training time is decreased by an average of 51.7%. We also embedded the attention mechanism jointly with the encoder-decoder structure inside the model to solve the problem of long-distance dependence loss. A new structure of separating the internal and external information in the encoder and nonlinear correctors to prevent the loss of combining them together has been proposed. The proposed method allows achieving better results than such alternative methods as LSTM, LSTM-FFNN, and ARIMA. It also allows saving more paid penalties by providing

Table 3.8: The comparison of Customized Loss for other methods

	LSTM	**MA**	**ARIMA**	**SARIMA**	**LSTM-FFNN**
Scenario 1	20.5979	44.1742	25.3279	21.3076	14.7130
Scenario 2	12.5741	24.8174	17.6934	14.8633	7.5889
Scenario 3	9.4951	20.0711	19.8986	16.1268	5.8356

Table 3.9: The comparison of Customized Loss for proposed methods

	I	**II**	**III**	**IV**
Scenario 1	7.2992	3.6023	2.3765	1.9046
Scenario 2	5.7310	2.9376	2.4431	1.5898
Scenario 3	5.6187	2.7056	1.9647	1.2953

more accurate power consumption with overestimate by using the customized loss function.

However, there still remains much room for improvement in this study. First, for multidimensional time series inputs, it is sometimes difficult to distinguish whether they are internal or external information. Second, for practical problems, it is more common to go for forecasting the values of multiple time series instead of one. Finally, the model's capture of time dependence is completely limited to a single point of transmission of the recurrent neural network and cannot distinguish between the different effects of short term dependencies and long term dependencies of the time series.

Therefore, in the next chapter, a special model for time series of factories with multiple sequence inputs and multiple sequence outputs will be presented. This model does not need to distinguish the types of input sequences, obtains more accurate multiple forecasted sequences with limited input sequences, and is able to capture the time dependence of different distances to some extent.

The works in this chapter has been published in:

J2 Song W, Chandramitasari W, Weng W, et al. Short-Term Electricity Consumption Forecasting Based on the Attentive Encoder-Decoder Model[J]. IEEJ Transactions on Electronics, Information and Systems, 2020, 140 (7): 846-855. https://doi.org/10.1541/ieejeiss.140.846

Chapter 4

Multivariate time series prediction for sensor data in the process industry (MLDNet)

4.1 Introduction

With the development of Industry 4.0, data has become one of the very important resources in the industry. With the advancement of data analysis technology, factories can quickly extract complex and useful information from large amounts of data, thus bringing value to the factory. And in process industries, sensors, as common big data collection devices, measure information about processed products and transmit the measurement results through electrical signals. By accurately predicting sensor data, we can anticipate changes in processed products in advance which helps us detect, diagnose and even predict possible failures to reduce unnecessary losses.

In our research, we aim to predict the values of multiple sensors simultaneously. Since the sensor data are time series data, we could regard this target as a special multivariate time series forecasting problem. In this field, researchers [65][5] are often interested in predicting possible future changes and trends based on historical time series data from sensors. However, the problem of sensor data prediction is complex and has been studied for several years.

One of the main challenges in analyzing such serial data formed by multiple time series is to determine the complex dependencies among multiple sensors in the time dimension. More specifically,

1. The relationship between time series of sensor data is unknown.

2. The prediction of sensor data is influenced not only by its historical value

but also by the correlated sensor data.

3. Different time lengths of perturbations in the sensor sequence may represent completely different meanings and variations.

In this research, we aspire to predict the possible values from different sensors in 3 main steps as follows.

1. prepocessing. For the values from the sensors, one of the drawbacks of the current research is the noise and delay in data [2], which most of the sensors are filled with, and may reduce the accuracy of the prediction. Therefore, we consider that the value of the sensors should be preprocesed. However, most of the research such as [66] [67] have the detailed information about the data which can easily define the threshold and SNR (signal to noise ratio) of the signal, and since the data from the process industries is not such kind of common, we try to find a way to de-noise signal without this information. We tried several signal de-noising methods such as SVD [68], FFT [3], Wavelet Transform [69] [70], etc. And we finally chose wavelet transformation for denoising since it reaches the best result and cross-correlation for removing the delay among sensors.

2. Extracting correlations among sensors. In practice, big data applications based on sensor data are popular. [71] proposed a contextual predicting technique for streaming sensor networks, where they focus more on the relationship between contexts in the behavior of individual sensors; [72] used extracting the stability state of each sensor and try to detect the stability of a sensor using a cluster approach. Since most of the current studies point out that the correlation among different sensors may be very helpful. Therefore, we should spend more effort on the correlations among sensors.

3. Modeling. From our perspective, this target is a kind of MTS forecasting problem. Most of the effects on one sensor caused by other sensors can be broadly classified into long-term and short-term effects, which are usually expressed as "trends" and "fluctuations", respectively. The different lengths of dependencies among sensors, i.e., their "perturbations," represent completely different meanings within a different length. As an example, in a chemical processing plant, a 2-minute change in sensor values and a 5-minute change in sensor values may still represent completely different chemical reactions, even when the range of value changes is the same. Furthermore, if we can capture the dependence of the sensor at different time lengths, we can distinguish whether this is a normal chemical reaction or a possible accident, leading to accurate prediction.

In this research, a new deep learning framework is proposed for the sensor data prediction problem, named the Mixed Length of time Dependencies network (MLDNet). It consists of two flows: the main flow and an AR highway. An AR highway is commonly used in deep-learning-based models, such as [4] and [47], for simulating the prediction scale. The main flow adopts the proposed structure called Mixed Length Dilation Block for capturing the mixed length of time dependencies. Finally, the output values of both the flows are added to generate the prediction result. The major contributions of this research can be outlined as follows:

1. We proposed a customized block structure and a new framework for sensor data prediction that can capture the mixed length of time dependencies.

2. We conducted spacious experiments on several datasets to compare our proposed framework with other baselines, and the experimental results indicate that our framework performs competitively on all the datasets.

The remainder of this reseach is organized as follows. In Section 2, we introduce the preparation and pre-processing of datasets. Subsequently in Section 3, the problem formulation and detailed structure of our proposed framework is described. Following this, the evaluation of the proposed structure is presented in Section 4. Finally, the conclusions are discussed in Section 5.

4.2 Previous related studies

For the previous research, one of the most important related research is LSTNet. In their proposed research, they first used a convolutional neural network to capture the relationship between multiple time series over a short-term time span. Secondly, they used recurrent neural networks to capture the connections between multiple time series over longer time horizons. Since this model could not capture ultra-long time dependencies, so they proposed "recurrent skip" to jump between RNNs to achieve better results within shorter time steps. They also proposed the very important AR highway to fit the final prediction scale, which solved the important problem of bias in deep learning on time series prediction.

However, their research still faces some problems in industrialized data. Industrialized data are very complex in terms of dependency lengths, and different lengths of time dependencies represent completely different meanings (e.g., a chemical reaction of 6 seconds may be completely different from a chemical reaction of 3 seconds). LSTNet [4] confuses these different length dependencies and cannot distinguish these different length dependencies well. Therefore, we need a model that can capture mixed dependencies of different lengths.

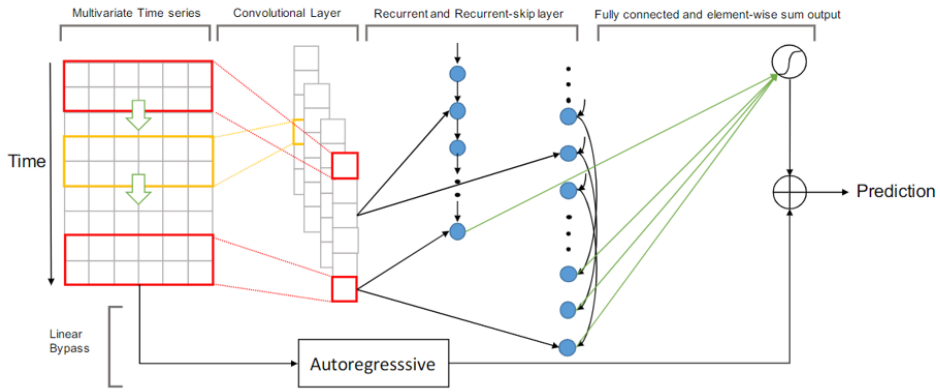

Figure 4.1: The structure of LSTNet

4.3 Preparation and pre-processing of sensor data

4.3.1 Remove noise from sensor data

In the processing factory, the signals from sensors remain some noise. Since all these signals are recorded from the machines and sensors, noise is filled with the signals and could affect the procedure of data analysis. For abnormal value detecting, the vibration amplitude could be completely different because of the noise. Therefore, we need to find a way to remove the noise from the signal.

The data is extracted from different sensors in a chemical process factory, most of the noise is white noise [73]. Moreover, we have no idea about the SNR (Signal to noise ratio) [74], so we could only choose a robust but high accuracy method to remove these incidental noises.

Since the time-frequency domain has a good localization characteristics, wavelet analysis has been widely used in many fields. Therefore, the white noise signal is suppressed by the characteristics of the wavelet decomposition coefficient, in which the weak correlation signal contained in the sequence is also collated and provided with more suitable processing data.

The following wavelet analysis method basically eliminates the potential white noise in the sequence and extracts practical information in the plant signal concisely and effectively.

Wavelet de-noising can be generalized mainly into three methods: (1) Mallat's [75] modulus maxima de-noising based on wavelet coefficients (2) Liu's [76] Beamforming correlation wavelet de-noising (3) Donoho [77] and Johnstone proposed the wavelet threshold de-noising method. Since the wavelet threshold method is

the development of the other two methods which seem to be more effective in our research, we finally chose soft-threshold wavelet de-noising method.

The basic idea of wavelet threshold de-noising proposed by Donoho is: After processing the signal through the wavelet transform (using Mallat algorithm), the signal generated by the wavelet coefficients contain important information on the signal, which means the signal's wavelet coefficient after wavelet decomposition is large. On the other hand, the wavelet coefficient of noise is comprised smaller. And the wavelet coefficient of the noise is smaller than the wavelet coefficients of the signal. For selecting a suitable threshold, only coefficients which are larger than the threshold are considered to be generated and should be preserved. The coefficients which are smaller than threshold values are considered as noise generation and would be set to zero to achieve the purpose of de-noising. The basic steps are (1) Decomposition: select a signal with N layers of the wavelet decomposition to decompose; (2) Threshold processing: After decomposing, the coefficients of each layer are quantified by choosing a suitable threshold value. (3) Reconstruction: Reconstruct the signal with the processed coefficients.

The basic problems of wavelet threshold de-noising include three aspects: the selection of wavelet bases, the selection of thresholds and the choice of threshold function. For the wavelet bases, we chose Daubechies wavelet [78] as db8 with vanishing moments order of 4.

For the threshold, we simply apply soft thresholding using the universal threshold [79]. The universal threshold is defined as

$$\lambda = \sigma\sqrt{2\ln(\mathrm{N})} \tag{4.1}$$

Where σ is a robust estimator of the standard deviation of the finest level detail coefficients, and N is the length of one signal. Here, we use the standardized median absolute deviation (MAD) [80] for this variable.

$$\sigma = MAD(\beta_{J-1}) \tag{4.2}$$

For a univariate dataset $(X_1, X_2, \ldots, X_n)$ the MAD is defined as the median of the absolute deviations from the data's median:

$$MAD = median(|X_i - median(X)|) \tag{4.3}$$

4.3.2 Recognize delay from signals

The signals of different sensors contain some delay. For different machines, signals are transmitted through different media, which may take different time to transfer. And for each pair of the signals, maybe a potential order exists between

them, which means a delay may occur according to a different time of transmissions. For this delay, it may occur different kinds of problems such as incorrect predictions and overfitting, which may not be proper in the following steps.

Since we do not know the range of the delay time of each signal, we need a direct way to calculate the delay and remove it. In this part, we used cross-correlation function to remove the delay part.

In signal processing, the value of cross-correlation shows the how similar the two signals are as the extent of relevance from the first signal relative to the other. However, the cross-correlation is only used for related signals. Finally, we tried to calculate the cross-correlation and then check whether it is related. If it is not related, the cross-correlation would not be considered.

For the time delay between the two signals is showed by the argument of when maximum (if the two signals are positively correlated, or minimum if the 2 signals are negatively correlated) is obtained, or the arg-max (or arg-min) of the cross-correlation, as in

$$\partial_{delay} = argmax((f * g)(t)) \tag{4.4}$$

Here f and g are the two signals and $*$ is the cross correlation operator.

However, the cross-correlation function is mainly used for 2 correlated function. According to our data, most of the signals (which means different signals) don't have a strong relationship with others. So here, we first calculated the cross-correlations of all the pairs, and then find the related ones to proceed the following steps.

