6-21-2022 3:00 PM

Machine Learning with Big Data for Electrical Load Forecasting

Alexandra L'Heureux, *The University of Western Ontario*

Supervisor: Capretz, Miriam, *The University of Western Ontario*
Co-Supervisor: Grolinger, Katarina, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Electrical and Computer Engineering

Abstract

Today, the amount of data collected is exploding at an unprecedented rate due to developments in Web technologies, social media, mobile and sensing devices and the internet of things (IoT). Data is gathered in every aspect of our lives: from financial information to smart home devices and everything in between. The driving force behind these extensive data collections is the promise of increased knowledge. Therefore, the potential of Big Data relies on our ability to extract value from these massive data sets. Machine learning is central to this quest because of its ability to learn from data and provide data-driven insights, decisions, and predictions. However, traditional machine learning approaches were developed in a different era and thus are based upon multiple assumptions that unfortunately no longer hold true in the context of Big Data.

This thesis presents the challenges associated with performing machine learning on Big Data and highlights the cause-effect relationship between the defining dimensions of Big Data and the applications of machine learning techniques. Additionally, emerging machine learning paradigms and how they can handle the challenges are identified. Although many areas of research and applications are affected by these challenges, this thesis focuses on tackling those associated with electrical load forecasting. Consequently, two of the identified challenges are addressed.

Firstly, an adaptation of the transformer architecture for electrical load forecasting is proposed in order to address the training time performance-related challenge associated with deep learning algorithms. The result showed improved accuracy for various forecasting horizons over the current state-of-the-art algorithm and addressed performance shortcomings through the architecture's ability to be parallelized.

Secondly, a transfer learning algorithm is proposed to scale the learning of load forecasting tasks and effectively address the performance challenges associated with transfer learning. Additionally, the diversity of the data was examined to analyze the portability of the results. In spite of facing various data distributions, the learned concepts and results were repeatable over multiple streams. The results showed significant improvements to machine learning model training time, where the scaled models were 1.7 times faster on average leading to much more efficient model deployment times.

Keywords: Big Data, Machine Learning, Deep Learning, Transfer Learning, Load Forecasting

Lay Summary

Today, the amount of data collected is exploding at an unprecedented rate due to developments in Web technologies, social media, mobile and sensing devices and the internet of things (IoT). Data is gathered in every aspect of our lives: from financial information to smart home devices and everything in between. The driving force behind these extensive data collections is the promise of increased knowledge. Therefore, the potential of Big Data relies on our ability to extract value from these massive data sets. Machine learning is central to this quest because of its ability to learn from data. However, traditional machine learning approaches were developed in a different era and thus are facing a number of challenges.

This thesis presents those challenges associated with performing machine learning on Big Data. Although many research areas and applications are affected by these challenges, this thesis focuses on tackling those related to electrical load forecasting. Consequently, two of the identified challenges are addressed. Firstly, an adapted architecture for electrical load forecasting is proposed in order to address the training time performance-related challenge associated with deep learning algorithms. Secondly, a transfer learning algorithm is proposed to improve the learning time of load forecasting tasks and effectively address the performance challenges associated with transfer learning.

Co-Authorship Statement

The thesis presented here has been written by Alexandra L'Heureux under the supervision of´ Dr. Miriam A. M. Capretz and Dr. Katarina Grolinger. Part of the content of this thesis has been published in a peer-reviewed journal. The research published has been mainly conducted and written by the principal author. The presented research is guided and supported by Dr. Miriam Capretz and Dr. Katarina Grolinger as research supervisors. A version of the material presented in Chapter 3 has been published in

- A. L'Heureux, K. Grolinger, H. F. Elyamany and M. A. M. Capretz, "Machine Learning With Big Data: Challenges and Approaches," in IEEE Access, vol. 5, pp. 7776-7797, 2017, doi: 10.1109/ACCESS.2017.2696365.
- A. L'Heureux - designed, researched, directed and wrote the work presented in the manuscript

A version of the material presented in Chapter 4 has been published in

- A. L'Heureux, K. Grolinger, and M. A. M. Capretz, "Transformer-Based Model for Electrical Load Forecasting," in Energies, vol. 15, no. 14, p. 4993, Jul. 2022, doi: 10.3390/en15144993
- A. L'Heureux - designed, researched, directed and wrote the work presented in the manuscript

Acknowlegements

First and foremost, I would like to thank my supervisors, Dr. Miriam Capretz and Dr. Katarina Grolinger, for their unwavering support over the last decade. I am beyond grateful for the time you have invested in me, for your continued guidance, and for the fact that you never stopped believing in me or in my abilities, even when I sometimes did. I have the utmost respect for you both and am beyond grateful for having worked under your supervision.

Secondly, I would like to thank my advisory committee members, Dr. Aleksander Essex and Dr. Abdallah Shami for overseeing my progress and providing invaluable advice and feedback. I am grateful for your support and the opportunities you have given me.

I would also like to thank my research team in TEB 244 for their support and inspiring ideas. I would especially like to thank Santiago Gomez for always checking in and extending offers of help, your support has always been appreciated.

The completion of this thesis would not have been possible without the support of my family. To my wife and best friend Melissa, your support, encouragements and love mean the world to me. Thank you for always believing in me and providing me with the time and space I needed to work, even in the midst of chaos. I am forever grateful to you. To my sons, Noah and Milo, thank you for understanding that Maman often had to work. I hope that I have shown you the value of hard work, dedication and the importance of not giving up and believing in yourself. To my mom and brother, France and Cyril, thank you for your unconditional love and support, it did not go unnoticed.

Lastly, I would like to thank my dad Jacques (in memoriam) for teaching me the value of education and hard work and for always believing in me. I wish you could see me today.

Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Networks
FFNN	Feed-Forward Neural Networks
FPGA	Field-Programmable Gate Array
GDP	Gross Domestic Product
GPU	Graphic Processing Units
GRU	Gated Recurrent Unit
IDC	Internation Data Corporation
IoT	Internet of Things
i.i.d	Independent and Identically Distributed
LLE	Locally Linear Embedding
LSTM	Long short-term memory
MHA	Multi-headed attention
ML	Machine Learning
MLlib	Machine Learning Library
MLP	Multi-Layer Perceptron
MOA	Massive Online Analysis
NLP	Natural Language Processing
NN	Neural Networks
PCA	Principal Component Analysis
RAMP	Reduce and Map Provenance
RNN	Recurrent Neural Networks
RDD	Resilient Distributed Datasets
S2S	Sequence-to-Sequence Architecture
SVM	Support Vector Machine
SVR	Support Vector Regression

Chapter 1

Introduction

Today the amount of data collected is exploding at an unprecedented rate as a result of developments in Web technologies, mobile and sensing devices, social media, and the internet of things (IoT). The International Data Corporation (IDC) predicts that the data generated from IoT devices alone will increase by over 500% between 2019 and 2025, rising from 13.6 ZB to 79.4 ZB with up to 41.9B connected devices worldwide [6]. These Big Data possess tremendous potential in terms of business value in a variety of fields such as health care, biology, transportation, online advertising, energy management, and financial services [7], [8]. However, the 2020 Vantage Partners Big Data and Artificial Intelligence (AI) Executive Survey [9], which queries more than 70 Fortune 1000 companies, reported that 73.4% of respondents struggle to become data-driven organizations and therefore have difficulties gaining insights from the data they generate. The main challenge seems to stem from the ability or inability to process the data and perform competing analytics adequately [9]. According to Forbes, the key to successful analytics in a data-driven organization is the use of appropriate AI technologies [10] and, more specifically, Machine Learning (ML) [11]. Therefore, the challenges associated with the use of ML with Big Data have a critical impact on the transitional success of these companies.

Additionally, according to NewVantage, 97.2% of companies are investing in Big Data and AI [1], billions of dollars are spent yearly, and as shown in Figure 1.1 ML dominates these spendings. Therefore, not only are the challenges of ML with Big Data important in terms of their technical impact to address current analytical shortcomings but they are also linked to a significant financial stake. The Wall Street Journal has reported that improvements and innovations in AI and machine learning have the potential to increase the global GDP by 14% by 2030 [12].

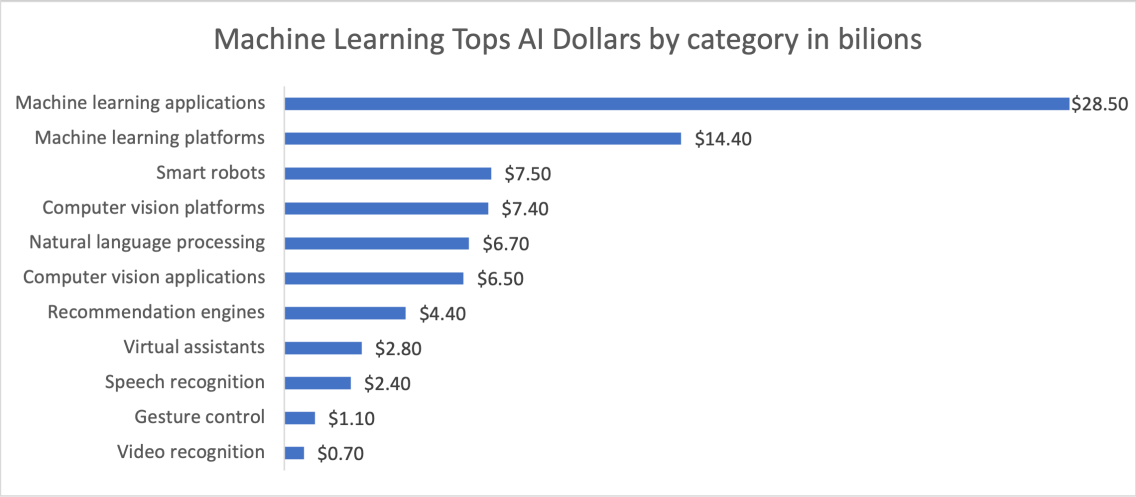


Figure 1.1: Distribution of AI funding worldwide [1]

From a business and applications perspective, organizations are facing practical challenges related to ML. Those challenges gathered by Statista in 2021 [2], are shown in Figure 1.2, and although those differ from the technical challenges directly related to the implementation of ML with Big Data, they are directly linked to the successful deployment of real-world machine learning applications.

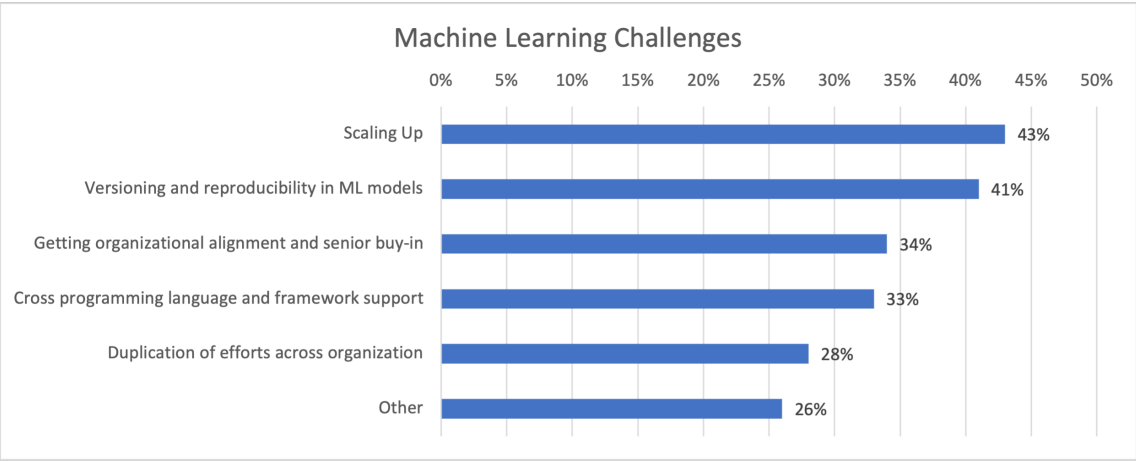


Figure 1.2: Machine Learning Challenges faced by Worldwide organizations [2]

Given the multifaceted importance of the challenges of ML with Big Data, this thesis identifies the technical challenges related to implementing machine learning algorithms with Big Data and then proposes to address some of the challenges that are best aligned with the practical challenges encountered by organizations worldwide, such as the need for scaling up solutions. Although organizations across all spheres of the economy are affected, this thesis will focus on

those from the energy sector and most specifically those dealing with the task of electrical load forecasting.

1.1 Motivation

There are two main factors that motivate this thesis: the need to identify and address the challenges of ML with Big Data and the emergence of the IoT and of the Smart Grid. The following subsections will describe in more details the motivation behind the work presented in this thesis.

1.1.1 Challenges of ML for Big Data

As previously described, the challenges of machine learning with Big Data have a significant impact on the deployment of machine learning solutions in real-world applications.

In order to address these shortcomings, we first need to establish and formalize the challenges associated with Big Data and the various machine learning paradigms. Such a foundation will provide support to appropriately tackle deployment issues and enable organizations to make more educated decisions in regards to their ML solutions based on their required tasks and data limitations.

Although almost all major worldwide enterprises consider Big Data and AI to be of utmost importance, 58.5% [9] of those organizations are still in the planning phases towards their shift to becoming data-driven. This represents an opportunity to identify the challenges ahead of deployment in order to allow for proper solution design. Although existing research has identified a number of challenges associated with ML [13, 14, 15] they either do not identify the root cause of the issue that lies specifically with Big Data or they do not address how existing ML paradigms can handle them. The identification of the Big Data dimensions causing each of these challenges along with the understanding and how different machine learning algorithms can handle these problems is needed in order to develop strongly engineered solutions. This need to establish a strong foundation and understanding of the relationship between Big Data challenges and existing ML paradigms serves as an important motivation for this work.

1.1.2 Smart Grid and the IoT

There are currently over 25B devices connected to the internet, and of those devices, more than 16B are a part of the IoT, vastly surpassing non-IoT connections. The Smart Grid is considered one of the most important applications of the IoT [16]. Figure 1.3 shows the forecasted number

of connected devices over the years; it can be observed that most new connected devices will be part of the IoT. [3].

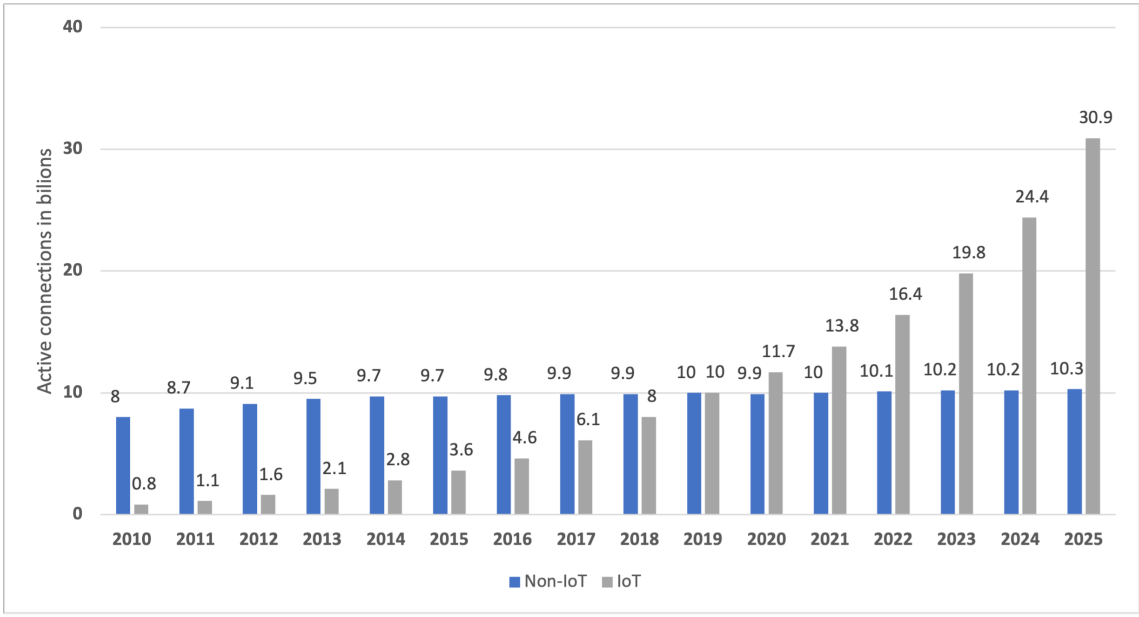


Figure 1.3: Number of internet-connected devices over the years [3]

A number of these devices are smart electrical meters, which are a key component of the Smart Grid. The number of smart meters is increasing steadily, now surpassing 115M in the US [4], as shown in Figure 1.4. The emergence of the IoT as well as the development of the Smart Grid worldwide has provided an unprecedented opportunity for both utility companies and consumers to discover new ways to not only manage their consumption and control their devices but also to enable resource conservation and reduce operational costs [17]. With a market value of over 520B USD [18], the IoT and its analytical market are key components of our economy. The vast availability of the Smart Grid has led to a number of new challenges for utility companies such as how they should manage the faster and larger amounts of data they are now collecting and how they can better provide insight into consumption through faster and more accurate load forecasting [19]. Many recent studies and approaches to energy forecasting have been published [20, 21], however, they are often evaluated on a small number of buildings or smart meters. Therefore having faster and more accurate load forecasting capable of being scaled to thousands of smart meters is also a motivation for this thesis.

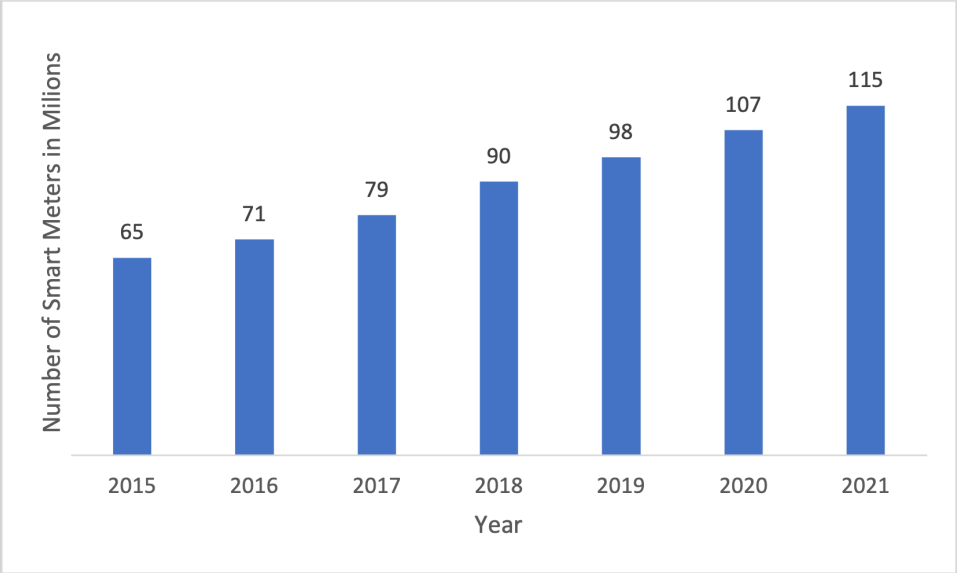


Figure 1.4: Number of Smart Electrical Meters in the US between 2015 and 2021 [4]

1.2 Objectives

This thesis can be divided into two main components, each with a different scope of focus. The first part takes a boarder approach and identifies the various challenges associated with ML with Big Data, while the second has a more narrow perspective and focuses on addressing industry relevant challenges for the task of electrical load forecasting.

The first part is composed of Chapter 3 and aims to provide a strong foundation for ML with Big Data without narrowing the challenges to a specific field of study. On the other hand, Chapters 4 and 5 make use of the findings identified in Chapter 3 and tackle issues relevant to electrical load forecasting.

Chapter 4 addresses the processing performances of electrical load forecasting using the deep learning paradigm, while Chapter 5 undertakes the concern of ML scaling with electrical load forecasting using transfer learning. Therefore, both address shortcomings identified in Chapter 3; the processing performance of deep learning and the performance challenge of transfer learning respectively.

1.3 Contributions

The contributions of this thesis are summarized in the following subsections.

1.3.1 Contributions of Chapter 3

The contributions found in Chapter 3 are:

- Provided a systematic review of the challenges associated with machine learning in the context of Big Data and categorized them according to the V dimensions of Big Data in order to create a foundation for a deeper understanding of machine learning with Big Data.
- Presented an overview of the various ML paradigms and discussed how these techniques can overcome the various identified challenges.
- Enabled the creation of connections among the various issues and solutions in this field of study by developing a comprehensive matrix that lays out the relationships between the various challenges and machine learning approaches, thereby highlighting the best choices given a set of conditions.
- Provided the academic community with potential directions for future work and served as the groundwork for improvements in the field of machine learning with Big Data.

1.3.2 Contributions of Chapter 4

The contributions found in Chapter 4 are:

- Identified the parallelizability of the transformer architecture as a potential solution to address the performance issues of load forecasting.
- Adapted the novel transformer architecture to handle time-series data in order to take advantage of its parallel nature.
- Addressed the performance issues of deep learning for load forecasting by successfully modifying the transformer architecture to perform load forecasting tasks.
- Demonstrated the potential of using existing architectures to address other challenges in other fields of applications.
- Investigated the impact of data variability on the performance of load forecasting tasks.

1.3.3 Contributions of Chapter 5

The contributions found in Chapter 5 are:

- Provided a strategy to effectively scale learning through algorithm modifications.
- Highlighted the suitability of algorithmic modification as an efficient solution to improve performance of already deployed solutions.
- Addressed the performance challenge of transfer learning in the context of load forecasting.
- Put forth a strategy to ensure result portability when dealing with load forecasting.
- Demonstrated the means to fast train ML models for load forecasting.

1.4 Thesis Organization

This thesis is organized as follows:

- Chapter 2 introduces the necessary background concepts and information required to understand and assess the work presented in this thesis. It first introduces and defines the concept of Big Data and its impact on the energy sector. Secondly, it presents ML algorithms upon which this works relies. Lastly it discusses load forecasting challenges and techniques.
- Chapter 3 presents a comprehensive review of ML challenges with Big Data. It explores the relationship between the Big Data dimensions and how they affect the use of machine learning. The main ML paradigms are then explored and we identify how they can address shortcomings of the more traditional machine learning techniques. Lastly, research opportunities for each paradigm are identified and discussed.
- Chapter 4 introduces an adaptation of the transformer architecture for time series data and a solution to the performance issues of deep learning algorithm with load forecasting is presented.
- Chapter 5 presents a scaling strategy to address the performance challenges related to transfer learning. Through algorithmic and workflow modification, existing models are used to effectively scale learning and address industry issues related to deployment repeatability. Additionally, a novel evaluation strategy is presented to ensure result portability for load forecasting
- Chapter 6 concludes this thesis by summarizing the research presented and providing direction and opportunities for future work.

Chapter 2

Background

This chapter will provide the necessary background information regarding the concepts discussed in this thesis. First, Section 2.1will introduce the idea of Big Data, and its impact on the energy domain will be examined. Secondly, Section 2.2 will give an introduction to machine learning and to the algorithms relevant to the concepts introduced in Chapters 4 and 5.

2.1 Big Data

The first academic reference to the term Big Data was made in 2003 by Diebold [22] in the context of macroeconomics. It was initially defined as ”the explosion in quantity (and sometimes, quality) of available and potentially relevant data, largely the result of recent and unprecedented advancements in data recording and storage technology.” Although it is not a new concept, it has continued to emerge and grow over the last two decades to become an industry worth over 162.6B USD in 2021 [23].

2.1.1 Big Data Definition

Although there is no official or formal definition of Big Data, the Gartner project defines it as “high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision-making, insight discovery, and process optimization” [24]. Volume, velocity and variety have been coined by the authors as the V characteristics [25] and have served as a foundation to define Big Data in academia. Many researchers have extended the definition to encompass further V dimensions such as value [26] and veracity [27]. However, this thesis makes use of the definition first proposed by IBM [27], which includes the veracity component, but not value. This definition is widely accepted and has been used by a number of authors [28, 29]. Although it is believed that value is of high importance

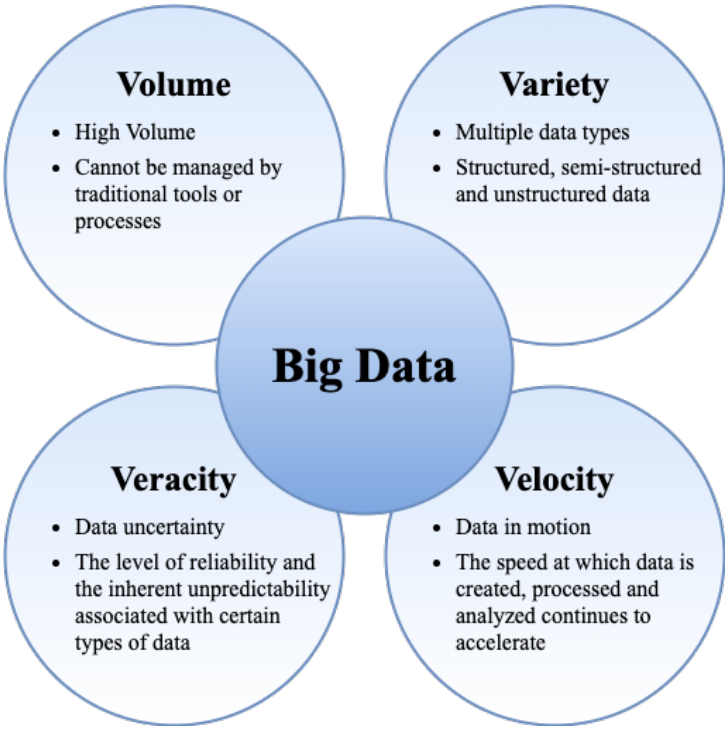


Figure 2.1: Definition of Big Data

in the context of Big Data, it is seen as a result of using Big Data rather than as a defining characteristic. Figure 2.1 depicts the definition of Big Data in accordance with IBM [30].

2.1.2 Big Data in Energy

Prior the introduction of the smart grid, existing power infrastructures in the US are aging at a critical rate. In 2015, 70% of transmission lines and transformers were over 25 years old. [31]. The emergence of the IoT and the affordability of its related technologies prompted an important shift towards a smart grid. It is estimated that the planned upgrade of the US power grid will cost upwards of 476B USD between 2011 and 2031, however, it is estimated to bring forth benefits worth over 2 trillion USD [32]. Those benefits will come mainly from the ability of the smart grid to enable energy savings, reduce cost, and increase reliability [32]. However, with its new capacity to capture more data through smart meters, utility companies will have to deal with 2880 times larger data load. It is estimated that meter readings alone will result in an annual data volume of 120 TB of raw data with data sampling every 15 minutes [33].

The influx of data in the energy sector will lead utility companies to be better equipped to improve operational efficiency and provide a better customer experience through the insights obtained from data analytics. However, they will also face a number of challenges, including

how to handle the large quantities of data, how to promptly build load forecasting models with much higher data granularity and how to analyze growing data sets in a continuous or real-time manner [19]. While there are very successful techniques using ML with data from a single smart meter, scaling these methods leads to numerous challenges mainly related to efficiency and performance. Simply building separate models for each meter is not practical or even feasible at a very large scale [34].

