DEGREE PROJECT IN INFORMATION AND COMMUNICATION TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

Federated Learning for Time Series Forecasting Using Hybrid Model

YUNTAO LI

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Abstract

Time Series data has become ubiquitous thanks to affordable edge devices and sensors. Much of this data is valuable for decision making. In order to use these data for the forecasting task, the conventional centralized approach has shown deficiencies regarding large data communication and data privacy issues. Furthermore, Neural Network models cannot make use of the extra information from the time series, thus they usually fail to provide time series specific results. Both issues expose a challenge to large-scale Time Series Forecasting with Neural Network models. All these limitations lead to our research question:

Can we realize decentralized time series forecasting with a Federated Learning mechanism that is comparable to the conventional centralized setup in forecasting performance?

In this work, we propose a Federated Series Forecasting framework, resolving the challenge by allowing users to keep the data locally, and learns a shared model by aggregating locally computed updates. Besides, we design a hybrid model to enable Neural Network models utilizing the extra information from the time series to achieve a time series specific learning. In particular, the proposed hybrid outperforms state-of-art baseline data-central models with NN5 and Ericsson KPI data. Meanwhile, the federated settings of purposed model yields comparable results to data-central settings on both NN5 and Ericsson KPI data. These results together answer the research question of this thesis.

Keywords

Federated Learning, Time Series Forecasting, Recurrent Neural Networks, Long Short-Term Memory, Hybrid Model

Sammanfattning

Tidseriedata har blivit allmänt förekommande tack vare överkomliga kantenheter och sensorer. Mycket av denna data är värdefull för beslutsfattande. För att kunna använda datan för prognosuppgifter har den konventionella centraliserade metoden visat brister avseende storskalig datakommunikation och integritetsfrågor. Vidare har neurala nätverksmodeller inte klarat av att utnyttja den extra informationen från tidsserierna, vilket leder till misslyckanden med att ge specifikt tidsserierelaterade resultat. Båda frågorna exponerar en utmaning för storskalig tidsserieprognostisering med neurala nätverksmodeller. Alla dessa begränsningar leder till vår forskningsfråga:

Kan vi realisera decentraliserad tidsserieprognostisering med en federerad lärningsmekanism som presterar jämförbart med konventionella centrala lösningar i prognostisering?

I det här arbetet föreslår vi ett ramverk för federerad tidsserieprognos som löser utmaningen genom att låta användaren behålla data lokalt och lära sig en delad modell genom att aggregera lokalt beräknade uppdateringar. Dessutom utformar vi en hybrid modell för att möjliggöra neurala nätverksmodeller som kan utnyttja den extra informationen från tidsserierna för att uppnå inlärning av specifika tidsserier. Den föreslagna hybrida modellen presterar bättre än state-of-art centraliserade grundläggande modeller med NN5- och Ericsson KPI-data. Samtidigt ger den federerade ansatsen jämförbara resultat med de data-centrala ansatserna för både NN5- och Ericsson KPI-data. Dessa resultat svarar tillsammans på forskningsfrågan av denna avhandling.

Nyckelord

Federerad Inlärning, Tidsserieprognostisering, Återkommande Neurala Nätverk, LSTMs, Hybrida Modeller

Acknowledgements

I would first like to thank my industrial supervisors Mr. Johan Haraldson and Mr. Tony Larsson at Ericsson, for giving me this opportunity. They allowed this paper to be my own work and steered me in the right direction by providing constructive guidance consistently.

I would like to thank my supervisor Prof. Sarunas Girdzijauskas at KTH. His full support encouraged me to pursue my research topic and carry out the work of completing this thesis. The insightful advice from him was an invaluable asset of this thesis.

I would like to thanks my examiner Prof. Henrik Boström at KTH for his constructive and comprehensive feedback on this thesis. His remarks greatly improved the quality of this thesis. I also want to express my gratitude to Prof. Henrik Boström for organizing my defense be an enjoyable moment, and for your profound comments and suggestion.

I would also like to thanks my defense opponent Mr. Benjamin Naoto Chiche, for providing critical but valuable comments during my defense.

A special thanks for my colleagues and friends who were involved in this research project: Mr. Fernando Díaz González, Mr. Lukas Bjarre, Mr. Dan Cristian Chirascu, Mr. Andreas Yokobori Sävö, for their passionate support and input. I am gratefully indebted to their valuable contribution to this thesis.

Finally, I would like to thank my family for everything. You are the ones who let me finish my degree.

Authors

Yuntao Li <yuntao@kth.se>
Information and Communication Technology
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden
Ericsson AI Research

Examiner

Prof. Henrik Boström
Stockholm, Sweden
KTH Royal Institute of Technology

Supervisor

Prof. Sarunas Girdzijauskas
Stockholm, Sweden
KTH Royal Institute of Technology

Acronyms

AI: Artificial Intelligence

ANN: Artificial Neural Network

ARIMA: Auto-regressive Integrated Moving Average

BNN: Bayesian Neural Network

CART: Classification And Regression Trees

BPTT: Back-Propagation Through Time

DGNN: Dynamic Computation Graph Neural Network

DNN: Deep Neural Network

ES: Exponential Smoothing

FFNN: Feed-forward Neural Network

KNN: K-nearest Neighbors

KPI: Key Performance Indicator

LSTM: Long Short-Term Memory

MASE: Mean Absolute Scaled Error

MIMO: Multiple Input Multiple Output

ML: Machine Learning

MLP: Multi Layer Perceptron

MIMO: Multiple Input Multiple Output

MSE: Mean Square Error

NN: Neural Network

RNN: Recurrent Neural Network

SGD: Stochastic Gradient Descent

sMAPE: Symmetric Mean Absolute Percentage Error

STL: Seasonal and Trend decomposition using Loess

TS: Time Series

TSF: Time Series Forecasting

Contents

1 Introduction

Classical machine learning and deep learning algorithms require a sufficient amount of data to obtain a fair performance [58]. Normally, the demanded datasets are large in scale, and they may be distributed in multiple sources which are laborious to access from a centralized point. It is natural to introduce a general distributed scheme while designing the ecosystem. In the era of Big Data, the increasing volume, and variety of the data at the edge impedes the performance of cloud computing [36, 50, 52]. This is also a challenge for designing a distributed deep learning system.

Recent research from Google AI presents a possible new way of achieving this. Federated Learning, a decentralized approach that attempts to tackle the aforementioned limitations, has been applied to Google's Gboard. It involves all the devices in the system collecting data and making individual improvements. These individual improvements are aggregated on a central service and every device is then updated with the combined result [29, 67].

On the other hand, in industries like telecommunications or healthcare, data is usually formed as a series of observations taken sequentially in time. In particular, Ericsson collects the Key Performance Indicators (KPIs) over time from all edge devices among its base stations. Extracting insights from this data and making predictions derived from it can help optimize and tailor the system.

Unlike problems of classification and regression, time series problems not only add the complexity of order or temporal dependence between observations but also require extra information of the series such as trend, level, and seasonality. Neural network models like recurrent neural network (RNN) can capture the temporal information from the sequential data. However, they cannot make use of the extra information from the time series, thus they usually fail to provide time series specific results. As a consequence, achieving large-scale time series forecasting with distributed deep learning is still a challenge.

In this thesis, we further investigate the large scaled time series forecasting, we build the distributed framework based on the idea of federated learning, which aims to resolve the tension of large data communication. Besides, we design a

time-series specific neural network model, which is inspired by S.Smyl [56], to enable the neural network model utilizing the extra information from the time series. We show that by following this approach, the forecasting performance is comparable to the conventional data centralized approach.

1.1 Background

The advent of 5G, a new standard for Radio Access Networks (RAN) with much higher bit-rates, lower latency, and able to operate at a throughput that is 100-1000 times faster than current networks [9, 65], has arrived just in time for the advancement of smart edge devices. The introduction of 5G makes it possible for edge devices to respond in a matter of milliseconds [65].

It has been estimated that the number of such devices will soar to a million by 2025 [20, 65]. Nowadays, many edge devices can already play some fundamental roles such as data collection and processing, and it is foreseeable that smart edge devices will permeate our daily lives by providing important mensuration for decision-making. With the advancement of information technology, it is expected to see that millions of sensors and edge devices will work collectively to support communication and controlling of the smart infrastructures over complex networks. Edge devices will gain more powerful computing capacity, allowing them to perform on-site complex computations [30, 68].

In the era of Big Data, the growth in the volume of the data at the edge results in costly communication. It has been shown that cloud computing is not suitable for the tasks involving sending large volumes of data under some circumstances [4, 49]. Also, the growth of the awareness of potential data-abusing forced lawmakers to introduce new regulations like GDPR [22] on data protection and privacy. All these limitations and concerns will increase the demand for edge computing.

The introduction of edge computing provides a promising solution for distributed machine learning and deep learning, with the premises of more powerful edge devices and faster network standard. However, as mentioned before, classical machine learning and deep learning algorithms require a sufficient amount of data to obtain a fair performance [58]. Normally, the demanded datasets are

distributed in multiple sources, which are laborious to access from a centralized point. Also, as mentioned before, the current cloud-based framework usually faces limitations due to the large data volumes and privacy issues. As a consequence, a new technique that allows learning at the edge is demanded.

Google's research on Federated Learning in 2017 [47], offered an alternative for AI research to achieve distributed machine learning in a decentralized manner. In Federate Learning, a central server will serve as a coordinator of the collaborating edge-devices to perform learning task. The central server is in charge of distributing the latest version of the shared model to all participants (edge-devices), meanwhile, the only communication from an edge device to the central server is the local updates. Repeating the communication rounds allows all edge devices learning collaboratively to improve the learning result without sharing the data in a center point.

Besides, in industries like telecommunications or healthcare, data is usually formed as a series of observations taken sequentially in time. Conventionally, time series forecasting is often considered on an individual basis, and predictive models are then fit with series-specific parameters. These series-specific models based on robust statistical methods can often make good predictions. However, the limitation of statistical methods is also obvious given the fact that they assume linearity modeling [1], thus not scaling well and failing to capture the expressive general patterns that can be learned from studying many fundamentally related series.

Neural Network (NN) models are also used for solving time series forecasting problems. Although the Neural Network models like Recurrent Neural Network and LSTM allow the preservation of the sequential information and persistence of the knowledge procured from subsequent time steps, the issue of using NN models is that they cannot make use of the extra information from the time series data. The research from Zhang and Qi pointed out that standalone Neural Networks are not able to model the trend and seasonality effectively, implying that forecasting performance with Neural Networks can be improved via proper detrending or deseasonalization of the raw time series ahead of modeling [70]. The study conducted by Nelson et al. using 68 monthly time series from M-competition [41]

showed that deseasonalization of the time series data achieved better forecasting performance with Neural Networks comparing to non-deseasonalized setup [48], as well as Ben Taieb et al. reached a similar conclusion based on the experiment on NN5 data [7].

On the other hand, Makridakis et al. announced that combined time series forecasting method outperforms in comparison to other methods based on their study [43]. Several of researches in the hybrid method in time series forecasting were conducted. In particular, The early approach of hybridizing the Artifical Neural Network and ARIMA was made by Zhang et al. The benefits of this proposal allows to understanding both linear (from ARIMA) and non-linear (from ANNs) patterns from the time series [69]. The subsequent works from Zhang et al. [70] and Ben Taieb et al. [7] further demonstrated the effectiveness of adding the conventional statistical time series methods prior to the Neural Network modeling for time series forecasting problem. Currently, the most promising approach is the Exponential Smoothing -Recurrent Neural Network (ES-RNN) hybrid model from S.Smyl at Uber Research. The model does not just simply combine the conventional Exponential Smoothing (ES) and Neural Networks in an ensembled manner, instead, it allows all parameters, including Exponential Smoothing seasonality and smoothing parameters, fitting concurrently with Neural Network weights in same Gradient Descent method, which enables the truly time-series-specific learning with Neural Networks [56].

1.2 Problem

Distributed time series forecasting is still coupling with the conventional conventional centralized approach. Wu et al. proposed a prediction framework for large-scale time series data sources [66]. Guo worked in the same direction with the additional investigation on correlation mining in distributed time series streams [26]. Both pieces of researches rely on the data aggregation on a center point. As aforementioned, in the era of Big Data, the increasing of the volume and variety of the data at the edge and sensitive nature of the data will be the bottleneck of the conventional centralized based methods.

As aforementioned, in the era of Big Data, the increasing of the volume and variety of the data at the edge and sensitive nature of the data will be the bottleneck of the conventional centralized based methods. Federated Learning will be the solution to overcome these limitations and concerns. What makes Federated Learning framework outstands comparing to the conventional centralized approach is that it allows data staying in the local devices instead of being aggregated to a single machine or a datacenter [47]. Therefore, it is worth investigating the Federated Time Series Forecasting.

1.3 Purpose

As mentioned before, Federated Time Series Forecasting is not studied in any literature. Given the superiority and benefit that one can gain from the edge-learning mechanism, we believe it is worthy of investigation in this direction for large scale time series problems. This leads to our prime research question:

Can we realize decentralized time series forecasting with a Federated Learning mechanism that is comparable to the conventional centralized setup in forecasting performance?

The above question concerns aspects of 1). if proposed forecasting model is effective, and 2). if the proposed Federated Time Series Forecasting approach has a comparable performance against data-centralized setup. In particular:

- In the context of our study, effectiveness is decided by comparing the forecasting performance of the proposed approach and the state-of-art models.
- The objective of Federated Learning is to achieve collaborative learning across all edge devices without sharing data at a center point. Due to the nature of Federated Learning, it is not always the case that all participants will be trained in all communication rounds [47]. The performance

of Federated settings should be comparable to conventional centralized approach, in term of prediction accuracy with same prediction settings.

The answer to this research question will advance the field of large-scale time series forecasting without centralizing the data.

1.4 Objectives

The objective of this thesis is to answer the research question in accordance with the aspects discussed in the previous section. In order to fully answer the research question, we need to:

- Build the Federated Learning framework, as there is no existing solution at the time composing this document.
- Propose a hybrid model for Time Series Forecasting problem, base on the work from S.Smyl.
- Build the baseline model(s)
- Analysis and evaluate the performance of the hybrid model in contrast to the baseline model(s), exposing the effectiveness of the hybrid model, as well as the performance of Federated setting and centralized setting.

The result of this work will be used to validate the possibility to realize Federated Time Series Forecasting, in terms of prediction performance. The performance will be quantified by using different datasets.

1.5 Benefits, Ethics and Sustainability

Increasingly, edge devices have been used widely used thanks to the development of mobile internet and Internet of Things (IoT) technology. The powerful sensors and computing units on these edge devices are frequently used for computation purposes, which mean they have access to an unprecedented amount of data. On the other hand, it is estimated by Cisco that the number of devices connected to IoT will become 50 billion by 2020. The emerging edge devices introduce new

challenges as it has been shown that cloud computing is not suitable for the tasks involving sending large volumes of data under some circumstances [4, 52]. As a consequence, edge computing-based architecture can be considered as a solution to this challenge.

The idea of Federated Time Series Forecasting, which is based on the very concept of Federated Learning, is an example of realizing the concept of edge computing in large scalable machine learning and deep learning. Comparing to the conventional centralized approach, it avoids sending the data to a center point, which reduces the network traffic load. The learning is achieved by collecting the benefits of shared models trained from the device point, therefore, it can achieve the learning objective regardless of the network traffic bottlenecks. Companies like Ericsson are looking forward to applying the technology to achieve real-time, intelligent and autonomous decision-making for its network [19].

On the other hand, the nature of Federated Learning does not require data to be stored at a centralized location. This feature has disrupted the current AI paradigm, by finding a solution of collecting more data for training more complex AI algorithm without concerns of data security issues. The compromising and abusing of data has become a prime concern in society in the past years. The notorious case of data-abusing like Facebook and Cambridge Analytica [24] forced lawmakers to introduce new regulations like GDPR [22] to protect the data and privacy. The introduce of Federated Time Series Forecasting can resolve the ethical concerns of data-abusing and compromise of confidential information while achieving a sustainable AI environment.

Additionally, the collaborative nature of Federated Learning leads to a sustainable use of the computational resources across the network.

1.6 Delimitations

We identify the following limitations of the work.

- We only experiment with NN5 data and Ericsson KPI data. Both data have strong seasonality. The data with no seasonality is not considered.

This means that the performance of our model may be different when non-seasonal data is used.