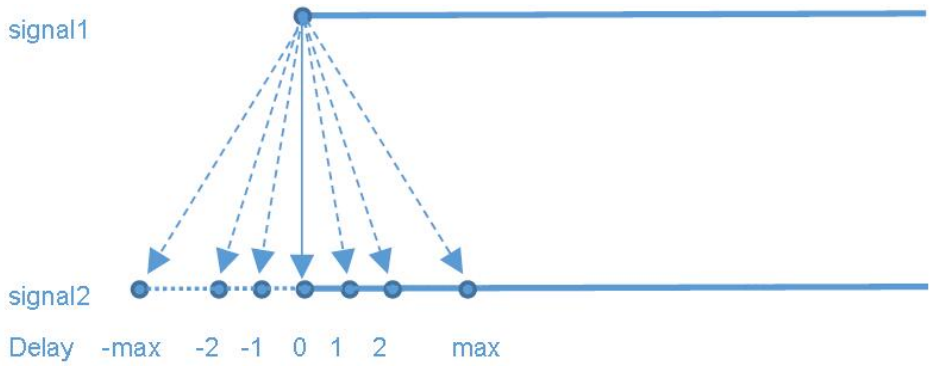

Figure 4.2: The structure of mixed length dilation block

Since the delay could be very large, so when we use the cross-correlation, some value would not take part into the calculation (which usually occurs when the signal has a wide fluctuation), and that will cause a deviation. Therefore, we tried several ways to normalize the data and finally, we chose the value which is

subtracted by the mean of the whole column (which is decided according to the shape of most of the data). In this way, the deviation will decrease to the lowest level.

Per this method, most of the delays among related sensors could be removed. Only if the delay is longer than considered will be missed according to the method (however the range of delay is artificially controlled and the case which delay is larger than considered is almost impossible).

4.3.3 Choosing correlated sensors

In this research, correlation coefficient is applied to choose sensors with potential linkage. Since we query both the linear and nonlinear correlation between sensors, the Pearson product-moment correlation coefficient [81] and Spearman rank correlation [82] is finally chosed as our correlation matrix

The Pearson correlation evaluates the linearity of the relationship between two continuous variables. Two variables have a neutral linear correlation when a variation in one variable is associated with a proportionate change in the other variable. We can use the Pearson correlation to estimate whether an increase in temperature in a plant facility is associated with a decrease in the thickness of the chemical feedstock.

Suppose two random variables are X and Y, and the amount of their elements is N. We assume that $i^{th}, i \in [1, N]$ values taken by the two stochastic variables are represented by X_i and Y_i. The Pearson correlation coefficient is calculated between the stochastic variables X and Y in the following way:

$$r = \frac{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sqrt{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2}\sqrt{\sum_{i=1}^{n} \left(Y_i - \bar{Y}\right)^2}} \tag{4.5}$$

Where $\bar{X}$ and $\bar{Y}$ are the average value of variable X and Y, respectively.

In statistics, the Spearman rank correlation coefficient (with the Greek letter ρ) is used to evaluate the rank correlation i.e., the correlation of order, between two variables X, Y. The Spearman correlation coefficient can be calculated if the two random variables do not take exactly the same values. The ρ between two variables can reach +1 or -1 when the two variables have the same tendency to change.

Suppose two stochastic variables are X and Y, and the amount of their elements is N. The $i^{th}, i \in [1, N]$ values taken by the two stochastic variables are denoted by X_i and Y_i, respectively. By sorting X and Y (either in increasing or decreasing order at the same time), we can obtain the ranking x and y of the elements of two groups. The variables in x and y are subtracted one by one to obtain an equal rank difference set S, where $S_i = x_i - y_i, i \in [1, N]$. The Spearman rank correlation

coefficient of the stochastic variables X and Y is then generated by S, which is shown below.

$$\rho = 1 - \frac{6\sum_{i=1}^{N} S_i^2}{N(N^2-1)} \tag{4.6}$$

By calculating the Pearson and Spearman correlation coefficients between two sensors, we can select the most relevant few sensor sequences for prediction from all the sensor time series.

4.4 Proposed network structure

4.4.1 Problem formulation

In our research, the problem we address belongs to the field of MTS forecasting in sensors. More specifically, given a certain number of time series of sensor data, $X = \{X_{T-w}, X_{T-w+1}, X_{T-w+2}, \ldots, X_{T-1}\}$, where $X_i \in R^m$ and m is the number of the sensors, we aim to forecast $X_{T-1+\Delta}$, where Δ indicates the required horizon ahead of the current timestamp. Here, X_i denotes the observed sensor data at time i, w is the time window, and $T-1$ denotes the current timestamp. In this task, only the time series from $T-w$ to $T-1$ is considered because all the values before time stamp $T-w$ are supposed to have no useful information for predicting the sensors in the future. In addition, both w and Δ are fixed values, which are previously chosen and customized for different tasks. Here the problem formalation is the same as the one in the previous chapter.

4.4.2 Whole structure

According to Figure 4.3, the first part of our proposed framework is constructed with a pure convolutional layer without pooling. Owing to the power of convolutions, we employed a CNN to capture the local dependencies among all the sensor data. Specifically, the convolution layer consists of k filters $C_1, C_2, \ldots, C_k \in R^{m*h}$ where the width of the CNN kernel is the same as dimension m of input time series. h denotes the height of CNN kernel. Particularly, when the width of the kernel equals 1, the convolution captures the linear combination patterns among the different sensors. In this layer, the i^{th} convolution sweeps over the whole input matrix, X, with a stride of 1 and generates,

$$D_i = Activation\left(C_i * X + b_i\right) \tag{4.7}$$

where $*$ implys the operation of convolution, b_i denotes the bias, and $D_i \in R^{w-h+1}$ denotes the i^{th} convolutional result. We used Leaky RELU [83] as our

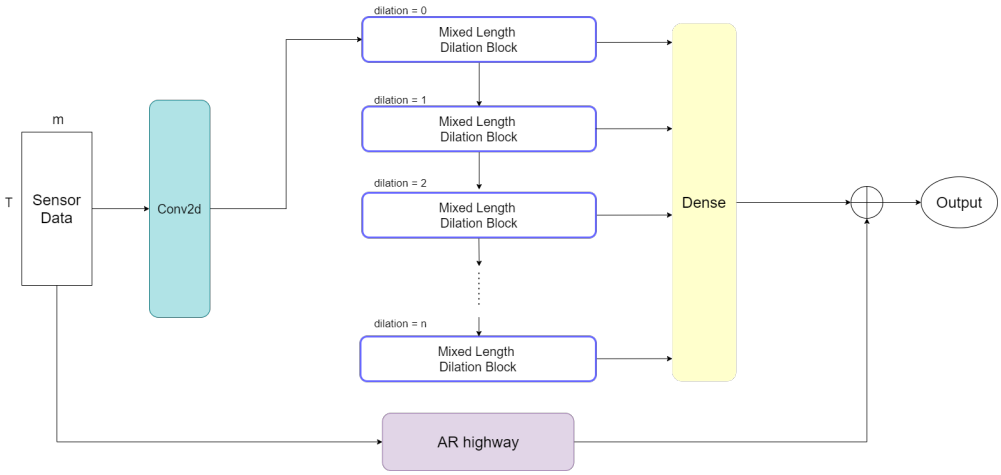

Figure 4.3: The whole structure of the proposed framework

activation function, and the output matrix of this layer is sized with $(w - h + 1) \times k$.

With different convolution kernels, different patterns of time series combinations are generated, and the useful ones are captured by training. For each pattern, which is named as one observation, D_i is the i^{th} observation sequence and $dlen$ is the length of each observation sequence.

The second part of the framework is constructed with several layers, in which each layer is combined with a mixed length dilation block. In this part, we aim to capture the dependencies of sensors within different time lengths. More specifically, we adopted different dilation degrees in different layers, which higher layer captures a longer term of time dependency.

Through these mixed length dilation blocks, dependencies of different time lengths are connected with a dense layer, which automatically learns the importance of different dependencies. More specifically,

$$HidOut = concat\left(hidden_0, hidden_1, \ldots, hidden_{L-1}\right) \tag{4.8}$$

$$DenOut = W_d * HidOut + b_d \tag{4.9}$$

Where $HidOut \in R^{L \times hidr}$ concatenates all the hidden outputs from the dilation blocks, $hidr$ is the number of features output from a dilation block, W_d is the weight matrices of the dense layers, and b_d is the biases. L is the number of mixed length dilation blocks.

Finally, the output of the dense layer DenOut is added along with the output of the Autoregressive highway, which is integrated as the final prediction of $X_{T-1+\Delta}$ as follows:

$$\hat{X}_{T-1+\Delta} = DenOut + ArOut \tag{4.10}$$

where $\hat{X}_{T-1+\Delta}$ denotes the forecasting value of $X_{T-1+\Delta}$. Here, detailed analysis and explanation of mixed length dilation block and Autoregressive highway would be introduced in the following section.

4.4.3 Mixed length convolutional filters

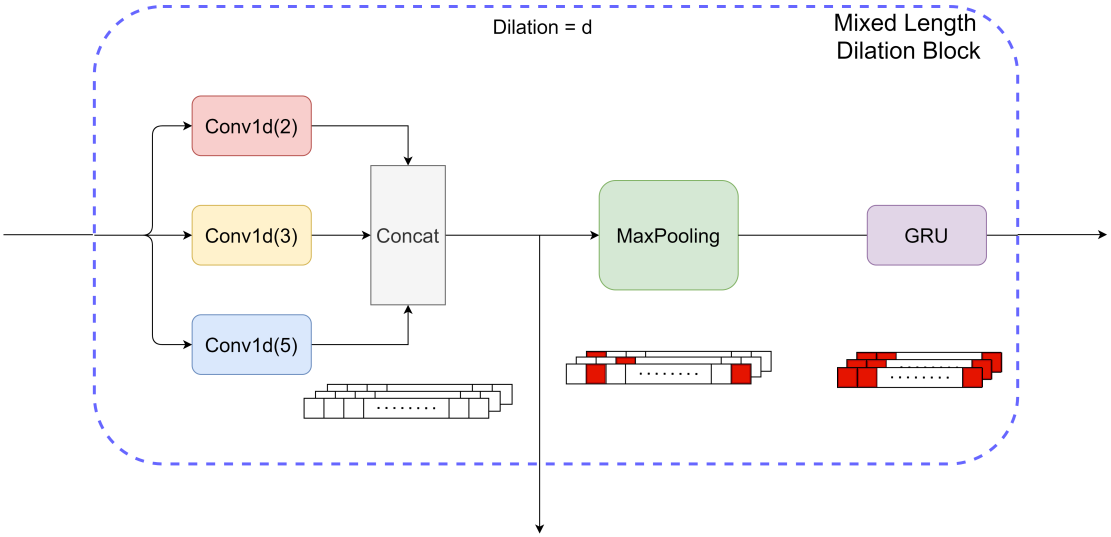

Figure 4.4: The structure of mixed length dilation block

In this section, we introduce a custom stackable block structure, which is also the most central part of this study.

According to Figure 4.4, the mixed length dilation block has one input and two outputs. The input is first connected with several 1-dimensional convolution kernels in different scales, which aim to capture the dependencies within the different lengths. For each dilated convolution kernel, the receptive field is the entire observation sequence (which is a one-dimensional (1-D) sequence).

Specifically,

$$G_{l,i} = Activation\left(dConv_{l,i} * Input_l + b_{l,i}\right) \tag{4.11}$$

Where $l \in [0, L)$ denotes the l^{th} dilation block at l^{th} layer. $dConv_{l,i}$ and $G_{l,i}$ denotes the i^{th} dilated convolution filter and convolution result at l^{th} dilation block, respectively. For each convolution filter at l^{th} layer, we choose the dilation factor

as the value of l. Particularly, when l equals 1, the dilated convolution becomes the normal convolution. When l equals 0, the dilated convolution degenerates to a direct connection.

Therefore, by each dilation block in our framework, the dilation factor becomes larger than the ones in the previous blocks. In each block, we choose several different sizes of convolutional filters and concatenate the result together, which help to capture the dependencies in a variety of different time length.

$$G_l = concat\left(G_l^2, G_l^3, G_l^5\right) \tag{4.12}$$

$$Input_{l+1} = G_l \tag{4.13}$$

Where G_l^i denotes the dilation convolution result with the filter size of i at l^{th} layer. Here we choose the filter size at 2,3 and 5 since these filters are small and can be stacked to capture a variety of length dependencies. With the help of dilated convolutions, all the input sequences are convoluted layer-wise, and a higher layer captures a longer term of time dependency.

After this step, g_l would be pop out as the input of the next mixed length dilation block at the next layer. A Max-pooling layer is then connected after the concatenate, which helps to filter and retain relatively important dependency information at different time steps.

$$R_l = MaxPooling\left(G_l\right) \tag{4.14}$$

Particularly, when l equals 0, the pooling layer would degenerate to a rear sequence slice, which remains the tail of the whole sequence. Finally, we applied Gated Recurrent Units (GRUs) [46] to capture the impact from the dependencies at previous time steps to those at subsequent time steps and reserve the impact from all previous time steps.

$$hidden_l = GRU(R_l) \tag{4.15}$$

Where $GRU()$ denotes Gated Recurrent Units and $hidden_l$ denotes the output of GRU at the final time step at l^{th} layer. At the end of block l, $hidden_l$ would be output to the following dense layer.

4.4.4 Autoregressive highway

Since previous research points out that recurrent-based models are insensitive to the non-periodical changes of data scales. In our research, we also used an AR highway, which has a similar structure to the ones in previous research. More precisely,

$$ArOut^i = \sum_{t\prime=T-window}^{T-1} W_{t'}^i X_{t'}^i + b^i \tag{4.16}$$

where $i \in [1, m]$, $X_{t'}^i \in R$ indicates the i^{th} data of input at time step t', $W_{t'}^i \in R$ denotes the weight matrix for the corresponding time series at time step t', $window$ presents the AR time window (which is also called as the order in autoregressive models), and b^i is the bias. Here, $ArOut^i \in R$ denotes the expected scale of the i^{th} value of $X_{T-1+\Delta}$. All the expected scales are concatenated as follows:

$$ArOut = concat\left(ArOut^1, ArOut^2, \ldots, ArOut^m\right) \tag{4.17}$$

where $ArOut \in R^{\mathbb{k}\times m}$ denotes the expected scale of the final prediction.