2.2 Machine Learning

This section will provide a basic introduction to the machine learning algorithms that serve as the foundation to chapters 4 and 5.

2.2.1 Artificial Neural Network (ANN)

An artificial neural network is a machine learning algorithm inspired by biological neurons. In their simplest form, they are called a single-layer perceptron and consist of a single hidden layer. However, more complex forms of neural networks involve a higher number of hidden layers and a multi-feature input. As the number of hidden layers grows, ANN can become a deep neural network (DNN). Figure 2.2 depicts a feed-forward neural network where data propagate from the input layer forward to the output layer.

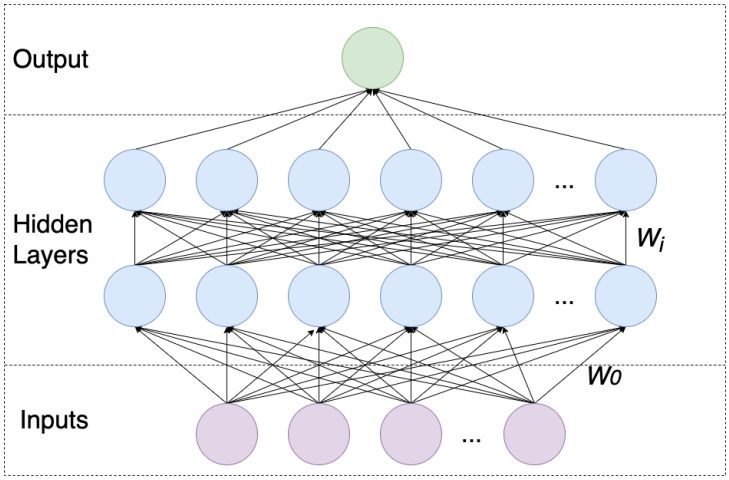


Figure 2.2: Basic Artificial Neural Network Architecture

Each input vector X is multiplied by weights W_0, then at each hidden node (represented by the light blue circles) the multiplied values are summed up and an activation function is applied to the sum. The process is repeated up to and including the output layer. In order

to train the ANN, a chosen loss function is then used to calculate the difference between the actual and the expected output. The backpropagation algorithm is applied to propagate the loss error throughout the network; this process will update the values of the weights W in order to minimize the loss function.

2.2.2 Recurrent Neural Network (RNN)

The recurrent neural network is a special class of ANN that deals with a temporal dimension or sequentiality. RNNs are characterized by the fact that the same weights are shared across time steps, as opposed to feed-forward networks that have different weights across each node [35]. Additionally, at each node, the activation is dependent on the output of all previous time steps, this recurrent input is what serves as the network's memory and is what distinguishes this type of network from a basic feedforward ANN. Figure 2.3 shows the rolled and unrolled representation of an RNN. The unrolled version enables us to visualize the sequential nature of the algorithm and its reliance upon a memory vector.

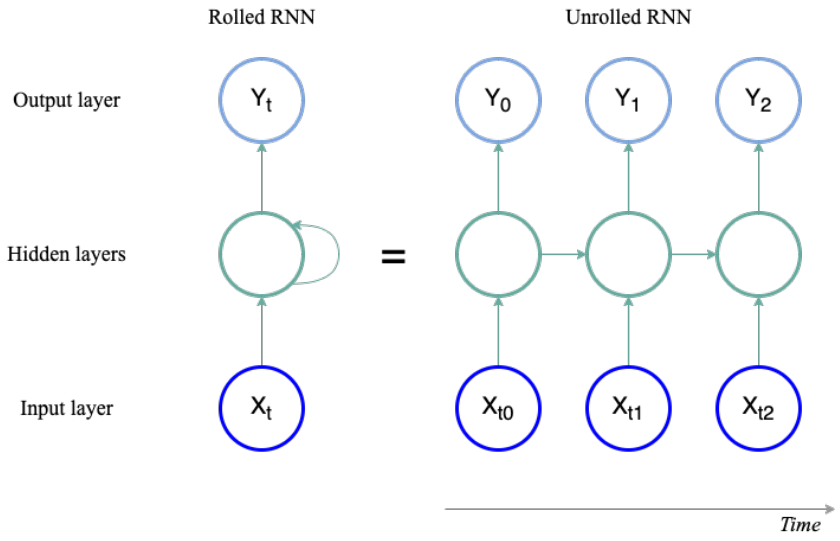


Figure 2.3: Basic Rolled and unrolled RNN Architecture

2.2.3 Sequence-to-Sequence Architecture (S2S)

The S2S architecture was first proposed by Sutskever in 2014 [36]. It is made up of three main components: the encoder, the decoder and the encoded vector. Figure 2.4 shows the basic S2S architecture.

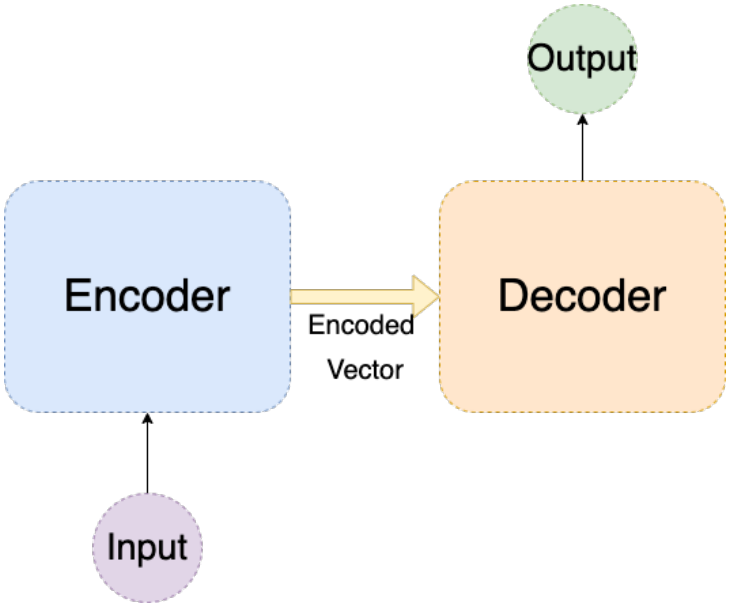


Figure 2.4: Basic Sequence to Sequence Architecture

The encoder is made up of a stack of RNNs or recurrent units such as long short-term memory (LSTM) or gated recurrent units (GRUs). The input is passed to the encoder sequentially and fed forward toward the decoder. The hidden state of each unit is also passed forward and, like in RNNs, it serves as the memory of the network and encapsulates the whole input information. The last hidden state of the encoder becomes the encoded vector, which is essentially an encoded representation of the entire sequential input. The decoder is also a stack of RNN where each unit accepts the previous hidden state (starting with the encoded vector) and previous timestamp output as an input. Each recurrent unit in the decoder also produces a part of the final output and forwards the hidden state. The strength of this model lies in the fact that it can output sequences of variable lengths which is not tied to the length of the input.

2.2.4 Transformer

The transformer architecture was first introduced by Vaswani et al. [37]. The architecture is shown in Figure 2.5. Like S2S, it is capable of outputting sequences of variable lengths and is made up of encoders and decoders, however, the input is not fed to the model sequentially but rather in parallel with multiple inputs at a time.

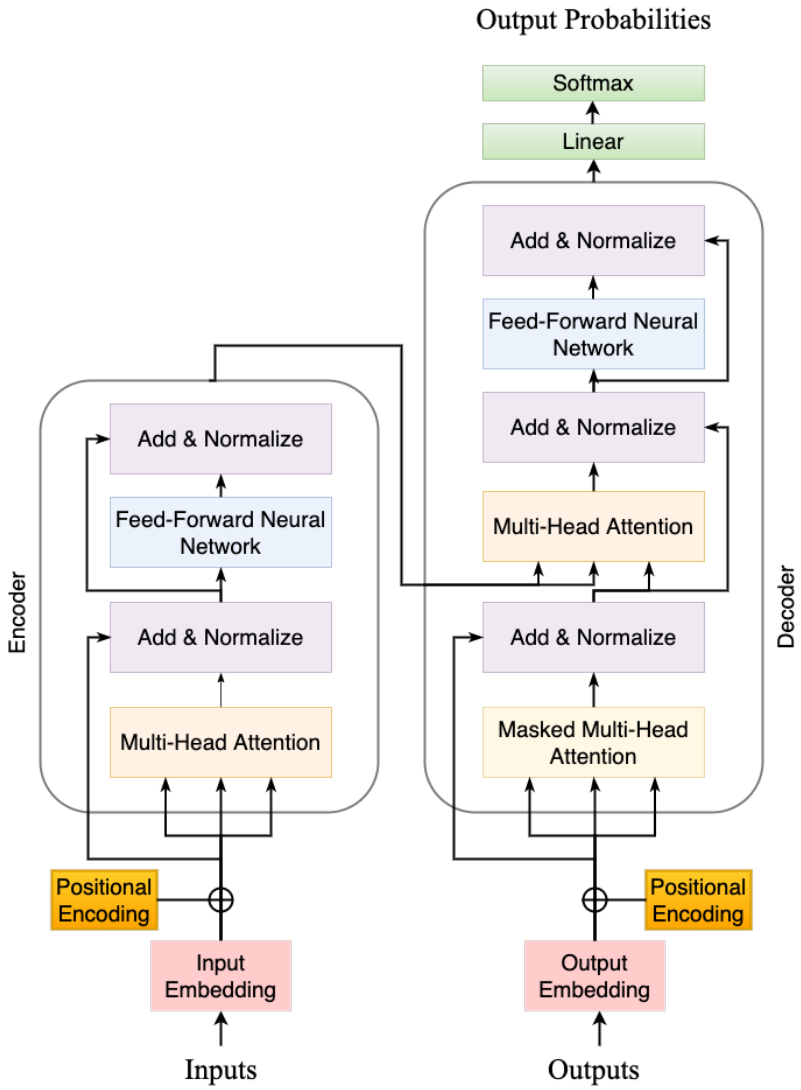


Figure 2.5: Transformer Architecture

Because the transformer was developed for natural language processing (NLP) tasks, the input must first be embedded, meaning that the words are transformed into numbers. Those embedding ensure that closely related words become closely related embedding in order for the input to retain its semantic values. The size of the embedding is dependent on the size of the chosen vocabulary. If a vocabulary of 10,000 words is used, then the size of the embedding will also be 10,000. In order to retain the models' ability to process sequential data, a positional encoding process is added. This adds or modifies the input in such a way that its position within the sequence becomes embedded within itself. Once, the data are embedded and the positional encoding is added, the input is fed to the encoder.

The encoder is made up of two main components, a multi-headed attention module and a feed-forward network. The attention mechanism is used to identify which parts of the input

are most relevant. If multiple encoders are used, the output of the encoder feeds the next one until there are no further encoders. The output of the encoder is an attention weighted encoded representation of the entire input. At this point, this representation is fed to the decoder.

The decoder, is auto-regressive, meaning that it uses previous outputs to create its own output. Therefore, as the output is generated, it is also used as input to the decoder. During the training phase, the expected output sequence is used as an input. The outputs first go through an embedding, just like the encoder and then these values are fed to the first multi-headed attention module. The output of this module is added, normalized, and passed as input to the second multi-head attention module as the values for the attention calculation. This second attention module also receives the encoder output. This process will allow for the attention module to match the decoder's and the encoder's input, this comparison will allow the decoder to determine which of the encoder's inputs it should focus on. Then the output is fed through a feed-forward network.

Lastly, the decoder output undergoes a linear transformation to reduce the dimensionality of the output to the size of the vocabulary used. The output is then passed through a softmax function in order to establish the probability for each vocabulary word. The word with the highest probability is chosen as the output and the process is repeated until the transformer emits an end of sequence token.

Chapter 3

Machine Learning with Big Data: Challenges and Approaches

Everyday, the world produces more and more data. For example, Twitter processes over 70M tweets per day, thereby generating over 8TB daily [38]. ABI Research estimates that by 2020, there will be more than 30 billion connected devices [39]. These Big Data possess tremendous potential in terms of business value in a variety of fields such as health care, biology, transportation, online advertising, energy management, and financial services [7], [8]. However, traditional approaches are struggling when faced with these massive data.

The concept of Big Data is defined by Gartner [24] as high volume, high velocity, and/or high variety data that require new processing paradigms to enable insight discovery, improved decision making, and process optimization. According to this definition, Big Data are not characterized by specific size metrics, but rather by the fact that traditional approaches are struggling to process them due to their size, velocity or variety. The potential of Big Data is highlighted by their definition; however, realization of this potential depends on improving traditional approaches or developing new ones capable of handling such data.

Because of their potential, Big Data have been referred to as a revolution that will transform how we live, work, and think [40]. The main purpose of this revolution is to make use of large amounts of data to enable knowledge discovery and better decision making [40]. The ability to extract value from Big Data depends on data analytics; Jagadish *et al*. [14] consider analytics to be the core of the Big Data revolution.

Data analytics involves various approaches, technologies, and tools such as those from text analytics, business intelligence, data visualization, and statistical analysis. This chapter focusses on machine learning (ML) as a fundamental component of data analytics. The McKinsey Global Institute has stated that ML will be one of the main drivers of the Big Data revolution [41]. The reason for this is its ability to learn from data and provide data driven

insights, decisions, and predictions [42]. It is based on statistics and, similarly to statistical analysis, can extract trends from data; however, it does not require the explicit use of statistical proofs. According to the nature of the available data, the two main categories of learning tasks are: *supervised learning* when both inputs and their desired outputs (labels) are known and the system learns to map inputs to outputs and *unsupervised learning* when desired outputs are not known and the system itself discovers the structure within the data. Classification and regression are examples of supervised learning: in classification the outputs take discrete values (class labels) while in regression the outputs are continuous. Examples of classification algorithms are k-nearest neighbour, logistic regression, and Support Vector Machine (SVM) while regression examples include Support Vector Regression (SVR), linear regression, and polynomial regression. Some algorithms such as neural networks can be used for both, classification and regression. Unsupervised learning includes clustering which groups objects based on established similarity criteria; k-means is an example of such algorithm. Predictive analytics relies on machine learning to develop models built using past data in an attempt to predict the future [43]; numerous algorithms including SVR, neural networks, and Naïve Bayes can be used for this purpose.

A common ML presumption is that algorithms can learn better with more data and consequently provide more accurate results [15]. However, massive datasets impose a variety of challenges because traditional algorithms were not designed to meet such requirements. For example, several ML algorithms were designed for smaller datasets, with the assumption that the entire dataset can fit in memory. Another assumption is that the entire dataset is available for processing at the time of training. Big Data break these assumptions, rendering traditional algorithms unusable or greatly impeding their performance.

A number of techniques have been developed to adapt machine learning algorithms to work with large datasets: examples are new processing paradigms such as MapReduce [44] and distributed processing frameworks such as Hadoop [45]. Branches of machine learning including deep and online learning have also been adapted in an effort to overcome the challenges of machine learning with Big Data.

This chapter first compiles, summarizes, and organizes machine learning challenges with Big Data. In contrast to other research [14], [15], [46],[47], the focus is on linking the identified challenges with the Big Data V dimensions (volume, velocity, variety, and veracity) to highlight the cause-effect relationship. Next, emerging machine learning approaches are reviewed with the emphasis on how they address the identified challenges. Through this process, this study provides a perspective on the domain and identifies research gaps and opportunities in the area of machine learning with Big Data. Although security and privacy are important considerations from an application perspective, they do not impede the execution of machine learning and are

therefore considered to be outside the scope of this research.

The remainder of this chapter is organized as follows: Section 3.1 reviews related work, and Section 3.2 presents machine learning challenges classified according to the Big Data dimensions. An overview of emerging machine learning approaches with discussion about challenges they address is provided in Section 3.3. Section 3.4 aggregates the findings and identifies future research directions. Finally, Section 3.5 concludes the Chapter.

3.1 Related Work

This research highlights the challenges specific or highly relevant to machine learning in the context of Big Data, associates them with the V dimensions, and then provides an overview of how emerging approaches are responding to them. In the existing literature, some researchers have described general machine learning challenges with Big Data [8], [46], [48], [49] whereas others have discussed them in the context of specific methodologies [46, 50].

Najafabadi *et al*. [46] focused on deep learning, but noted the following general obstacles for machine learning with Big Data: unstructured data formats, fast moving (streaming) data, multi-source data input, noisy and poor-quality data, high dimensionality, scalability of algorithms, imbalanced distribution of input data, unlabelled data, and limited labeled data. Similarly, Sukumar [48] identified three main requirements: designing flexible and highly scalable architectures, understanding statistical data characteristics before applying algorithms; and finally, developing ability to work with larger datasets. Both studies, Najafabadi *et al*. [46] and Sukumar [48] reviewed aspects of machine learning with Big Data; however, they did not attempt to associate each identified challenge with its cause. Moreover, their discussions are on a very high level without presenting related solutions. In contrast, our work includes a thorough discussion of challenges, establishes their relations with Big Data dimensions, and presents an overview of solutions that mitigate them.

Qiu *et al.* [49] presented a survey of machine learning for Big Data, but they focused on the field of signal processing. Their study identified five critical issues (large scale, different data types, high speed of data, uncertain and incomplete data, and data with low value density) and related them to Big Data dimensions. Our study includes a more comprehensive view of challenges, but similarly relates them to the V dimensions. Furthermore, Qiu *et al.* [49] also identified various learning techniques and discussed representative work in signal processing for Big Data. Although they do a great work of identifying existing problems and possible solutions, the lack of categorization and direct relationship between each approach and its challenges makes it difficult to make an informed decision in terms of which learning paradigm or solution would be best for a specific use case or scenario. Consequently, in our

work emphasis is on establishing correlation between solutions and challenges.

Al-Jarrah *et al.* [8] reviewed machine learning for Big Data focussing on the efficiency of large-scale systems and new algorithmic approaches with reduced memory footprint. Although they mentioned various Big Data hurdles, they did not present a systematic view as is done in this work. Al-Jarrah *et al.* were interested in the analytical aspect, and methods for reducing computational complexity in distributed environments were not considered. This work, on the other hand, considers both the analytical aspect and computational complexity in distributed environments.

Existing studies have effectively discussed the obstacles encountered by specific techniques such as deep learning [46], [50]. However, these studies focussed on a narrow aspect of machine learning; a more comprehensive view of challenges and approaches in the Big Data context is needed.

Similar to our work, Gandomi and Haider categorized challenges in accordance with the Big Data Vs [51]. However, their characterization is general and not in terms of machine learning.

Surveys on platforms for Big Data analytics have also been presented [52], [53]. Singh and Reddy [52] considered vertical and horizontal scaling platforms. They discussed the advantages and disadvantages of different platforms in terms of attributes such as scalability, I/O performance, fault tolerance, real-time processing, and iterative task support. Similarly, de Almeida and Bernardino [53] reviewed open source platforms including Apache Mahout, massive online analysis (MOA), the R Project, Vowpal, Pegasos, and GraphLab. These studies reviewed and compared existing platforms, while the present study relates these platforms to the challenges they address. Moreover, in this work, Big Data platforms are just one category of reviewed solutions.

The challenges of data mining with Big Data have been explored in the literature [54], [55]. Fan and Bifet [54] focused on challenges for data mining with Big Data and, as opposed to this work, they do not classify those challenges nor provide possible solutions. The work of Wu *et al.* [55], categorized the challenges, but their categorization is according to three tiers: Tier I (Big Data mining platforms), Tier II (Semantics and application knowledge), and Tier III (Big Data mining algorithms). In contrast, the categorization in this study is according to the V dimensions. Whereas Wu *et al.* considered data mining, this study deals with machine learning. Moreover, the present study relates Big Data solutions to the challenges that they address.

To understand the origin of machine learning challenges, the present work categorizes them using the Big Data definition. In addition, various machine learning approaches are reviewed, and how each approach is capable of addressing known challenges is discussed. This enables

researchers to make better informed decision regarding which learning paradigm or solution to use based on the specific Big Data scenario. It also makes it possible to identify research gaps and opportunities in the domain of machine learning with Big Data. Consequently, this work serves as a comprehensive foundation and facilitator for future research.

3.2 Machine Learning Challenges originating from Big Data definition

Big Data are often described by its dimensions, which are referred to as its Vs. Earlier definitions of Big Data focussed on three Vs [56] (volume, velocity, and variety); however, a more commonly accepted definition now relies upon the following four Vs [57]: volume, velocity, variety, and veracity. It is important to note that other Vs can also be found in the literature. For example, value is often added as a 5th V [54], [58]. However, value is defined as the desired outcome of Big Data processing [59] and not as defining characteristics of Big Data itself. For this reason, this work considers only the four dimensions that characterize Big Data [60]. This provides an opportunity to relate challenges directly to the defining characteristics of Big Data, rendering the origin and cause of each explicitly. This section identifies machine learning challenges and associates each challenge with a specific dimension of Big Data. Figure 3.1 illustrates the dimensions of Big Data along with their associated challenges as further discussed in the following subsections.

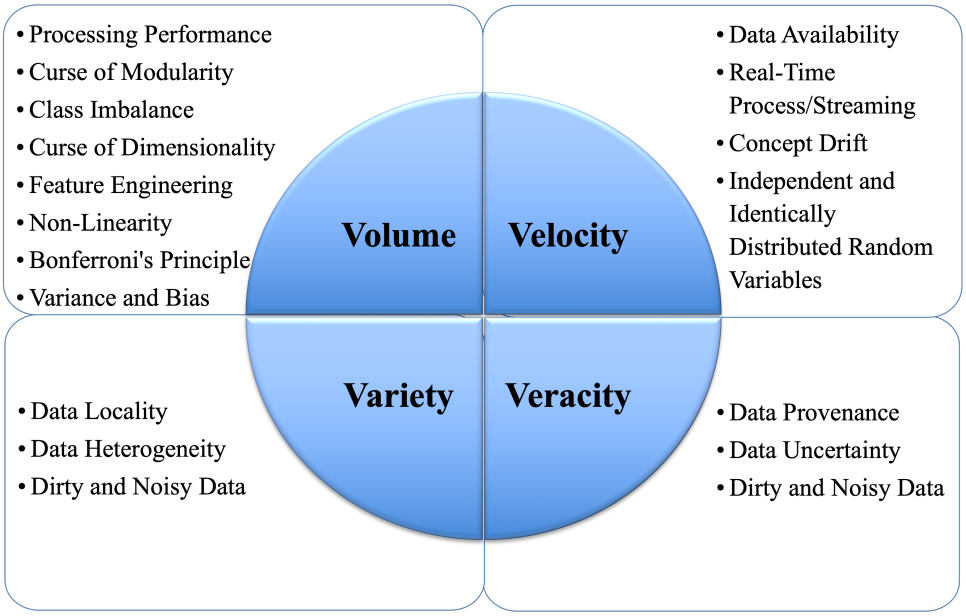


Figure 3.1: Big Data characteristics with associated challenges

3.2.1 Volume

The first and the most talked about characteristic of Big Data is volume: it is the amount, size, and scale of the data. In the machine learning context, size can be defined either vertically by the number of records or samples in a dataset or horizontally by the number of features or attributes it contains. Furthermore, volume is relative to the type of data: a smaller number of very complex data points may be considered equivalent to a larger quantity of simple data [51]. This is perhaps the easiest dimension of Big Data to define, but at the same time, it is the cause of numerous challenges. The following subsections discuss machine learning challenges caused by volume.

3.2.1.1 Processing Performance

One of the main challenges encountered in computations with Big Data comes from the simple principle that scale, or volume, adds computational complexity. Consequently, as the scale becomes large, even trivial operations can become costly. For example, the standard support vector machine (SVM) algorithm has a training time complexity of $O(m^3)$ and a space complexity of $O(m^2)$ [61], where m is the number of training samples. Therefore, an increase in the size m will drastically affect the time and memory needed to train the SVM algorithm and may even become computationally infeasible on very large datasets. Many other ML algorithms also exhibit high time complexity: for example, the time complexity of principal component analysis is $O(mn^2+n^3)$, that of logistic regression $O(mn^2+n^3)$, that of locally weighted linear regression $O(mn^2+n^3)$, and that of Gaussian discriminative analysis $O(mn^2+n^3)$ [62], where m is the number of samples and n the number of features. Hence, for all these algorithms, the time needed to perform the computations will increase exponentially with increasing data size and may even render the algorithms unusable for very large datasets.

Moreover, as data size increases, the performance of algorithms becomes more dependent upon the architecture used to store and move data. Parallel data structures, data partitioning and placement, and data reuse become more important with growth in data size [63]. Resilient distributed datasets (RDDs) [63] are an example of a new abstraction for in-memory computations on large clusters; RDDs are implemented in the Spark cluster computing framework [64]. Therefore, not only does data size affect performance, but it also leads to the need to re-think the typical architecture used to implement and develop algorithms.

3.2.1.2 Curse of Modularity

Many learning algorithms rely on the assumption that the data being processed can be held entirely in memory or in a single file on a disk [65]. Multiple classes of algorithms are designed

on strategies and building blocks that depend on the validity of this assumption. However, when data size leads to the failure of this premise, entire families of algorithms are affected. This challenge is referred to as the curse of modularity [47].

One of the approaches brought forward as a solution for this curse is MapReduce, a scalable programming paradigm for processing large datasets by means of parallel execution on a large number of nodes. Some machine learning algorithms are inherently parallel and can be adapted to the MapReduce paradigm, whereas others are difficult to decompose in a way that can take advantage of large numbers of computing nodes. Grolinger *et al.* [15] have discussed challenges for MapReduce in Big Data. The three main categories of algorithms that encounter the curse of modularity when attempting to use the MapReduce paradigm include iterative graph, gradient descent, and expectation maximization algorithms. Their iterative nature together with their dependence on in-memory data create a disconnect with the parallel and distributed nature of MapReduce. This leads to difficulties in adapting these families of algorithms to MapReduce or to another distributed computation paradigm.

Consequently, although some algorithms such as k-means can be adapted to overcome the curse of modularity through parallelization and distributed computing, others are still bounded or even unusable with certain paradigms.

3.2.1.3 Class Imbalance

As datasets grow larger, the assumption that the data are uniformly distributed across all classes is often broken [66]. This leads to a challenge referred to as class imbalance: the performance of a machine learning algorithm can be negatively affected when datasets contain data from classes with various probabilities of occurrence. This problem is especially prominent when some classes are represented by a large number of samples and some by very few.

Class imbalance is not exclusive to Big Data and has been the subject of research for more than a decade [67]. Experiments performed by Japkowicz and Stephen [67] have shown that the severity of the imbalance problem depends on task complexity, the degree of class imbalance, and the overall size of the training set. They suggest that in large datasets, there is a good chance that classes are represented by a reasonable number of samples; however, to confirm this observation, evaluations of real-world Big Data sets are needed. On the other hand, the complexity of Big Data tasks is expected to be high, which could result in severe impacts from class imbalance.