- The critical factors for Time Series Forecasting problem like holiday effect, will not be our scope in this thesis. The holiday effect of the Time Series data can be vital for the final forecasting, however, due to the time constraint, we will not include this part in our investigation.
- As aforementioned, the performance of Federated Learning can be influenced by resources like networks setup and other computational resources. We are not going to validate the efficiency factors in this thesis but only focus on the realization of the Federated Time Series Forecasting itself.

1.7 Outline

The rest of the thesis is organized as follows:

- In Chapter 2, we introduce the relevant theory and related work.
- In chapter 3, we present the research methodology.
- In Chapter 3.5, we present experiments and their results
- In Chapter 4, we discuss the results from the experiments, the further we will present the result to answer the research question.
- In Chapter 5, we will summarise our works and propose potential future works.

2 Extended Background

This chapter introduces the background of our research. Our research problem concerns areas of Time Series Forecasting and Federated Learning, so we will introduce the relevant theories with respect to these two topics. The general setup for this chapter is 1). introduce the time series forecasting problem and some state-of-art work, 2). introduce Artificial Neural Networks, mainly RNNs and LSTMs they are used our study and 3). distributed learning and federated learning.

2.1 Time Series Forecasting

Forecasting future observation intervals by using historical observations is desired in many applications. It can help people make decisions prior to the occurrence of the risk and undesirable incidents, thus reducing the potential loss. As a consequence, there are many studies on Time Series data, such as financial series [28, 61], the prediction of the stock market [53], macroeconomic variables [37], and accounting balance sheet information [21]. The objective of Time Series Forecasting is to utilize the information in a time series to generate more accurate forecast results.

2.1.1 Time Series Data

A time series can be regarded as a sequence of numbers, with temporal information about when those numbers were recorded. For instance, a simple monthly time series dataset can be represented as in table 2.1

Besides the observation values, time series usually contains some extra information called patterns. The patterns together with the observation value will be used for the time series forecasting. Normally, the pattern *trend* and *seasonality* are used frequently.

Year	Observation
2000	12
2001	15
2002	18
2003	13
2004	15

Table 2.1: An example of yearly time series

Trend

A *trend* pattern describes the long-term increase or decrease of the time series. As it is used to describe the overall change of the series direction, it does not necessarily come in linear form. It can be an increasing form following by a decreasing form or vice versa. In figure 2.1, we show the *trend* of an NN5 series.

Seasonality

To address seasonality, we first need to introduce the idea of ´frequency' of the time series. Unlike the frequency from physics, the ´frequency' in time series is more close to the analogy ´period' in physics. In time series, frequency represents the number of the observations before the seasonality pattern repeats. If it is not pre-defined by the use case, the frequency of a time series, proposed by Hyndman [35], can be explained as shown in table 2.2 2.2.

Series Type	Frequency
Yearly	1
Quarterly	4
Monthly	12
Weekly	52

Table 2.2: Frequency of the time series

The seasonal pattern in the time series is primarily decided by the type of the series (e.g yearly, monthly, etc.). It can be explained as a fixed frequency in the time

series data, and repeatedly occurs across the entire series. In figure 2.1, we show a the *seasonality* of an NN5 series.

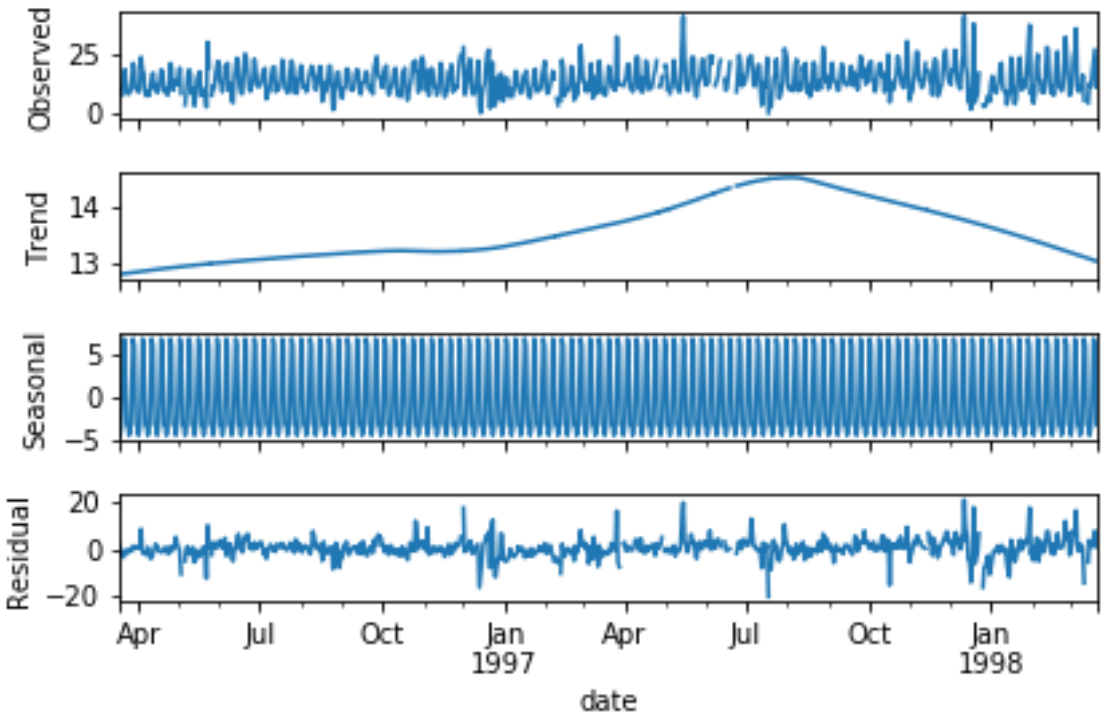


Figure 2.1: The trend, seasonality of a NN5 series

2.1.2 Time Series Decomposition

The variety of patterns exhibit from time series can help improve the forecasting result, as a consequence, it is helpful to split a time series into several components that represent pattern categories.

Simple Decomposition

There are usually two ways of decomposing the time series. Based on the nature of time series combinations, we can use additive decomposition and multiplicative decomposition. Let y_t represent the time series data, S_t is the seasonality component, T_t is the trend of the series, R_t is the remainder component, at time

t, an additive series y_t can be expressed as:

$$y_t = s_t + T_t + R_t \tag{2.1}$$

Alternatively, the multiplicative series is:

$$y_t = s_t * T_t * R_t \tag{2.2}$$

Actually, multiplicative decomposition can be regarded as additive decomposition, once we apply the log transformation:

$$\log(y_t) = \log(s_t * T_t * R_t) = \log(s_t) + \log(T_t) + \log(R_t) \tag{2.3}$$

STL Decomposition

Apart from The simple decomposition discussed above, there are many other decomposition methods. One versatile and robust method for Time Series Decomposition is the Seasonal and Trend decomposition using Loess (STL) method proposed by Cleveland et al. [5]. The term ´loess' stands for local regression smoothing [12]. The benefits from this method are that it can handle any type of seasonality, and is robust to outliers [35].

2.1.3 Simple Statistical Forecasting methods

In Time Series Forecasting, sometimes simple methods are surprisingly effective [35]. As a matter of fact, many of these simple methods are used as benchmarks in the Time Series Forecasting research [6, 42–44]. We will introduce two mostly used simple methods, as they also serve as baseline models in this thesis.

Naïve method

Given the historical observation $y_1, ..., y_t$. The Naïve forecast for horizon h is defined as:

$$\hat{y}_{t+h|t} = y_t \tag{2.4}$$

An example of Naïve forecasting can be found in figure 2.2.

This method is always chosen as the benchmark baseline. Because it produces the forecasting by using the last observation from history. As a consequence, any forecasting methods will usually be compared to this method to validate the effectiveness of the forecasting. If the new forecasting method has worse performance compared to this model, then the new forecasting method is not worth considering. This feature has been using in MASE [34] metric, which is used as one of the evaluation metrics in this thesis.

Seasonal Naïve method

When we encounter the data with seasonality, a similar approach can be used for seasonality forecasting. The idea is simply use the value from the last seasonal period in the history as the future prediction. Given the historical observation $y_1, ..., y_t$, and seasonal frequency f, the forecast for horizon h starting from time t can be expressed as:

$$\hat{y}_{t+h|t} = y_{t+h-f*\lfloor \frac{h-1}{f}+1 \rfloor} \tag{2.5}$$

An example of Seasonal Naïve forecasting can be found in figure 2.2.

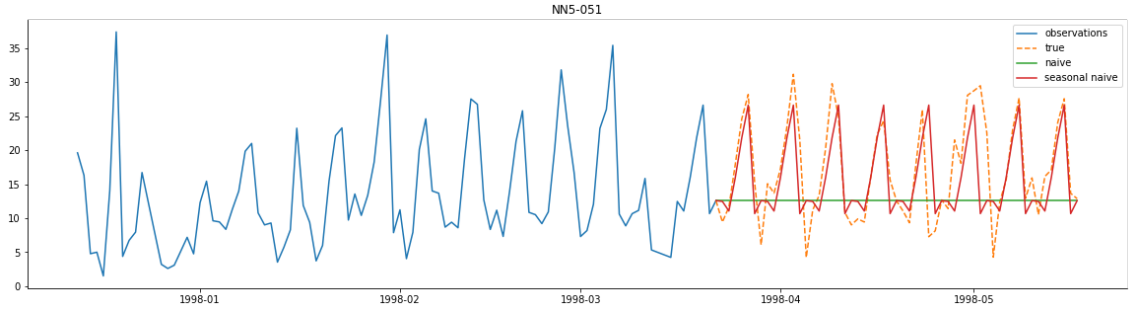


Figure 2.2: Forecasting a series from NN5 dataset with Naïve and Seasonal Naïve method

2.1.4 Exponential Smoothing method

Exponential smoothing was proposed by Brown [8] in 1959, then further developed by Holt [33] and Winters [64]. It has been regarded as one of the most successful forecasting methods. The idea of Exponential Smoothing is to produce weighted averages of historical observations as forecasting results. However, the weights are decaying exponentially for the older observations, resulting in more recent observations having a higher weight, which has a larger impact on final forecasting. The benefit of this method is that it has the capability to generate reliable forecasts for a wide range of time series efficiently. Therefore, it has been widely used in the industry [35]. We will discuss Holt-Winters method as it is used in this thesis.

Holt-Winters method adds the ability to handle the seasonality and trend component to the original Exponential Smoothing method aforementioned. For additive and multiplicative series, there are two variants of the method. The additive method is defined as:

$$s_0 = y'_0 \tag{2.6}$$

$$l_t = \alpha \frac{y'_t}{s_t} + (1 - \alpha) * l_{t-1} \tag{2.7}$$

$$s_{t+f} = \beta \frac{y'_t}{l_t} + (1 - \beta) * s_t \tag{2.8}$$

Alternatively, the multiplicative method can be written into the format of the additive method with log transformation, therefore we will not address it explicitly.

l_t is the level of the observation at time t, s_t is the seasonality of the observation at time t, α and β are smoothing coefficients, f is the frequency of the seasonality. The smoothing level is controlled by α and β, which both in $[0, 1]$. When both α and β equal to 1, it means the result will just be the current observation, if both α and β equal to 0, it means the forecasting metric will follow a naive approach by using the previous observation for the next prediction. Usually, these two parameters need to be tuned in order to achieve the best forecasting.

2.1.5 Machine Learning method

Machine Learning (ML) methods have been gaining prominence over time as interest in AI has been rising [45]. As a consequence, there are many studies on applying Machine Learning method to Time Series Forecasting problem [21, 28, 37, 53, 61]. One of the most comprehensive research was on validating machine learning method for Time Series Forecasting problem was conducted by Makridakis et al. in the extension study of the M3 competition result. They evaluated the performance across multiple forecasting methods, including classical machine learning methods like MLP, KNN, BNN, CART, RNN, LSTM, etc. using the monthly time series from M3 Competition data. The results show that the accuracy of ML models is lower to the statistical methods. However, they still pointed out that there was a great potential of ML methods for forecasting applications if the ML methods can obtain the time series information and enable the automated preprocessing [43].

Later, in M4 Competition, S.Smyl et al. proposed a time series-specific ML method. It combined the Exponential Smoothing (ES) with RNN model, allowing ES to handle the time series components and RNN learn the temporal dependencies. As both ES and RNN parameters are fitting into the same gradient

descent function, this hybrid model is specialized for handling the time series. It achieved the best score among all submitted solutions by beating other statistical methods [44]. Encouraging by the result, we decided to use the similar approach to achieve a Time Series-specific ML model in this thesis.

2.2 Artificial Neural Network

Artificial Neural Networks (ANNs) are information-processing techniques, which are used to discover the knowledge, patterns and unknown structures from the data. The mechanism of ANNs is inspired by the structure of the biological neural network. It uses a series of interconnected processing units, which are called artificial neurons, to calculate received inputs and produces the result of a weighted sum of inputs. The result will be then fed to a non-linear function called *activation function* to generate the final result. The classical artificial neuron is called *Perceptron*[46], as a *supervised learning* algorithm for binary classifiers shown in figure 2.3

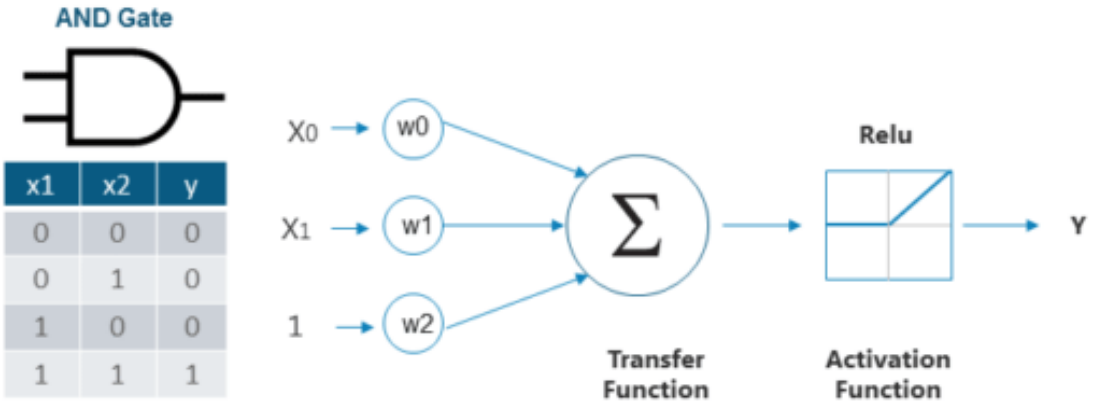


Figure 2.3: A Perceptron, Source:`https://www.edureka.co/blog/perceptron-learning-algorithm/`

The simplest ANNs is called feedforward neural network (FFNN) or a multi-layer perceptron (MLP). It contains multiple perceptrons, targeting to approximate a function $f*$ from the known data x and the target data y. The mapping is defined as $y = f*(x, w)$, and the main purpose of the learning on the mapping is to learn on weights w to obtain the best function approximation. The layers in Neural

Network (NN) are the mapping functions. The number of stacked layers in a Neural Network defines how ´deep' the model will be. The term *Deep Learning* is defined by the number of the stacked hidden layers of the model. Given n layers, the mapping of the NN is then expressed as:

$$f(x) = f^n(f^{n-1}...(f^{1(x)})...)) \tag{2.9}$$

Training a machine learning model means we need to make the output of the model $f(x)$ as close as the real target values. Therefore, the training data and test data is used during the process. We defined a pair of values (x, y), telling the model to generated y based on the given x value.

However, there is still a need for a mechanism that can help the model to decide wellness for their calculation, which is the difference between the calculated value $\hat{(}y)$ and real value y. As a consequence, we need to use a *loss function* to quantify this difference across the training process to tell the model the performance of its learning. Usually, the Mean Square Error (MSE) is used:

$$L(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2 \tag{2.10}$$

The objective of the training is to reduce the error between the model estimation $\hat{y}$ and the real value y, which is continuously minimizing the loss function L. As a consequence, we used an optimization procedure to reduce the error. The common optimization approach is the *Stochastic Gradient Descent* (SGD). SGD constantly evaluates the gradient of the loss function regarding each associated weights w, yielding the information to tell the model to reduce the difference between model estimation $\hat{y}$ and the real value y [3]

2.2.1 Recurrent Neural Network

Recurrent Neural Network (RNN) is designed to process the data in the sequential form. It has the ability to handle the temporal or sequential dependency of the data. The RNN has the input layer and output layer as the conventional Neural Network models. Its hidden layers are composed of recurrently connects cells,

which are usually referred to as *memory states*. This structure helps the RNN model preserve the sequential information and maintain the knowledge acquired. The RNN uses the output from the previous time step during training, therefore, it retains the historical information. A simple demonstration of the model is shown in figure 2.4 by Goodfellow et al. [23].