4.4.5 Objective function

For the objective function in the framework, we used the Absolute error (also called L1 loss) for different datasets, which is defined as:

$$Loss = min \sum_{t \in train} \sum_{i=1}^{m} \left| X_{t-1+\Delta}^i - \hat{X}_{t-1+\Delta}^i \right| \tag{4.18}$$

In our research, we compared the convergence of the framework based on both the square error (also called L2 loss) and the absolute error. We found that the absolute error performs more smoothly and more robustly than the square error on the experimental data owing to some drastic fluctuation in the time series. Therefore, we use the absolute error for our datasets.

4.5 Evaluation

4.5.1 Related comparison methods

The following methods are used for comparative evaluation:

- AR[84] denotes the autoregression model, which is the simplest and most traditional time series forecasting algorithm that can be used to predict multiple time series linearly.
- LSVR[85] denotes a VAR model in which an SVR is used as the objective function.
- VAR-MLP[36] denotes a model that combines a VAR model with multilayer perception (MLP).

- GRU denotes a fully connected RNN model using a GRU as its recurrent cell.

- LSTNet[4] is a well-known CNN and RNN-based deep neural network, which uses a recurrent skip to capture long-term dependencies.

4.5.2 Calculation matrices

We used the conventional calculation metrics in the sensor data prediction problems: root relative squared error (RSE)

$$RSE = \frac{\sqrt{\frac{1}{m}\sum_t \sum_{i=1}^{m} \left(Y_{it} - \widehat{Y_{it}}\right)^2}}{\sqrt{\frac{1}{m}\sum_t \sum_{i=1}^{m} \left(Y_{it} - mean\left(\widehat{Y_i}\right)\right)^2}} \tag{4.19}$$

Here, $Y, \hat{Y} \in R^{m \times T}$ denote the ground truth and predicting value of sensor data, respectively. We used RSE as our calculation metrics because it evaluates the predicted value regardless of the scale of the data. Here smaller RSE implies better prediction.

4.5.3 Data description

To evaluate our proposed framework, the following 3 benchmark datasets are used:

- Exchange Rate: The data are provided by the author of LSTNet. They describe the daily exchange rates from 8 foreign countries started from 1990 to 2016.

- Sensor20 [18]: The data is recorded from 373 sensors through one year in a Japanese chemical process factory named "PLANET MEISTER". We chose 20 correlated sensors among all the sensors. The whole dataset was splited with 60% for trainning, 20% for both validation and testing.

- Sensor50: Same data source as the one in Sensor20, but with 50 correlated sensors.

The details of these datasets are outlined in the table 4.1:

Table 4.1: The structure of the dataset

Dataset	Length	Dimension	Time Interval
Exchange Rate	7588	8	1 day
Sensor20	20000	20	1 minute
Sensor50	20000	50	1 minute

4.5.4 Experimental details

To evaluate the performance of our proposed framework, a grid search on all the tunable hyperparameters of each method is conducted in all the datasets. More specifically, on all the datasets, the input window size, w, is chosen from {24, 36, 48}. For AR and LSVR, we chose the regularization coefficient λ from $\{2^{-8}, 2^{-6}, \cdots, 1, 2^2\}$ [4]. For VAR-MLP, the hidden size of the MLP is chosen from {20, 30, ..., 100}. For the GRU, LSTNet, and our proposed framework, the hidden size of the recurrent layer is chosen from {20, 30, 50}.For both the LSTNet and our proposed framework, the filter number of the convolutions is chosen from {50, 100, 200, 300}. For the LSTNet, the recurrent skip length is chosen from {3, 6, 9, 12, 24}. For our proposed MLDNet, the number of blocks is chosen from {1, 2, ..., 6}.

During the training procedure, we choose the batch size as 128. We used 2×10^{-3} learning rate for all the dataset.

4.5.5 Experimental results

To evaluate the performance of our proposed framework, we conduct a grid search on all the tunable hyperparameters of each method in all the datasets. Particularly, we chose $L = 3$ in the exchange rate dataset and $L = 4$ in both sensor20 and sensor50. The evaluation results for all the methods are summarized in Tables 4.2, 4.3, 4.4.

Table 4.2: The RSE result of the Exchange Rate Dataset

Methods	Horizon			
	3	6	9	12
AR	0.0228	0.0279	0.0353	0.0445
LSVR	0.0189	0.0284	0.0425	0.0662
VAMLP	0.0265	0.0394	0.0407	0.0578
GRU	**0.0192**	0.0264	0.0408	0.0626
LSTNet	0.0226	0.0280	0.0356	0.0449
MLDNet	0.0197	**0.0261**	**0.0332**	**0.0425**

Table 4.3: The RSE result of the SENSOR20 Dataset

Methods	**Horizon**			
	3	5	10	15
AR	0.0737	0.0701	0.0815	0.0914
LSVR	0.0658	0.0694	0.0832	0.0993
VAMLP	0.0671	0.0807	0.0841	0.0948
GRU	0.0626	0.0630	0.0678	0.0961
LSTNet	0.0395	0.0449	0.0483	0.0772
MLDNet	**0.0216**	**0.0232**	**0.0251**	**0.0428**

Table 4.4: The RSE result of the SENSOR50 Dataset

Methods	**Horizon**			
	3	5	10	15
AR	0.0536	0.0814	0.1252	0.1829
LSVR	0.0442	0.0657	0.1054	0.1592
VAMLP	0.0437	0.0615	0.0930	0.1484
GRU	0.0425	0.0578	0.0914	0.1096
LSTNet	0.0414	0.0568	0.0722	0.1018
MLDNet	**0.0399**	**0.0532**	**0.0703**	**0.0978**

The evaluation results cover all the three metrics on all four datasets. Here, the horizon size represents the different predicting lengths ahead of the current time point. In these tasks, a larger horizon is associated with a more difficult task. On each dataset for each calculation metric, the best result is highlighted in bold in each column in the tables. From the tables above, the evaluation results provide strong evidence of the success of our proposed framework in both short and long horizons and we can conclude that our proposed MLDNet outperforms the other related baseline methods on all the 3 datasets.

4.6 Conclusion

This research aims to predict multiple sensor data in smart industry. However, sensor data is susceptible to variation from other sensors. In addition, sensor data are noisy and there are signal delays between sensors, which can seriously affect the prediction results.

During this research, we found a generally suitable denoising method for industrial sensors by applying wavelet transform to remove the noise and removing the delay between sensors using the correlation coefficients, and finally using the correlation coefficients to select the appropriate sensor data to form our dataset.

In this research, a new deep learning framework (MLDNet) is proposed to solve the sensor data prediction problem. In this framework, by applying the proposed Mixed Length Dilation Blocks, the mixed length of dependencies among related sensor data is captured well. Experiments show that the proposed framework yields ambitious results through all the benchmark datasets compared to those of several baselines methods.

However, there still remains much room for improvement in this research. First, in the Mixed Length Dilation Blocks, currently, 3 kinds of convolution filters are used to capture the combination dependency patterns. However, the capability of this component is significantly restricted by the number of convolutional sizes. Thus, whether there is a better structure that can be applied instead needs to be explored. Second, for this proposed network, we can only capture the effect between the sensor time series in the same block. We know that changes in certain factors (like temperature, etc.) can have a dramatic impact on sensor data, but this model only captures the interactions between sensors within the similar time dependency length and does not capture the impact of mixture of length. Finally, for predicting multivariate time series, it would be more practical if we could extend our model from ordinary sensor time series prediction to various time series in industry.

Therefore, in the next chapter, a special framework for various of multivariate time series forecasting will be presented. This model can predict various of multivariate time series without the need to set the size of the convolution kernel in advance, and can capture the combination pattern of time dependency on the input series, thus predicting multivariate time series more accurately.

The paper in this chapter has been published in:

C1 Song W, Weng W, Fujimura S. Abnormal data analysis in process industries using deep-learning method[C]//2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2017: 2356-2360. https://doi.org/10.1109/IEEM.2017.8290313

C2 Song W, Fujimura S. Sensor Data Prediction in Process Industry by Capturing Mixed Length of Time Dependencies[C]//2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2021: To be published

Chapter 5

Multivariate time series forecasting in smart industry (MDTNet)

5.1 Introduction

Multivariate time series (MTS) data can be found everywhere in our daily lives, such as stock prices, car traffic data, and electricity consumption bills. In these fields, researchers are typically interested in predicting the changes and the trends that may occur in the future based on the historical time series data. Considering electricity consumption as an example, a good electricity consumption forecast can assist power plants in distributing power with increased efficiency and at a decreased cost. However, the problem of MTS forecasting is complex and has been already studied for several years.

In the studies analyzing such sequence data formed by multiple time series, one of the major challenges [86] [87]is the method to determine the complex dependencies among multiple variables in the time dimension. More specifically,

1. The prediction of a variable is not only affected by its historical value but also by the correlated variables.

2. The prediction of one variable will vary with several other input variables, and the dependence on these variables may be different.

3. The prediction of a variable can also vary with the influence of some additional factors that for some reason are not available as input variables.

Most of the effects on a variable caused by other variables can be broadly classified into long-term effects and short-term impacts, which are typically reflected

as "trends" and "fluctuations," respectively.

Thus far, many approaches have been proposed to capture the correlation among variables and dependencies along the time dimension in an MTS forecasting problem. In traditional statistical models, i.e. the vector auto-regressive (VAR) model [8][84][88] and the Gaussian process model [89][90], linear correlations are assumed among the variables. However, owing to the numerous variables, these linear models are prone to overfitting problems. Recently, deep-learning-based models have become well-known and efficient for solving forecasting problems, such as the innovative method of LSTNet [4] and temporal pattern attention long short-term memory model (TPA-LSTM) [47], which achieve STOA performance. Although these models attempt to catch the dependencies between variables, they separately capture the different time length of dependencies. Dependencies of a particular length are captured and subsequently combined for generating the forecasting result of different variables.

However, the long- and short-term dependencies among variables, i.e., their "influence," are not completely independent but mutually affect and restrict each other. For example, the prices of foods, such as vegetables and pork, are simultaneously affected by many factors. They are not only impacted by the long-term effects of supply, demand, and inflation and the cyclical effects of the production season but also by some short-term abrupt effects, such as natural disasters (e.g., animal epidemics or hail). Different factors have different degrees of impact on food prices, and simultaneously, one influencing factor has different effects on food prices in different periods. These long- or short-term factors are strongly related. They restrict and influence each other differently in the entire period. For example, although vegetable prices are affected by the temperature variation (long-term) in summer, they are more susceptible to the abrupt and drastic changes in temperature (short-term) caused by hail. In this example, the dependency of vegetable prices on temperature is the mixture of long- and short-term effects. Furthermore, to forecast vegetable prices, we should definitely consider the effects of several possible factors [91], such as temperature, rainfall, and even prices of other agricultural products. However, even when only the impact of temperature is examined, both the long- and short-term influences should be considered. Therefore, it is very important to capture the mixed dependencies of both the long- and short-term effects of all correlated variables.

In this research, a new deep learning framework is proposed to solve the MTS forecasting problem, named as the mixed dependence time-series network (MDTNet). It consists of two flows: a main flow and an AR highway. An AR highway is commonly used in deep-learning-based models, such as LSTNet[4] and TPA-LSTM [47], for simulating the forecasting scale. The main flow is used for capturing the combination patterns of the long-and short-term dependencies; it has three key

components:

- Stacked dilated convolution component, which applies several dilated convolution filters to parallelly capture the dependencies of all the multivariates in different time lengths.
- Dependencies combination component, which uses vanilla convolution filters to deconstruct the complex combination among the different dependencies.
- Recurrent component, which applies a recursive layer to capture the changes in the different combinations along all the time step.

Finally, the output values of both the flows are added to generate the forecasting result.

From the point of view of application, the proposed framework is capable for predicting the future values for time series with different scale and curvature. First, in the stacked dilated convolution component, CNN is applied to capture the dependencies within different length. From a signaling perspective, CNNs work well as filters to extract useful data fluctuations and trends. These fluctuations and trend changes can be linearly combined into different curvatures through the training process, which also captures the dependencies between these otherwise less correlated time series. Second, due to the existence of the AR highway, the scale of vast majority of related variables are intercepted in the highway. Hence, in the main flow, the model can more easily capture the volatility outside of the scale, and thus can over-predict to some extent.

For the prepocessing of the input data, we believe the following 2 methods are necessary:

1. Rank-based correlation matrices (Spearman) for choosing related time series is first required. Since the rank-based matrices focus only on the order of the magnitude of the values in the sequence, the scale of the sequence itself will not affect the degree of correlation between the two sequences, which also solves the problem of low correlation coefficients of originally correlated sequences because of different scales.
2. In some special cases, logarithmic processing is an effective way to unify the curvature of a sequence. In suitable cases, using log function of different sequences with different bases can lead to more easily converged data, thus accelerating the training and obtaining better results.

To conclude, the main contributions of this reseach are summarized below

1. We proposed a new framework for MTS forecasting which captures the combination patterns of long- and short-term dependencies.