It is to expect that this challenge would be more common, severe, and complex in the Big Data context because the extent of imbalance has immense potential to grow due to increased data size. The same authors, Japkowicz and Stephen [67], showed that decision trees, neural networks, and support vector machine algorithms are all very sensitive to class imbalance.

Therefore, their unaltered execution in the Big Data context without addressing class imbalance may produce inadequate results. Similarly, Baughman *et al.* [68] considered extreme class imbalance in gamification and demonstrated its negative effects on Watson machine learning.

Consequently, in the Big Data context, due to data size, the probability that class imbalance will occur is high. In addition, because of the complex problems embedded in such data, the potential effects of class imbalance on machine learning are severe.

3.2.1.4 Curse of Dimensionality

Another issue associated with the volume of Big Data is the curse of dimensionality [69] which refers to difficulties encountered when working in high dimensional space. Specifically, the dimensionality describes the number of features or attributes present in the dataset. The Hughes effect [70] states that for a training set of static size, the predictive ability and effectiveness of an algorithm decreases as the dimensionality increases. Therefore, as the number of features increases, the performance and accuracy of machine learning algorithms degrades. This can be explained by the breakdown of the similarity-based reasoning upon which many machine learning algorithms rely [69]. Unfortunately, the greater the amount of data available to describe a phenomenon, the greater becomes the potential for high dimensionality because there are more prospective features. Consequently, as the volume of Big Data increases, so does the likelihood of high dimensionality.

In addition, dimensionality affects processing performance: the time and space complexity of ML algorithms is closely related to data dimensionality [62]. The time complexity of many ML algorithms is polynomial in the number of dimensions. As already mentioned, the time complexity of the principal component analysis is $O(mn^2+n^3)$ and that of logistic regression $O(mn^2+n^3)$, where m is the number of samples and n is the number of dimensions.

3.2.1.5 Feature Engineering

High dimensionality is closely related to another volume challenge: feature engineering. It is the process of creating features, typically using domain knowledge, to make machine learning perform better. Indeed, the selection of the most appropriate features is one of the most time consuming pre-processing tasks in machine learning [46]. As the dataset grows, both vertically and horizontally, it becomes more difficult to create new, highly relevant features. Consequently, in a manner similar to dimensionality, as the size of the dataset increases, so do the difficulties associated with feature engineering.

Feature engineering is related to feature selection: whereas *feature engineering* creates new features in an effort to improve learning outcomes, *feature selection* (dimensionality reduction)

aims to select the most relevant features. Although feature selection reduces dimensionality and hence has the potential to reduce ML time, in high dimensions it is challenging due to spurious correlations and incidental endogeneity (correlation of an explanatory variable with the error term) [71].

Overall, both feature selection and engineering are still very relevant in the Big Data context, but, at the same time they become more complex.

3.2.1.6 Non-Linearity

Data size poses challenges to the application of common methodologies used to evaluate dataset characteristics and algorithm performance. Indeed, the validity of many metrics and techniques relies upon a set of assumptions, including the very common assumption of linearity [72]. For example, the correlation coefficient is often cited as a good indicator of the strength of the relationship between two or more variables. However, the value of the coefficient is only fully meaningful if a linear relationship exists between these variables. An experiment conducted by Kiang [73] showed that the performance of neural networks and logistic regression is very negatively affected by non-linearity. Although this problem is not exclusive to Big Data, non-linearity can be expected to be more prominent in large datasets.

The challenge of non-linearity in Big Data also stems from the difficulties associated with evaluating linearity. Linearity is often evaluated using graphical techniques such as scatterplots; however, in the case of Big Data, the large number of points often creates a large cloud, making it difficult to observe relationships [72] and assess linearity.

Therefore, both the difficulty of assessing linearity and the presence of non-linearity pose challenges to the execution of machine learning algorithms in the context of Big Data.

3.2.1.7 Bonferonni's Principle

Bonferonni's principle [74] embodies the idea that if one is looking for a specific type of event within a certain amount of data, the likelihood of finding this event is high. However, more often than not, these occurrences are bogus, meaning that they have no cause and are therefore meaningless instances within a dataset. This statistical challenge is also often described as spurious correlation [51]. In statistics, the Bonferonni correction theorem provides a means of avoiding those bogus positive searches within a dataset. It suggests that if testing m hypotheses with a desired significance of α, each individual hypothesis should be tested at a significance level of α/m [75].

However, the incidences of such phenomena increase with data size, and as data become exponentially bigger, the chances of finding an event of interest, legitimate or not, is bound to

increase. Recently, Calude and Longo [76] have discussed the impact and incidence of spurious correlations in Big Data. They have shown that given a large enough volume, most correlations tend to be spurious. Therefore, including a means of preventing those false positives is important to consider in the context of machine learning with Big Data.

3.2.1.8 Variance and Bias

Machine learning relies upon the idea of generalization; through observations and manipulations of data, representations can be generalized to enable analysis and prediction. Generalization error can be broken down into two components: variance and bias [77]: Figure 3.2 illustrates the relationship between them. *Variance* describes the consistency of a learner's ability to predict random things, whereas *bias* describes the ability of a learner to learn the wrong thing [69]. Ideally, both the variance and the bias error should be minimized to obtain an accurate output. However, as the volume of data increases, the learner may become too closely biased to the training set and may be unable to generalize adequately for new data. Therefore, when dealing with Big Data, caution should be taken as bias can be introduced, compromising the ability to generalize.

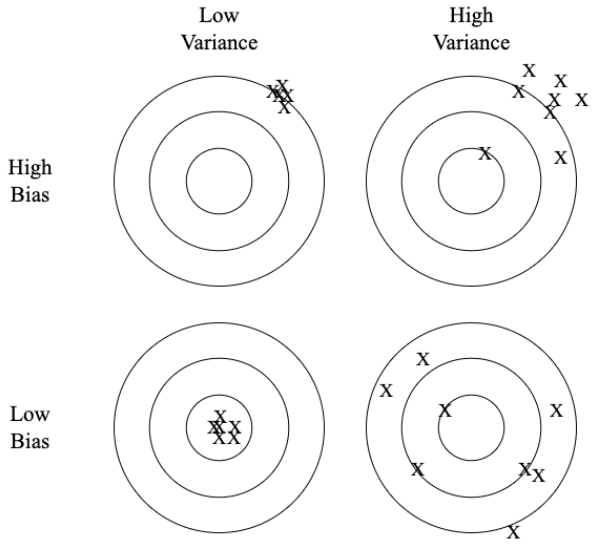


Figure 3.2: Variance and bias

Regularization refers to techniques that aim to improve generalization and reduce overfitting; examples of regularization techniques include early stopping, Lasso, and Ridge [78]. Although these techniques improve generalization, they also introduce additional parameters that must be tuned to achieve good fit to unseen data. This is often done using approaches such as cross-validation, possibly with grid search; however, those require additional processing

time, especially in the case of large datasets. Regularization techniques are well established in machine learning, but further investigation is needed with respect to their efficiency with Big Data.

3.2.2 Variety

The variety of Big Data describes not only the structural variation of a dataset and of the data types that it contains, but also the variety in what it represents, its semantic interpretation [14] and its sources. Although not as many as for other V dimensions, the challenges associated with this dimension have substantial impact.

3.2.2.1 Data Locality

The first challenge associated with variety is data locality [74]. Machine learning algorithms once again assume that the entire dataset is found in memory or in a single disk file [47]. However, in the case of Big Data, this may not be possible due to sheer size; not only do the data not fit into memory, but they are commonly distributed over large numbers of files residing in different physical locations. Traditional machine learning would first require data transfer to the computing location. With large datasets, transfer would result in processing latency and could cause massive network traffic.

Consequently, an approach of bringing computation to data as opposed to bringing data to computation has emerged. This is based on the premise that moving computation is cheaper, in terms of time and bandwidth, than moving data. This approach is especially prominent with Big Data. The MapReduce paradigm also uses it: map tasks are executed on the nodes where data reside, with each map task processing its local data. Moreover, a large number of NoSQL data stores adapt this model; as distributed storage solutions, they store data over a large number of nodes and then use the MapReduce paradigm to bring computation to data [79]. However, as already mentioned, MapReduce-based approaches encounter difficulties when working with highly iterative algorithms.

With small datasets, physical location is a non-issue; however, with Big Data, data locality is a paramount challenge that must be addressed in any successful Big Data system.

3.2.2.2 Data Heterogeneity

Big Data analytics often involve integrating diverse data from several sources. These data may be diverse in terms of data type, format, data model, and semantics. Two main heterogeneity categories can be recognized: syntactic and semantic heterogeneity.

Syntactic heterogeneity refers to diversity in data types, file formats, data encoding, data model, and similar. To carry out analytics with integrated datasets, these syntactic variations must be reconciled [14]. Machine learning often requires a data pre-processing and cleaning step to configure data to fit within a specific model. However, with data coming from different sources, these data are likely formatted differently. Furthermore, the data to be processed may be of completely different types; for example, images may need to be processed along with categorical and numerical data. This causes difficulties for machine learning algorithms because they are not designed to recognize various types of representations at one time and to create efficient unified generalizations.

Semantic heterogeneity refers to differences in meanings and interpretations. As with syntactic, semantic heterogeneity increases in the case of Big Data when a number of datasets developed by different parties are integrated [80]. Again, machine learning approaches were not developed to handle semantically diverse data, and therefore heterogeneity must be resolved before applying such approaches.

In statistics, heterogeneity also refers to differences in statistical properties among the different parts of an overall dataset. Although present in small datasets, this challenge is enlarged in Big Data because datasets typically involve parts coming from different sources. This statistical heterogeneity breaks the common machine learning assumption that statistical properties are similar across a complete dataset.

Both syntactic and semantic heterogeneity as well as statistical heterogeneity have been active research topics for a long time, but with the emergence of Big Data, they have attracted renewed attention [80]. The business value of data analytics typically involves correlating diverse datasets, and integration is crucial for carrying out machine learning over such datasets.

3.2.2.3 Dirty and Noisy Data

According to Ratner [72], data possess their own set of distinct features that can be used for characterization:

- *Condition* defines the readiness of the data for analysis.
- *Location* refers to where the data physically reside.
- *Population* describes the entities and their sets of common attributes that together form the dataset.

Big Data are typically described as ill-conditioned due to the amount of time and resources necessary to get them ready for analysis. They also come from various locations and unknown

populations. The combination of these properties leads to Big Data often being described as dirty.

Fan *et al.* [71] referred to such data as noisy data; they contain various types of measurement errors, outliers, and missing values. They discussed noise accumulation, which is especially severe with the high dimensionality typical in Big Data. It is important to note that Fan *et al.* considered noisy data one of the three main challenges of Big Data analysis.

Swan [81] suggested that data analysis should include a step to extract signal from noise directly following the steps of data collection and integration. She also recognized that Big Data may be too noisy to produce meaningful results.

The studies described above demonstrate the importance of dealing with noise in the context of generic Big Data analysis. Likewise, noise needs to be considered in machine learning with Big Data.

3.2.3 Velocity

The velocity dimension of Big Data refers not only to the speed at which data are generated, but also the rate at which they must be analyzed. With the omnipresence of smartphones and real-time sensors and the impending need to interact quickly with our environment through the development of technologies such as smart homes, the velocity of Big Data has become an important factor to consider.

3.2.3.1 Data Availability

Historically, many machine learning approaches have depended on data availability, meaning that before learning began, the entire dataset was assumed to be present. However, in the context of streaming data, where new data are constantly arriving, such a requirement cannot be fulfilled. Moreover, even data arriving at non-real-time intervals may pose a challenge.

In machine learning, a model typically learns from the training set and then performs the learned task, for example classification or prediction, on new data. In this scenario, the model does not automatically learn from newly arriving data, but instead carries out the already learned task on new data. To accommodate the knowledge embedded in new data, these models must be retrained. Without retraining, they may become outdated and cease to reflect the current state of the system.

Therefore, to adapt to new information, algorithms must support incremental learning [82], sometimes referred to as sequential learning, which is defined as an algorithm's ability to adapt its learning based on the arrival of new data without the need to retrain on the complete dataset. This approach does not assume that the entire training set is available before learning begins,

but processes new data as they arrive. Although incremental learning is a relatively old concept, it is still an active research area due to the difficulty of adapting some algorithms to continuously arriving data [83].

3.2.3.2 Real-Time Processing/Streaming

Similar to the already discussed data availability challenge, traditional machine learning approaches are not designed to handle constant streams of data [51], which leads to another velocity dimension challenge - the need for real-time processing. This is subtly different from the data availability challenge: whereas data availability refers to the need to update the ML model as new data arrive, real-time processing refers to the need for real-time or near-real-time processing of fast-arriving data. The business value of real-time processing systems lies in their ability to provide instantaneous reaction; developers of algorithmic trading, fraud detection, and surveillance systems have been especially interested in such solutions [15].

The importance of real-time processing in today's era of sensors, mobile devices, and IoT has resulted in the emergence of a number of streaming systems; examples include Twitter's Storm [84] and Yahoo's S4 [85]. Although those systems have seen great success in real-time processing, they do not include sophisticated or diverse ML, but users can add ML features using external languages or tools.

The need exists to merge these streaming solutions with machine learning algorithms to provide instantaneous results; however, the complexity of such algorithms and the sparse availability of online learning solutions make this a difficult task.

3.2.3.3 Concept Drift

Big Data are non-stationary; new data are arriving continuously. Consequently, acquiring the entire dataset before processing it is not possible, meaning that it cannot be determined whether the current data follow the same distribution as future data. This leads to another interesting challenge in machine learning with Big Data: concept drift [47]. *Concept drift* can be formally defined as changes in the conditional distribution of the target output given the input, while the distribution of the input itself may remain unchanged [86]. Specifically, this leads to a problem that occurs when machine learning models are built using older data that no longer accurately reflect the distribution of new data [87]. For example, energy consumption and demand prediction models can be built using data from electricity meters [88], but when buildings are retrofitted to improve their energy efficiency, the present model does not accurately represent the new energy characteristics. Sliding window is a possible way of dealing with concept drift: the model is built using only the samples from the training window which is moved to include

only the most recent samples. Windowing approach assumes that the most recent data are more relevant which may not always be true [86].

There exist various types of concept drift: incremental, gradual, sudden, and recurring [89], each bringing its own set of issues. However, the challenges typically lie in quickly detecting when concept drift is occurring and effectively handling the model transition during these changes. Like several already mentioned concepts, concept drift is not a new issue; mentions of it date back to 1986 [86]. However, the advent and nature of Big Data have increased frequency of its occurrence and have rendered some previous methodologies unusable. For example, Lavaire *et al.* [90] conducted an experiment on the influence of high dimensional Big Data on existing concept drift mitigation techniques. Their conclusions were that algorithm performance was highly degraded by the changes in the data. Therefore, finding new means to handle concept drift in the context of Big Data is an important task for the future of machine learning.

3.2.3.4 Independent and Identically Distributed Random Variables

Another common assumption in machine learning is that random variables are independent and identically distributed (i.i.d.) [91]; it simplifies underlying methods and improves convergence. In other words, i.i.d. assumes that each random variable has the same probability distribution as the others and that all are mutually independent. In reality, this may or may not be true. Moreover, some algorithms also depend on other distributions; for example the Markov sequence assumes that probability distribution of the next state depends only on the current state [92].

Nonetheless, Big Data by their very nature may prevent reliance on i.i.d. assumption based on the following [47]:

- i.i.d. requires data to be in random order while many datasets have a pre-existing non-random order. A typical solution would be to randomize the data before applying the algorithms. However, when dealing with Big Data, this becomes a challenge of its own and is often impractical.

- By their very nature, Big Data are fast and continuous. It is therefore not realistic to randomize a dataset that is still incomplete, nor is it possible to wait for all the data to arrive.

Dundar *et al.* [93] have shown that many typical machine learning algorithms such as back-propagation neural networks and support vector machines depend upon this assumption and could benefit greatly from a way of accounting for it. The high likelihood of a broken i.i.d. assumption with Big Data makes this challenge an important one to address.

3.2.4 Veracity

The veracity of Big Data refers not only to the reliability of the data forming a dataset, but also, as IBM has described, to the inherent unreliability of data sources [51]. The provenance and quality of Big Data together define the veracity component [94], but also pose a number of challenges as discussed in the following subsections.

3.2.4.1 Data Provenance

Data provenance is the process of tracing and recording the origin of data and their movements between locations [95]. Recorded information, the provenance data, can be used to identify the source of processing error since it identifies all steps, transactions, and processes undergone by invalid data, thus providing contextual information to machine learning. It is therefore important to capture and retain this metadata [14].

However, as pointed out by Wang *et al*. [94], in the context of Big Data, the provenance dataset itself becomes too large, therefore, while these data provide excellent context to machine learning, the volume of these metadata creates its own set of challenges. Moreover, not only is this dataset too large, but the computational cost of carrying this overhead becomes overwhelming [94]. Although, certain methods have been brought forward to capture data provenance for specific data processing paradigms, such as the Reduce and Map Provenance (RAMP) developed for MapReduce as an extension for Hadoop [96], the added burden of provenance generally adds to the already high complexity and computational cost of machine learning with Big Data. Consequently, as provenance data provide a way to establish the veracity of Big Data, means of balancing its computational overhead and cost with the veracity value are needed.

3.2.4.2 Data Uncertainty

Data are now being gathered about various aspects of our lives in different ways; however, the means and methods used to gather data can introduce uncertainty and therefore impact the veracity of a dataset.

For example, sentiment data are being collected through social media [97], but although these data are highly important because they contain precious insights into subjective information, the data themselves are imprecise. The certainty and accuracy of this type of data is not objective because it relies only upon human judgment [52]. The lack of objectivity, or of absolute truth, within the data makes it difficult for a machine learning algorithm to learn from it.

Another recent method of capturing data is crowdsourcing; it solicits services or ideas from a large group of people. The data obtained from crowdsourcing, more particularly those gathered through participatory sensing, contain an even higher degree of uncertainty than sentiment data [14].

Moreover, inherent uncertainties exist in various types of data, such as weather or economic data for example, and even the most sophisticated data pre-processing methods cannot expunge this intrinsic unpredictability [27]. Once again, machine learning algorithms are not designed to handle this kind of imprecise data, thus resulting in another set of unique challenges for machine learning with Big Data.

3.2.4.3 Dirty and Noisy Data

Furthermore, in addition to being imprecise, data can also be noisy [98]. For example, the labels or contextual information associated with the data may be inaccurate, or readings could be spurious. From the machine learning perspective this is different from imprecise data; having an unclear picture is different from having the wrong picture, although it may yield similar results. Noise and dirtiness come from various sources and are related to variety; many of the causes related to variety have been discussed in previous sections. However, the noise challenge associated with crowdsourcing has yet to be discussed.

Crowdsourcing leads to uncertainty, especially when used for participatory sensing, but it can also lead to noisy data because it makes use of human judgment to assign labels to data. Moreover, the incorrect label can be either purposely or accidently assigned. The number of incorrect or noisy labels not only influences data veracity, but can also affect the performance of machine learning by potentially providing them with improperly labelled data.

Dirty and noisy data are not unique to Big Data, but the means by which they can be handled may not be easily adaptable to large datasets.

3.3 Approaches

In response to the presented challenges, various approaches have been developed. Although designing entirely new algorithms would appear to be a possible solution [99], researchers have mostly preferred other methods. Many approaches have been suggested and surveys have been published on specific categories of solutions; examples include surveys on platforms for Big Data analytics [52], [53] and review of data mining with Big Data [55]. This chapter reviews and organizes various proposed machine learning approaches and discusses how they address the identified challenges. The big picture of approach-challenge correlations is presented in

Table 3.1; it includes a list of approaches along with the challenges that each best addresses. The symbol ✓ indicates a high degree of remedy while '*' represents partial resolution.

APPROACHES				CHALLENGES																		
				VOLUME								VARIETY			VELOCITY				VERACITY		
				Processing Performance	Curse of Modularity	Class Imbalance	Curse of Dimensionality	Feature Engineering	Non-linearity	Bonferroni's Principle	Variance and Bias	Data locality	Data Heterogeneity	Dirty and noisy Data	Data availability	Real-time Processing/Streaming	Concept drift	I.i.d	Data Provenance	Data Uncertainty	Dirty and Noisy Data
MANIPULATIONS	Data Manipulations	Dimensionality Reduction		✓			✓															
		Instance Selection		✓	✓																	
		Data Cleaning												✓								✓
	Processing Manipulations	Vertical Scaling		✓															*			
		Hori-zontal Scaling	Batch-oriented	✓	✓		*					✓							*			
			Stream-oriented	✓	✓										✓	✓			*			
	Algorithm Manipulations	Algorithm Modifications		✓	*		*					✓				✓						
		Algorithm Mod. with new Paradigm		✓	*		*					✓				✓						
LEARNING PARADIGMS	Deep Learning							✓	✓				✓	*						*	*	
	Online Learning			✓	✓	*						✓		*	✓	✓	*	✓			*	
	Local Learning			✓	✓	✓					✓	✓										
	Transfer Learning					✓							✓	*						*	*	
	Lifelong Learning			✓		✓							✓	*	✓	✓	*			*	*	
	Ensemble Learning			✓	✓												✓					

Table 3.1: Machine Learning Approaches and the Challenges they address.

As it can be seen from the table, there are two main categories of solutions. The first category relies on data, processing, and algorithm manipulations to handle Big Data. The second category involves the creation and adaptation of different machine learning paradigms and the modification of existing algorithms for these paradigms.

In addition to these two categories, it is important to note several machine learning as a service offerings: Microsoft Azure Machine Learning, now part of Cortana Intelligence Suite [100]; Google Cloud Machine Learning Platform [101]; Amazon Machine Learning[102]; and IBM Watson Analytics [103]. Because these services are backed up by powerful cloud providers, they offer not only scalability but also integration with other cloud platform services. However, at the moment, they support a limited number of algorithms compared to

the R language [104], MATLAB [105], or Weka [106]. Moreover, computation happens on cloud resources, which requires data transfer to remote nodes. With Big Data, this results in high network traffic and may even become infeasible due to time or bandwidth requirements. Because these ML services are proprietary, information about their underlying technologies is very limited; therefore, this study does not discuss them further.

The following subsections introduce techniques and methodologies being developed and used to handle the challenges associated with machine learning with Big Data. First, manipulation techniques used in conjunction with existing algorithms are presented. Second, various machine learning paradigms that are especially well suited to handle Big Data challenges are discussed.

3.3.1 Manipulations for Big Data

Data analytics using machine learning relies on an established suite of events, also known as the *data analytics pipeline*. The approaches presented in this section discuss possible manipulations in various steps of the existing pipeline. The purpose of these modifications is to respond to the challenges of machine learning with Big Data. Figure 3.3 shows a representation of the pipeline based on the work of Labrinidis and Jagadish [107], along with the three types of manipulations to be discussed in this section: data manipulations, processing manipulations, and algorithm manipulations. These three categories, along with their corresponding sub-categories and sample solutions, are presented in Figure 3.4. The examples included are only representatives and in no way provide a comprehensive list of solutions.

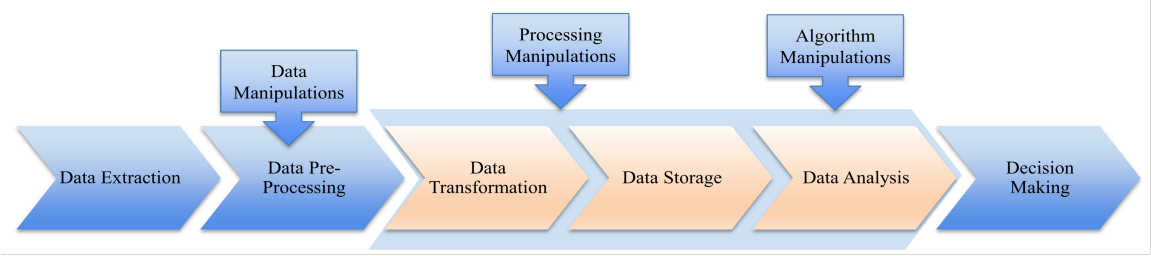


Figure 3.3: Data Analytics Pipeline

3.3.1.1 Data Manipulations

One of the first manipulations to be attempted in an effort to adapt Big Data for machine learning is to try to modify the data in order to mimic non-Big Data. This modification takes place in the data pre-processing stage of the pipeline, as illustrated in Figure 3.3.

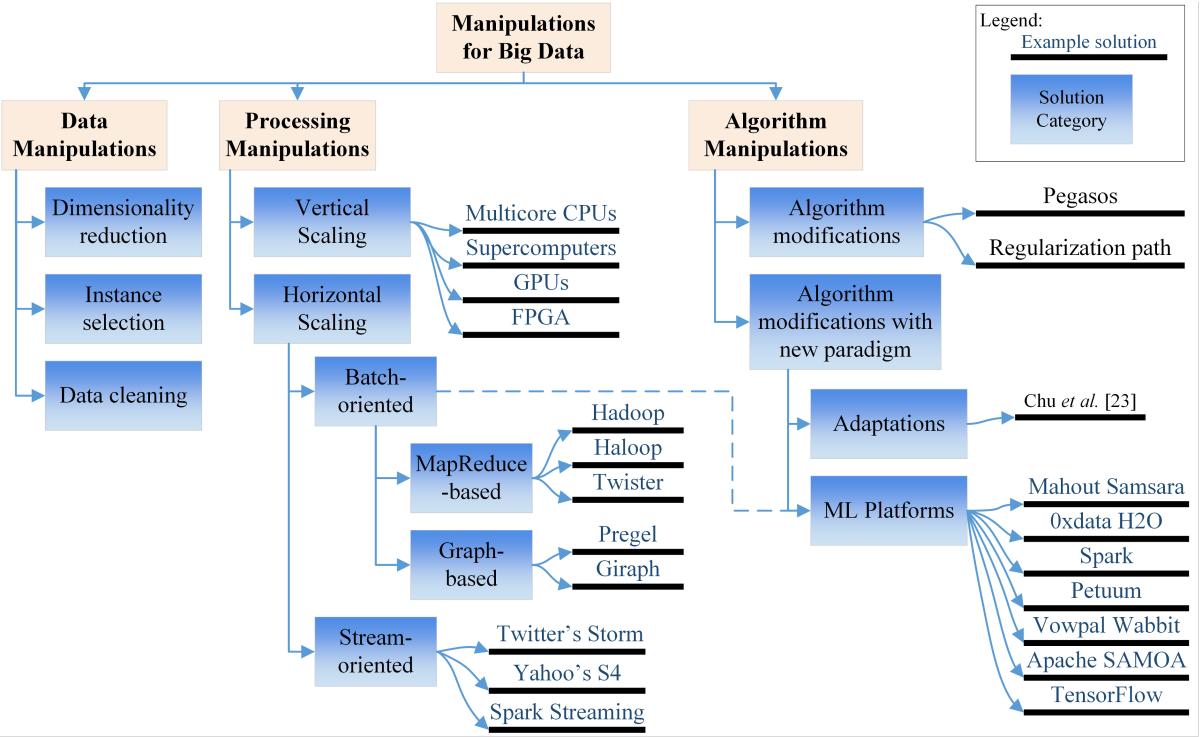


Figure 3.4: Manipulations for Big Data

Two of the most important data-related aspects affecting machine learning performance are high dimensionality (wide datasets) and large number of samples (high datasets). Therefore, two intuitive data manipulations for learning with Big Data are dimensionality reduction and instance selection as shown in Figure 3.4. The term *data reduction* sometimes refers to both these manipulations, but occasionally specifically denotes instance selection. Additionally, data clearing is another important aspect of data manipulations.