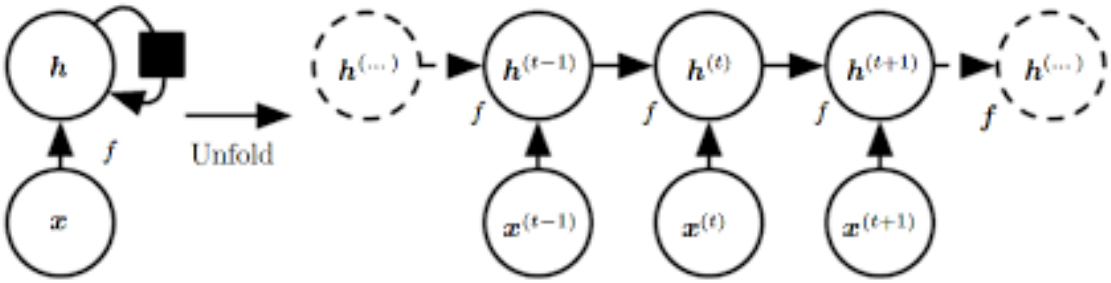


Figure 2.4: An unrolled recurrent neural network in time. The black square on the left implies the time delay. Source: Goodfellow et al. [23]

Particularly, h^t represents the *memory state* of the network at time t, which are calculated based on the current input x^t and previous state $h^{(t-1)}$. In sum, the knowledge acquired from the network will be accumulated. Given an output at time $t-1$ as $h^{(t-1)}$ and input at time t as x^t. The output o at time t is:

$$h^t = f(Ux^t, Wh^{(t-1)}) \tag{2.11}$$

$$o^t = softmax(Vh^t) \tag{2.12}$$

A further demonstration of this process is shown in figure 2.5.

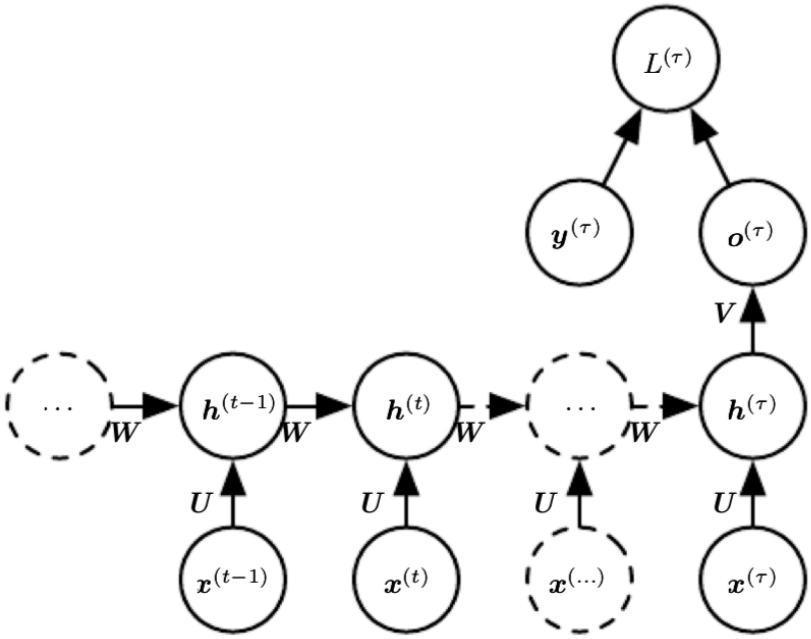


Figure 2.5: A recurrent neural network with output. Source: Goodfellow et al. [23]

f is a non-linear function. Normally $Tanh$ or $RectifiedLinearUnits(ReLU)$ is chosen for f. The final output o^t is regarded as the prediction result, and it will be fit into the loss function L together with the target value for the purpose of computing the gradient. RNN usually uses back-propagation through time (BPTT) to learn the unfolded graph [63].

Although RNN has the capability of capturing the temporal dependence in sequential data, it has a disadvantage when facing a long-term past observation. As the length of the sequence grows, the gradient decays exponentially when learning through time, resulting in little information retained by the RNNs. This drawback will lead to the fact that RNNs will fail to capture the long-term dependency.

2.2.2 Long Short-Term Memory

The introduction of Long Short-Term Memory Networks (LSTM) is targeting to tackle the vanishing gradient problem of RNNs when processing long-term temporal dependencies [32]. The LSTM inherently extend the architecture of

RNN. Additionally, it has a standalone memory cell and a series of gates that manipulates the information flow across the network. There are three gates working collaboratively, namely:

- Input Gate: It updates the cell state. The previous state and current input will be passed to the gate. The gate will then decide which information needs to be updated. The sigmoid function is used in this gate to decide the importance of the information update.
- Forget Gate: It decides if the information should be forgotten or kept. Information from the previous state and current input will be pass through this gate. The sigmoid function is used to output the decision of the gate.
- Output Gate: It outputs the result of the cell calculation. It utilizes the results from the input gate and forget gate, then update the cell state to new values. Eventually, we will obtain a new cell state.

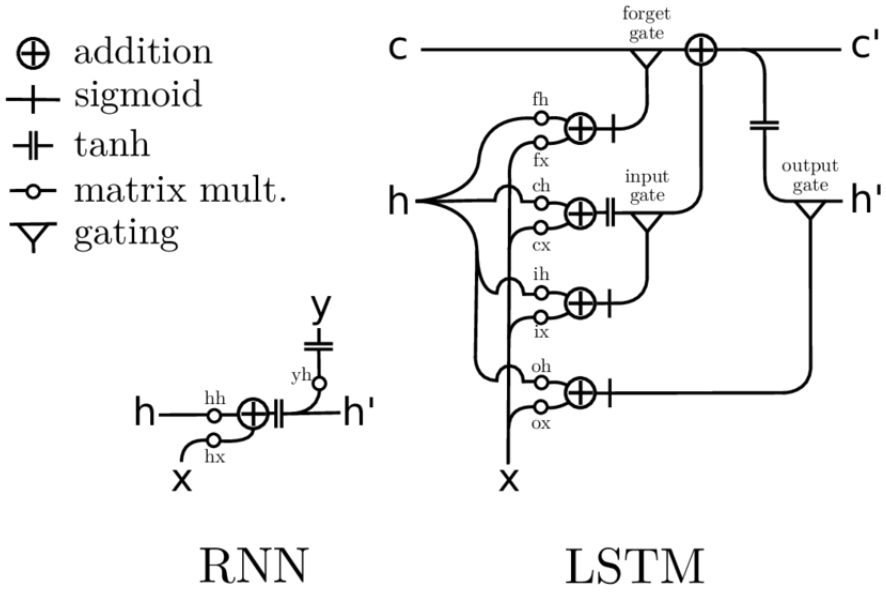


Figure 2.6: A basic RNN cell and a basic LSTM cell Source: M. Stollenga. [57]

A schematic representation of LSTM is shown in figure 2.6. Given a simple LSTM cell with input gate (i), forget gate (f), output gate (o) and memory cell (c), the

input at time t as x_t, the output of LSTM at time t h_t can be calculated as:

$$i_t = \sigma(U_i \cdot x_t + W_i \cdot h_{t-1} + bias) \tag{2.13}$$

$$f_t = \sigma(U_f \cdot x_t + W_f \cdot h_{t-1} + bias) \tag{2.14}$$

$$\tilde{c_t} = tanh(U_{\tilde{c_t}} \cdot x_t + W_{\tilde{c_t}} \cdot h_{t-1} + bias) \tag{2.15}$$

$$c_t = \tilde{c_t} \odot i_t + c_{t-1} \odot f_t \tag{2.16}$$

$$o_t = \sigma(U_o \cdot x_t + W_o \cdot h_{t-1} + bias) \tag{2.17}$$

$$h_t = o_t \odot tanh(c_t) \tag{2.18}$$

where W_i, W_o, W_f, W_c represents the weight matrices of input gate, output gate, forget gate, and memory cell respectively. Likewise, U_i, U_o, U_f, U_c are the corresponding input weight matrices. $\odot$ is the element wise multiplication. σ is the standard sigmoid function.

From the figure 2.6, we can see how the gates work to control the information flow through the LSTM cell. The forget gate that connects the current state c and next state c' decides if the information should be forgotten or not. Once it is on, the information can be propagated to the next time step directly. The input gate controls if the cell needs to be updated. Once it is on, the new information will be applied to the cell, thus the cell state will be updated. The output gate controls the output of the cell. The collaboration of the gates helps LSTM control the information flow, meanwhile retain the information across the previous time steps.

2.3 Distributed Deep Learning

Deep Neural Networks (DNNs) models trained on a large dataset can produce impressive performance, and it has benefited the domains such as speech recognition [15, 54], object detection [11, 13], and natural language processing [14, 18]. One can observe that, with more data involved in the training process, and increasing complexity of the DNN models, training a DNN model can be expensive in terms of the computational demand. This problem becomes more significant for deploying the model on a commodity device. As a consequence,

research has been conducted on exploring distributed deep learning. There are two most common approaches to achieve the distributed deep learning, namely Data Parallelism and Model Parallelism.

Data Parallelism

Data parallelism, as shown in figure 2.7, is achieved by data partitioning. First, the data will be split into a series of partitions, then these partitions will be distributed to the number of worker nodes. Subsequently, each worker will perform computation on its own data and generate a set of parameters. All nodes will synchronize their parameter states via network communication. These procedures will repeat until the model converges. Many large-scale computational tools such as MapReduce [16] and GraphLab [39] use a similar mechanism in the parallel data processing. Notably, the idea of local training and aggregate weight updates is similar to the Federated Learning paradigm.

Model Parallelism

Unlike data parallelism, model parallelism, as shown in figure 2.7, partitions the DNN model to distribute the workload to multiple computational worker nodes. The different parts of the model (e.g. layers) will be distributed to each worker nodes as each of the workers will contribute partially to the final parameter update. As a consequence, the different sectors of the model will be computed separately by each worker. Then all computed outputs at the worker side will be aggregated via network communication to the central node, where the model will be updated. A notable example of using model parallelism to achieve distributed learning is Google's *DistBelief* [17].

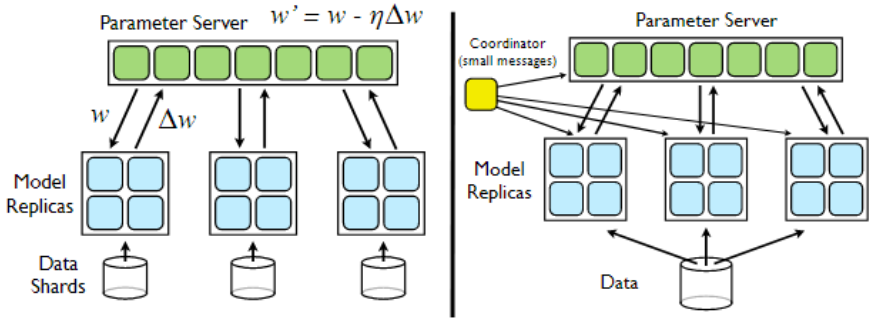


Figure 2.7: A comparison of Data Parallelism and Model Parallelism. Source: Dean et al.[17]

2.3.1 Federated Learning

Federated Learning avoids centrally storing the data, instead, it allows user training a shared model collaboratively by using its local data [47], therefore, it guarantees data privacy and decreases the communication cost. In general, the learning task is achieved by a loose federation of participants (clients) supervised by a central coordinator (server). The local data at the client side will never be uploaded to the server, alternately, the clients compute the updates of the shared model maintained by the server, and only this update is communicated. The idea is similar to the Data-Parallelism from chapter 2.3, however, the advantage of federated learning is that it decouples the model training from the demand of accessing to the training data, which means the data is parallel naturally thus no partition operation is needed.

Federated learning has distinct advantages when the datasets are distributed on a series of devices and data privacy is sensitive or large in size [47]. However, in order to make federated learning practical, the optimization problem in the context of federated learning need to be solved. Unlike the typically distributed optimization problem, federated optimization has several unique properties:

- Non-IID: The training data at client side will not be representative of the population distribution.
- Unbalanced: Varying amounts of local training data due to different user

data.

- Massively distributed: The number of clients can be much bigger than the average number of training data stored on a given client node.
- Limited communication: Client(s) may be offline or on slow or expensive connections.

Naively, SGD can be applied to the federated optimization problem. The single batching gradient calculation can be achieved by a communication round in federated network [47]. The process starts selecting K clients from the federated network, then these K clients compute the updates using their local datasets respectively. Subsequently, the updates are aggregated by computing a weighted average. This algorithm is called *Federated Averaging*, shown in algorithm 1.

Algorithm 1 Federated Averaging. The K clients are indexed by k, B is the local mini-batch size, E is the number of local epochs, and η is the learning rate

Server executes:

initialize w_0

for each round t = 1, 2,... **do**

$m \leftarrow max(\mathrm{C}\cdot\mathrm{K}, 1)$

$S_t \leftarrow$ (random set of m clients)

for each client $k \in S_t$ in parallel **do**

$w_{t+1}^k \leftarrow$ ClientUpdate(k,w_t)

end for

$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

end for

ClientUpdate(k,w):

$B \leftarrow$ (split P_k into batch of size B)

for each local epoch i from 1 to E **do**

for batch $b \in B$ **do**

$w \leftarrow w - \eta \nabla l(w, b)$

end for

end for

return w to server

K, B, E stand for the number of clients, local mini-batch size, local epochs respectively. These are the hyper-parameters of the Federated network, which are not directly learned from the training data. They usually depend on the network setup and problems. Notably, these parameters also have impacts on the Federated Learning [47]. For instance, increasing K will increase the speed of the convergence of the network and reduce the communication rounds. However, increasing K also means the cost per communication round will increase. There is a clear trade-off between computational cost and convergence speed.

3 Methodology

In this chapter, we will describe the strategy to answer the research question formulated in Chapter 1.3.

As discussed in Chapter 1.2, in order to fulfill our research purpose, we need to tackle the problem of using conventional Neural Network to achieve a reliable Time Series Forecasting by proposing a hybrid model, then we can evaluate and analyze the idea of Federated Time Series Forecasting on the proposed model.

3.1 Research Methods

In order to answer the research question, we need a strategy to implement all necessary steps to achieve the objective. Particularly, A. Håkansson provides a guide to show the difference in the methods mostly used in information and communication technology research [27]. We will follow this guide to choose the most suitable method for our study, which is to answer the research question:

Can we realize decentralized time series forecasting with a Federated Learning mechanism that is comparable to the conventional centralized setup in forecasting performance?

that mainly concerns two aspects 1). if proposed forecasting model is effective, and 2). if the proposed Federated Time Series Forecasting approach has a comparable performance against data-centralized setup.

To answer the first aspect of the research question, we reflect on chapter 1.2, the disadvantage of using the Neural Network (NN) based model for TSF problem is because they fail to explain the time series specific factors like seasonality and trend. As a consequence, S.Smyl proposed a hybrid model structure to tackle

this issue for using NN models in TSF problems. As our work is based on the result of S.Smyl [56], the *fundamental* and *applied* method scheme does not fit in the context since there is no new theory established in this domain. Also, we are not studying any causes and effects, instead, the main focus for this task is to study the performance of a model, which is the accuracy of the proposed hybrid model. Therefore, to address this aspect, it is natural to choose the *empirical* method.

For the same reason, *empirical* method will be applied to validate the Federated settings for scalable Time Series Forecasting problems. The choice of the method, inherently, was used by McMahan et al. in the paper of Federated Learning [47]. The paper showed the effectiveness and practicality of Federated Learning by conducting a variety of experiments. Furthermore, Konecny et al. provided an optimized Federated Averaging Algorithm that can handle the issue when data is unevenly distributed over an extremely large number of nodes [38], and the authors validated the idea based on the experiments on *Google+ post* dataset. This work on validating the effectiveness of the Federated Learning framework encourages us to use the same approach for our research question.