2. Our proposed framework can capture a very long-term dependency and can replace the "recurrent skip" in [4] without previously setting the length.

3. Our proposed framework can capture the impact of additional factors on the current forecast to some extent.

4. We conducted several experiments on 4 benchmark datasets for comparing the proposed framework with other baselines, and the experimental results turn out that our framework performs competitively on all the datasets.

The rest of this chapter is organized as follows. In Section 2, the problem formulation is established. Subsequently, in Section 3 we describe the detailed structure of our proposed framework. Following this, the experimental results and an ablation study are discussed in Section 4. Finally, the conclusions are established in Section 5.

5.2 Previous related studies

For the previous research MLDNet, we have proposed a relative competive framework for time series prediction for chemical sensor data. In this research, a new deep learning framework (MLDNet) is proposed to solve the sensor data prediction problem. In this framework, by applying the proposed Mixed Length Dilation Blocks, the mixed length of dependencies among related sensor data is captured well. Experiments show that the proposed framework yields ambitious results through all the benchmark datasets compared to those of several baselines methods.

However, there still remains much room for improvement in this research. First, in the Mixed Length Dilation Blocks, currently, 3 kinds of convolution filters are used to capture the combination dependency patterns. However, the capability of this component is significantly restricted by the number of convolutional sizes. Thus, whether there is a better structure that can be applied instead needs to be explored. Second, for this proposed network, we can only capture the effect between the sensor time series in the same block. We know that changes in certain factors (like temperature, etc.) can have a dramatic impact on sensor data, but this model only captures the interactions between sensors within the similar time dependency length and does not capture the impact of mixture of length. Finally, for predicting multivariate time series, it would be more practical if we could extend our model from ordinary sensor time series prediction to various time series in industry.

5.3 Problem formulation

In our research, the problem we address belongs to the field of MTS forecasting. More specifically, given a certain number of time series, $X = \{X_{T-w}, X_{T-w+1}, X_{T-w+2}, \ldots, X_{T-1}\}$, where $X_i \in \mathbb{R}^m$ and m is the dimension of the multivariables, we aim to forecast $X_{T-1+\Delta}$, where Δ denotes the required horizon ahead of the current time stamp. Here, X_i denotes the observed multivariables at time i, w is the time window, and $T-1$ denotes the current time stamp. In this task, only the time series from $T-w$ to $T-1$ is considered because all the values before time stamp $T-w$ are supposed to have no useful information for forecasting the multivariates in the future. This is a common assumption, which was also applied in [4] and [47]. In addition, both w and Δ are fixed values, which are previously chosen and customized for different tasks. At time stamp $T-1$, input matrix $X = \{X_{T-w}, X_{T-w+1}, X_{T-w+2}, \ldots, X_{T-1}\} \in \mathbb{R}^{m*w}$ is given to forecast $Y = X_{T-1+\Delta} \in \mathbb{R}^m$.

5.4 Combination patterns of long- and short-term dependencies

In this section, we give the detailed explanation about why the combination pattern of long- and short-term dependencies is important and how it impacts the forecasting time series. Take the vegetable price as an toy example. Assume we want to use check the impact of temperature on vegetable prices, and here are 2 situations:

- In summer, the temperature reduced from 32 degrees centigrade to 28 degrees centigrade in one month, the vegetable price goes down slowly because the cost of preserving vegetables decreased.
- Still in summer, the temperature reduced from 32 degrees centigrade to 28 degrees centigrade in 6 hours, the vegetable prices are soaring. The reason for this is because of the rainstorm that have occurred.

Comparing these 2 situations, the temperature both reduced 4 degrees, but in a different time length, which could be regarded as long-term trend and short-term fluctuation of temperature, respectively. As the result, the vegetable price shows completely different changes in these 2 situations. In particular, even in the same case of temperature drop, if the time length of the drop is different, it causes completely different results. Indeed, the more direct factors that affect vegetable prices are the cost of preserving and damage caused by rain, but these factors are

beyond observations. Fortunately, the unobserved factors still affect the related observed time series and reflect the time series in the time domain with different lengths, which is exactly the different length of dependencies between different time series. Therefore, to capture the impact from these unobserved factors, we choose to capture different time lengths of dependencies from observed related time series. Besides, since different unobserved factors may have different impact on the forecasting time series in different time, it is important to capture the combination pattern of these factors.

Specifically, according to the Figure 5.1, we applied Stacked dilated Conv component to capture the dependencies with different time length. In Dependency Combination component, we deconstruct the combination pattern of the mixed dependencies due to the different impact from different unobserved factors. In Recurrent component, we extract the different impact along different time steps since combination patterns may have an impact to future patterns.

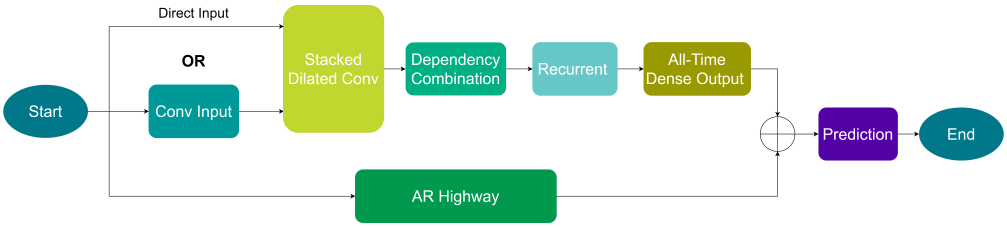

Figure 5.1: Abbreviated flow chart of the whole framework

5.5 Proposed network structure

5.5.1 Conv input/Direct input

The first component of our proposed framework is constructed with a pure convolutional layer without pooling. Owing to the power of convolutions, we employed a CNN to capture the linear combination patterns and local dependencies among all the time series data. Specifically, the convolution layer consists of k filters $C_1, C_2, \ldots, C_k \in \mathbb{R}^{m*h_1}$, where the width of the CNN kernel is same as dimension m of the input time series and h_1 is the height of the CNN kernel. Particularly, when the width of the kernel equals 1, the convolution captures the linear combination patterns among the different time series. In this layer, the i^{th} convolution sweeps over the whole input matrix, X, with a stride of 1 and generates

$$obs_i = Activation\left(C_i * X + b_i\right), \tag{5.1}$$

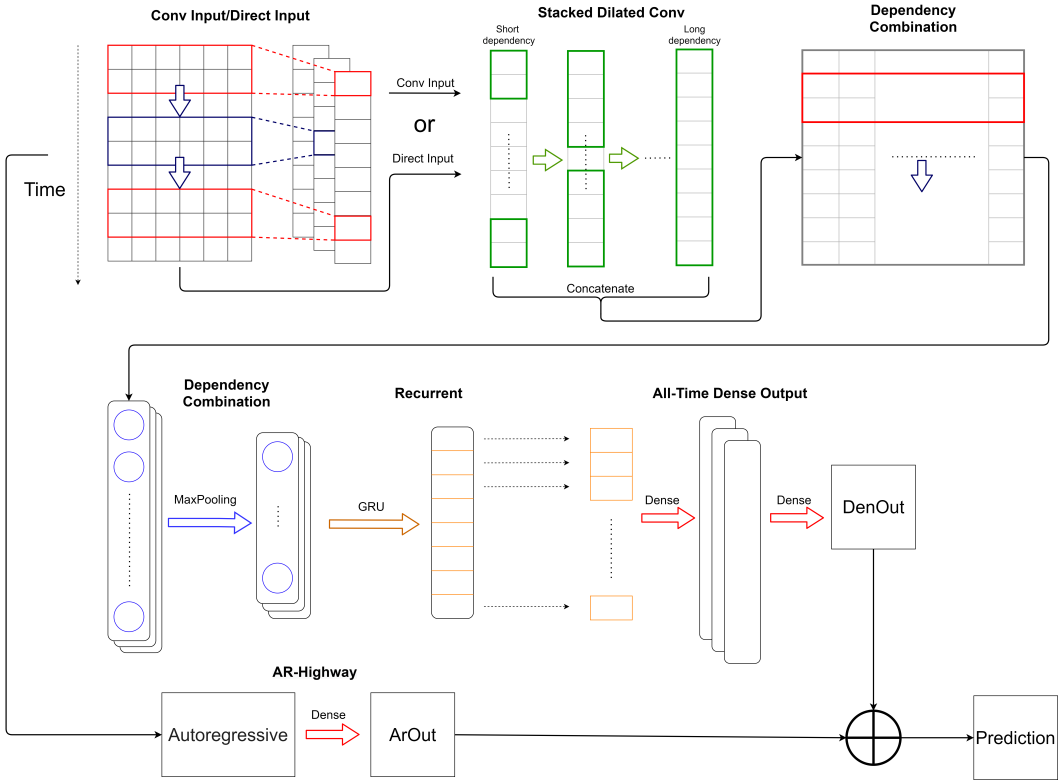

Figure 5.2: the whole structure of the framework

where $*$ indicates the convolution operation, b_i denotes the bias, and $obs_i \in \mathbb{R}^{w-h_1+1}$ denotes the i^{th} convolutional result. We used Leaky RELU[92] as our activation function. Subsequently, the output matrix of this layer is sized with $(w - h_1 + 1) \times k$. With different convolution kernels, different patterns of time series combinations are generated, and the useful ones are captured by training. For each pattern, which is named as one observation, obs_i is the i^{th} observation sequence and *obslen* is the length of each observation sequence. This convolutional layer achieve an outstanding performance when the dimension, m, of the input matrix, X, is relatively small. However, when m becomes extremely large, the input matrix becomes huge, and the number of convolutional filters significantly restricts the capability of this layer. Therefore, we also attempted to directly apply each time series of the input as obs_i in this part, which is called as "direct input." We compare both the types of inputs and show the results in the evaluation discussion.

5.5.2 Stacked dilated conv component

The second component of our proposed framework is constructed with several dilated convolutions[93], aiming to capture all the time dependencies from short to long ones. For sequential problems, a common approach to capture time dependencies is to use RNNs [94]. However, RNNs cannot easily capture a very long-term correlation [95]. Therefore, [4] proposed a novel recurrent-skip component that skip-connects RNN cells to solve this problem. Because for different datasets, the skip length should be pre-defined to capture different lengths of time dependencies, a more appropriate method for solving this problem is to allow the framework to automatically capture all the scales of the time dependencies. Because RNNs cannot be parallelized trained, using several RNNs to capture different lengths of time dependencies may require a high training cost. Thus, in our research, the solution for capturing both short- and long-term dependencies is using stacked dilated convolutions. Each observation sequence is sequentially connected to several dilated convolutional layers. More specifically, each observation sequence output from the previous component is connected with p layers of the dilated convolutions. For each dilated convolution kernel, the receptive field is the entire observation sequence (which is a one-dimensional (1-D) sequence), and the dilated convolution, F, over an element of $x \in \mathbb{R}^n$ is established as:

$$F(s) = (x * f)(s) = \sum_{j=0}^{fz-1} f(j) \cdot x_{s-d \cdot j}, \tag{5.2}$$

where d denotes the dilation factor, fz denotes the size of filter, $x_{s-d \cdot j}$ illustrates the $(s - d \cdot j)^{th}$ value of x, and $f()$ is a filter satisfying $\{0, \ldots, fz-1\} \in \mathbb{R}$. Particularly, when d equals 1, the dilated convolution, F, degenerates to a normal convolution. When d increases, the receptive field of each stride enlarges and captures long-term dependencies. For each dilated convolution layer,

$$obs_i^{l+1} = Activation\left(F^l * obs_i^l + b_i^l\right) \tag{5.3}$$

where obs_i^l denotes the i^{th} observation sequence at the l^{th} layer and F^l denotes the dilated convolution at layer l. Specifically, at the first layer, we consider each observation sequence as one channel, and sweep it with dilated convolutions to obtain several convoluted sequences. For the following layers, these convoluted sequences are treated as new observation sequences, and padding is used at the head of these sequences to obtain same observation lengths as those in the previous layer, which assists the following layers in parallel computing. By each dilated convolutional layer in our framework, the dilation factor, d, becomes twice larger than the ones in the previous layers. Because there are k_1 dilated filters in each layer, the shape of $obs^l \in \mathbb{R}^{obslen \times k_1}$, where $obslen$ is the length of each observation

sequence. After all the dilated convolution layers, the observation sequences are concatenated as expressed below:

$$dep = concat\left(obs^0, obs^1, obs^2, .., obs^p\right) \tag{5.4}$$

where obs^0 in the first layer is grouped up with obs_i output from the previous component. Here $obs^0 \in \mathbb{R}^{obslen \times k}$, $obs^{1..p} \in \mathbb{R}^{obslen \times k_1}$, $dep \in \mathbb{R}^{obslen \times (k+k_1 \times p)}$. In this part of our proposed framework, all the observations are convoluted layer-wise, and a higher layer captures a longer term of time dependency. The concept of using a stack of dilated convolutions originated from [96]; however, in our framework, different structures were used in each dilated convolutional layer. Considering the weights of the time dependencies captured by each observation sequence, to forecast different time series, these weights should be different and learned by training. Thus, all the observation sequences are concatenated, and the method to learn these weights of the time dependencies is explained in the next part.

5.5.3 Dependency combination component

The third component of our framework is constructed with another convolutional layer, aiming to deconstruct the complex combination among different dependencies.