Dimensionality reduction aims to map high dimensionality space onto lower-dimensionality one without significant loss of information. A variety of means exists to reduce dimensions in the context of Big Data. One popular, but very old technique (it originates from 1901) is *principal component analysis* (PCA). PCA belongs to the family of linear mapping techniques: orthogonal transformations are applied to transform a set of possibly correlated variables into a set of linearly uncorrelated variables, called principal components. The first principal component accounts for the largest proportion of the variability in the data, the second one has the next highest variance and is orthogonal to the first, and so on. Thus, choosing only the first p principal components can reduce dimensionality.

Examples of non-linear dimensionality reduction techniques, sometimes referred to as *manifold learning*, include kernel PCA, Laplacian Eigenmaps, Isomap, locally linear embedding

(LLE), and Hessian LLE [108]. Random projection is another well-developed dimensionality reduction technique [109]. The idea behind it is to make use of random unit matrices to project the original dataset onto a lower-dimensional space. This technique has been used for a variety of data types such as text and images. However, it is limited to locally available static data. Therefore, although interesting, random projection addresses only the issues related with high dimensionality and is unable to mitigate other challenges such as data locality and availability.

Autoencoders have also been used to reduce dimensions; they learn an encoding of a dataset [46]. Their architecture is similar to a multi-layer perceptron (MLP): one input, one output, and one or more hidden layers. The difference from the MLP is that an autoencoder always has the same number of input and output nodes. Whereas the MLP learns the mapping between the input and target variables, an autoencoder learns to reconstruct its inputs. The hidden layers are responsible for encoding, they map the input feature space X to a lower-dimension space F, creating a compressed representation of X. The output layer on the other hand, serves as the decoder and reconstructs the input X from the compressed representation F.

Dimensionality reduction primarily addresses the curse of dimensionality and the processing performance challenges.

Instance selection refers to techniques for selecting a data subset that resembles and represents the whole dataset. Whereas dimensionality reduction deals with wide datasets, data reduction, more specifically instance selection, aims to reduce a dataset's height. The subset is consequently used to make inferences about the whole dataset. Instance selection approaches are diverse and include random selection, genetic algorithm-based selection, progressive sampling, using domain knowledge, and cluster sampling [110].

Although instance selection reduces dataset size thus improves processing performance and eases the curse of modularity, a number of questions arise:

- How big should the sample be? The sample size should balance accuracy and computing time.
- What sampling approach should be used? The choice of approach has a major impact on how well the subset represents the whole.
- How good will the model be? Instance selection introduces sampling error due to the differences between the sample and the whole dataset.

These issues, although well researched in learning with small datasets, are enlarged in the Big Data context because data size makes it more difficult to evaluate different properties or models.

Moreover, as already mentioned, in the Big Data context, challenges of class imbalance, noise, variance, and bias are more common and more difficult. In turn, this makes it more challenging to select a subset that will adequately represent the whole set. For example, with a large class imbalance, the selection approach must ensure that instances from all classes are selected. On the other hand, an appropriate instance selection can remedy class imbalance.

Data cleaning is another type of data manipulations; it refers to pre-processing such as noise and outlier removal. Thus, it tackles the challenges of dirty and noisy data. In this area, there is no significant development with respect to Big Data. Noise removal has been an especially active research topic in the audio, image, and video domains. Example techniques include smoothing filters and wavelet transforms [111]. However, such pre-processing is not practical for real-time processing or when the data distribution may change over time. Autoencoders, in addition to dimensionality reduction, can perform denoising when they recover a signal from partially corrupted input data. Therefore, the challenges of noisy and dirty data can also be addressed by this method.

3.3.1.2 Processing Manipulations

To improve machine learning performance with Big Data, processing manipulations focus on modifying how data are processed and stored. Here, the term *storage* refers not only to physical storage on a permanent medium, but also to how data are represented in memory. As illustrated in Figure 3.3, processing manipulations can happen during three phases of the data analytics pipeline: data transformation, data storage, and data analysis.

In these stages, independent of the category of manipulations, processes can be embedded to capture data provenance and therefore remedy provenance challenge; an example is the Reduce and Map Provenance (RAMP) developed for MapReduce as an extension for Hadoop [96]. However, such process carries a significant computational overhead.

The main stream of solutions from this category includes those based on parallelization. Processing techniques can take advantage of the inherent parallel nature of certain algorithms. Many learning algorithms, such as brute-force search and genetic algorithms, are trivially parallel, and therefore parallelization can provide massive performance improvements. Consequently, researchers have developed techniques and tools to parallelize machine learning. Two categories of parallel systems can be distinguished: vertical and horizontal scaling paradigms.

The ***vertical scaling*** (scaling up) paradigm includes multicore CPUs, supercomputers (blades), hardware acceleration including graphic processing units (GPUs) and field-programmable gate arrays (FPGAs). In the Big Data context, it is often discarded because it is limited to resources available on a single node; however, for machine learning, it is important to highlight the GPU approach. GPUs are specially designed for manipulating images for output displays. The large

number of cores (in thousands) compared to CPUs and the development of GPU interfaces such as Nvidia's CUDA have resulted in increased use of GPUs for general purpose processing. Because GPUs were originally designed for graphic display, image processing, matrix operations, and vector operations are especially suited for such systems. However, other ML algorithms can also be implemented on a GPU if they can be parallelized to a sufficiently high degree. Today, a large number of GPU accelerated ML algorithms are available [112]. Although it provides excellent performance through highly parallel processing, the size of data that GPU machine learning can process is limited by memory because typically processing happens on a single node.

FPGAs have been rarely mentioned in the context of Big Data. They are hardware components especially built for a specific purpose or application. Although this limits their applicability to Big Data, it is important to note the excellent FPGA performance achieved when scanning large amounts of network data [113].

The ***horizontal scaling*** (scaling out) paradigm refers to distributed systems where processing is dispersed over networked nodes. As with vertical scaling, processing is parallelized, but it also involves distributed nodes and hence network communication. Due to its capability to scale over large numbers of commodity nodes, distributed systems have been the focus of Big Data research.

This paradigm has been developed in two streams: batch- and stream-oriented systems.

Batch-oriented systems process a large amount of data at once, have access to most of the data, and typically are more concerned with throughput than with latency. MapReduce-based solutions such as Hadoop [114] and NIMBLE [115] belong to this category. Because MapReduce encounters difficulties when dealing with iterative algorithms, new solutions have been proposed: HaLoop [116] and Twister [117] extend Hadoop to provide better support for iterative processing. MapReduce-based solutions address the curse of modularity as they typically do not require the complete dataset to be held in memory. Moreover, data locality is also resolved as those solutions support work with data residing on different physical location. Such solutions facilitate work with high dimensional data, but they do not resolve the breakdown of the similarity-based reasoning, thus they provide partial resolution for the curse of dimensionality.

Similarly, to support this type of processing more effectively, to handle highly interconnected data, and accommodate graph algorithms, graph-based solutions have emerged. Pregel [118], the algorithm behind Google's PageRank, and Giraph[119], used by Facebook to analyze social connections, are examples from this category. They are based on the bulk synchronous parallel paradigm, and they retain states in memory, which facilitates iterative processing. Solutions from this category deal with the challenge of processing performance.

Stream-oriented systems operate on one data element or a small set of recent data in real-time or near real-time. Typically, computations performed in such a system are not as complex as those performed by batch systems. Examples from this category are Apache Storm [84], Yahoo's S4 [85], and Spark Streaming [64]. Although inspired by MapReduce, these platforms present a significant departure from Hadoop MapReduce; they have moved from in-memory data dependence to streams. Both Storm and S4 express computations using a graph topology, and their runtime engines handle parallelization, message passing, and fault tolerance. In contrast to Storm and S4, which perform one-by-one processing, Spark Streaming divides data into micro-batches and carries out computation on the micro-batches.

Stream-oriented systems enable mitigation of processing performance and real-time processing issues. They also mitigate the curse of modularity as they do not require the dataset to fit into memory and remedy the data availability challenge as they work with continuously arriving data. However, streaming systems are only suitable for very simple machine learning. Gorawski *et al.* [120] and Cugola and Margara [121] surveyed data stream processing tools and complex event processing. While they do not mention machine learning specifically, their discussion on the ability of each tool or approach to meet specific requirements provides insights about proper tool and approach selection.

Research in ML with Big Data has focussed mainly on the horizontal scaling paradigm: MapReduce-based solutions, graph-based solutions, and streaming. As discussed earlier, each category addresses specific problems and encounters difficulties with others. All solutions from the processing manipulation category primarily focus on improving performance (throughput or latency) and do not remedy a number of other challenges as illustrated in Table 3.1. The combination of processing manipulations with algorithms and new learning paradigms provides research opportunities to undertake the remaining Big Data challenges.

3.3.1.3 Algorithm Manipulations

Algorithm manipulations include approaches that modify existing algorithms, with or without applying new paradigms. Since the very beginning of machine learning, researchers have been trying to improve existing algorithms and to reduce their time and/or space complexity. With Big Data, these efforts have intensified because it has become more important to handle large datasets.

As illustrated in Figure 3.4, this study distinguishes two categories: *algorithm modifications* and *algorithm modifications with new paradigms* (approaches that involve modifying existing algorithms, applying process manipulations, and/or new paradigms).

Algorithm modifications have focussed on modifying algorithms to improve their performance. For example, the following approaches have been developed for specific machine

learning algorithms to address volume challenges:

- Pegasos [122] provides an optimized version of the support vector machine (SVM) algorithm for large-scale text processing. Its runtime does not depend directly on training set size, and hence Pegasos is especially suitable for large datasets.
- Regularization paths [123] for linear models supports linear regression, two-class logistic regression, and multinomial regression problems. This approach enables processing of large datasets and efficiently handles sparse features.

Solutions from this category deal with processing performance and real-time processing. As they are typically distributed computing solutions, they also address the data locality challenge. Moreover, the curses of dimensionality and modularity are partially remedied as those solutions support work with high dimensional data and may provide smaller memory footprint.

Algorithm modifications with new paradigms category involves modifying ML algorithms to work better with new process manipulations and/or new paradigms. An example from this category would be to modify an algorithm through parallelization and to adapt the algorithm for a new parallel processing paradigm such as MapReduce. Chu *el al.* [62] adapted several algorithms to multicore MapReduce, including naïve Bayes, Gaussian discriminative analysis, *k*-means, neural networks, support vector machines, and others. As Chu *et al.* [62] have shown, by using an approach such as MapReduce, some algorithms can be modified to improve performance. However, other algorithms, especially iterative ones, cannot be easily parallelized, and consequently, due to the curse of modularity, whole families of algorithms may not be usable for this paradigm [15].

ML platforms are another type of solutions from this category; they combine algorithm adaptation with new paradigms. Solutions from this category started with disk-based systems such as Apache Mahout [124] with underlying Hadoop. Because disk access is slow, ML platforms evolved to memory-based solutions: examples include Apache Spark [64] and 0xdata H2O [125]. Spark is a distributed computing framework based on distributed datasets and in-memory processing. In addition to the Spark Core, which provides a distributed computing foundation, Spark also includes libraries built on top of the core, including Spark SQL, Spark Streaming, MLlib (machine learning library), and GraphX. The MLlib library offers a large number of machine learning algorithms, but it is still limited compared to the R language or MATLAB.

0xdata H2O is another distributed machine learning platform. Like Spark, it is an in-memory distributed system and can therefore support massively scalable data analytics. Whereas Spark focuses on providing a platform for in-memory analytics, the emphasis of H2O is specifically on scalable machine learning. H2O can be installed on top of Spark (Sparkling Water)

to combine H2O's machine learning capabilities with the powerful Spark distributed platform. With the Mahout Samsara release, Mahout has also been transformed into a memory-based system.

Another example of a distributed machine learning platform is Petuum [126]. While Spark MLlib or H2O rely on general purpose distributed platforms (MLlib on Spark and H2O on Spark or Hadoop), Petuum is a complete platform developed specifically for machine learning. Vowpal Wabbit [127] is yet another interesting solution from this category; it uses an online learning approach. This means that data do not have to be pre-loaded into memory, and hence the memory footprint is not dependent on the number of samples.

Lastly Apache SAMOA [128] and Google's TensorFlow [129] both provide machine learning libraries for distributed processing environments. Unlike SAMOA which abstracts the distributed streaming nature of the processing from the user, TensorFlow makes use of data flow graphs as a flexible architecture to enable users to deploy computations across devices. Although these solutions are quite promising, they each are bound to a subset of challenges: SAMOA is bound to stream processing and TensorFlow is designed for Deep Learning.

This work classifies ML platforms as an *algorithm modifications* solution; however, they also can be considered *processing modifications*, as illustrated by the dashed line in Figure 3.4.

All algorithm modification solutions (with or without new paradigms) focus on providing the capability to process large datasets, improving performance, or providing real-time processing capabilities. They also remedy data locality as distributed computation can be performed with data residing on different physical locations. Algorithm modification with new paradigms, specifically ML platforms, mitigate the curse of modularity as they use memory of all nodes in the cluster and the curse of dimensionality as they facilitate work with high dimensional data. However, they do not specifically address a number of other challenges as illustrated in Table 3.1. Although unable to provide a complete solution to all the challenges discovered, algorithm manipulations offer a means for researchers to deploy, modify, and adapt existing algorithms for Big Data in order to address some of the unaddressed concerns.

3.3.2 Machine Learning Paradigms for Big Data

A variety of learning paradigms exists in the field of machine learning; however, not all types are relevant to all areas of research. For example, Deng and Li [130] presented a number of paradigms that were applicable to speech recognition. Congruently, the work presented here includes machine learning paradigms relevant in the Big Data context, along with how they address the identified challenges.

3.3.2.1 Deep Learning

Deep learning is an approach from the representation learning family of machine learning. Representation learning is also often referred to as feature learning [131]. This type of algorithm gets its name from the fact that it uses data representations rather than explicit data features to perform tasks. It transforms data into abstract representations that enable the features to be learnt. In a deep learning architecture, these representations are subsequently used to accomplish the machine learning tasks. Henceforth, because the features are learned directly from the data, there is no need for feature engineering. In the context of Big Data, the ability to avoid feature engineering is regarded as a great advantage due to the challenges associated with this process.

Deep learning uses a hierarchical learning process similar to that of neural networks to extract data representations from data. It makes use of several hidden layers, and as the data pass through each layer, non-linear transformations are applied. These representations constitute high level complex abstractions of the data [46]. Each layer attempts to separate out the factors of variation within the data. Because the output of the last layer is simply a transformation of the original input, it can be used as an input to other machine learning algorithms as well. Deep learning algorithms can capture various levels of abstractions, thus this type of learning is an ideal solution to the problem of image classification and recognition. Figure 3.5 provides an abstract view of the deep learning process [5]. Each layer learns a specific feature: edges, corners and contours, and object parts.

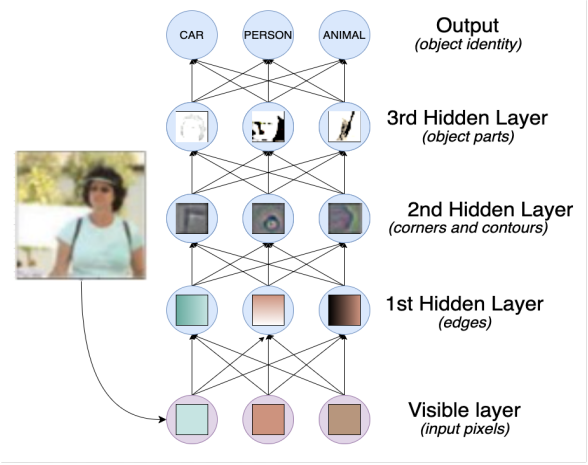


Figure 3.5: Representation of the Deep Learning process when performing image-based tasks [5]

The deep learning architecture is versatile and can be built using a multitude of components: autoencoders and restricted Boltzmann machine are typical building blocks [46]. Autoencoders

are unsupervised algorithms that can be used for many purposes such as anomaly detection, but in the context of Big Data, they typically serve as a precursor step with neural networks [132]. They work through backpropagation by attempting to set their target output as their input, thereby auto-encoding themselves. Boltzmann machines [133] are similar except that they use a stochastic rather than deterministic process. Deep belief networks [134] are another example of deep learning algorithms.

Furthermore, the reliance upon an abstract representation also makes these algorithms more flexible and adaptable to data variety. Because the data are abstracted, the diverse data types and sources do not have a strong influence on the algorithm results, making deep learning a great candidate for dealing with data heterogeneity.

Interestingly, deep learning can be used for both supervised and unsupervised learning [50]. This is possible due to the very nature of the technique; it excels at extracting global relationships and patterns from data because of its reliance upon creating high level abstractions. In the context of Big Data, this is a great advantage as it renders the algorithms less sensitive to veracity challenges such as dirty, noisy, and uncertain data [50]. Moreover, its multiple layers of non-linear transformations addresses the challenge associated with data non-linearity. Deep compression has been proposed as a way of speeding up processing without the loss of accuracy [135].

According to the described characteristics, deep learning seems to be well suited to address many of the previously identified challenges such as feature engineering, data heterogeneity, non-linearity, noisy and dirty data, and data uncertainty. However, those algorithms are not fundamentally built to learn incrementally [136] and are therefore susceptible to the data velocity issue. Although they are especially well adapted to handle large datasets with complex problems, they do not do so in a computationally efficient way. For high dimensional data [46] or large numbers of samples such algorithms may even become infeasible, making deep learning susceptible to the curse of dimensionality.

3.3.2.2 Online Learning

Because it responds well to large-scale processing by nature, online learning is another machine learning paradigm that has been explored to bridge efficiency gaps created by Big Data. Online learning can be seen as an alternative to batch learning, the paradigm typically used in conventional machine learning. As its name implies, batch learning processes data in batches and requires the entire dataset to be available when the model is created [74]. Furthermore, once generated, the model can no longer be modified. This makes it difficult to deal with the dimensions of Big Data for the following reasons:

- Volume: having to process a very large amount of data at one time is not computationally efficient or always feasible.
- Variety: the need to have the entire dataset available at the beginning of the processing limits the use of data from various sources.
- Velocity: the requirement to have access to the entire dataset at the time of processing does not enable real-time analysis or use of data from various sources.
- Veracity: because the model cannot be altered, it is highly susceptible to performance impediments caused by poor data veracity.

Conversely, online learning uses data streams for training, and models can learn one instance at a time [50]. This "learn-as-you-go" paradigm alleviates the computational load and processing performance because the data do not have to be entirely held in memory. This enables processing of very large volumes of data, remedies the curse of modularity, facilitates real-time processing, and provides the ability to learn from non-i.i.d. data [50]. Moreover, as it does not require all data to be present at once or located at the same place, this paradigm remedies data availability and locality.

Furthermore, the descriptor "online" also reflects the fact that this paradigm continuously maintains its model; the model can be modified whenever the algorithm sees fit. Its adaptive nature makes it possible to handle a certain amount of dirty and noisy data, class imbalances, and concept drift. Indeed, Mirza *et al*. [137] proposed an ensemble of subset online sequential extreme learning machines to achieve a solution for concept drift detection and class imbalance, while Kanoun and van der Shaar [138] presented an online learning solution for remedying the challenge of concept drift.

It is apparent that the online learning architecture responds well to the challenges associated with Big Data velocity; its incremental learning nature alleviates challenges of data availability, real-time processing, i.i.d, and concept drift. For example, this paradigm could be used to handle stock data prediction due to the ever-changing and rapidly evolving nature of the stock market. However, the issues associated with dimensionality, feature engineering, and variety remain unresolved. Moreover, not all machine learning algorithms can be easily adapted to the online learning.

3.3.2.3 Local Learning

First proposed by Bottou and Vapnik in 1992 [139], local learning is a strategy that offers an alternative to typical global learning. Conventionally, ML algorithms make use of global

learning through strategies such as generative learning [140]. This approach assumes that based upon the data's underlying distribution, a model can be used to re-generate the input data. It basically attempts to summarize the entire dataset, whereas local learning is concerned only with subsets of interest. Therefore, local learning can be viewed as a semi-parametric approximation of a global model. The stronger but less restrictive assumptions of this hybrid parametric model yield low variance and bias [8].

Figure 3.6 provides an abstract view of the local learning process. The idea behind it is to separate the input space into clusters and then build a separate model for each cluster. This reduces overall cost and complexity. Indeed, it is much more efficient to find a solution for k problems of size m/k than for a single problem of size m. Consequently, such approach could enable processing of datasets that were considered too large for global paradigms. Another way of implementing local learning is to modify the learning algorithms so that only neighbouring samples influence the output variable.

A typical example where local learning would be beneficial is, for instance, predicting the energy consumption of several customers. Building a model for similar customers could be favourable to building one unique model for all customers or to building one model per customer.

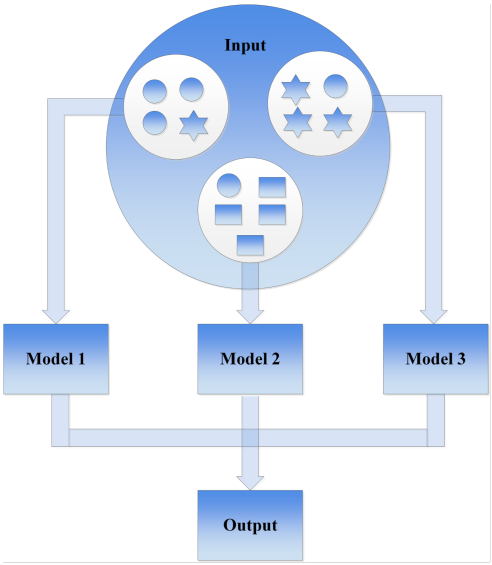


Figure 3.6: Representation of the Local Learning paradigm

Recently, Do and Poulet [141] developed a parallel ensemble learning algorithm of random local support vector machines that was able to perform much better than the typical SVM algorithm in addressing volume related issues, thereby demonstrating how local learning can help alleviate some of the issues associated with Big Data. Moreover, local learning has out-

performed global learning in terms of accuracy and computation time in several forecasting studies [142], [143].

Dividing the problem into manageable data chunks reduces the size of data that need to be handled and potentially loaded into memory at once; therefore, this paradigm alleviates the curse of modularity. In addition, because of the locality of each cluster, models are not significantly affected by the challenges associated with class imbalances and data locality. Recent work has shown that local learning often yields better results than global learning when dealing with imbalanced datasets [8].

Therefore, the challenge of the curse of modularity, class imbalance, variance and bias, and data locality can be alleviated by a local approach. However, matters of dimensionality and velocity, such as concept drift among others, have yet to be addressed. Overall, in the Big Data context, the local approach remains largely unexplored; studying how this paradigm could better handle velocity and veracity challenges appears to be particularly open.

3.3.2.4 Transfer Learning

Transfer learning is an approach for improving learning in a particular domain, referred to as the *target domain,* by training the model with other datasets from multiple domains, denoted as *source domains*, with similar attributes or features, such as the problem and constraints. This type of learning is used when the data size within the target domain is insufficient or the learning task is different [144]. Figure 3.7 shows an abstract view of transfer learning.

The distinguishing characteristic of transfer learning from other traditional ML approaches is fact that the training set does not necessarily come from the same domain as the testing set. Moreover, it can train on data from several domains individually or combined together using regular or adapted machine learning algorithms; domains do not have to have the same data distribution or the same feature space [145]. Different elements can be transferred from the source domains into the target domain: instances, feature representations, model parameters, and relational knowledge [146]. An example use case is energy consumption prediction for a new building (the target domain) using datasets collected from other similar buildings (the source domains) that probably have similar consumption patterns, but are different in size and efficiency.

Consequently, transfer learning is a possible candidate for resolving some of the challenges related to the volume, variety, and veracity dimensions of Big Data environments. From the volume category, it can remedy class imbalance as the instances can be transferred from other diverse domains to better balance classes in the target domain. Because of the ability to learn from different domains and transfer knowledge between different datasets, transfer learning is a promising solution for the data heterogeneity challenge (the variety category). Moreover,

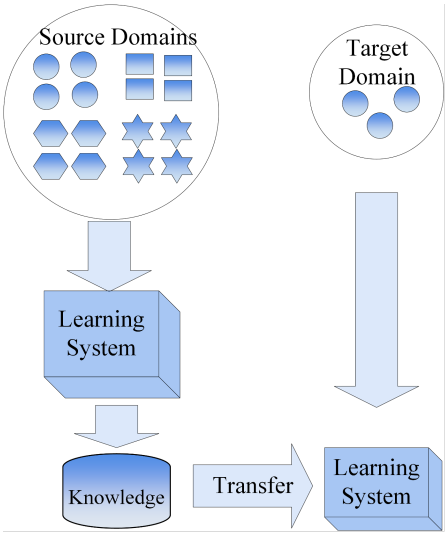


Figure 3.7: Representation of the Transfer Learning paradigm

instance transfer from different domains can contribute to reducing challenges of dirty and noisy data as well as data uncertainty (the variety and veracity categories).

A few studies have discussed transfer learning for Big Data. Yang *et al.* [145] introduced an automatic transfer learning algorithm for Big Data. They improved a supervised learning approach, the Laplacian eigenmaps algorithm, by enabling automatic knowledge transfer from the source domains to the target domain. The system was designed for analyzing short text data using knowledge from the long text obtained from the Web. Zhang [80] described several examples of transfer learning with Big Data based on the concept of data fusion among homogenous and heterogeneous datasets; data fusion refers to combining data from several sources. For these reasons, transfer learning is one of the paradigms that address data size and heterogeneity.

3.3.2.5 Lifelong Learning

Lifelong learning mimics human learning; learning is continuous; knowledge is retained and used to solve different problems. It is directed to maximize overall learning, to be able to solve a new task by training either on one single domain or on heterogeneous domains collectively [147]. The learning outcomes from the training process are collected and combined together in a space called the *topic model* or *knowledge model*. In the case of training on heterogeneous domains, transfer learning might be used in the combining step to create such a topic model. The existing knowledge in this topic model is used to perform a new task regardless of where the knowledge comes from.

Lifelong learning is related to online learning and transfer learning. Like online learning, lifelong learning is a continuous process; however, whereas online learning considers only a single domain, lifelong learning includes a multitude of domains. Like transfer learning, lifelong learning is capable of transferring knowledge among domains. But, unlike transfer learning, lifelong learning is a continuous process over time because the topic space is refined each time a new learning outcome arrives. For example, consider the process of inferring individuals' sport interests when various data are available such as their physical location, social media interactions, weather data, and Web browsing history. The prediction of individual interests needs to change periodically over time when new data become available because the area of interest may shift over time. Figure 3.8 depicts a high level view of lifelong learning.