3.2 Data collection

Given the fact that the research will be conducted with the accordance with the scheme of *empirical* method, we will draw the conclusion based on the quantification of the experiment results. This requires us to find the adequate dataset for the experiments. We choose the *experiment* data collection as our data collection method as there are many datasets available publicly, and in particular, we have the dataset from Ericsson that was collected in particular for the purpose of carrying out our research. The selected datasets are further described in section 3.5.

Notably, there will be data generated by our experiments. The datasets generated from the experiments are forecasting results for a certain prediction horizon defined in experiment settings. These datasets will be further analyzed and quantified in order to help us evaluate the performance. The final conclusion will

be established based on these results.

3.3 Data Analysis

After obtaining the data from the previous step, we need to analyze the data to see if collected data is sufficient for the experiment purposes. Also, the appropriate analysis metrics need to be chosen for the data generated from the experiment. These aspects will be addressed in this section.

3.3.1 Analysis on collected data

For the Time Series Forecasting problem, missing values especially consecutive missing values are critical factors that can impact on the forecasting performance by corrupting the repetitive information (seasonality), as well as introducing difficulties for conventional statistical approaches that are based on autoregression [51]. To tackle this issue, we will choose *computational mathematics* methods to identify the missing values in the dataset. If there are more than 50% missing value existing in a time series, we will drop it as it will not contribute to our study as the majority of the observation is missing. For those series with less than 50% missing values, we will impute the missing values by linear interpolation. Although some autoregressive based methods show more reliability than linear interpolation, however, the improvement comes at a cost of computation complexity with the increasing size of the dataset. For simplicity, we decided to use the basic linear interpolation for the imputation.

3.3.2 Evaluating the effectiveness of the forecasting model

To evaluate the effectiveness of the forecasting model, we need to access the forecasting performance of the model and baseline models. We decided to use *statistical* method. Our choice is backed up by similar studies from Makridakis et al. on M3-competition [42, 43] and proceeding on M4-competition[44]. The purpose of both studies was aimed to analyze the submitted solutions for M3 and M4 Time Series Forecasting problems [41]. Also, in an extended work, they

investigated if machine learning models can be used for Time Series Forecasting by testing a variety of machine learning models and statistical models, then they drew the conclusion based on the performances of these models. Our proposal for evaluating the effectiveness of the forecasting model shares a similar context and will therefore will conduct the same strategy.

Naturally, as we are going to measure the performance of the forecasting model, we need metrics that can help us to quantify the performance in order to generate an interpretable result for our decision-making. One of the most used metric for Time Series Forecasting particularly is the symmetric Mean Absolute Percentage Error (sMAPE)[55]:

$$sMAPE = \frac{2}{k} \sum_{t=1}^{k} \frac{|Y_t - \hat{Y}_t|}{|Y_t| + |\hat{Y}_t|} * 100\% \tag{3.1}$$

where k is the forecasting horizon, Y_t is the actual observation and $\hat{Y}_t$ is the forecasts produced by the model at time t. Differing from its name, sMAPE does not handle large positive errors and negative errors equally, as there is more penalization on large positive errors. As a consequence, another evaluation metric, Mean Absolute Scaled Error (MASE)[34], is usually used in parallel with the sMAPE to avoid potential issues caused by the imbalanced penalization. MASE is defined as:

$$MASE = \frac{\frac{1}{k} \sum_{t=1}^{k} |Y_t - \hat{Y}_t|}{\frac{1}{n-m} \sum_{t=m+1}^{n} |Y_t - Y_{t-m}|} \tag{3.2}$$

where k is the forecasting horizon, Y_t is the actual observation, $\hat{Y}_t$ is the forecasts produced by the model at time t, n is the number of available observations, and m is the frequency of the time series data.

The purpose of MASE is to explain the forecasting performance with respect to the Naive forecast. Generally, the model can be expressed as Mean Absolute Error of Prediction on a given time series divided by Mean Absolute Error of Naive Prediction on given time series without the last seasonal period. The naive prediction is defined by using previous observation value for the upcoming forecasting result [35]. Essentially, the MASE shows the quantitative relation

between the actual model forecasting and naive forecasting. One can clearly see that if the final MASE score is smaller than 1, it means on average the proposed model provides smaller forecasting error comparing to the naive method. Contrarily, a score greater than 1 means on average the actual forecasting is worse. When the score equals 1, it means the actual model as the same performance as a naive approach. The application of this metric can provide instant information to decide the validity of the forecasting model.

The general strategy to utilize the benefits of both sMAPE and MASE metric is that we first use the MASE score to check the validity of the forecasting model, then use the sMAPE to decide which model has a better forecasting performance. By comparing these two metrics results of the proposed model against the baseline models, we can see if the proposed model is effective.

3.3.3 Evaluating Federated Time Series Forecasting

We use the *statistical* method for evaluation purposes. The reason is identical to our choice for evaluating the effectiveness of the forecasting model. Again, McMahan et al. performed the studies on MNIST dataset and CIFAR10 dataset to analyze the performance of baseline and proposed Federated Averaging algorithms. The conclusion was drawn based on the accuracy metric results obtained from the experiments [47]. A similar approach was applied by Konecny et al. to validate the Federated Optimization framework on *Google+ post* dataset [38]. Both of the studies were targeting to evaluate the practicality of Federated Learning specific frameworks. As a consequence, we will apply the same approach to our work.

The core idea of Federated Learning is to achieve better learning at edge device [47]. As a consequence, to assess the performance of the Federated Learning framework, one needs to perform the evaluation on the edge node with the local dataset respectively. Konecny et al. [38] formulated Federated Learning problem as:

Given the Federated network with n distributed nodes. Each node i stores a local dataset X_i. The entire federated dataset is defined as:

$$D = X_1 \cup X_2 \cup ... \cup X_n$$

The local learning objective can be defined by using a performance metric function f with model parameter w_i and dataset X_i, the local performance P of node i is defined as:

$$P_i = f(w_i, X_i) \tag{3.3}$$

Furthermore, the average performance across the entire network is computed by:

$$P = \frac{1}{n} \sum_{i=1}^{n} P_i \tag{3.4}$$

So when we evaluate the performance P of a Federated Learning network, we are using the average performance of all the local nodes.

The metric for evaluation f will be sMAPE and MASE introduced in chapter 3.3.2. and the reason is the same as we have discussed in chapter 3.3.2. Again, by comparing these two metrics results of the federated settings of the model against the data-centralized settings, we can conclude the results are comparable.

3.4 Model Design

Reflecting on the research purpose in chapter 1.3, we need a reliable Neural Network model for the Time Series Forecasting (TSF) problem. Once we verify the robustness of the model, we can then investigating the Federated Time Series Forecasting. Therefore, we need to first design the model that fits the TSF context, then design the Federated framework for examining our hypothesis.

3.4.1 Time Series Forecasting Model

As discussed in chapter 1.1 and chapter 1.2, we see that the Neural Network models are not able to capture and utilize the Time Series specified information (seasonality, trend, level, etc.). Therefore, they are usually outperformed by conventional statistical approaches. A natural way to tackle the problem is to make a Time Series specified Neural Network model. The introduction of Recurrent Neural Network (RNN) and other RNN-based models enable learning on sequences, furthermore, the introduction of Dynamic Computation Graph Neural Network systems (DGNNs) allows building hybrid models that can achieve hierarchically by including additional time series specifications to the Neural Network models. The first implementation of this hybrid ecosystem was from S. Smyl in M4-competition [56], and it achieved the best score among all submitted solutions by beating other statistical methods [44]. Encouraging by this result, we decided to use the same approach to build our Time Series model for this thesis.

Time Series Handling

In general, the hybrid model has two main components. The first part of the model is the sector to extract the seasonality and level information from the time series. Many statistical methods such as ARIMA, STL, Exponential Smoothing can fill the gap. Although other methods like STL may have a better result for the time series decomposition, however, it is more natural that time series decomposition can be integrated into our forecasting algorithm and this tactic was also motivated by S.Smyl [56]. Therefore, we choose the Exponential Smoothing in this thesis.

When starting modeling the time series with Exponential Smoothing, one needs to understand if the given time series has an additive seasonality or multiplicative seasonality. This can be verified by using statistical approaches like autocorrelation function (ACF). However, we decided to apply the log transform on the dataset, thus the seasonality will be transformed into the multiplicative

pattern [35]. So for a observation y_t from the time series, we have:

$$y'_t = log(y_t) \tag{3.5}$$

Then, we start to extract seasonality and level information from the transformed time series with Exponential Smoothing method. The basic Exponential Smoothing can be achieved with Holt-Winter method discussed in Chapter 2. The method is used to smooth the time series data acting as a low-pass filter. In our case, as it is an integration of the Neural Network model, the coefficients of the Exponential Smoothing will be a part of model parameters fitting to the same model optimizer.

Neural Network Architecture

For the Neural Network (NN), we are going to use a 2 layers NN model. The first layer is an LSTM layer. The second layer is a dense layer with output size equals to the forecasting horizon. For the convenience of the performance study, we fixed the model hyperparameters during the experiments. Also, the seeds are applied to eliminate the randomness during the experiment.

Model Parameter	**Value**
Hidden Neurons	50
Batch Size	128
Optimizer	RMSProp
Learning rate	0.001
Gradient Clipping	10

Table 3.1: Hyperparameters of the NN Architecture

Eventually, the forecasting model is expressed by:

$$\hat{y}_{t+1} = LSTM(x_t) * l_t * s_{t+1} \tag{3.6}$$

For forecasting horizon size f, the forecasting can be expressed as:

$$\hat{y}_{t+1\ldots t+f} = LSTM(x_t) * l_t * s_{t+1\ldots t+f} \tag{3.7}$$

x_t is the input vector, which is the pre-processed time series with seasonality and level information removed. From the equation above we can clearly see the structure of the model, which are Exponential Smoothing part for handling seasonality and LSTM part to perform learning on remainders.

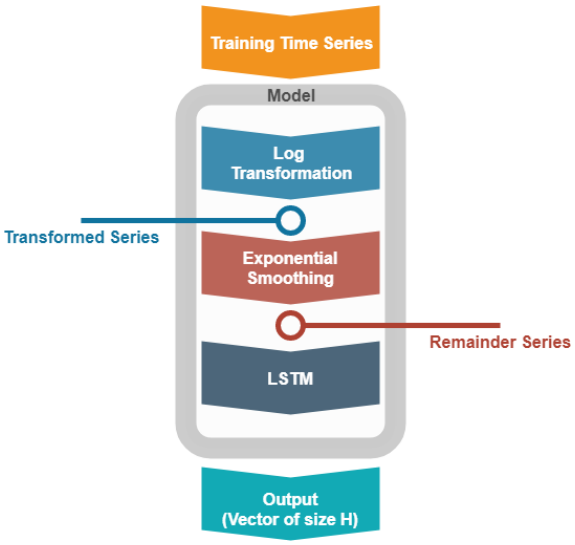


Figure 3.1: An overview of the hybrid model

Moving Window Strategy

Notably, we did not discuss the input size in previous forecasting model design and we did not mention how the model is trained. The purpose of the Neural Network is to find a relation between the input and output. Essentially, as we used the LSTM, which is a Recurrent Neural Network based model. We need to transform the Time Series into a pair of input series and output series. On the other hands, the reason for choosing the Recurrent Neural Network (RNN) based model is because its capability to learn the temporal dependencies. However, the nature of RNN models is using the internal state to keep track of all temporal information. When the dependency is complex, it will make the computation expensive. So feeding the series directly to obtain forecasting results usually comes with a cost,

especially when the series data is big, or when multi-step forecasting is required. MIMO strategy can resolve this tension by creating multiple input and output mappings while preserving the stochastic dependencies between the predictions [2, 59]. The studies from Ben Taieb et al. illustrated the benefits of applying the method in multi-step forecasting [59, 60], and Bandaraa et al. also used this strategy for forecasting with RNN [6]. Therefore, we decided to apply this approach to this thesis.

The idea of MIMO strategy is simple, namely:

Given a time series $ts = [y_1, ..., y_n]$*, the input size* I*, the forecasting horizon size* H*, a mappings* f *of inputs and outputs is defined as:*

$$[y_{t+1+I}, ..., y_{t+H+I}] = f([y_{t+1}, ..., y_{t+I}]) \tag{3.8}$$

where $t \in [0, n - I - H]$.

The forecasting horizon H is defined by the forecasting problem itself. For input size, it is natural to set it as the size of seasonality frequency, at least. The reason behind this choice is that we need to provide the model capability to see a seasonality-size number of past observations during the model training, as the model is still trying to learn how to deseasonalize the data properly.

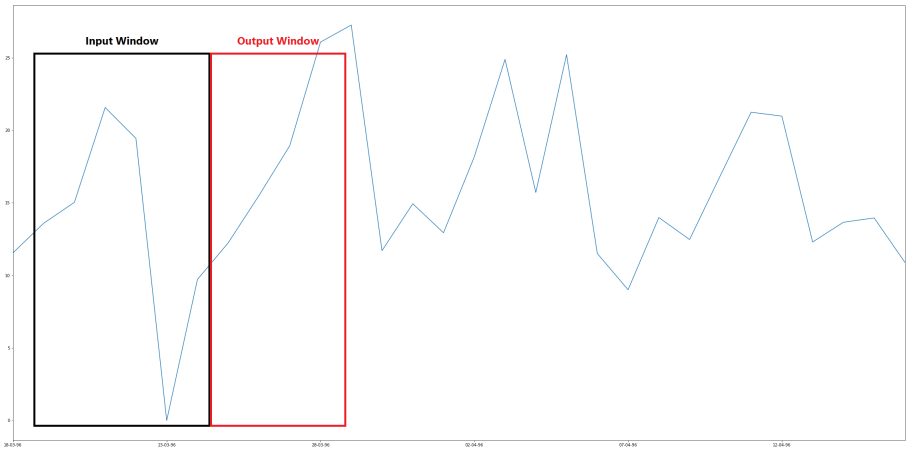


Figure 3.2: An example of applying the moving window approach to the series NN5-001 (partially) of the NN5 dataset

With MIMO strategy, the time series will be partition to the pairs of I-size and H-

size. The model will train on these pairs recursively till the last window, and the last window will be used for the validation purpose.

In addition, one may notice that MIMO has a clear disadvantage to achieve flexible forecasting, as the output window are fixed for the same model. This is not an issue for continuing on our study and it is not a concern for the follow up experiment settings, as all forecasting horizons are fixed. Also, the size of the input will have an impact on the forecasting horizon. With a larger input window, one may learn more sophisticated features. However, taking more data for the input window will also increase the computation complexity (e.g. in the worst case, when the input window contains the entire train set, the model will have the best knowledge but then the MIMO is not needed). There should be a trade-off point that balancing the complexity and features covering, however, we will not investigate this matter due to the time constraint.

3.4.2 Federated Learning Framework

The Federated Learning is achieved by the execution of the *FederatedAveraging* algorithm on the *parameter server* and *client* interactively. An example is shown in figure 3.3.

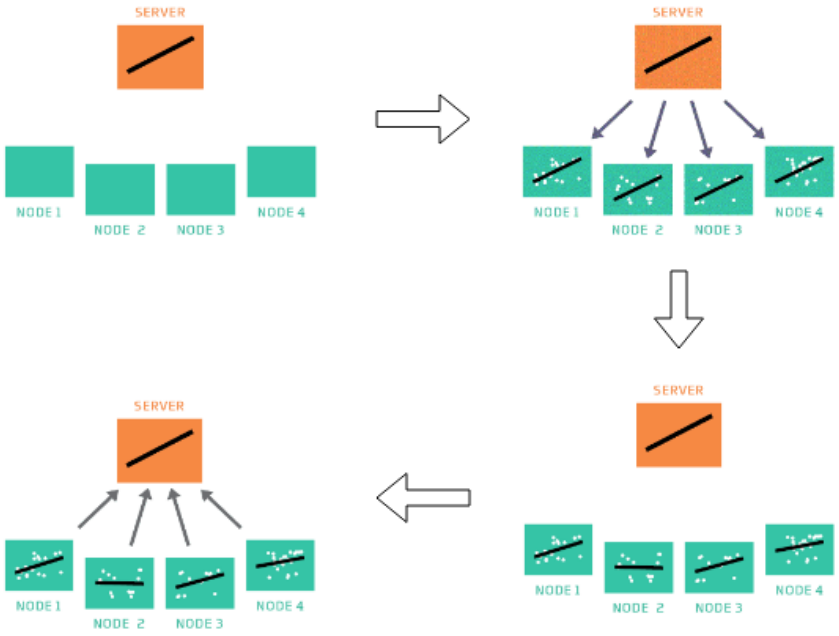


Figure 3.3: An example of one-communication round in Federated Network. Source: Cloudera [62]

A communication round between *parameter server* and *client* can be further motivated as:

1. Server samples m_t out of K clients at around t.
2. Server distributes center model to sampled clients m_t.
3. Clients optimize the model on their data respectively.
4. The update of model parameters are then centralised, averaged, for the purpose of updating central model.