For previous researches, one of the main difficulties is to capture the complex combination patterns among different related time series. The impact from different related time series to the forecasting time series are different, and would be even more complicate along the time axis. In this component, we mainly have 2 targets: 1. We aim to capture the useful dependency patterns at each time steps 2. We aim to keep the useful patterns along the time steps, especially in a very long-term domain, which can replace the "recurrent skip" in the LSTNet.

In this component, we use a large CNN kernel to sweep over all the observations to capture the useful dependency patterns. At each stride of the CNN, the width of the kernel equals the number of observations, and the height of the kernel covers the entire time window by the longest term of the observations. Specifically, in this layer, the i^{th} convolutional filter, D_i, sweeps through the whole input matrix, dep, with a stride of 1 and generates

$$R_i = Activation\left(D_i * dep + b_i\right) \tag{5.5}$$

where $D_i \in \mathbb{R}^{(k+k_1 \times p) \times h_2}$,$R_i \in \mathbb{R}^{obslen-h_2+1}$. Here, $k + k_1 \times p$ is the width of the CNN kernel and h_2 is the height of the kernel. Here, h_2 satisfies $h_1 \times fz \times 2^{p-1} \times h_2 < w$, because we only focus on dependency lengths shorter than the window size. A pooling layer is connected after the convolution, and all the R_i are concatenated as follows:

$$R_i{}' = MaxPooling\left(R_i\right) \tag{5.6}$$

$$R' = concat\left(R'_1, R'_2, R'_3, \ldots, R'_q\right) \tag{5.7}$$

where q denotes the number of convolutional filters, and MaxPooling is a 1-D max pooling layer to compress the sequence for the following recurrent components and help the framework to select extremely long patterns. The size of $R_i\prime \in \mathbb{R}^{\lfloor (obslen-h_2+1)/stride \rfloor}$, where $stride$ is the stride length of the pooling layer. Here, $R' \in \mathbb{R}^{flen \times q}$, and each row of R' denotes the combination pattern captured by the CNN from the entire time window. We assign $flen = \lfloor (obslen - h_2 + 1)/stride \rfloor$.

In this component, the convolutional layer captures the mixture pattern of short- and long-term dependencies and MaxPooling remains more influential dependency patterns and decreases the dimension on time domain. After this pooling layer, dependency patterns along the time axis are "skip-concatenated" together, which shortens the time length between each consecutively captured dependencies and greatly reduces the whole distance of along the time axis. Therefore, we believe that this pooling layer plays the same role as the "recurrent skip" in the LSTNet, which allows capturing the extremely long-term dependencies among multiple variables.

5.5.4 Recurrent component

The following part is a recurrent layer constructed with gated recurrent units (GRUs)[46]. In this component, we aim to capture the impact from the combinations at previous time steps to those at subsequent time steps and reserve the impact of combination pattern from all previous time steps.

To explain more clearly about how previous dependency patterns would have an impact on dependencies in future time steps, again we take the vegetable price as the example. Assuming that vegetables have already experienced one devastating impact caused by a rainstorm which happens long time ago, then Vegetable price would continue to increase slowly due to the cooling of the second rainstorm (because the price has risen to a very high level), but will not slow down due to the reduction in preservation costs (because vegetables will be in short supply for a long time).

Therefore, we used a vanilla recurrent layer constructed by GRUs to learn the "importance pattern" for different combinations along all the time steps. Specifically, the hidden state of the recurrent units at time step t' is computed as follows:

$$r_t = \sigma\left(W_{R'_r} R'_t + W_{hidden_r} hidden_{t-1} + b_r\right) \tag{5.8}$$

$$z_t = \sigma\left(W_{R'_z} R'_t + W_{hidden_z} hidden_{t-1} + b_z\right) \tag{5.9}$$

$$n_t = Relu\left(W_{R'_n} R'_t + r_t \odot (W_{hidden_n} hidden_{t-1}) + b_n\right) \tag{5.10}$$

$$hidden_t = (1 - z_t) \odot n_t + z_t \odot hidden_{t-1} \tag{5.11}$$

where σ is a sigmoid function, $\odot$ denotes the Hadamard product, and $Relu$ indicates the hidden update activation function. R'_t indicates the t^{th} row of R' that outputs from the previous dependence combination component.

Here we choose RNN based structure (GRU) in this component because we need to use a structure to keep the impact from the previous patterns to future pattern, and it is natural to use a hidden state to reserve all these impacts, which happens to be the basic structure of RNN. Comparing with CNN based structures, RNN based structure as Hidden Markov Chain-like models, are more suitable for capturing such features with temporal sequences. CNNs based structures, on the other hand, lose this Markovian property. Because we care more about the sequential order of the mixed patterns and the impact between before and after, we prefer to use the RNN based structure in this component.

5.5.5 All-time dense output

For the last part in our framework, dense layers are used to merge all the hidden outputs from the recurrent units. Based on our experiments, we observed that each hidden output of the recurrent units at time step t denotes different time steps of the dependencies. Moreover, the required horizon ahead of the last time step (which is also the forecasting point) could be impacted by all the dependencies. Thus, we concatenate all the hidden outputs from the recurrent units in all the time steps and append two dense layers. More specifically,

$$HidOut = concat\left(hidden_0, hidden_1, \ldots, hidden_{obslen-h_2}\right) \tag{5.12}$$

$$DenOut_1 = W_{d1} * HidOut + b_{d1} \tag{5.13}$$

$$DenOut = W_{d2} * DenOut_1 + b_{d2} \tag{5.14}$$

where $HidOut \in \mathbb{R}^{flen \times hidr}$ concatenates all the hidden outputs in this step, $hidr$ is the amount of features in the hidden state of a GRU,W_{d1} and W_{d2} are the weight matrices of the dense layers, and b_{d1} and b_{d2} are the biases. $DenOut_1 \in \mathbb{R}^{flen \times m}$ aims to learn the mapping from all the hidden outputs to m sequences, and

here, each sequence denotes one corresponding time series $sig_i \in \mathbb{R}^{flen \times 1}$, where $i = 1, 2, .., m$, which reserves all the time dependencies in different time steps. This dense layer is designed to capture the different importance of several hidden output values at each time step by training. $DenOut \in \mathbb{R}^{1 \times m}$ aims to learn the mapping from all the forecasting time series, sig_i, to the corresponding forecasting result. This dense layer is designed to catch different importance of the time series values along all the time steps.

Our experimental result also showed that the dense layers after concatenation of all the hidden GRU outputs perform better than those after the final output of the GRU. The detailed comparison result will be showed in the evaluation part.

5.5.6 AR highway

Because previous research [4][47] point out that recurrent -based models are insensitive to the nonperiodical changes of data scales, which is also one of the major drawbacks of neural network-based models. In [4], the classical AR model is the most commonly used linear time series forecasting method and was proved to be effective to capture the local scale of the data with highway connections. In our research, we also used an AR highway, which has a similar structure to the ones in previous research. More precisely,

$$ArOut^i = \sum_{t\prime=T-window}^{T-1} W_{t'}^i X_{t'}^i + b^i \tag{5.15}$$

where $i \in [1, m]$, $X_{t'}^i \in \mathbb{R}$ indicates the i^{th} input data at time step t', $W_{t'}^i \in \mathbb{R}$ indicates the weight matrix for the corresponding time series at time step t', $window$ indicates the AR time window (which is also called as order in autoregressive models), and b^i is the bias. Here, $ArOut^i \in \mathbb{R}$ denotes the expected scale of the i^{th} value of $X_{T-1+\Delta}$. All the expected scales are concatenated as follows:

$$ArOut = concat\left(ArOut^1, ArOut^2, \ldots, ArOut^m\right) \tag{5.16}$$

where $ArOut \in \mathbb{R}^{1 \times m}$ denotes the expected scale of the final forecasting. The outputs of the 2 parts are integrated as the final forecasting of $X_{T-1+\Delta}$ as follows:

$$\hat{X}_{T-1+\Delta} = DenOut + ArOut \tag{5.17}$$

where $\hat{X}_{T-1+\Delta}$ is the forecasting value of $X_{T-1+\Delta}$.

5.5.7 Objective function

For the objective function in the framework, we used two different objectives for different datasets.

1. Absolute error (also called L1 loss), which is defined as

$$Loss = min \sum_{t \in train} \sum_{i=1}^{m} \left| X_{t-1+\Delta}^{i} - \widehat{X}_{t-1+\Delta}^{i} \right|. \tag{5.18}$$

In our research, we compared the convergence of the framework based on both the square error (also called as L2 loss) and the absolute error. We found that the absolute error performs more smoothly and more robustly than the square error on the experimental data owing to some drastic fluctuation in the time series. Therefore, we use the absolute error for most of the datasets.

2. Huber Loss[97], which is defined as

$$Loss = min \sum_{t \in train} Z_t, \tag{5.19}$$

$$Z_t = \begin{cases} \frac{1}{2}\left(X_{t-1+\Delta}^{i} - \widehat{X}_{t-1+\Delta}^{i} \right)^2 / \beta & if \left| X_{t-1+\Delta}^{i} - \widehat{X}_{t-1+\Delta}^{i} \right| < \beta \\ \left| X_{t-1+\Delta}^{i} - \widehat{X}_{t-1+\Delta}^{i} \right| - \frac{\beta}{2} & otherwise \end{cases} \tag{5.20}$$

On some datasets, the loss of the absolute error is extremely large during the training procedure, which leads to the gradient explosion problem. However, the Huber loss is more robust to abnormal data points and uses β to decrease the scale of the loss. Thus, we use the Huber loss on the distinct dataset.

5.6 Evaluation

5.6.1 Related comparison methods

The following methods are used for comparative evaluation:

- AR[84] denotes the autoregression model, which is the simplest and most traditional time series forecasting algorithm that can be used to predict multiple time series linearly.
- LSVR[85] denotes a VAR model in which an SVR is used as the objective function.
- VAR-MLP[36] denotes a model that combines a VAR model with multilayer perception (MLP).
- GRU denotes a fully connected RNN model using a GRU as its recurrent cell.

- LSTNet-skip[4] is a well-known CNN and RNN-based deep neural network, which uses a recurrent skip to capture long-term dependencies.
- LSTNet-attn[4] denotes an attention-based version of LSTNet.
- TPA-LSTM[47] denotes an attention-based RNN, which uses an attention mechanism to select related time series.
- MLCNN[98] denotes a multi-task structure which applies the integration of predictive information.
- TEGNN[48] denotes A novel deep learning framework based on transfer entropy graph structure using causal associativity.

5.6.2 Calculation metrics

We used two conventional calculation metrics in the MTS problems: root relative squared error (RSE) and empirical correlation coefficient (CORR)[4], which followed the same evaluation metrics in LSTNet and TPA-LSTM.

$$RSE = \frac{\sqrt{\frac{1}{m}\sum_t \sum_{i=1}^{m} \left(Y_{it} - \widehat{Y}_{it}\right)^2}}{\sqrt{\frac{1}{m}\sum_t \sum_{i=1}^{m} \left(Y_{it} - mean\left(\widehat{Y}_i\right)\right)^2}} \tag{5.21}$$

$$CORR = \frac{1}{m}\sum_{i=1}^{m} \frac{\sum_t \left(Y_{it} - mean\left(Y_i\right)\right)\left(\widehat{Y_{it}} - mean\left(\widehat{Y}_i\right)\right)}{\sqrt{\sum_t \left(Y_{it} - mean\left(Y_i\right)\right)^2}\sqrt{\sum_t \left(\widehat{Y_{it}} - mean\left(\widehat{Y}_i\right)\right)^2}} \tag{5.22}$$

Here, Y, $\hat{Y} \in \mathbb{R}^{m \times T}$ indicate the ground truth and forecasting value of MTS, respectively. We used RSE as one of our calculation metrics because it evaluates the forecasting value regardless the scale of the data. Smaller RSE and higher CORR values imply better forecasting.

5.6.3 Data description

To evaluate our proposed framework, the following four benchmark datasets are used, which are the same datasets used in LSTNet and TPA-LSTM.

- Traffic[99]: The data are collected from the Caltrans Performance Measurement System (PEMS). They were sampled every hour for 48 months by the

California Department of Transportation. These data depict the road occupancy (between 0 and 1) in the San Francisco Bay Area freeways as measured by different sensors.

- Solar Energy[100]: The data are collected from 137 PV plants in Alabama state. They contain the solar power production records of 2006. The time interval between two data points is 10 min.
- Electricity[101]: The data are collected from a solar-electricity power supplier company in Portugal. They present the electricity consumption of 321 clients from 2012 to 2014. The time interval between two data points is 15 min, and we convert the dataset into hourly data.
- Exchange Rate: The data are provided by the author of LSTNet. They describe the daily exchange rates from 8 foreign countries started from 1990 to 2016.

All these datasets present a strong correlation among different time series. The details of the above datasets are outlined in table 5.1:

Table 5.1: Dataset structures

Dataset	**Length**	**Dimension**	**Time Interval**
Exchange Rate	7588	8	1 day
Electricity	26304	321	1 hour
Solar Energy	52560	137	10 minutes
Traffic	17544	862	1 hour

In Table 5.1, "Length" denotes the total time length of each dataset, "Dimension" represents the number of variables in each dataset (which is also m in Section 3.1), and "Time Interval" denotes the length of the period between two data points.

To compare our results with those of previous research, all the datasets are divided into training, validation, and test sets in the proportion of 3:1:1, which is the same splitting scale as in the previous research. After training, the best performing model in the validation set is used for testing. All the data are normalized by vanilla Z-score normalization [102] for each sequence before training.