Traditional machine learning algorithms have a single target domain and cannot transfer learning from one task to another (with the exception of transfer learning). Consequently, Khan *et al.* [148] consider such approaches unsuitable for Big Data because of the many domains covered by such data and the constant appearance of new ones. They consider lifelong learning to be a good candidate solution for Big Data because the model is continuously refined and the learned task is applicable to different domains.

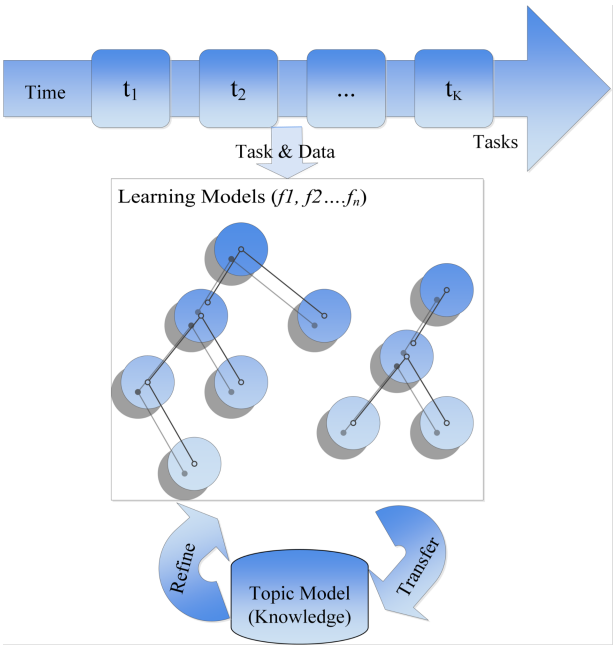


Figure 3.8: Representation of the Lifelong Learning paradigm

Similar to online learning, lifelong learning is a promising solution for processing performance, real-time processing, data availability, and concept drift. This is because it does not rebuild the model every time a new piece of data arrives, but only updates the existing knowl-

edge based on the new incoming data. New knowledge is of course added to the existing knowledge base for all future tasks. Since lifelong learning incorporates transfer learning as its integral part, it addresses the same issues as transfer learning: the data heterogeneity, class imbalance, dirty and nosy data, and data uncertainty. However, presently there has been little development in this area because achieving this vision is very challenging.

Lifelong learning has been applied in various domains. Chen and Liu [149] suggested an unsupervised learning algorithm, the Gibbs Sampler, for mining the topic model and for generating a better knowledge space in the context of e-commerce reviews. In their work, the Big Data are divided into small sets, each set is trained individually, and the learning results are compared to yield a better topic model. Suthaharan [150] discussed issues related to combining machine learning, including lifelong learning, with Big Data technologies such as Hadoop and Hive. They focussed on network intrusion detection where data are growing quickly and arriving fast from different sources.

Khan *et al.* [148] surveyed lifelong learning models, including probabilistic topic models and knowledge-based topic models, for natural language processing with large data volumes. Their work discussed how these models can be used with Big Data to improve learning performance and accuracy.

3.3.2.6 Ensemble Learning

Ensemble learning combines multiple learners to obtain better learning outcomes (e.g., prediction, classification) than those obtained from any constituent learner [151]. Figure 3.9 presents an abstract view of the ensemble learning process. Typically, the overall outcome is determined by a voting process among the weighted outcomes of individual learners [151]. These individual learners can be similar or from completely different categories, including those belonging to supervised and unsupervised ML. The weighting mechanism assigns a value to each learning output point and combines them. The voting process could be implemented in a straightforward way by directly aggregating the values of the learning points or through the use of the statistical techniques to obtain a combined value of the learning outputs that may lead to better learning performance [152]. Waske and Benediktsson [153] have applied SVM for individual learners and also used an SVM in the voting process.

There are two main ways to apply ensemble learning: the first one trains different learners, each one on the complete dataset, whereas the second one splits the dataset and trains each learner (same or different) only on a subset. The second approach has potential in the Big Data context because it can speed up and improve the learning process. Basically, improvement is achieved by splitting up large volumes of data into small disjoint datasets. One or more machine learning algorithm(s) are then trained on a different disjoint dataset. The outcomes

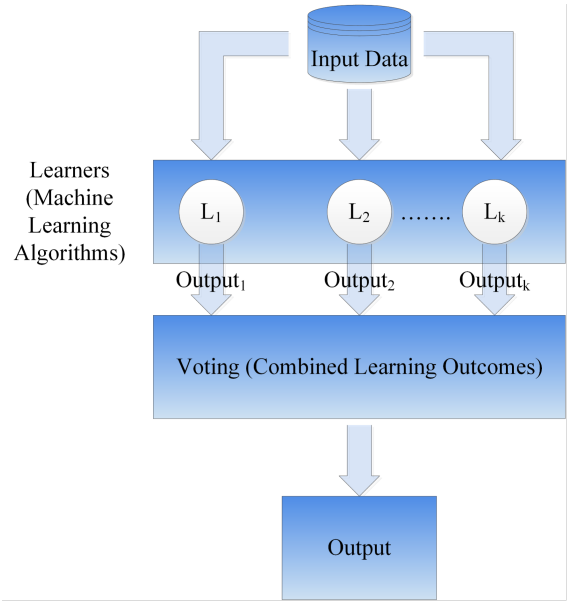


Figure 3.9: Representation of the Ensemble Learning paradigm

from all learners are combined using some kind of voting scheme to improve the accuracy of the overall solution. For example, to detect anomalous behavior, ensemble learning could be applied by breaking a large dataset into small ones, training learners on small sets, and combining results to identify anomalies with high accuracy, but with few false alarms.

Several studies have discussed applying ensemble learning to Big Data. For example, Tang *et al.* [154] used it to decrease computation time and simultaneously improve learning accuracy. They split the large dataset into smaller datasets using a probabilistic approximation technique called the Reservoir sampling method.

Research studies have shown that ensemble learning performs well with different datasets [155], [156]. Zang *et al.* [155] presented a comparative study of ensemble and incremental learning. They observed that incremental algorithms were faster, but ensemble models performed better in the presence of concept drift. Alabdulrahman [156] experimented with four different datasets; he investigated correlations among ensemble sizes, base classifiers, and voting methods.

Works of Tang *et al.* [154] and Gruber *et al.* [157] used ensemble learning mainly to minimize computing time by dividing a big dataset into small training sets. Unlike the work of Tang *et al.* [154], Gruber *at al.* [157] suggested invoking several different machine learning algorithms on the small sets. Then the output values of all algorithms were weighted, combined, and evaluated using inverse probability weights (i.e., the voting phase). Finally, the authors showed that ensemble learning could also be used to choose learning algorithms by removing

those that did not have an impact on the final outcome. This approach consequently decreased computing time.

Ensemble learning plays a key role in emphasizing correctness of learning outcomes as well as speeding up the learning process. With respect to correctness, using several different learners or training with different subsets has the potential to minimize the error rates. On the other hand, similar to local learning, splitting the dataset and training on subsets improves processing performance as it is more efficient to find a solution for k problems of size m/k than for a single problem of size m. This data splitting also reduces data chunks that need to be loaded into memory at once making ensemble learning capable of addressing the curse of modularity. Moreover, ensemble learning is capable of addressing concept drift [155]. How ensemble learning could be modified to best handle the issues related to variety and velocity remains to be explored.

3.4 Discussion

Ratner [72] describes machine learning as a field that makes use of algorithms to solve problems with an underlying statistical component, such as regression, classification, and clustering, by means of an assumption-free non-parametric approach. For years, researchers have used machine learning to solve such problems without worrying about whether the situations they were facing met the requirements and the classical statistical assumptions upon which certain methodologies rely [72]. Arguably, the lack of concern with regard to these assumptions has enabled scientists to advance more quickly in the field of machine learning than in the field of statistics [72]. However, with the advent of Big Data, many of the assumptions upon which the algorithms rely have now been broken, thereby impeding the performance of analytical tasks. In response to those pitfalls, together with the need to process large datasets fast, a number of new machine learning approaches and paradigms have been developed. However, it remains consistently difficult to find the best tools and techniques to tackle specific challenges.

This chapter has identified and presented challenges in machine learning with Big Data, reviewed emerging machine learning approaches, and discussed how each approach is capable of addressing the identified challenges. The overview of the relation between challenges and approaches has been presented in Table 3.1. A single application or a use case may encounter several challenges and may need to combine several approaches to handle those challenges. The following use cases demonstrate the use of the presented work:

Case study 1: Energy Prediction. Sensor-based forecasting using historical sensor readings to infer future energy consumption has gained popularity due to proliferation of sensors and smart meters. Grolinger *et al.* [133] considered energy forecasting with large sensor data.

They encountered two main challenges from the volume category: processing performance (long time to build the model) and curse of modularity (complete data set fitting into memory). To resolve those challenges, they applied local learning, thus complying with Table 3.1 which indicates that local learning addresses challenges of processing performance and curse of modularity. As indicated in Table 3.1, those challenges could potentially be solved with other approaches such as horizontal scaling. In their study, local learning not only significantly improved the performance, but also increased prediction accuracy.

Case study 2: Recommender system. Collaborative filtering mechanisms are capable of suggesting relevant online content to users based on their browsing history. Millions of digital transactions are collected daily and they need to be continuously analyzed to improve recommendations and increase user engagement. Like case study 1, to address processing performance and curse of modularity challenges, Bachmann [158] applied local learning. Additionally, local learning enabled embedding contextual awareness into collaborative filtering.

Case study 3: Machine translation. Machine translation systems are computationally expensive and, in the case of large data sets, may even be prohibitive [159]. To handle this processing performance challenge, Google uses deep learning, specifically TensorFlow which is one of the ML platforms as illustrated in Figure 4. TensorFlow can be deployed on one or more CPUs or GPUs making this a horizontally and vertically scalable approach. Thus, Google machine translation uses a combination of several approaches (deep learning, horizontal and vertical scaling) to address the performance challenge.

Case study 4: Activity recognition. The constant advancements of the IoT has led to the development of numerous sensor equipped wearable devices. Saeedi *et al.* [160] proposed autonomous reconfiguration of wearable systems in order to handle impact of configuration changes on activity recognition. The main challenge they faced lied in the heterogeneity of the data; wearable sensor data contains a variety of signal heterogeneity such as subject, device and sampling frequency related heterogeneity. In accordance with Table 3.1, transfer learning was used to tackle this Big Data challenge. Deep learning and lifelong learning could have also been considered. Nevertheless, the authors reported a performance increase between 3-13% due to their solution.

The correlation between approaches and Big Data challenges presented in Table 3.1. makes it possible to identify the main opportunities and directions for future research in machine learning with Big Data. Because a single approach may address more than one challenge and several challenges may be addressed with a single approach, the future directions are not categorized according to V dimensions, but they represent broad research opportunities:

- Data fusion will become even more important as researchers and industry try to combine data from a number of different sources with different formats and semantics to provide

new insights.

- Process and algorithm manipulations are expected to continue to attract significant research interest because they make it possible to use traditional algorithms adapted to work with Big Data.
- Paradigms that enable updating of existing models upon arrival of new data without the need to retrain the complete model are very promising because they can accommodate bigger datasets than batch learning. Paradigms from this category are online learning and lifelong learning.
- Stream processing is presently limited to relatively simple problems. However, with new developments, it is anticipated that it will be able to handle more complex computations.
- Online learning may possibly merge with stream processing. If online learning can update its model in real time or near real time, it can be integrated into stream processing.
- Integration of various approaches such as deep learning and online learning presents itself as an interesting and promising research area worthy of further consideration. A combination of various approaches would ensure better coverage of the issues related to machine learning with Big Data.

From Table 3.1 it can be noted that there is no correlation between Bonferonni's principle and any of the reviewed approaches. Although its impact in the Big Data context is large [76], we are not aware of a specific Big Data solution addressing this problem. Preventing those false positives is important, but it appears it attracted very limited attention from the Big Data community.

3.5 Conclusions

This chapter has provided a systematic review of the challenges associated with machine learning in the context of Big Data and categorized them according to the V dimensions of Big Data. Moreover, it has presented an overview of ML approaches and discussed how these techniques overcome the various challenges identified.

The use of the Big Data definition to categorize the challenges of machine learning enables the creation of cause-effect connections for each of the issues. Furthermore, the creation of explicit relations between approaches and challenges enables a more thorough understanding of ML with Big Data. This fulfills the first objective of this work; to create a foundation for a deeper understanding of machine learning with Big Data.

Another objective of this study was to provide researchers with a strong foundation for making easier and better-informed choices with regard to machine learning with Big Data. This objective was achieved by developing a comprehensive matrix that lays out the relationships between the various challenges and machine learning approaches, thereby highlighting the best choices given a set of conditions. This research enables the creation of connections among the various issues and solutions in this field of study, which was not easily possible on the basis of the existing literature.

From the development or adaptation of new machine learning paradigms to tackle unresolved challenges, to the combination of existing solutions to achieve further performance improvements, this work has identified research opportunities. This work has therefore accomplished its last objective by providing the academic community with potential directions for future work and will hopefully serve as groundwork for great improvements in the field of machine learning with Big Data.

Chapter 4

Transformer Adaptation for Load Forecasting with Big Data

4.1 Introduction

Smart meter technologies and the rise of the IoT have deeply affected the challenge of electrical load forecasting by taking it to a much larger scale [161]. Electrical load forecasting is not a recent challenge. The literature shows method and application research from as early as the 1970s [162]. However, the methods previously used to perform this task are affected by a number of new factors: the data are now collected at a much faster interval and is now accessible in real-time, making it Big Data, both in terms of volume and velocity. Utility companies are looking for means to deploy models quickly for large amounts of meters without jeopardizing accuracy. This has led to a need for more efficient forecasting, more specifically in terms of reducing model training and processing time, therefore highlighting the need for solutions such as model parallelization. As a response, this research will present a modification of the transformer architecture. The following section will serve as an introduction to this chapter.

4.2 Load Forecasting for Diverse Smart Meters

In 2016, three-quarters of the global emissions of carbon dioxide (CO2) were directly related to energy consumption [163]. Amongst the energy-related emissions, the international energy agency reported that electricity was the largest single contributor with 36% of the responsibility [164]. In the United States alone, in 2019, 1.72 billion metric tons of CO2 [165] were generated by the electric power industry, 25% more than those related to the gasoline and diesel

fuel consumption combined. The considerable impact of electricity consumption on the environment has led to many measures being implemented by various countries to reduce their carbon footprint. Canada, for example, as part of its commitment to reduce its greenhouse gas emission by 30% below 2005 levels before 2030, developed the Pan-Canadian Framework on Clean Growth and Climate Change (PCF). One of the key parts of the plan involves the modernization of the power system through the use of smart grid technologies [166].

The Electric Power Research Institute (EPRI) defines the smart grid as one that integrates various forms of technologies within each aspect of the electrical pipeline, from generation to consumption. The purpose of technological integration is to minimize the environmental burden, improve markets, and strengthen reliability and services while minimizing costs and enhancing efficiency [167]. Electric load forecasting plays a key role [168] to fulfill those goals by providing intelligence and insights to the grid [169].

There exist various techniques used to perform load forecasting. However, Recurrent Neural Network (RNN) based approaches have been outperforming other methods [170, 168, 171]. Although these techniques have been successful in terms of accuracy, a number of downsides have been identified. Firstly, studies typically evaluate models on one, or a very few, buildings or households [172, 173, 88]. We suggest that using a single data stream is not sufficient to bring conclusions about the applicability of results over a large set of diverse electricity consumers. Accuracy for different consumers will vary greatly as demonstrated by Fekri et al. [174]. Therefore, this research uses various data streams, exhibiting different statistical properties, in order to ensure the portability and reproducibility of the results. Secondly, although RNN-based methods are achieving high accuracy, their high computational cost further emphasizes the processing performance challenges associated with deep learning when dealing with Big Data.

In response to the computational cost, the transformer architecture has been developed in the field of Natural Language Processing (NLP) [37]. It has allowed great strides in terms of accuracy and, through its ability to be parallelized, has led to performance increase for NLP tasks. However, the architecture is highly tailored to linguistic data and cannot directly be applied to time series. Nevertheless, linguistic data and time series both have an inherently similar characteristic, their sequentiality. This similarity along with the high level of success of transformers in NLP motivated this work.

Consequently, this chapter proposes an adaptation of the complete transformer architecture for load forecasting, including both the encoder and the decoder components. A contextual module, an N-Space transformation module, a modified workflow, and a dimensionality reduction module were developed to account for the differences between language and electrical data. More specifically, they are used to address the numerical nature of the data and the con-

textual features associated with load forecasting tasks and that have no direct counterparts in NLP. The proposed approach is evaluated on 20 data streams and the results show that the adapted transformer is capable of outperforming cutting-edge Sequence-to-Sequence RNNs [175] while having the ability to be parallelized, therefore addressing the performance issues of the deep learning architecture, as identified in Chapter 3, in the context of load forecasting.

The remainder of this chapter is organized as follows: Section 4.3 presents related work; the proposed architecture and methodology are shown in Section 4.4; Section 4.5 introduces and discusses our results and finally, Section 4.6 presents the conclusion.

4.3 Related Work

Due to technological changes, the challenge of load forecasting has grown significantly in recent years [161]. Load forecasting techniques can be categorized under two large umbrellas:

- Statistical Models
- Modern Models based on machine learning (ML) and artificial intelligence (AI)[176].

Statistical methods used in load forecasting include Box-Jenkins model-based techniques such as ARMA [177, 178, 179] and ARIMA [180, 181], which rely upon principles including autoregression, differencing and moving average [182]. Other statistical approaches include Kalman Filtering algorithms [183], grey models [184] and exponential smoothing [185]. However, traditional statistical models face a number of barriers when dealing with Big Data [186] which can lead to poor forecasting performance [176]. In recent years ML/AI have seen better successes and appear to dominate the field.

Modern techniques based on machine learning and artificial intelligence provide an interesting alternative to statistical approaches because they are designed to autonomously extract patterns and trends from data without human interventions. One of the most challenging problems of load forecasting is the intricate and non-linear relationships within the features of the data [187]. Deep learning, a machine learning paradigm, is particularly well suited to handle those complex relationships [188]. Recently, deep learning algorithms have been used heavily in load forecasting. Singh et al. [189] identified three main factors that influenced the rise of the use of deep learning algorithms for short-term load forecasting tasks: its suitability to be scaled on big data, its ability to perform unsupervised feature learning, and its propension for generalization.

Convolutional Neural Networks (CNN) have been used to address short-term load forecasting. Dong et al. [190] proposed combining CNN with k-means clustering in order to create

subsets of their data and apply convolutions on smaller sub-samples. The proposed method performed better than Feed-Forward Neural Network (FFNN), Support Vector Regression (SVR) and CNN without k-means. SVR in combination with local learning had been previously successfully explored by Grolinger et al. [88] but was subsequently outperformed by deep learning algorithms. Recurrent Neural Networks (RNN) were also used to perform load forecasting due to their ability to retain information and recognize temporal dependencies. Zheng et al. [168] compared their results using RNN favourably to FFNN and SVR. Improving on these ideas, Rafi et al. [191] proposed a combination of convolutional neural network (CNN) and long short-term memory (LSTM) networks for short-term load forecasting which performed well but was not well suited to handle input and outputs of different lengths. This caveat is important due to the nature of load forecasting. In order to address this shortcoming, Sequence-to-Sequence models were explored. Marino et al. [192] and Sehovac et al. [193] both achieved good accuracy using this architecture. These works positioned the S2S RNN algorithm as the state-of-the-art approach for electrical load forecasting tasks by successfully comparing S2S to DNN [193], RNN [193, **?**], LSTM [193, **?**, **?**], and CNN [**?**]. Subsequently, Sehovac et al. [194] further improved their model's accuracy by including attention to their architecture. Attention is a mechanism that enables models to place or remove focus, as needed, on different parts of an input based on their relevance.

However, accuracy is not the only metric or challenge of interest with load forecasting with Big Data. Online learning approaches, such as the work of Fekri et al. [174] have been used to address velocity-related challenges and enable models to learn on the go. However, their proposed solution does not address the issue of scaling and cannot be parallelized to improve performance as it relies upon RNNs. In order to address the need for more computationally efficient ways to create multiple models, a distributed ML load forecasting solution in the form of federated learning has been recently proposed [34]. However, the evaluation took place on a small dataset and the chosen architecture still prohibited parallelization. Similarly, Tian et al. proposed a transfer learning adaptation of the S2S architecture in order to partly address the challenge of processing performance associated with deep learning [175]. The processing performance was improved by reducing the training time of models by using transfer learning.

The reviewed RNN approaches [174, 194, 175], achieved better accuracy than other deep algorithms. However, RNNs are difficult to parallelize because of their sequential nature. This can lead to time-consuming training which prevents scaling to a large number of energy consumers. In contrast, our work uses transformers, which enable parallelization and thus reduce computation time. Moreover, our transformer-based approach outperforms state-of-the-art S2S RNN.

4.4 Load Forecasting with Transformers

The traditional transformer architecture as proposed by Vaswani et al. [37] was originally developed for natural language processing (NLP) tasks. In NLP, the input to the transformer are sentences or phrases which are first converted to numbers by an embedding layer, and then are passed to the encoder portion of the transformer. The complete sentence is processed at once, therefore the transformer needs another way to capture sequence dependency among words: this is done through the positional encoding.

In contrast, load forecasting deals with very different data. The two main components of load forecasting data are energy consumption readings and contextual elements including information such as the day of the week, the hour of the day, and holidays. The similarity between load forecasting and NLP comes from the sequentiality present in data and dependencies between words/readings: this motivated the idea of adapting transformers for load forecasting. However, the difference in the nature of the data and its structure impacts the ability to directly utilize the transformer architecture and requires adaptations.

Therefore, to perform load forecasting with transformers, this research introduces a contextual and an N-Space transformation modules along with modifications to the training and inference workflows. The following section will present how the transformer architecture was adapted for load forecasting by introducing the two workflows. For each of these, we will describe how each component of the original architecture was modified and which components were added in order to successfully design the adapted model.

4.4.1 Model Training

The transformer training is carried out through the contextual model, N-Space transformation module, and lastly the actual transformer. Figure 4.1 depicts the modified architecture.

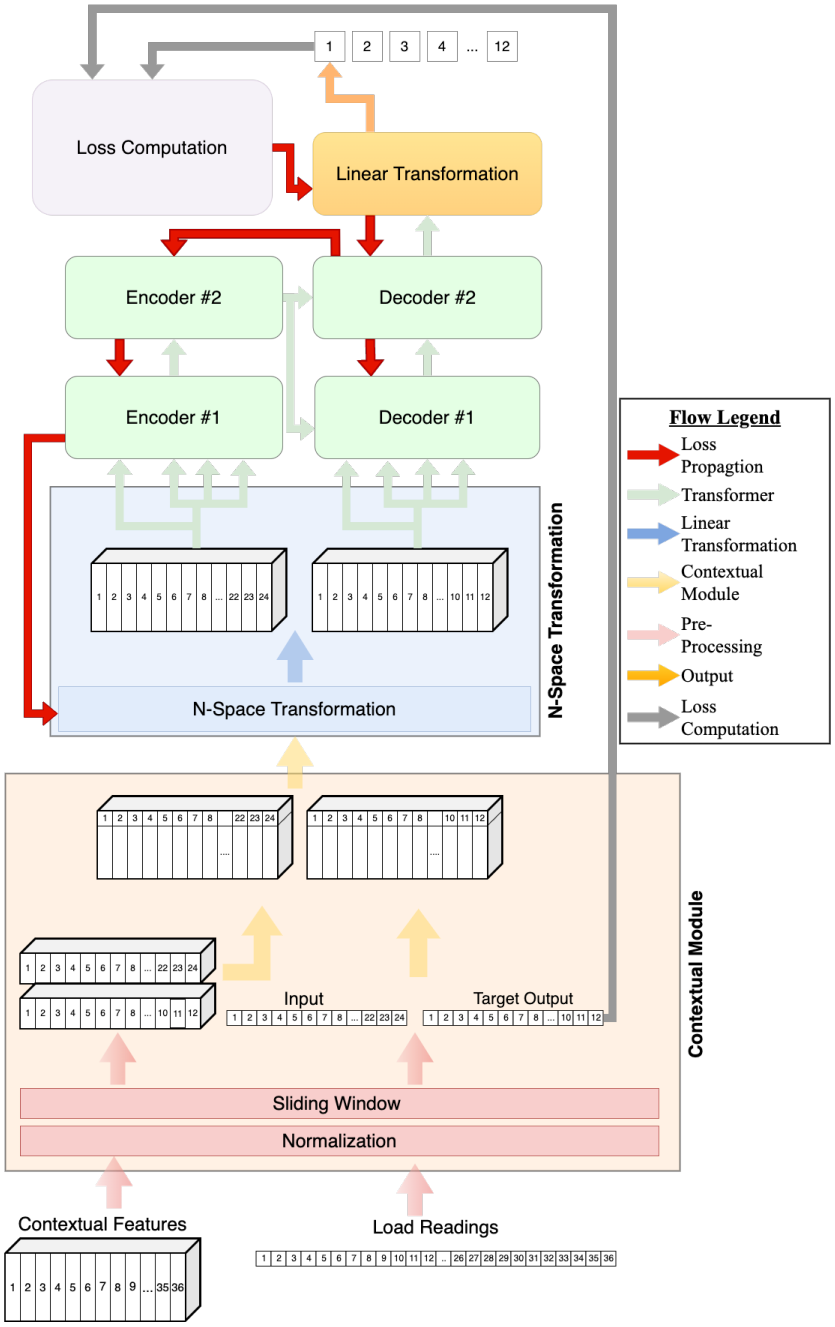


Figure 4.1: Training Workflow

4.4.1.1 Contextual Module

The objective of the contextual model is to enable different processing pathways for the energy readings and the contextual features. Energy readings are both the input to the system and the required output because past readings are used to predict future consumption. In contrast, contextual features are known values and do not need to be predicted by the transformer. For

example, if trying to predict a load 24 steps ahead, the day of the week and the hour of the day for which we are forecasting are known. Therefore, these two components of the data can be processed differently.

The input of the system consists of energy readings combined with the contextual features while the expected output is comprised only of load values. Contextual features commonly used with load forecasting [175] include those extracted directly from the reading such as the time of the day and the day of the week. After extraction, these features are represented numerically and combined to form contextual vectors.

These numerical vectors are merged with the load readings resulting in n features for each time step where n is the number of contextual features plus one to account for the load reading. At this point, the data are a stream of readings while the transformer in NLP expects sentences. To convert those streams into a format suitable for transformers, the sliding window technique is used. The process is depicted in details in Figure 4.2.