Based on the example, we can further motivate the framework design with a class diagram.

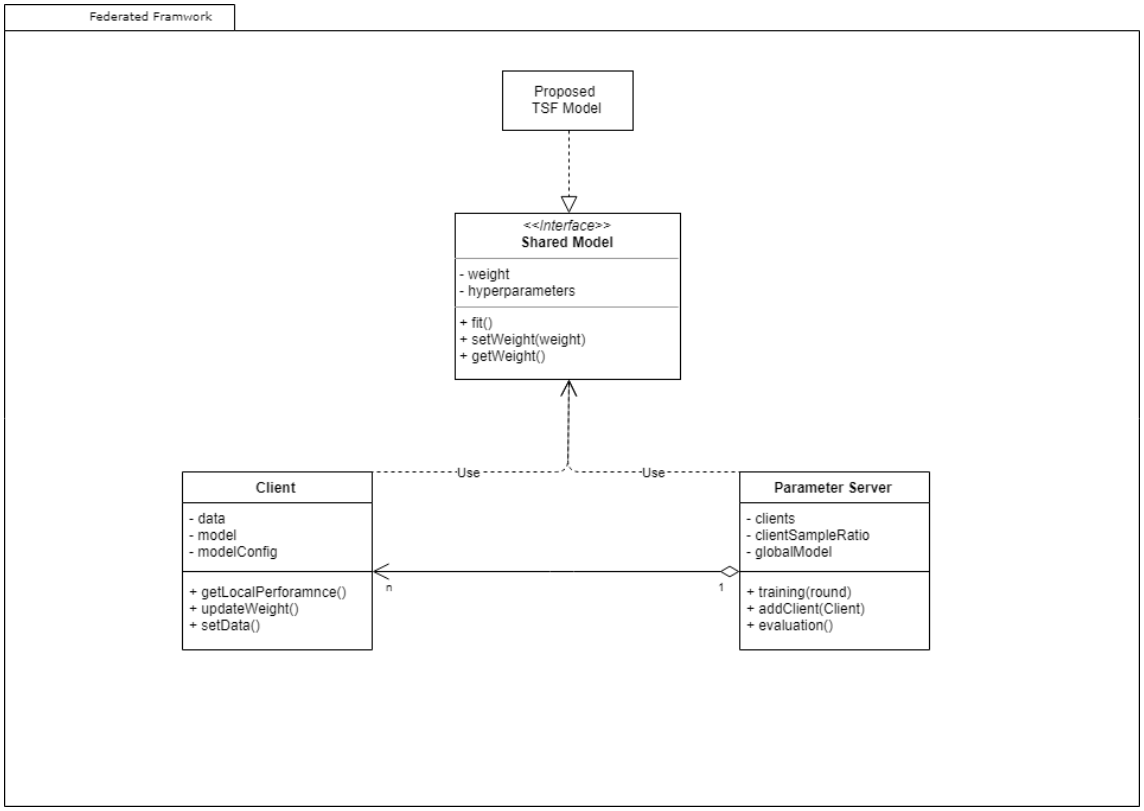


Figure 3.4: Class Diagram of Federated Time Series Forecasting

3.5 Experiment Setup

In this section, we present the setting of the experiment on collected datasets.

3.5.1 Experiment I: Effectiveness of the Forecasting Model

This experiment is targeting to validate the effectiveness of our forecasting model described in Chapter 3.4.1. We will measure the performances of the metric explained in Chapter 3.3.2, and compare them to those of a series of baseline models.

Baseline models

We choose 2 simple and 1 advanced forecasting model as baseline. The simple models are Naïve and Seasonal Naïve. The advanced model is STL decomposition forecasting. We add the simple models as our baseline setup, as sometimes the simple models are effective enough to produce fair forecasting results [35]. On the other hand, the simple models we included are used in many TSF studies[6, 42, 44].

In additional, as NN5 data has been used for many TSF papers, we will also include some model results for comparison, in order to further motivate the effectiveness of our forecasting model.

Datasets

We select 2 Time Series datasets for this experiment. They are:

- NN5 Dataset: The data contains 2 years of daily cash withdrawals at various automatic teller machines (ATMs) located in the UK (Crone, 2008), with 792 observations per series, 111 time series. The task is to predict a horizon length of 56 days ahead using 70 previous observations. The dataset has a clear seasonality pattern, as showed in figure 3.5.

Parameter	Value
Observations per series	792
Number of series	111
Prediction horizon	56
Prediction input	70
Size of training series	792 - 56
Size of test series	56

Table 3.2: NN5 dataset information

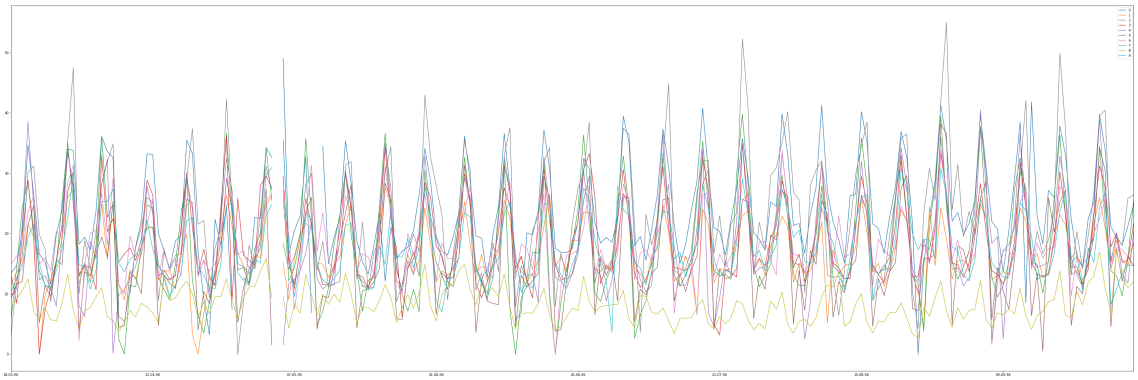


Figure 3.5: Sample of NN5 data: 10 series with 50 observations

- Ericsson KPI dataset: The data contains sensor data from 13 base stations of 105 sensors. Starting from 2016-05-04 00:00:00 to 2016-07-14 23:00:00 hourly. With 1608 obs per series. We use previous 7 days observation to predict next 24 hours. The dataset has a clear seasonality pattern, as showed in figure 3.6

Parameter	**Value**
Observations per series	1608
Number of series	105
Prediction horizon	24 hours
Prediction input	7 days * 24 hours
Size of training series	1608 - 24
Size of test series	24

Table 3.3: Ericsson KPI dataset information

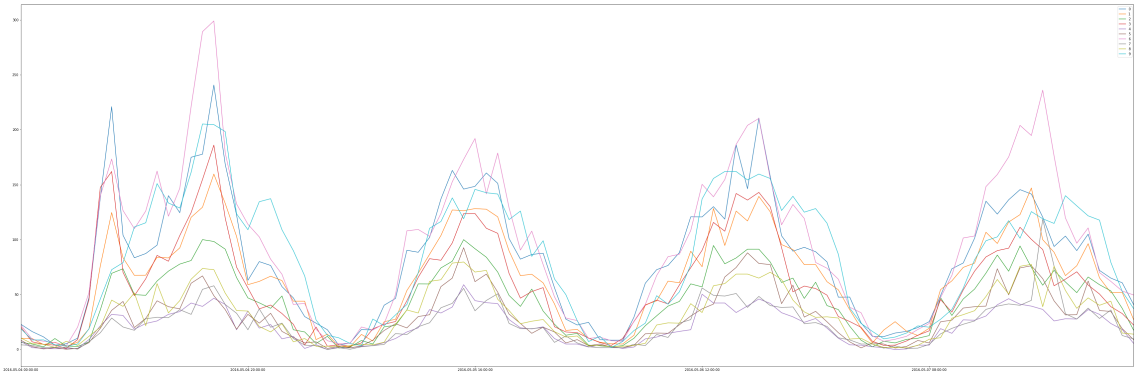


Figure 3.6: Sample of Ericsson KPI data: 10 series with 100 observations

Process

The illustration of the experiment can be found in figure 3.7. Each model, including the baselines and the proposed hybrid model, will make use of the selected dataset for training, then generate the prediction results based on the requirements from table 3.2 and 3.3 respectively. After obtaining the prediction results, we quantify the performance based on the evaluation metric explained in Chapter 3.3.2. Note that the setup of the hybrid model is explained in Chapter 3.4.

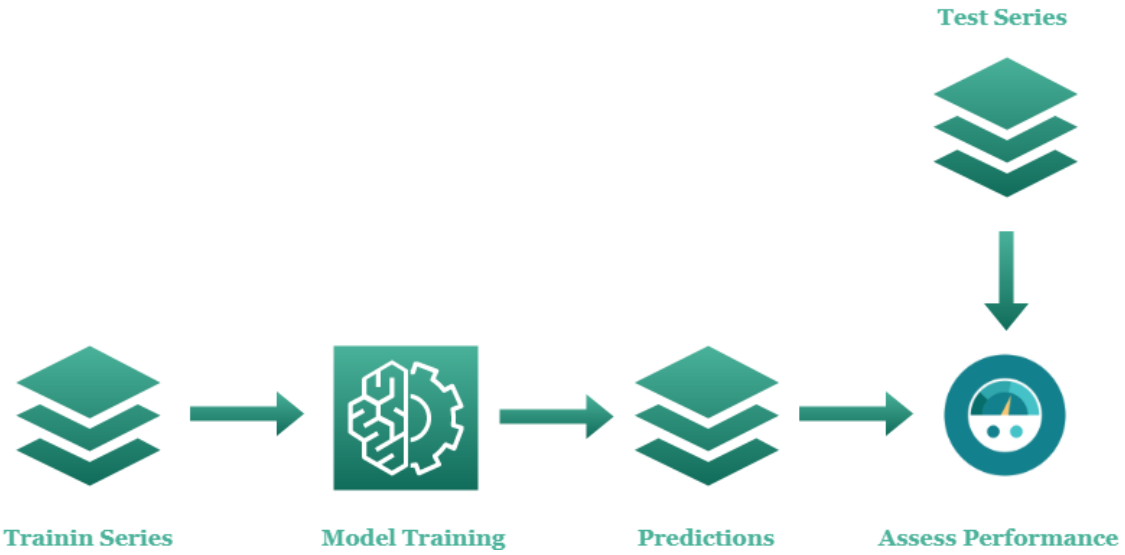


Figure 3.7: An illustration of the experiment process

Evaluation Metrics

This has been discussed in Chapter 3.3.2.

3.5.2 Experiment II: Federated Time Series Forecasting

This experiment is targeting to validate if it is possible to achieve Federated Time Series Forecasting. Meanwhile, the result of the experiment should also be able to address the aspect that if the forecasting result is comparable to the centralized setup.

Datasets

We decided to use the same dataset from Experiment I in Chapter 3.5.1. Experiment I feeds the entire data to the model, so it can serve as an analogy of centralized approach. Furthermore, we can use the forecasting results of Experiment I to answer if the forecasting result is comparable to the centralized setup.

In federated settings, there will be no central-point entry for the entire dataset. The series are distributed to the clients in the network.

Process

The idea and configuration of the Federated framework for this thesis are explained in Chapter 3.4.2.

We deployed the hybrid model at each client node. The model settings are explained in Chapter 3.4. As mentioned, the data will be distributed to the clients respectively. We will let each client have one series (w.r.t. both training series and test series), they will then use their own series for local training and local testing.

For *Federated Averaging* algorithm, we fixed the local epochs E and local batch size B, and the setting for NN5 dataset and Ericsson KPI data can be found in table 3.4 and 3.5. The fraction of clients per round C will be 30%.

The network will be trained for 300 communication rounds to ensure the model convergence. After finalizing the training, the result will be evaluated based on the metric introduced in Chapter 3.3.3.

Evaluation Metrics

This has been discussed in Chapter 3.3.3.

Parameter	Value
Local Epochs E	1
Local Batch size B	64
Communication Rounds r	300
Ratio of selected clients per communication Rounds C	30%

Table 3.4: NN5 case setup for *Federated Averaging* algorithm

Parameter	Value
Local Epochs E	4
Local Batch size B	128
Communication Rounds r	300
Ratio of selected clients per communication Rounds C	30%

Table 3.5: Ericsson KPI case setup for *Federated Averaging* algorithm

4 Result

This chapter presents the experiments results.

4.1 Result of Experiment I

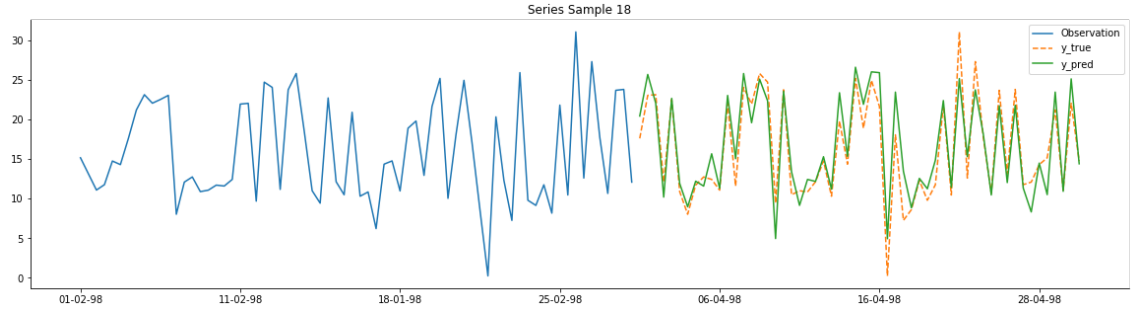
NN5 Case

The NN5 results of Experiment I are shown in table 4.1

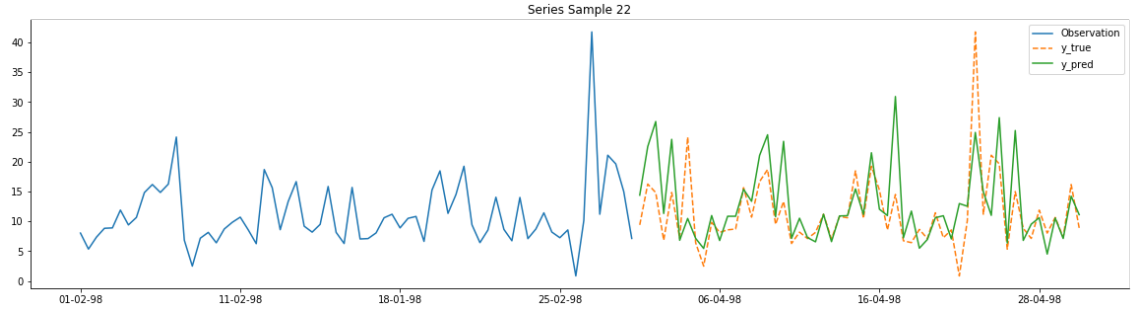
NN5 Case	Mean sMAPE	Mean MASE
Proposed Model	**20.20**	**0.57**
Hybrid ESDRNN (S.Smyl [56])	21.57	0.64
Combined STL+LSTM (Bandara [6])	23.46	0.96
ES (Bandara [6])	21.44	0.86
ARIMA (Bandara [6])	25.29	0.97
STL Seasonal	23.80	0.94
Naive Seasonal	28.13	1.04
Naive	48.26	1.83

Table 4.1: Results of NN5 dataset

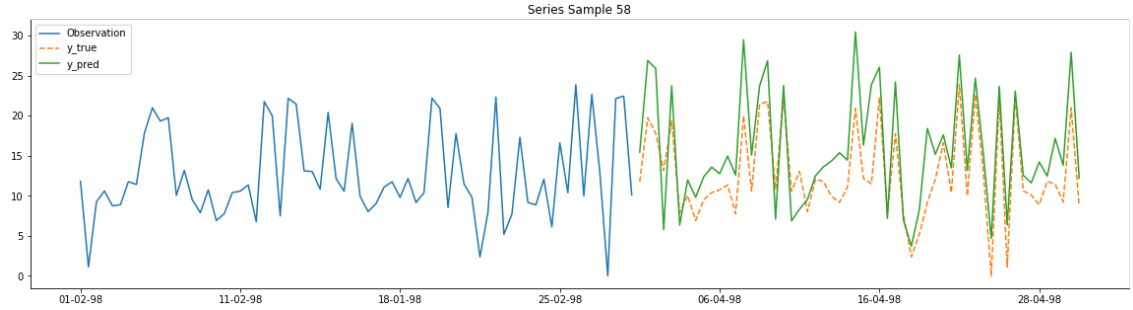
Table 4.1 shows the results of various models for the NN5 data forecasting. It can be seen that the proposed model performs better than other models listed in the table, include the state-of-art methods like Exponential Smoothing, ARIMA, in terms of Mean sMAPE and Mean MAPE on average. In figure 4.3, we show some predictions made by proposed model.