5.6.4 Experimental details

To evaluate the performance of our proposed framework, we conduct a grid search on all the tunable hyperparameters of each method in all the datasets. More specifically, on all the datasets, the input window size, w, is chosen from {24, 48,

96, 120, 144, 168}. For AR and LSVR, twe chose the regularization coefficient λ from $\{2^{-12}, 2^{-10}, \cdots, 1, 2^2\}$ [4]. For VAR-MLP, the hidden size of the MLP is chosen from {20, 30, ..., 100}. For the GRU, LSTNet, and our proposed framework, the hidden size of the recurrent layer is chosen from {20, 30, 50, 100}.For both the LSTNet and our proposed framework, the filter number of the convolutions is chosen from {50, 100, 200, 300}. For the LSTNet, the recurrent skip length is chosen from {3, 6, 9, 12, 24, 36}. For the TPA-LSTM, the amount of hidden units, m, is chosen from {25, 40, 55, 70}. For all neural network-based models, the dropout is chosen from {0.1, 0.2, 0.25}, and Adam[103] is chosen to optimize the parameters.

For our proposed MDTNet, because it is difficult to select the input style (Conv input or Direct input) to be applied as the first component of our framework for the different datasets, we simply use two frameworks: MDTNet-conv (the proposed framework that uses conv input as the first component) and MDTNet-direct (the proposed framework that uses direct input as the first component) to test all the datasets.

During the training procedure, we choose the batch size as 128. We used a 10^{-3} learning rate for the electricity, solar energy, and traffic datasets and a 2×10^{-3} learning rate for the exchange-rate dataset. Finally, for the electricity dataset, the Huber loss is chosen as the objective function. For the other datasets, L1 loss is used as the objective function.

5.6.5 Experimental results

The evaluation results for all the methods are summarized in Tables 5.2,5.3,5.4,5.5. The evaluation results cover all the three metrics on all the four datasets. Here, the horizon size is set as {3, 6, 12, 24}, respectively, representing the different forecasting lengths ahead of the current time point. In these tasks, a larger horizon is associated with a more difficult task. On each dataset for each calculation metric, the best result is highlighted in bold in each column in the tables. From the tables above, the total number of results in bold is 7 for MDTNet-conv (the proposed framework that uses conv input as the first component), 11 for MDTNet-direct (the proposed framework that uses direct input as the first component), and 15 for the remaining methods.

The evaluation results according to these tables illustrate the good performance of our proposed framework, particularly in the cases with large horizons. Our proposed frameworks, MDTNet-conv and MDTNet-direct, outperform the model, TEGNN, on both the exchange-rate and solar energy datasets by 3.60% and 6.26% in terms of the RSE metric, respectively. Also MDTNet-conv and MDTNet-direct, outperform the current best model, TPA-LSTM, on both the exchange-rate and electricity datasets by 1.17% and 2.01% in terms of the RSE metric, respectively.

This shows the effectiveness and robustness of our proposed framework in capturing long- and short-term mixed dependencies.

Between the two proposed frameworks, MDTNet-conv shows a better performance on the exchange-rate dataset, which proves that the conv input part in our proposed framework is highly effective for this dataset. However, the results show that MDTNet-conv is slightly worse than MDTNet-direct on the traffic and electricity datasets. The probable reason is the amount of multivariables in the input data. In our proposed MDTNet-conv, we use convolutions to capture the linear correlation, which significantly restricts the capability of this part by the number of convolutional filters. Specifically, numerous variables in the dataset imply poor effectiveness of this part. Therefore, MDTNet-direct performs better than MDTNet-conv on both the traffic and electricity datasets, which have relatively

Table 5.2: Evaluation result of Exchange Rate

Matrix	**Methods**	**Horizon**			
		3	6	12	24
RSE	*AR*	0.0228	0.0279	0.0353	0.0445
	LSVR	0.0189	0.0284	0.0425	0.0662
	VAR-MLP	0.0265	0.0394	0.0407	0.0578
	GRU	0.0192	0.0264	0.0408	0.0626
	LSTNet-skip	0.0226	0.028	0.0356	0.0449
	LSTNet-attn	0.0276	0.0321	0.0448	0.0590
	TPA-LSTM	0.0174	0.0243	0.0345	0.0444
	MLCNN	**0.0172**	0.0249	0.0519	0.0438
	TEGNN	0.0178	0.0245	0.0363	0.0449
	MDTNet-conv	0.0176	**0.0243**	**0.0338**	**0.0435**
	MDTNet-direct	0.0186	0.0251	0.035	0.0445
CORR	*AR*	0.9734	0.9656	0.9526	0.9357
	LSVR	0.9782	0.9697	0.9546	0.9370
	VAR-MLP	0.8609	0.8725	0.828	0.7675
	GRU	0.9786	**0.9712**	0.9531	0.9223
	LSTNet-skip	0.9735	0.9658	0.9511	0.9354
	LSTNet-attn	0.9717	0.9656	0.9499	0.9339
	TPA-LSTM	0.9790	0.9709	0.9564	0.9381
	MLCNN	0.9780	0.9610	0.9550	**0.9407**
	TEGNN	**0.9815**	0.9703	0.9566	0.9352
	MDTNet-conv	0.9796	0.9710	**0.9567**	0.9383
	MDTNet-direct	0.9785	0.9710	0.9563	0.9381

Table 5.3: Evaluation result of Electricity

Matrix	Methods	Horizon			
		3	6	12	24
RSE	*AR*	0.0995	0.1035	0.1050	0.1054
	LSVR	0.1523	0.1372	0.1333	0.1180
	VAR-MLP	0.1393	0.1620	0.1557	0.1274
	GRU	0.1102	0.1144	0.1183	0.1295
	LSTNet-skip	0.0864	0.0931	0.1007	0.1007
	LSTNet-attn	0.0868	0.0953	0.0984	0.1059
	TPA-LSTM	0.0823	0.0916	0.0964	0.1006
	MLCNN	0.0851	0.0939	0.0992	0.1021
	TEGNN	0.0822	0.0902	**0.0945**	0.0989
	MDTNet-conv	0.0823	0.0906	0.0957	0.0990
	MDTNet-direct	**0.0821**	**0.0889**	**0.0945**	**0.0981**
CORR	*AR*	0.8845	0.8632	0.8591	0.8595
	LSVR	0.8890	0.8594	0.8003	0.8806
	VAR-MLP	0.8708	0.8389	0.8192	0.8679
	GRU	0.8597	0.8623	0.8472	0.8651
	LSTNet-skip	0.9283	0.9135	0.9077	0.9119
	LSTNet-attn	0.9243	0.9095	0.9030	0.9025
	TPA-LSTM	0.9439	**0.9337**	0.9250	0.9133
	MLCNN	0.9326	0.9011	0.9030	0.9125
	TEGNN	0.9465	0.9330	0.9261	0.9136
	MDTNet-conv	0.9383	0.9217	0.9185	**0.9141**
	MDTNet-direct	**0.9472**	0.9331	**0.9267**	0.9140

more variables. Comparing the performance of the proposed framework with those of the other methods, we find that the following:

1. For large horizons (horizons of 6, 12, and 24), our proposed framework outperforms all the other methods on the electricity, solar energy, and exchange-rate datasets, whereas it performs worse than TEGNN on the traffic dataset. Based on our study of the data, we find that the traffic dataset has more than 800 variables, and each variable represents the road occupancy rate, which is significantly affected by the variable of the nearest road based on its location. The dependencies between each road variable and the other variables are extremely different. In our framework, we use a large convolution layer to capture these combination dependencies and one recurrent layer to sweep these combinations in different time steps. The size of the

Table 5.4: Evaluation result of Solar Energy

Matrix	Methods	Horizon			
		3	6	12	24
RSE	*AR*	0.2435	0.3790	0.5911	0.8699
	LSVR	0.2021	0.2999	0.4846	0.7300
	VAR-MLP	0.1922	0.2679	0.4244	0.6841
	GRU	0.1932	0.2628	0.4163	0.4852
	LSTNet-skip	0.1843	0.2559	0.3254	0.4643
	LSTNet-attn	0.1816	0.2538	0.3466	0.4403
	TPA-LSTM	0.1803	0.2347	0.3234	0.4389
	MLCNN	**0.1794**	0.2983	0.3373	0.4491
	TEGNN	0.1824	0.2612	0.3289	0.4733
	MDTNet-conv	0.1824	0.2419	**0.3226**	0.4454
	MDTNet-direct	0.1805	**0.2336**	0.3236	**0.4357**
CORR	*AR*	0.9710	0.9263	0.8107	0.5314
	LSVR	0.9807	0.9562	0.8764	0.6789
	VAR-MLP	0.9829	0.9655	0.9058	0.7149
	GRU	0.9823	0.9675	0.9150	0.8823
	LSTNet-skip	0.9843	0.9690	0.9467	0.8870
	LSTNet-attn	0.9848	0.9696	0.9397	0.8995
	TPA-LSTM	**0.9850**	**0.9742**	0.9487	0.9081
	MLCNN	0.9814	0.9692	0.9410	0.8913
	TEGNN	0.9847	0.9676	0.9379	0.8933
	MDTNet-conv	0.9847	0.9699	**0.9493**	0.8991
	MDTNet-direct	0.9841	0.9725	0.9467	**0.9097**

convolutional filter highly restricts the capability of capturing the dependencies among variables. In fact, it is better to apply one recurrent sequence for each variable and use an attention map to capture the dependencies at each time step; however, this will lead to an extremely high computational cost. Therefore, our proposed framework shows a slight disadvantage on this type of extremely big datasets.

2. For a small horizon (horizon of 3), our proposed framework outperforms the other baselines; however, it sometimes slightly underperforms than other baselines. The main reason is probably that a small horizon leads to few combinations of the dependencies among the variables but a high correlation among the variable in the time steps. To validate whether capturing combinations of the dependencies hinders the performance of the framework in

Table 5.5: Evaluation result of Traffic

Matrix	Methods	Horizon			
		3	6	12	24
RSE	*AR*	0.5991	0.6218	0.6252	0.6293
	LSVR	0.5740	0.6580	0.7714	0.5909
	VAR-MLP	0.5582	0.6579	0.6023	0.6146
	GRU	0.5358	0.5522	0.5562	0.5633
	LSTNet-skip	0.4777	0.4893	0.4950	0.4973
	LSTNet-attn	0.4897	0.4973	0.5173	0.5300
	TPA-LSTM	0.4487	0.4658	0.4641	0.4765
	MLCNN	0.4492	0.4792	0.4913	0.5353
	TEGNN	**0.4421**	**0.4433**	**0.4508**	**0.4692**
	MDTNet-conv	0.4799	0.4931	0.4782	0.5033
	MDTNet-direct	0.4513	0.4754	0.4710	0.4851
CORR	*AR*	0.7752	0.7568	0.7544	0.7519
	LSVR	0.7993	0.7267	0.6711	0.7850
	VAR-MLP	0.8245	0.7695	0.7929	0.7891
	GRU	0.8511	0.8405	0.8345	0.8300
	LSTNet-skip	0.8721	0.869	0.8614	0.8588
	LSTNet-attn	0.8704	0.8669	0.854	0.8429
	TPA-LSTM	0.8812	0.8717	0.8717	0.8629
	MLCNN	0.8629	0.8416	0.8320	0.8255
	TEGNN	**0.8853**	**0.8820**	0.8743	0.8617
	MDTNet-conv	0.8765	0.8646	0.8642	0.8560
	MDTNet-direct	0.8809	0.8698	**0.8774**	**0.8631**

small horizons, we designed an ablation test for the dependency combination component.

The evaluation results provide strong evidence of the success of our proposed framework on relatively small datasets. However, our proposed framework performs comparably with other representative baselines on big datasets.

5.6.6 Ablation study

In this section, the ablation study conducted to validate the effectiveness of the key components and completeness of the whole framework structure is discussed. More specifically, we removed the key components and constructed our proposed MDTNet with other possible components, and compared all the methods on a

dataset. Because we believe that exchange rates include the most complex combination patterns of long- and short-term dependencies, the exchange-rate dataset was chosen for the ablation study. The detailed experimental frameworks with other components are named as follows:

- *MDT_wo_DC*:
  MDTNet without the dependency combination component.
- *MDT_wo_RC*:
  MDTNet without the recurrent component.
- *MDT_wo_Pooling*:
  MDTNet without the pooling layer in the dependency combination component.
- *MDT_wi_LO*:
  MDTNet with a dense connection only to the output of the GRU at last time step instead of at all the time steps in the all-time dense output component.
- *MDT_wi_Skip*:
  MDTNet with the recurrent skip in the recurrent component, without a pooling layer in the dependency combination component, and with a dense connection only to the output of the GRU at last time step.

Table 5.6: RSE matrix of ablation test on Exchange Rate

RSE	3	6	12	24
MDTNet-conv	**0.0176**	**0.0243**	**0.0338**	**0.0435**
MDT_wo_DC	0.0227	0.0282	0.0356	0.0451
MDT_wo_RC	0.0185	0.0259	0.0355	0.0458
MDT_wo_Pooling	0.0176	0.0244	0.0341	0.0440
MDT_wi_LO	0.0178	0.0246	0.0340	0.0439
MDT_wi_Skip	0.0177	0.0244	**0.0338**	0.0437

We fine tune all the baseline methods, and the test results measured using the calculation metrics are tabulated in Tables 5.6 and 5.7. The results show that

1. The original MDTNet shows the best performance among all the methods.
2. After removing the dependency combination component (in *MDT_wo_DC*), the performance drops significantly, which proves the importance of capturing the combination patterns of both long- and short-term dependencies for all the correlated variables.