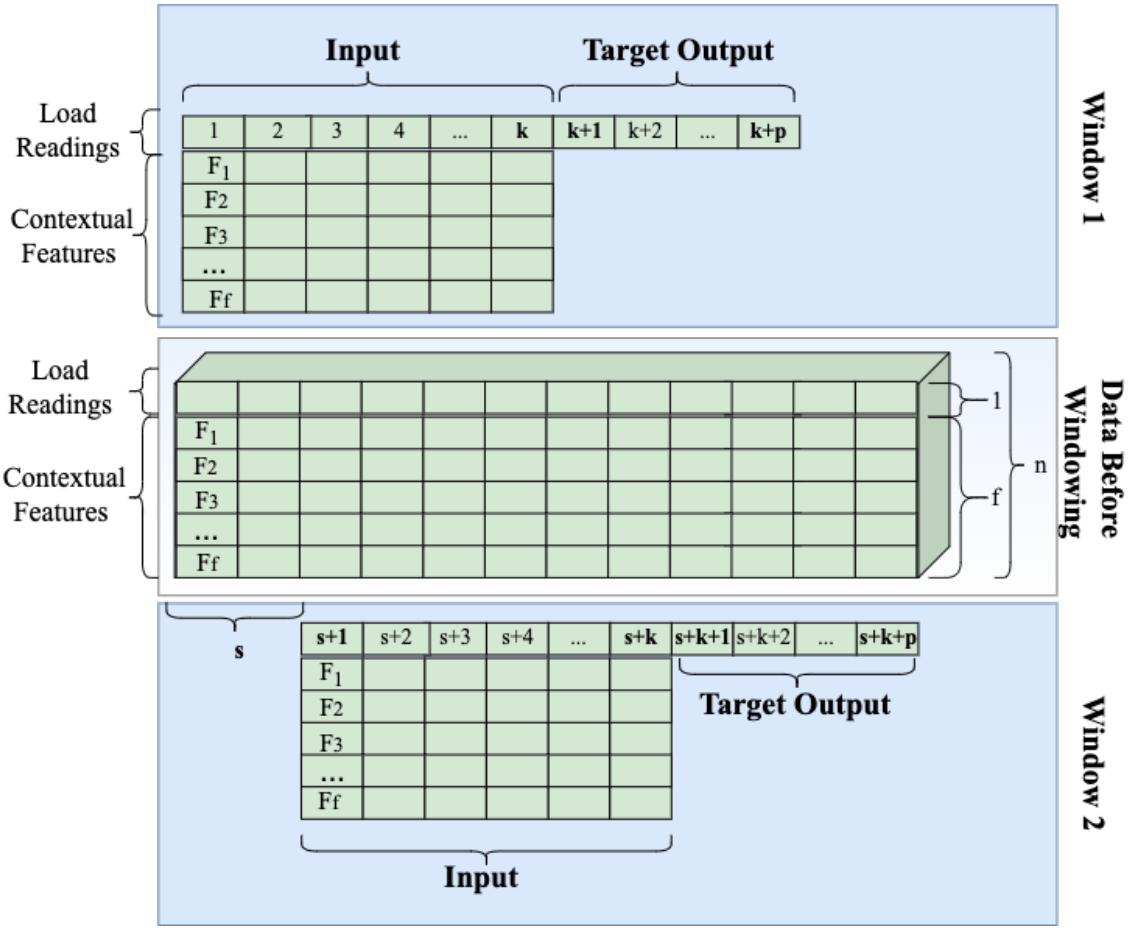


Figure 4.2: Overlapping sliding window with contextual features

In the sliding window approach, as shown in Figure 4.2, the first input sample consists of the first k readings, thus one sample has a dimension of $k \times n$ where n is the number of features. If predicting p steps ahead, the expected output contains load readings, without contextual features, at time step $k + 1$ to $k + p$. Then, the window slides for s steps and the next input sample consists of readings from $s + 1$ to $s + k$, and the expected output contains load readings for time steps $s + k + 1$ to $s + k + p$. In this study, overlapping sliding widows are used, thus $s < k$.

In the training, contextual features with the load values are the inputs to the encoder while the inputs to the decoder are contextual features with the expected (target) load values. The output of the decoder consists of the predicted load values which are used to compute the loss function and, consequently, to calculate gradients and update the weights.

4.4.1.2 N-Space Transformation Module

The purpose of the N-Space transformation module is to improve the performance of the multi-headed attention (MHA) component of the transformer by transforming the input data to better capture relationships between energy readings and contextual features. Once the data passes through the contextual module, the energy reading and its contextual features form a vector. That vector n_t at time t is a concatenation of a contextual vector c_t and an energy reading er_t such that n_t=($er_t, c_{1t}, c_{2t}, c_{3t}, ..., c_{ft}$) where f is the number of contextual features. Therefore, the length of any vector n is equal to the length of $f + 1$. In order to enter the encoder, each of the k vectors corresponding to k time steps of the window are stacked together to form an augmented input matrix of dimensions $n \times k$, effectively rotating the matrix.

Since the contextual features are concatenated to the energy reading, only one column of the input matrix contains the energy reading. As the multi-head attention (MHA) deals with subsets of features, several heads will only be handling contextual features; however, in energy forecasting, we are interested in the relationship of contextual features with the target variable, energy consumption.

The MHA performs attention calculations on different projections of the data, those calculations are as follows:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{4.1}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{4.2}$$

where Q, K, and V are query, keys, and values vectors used to calculate the attention and taken from the input matrix, W^o are parameters among heads while W_i^Q, W_i^K, and W_i^V are projection parameters for Q, K, and V [37]. Given that h is the number of heads and d_{model} represents the

number of input features, the dimensions of the projection parameters are as follows:

$$W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v} W^O \in \mathbb{R}^{hd_v \times d_{model}} \tag{4.3}$$

$$d_k = d_v = d_{model}/h = n/h \tag{4.4}$$

The subspace defined by the projection parameters divides the input matrix along the feature dimension n making each head responsible for only a subset of features which leads to the energy reading er not being assigned to all heads. Therefore, some heads deal with determining relationships along contextual features only which is undesirable as the model is trying to predict energy consumption from other features.

In order to address this challenge while taking full advantage of MHA, the proposed approach transforms the input matrix into a higher N-Space before entering the transformer component. Each dimension of the new space becomes a different combination of the original features through the use of a linear transformation such that:

$$I = X \cdot A \tag{4.5}$$

$$I = \begin{bmatrix} er_{11} & c_{11} & ... & c_{1f} \\ ... & & & \\ er_{k1} & c_{k1} & ... & c_{kf} \end{bmatrix} \cdot \begin{bmatrix} A_{11} & ... & A_{1\theta} \\ ... & & \\ A_{n1} & ... & A_{n\theta} \end{bmatrix} \tag{4.6}$$

$$I = \begin{bmatrix} (er_{11}A_{11} + ... + c_{1f}A_{n1}) & ... & (er_{11}A_{1\theta} + ... + c_{1f}A_{n\theta}) \\ & ... & \\ (er_{k1}A_{11} + ... + c_{kf}A_{n1}) & ... & (er_{k1}A_{1\theta} + ... + c_{kf}A_{n\theta}) \end{bmatrix} \tag{4.7}$$

where X is the input matrix of dimension $k \times n$, I is the matrix after transformation, and A are the transformation weights which will be learned during training. The size of the transformed space is determined by θ: the matrix I has dimension $k \times \theta$ after transformation. The size of θ becomes a hyper-parameter of the algorithm, where finding the appropriate value may require some experimentation, this parameter will be referred to as the model dimension parameter.

After transformation, each component in matrix I includes energy readings as seen in equation 4.7. By distributing the energy reading over many dimensions, we ensure that attention deals with the feature the model is trying to predict.

4.4.1.3 Transformer Module

The transformer is composed of an encoder and a decoder module as seen in Fig. 4.1. In this study a stack of two encoders was used. The matrix I created during N-Space transformation is the input to the transformer and the expected output are the energy readings for desired output window.

The encoder processes the input matrix I and the encoder output gets passed to the decoders. Additionally, the decoder receives the expected output window which went through the contextual model and N-Space transformation. After the data travels through the encoder and the decoder, the linear transformation is the last step. The transformation produces the predicted energy consumption by receiving data which has θ dimensions and transforming it into a single value, effectively reversing the N-Space transformation. Its output gets compared to the target output to compute the loss. This loss is then used to compute the gradients and for backpropagation to update weight in the decoder, encoder, and N-Space transformation; this process is depicted by the red arrows in Figure 4.1.

In the original architecture [37], because the decoder receives the expected target value as an input, the input to the decoder was shifted to the right during the training to avoid the decoder learning to predict itself as opposed to the next reading. However, in this adaptation, because of the N-Space transformation, the input value is no longer the same as the target output and this step is no longer needed and is handled by various added transformations. However, the decoder input was still processed using a look-ahead mask, meaning that any data from future time steps were hidden until the prediction are made.

Note that the input contains energy consumption for past time steps while the target output (expected output) is also energy consumption, but for the desired steps ahead. The length of the input specifies how many historical readings the model uses for the forecasting, while the length of the output window corresponds to the forecasting horizon. For example, in one of our experiments we use 24 past readings to predict 12, 24 and 36 time steps ahead.

4.4.2 Load Forecasting with Trained Transformer

Once again, due to the differences in the nature of the input data, further modifications to the workflow of the transformer are required in order to be able to use the trained model for inference tasks. The transformer is composed of an encoder and a decoder module as seen in Figure 4.3 which depicts the inference workflow.

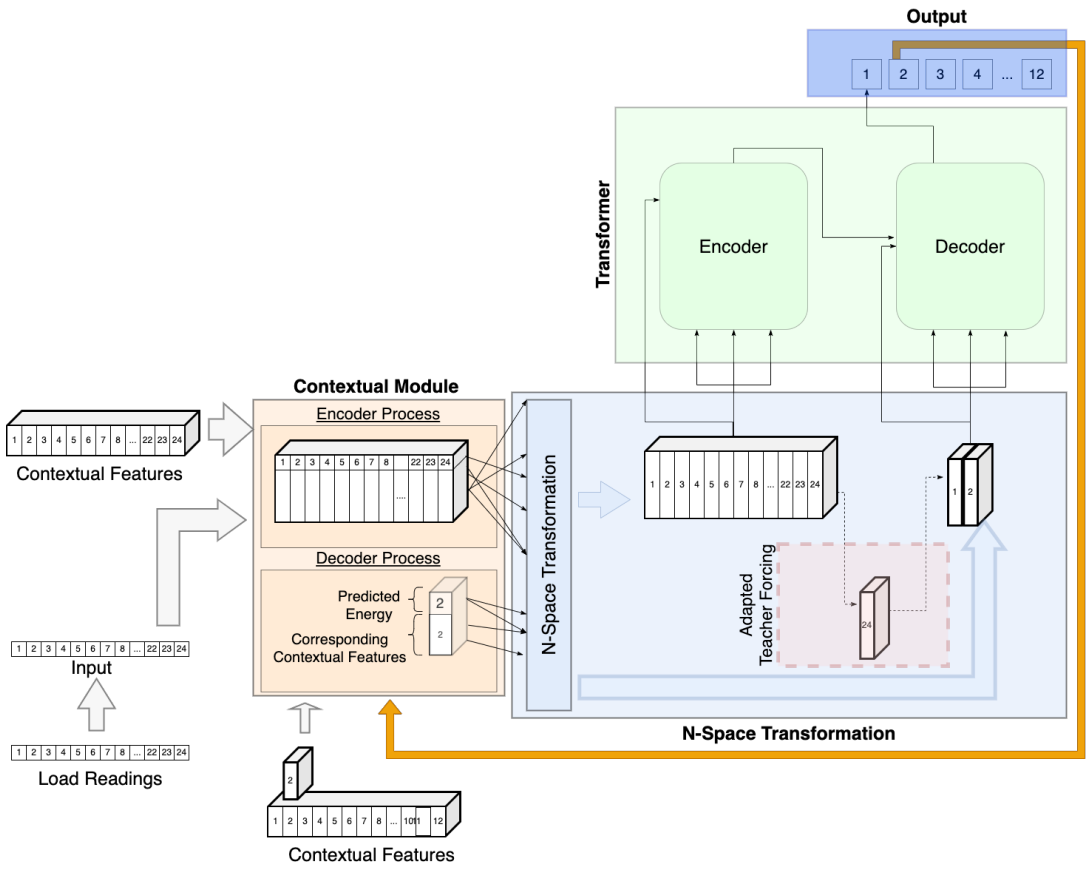


Figure 4.3: Load Forecasting Inference Workflow of the Transformer Architecture

4.4.2.1 Contextual Module

During the training task, the contextual module was used in the same manner for both the encoder and the decoder because, throughout the entire training process, data are fully accessible, meaning that the actual and expected values can be accessed at any time. However, this is not the case for the inference task because we do not have access to future data.

This difference is particularly significant when dealing with transformers because the transformer builds its output one step at a time and uses each step of the forecasting horizon to predict the next. The input of the decoder at each step consists of all of the previously predicted steps. This is problematic in our workflow because, while the expected input consists of load readings along with their contextual component, the output of the transformer consists only of load readings. Therefore, if we feed the transformer output directly into the decoder, the contextual features will be missing. The goal of the contextual module is to address this shortcoming through the separation of the contextual and the load data pathways.

The typical transformer workflow is modified by sending each output step of the trans-

former back through the contextual and N-Space transformation module before they enter the decoder.

The contextual module is responsible for retrieving the corresponding contextual data and creating the f features numerical vectors. This is also the case in the training workflow where it has, however, very little impact. At each step t of the prediction, the contextual data are retrieved and transformed into a contextual vector $c_t = (c_{1t}, c_{2t}, c_{3t}, ..., c_{ft})$ where f is the number of contextual features. By definition, the contextual values such as day, time and even temperature are either known or external to the system.

After the context is retrieved, the predicted energy reading per_t is combined with the contextual vector to create a vector n such that $m = (per_t, c_{1t}, c_{2t}, c_{3t}, ..., c_{ft})$. This vector will then go through the N-Space transformation module.

4.4.2.2 N-Space Transformation Module

The N-Space transformation module behaves in the same way during the inference task as it did during training. The only difference is that it is used after each step t of the output as opposed to once.

After each prediction t, the output vector m, which has a magnitude equal to $f + 1$, is transformed by computing the dot product of the vector m to the transformation weights A such that

$$I_t = m \cdot A \tag{4.8}$$

$$I_t = \begin{bmatrix} per_{t1} & c_{t1} & ... & c_{tf} \end{bmatrix} \cdot \begin{bmatrix} A_{11} & ... & A_{1\theta} \\ ... & & \\ A_{n1} & ... & A_{n\theta} \end{bmatrix} \tag{4.9}$$

$$I_t = \begin{bmatrix} (per_{t1}A_{11} + ... + c_{tf}A_{n1}) & ... & (per_{t1}A_{1\theta} + ... + c_{tf}A_{n\theta}) \end{bmatrix} \tag{4.10}$$

As opposed to during training where the output matrix I was computed once per input sequence, it is now built slowly over time. Before each next prediction, the newly created vector I_t is combined with those previously created within this forecasting horizon such that the decoder input I is:

$$I = \begin{bmatrix} (per_{t1}A_{11} + ... + c_{tf}A_{n1}) & ... & (per_{t1}A_{1\theta} + ... + c_{tf}A_{n\theta}) \\ (per_{(t-1)1}A_{11} + ... + c_{(t-1)f}A_{n1}) & ... & (per_{(t-1)1}A_{1\theta} + ... + c_{(t-1)f}A_{n\theta}) \\ ... & & \\ (per_{(t-q)1}A_{11} + ... + c_{(t-q)f}A_{n1}) & ... & (per_{(t-q)1}A_{1\theta} + ... + c_{(t-q)f}A_{n\theta}) \end{bmatrix} \tag{4.11}$$

Where q is the number of steps previously predicted and $t < horizon$. This input I is then

fed to the decoder.

4.4.2.3 Transformer Module

The transformer is composed of an encoder and a decoder module as seen in Figure 4.3. The encoder's input is a matrix I created by taking a window of historical reading of length k and processing it through the contextual and N-Space transformation module in the same manner described in the training workflow. The encoder output then gets passed to the decoder.

Additionally, the decoder receives the input matrix I described in the previous subsection. However, one further modification is required when the first prediction is made since there is no previous prediction available to feed the decoder. This modification is described as an Adaptive teacher forcing method. The teacher forcing [195] method is a technique used in transformers with NLP where the target word is passed as the next input to the decoder. In our case, such a word is not available therefore we make use of an adaptive teacher forcing technique where the input for the prediction step 1 is equal to the last value of the encoder input, that is the last row of the I input matrix. This step allows us to have an effective starting point for our predictions without compromising future results.

4.5 Evaluation and Results

This section will present the evaluation methodology of our various experiments along with their results.

4.5.1 Evaluation Methodology

In order to evaluate the suitability of the proposed architecture for load forecasting tasks, the solution is evaluated by performing load predictions under various settings and comparing those results with the state-of-the-art method.

In the evaluation, we compare the proposed transformer approach with S2S as S2S has been shown to outperform RNN approaches based on GRUs and LSTMs as well as other neural networks including feedforward neural networks [193]. Note that the original transformer [37] cannot be directly used for comparison as it is designed for text and is not directly applicable to load forecasting. The S2S model with attention [194] achieves very similar accuracy to the S2S model while significantly increasing computational complexity [194]; therefore, the S2S model is used for comparison.

Fekri et al. [174] suggest that it may not be sufficient to evaluate and compare load forecasting results on one data stream and that it may lead to misleading conclusions. Therefore

in order to validate the portability and repeatability of our results, various data streams will be used. Moreover, different input lengths and forecasting horizons must also be considered as the algorithm may perform better or worse depending on the considered input and output lengths.

Consequently, experiments are conducted to examine the transformer's and S2S's behaviours for different forecasting horizons and input sequence lengths. Specifically, input lengths of 12, 24, and 36 steps are considered. For each input window length, forecasting horizons of 12, 24, and 36 steps are chosen. This makes 9 experiments for each algorithm, transformer and S2S, thus, 18 experiments per data stream. Our chosen dataset contains 20 different streams, for a total of 360 experiments.

For each data stream, the last 20% of the data was reserved for testing, while the remainder was used for training. Furthermore, to avoid getting stuck in local minimum [175] training was repeated 10 times for each experiment and both algorithms, with the best performing model being selected.

In order to asses the accuracy of the models, the most commonly used metrics in load forecasting; mean absolute percentage error (MAPE), mean absolute error (MAE) and root mean square error (RMSE) [196] are used. These metrics can be formally defined as follow:

$$\text{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\frac{|y_t - \hat{y}_t|}{y_t} \tag{4.12}$$

$$\text{MAE} = \frac{1}{N}\sum_{t=1}^{N}|y_t - \hat{y}_t| \tag{4.13}$$

$$\text{RMSE} = \sqrt{(\frac{1}{N})\sum_{t=1}^{N}(y_t - \hat{y}_t)^2} \tag{4.14}$$

Where y_t is the actual value, $\hat{y}_t$ the predicted value and N the total number of samples.

4.5.2 Data set and Pre-processing

The experiments were conducted on an open-source dataset [197] containing aggregated hourly loads (in kW) for 20 zones from a US utility company.The zones are defined as different geographical areas with similar characteristics, however no further features about the zones were provided. Therefore, 20 different data streams were used. Each of these streams contains hourly data for 487 days or 11,688 consecutive readings. The unit used for each input and horizon step were therefore hours. The dataset provided the usage, hour, year, month and day of the month for each reading. The following temporal contextual features were also obtained (created from reading date/time): the day of the year, day of the week, weekend indicator,

weekday indicator and season. During the training phase, the contextual information and the load value were normalized using standardization [193] using the following equation:

$$\hat{x} = \frac{x - \mu}{\sigma} \tag{4.15}$$

Where x is the original vector, $\hat{x}$ is the normalized value and μ and σ are the mean and standard deviation of the feature vector respectively.

Once normalized, the contextual information and data streams were combined using the temporal information and then a window sliding technique was applied to create appropriate samples. In order to increase our training and testing set, an overlap of the windows was allowed and the step s was set to 1.

4.5.3 Experiments

The experiments were conducted on a machine running Ubuntu OS, AMD Ryzen 4.20 GHz processor, 128 GB DIMM RAM, and four NVIDIA GeForce RTX 2080 Ti 11GB graphics cards. In order to alleviate the processing time, GPU acceleration was used while training the models. The models were developed using the PyTorch library.

The transformer used in these experiments was made up of a stack of two encoders and decoders, two attention heads and the feed-forward neural network in each encoder and decoder contained 512 neurons. The fully connected hidden layers of the transformer are those of the feed-forward networks embedded in the encoder and decoder. The Adam optimizer [198] was used during training with beta values of 0.9 and 0.98 and epsilon values of $1e-9$ as suggested by Vaswani et al. [37]. The learning rate along with the transformer dropout [199] rate used to prevent overfitting, the N-Space dimension and the batch size used to send data to the transformer was optimized using a Bayesian optimization approach on the Weights and Biases platform [200]. A total of 3477 runs were performed in order to identify the most relevant hyper-parameters. Figure 4.4 shows the importance of each parameter and whether or not it has a positive or negative correlation with optimizing loss.

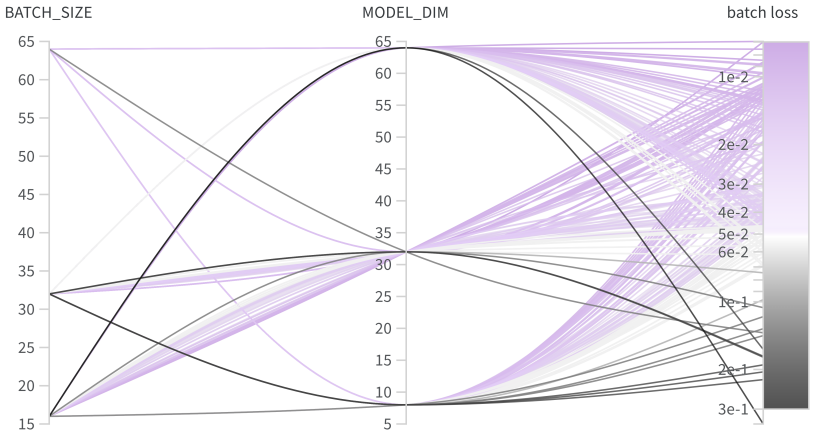


Figure 4.5: Parameter Optimization Visualization

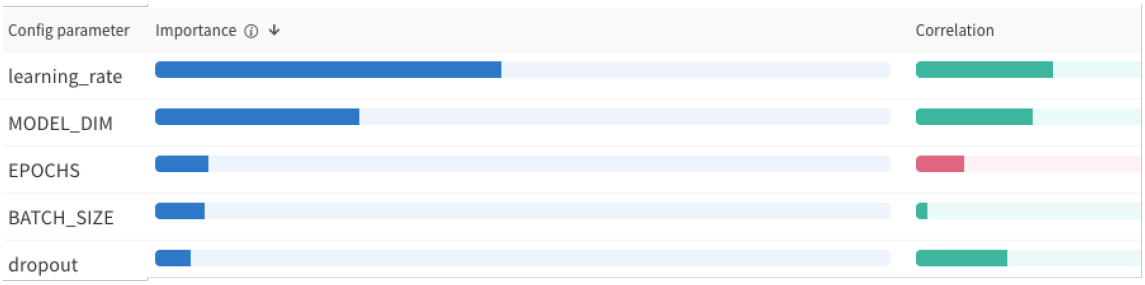


Figure 4.4: Hyper-Parameter Relevancy

Based on the information received by the sweeps, we were able to establish that the models appeared to reach convergence after 13 epochs and that a learning rate of 0.001 and a dropout of 0.3 yielded the best batch loss. However, the optimal N-space dimension (model_dim) and batch size were not as easily determined. A visualization of these parameters shown in figure 4.5 allowed us to choose a batch size of 64 and a model dimension of 32 because they appear to best minimize the overall loss. The model dimension refers to the size θ of the N-Space transformation module.

Lastly, the transformer weights were initialized using the Xavier initialization as it has been shown to be an effective method [201, 202].

The S2S algorithm used for comparison was built using GRU cells, with a learning rate of 0.001, 128 hidden layers and ran for 10 epochs. These parameters were established as successful in previous works [193, 175].

4.5.4 Results

The results obtained in the various experiments are presented and discussed below:

4.5.4.1 Overall

The goal of the experiments is to assess the ability of the transformer to predict electrical load under different circumstances. In order to establish the impact of the size of the input on performance 12, 24 and 36 historical samples were used as input for each experiment. Additionally, since the accuracy on one horizon does not guarantee a good performance in another [174], forecasting horizons of 12, 24 and 36 steps were also tested. Each of these parameters was evaluated in combination for a total of 9 experiments per stream.

The detailed average statistics over all 20 streams can be seen in Table 4.1. It can be observed that 7 out of 9 times, the transformer outperforms S2S.

Input Window	Horizon	MAPE		MAE		RMSE	
		Trans.	S2S	Trans.	S2S	Trans.	S2S
12	12	0.0946	0.1039	6180.46	6262.30	8475.70	8350.62
12	24	0.1035	0.1274	6863.35	7812.95	9695.31	10351.87
12	36	0.1076	0.1397	7045.74	8564.05	9969.60	11267.95
24	12	0.1077	0.0931	6607.10	5622.66	9043.54	7665.20
24	24	0.1118	0.1148	7011.42	7166.48	9717.14	9659.01
24	36	0.1122	0.1271	7267.98	8166.71	10347.00	10860.76
36	12	0.0888	0.0880	5853.80	5465.43	8002.23	7440.19
36	24	0.0928	0.1046	6197.62	6779.88	8649.07	9117.58
36	36	0.0975	0.1276	6794.74	8010.88	9542.88	10660.90

Table 4.1: Average Accuracy over the streams for each combination of input size (in hours) and forecasting horizon (in hours)

The MAPE accuracy differences range from 0.0093 to 0.0321 with the transformer performing 0.0121 better on average when considering all experiments and 0.0178 when considering only the experiments where the transformer outperforms S2S. Note that the RMSE and MAE are sometimes larger for the transformer than S2S while MAPE remains smaller, this is due to the large scale of the data for some meter, which leads the value to thread upwards. This further emphasizes the importance of looking at a number of metrics in order to gain a full picture of the model performance.

The average of all stream results for each of these experiments is shown in Figure 4.6. In this case, only MAPE is shown as MAE and RMSE exhibit similar behaviours.