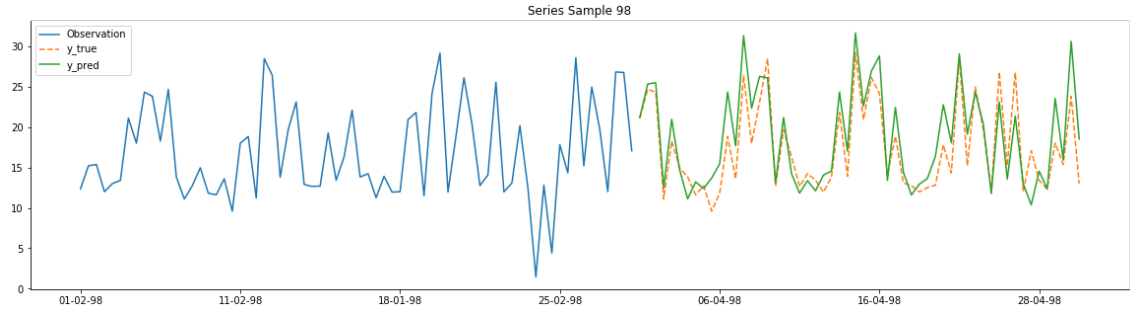
(a) NN5 Sample Series 18



(b) NN5 Sample Series 22

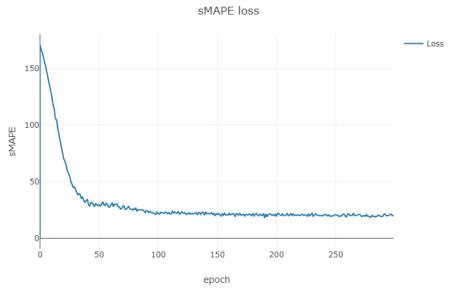


(c) NN5 Sample Series 58

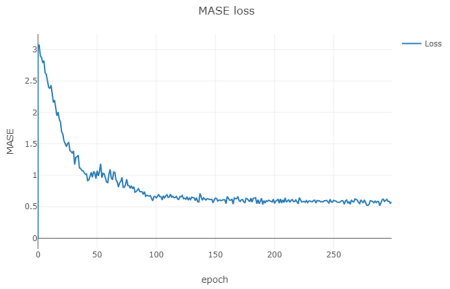


(d) NN5 Sample Series 98

Figure 4.1: Some forecasting results from NN5, blue line shows the last 50 observation of each series, green line represents the true value, and orange dash line is the prediction made by the proposed model.



(a) sMAPE vs epoch training on NN5 dataset



(b) MASE vs epoch training on NN5 dataset

Figure 4.2: sMAPE and MASE vs epoch training on NN5 dataset

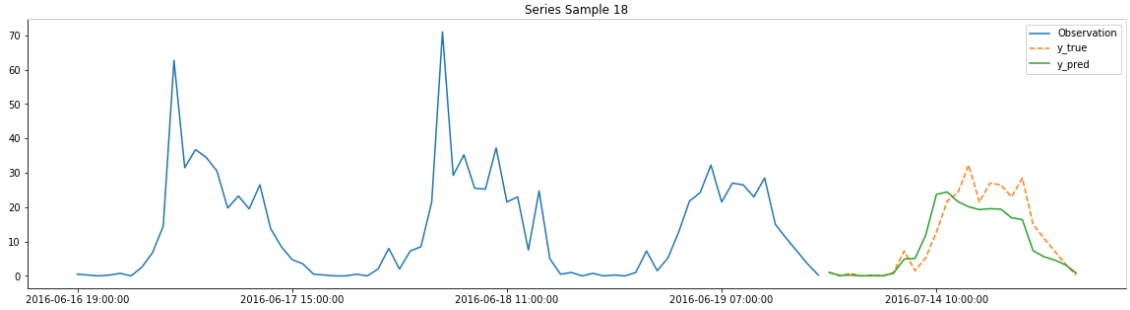
Ericsson KPI Case

The Ericsson KPI results of Experiment I are shown in table 4.2

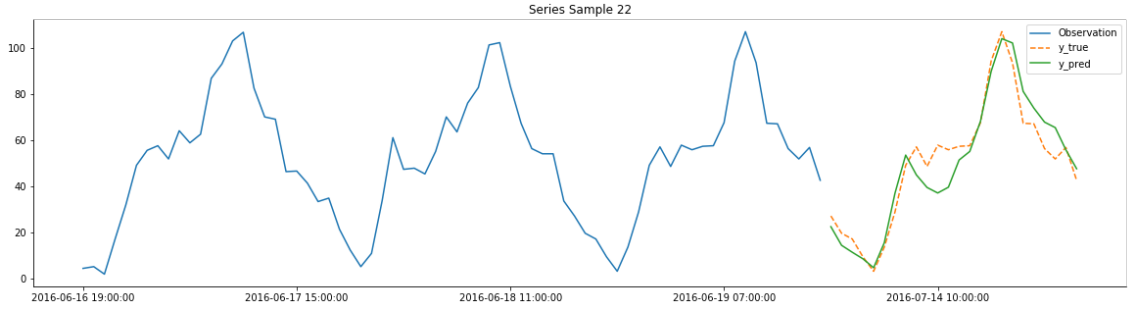
Ericsson KPI Case	**Mean sMAPE**	**Mean MAPE**
Proposed Model	**27.90**	**0.28**
Hybrid ESDRNN (S.Smyl [56])	28.07	**0.28**
STL Seasonal	42.57	0.52
Naive Seasonal	36.47	0.53
Naive	74.54	1.29

Table 4.2: Results of Ericsson KPI dataset

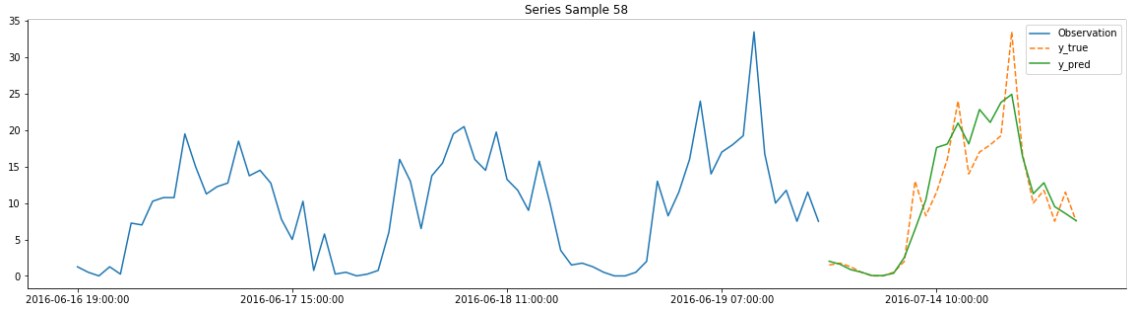
Table 4.2 shows the results of various models for the Ericsson KPI data forecasting. It can be seen that the proposed model performs better than most of the other models listed in the table in terms of Mean sMAPE and Mean MAPE on average. In this case, the model proposed by S.Smyl [56] has a better performance for Mean sMAPE score. However, both the proposed model and Smyl's model have the same score in Mean MAPE, implying that both models have same performance on forecasting Ericsson KPI dataset compared to naive method on average. In figure 4.3, we show some predictions made by proposed model.



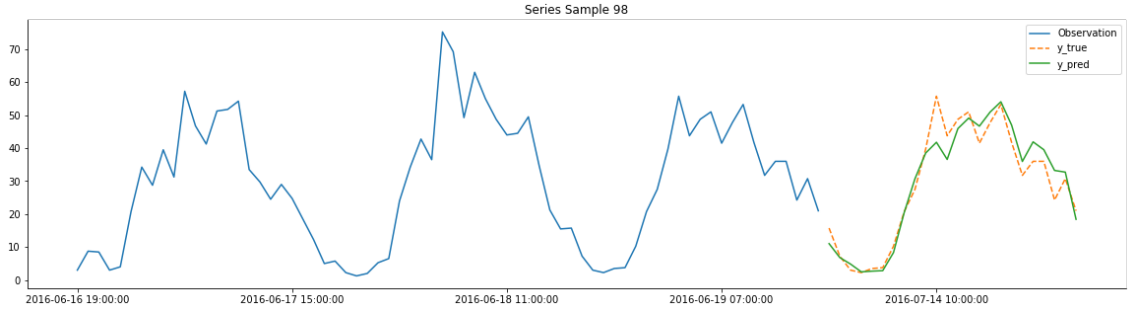
(a) Ericsson KPI Sample Series 18



(b) Ericsson KPI Sample Series 22

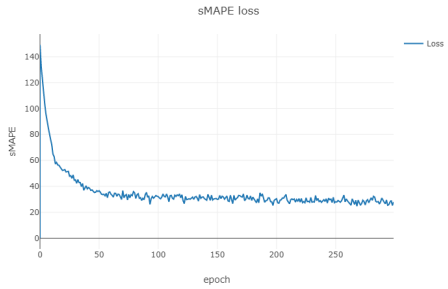


(c) Ericsson KPI Sample Series 58

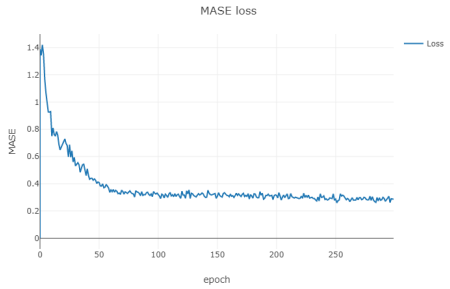


(d) Ericsson KPI Sample Series 98

Figure 4.3: Some forecasting results from Ericsson KPI data



(a) sMAPE vs epoch training on Ericsson KPI dataset



(b) MASE vs epoch training on Ericsson KPI dataset

Figure 4.4: sMAPE and MASE vs epoch training on Ericsson KPI dataset

4.2 Result of Experiment II

NN5 Case

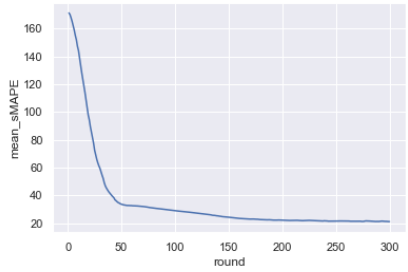
The NN5 results of Experiment II are shown in table 4.3.

NN5 Case	Mean sMAPE	Mean MASE
Proposed Model	**20.20**	**0.57**
Federated Proposed Model	21.05	0.79
Hybrid ESDRNN (S.Smyl [56])	21.57	0.64
Combined STL+LSTM (Bandara [6])	23.46	0.96
ES (Bandara [6])	21.44	0.86
ARIMA (Bandara [6])	25.29	0.97
STL Seasonal	23.80	0.94
Naive Seasonal	28.13	1.04
Naive	48.26	1.83

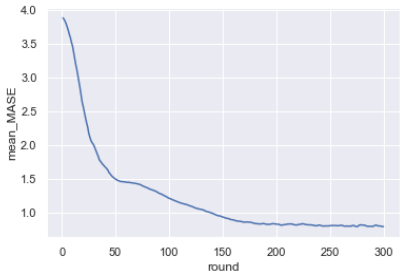
Table 4.3: Results of NN5 dataset

We can clearly see that the Federated setting for the proposed model performs a bit worse in both sMAPE and MASE on average. However, its mean MASE is < 1, means on average the model performs better than the naive method. Also, we

can see that the proposed model with federated setup outperforms the most of baseline models that are running in the centralized setup, including state-of-art forecasting methods like ARIMA. In figure 4.5, we show some predictions made on client side.

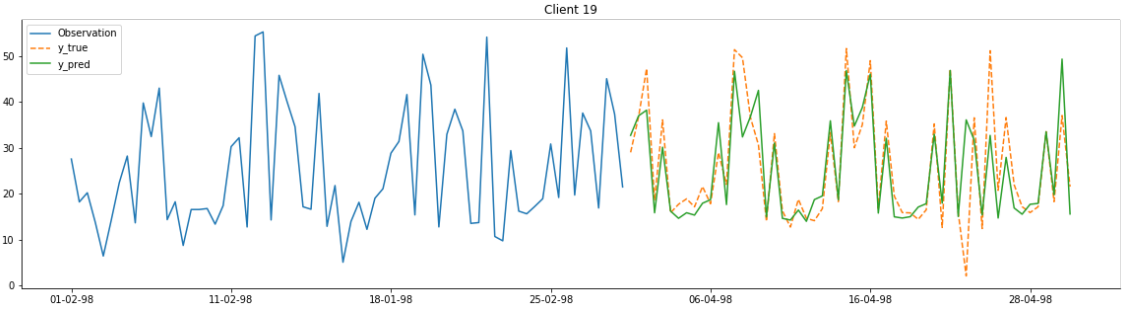


(a) sMAPE vs round training on NN5 dataset



(b) MASE vs round training on NN5 dataset

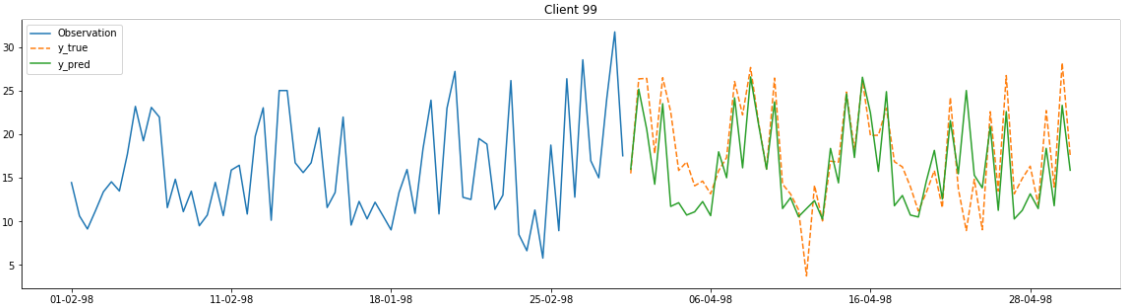
Figure 4.6: sMAPE and MASE vs round training on NN5 dataset



(a) Client 19

(b) Client 22

(c) Client 58



(d) Client 19

Figure 4.5: Some forecasting results from client

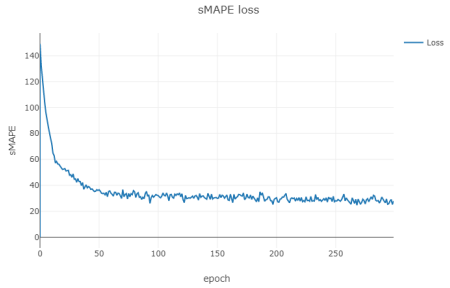
Ericsson KPI Case

The Ericsson KPI results of Experiment II are shown in table 4.4.

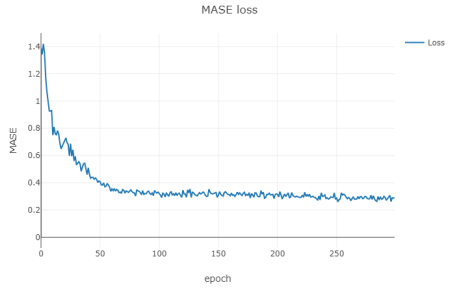
Ericsson KPI Case	Mean sMAPE	Mean MASE
Proposed Model	**27.90**	**0.28**
Federated Proposed Model	29.58	0.51
Hybrid ESDRNN (S.Smyl [56])	28.07	**0.28**
STL Seasonal	42.57	0.52
Naive Seasonal	36.47	0.53
Naive	74.54	1.29

Table 4.4: Results of Ericsson KPI dataset

Similar to the NN5 case, we can clearly see that the Federated setting for the proposed model performs a bit worse in both sMAPE and MASE on average. However, its mean MASE is < 1, means on average the model performs better than the naive method. Also, we can see that the proposed model with federated setup outperforms the most of baseline models that are running in the centralized setup. In figure 4.7, we show some predictions made on client side.