Table 5.7: CORR matrix of ablation test on Exchange Rate

CORR	3	6	12	24
MDTNet-conv	**0.9796**	0.9710	**0.9567**	**0.9383**
MDT_wo_DC	0.9734	0.9655	0.9517	0.9362
MDT_wo_RC	0.9753	0.9701	0.9446	0.9359
MDT_wo_Pooling	0.9788	0.9704	0.9553	0.9364
MDT_wi_LO	0.9777	0.9689	0.9541	0.9355
MDT_wi_Skip	0.9792	**0.9711**	0.9564	0.9381

3. After removing the recurrent component (in MDT_wo_RC), the performance is slightly degraded, which demonstrates the importance of capturing the impact from the previous combination patterns to the futures ones along the timeline.

4. Removing the pooling layer or applying the dense connection only to the output of the GRU at the last time step (in $MDT\_wo\_Pooling$ and MDT_wi_LO) cause the performance to drop to a certain extent. This proves the contributions of both the pooling layer and the full connection of the GRU output.

5. As shown in MDT_wi_Skip, when we use the "recurrent skip" instead of the pooling and the dense connection at all the time steps, the performance slightly drops. This proves that our proposed framework achieves the performance of replacing the "recurrent skip" without setting the length.

To conclude, the ablation study shows that our proposed MDTNet has the most robust architecture, particularly with large horizons.

5.7 Conclusion

In this research, a new deep learning framework (MDTNet) is proposed to solve the MTS forecasting problem. In this framework, by applying stacked dilated convolutions with vanilla convolutional and recurrent networks, the mixed dependencies of both long- and short-term factors among multivariates are captured well. Experiments expose that our proposed framework yields competitive results on all the benchmark datasets compared to those of several baselines methods. A comprehensive analysis and an ablation study prove that our proposed MDTNet has a robust architecture and excellent capability to yield accurate forecasts.

For further research, we have several directions for expansion. First, in the dependency combination component, currently a convolution layer is used to capture

the combination dependency patterns. However, the capability of this component is significantly restricted by the number of convolutional filters. Thus, whether there is a better structure that can be applies instead needs to be explored. Second, we used a GRU in the recurrent component. For large-scale inputs, the training will require substantial time because GRUs cannot be trained parallelized on GPUs. Because of the current prominent of attention mechanisms, finding a method to integrate them into MDTNet is another problem worth exploring. Finally, although this study is based on multiple time series forecasting, time series in smart industries are different from general time series. The industrial series are full of noise, and how to select the appropriate time series and how to denoise them are still important issues that need to be carefully studied.

The paper in this chapter will be published in:

J3 Song W, Fujimura S. Capturing combination patterns of long-and short-term dependencies in multivariate time series forecasting[J]. Neurocomputing, 2021. https://doi.org/10.1016/j.neucom.2021.08.100

Chapter 6

Conclusions and prospect

This dissertation mainly discuss about solving time series forecasting problems in smart industries using deep learning methods. In this dissertation, the researches are conducted in both univariate time series forecasting and multivariate time series forecasting. Deep learning methods are commonly used in our research due to their efficiency and accuracy, which leads to better results. In summary, this book encompassed three researches. For each research, a novel framework was proposed and evaluated.

6.1 Summary of research

The purpose of the first study was to forecast the power consumption of a day based on the observed power consumption of the previous days. In this study, we first introduce the concept of internal and external information, proposing an idea to consider time-series information separately. Then, the original encoder-decoder structure based on the attention mechanism is proposed and embedded into the model to solve the dependency loss problem at long distances. A new structure for separating internal and external information is proposed in the encoder and nonlinear corrector to prevent losses that combine them. Due to the specificity of the research purpose, a custom loss function is proposed. This function can save more paid penalties by higher valuations. The experimental results show that the proposed method can achieve better results compared with benchmark methods such as LSTM, LSTM-FFNN and ARIMA.

In the second study, we generalize the type of subject under study from univariate time series forecasting to multivariate time series forecasting, and the data shift from electricity consumption forecasting to more common sensor data prediction. Due to the complexity of the sensor data itself, pre-processing of the data is required. A denoising method generally applicable to industrial sensors is pro-

posed to remove the noise by applying wavelet transform and remove the delay between sensors using correlation coefficients, and finally the appropriate sensor data is selected using correlation coefficients to form our dataset.

In this study, a new suitable deep learning framework (MLDNet) is proposed in this subsection to specifically address the sensor data prediction problem due to the specificity of sensor data in terms of time dependence. In this framework, the mixed length dependencies between the relevant sensor data are well captured by applying the proposed mixed length dilation blocks. Experiments show that the proposed framework yields great results on all benchmark datasets compared to several benchmark methods.

In the third study, we continue to extend the experimental subjects from sensor data to diverse time series forecasting. In order to capture the impact of unknown factors on the target variable forecasting, a new deep learning framework (MDTNet) is proposed to solve the MTS forecasting problem. In this framework, we applied stacked dilated convolution component, which applies several dilated convolution filters to parallelly capture the dependencies of all the multivariates in different time lengths. Dependencies combination component is also applied to deconstruct the complex combination among the different dependencies. Recurrent component is then applied to capture the changes in the different combinations along all the time step. After all, the mixed dependence of long-term and short-term factors in multivariate can be well captured by applying the proposed framework, which can capture both the complex effects between multiple variables and the impact of additional factors on the current forecast to some extent.

In this study, our proposed framework can capture a very long-term dependency and can replace the "recurrent skip" in previous research without previously setting the length. We conducted several experiments on 4 benchmark datasets for comparing the proposed framework with other baselines, and the experimental results turn out that our framework performs competitively and robustly on all the datasets. Comprehensive analysis and ablation studies demonstrate the robust structure and superior ability of our proposed MDTNet to produce accurate forecasting.

In summary, we have studied a variety of time series forecasting under the smart industry. Deep learning methods show better performance in both univariate time series and multivariate time series forecasting. In addition, we propose not only targeted deep learning frameworks for different datasets, but also frameworks for various dataset with generalizability. In the experiments, the proposed frameworks all show considerable competitiveness.

6.2 Applicability of research

With the development of intelligent industry and the demand of productivity, time series has become one of the more and more important production tools. Through time series forecasting, industrial production will be scheduled and planned more properly, and the productivity of intelligent industrialization can be effectively improved.

To achieve industrial intelligence, manufacturers need an accurate forecasting system to implement data monitoring and resource scheduling for application scenarios ranging from simple power forecasting of consumption, to data prediction from sensors, to energy, transportation, and so on. The multiple frameworks proposed in this study can be applied to such situations. Depending on the prediction goals of the dataset and the characteristics of the data, manufacturers are free to choose different frameworks. In particular, for high dimensionality and highly volatile data, the number of MLD blocks can be customized from the MLDNet in the second study. And for data that are heavily influenced by unknown factors or are highly dependent, MDTNet in the third study can be chosen for forecasting.

6.3 Future research direction

For further research, we have several extension directions.

First, in for the capture of dependencies, various convolutional layers are currently used to capture the combined dependency patterns. However, the capability of this component is greatly limited by the number of convolutional filters. If a more suitable component for capturing dependency properties in time series could be created, there would be no need to apply convolution to capture Therefore, whether there is a better structure that can be applied instead needs to be explored.

Second, we use a GRU in the recursive component. for large-scale inputs, training will take a significant amount of time because the GRU cannot be trained in parallel on the GPU. Due to the current popularity of attention mechanisms, various frameworks based on Transfomer have shown powerful capture capabilities in various domains. However, it is difficult to be directly applied in time series forecasting because the Transformer inputs are not sequential. Even with the addition of Position Embedding, the encoding process of Transfomer still loses the natural, time-series-compliant Markovian nature of RNNs. Therefore, how to perform massively parallel training needs to be investigated in depth.

Finally, although this study is based on the forecasting of multiple time series, the forecasting process is still a "offline forecasting" based on the existing real values. Since the time series of intelligent industries have the need for long-

term or even ultra-long-term forecasting, it is more appropriate to build on the forecasting values of the framework to make "online forecasting". Therefore, it is worth exploring if the current framework structure can be improved to suit "offline forecasting".

Publications

Journal papers

J1 Bobby Kurniawan, Wen Song, Wei Weng, Shigeru Fujimura, "Distributed-elite local search based on a genetic algorithm for bi-objective job-shop scheduling under time-of-use tariffs", Evolutionary Intelligence, published online: 31 May 2020, Springer, 15 pages, 31 May 2020. https://doi.org/10.1007/s12065-020-00426-4

J2 Wen Song, Widyaning Chandramitasari, Wei Weng, Shigeru Fujimura, "Short-Term Electricity Consumption Forecasting Based on the Attentive Encoder-Decoder Model", IEEJ Transactions on Electronics, Information and Systems, Vol. 140, No. 7, pp. 846-855, 1 July 2020. https://doi.org/10.1541/ieejeiss.140.846

J3 Wen Song, Shigeru Fujimura, "Capturing combination patterns of long- and short-term dependencies in multivariate time series forecasting", Neurocomputing, Volume 464, Elsevier, pp. 72-82, 13 Nov. 2021. https://doi.org/10.1016/j.neucom.2021.08.100

Refereed conferences papers

C1 Wen Song, Shigeru Fujimura, "Sensor Data Prediction in Process Industry by Capturing Mixed Length of Time Dependencies," 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 1174-1178, Dec 2021, https://doi.org/10.1109/IEEM50564.2021.9672826

C2 Wen Song, Wei Weng, Shigeru Fujimura, "Abnormal data analysis in process industries using deep-learning method", 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 2356-2360, Dec. 2017. https://doi.org/10.1109/IEEM.2017.8290313

Declaration by the author

I hereby confirm that:

- this thesis, submitted in partial fulfilment for the degree of Doctor (Engineering) at the Graduate School of Information, Production and Systems, Waseda University, is my original work;
- quotations and citations have been duly identified by use of quotation marks, change in formatting and use of bibliographical references;
- I have upheld the principles of academic integrity, and I certify that:
  - there is no data falsification in this thesis,
  - there is no data fabrication in this thesis,
  - there is no plagiarism in this thesis;
- this thesis has not been submitted previously or concurrently and, will not be submitted by myself in the future, for any other degree at any other institution.

Name: Song Wen Student No: 44172508

Signature: ..... Date: .....

References

[1] C. J. Bartodziej, "The concept industry 4.0," in *The concept industry 4.0*, pp. 27–50, Springer, 2017.

[2] S. Basu and M. Meckesheimer, "Automatic outlier detection for time series: an application to sensor data," *Knowledge and Information Systems*, vol. 11, no. 2, pp. 137–154, 2007.

[3] B. Ergen, *Signal and image denoising using wavelet transform.* InTech Rijeka, Croatia, 2012.

[4] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018.

[5] S. Casas, C. Gulino, R. Liao, and R. Urtasun, "Spatially-aware graph neural networks for relational behavior forecasting from sensor data," *arXiv preprint arXiv:1910.08233*, 2019.

[6] W. Chandramitasari, B. Kurniawan, and S. Fujimura, "Building deep neural network model for short term electricity consumption forecasting," in *2018 International Symposium on Advanced Intelligent Informatics (SAIN)*, pp. 43–48, IEEE, 2018.

[7] A. J. Chapman and K. Itaoka, "Energy transition to a future low-carbon energy society in japan's liberalizing electricity market: Precedents, policies and factors of successful transition," *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 2019–2027, 2018.

[8] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[9] A. Gonzalez-Vidal, F. Jimenez, and A. F. Gomez-Skarmeta, "A methodology for energy multivariate time series forecasting in smart buildings based on feature selection," *Energy and Buildings*, vol. 196, pp. 71–82, 2019.

[10] P. Newbold and C. W. Granger, "Experience with forecasting univariate time series and the combination of forecasts," *Journal of the Royal Statistical Society: Series A (General)*, vol. 137, no. 2, pp. 131–146, 1974.

[11] F. J. Nogales, J. Contreras, A. J. Conejo, and R. Espínola, "Forecasting next-day electricity prices by time series models," *IEEE Transactions on power systems*, vol. 17, no. 2, pp. 342–348, 2002.

[12] O. Hyde and P. Hodnett, "An adaptable automated procedure for short-term electricity load forecasting," *IEEE Transactions on Power Systems*, vol. 12, no. 1, pp. 84–94, 1997.

[13] P.-F. Pai and W.-C. Hong, "Support vector machines with simulated annealing algorithms in electricity load forecasting," *Energy Conversion and Management*, vol. 46, no. 17, pp. 2669–2688, 2005.

[14] D. L. Marino, K. Amarasinghe, and M. Manic, "Building energy load forecasting using deep neural networks," in *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pp. 7046–7051, IEEE, 2016.

[15] G. M. Jenkins and A. S. Alavi, "Some aspects of modelling and forecasting multivariate time series," *Journal of time series analysis*, vol. 2, no. 1, pp. 1–47, 1981.