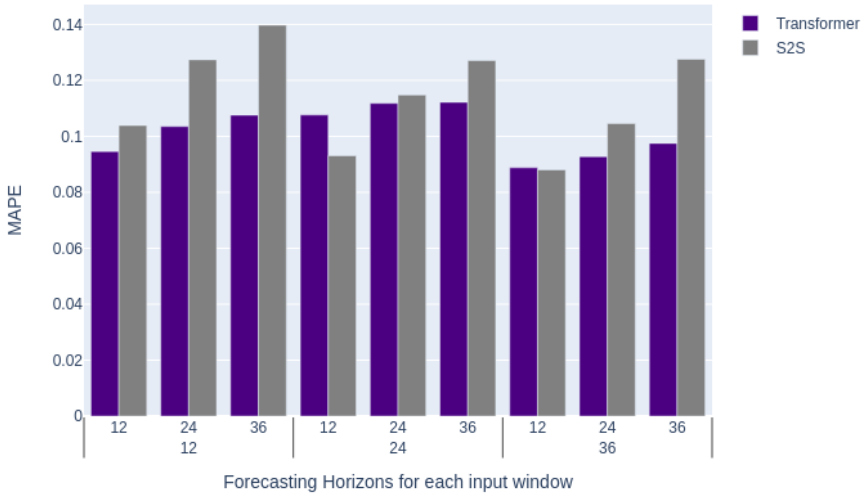


Figure 4.6: Average MAPE comparison over various input and forecasting horizon (in hours)

In Figure 4.6, we can make various observations such that the transformer performs much better than S2S when using a 12h input window but that the results are much closer when dealing with a 24h input window. We can also see that, regardless of the input size, the transformer outperforms S2S significantly when using a 36h horizon.

In order to investigate these observations further, we present the average performance for each forecasting horizon over all streams and experiments in Table 4.2.

Horizon	MAPE		MAE		RMSE	
	Trans.	S2S	Trans.	S2S	Trans.	S2S
12	0.0970	0.0949	6213.7865	5783.4635	8507.1581	7818.6714
24	0.1026	0.1155	6690.7948	7253.1038	9353.8384	9709.4885
36	0.1057	0.1314	7036.1530	8247.2149	9953.1587	10929.8686

Table 4.2: Average performance metric per forecasting horizon (in hours)

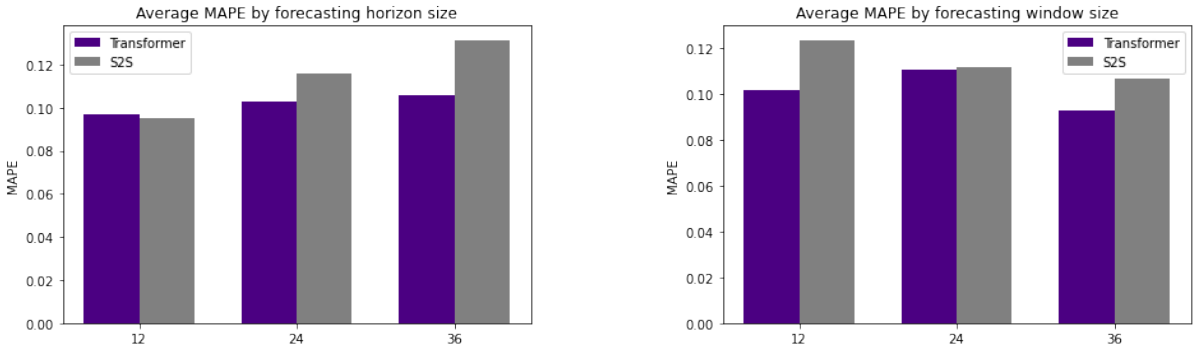
We can observe that the transformer performs better than S2S for longer forecasting horizons but more similarly for a shorter range. When predicting 36 steps ahead, the transformer performs 2.571% better on average than S2S. These results are also highlighted in Figure 4.7a.

Similarly when looking at aggregated results over various input window sizes, as shown in Table 4.3.

Window	MAPE		MAE		RMSE	
	Trans.	S2S	Trans.	S2S	Trans.	S2S
12	0.1019	0.1237	6696.5146	7546.4348	9380.2027	9990.1481
24	0.1105	0.1116	6962.1674	6985.2825	9702.5603	9394.9921
36	0.0930	0.1067	6282.0524	6752.0650	8731.3925	9072.8884

Table 4.3: Average performance metric per input window (in hours)

We can observe that transformers perform better than S2S when provided with smaller amounts of data or shorter input sequences. The transformer outperforms S2S by 2.18% on average using a 12 sample input window. These results are also highlighted in Figure 4.7b. A trend can be observed in this table where it appears that the transformer outperforms S2S when using a 12h and 36h input window but does not when using 24h. This may be explained by the fact that some patterns of consumption are harder to capture than others as suggested by Fekri et al. [174].



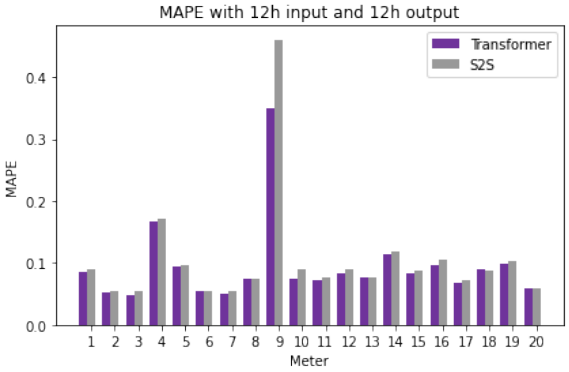
(a) Average MAPE by Horizon (in hours)

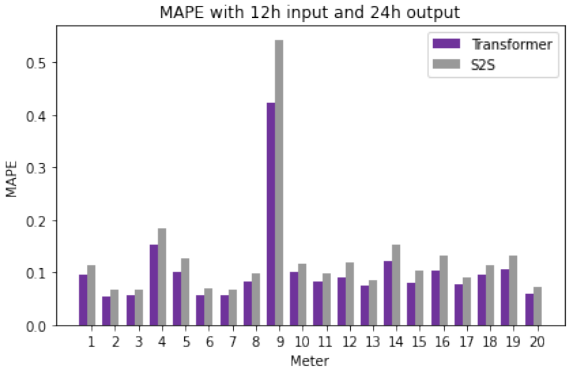
(b) Average MAPE by Window Size (in hours)

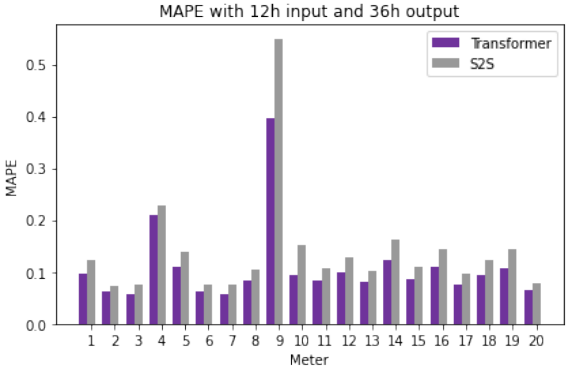
Figure 4.7: Comparisons of Average MAPE

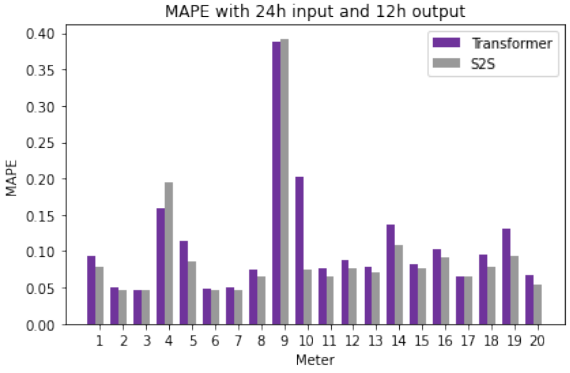
4.5.4.2 Stream based

The previous subsection showcased the overall performance of transformers, however, the accuracy of the individual streams must also be investigated. Figure 4.8, 4.9 and 4.10 depict the performance of each stream under each window sizes and forecasting horizon over the three main metrics: MAPE, MAE and RMSE.


MAPE with 12h input and 12h output
Transformer
S2S
MAPE
Meter


MAPE with 12h input and 24h output
Transformer
S2S
MAPE
Meter


MAPE with 12h input and 36h output
Transformer
S2S
MAPE
Meter


MAPE with 24h input and 12h output
Transformer
S2S
MAPE
Meter

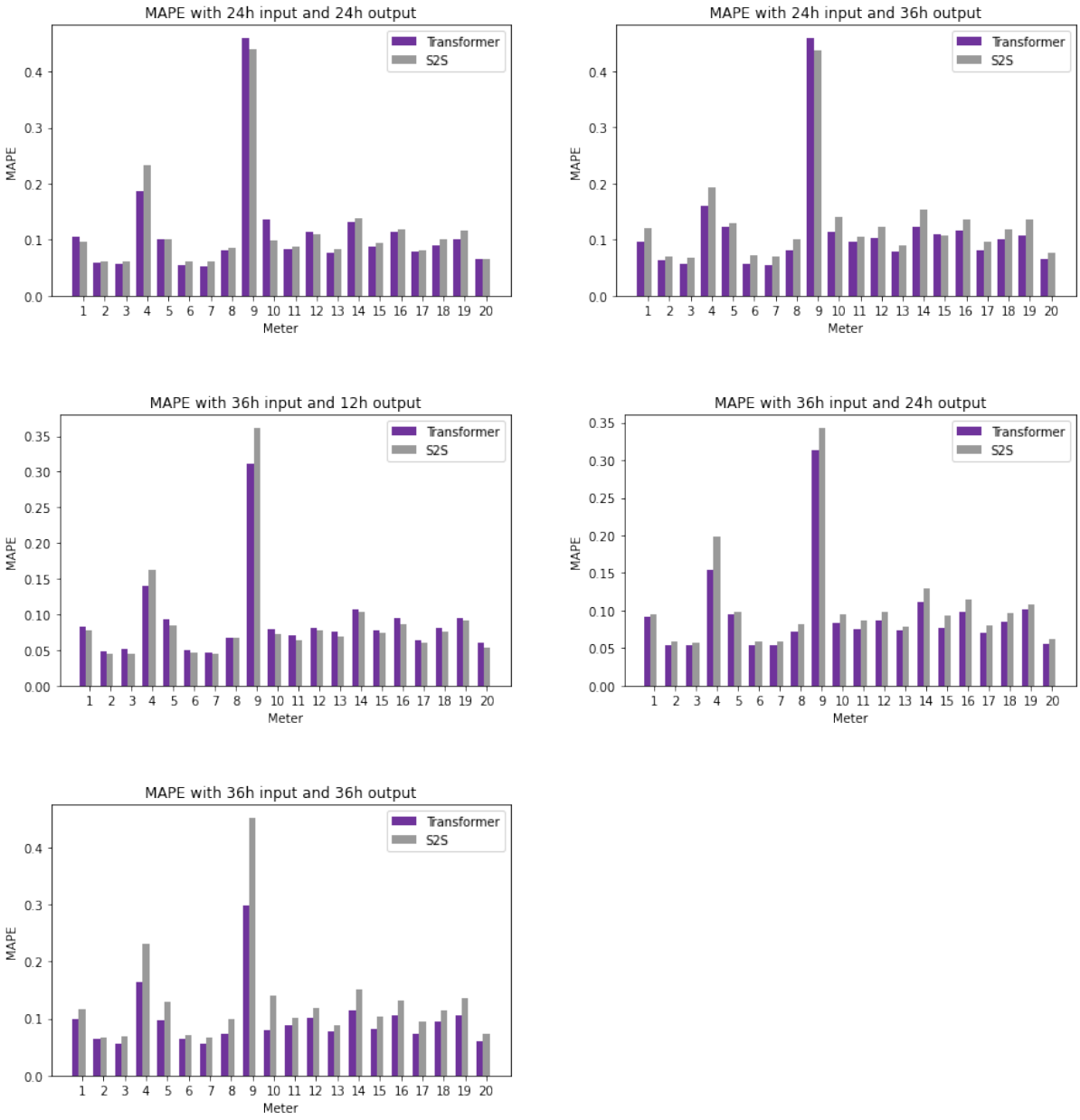
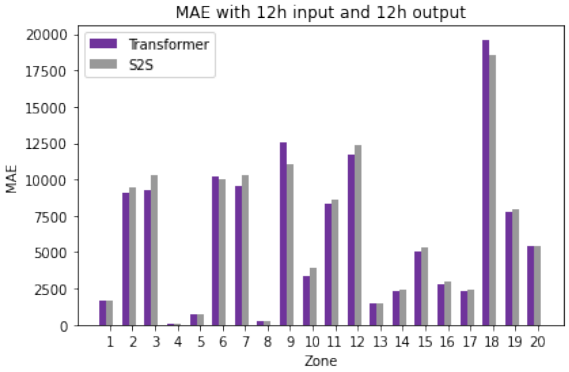
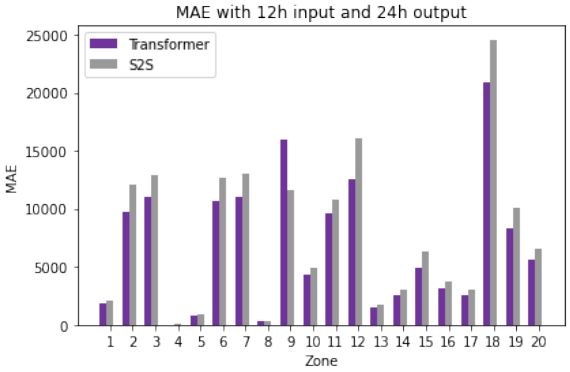
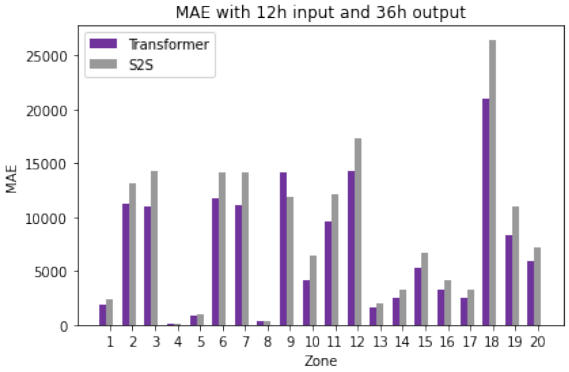
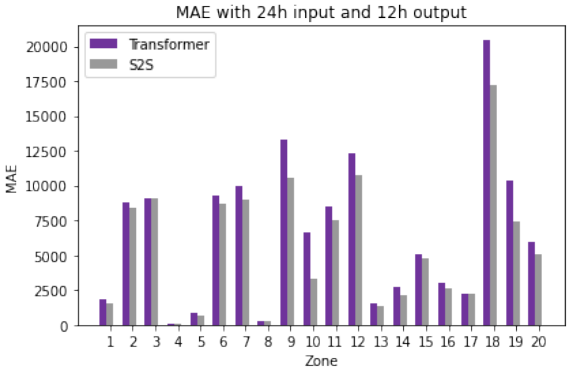


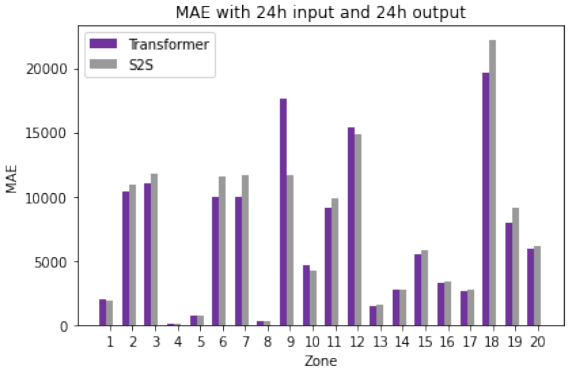
Figure 4.8: MAPE comparison for each Stream under each forecasting horizon and input window size

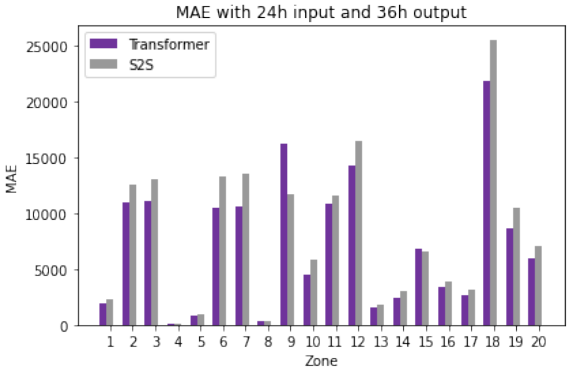

MAE with 12h input and 12h output
Transformer
S2S
MAE
Zone

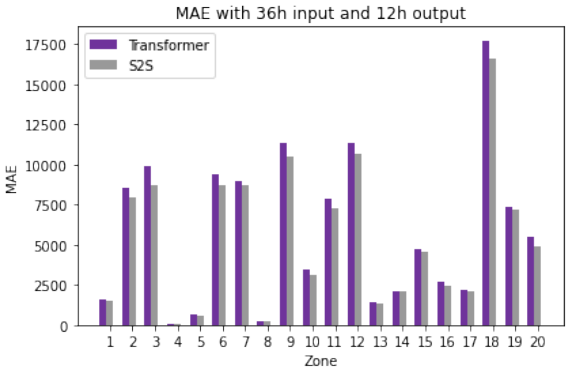

MAE with 12h input and 24h output
Transformer
S2S
MAE
Zone

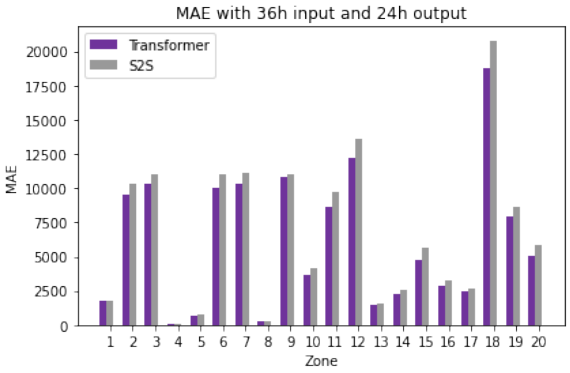

MAE with 12h input and 36h output
Transformer
S2S
MAE
Zone


MAE with 24h input and 12h output
Transformer
S2S
MAE
Zone


MAE with 24h input and 24h output
Transformer
S2S
MAE
Zone


MAE with 24h input and 36h output
Transformer
S2S
MAE
Zone


MAE with 36h input and 12h output
Transformer
S2S
MAE
Zone


MAE with 36h input and 24h output
Transformer
S2S
MAE
Zone

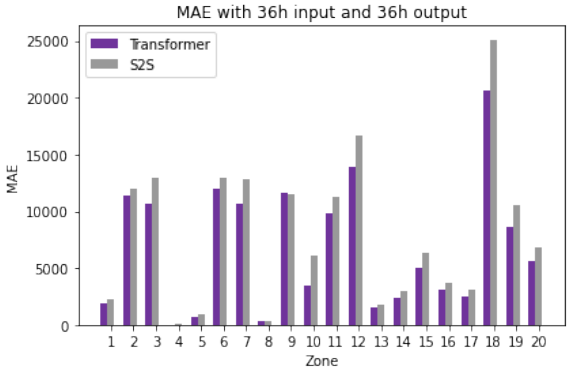
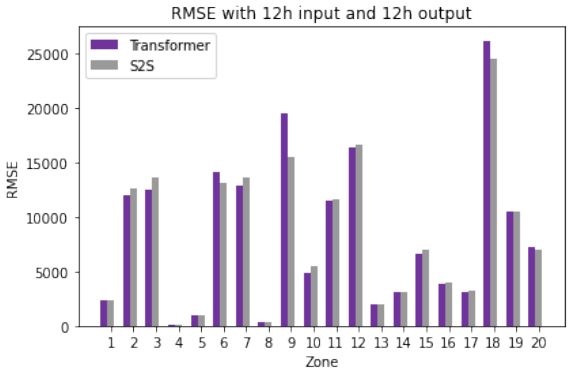
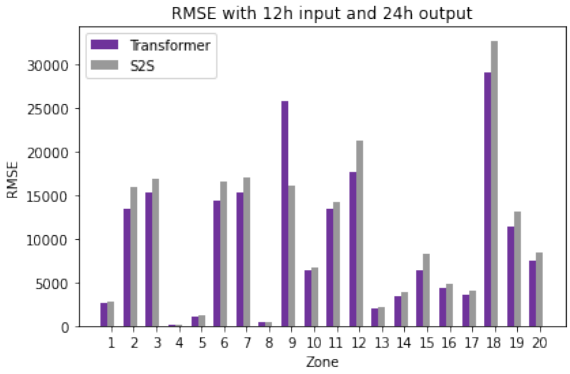
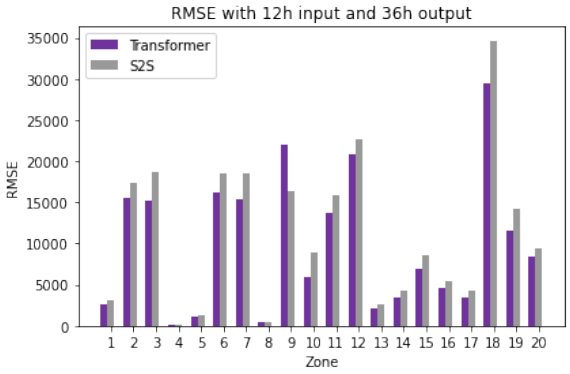
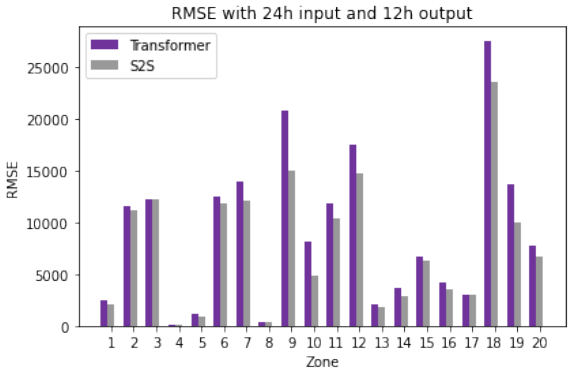


Figure 4.9: MAE comparison for each Stream under each forecasting horizon and input window size









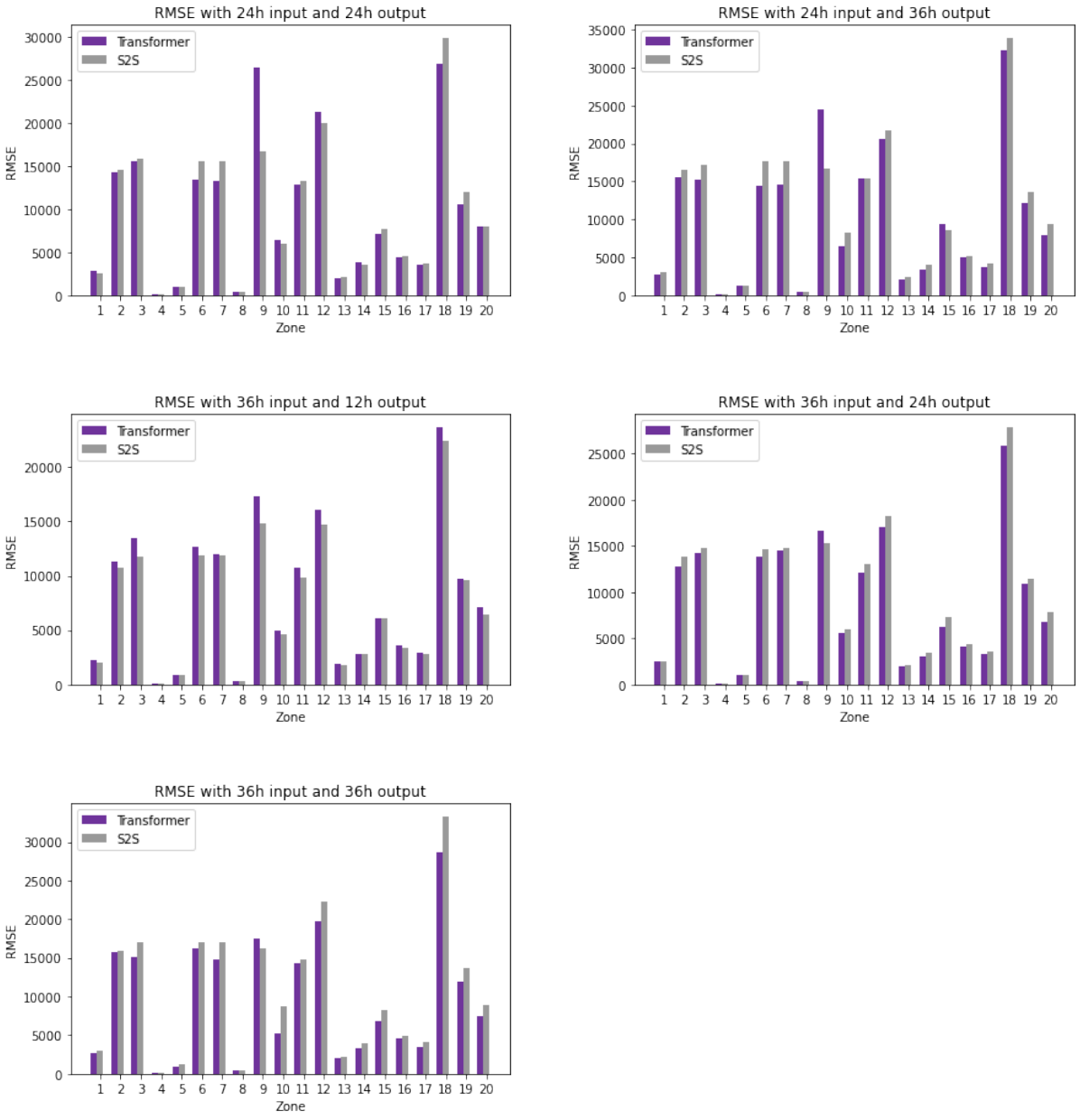


Figure 4.10: RMSE comparison for each Stream under each forecasting horizon and input window size

The visualization of the various metrics highlights some differences in our results. For example, looking at detailed MAPE results shows us that zones 4 and 9 appear to behave differently than others. The MAE and RMSE show a large variance in the ranges of data that we would not have otherwise observed.

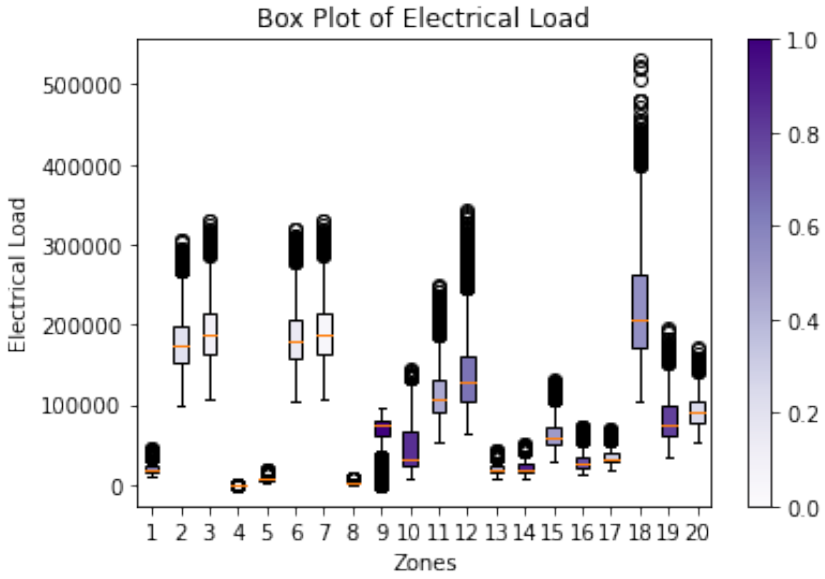
After looking at the detailed results of each experiment, we further investigated the average accuracy of each stream. The results are shown in Table 4.4.