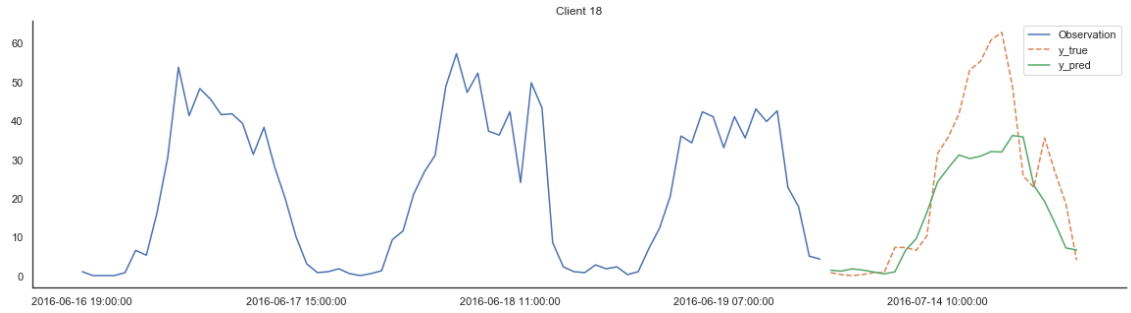


(a) sMAPE vs round training on Ericsson KPI dataset

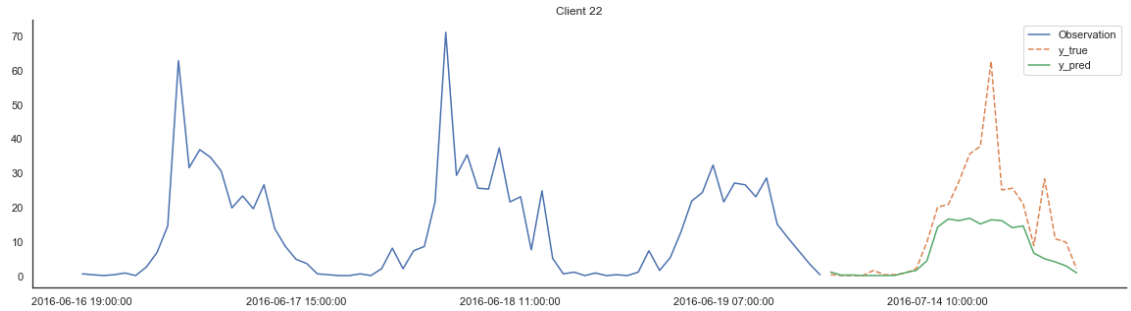


(b) MASE vs round training on Ericsson KPI dataset

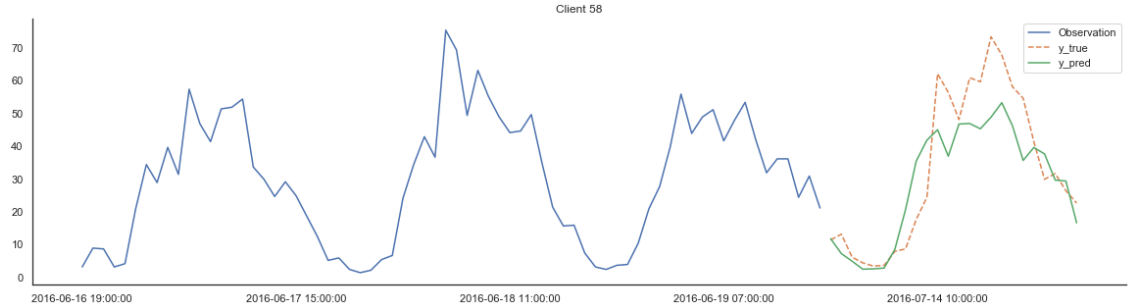
Figure 4.8: sMAPE and MASE vs round training on Ericsson KPI dataset



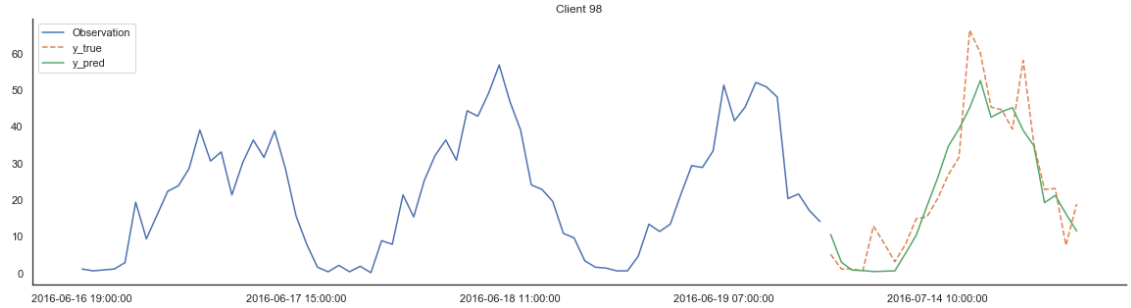
(a) Client 18



(b) Client 22



(c) Client 58



(d) Client 98

Figure 4.7: Some forecasting results from client

5 Concluding Remarks

We discussed various Time Series Forecasting methods in chapter 2. The majority of those methods rely on statistics, therefore, the results from those methods are usually robust due to strong mathematical support. However, when forecasting a large scale time series data, such methods can be costly. Meanwhile, the investigation of using Machine Learning methods to solve Time Series Forecasting problems has been carried out in various directions such as sales prediction [40], revenue prediction [25], and stock prediction [31]. These studies show that it is pragmatic to apply machine learning techniques to solve Time Series Forecasting problems. A breakthrough was made by S.Smyl, who proposed the hybrid model utilizing the Exponential Smoothing for Time Series specific features and Recurrent Neural Network to learn the temporal dependence [56]. This hybrid Neural Network model showed a promising result by beating many other sophisticated statistical design [44]. This result enables a time series-specific learning with Neural Network model, which motivates this thesis.

On the other hand, in the Big Data era, handling the increasing volume and variety of data is the main theme for designing a scalable system. The difficulties of communicating with large volumes of data and the growth of privacy concerns pushed the technology migrating from clouding computing to edge computing. The same analogy was also introduced to distributed deep Learning field, one example is the Federated Learning framework.

In this thesis, we stepped further on large-scale Time Series forecasting, by proposing a Federated Time Series Foresting. It utilizes the benefits of both hybrid Time Series Specific NN model and Federated Learning framework. By comparing the proposed forecasting model with state-of-art baseline models, the proposed forecasting model outperforms the state-of-art baseline models. This result shows that the proposed forecasting model is effective for the forecasting task. By comparing the forecasting performance of the federated network and the data-centralized network with proposed model, the federated setting achieved a comparable performance by losing a small margin on two real-life datasets. This result shows that the federated forecasting can produce comparable result against

conventional data-centralized approach.

We conclude this thesis as our results answer the research question raised in chapter 1:

Can we realize decentralized time series forecasting with a Federated Learning mechanism that is comparable to the conventional centralized setup in forecasting performance?

5.1 Discussion

5.1.1 Effectiveness of the Forecasting Model

We compared the forecasting performances among the state-of-art baseline models and proposed model. The proposed model outperformed all the baseline models in terms of Mean sMAPE and Mean MASE in both NN5 dataset and Ericsson KPI dataset. In both cases, the proposed model achieved the highest mean MASE score, which implies that the proposed model has a relatively good forecasting performance on both datasets. The results help us validate the effectiveness of the model for these two datasets. During the experiment, several state-of-art models, especially in NN5 case we include the results from Bandara's research [6], are used as baseline models. Also, NN5 and Ericsson KPI data have different seasonality patterns. The results further imply the proposed model can provide good forecasting for time series data that has seasonality. However, the results also imply that our conclusion is drawn based on the fact that the dataset has a clear seasonality. So the existence of seasonality in the dataset will affect the forecasting performance of the proposed model.

5.1.2 Federated Time Series Forecasting

The results from both NN5 and Ericsson KPI cases showed the sufficiency of deploying the proposed model in a Federated Learning framework to achieve

Federated Time Series Forecasting. We noticed that after 300 rounds of communication the network converged. The final Mean sMAPE and Mean MASE comparing to the conventional data centralized approach, losing 0.85, 0.22 in NN5 case and 1.68, 0.23 in Ericsson case respectively. Note that in both experiments we only enabled 30% of clients per training round. It is possible that some clients need even more round to train, as both Mean sMAPE and Mean MASE show the average case. Nonetheless, the results of Federated Forecasting in both cases are still comparable to the baseline models from Experiment I. There is clearly a trade-off between centralized approach and federated framework in Time Series Forecasting, in terms of the amount of information the model can see during the training. This can be addressed by the hyperparameters of the federated averaging algorithm, which are the number of clients K, local mini-batch size B, and local epochs E and the ratio of selected clients per communication rounds C. With more K means the network will have more training data, bigger B and E will increase the local training quality, and bigger C results in the network converge with fewer communication rounds. However, these changes come with a cost, as bigger B and E means more computational pressure at the edge device, bigger C means more communication workload, and due to the limited communication property of federated learning, we may not have the entire K all the time. These aspects affect the general forecasting performance of the federated network. It is not trivial to choose these parameters since the choice differs for different scenarios.

5.2 Drawbacks and Future Work

We conclude the thesis by answering the research question with some exciting findings. However, there are some drawbacks.

First, both NN5 and Ericsson dataset have a strong seasonality, which means they work well with Exponential Smoothing. However, there are cases that time series does not have strong seasonality or time series dataset can have multiple types of the series. An intuitive remedy is to create an ensembling metric on top of the hybrid model to tackle this issue.

Another drawback of this thesis is that it did not take into account the holiday effects. Most Time Series data reflects a phenomenon or event with respect to the time, it is possible that some unusual event happened in a particular time that can affect the forecasting result. For instance, traffic information can be different during the National Holidays. Both of the datasets we used may have certain unusual events, however, due to the time constraints, we did not look into this direction. There are approaches to resolve this issue with Encoder-Decoder LSTMs [71]. Integrating this feature to the model can be a promising future work.

A promising future work can be using a federated meta-learning framework to share the algorithm instead of the model. Our current setup relies on a shared model, which means all clients will use the same model structure, which is acceptable in this work. However, this approach has limitations when the shared model is complicated, as edge devices often have limited network bandwidth and computation resource to run complicated models. Fei et al proposed a federated meta-learning framework for Recommendation Systems, solving the issue by sharing the information at algorithms level instead of the model level. It has already been proved on their production dataset that Federated meta-Learning framework achieves a better accuracy with fewer parameters compared to Federated Learning framework [10].

References

[1] Adhikari, Ratnadip and Agrawal, R. K. “An Introductory Study on Time Series Modeling and Forecasting”. In: *CoRR* abs/1302.6613 (2013). arXiv: 1302.6613. URL: http://arxiv.org/abs/1302.6613.

[2] An, N. H. and Anh, D. T. “Comparison of Strategies for Multi-step-Ahead Prediction of Time Series Using Neural Network”. In: *2015 International Conference on Advanced Computing and Applications (ACOMP)*. Nov. 2015, pp. 142–149. DOI: 10.1109/ACOMP.2015.24.

[3] James A. Anderson and Edward Rosenfeld, eds. *Neurocomputing: Foundations of Research*. Cambridge, MA, USA: MIT Press, 1988. ISBN: 0-262-01097-6.

[4] Armbrust, M. “Identification of Factors that Enable or Impede the Cloud Computing Adoption Decision”. In: *Communications of the ACM* 53.4 (2010), pp. 50–8.

[5] B Cleveland, Robert et al. “STL: A Seasonal-Trend Decomposition Procedure Based on Loess”. In: *Journal of Official Statistics* 6 (Jan. 1990), pp. 3–33.

[6] Bandara, Kasun, Bergmeir, Christoph, and Smyl, Slawek. “Forecasting Across Time Series Databases using Long Short-Term Memory Networks on Groups of Similar Series”. In: *CoRR* abs/1710.03222 (2017). arXiv: 1710.03222. URL: http://arxiv.org/abs/1710.03222.

[7] Ben Taieb, Souhaib et al. “A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition”. In: *Expert Systems with Applications* 39 (2011). DOI: 10.1016/j.eswa.2012.01.039.

[8] Brown, R. G. *Statistical forecasting for inventory control*. English. USA: McGraw/Hill, 1959.

[9] Bryson, S. “5G technology needs edge computing architecture”. In: (2018). URL: https://www.cisco.com/c/en/us/solutions/enterprise-networks/edge-computing-architecture-5g.html.

[10] Chen, Fei et al. “Federated Meta-Learning for Recommendation”. In: *CoRR* abs/1802.07876 (2018). arXiv: 1802.07876. URL: http://arxiv.org/abs/1802.07876.

[11] Ciresan, Dan Claudiu et al. “Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition”. In: *CoRR* abs/1003.0358 (2010). arXiv: 1003.0358. URL: http://arxiv.org/abs/1003.0358.

[12] Cleveland W.S., Loader C. *Smoothing by Local Regression: Principles and Methods*. Jan. 1996. DOI: 10.1007/978-3-642-48425-4_2.

[13] Coates, Adam, Lee, Honglak, and Ng, Andrew Y. “An analysis of single-layer networks in unsupervised feature learning”. In: *In AISTATS 14*. 2011.

[14] Collobert, Ronan and Weston, Jason. *A unified architecture for natural language processing: Deep neural networks with multitask learning*. 2008.

[15] Dahl, G. E. et al. “Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition”. In: *Trans. Audio, Speech and Lang. Proc.* 20.1 (Jan. 2012), pp. 30–42. ISSN: 1558-7916. DOI: 10.1109/TASL.2011.2134090. URL: https://doi.org/10.1109/TASL.2011.2134090.

[16] Dean, Jeffrey and Ghemawat, Sanjay. “MapReduce: Simplified Data Processing on Large Clusters”. In: vol. 51. Jan. 2004, pp. 137–150. DOI: 10.1145/1327452.1327492.

[17] Dean, Jeffrey et al. “Large Scale Distributed Deep Networks”. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1223–1231. URL: http://dl.acm.org/citation.cfm?id=2999134.2999271.

[18] D’informatique Et Recherche Operationnelle, Departement et al. “A Neural Probabilistic Language Model”. In: (Oct. 2001).

[19] Ericsson. “Making waves with AI”. In: (2018). URL: https://www.ericsson.com/en/%20mobility-report/reports/june-2018/applying-machine-intelligenceto-%20network-management.

[20] Ericsson. “This is 5G”. In: (2018). URL: https://www.ericsson.com/assets/local/newsroom/media-kits/5g/doc/ericsson_this-is-5g_pdf_v4.pdf.

[21] Gabor, Manuela Rozalia and Ancuta Dorgo, Lavinia. “Neural Networks Versus Box-Jenkins Method for Turnover Forecasting: a Case Study on the Romanian Organisation”. In: *Transformations in Business and Economics* 16 (Mar. 2017), pp. 187–211.

[22] GDPR. “General Data Protection Regulation”. In: (2018). URL: https://gdpr-info.eu/.

[23] Goodfellow, Ian J., Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[24] Granville, Kevin. “Facebook and Cambridge Analytica: What You Need to Know as Fallout Widens”. In: (2018). URL: https://www.nytimes.com/2018/03/19/technology/facebook-cambridge-analytica-explained.html.

[25] Al-Gunaid, M. A. et al. “Time Series Analysis Sales of Sowing Crops Based on Machine Learning Methods”. In: *2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)*. July 2018, pp. 1–6. DOI: 10.1109/IISA.2018.8633610.

[26] Guo, T. “Distributed Time Series Analytics”. In: (2017). URL: https://infoscience.epfl.ch/record/224052/files/EPFL_TH7395.pdf.

[27] Håkansson, A. “Portal of Research Methods and Methodologies for Research Projects and Degree Projects”. In: (2013). URL: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-136960.

[28] Hamid, S.A. and Habib, A. “Financial forecasting with neural networks”. In: *Academy of Accounting and Financial Studies Journal* 18 (Jan. 2014), pp. 37–55.