[16] K. Chakraborty, K. Mehrotra, C. K. Mohan, and S. Ranka, "Forecasting the behavior of multivariate time series using neural networks," *Neural networks*, vol. 5, no. 6, pp. 961–970, 1992.

[17] W. Song, W. Chandramitasari, W. Weng, and S. Fujimura, "Short-term electricity consumption forecasting based on the attentive encoder-decoder model," *IEEJ Transactions on Electronics, Information and Systems*, vol. 140, no. 7, pp. 846–855, 2020.

[18] W. Song, W. Weng, and S. Fujimura, "Abnormal data analysis in process industries using deep-learning method," in *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pp. 2356–2360, IEEE, 2017.

[19] W. Song and S. Fujimura, "Capturing combination patterns of long-and short-term dependencies in multivariate time series forecasting," *Neurocomputing*, 2021.

[20] I. Melnyk and A. Banerjee, "Estimating structured vector autoregressive models," in *International Conference on Machine Learning*, pp. 830–839, 2016.

[21] H. Qiu, S. Xu, F. Han, H. Liu, and B. Caffo, "Robust estimation of transition matrices in high dimensional heavy-tailed vector autoregressive processes," in *Proceedings of the... International Conference on Machine Learning. International Conference on Machine Learning*, vol. 37, p. 1843, NIH Public Access, 2015.

[22] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Advances in neural information processing systems*, pp. 847–855, 2016.

[23] E. McKenzie, "General exponential smoothing and the equivalent arma process," *Journal of Forecasting*, vol. 3, no. 3, pp. 333–344, 1984.

[24] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "Arima models to predict next-day electricity prices," *IEEE transactions on power systems*, vol. 18, no. 3, pp. 1014–1020, 2003.

[25] S. Makridakis and M. Hibon, "Arma models and the box–jenkins methodology," *Journal of forecasting*, vol. 16, no. 3, pp. 147–163, 1997.

[26] P. C. Phillips, "Fully modified least squares and vector autoregression," *Econometrica: Journal of the Econometric Society*, pp. 1023–1078, 1995.

[27] I. Basawa and R. Lund, "Large sample properties of parameter estimates for periodic arma models," *Journal of Time Series Analysis*, vol. 22, no. 6, pp. 651–663, 2001.

[28] R. Östermark and H. Saxén, "Varmax-modelling of blast furnace process variables," *European Journal of Operational Research*, vol. 90, no. 1, pp. 85–101, 1996.

[29] P. P. Balestrassi, E. Popova, A. d. Paiva, and J. M. Lima, "Design of experiments on neural network's training for nonlinear time series forecasting," *Neurocomputing*, vol. 72, no. 4-6, pp. 1160–1178, 2009.

[30] W. P. Cleveland and G. C. Tiao, "Decomposition of seasonal time series: a model for the census x-11 program," *Journal of the American statistical Association*, vol. 71, no. 355, pp. 581–587, 1976.

[31] W. W. Cooper, J. T. Pastor, J. Aparicio, and F. Borras, "Decomposing profit inefficiency in dea through the weighted additive model," *European Journal of Operational Research*, vol. 212, no. 2, pp. 411–416, 2011.

[32] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "Stl: A seasonal-trend decomposition," *J. Off. Stat*, vol. 6, no. 1, pp. 3–73, 1990.

[33] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[34] F. E. H. Tay and L. J. Cao, "Improved financial time series forecasting by combining support vector machines with self-organizing feature map," *Intelligent Data Analysis*, vol. 5, no. 4, pp. 339–354, 2001.

[35] L.-J. Cao and F. E. H. Tay, "Support vector machine with adaptive parameters in financial time series forecasting," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1506–1518, 2003.

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

[37] M. Khashei and M. Bijari, "A novel hybridization of artificial neural networks and arima models for time series forecasting," *Applied Soft Computing*, vol. 11, no. 2, pp. 2664–2675, 2011.

[38] K.-j. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, 2003.

[39] A. Dedinec, S. Filiposka, A. Dedinec, and L. Kocarev, "Deep belief network based electricity load forecasting: An analysis of macedonian case," *Energy*, vol. 115, pp. 1688–1700, 2016.

[40] X. Qiu, Y. Ren, P. N. Suganthan, and G. A. Amaratunga, "Empirical mode decomposition based ensemble deep learning for load demand time series forecasting," *Applied Soft Computing*, vol. 54, pp. 246–255, 2017.

[41] S. Ryu, J. Noh, and H. Kim, "Deep neural network based demand side short term load forecasting," *Energies*, vol. 10, no. 1, p. 3, 2017.

[42] L. Kuan, Z. Yan, W. Xin, C. Yan, P. Xiangkun, S. Wenxue, J. Zhe, Z. Yong, X. Nan, and Z. Xin, "Short-term electricity load forecasting method based on multilayered self-normalizing gru network," in *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pp. 1–5, IEEE, 2017.

[43] C. Liu, Z. Jin, J. Gu, and C. Qiu, "Short-term load forecasting using a long short-term memory network," in *2017 IEEE PES innovative smart grid technologies conference Europe (ISGT-Europe)*, pp. 1–6, IEEE, 2017.

[44] W. He and Y. Chai, "An empirical study on energy disaggregation via deep learning," *Advances in Intelligent Systems Research*, vol. 133, pp. 338–342, 2016.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[46] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[47] S.-Y. Shih, F.-K. Sun, and H.-y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Machine Learning*, vol. 108, no. 8-9, pp. 1421–1441, 2019.

[48] H. Xu, Y. Huang, Z. Duan, X. Wang, J. Feng, and P. Song, "Multivariate time series forecasting with transfer entropy graph," *arXiv preprint arXiv:2005.01185*, 2020.

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

[50] V. Mnih, N. Heess, A. Graves, *et al.*, "Recurrent models of visual attention," in *Advances in neural information processing systems*, pp. 2204–2212, 2014.

[51] R. Drossu and Z. Obradovic, "Rapid design of neural networks for time series prediction," *IEEE Computational Science and Engineering*, vol. 3, no. 2, pp. 78–89, 1996.

[52] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[53] B. Liu and I. Lane, "Attention-based recurrent neural network models for joint intent detection and slot filling," *arXiv preprint arXiv:1609.01454*, 2016.

[54] B. M. Williams, "Multivariate vehicular traffic flow prediction: evaluation of arimax modeling," *Transportation Research Record*, vol. 1776, no. 1, pp. 194–200, 2001.

[55] H. T. Pham, B.-S. Yang, *et al.*, "A hybrid of nonlinear autoregressive model with exogenous input and autoregressive moving average model for long-term machine state forecasting," *Expert Systems with Applications*, vol. 37, no. 4, pp. 3310–3317, 2010.

[56] M. Cools, E. Moons, and G. Wets, "Investigating the variability in daily traffic counts through use of arimax and sarimax models: assessing the effect of holidays on two site locations," *Transportation research record*, vol. 2136, no. 1, pp. 57–66, 2009.

[57] K. Siwek, S. Osowski, and R. Szupiluk, "Ensemble neural network approach for accurate load forecasting in a power system.," *International Journal of Applied Mathematics & Computer Science*, vol. 19, no. 2, 2009.

[58] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble deep learning for regression and time series forecasting," in *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*, pp. 1–6, IEEE, 2014.

[59] J. W. Taylor and R. Buizza, "Neural network load forecasting with weather ensemble predictions," *IEEE Transactions on Power systems*, vol. 17, no. 3, pp. 626–632, 2002.

[60] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[61] "Calendar for year 2016."

[62] T. Gneiting, "Making and evaluating point forecasts," *Journal of the American Statistical Association*, vol. 106, no. 494, pp. 746–762, 2011.

[63] J. Konečnỳ, J. Liu, P. Richtárik, and M. Takáč, "Mini-batch semi-stochastic gradient descent in the proximal setting," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 242–255, 2015.

[64] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, PMLR, 2013.

[65] K. Y. Chan, S. Khadem, T. S. Dillon, V. Palade, J. Singh, and E. Chang, "Selection of significant on-road sensor data for short-term traffic flow forecasting using the taguchi method," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 255–266, 2011.

[66] F. Liu, F. Liu, W. Wang, and B. Xu, "Mems gyro's output signal de-noising based on wavelet analysis," in *2007 International Conference on Mechatronics and Automation*, pp. 1288–1293, IEEE, 2007.

[67] O. A. Omitaomu, V. A. Protopopescu, and A. R. Ganguly, "Empirical mode decomposition technique with conditional mutual information for denoising operational sensor data," *IEEE sensors journal*, vol. 11, no. 10, pp. 2565–2575, 2011.

[68] J. Orchard, M. Ebrahimi, and A. Wong, "Efficient nonlocal-means denoising using the svd," in *2008 15th ieee international conference on image processing*, pp. 1732–1735, IEEE, 2008.

[69] Q. Pan, L. Zhang, G. Dai, and H. Zhang, "Two denoising methods by wavelet transform," *IEEE transactions on signal processing*, vol. 47, no. 12, pp. 3401–3406, 1999.

[70] M. Alfaouri and K. Daqrouq, "Ecg signal denoising by wavelet transform thresholding," *American Journal of applied sciences*, vol. 5, no. 3, pp. 276–281, 2008.

[71] A. Jain and E. Y. Chang, "Adaptive sampling for sensor networks," in *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*, pp. 10–16, 2004.

[72] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Computer communications*, vol. 30, no. 14-15, pp. 2826–2841, 2007.

[73] F. Schweppe, "Sensor-array data processing for multiple-signal sources," *IEEE Transactions on Information Theory*, vol. 14, no. 2, pp. 294–305, 1968.

[74] D. H. Johnson, "Signal-to-noise ratio," *Scholarpedia*, vol. 1, no. 12, p. 2088, 2006.

[75] E.-C. Chang, S. Mallat, and C. Yap, "Wavelet foveation," *Applied and Computational Harmonic Analysis*, vol. 9, no. 3, pp. 312–335, 2000.

[76] Z. Liu, Z. He, W. Guo, and Z. Tang, "A hybrid fault diagnosis method based on second generation wavelet de-noising and local mean decomposition for rotating machinery," *ISA transactions*, vol. 61, pp. 211–220, 2016.

[77] D. L. Donoho and J. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[78] J. Ma, J. Xue, S. Yang, and Z. He, "A study of the construction and application of a daubechies wavelet-based beam element," *Finite Elements in Analysis and Design*, vol. 39, no. 10, pp. 965–975, 2003.

[79] T. Downie and B. Silverman, "The discrete multiple wavelet transform and thresholding methods," *IEEE Transactions on signal processing*, vol. 46, no. 9, pp. 2558–2561, 1998.

[80] P. J. Rousseeuw and C. Croux, "Alternatives to the median absolute deviation," *Journal of the American Statistical association*, vol. 88, no. 424, pp. 1273–1283, 1993.

[81] M.-T. Puth, M. Neuhäuser, and G. D. Ruxton, "Effective use of pearson's product–moment correlation coefficient," *Animal behaviour*, vol. 93, pp. 183–189, 2014.

[82] J. H. Zar, "Significance testing of the spearman rank correlation coefficient," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 578–580, 1972.

[83] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[84] J. D. Hamilton, *Time series analysis*, vol. 2. Cambridge Univ Press, 1994.

[85] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation and signal processing," *Advances in neural information processing systems*, vol. 9, pp. 281–287, 1996.

[86] J. Aitchison and I. R. Dunsmore, *Statistical prediction analysis*. CUP Archive, 1980.

[87] J. Wolberg, "Prediction analysis," in *Designing Quantitative Experiments*, pp. 90–127, Springer, 2010.

[88] H. Lütkepohl, *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[89] R. Frigola, F. Lindsten, T. B. Schon, and C. Rasmussen, "Bayesian inference and learning in gaussian process state-space models with particle mcmc," in *Advances in Neural Information Processing Systems*, pp. 3156–3164, 2013.

[90] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain, "Gaussian processes for time-series modelling," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 371, p. 20110550, Feb 2013.

[91] F. V. Waugh, "Quality factors influencing vegetable prices," *Journal of farm economics*, vol. 10, no. 2, pp. 185–196, 1928.

[92] S. S. Liew, M. Khalil-Hani, and R. Bakhteri, "Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems," *Neurocomputing*, vol. 216, pp. 718–734, 2016.

[93] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[94] X. Cai, N. Zhang, G. K. Venayagamoorthy, and D. C. Wunsch II, "Time series prediction with recurrent neural networks trained by a hybrid pso–ea algorithm," *Neurocomputing*, vol. 70, no. 13-15, pp. 2342–2353, 2007.

[95] A. B. Dieng, C. Wang, J. Gao, and J. Paisley, "Topicrnn: A recurrent neural network with long-range semantic dependency," *arXiv preprint arXiv:1611.01702*, 2016.

[96] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[97] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*, pp. 492–518, Springer, 1992.

[98] J. Cheng, K. Huang, and Z. Zheng, "Towards better forecasting by fusing near and distant future visions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 3593–3600, 2020.

[99] PEMS, *Traffic data from 2015 2016*, 2016 (accessed December 1, 2020).

[100] NREL, *Solar power data of 2006*, 2006 (accessed December 1, 2020).

[101] Elergone, *Electricity Load data from 2012 to 2014*, 2014 (accessed December 1, 2020).

[102] A. Jain, K. Nandakumar, and A. Ross, "Score normalization in multimodal biometric systems," *Pattern recognition*, vol. 38, no. 12, pp. 2270–2285, 2005.

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