[29] Hard, Andrew et al. “Federated Learning for Mobile Keyboard Prediction”. In: *CoRR* abs/1811.03604 (2018). arXiv: 1811.03604. URL: http://arxiv.org/abs/1811.03604.

[30] Hassan, N. et al. “The Role of Edge Computing in Internet of Things”. In: *IEEE Communications Magazine* 56.11 (2018), pp. 110–115. ISSN: 0163-6804.

[31] Hegazy, Osman, Soliman, Omar S., and Abdul Salam, Mustafa. “A Machine Learning Model for Stock Market Prediction”. In: *International Journal of Computer Science and Telecommunications* 4 (Dec. 2013), pp. 17–23.

[32] Hochreiter, Sepp and Schmidhuber, Jürgen. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[33] Holt, Charles C. “Forecasting seasonals and trends by exponentially weighted moving averages”. In: *International Journal of Forecasting* 20.1 (2004), pp. 5–10. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2003.09.015. URL: http://www.sciencedirect.com/science/article/pii/S0169207003001134.

[34] Hyndman, Rob J. and Koehler, Anne B. “Another look at measures of forecast accuracy”. In: *International Journal of Forecasting* 22.4 (2006), pp. 679–688. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2006.03.001. URL: http://www.sciencedirect.com/science/article/pii/S0169207006000239.

[35] Hyndman, Robin John and Athanasopoulos, George. *Forecasting: Principles and Practice*. English. 2nd. Australia: OTexts, 2018.

[36] Islam, Mohammad Manzurul, Morshed, Sarwar, and Goswami, Parijat. “Cloud Computing: A Survey on its limitations and Potential Solutions”. In: *International Journal of Computer Science Issues* (2013).

[37] Kock, Anders and Terasvirta, Timo. “Forecasting Macroeconomic Variables Using Neural Network Models and Three Automated Model Selection Techniques”. In: *Econometric Reviews* (Jan. 2011). DOI: 10.1080/07474938.2015.1035163.

[38] Konecný, Jakub et al. “Federated Optimization: Distributed Machine Learning for On-Device Intelligence”. In: *CoRR* abs/1610.02527 (2016). arXiv: 1610.02527. URL: http://arxiv.org/abs/1610.02527.

[39] Low, Yucheng et al. “Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud”. In: *Proc. VLDB Endow.* 5.8 (Apr. 2012), pp. 716–727. ISSN: 2150-8097. DOI: 10.14778/2212351.2212354. URL: https://doi.org/10.14778/2212351.2212354.

[40] M. Pavlyshenko, Bohdan. “Machine-Learning Models for Sales Time Series Forecasting”. In: *Data* 4 (Jan. 2019), p. 15. DOI: 10.3390/data4010015.

[41] *Makridakis Competitions*. URL: https://www.mcompetitions.unic.ac.cy/m-competitions/.

[42] Makridakis, Spyros and Hibon, Michele. “The M3-Competition: results, conclusions and implications”. In: *International Journal of Forecasting 16.4* (2000), pp. 451–476. ISSN: 0169-2070.

[43] Makridakis, Spyros, Spiliotis, Evangelos, and Assimakopoulos, Vassilios. “Statistical and Machine Learning forecasting methods: Concerns and ways forward”. In: *PLOS ONE 13.3* (2018). ISSN: 1932-6203.

[44] Makridakis, Spyros, Spiliotis, Evangelos, and Assimakopoulos, Vassilios. “The M4 Competition: Results, findings, conclusion and way forward”. In: *International Journal of Forecasting 34.4* (2018), pp. 802–808. ISSN: 0169-2070.

[45] Makridakis, Spyros, Spiliotis, Evangelos, and Assimakopoulos, Vassilis. “The Accuracy of Machine Learning (ML) Forecasting Methods versus Statistical Ones: Extending the Results of the M3-Competition”. In: (Oct. 2017).

[46] Malsburg, Christoph von der. “Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms”. In: *Brain Theory* (Jan. 1986), pp. 245–248. DOI: 10.1007/978-3-642-70911-1_20.

[47] McMahan, H. Brendan et al. “Federated Learning of Deep Networks using Model Averaging”. In: *CoRR* abs/1602.05629 (2016). arXiv: 1602.05629. URL: http://arxiv.org/abs/1602.05629.

[48] Nelson, Michael et al. “Time series forecasting using neural networks: should the data be deseasonalized first?” In: *Journal of Forecasting* 18.5 (1999), pp. 359–367. DOI: 10.1002/(SICI)1099-131X(199909)18:5<359::AID-FOR746>3.0.CO;2-P.

[49] P. Hofmann, D. Woods. “Cloud computing: the limits of public clouds for business applications’, Internet Computing”. In: *IEEE Access* 14.6 (2010).

[50] Patgiri, Ripon and Ahmed, Arif. “Big Data: The V’s of the Game Changer Paradigm”. In: (2016). DOI: 10.1109/HPCC-SmartCity-DSS.2016.0014.

[51] Pratama, I. et al. “A review of missing values handling methods on time-series data”. In: *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*. Oct. 2016, pp. 1–6. DOI: 10.1109/ICITSI.2016.7858189.

[52] Prerna Lal, Sangeeta Shah Bharadwaj. “Identification of Factors that Enable or Impede the Cloud Computing Adoption Decision”. In: *IRACST – International Journal of Commerce, Business and Management (IJCBM)* 4.6 (2015). ISSN: 2319–2828.

[53] Qiu, Mingyue and Song, Yu. “Predicting the Direction of Stock Market Index Movement Using an Optimized Artificial Neural Network Model”. In: *PLOS ONE* 11 (May 2016), e0155133. DOI: 10.1371/journal.pone.0155133.

[54] Sainath, Tara N. et al. “Deep Convolutional Neural Networks for Large-scale Speech Tasks”. In: *Neural Netw.* 64.C (Apr. 2015), pp. 39–48. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.08.005. URL: http://dx.doi.org/10.1016/j.neunet.2014.08.005.

[55] Shcherbakov, Maxim et al. “A survey of forecast error measures”. In: *World Applied Sciences Journal* 24 (Jan. 2013), pp. 171–176. DOI: 10.5829/idosi.wasj.2013.24.itmies.80032.

[56] Smyl, Slawek, Ranganathan, Jai, and Pasqua, Andrea. “M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model”. In: (2018). URL: https://eng.uber.com/m4-forecasting-competition/.

[57] Stollenga, Marijn. “Advances in Humanoid Control and Perception”. PhD thesis. May 2016.

[58] Sun, Chen et al. “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era”. In: *CoRR* abs/1707.02968 (2017). arXiv: 1707.02968. URL: http://arxiv.org/abs/1707.02968.

[59] Taieb, S. B. et al. “Long-term prediction of time series by combining direct and MIMO strategies”. In: *2009 International Joint Conference on Neural Networks*. June 2009, pp. 3054–3061. DOI: 10.1109/IJCNN.2009.5178802.

[60] Taieb, Souhaib Ben and Hyndman, Rob J. “Recursive and direct multi-step forecasting: the best of both worlds”. In: 2012.

[61] Wang, Jie and Wang, Jun. “Forecasting Stochastic Neural Network Based on Financial Empirical Mode Decomposition”. In: *Neural Netw.* 90.C (June 2017), pp. 8–20. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2017.03.004. URL: https://doi.org/10.1016/j.neunet.2017.03.004.

[62] Williams, Mike Lee. “An introduction to Federated Learning”. In: (2018). URL: http://vision.cloudera.com/an-introduction-to-federated-learning/.

[63] Williams, Ronald J. and Zipser, David. *Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity*. 1995.

[64] Winters, Peter R. “Forecasting Sales by Exponentially Weighted Moving Averages”. In: *Manage. Sci.* 6.3 (Apr. 1960), pp. 324–342. ISSN: 0025-1909. DOI: 10.1287/mnsc.6.3.324. URL: http://dx.doi.org/10.1287/mnsc.6.3.324.

[65] WIRED Brand Lab, Western Digital Corporation. “Why Edge Computing is Key to a 5G Future”. In: (2018). URL: https://datamakespossible.westerndigital.com/edge-computing-key-5g-future/.

[66] Wu, Kehe et al. “A distributed real-time data prediction framework for large-scale time-series data using stream processing”. In: *International Journal of Intelligent Computing and Cybernetics* 10 (2017), pp. 145–165. DOI: 10.1108/IJICC-09-2016-0033.

[67] Yang, Timothy et al. “Applied Federated Learning: Improving Google Keyboard Query Suggestions”. In: *CoRR* abs/1812.02903 (2018). arXiv: 1812.02903. URL: http://arxiv.org/abs/1812.02903.

[68] Yu, W. et al. “A Survey on the Edge Computing for the Internet of Things”. In: *IEEE Access* (2018), pp. 6900–6919. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2778504.

[69] Zhang, Peter. “Zhang, G.P.: Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model. Neurocomputing 50, 159-175”. In: *Neurocomputing* 50 (Jan. 2003), pp. 159–175. DOI: 10.1016/S0925-2312(01)00702-0.

[70] Zhang, Peter and Qi, Min. “Neural network forecasting for seasonal and trend time series”. In: *European Journal of Operational Research* 160 (Feb. 2005), pp. 501–514. DOI: 10.1016/j.ejor.2003.08.037.

[71] Zhu, Lingxue and Laptev, Nikolay. “Deep and Confident Prediction for Time Series at Uber”. In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)* (2017). DOI: 10.1109/ICDMW.2017.19. arXiv: 1709.01907. URL: https://arxiv.org/abs/1709.01907.

Appendix A: Additional plots for the NN5 dataset

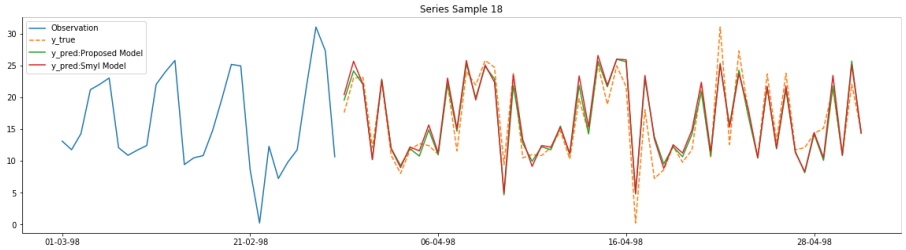


Figure 5.1: A comparison of Proposed model and Smyl's model in forecasting sample series 18 of NN5 dataset

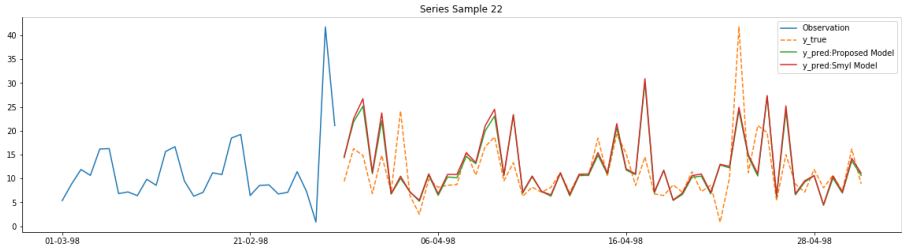


Figure 5.2: A comparison of Proposed model and Smyl's model in forecasting sample series 22 of NN5 dataset

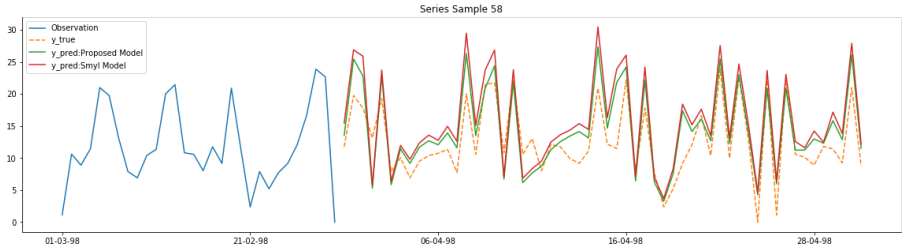


Figure 5.3: A comparison of Proposed model and Smyl's model in forecasting sample series 58 of NN5 dataset

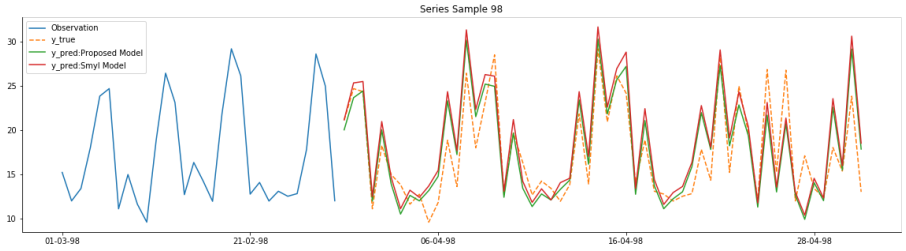


Figure 5.4: A comparison of Proposed model and Smyl's model in forecasting sample series 98 of NN5 dataset

Appendix B: Additional plots for the Ericsson KPI dataset

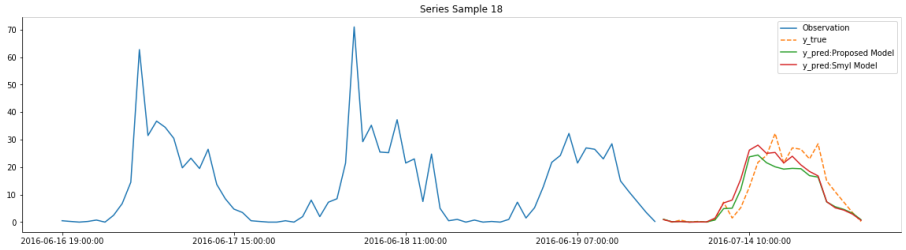


Figure 5.5: A comparison of Proposed model and Smyl's model in forecasting sample series 18 of Ericsson KPI dataset

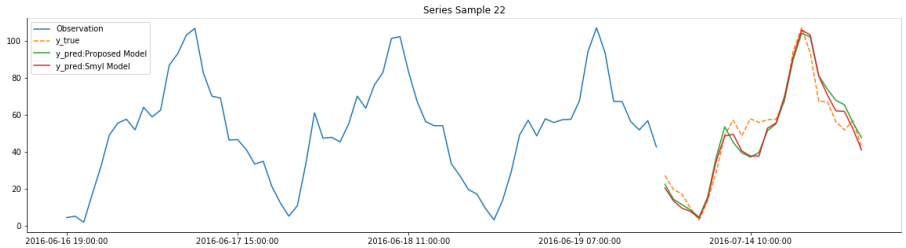


Figure 5.6: A comparison of Proposed model and Smyl's model in forecasting sample series 22 of Ericsson KPI dataset

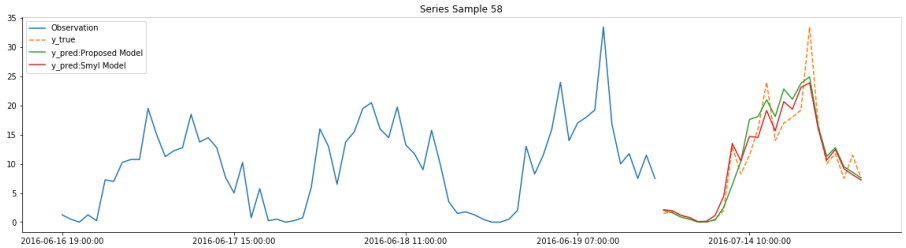


Figure 5.7: A comparison of Proposed model and Smyl's model in forecasting sample series 58 of Ericsson KPI dataset

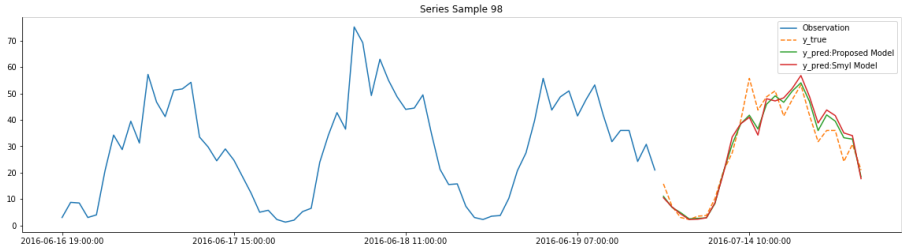


Figure 5.8: A comparison of Proposed model and Smyl's model in forecasting sample series 98 of Ericsson KPI dataset

TRITA-EECS-EX-2019:472