Zone	MAPE		MAE		RMSE	
	Trans.	**S2S**	**Trans.**	**S2S**	**Trans.**	**S2S**
1	0.0939	0.1012	1826.3836	1929.7862	2540.3900	2607.0799
2	0.0561	0.0606	9955.8756	10766.6417	13568.4506	14284.6790
3	0.0539	0.0603	10379.6343	11554.4385	14299.3237	15336.5755
4	0.1660	0.1992	36.4677	36.0893	52.0658	49.6033
5	0.1031	0.1100	761.4848	802.5733	1052.5361	1078.3245
6	0.0560	0.0619	10434.4980	11459.4040	14215.9081	15204.1639
7	0.0531	0.0606	10246.9099	11588.6249	14084.0466	15359.2081
8	0.0766	0.0863	298.2776	337.0904	412.7754	446.2403
9	0.3780	0.4420	13743.3821	11256.9949	21161.6973	15840.7173
10	0.1069	0.1085	4238.9889	4663.3933	6007.8092	6628.5378
11	0.0805	0.0883	9140.0013	9842.5011	12841.2556	13162.4490
12	0.0939	0.1046	13109.0104	14312.1861	18581.9497	19152.9508
13	0.0769	0.0827	1486.8373	1614.8049	1996.4151	2126.6341
14	0.1200	0.1352	2445.5682	2696.8011	3335.8576	3543.0538
15	0.0846	0.0945	5235.9749	5785.0526	6953.8205	7544.2737
16	0.1050	0.1176	3058.6245	3351.4500	4283.1468	4471.7848
17	0.0723	0.0820	2457.7264	2759.0086	3346.5709	3644.6227
18	0.0919	0.1012	20060.7726	21882.9082	27734.4832	29208.1563
19	0.1058	0.1176	8363.9343	9147.0767	11380.2885	12026.7175
20	0.0620	0.0661	5657.8772	6105.0565	7578.9121	8004.4178

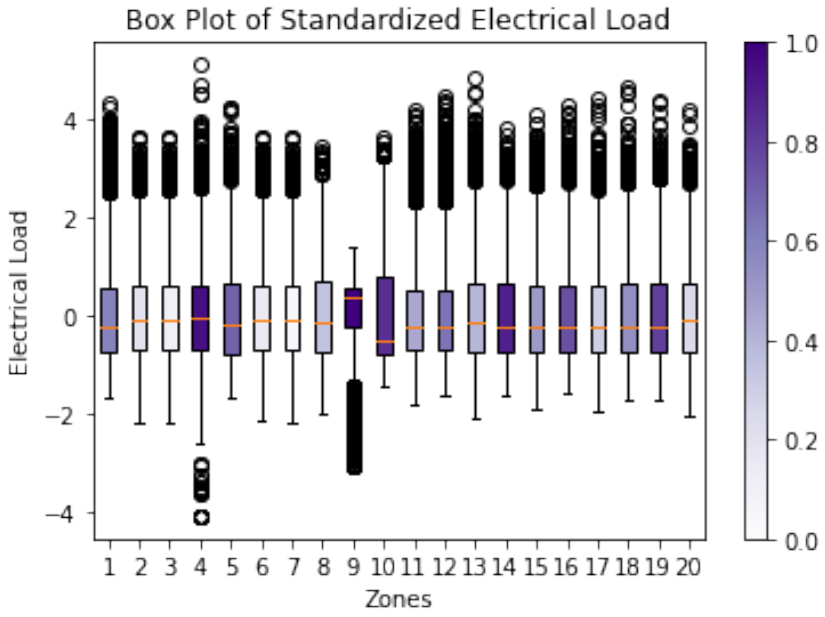
Table 4.4: Accuracy comparison of S2S and Transformer for each Stream

The average MAPE accuracy of each stream, over all window sizes and horizons, is 10.18% for transformers and 11.40% for S2S, but it can be seen in Table 4.4 that it varies from 5.31% to 37.8%. The transformer's MAPE error for the majority of the zones is below 12%. The exceptions are zone 4 and 9 which have higher errors; still, the transformer outperforms S2S.

Based on the observed detailed results for each experiment presented in Figures 4.8, 4.9 and 4.10, we decided to investigate the variability of our data. For each stream, the variance was drawn using both normalized and un-normalized data. The results are shown in Figure 4.11.



(a) Stream Usage data



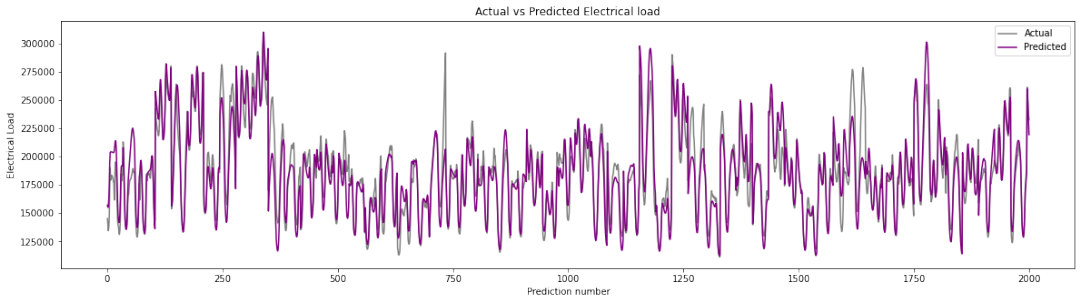
(b) Stream Normalized usage data

Figure 4.11: Box Plot of Stream Data

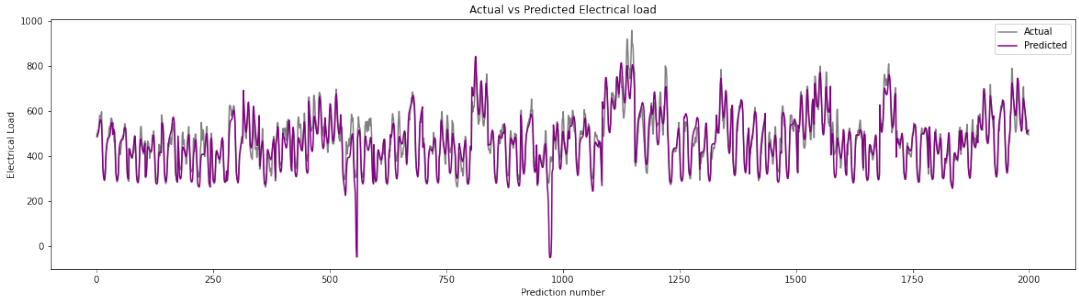
It can be observed that the two streams that appeared to perform differently, 4 and 9, have a high number of outliers close to 0, which can affect the accuracy. This may explain why, regardless of the algorithm used, the error is higher than with other streams.

In order to better assess the suitability of our models, the predicted and actual energy readings using a 36 steps input and 36 steps horizon for the streams for zone 3,4 and 9 are shown in Figure 4.12.

(a) Data from Zone 3



(b) Data from Zone 4



(c) Data from Zone 9

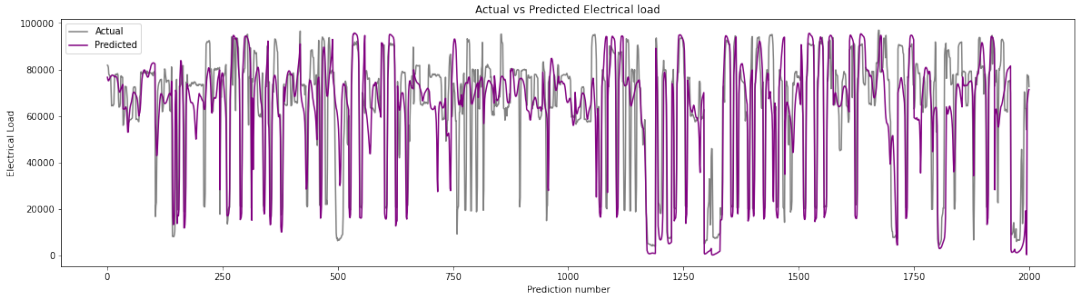


Figure 4.12: Usage Data of different Stream over Time

Each of the models shown have a MAPE of 0.05484, 0.1636 and 0.2975 respectively. The models appear to capture the variations quite well, however, the lower values do appear to create some inconsistencies in the predictions. These graphs only show a subset of the predicted test set but are able to showcase the differences across the various streams such as the differences in range, minimum and maximum values and overall patterns of consumption.

In order to gain insights into the ability of the transformer to perform accurately over various streams using different input sizes, Table 4.5 shows the average MAPE for each stream over each input size.

Zone	**36**		**24**		**12**	
	Trans.	S2S	Trans.	S2S	Trans.	S2S
1	0.0907	0.0960	0.0988	0.0986	0.0922	0.1089
2	0.0552	0.0568	0.0572	0.0596	0.0558	0.0655
3	0.0535	0.0567	0.0539	0.0588	0.0543	0.0655
4	0.1523	0.1967	0.1685	0.2067	0.1773	0.1941
5	0.0942	0.1037	0.1129	0.1059	0.1022	0.1203
6	0.0565	0.0587	0.0535	0.0599	0.0580	0.0670
7	0.0517	0.0568	0.0526	0.0596	0.0549	0.0654
8	0.0715	0.0822	0.0787	0.0842	0.0795	0.0925
9	0.3074	0.3856	0.4360	0.4228	0.3904	0.5174
10	0.0807	0.1028	0.1505	0.1042	0.0894	0.1186
11	0.0776	0.0844	0.0849	0.0862	0.0791	0.0942
12	0.0896	0.0984	0.1016	0.1027	0.0906	0.1126
13	0.0757	0.0786	0.0778	0.0812	0.0772	0.0884
14	0.1106	0.1280	0.1296	0.1330	0.1196	0.1447
15	0.0782	0.0903	0.0929	0.0928	0.0825	0.1004
16	0.0993	0.1110	0.1115	0.1149	0.1042	0.1269
17	0.0694	0.0781	0.0750	0.0809	0.0725	0.0870
18	0.0868	0.0953	0.0957	0.0999	0.0931	0.1083
19	0.1006	0.1114	0.1134	0.1153	0.1034	0.1260
20	0.0585	0.0630	0.0657	0.0658	0.0616	0.0696

Table 4.5: Comparison of average MAPE for each stream over each input size (in hours)

It can be observed that transformers outperform S2S 55 times out of 60. Therefore, we can conclude that transformers can outperform S2S in the majority of cases, regardless of the variance within data streams.

In the context of electrical load forecasting, the statistical significance of the improvement in accuracy is much more open to interpretation than in other fields. This is due to the disconnect between the statistical significance of results and their real-world impact. In many contexts, a statistical significance test would assess a small percentage improvement as non-significant, or not significant enough to disprove the hypothesis. However, it has been shown, that a very small improvement in accuracy can result in considerable real-world impact. For example, it was shown that a simple 1% improvement in a 6-hour horizon forecast lead to

savings of 972$ thousands over six months [203]. For this reason, we consider our gains in accuracy to be meaningful in the context of load forecasting.

4.6 Conclusion

Transformers have revolutionized the field of natural language processing (NLP) and have rendered NLP tasks more accessible through the development of large pre-trained models. However, such advancement have not yet taken place in the field of load forecasting.

This chapter investigated the suitability of transformer for load forecasting tasks and provided a mean of adapting the existing transformer architecture to perform such task. The model was evaluated through comparisons with a state of the art algorithm for energy prediction, S2S, and it was shown to outperform the latter under various input and output settings. Furthermore, the repeatability of the results was successfully challenged by testing the model on a large number of data streams.

On average, the proposed architecture provided a 2.571% MAPE accuracy increase when predicting 36h ahead and and an increase of 2.18% when using a 12h input window. The solution is performing better on longer horizons with smaller amounts of input data. The transformer provided increased accuracy over 90% of the time, while having the potential to be parallelized and further improve performance.

Chapter 5

Scaling Sequence-to-Sequence Models for Load Forecasting

5.1 Introduction

A number of changes in the technological world have made the need for real-time data analytics more pressing. Data-driven enterprises want to have the ability to make decisions and recommendations in real-time in order to remain competitive. The IoT has conditioned consumers to want to gain insights from their various smart devices in a timely, if not immediate, manner. In order to meet these expectations, companies must improve their processing times. However, one of the largest bottlenecks in gaining such access lies in the fact that in order to get faster access to information, algorithms must perform more efficiently to be deployed more quickly. However, existing infrastructure and the associated computational cost requires them to make some trade-offs in terms of the monetary cost, accuracy, and processing times. Therefore, finding means to efficiently improve the scalability of machine learning is needed.

Scalability is defined by Gartner as a system's ability to respond to variations in demands and tasks by adapting its performance and processing cost [204]. Improving the scalability of machine learning models is tedious under regular circumstances, but becomes a much greater issue when dealing with Big Data. One of the means by which machine learning models are being scaled is by modifying their deployment strategy. Indeed, if many instances of a model can be deployed, the system may be able to better handle the demands on the system. The use of distributed and parallel computing platforms is often explored to accomplish this and models are often adapted to run on various parallelization platforms like FPGAs and GPUs or by using concurrent programming frameworks such as CUDA, MPI, MapReduce, and DryadLINQ [205]. However, this type of scaling endeavours often requires more computing resources and

therefore is linked to an increased burden of cost.

The goal of scaling up machine learning models is to enable the models to remain accessible and provide timely results regardless of the conditions and stresses upon which it is placed. Many scaling techniques rely on finding new distributed schemes to deploy models; if a better deployment method can be implemented, new models can be more easily deployed and therefore the demand may be more easily handled [205]. However, such a strategy requires large changes to the ML models or to the application's structure.

The work presented in this chapter proposes to modify and adapt a transfer learning algorithm to lower the training time of load forecasting models and effectively address the performance challenges associated with transfer learning as discussed in Chapter 3 without requiring deeper modifications to the structure or design of existing machine learning applications. Secondly, the adapted S2S model offers a more efficient approach to the task of load forecasting than the state-of-the-art S2S models proposed by Tian et al. [175]. Lastly, we propose a novel evaluation strategy to further ensure the portability of our results.

This chapter is organized as follows: Section 5.2 presents related work, Section 5.3 introduces and details the transfer learning adaptation workflow and a novel technique to provide scalability to the S2S architecture. Section 5.4 presents the novel evaluation strategy and various experiments that demonstrate the scalability of our model under different forecasting conditions.

5.2 Related Work

Sequence-to-Sequence was first described in 2014 by Sutskever et al. [36]. The new architecture was developed for the purpose of translating text from English to French. The idea behind this design was to allow for flexibility in terms of the dimensions of the input and the output. Prior to this innovation, deep neural networks were limited to input and output of a fixed known length. This work enabled the translation of input sequences of various lengths to output sequences of different and unknown lengths. Additionally, the input was processed sequentially, the model consuming one word at a time in the order of the sentence. Due to its sequential nature, this model could not be parallelized and therefore although the accuracy was great, the performance on larger input and the training time were larger.

Adapting the idea of Sequence-to-Sequence models for load forecasting, Sehovac et al. [193] showed promising results by outperforming RNNs and Feed Forward Neural Networks in load forecasting tasks by using LSTM and GRUs in a Sequence-to-Sequence architecture. However, they suggested that because the encoder compresses the entire input sequence into a simple context vector to serve as its memory, that longer input sequence may lead to poorer

results. They improved upon their initial model by successfully adding an attention component to their model to address this shortcoming [194]. However, the added attention component further added to the performance load and although the accuracy improved, the performance was hindered.

To address some of the training performance issues, the approach proposed by Tian et al. [175] made use of transfer learning in a Sequence-to-Sequence model. The similarities between various load forecasting data streams were calculated and those results were used to create a chain order. This chain was used to build an initial model and thereafter transfer the weights and hyper-parameters learned from one zone to another based on the chained results. They made use of a parameter-transfer approach for inductive transfer learning [206]. This technique has been further classified by Tan et al. [207] as a Network-Based Deep Transfer Learning approach using a weight initialization strategy [208]. The results obtained were similar in accuracy to Sehovac's approach [193] but resulted in a significantly faster training time. This work provided a foundation for transfer learning with Sequence-to-Sequence architecture.

Similarly, Skomski et al. [209] explored the simple transferability of Sequence-to-Sequence models by training the models on a single building and then using the trained model with data from another building. Their experiment was conducted on four commercial buildings, their results showed that some pairs performed very well, but that others did not. However, no further explorations of the root causes of the transferability were made. This work laid out the idea that pre-trained S2S models could be used, but it failed to identify under which circumstances.

Ribeiro et al. [210] suggested a transfer learning method that made use of pre and post-processing techniques to perform seasonal and trend adjustments on load forecasting tasks using multi-feature regression. The proposed method can be considered a hybrid solution which combined parameter and instance-based transfer learning. The results were very promising and suggested that one month of transferred data could provide accuracy comparable to that of twelve months of historical data. Therefore, the idea of combining various types of transfer learning is worth exploring. Unfortunately, the testing was limited to a fairly small number of buildings and although accuracy was improved, it did not discuss its impact on efficiency or scaling.

In order to further scale the results obtained using Tian's [175] approach, we propose to add a feature extraction transfer learning strategy as a novel solution. Inspired by Stickland et al. [211] who proposed to pre-train BART, a Sequence-to-Sequence model used in NLP, using the English language and then freeze the various portion of the design in order to improve performance. They attempted to perform machine translation using various embeddings and compared freezing the encoder, decoder and attention mechanism. They received some positive results but mostly noticed the lack of model flexibility. Although some models performed better

than the base model, it was not the case for the majority of the languages. They, however, noted significant improvement in terms of memory demand, since the gradients required by the encoder no longer had to be stored. They also suggested that in a one-to-one translation context, the freezing of the encoder performed better than a random baseline. The method proposed by Tian et al. enabled improvement over the baseline, we suggest that the addition of the frozen encoder could further improve the performance without jeopardizing the accuracy.

5.3 Methodology

In transfer learning, a task can be formally defined as follow [206]:

$$T = \{Y, f(\cdot)\} \tag{5.1}$$

Where Y is the label space and $f(\cdot)$ is the function $f(x)$ used to predict the label for x. If either the label space of the function is different, the task T differs.

Inductive transfer learning is a type of transfer learning where the task T_s performed on the source domain is different than the task T_t performed on the target domain. Meaning that the underlying complex functions of the source and of the target used to predict the loads are different. This is the case when trying to transfer the learning from one zone or building to another.

Pinto et al. [208] further categorized inductive learning for load forecasting into homogeneous and heterogeneous based on the similarities between the source and the target space. The space is defined by its features X and labels Y. A homogeneous space is one where the source and the target features are the same such that $X_s = X_t$ as well as its labels such that $Y_s = Y_t$. On the other hand, a heterogeneous space is one where either $X_s ! = X_t$ or $Y_s ! = Y_t$ or both. An example of heterogeneous space is the work presented by Zhang et al. [212] where the input features of the source domain dealt with time-series data , but the target handled images.

The work presented herein proposes a solution for inductive transfer learning in a homogeneous space of the same domain. A number of strategies could be used to implement transfer learning: instance-based, feature representation-based, model-parameter based or relational knowledge-based [206, 208]. However, the instance-based approach would not be suitable for scaling since it would only increase the data used for training. A relational knowledge-based approach could not be used when considering a homogeneous space within the same domain. Therefore, only a model-parameter approach or a representation-based approach may be suit-

able.

The work presented by Tian et al. [175] makes use of a model-parameter approach to initialize the weights of the decoder and encoder components of the S2S model. Based on calculated similarity, new models make use of the weights and hyper-parameters of previously trained models that are most closely related to them. Building upon this approach, we propose a hybrid strategy which relies on the same similarity measure, but in addition to using the same initialized weights and parameters of closely related models, the entire learned feature representation would also be transferred. The proposed strategy can be considered a feature-representation-based approach because it relies on the idea that related tasks can learn and share a low-dimensional representation. This is typically done by solving an optimization problem over the common features. However, when using a deep learning based model, this inner feature extraction and optimization is actually achieved within some of the deep learning layers.

The proposed strategy relies on the idea that a part of the existing deep learning architecture can be used as a feature extractor. We posit that this generalization, or this way of extracting features, can hold regardless of the load forecasting task T and without performing any explicit optimization tasks. We propose to achieve this feature representation-based transfer learning by freezing the encoder component of the Sequence-to-Sequence architecture. The idea relies on the assumption that in a S2S architecture the purpose of the encoder and its hidden state is to summarize the whole input sequence [213].

Therefore, in the proposed solution the encoder portion of the S2S creates a feature and input representation. Based on data similarity, different representations are created and copies or blueprints of these representations are stored. Through the freezing of the encoder, we are essentially using these stored representations and fine-tuning the model by only training the decoder. The knowledge of how features are to be extracted is transferred across streams through those blueprints.

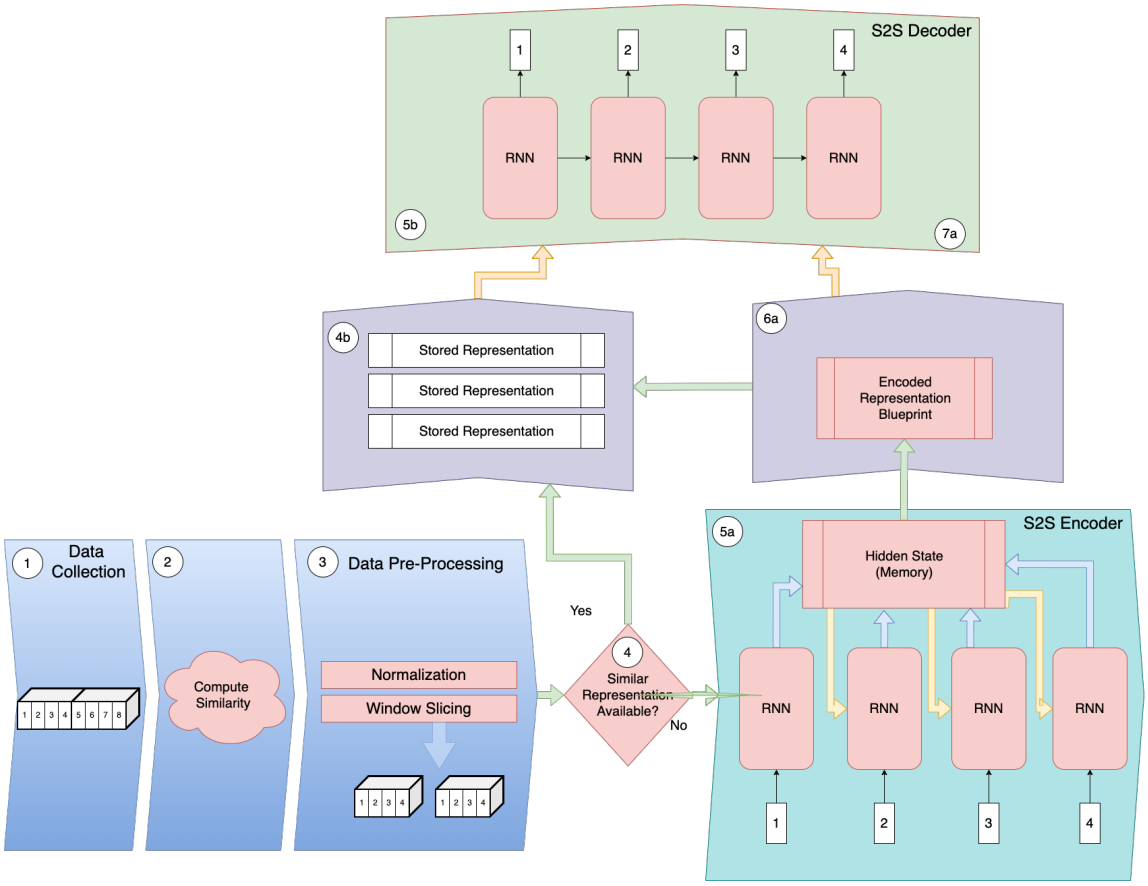


Figure 5.1: Transfer Learning Scaling for S2S Workflow

Figure 5.1 shows the proposed workflow used to scale the transfer learning model. The adapter workflow can be described as follows:

1. Data Collection
   This component is responsible for collecting and storing historical data for each zone.

2. Compute Similarity
   Once the data are collected, or a subset is obtained, the data are compared to the data we have stored for other zones.

3. Data Pre-Processing
   The data are normalized and divided into input windows based on the desired input sequence and desired forecasting horizon. In this implementation, the windows are overlapping.

4. Assess if a Similar Representation is available

The role of the encoder in a sequence to sequence model is to learn how to extract a data representation. A parallel would be how the encoder in S2S models in NLP has been shown to extract the generic meaningful language pattern, and the decoder translates those patterns for a specific language. The obtained representation does not vary when translating to different languages as long as the source language remains the same. Therefore, if the encoder is able to capture the generic representation or the electrical load blueprint, then the decoder should be able to be fine-tuned to infer values for a specific zone, effectively transferring the feature representation from one zone to another. Based on the idea provided by Tian et al. [175] a similarity chain is created and blueprint representations are made for the model designed to serve as seeds, meaning that the models that are most different from one another should get their own blueprints, others will reuse those blueprints and further fine-tune them in accordance with the similarity chain. As new zones become available, the similarity of new data with those of existing blueprints can be assessed and either an existing blueprint can be used or a new one created.

At this point, the workflow is split into 2 separate paths. It is depicted by path A and path B. Path A is used when no data similar enough can be found, and Path B is used when such data are available.

(A) No similar Representation

There are no existing representations, therefore we are at the beginning of the similarity chain and the S2S model is trained traditionally. The data of each input window is then fed sequentially into the encoder (step 5a), then the encoded representation is stored (step 6a), lastly the data are passed to the decoder (step 7a) which will output as many values as set out by the forecasting horizon. With each window, the difference between the output and the expected target would be calculated and the error would be fed back into the model to reduce the loss function and adjust the parameters throughout the entire model (decoder and encoder). Once the encoder is done training, the new representation is then stored and new data may use it if they are found to be most closely related to it (step 6a).

(B) Transfer

If existing representations are available, the most closely related blueprint should be used. In this implementation, the Euclidean distance was used to assess data similarity on the unprocessed time-series data. The previously trained blueprint is then loaded (step 4b), the encoder portion of the model is then frozen and training is continued using the

new zone data. The data are then processed through the frozen encoder and then fed into the decoder (step 5b). When the data are outputted, the difference between the expected output and the output is calculated and the loss function is used to optimize only the parameters of the decoder.

5.4 Experimental setup and result discussion

This section will first describe and analyze the dataset, then detail the conducted experiments and lastly discuss the corresponding results.

5.4.1 Dataset Description and analysis

The following subsection will introduce the dataset used to perform our experiments as well as our novel evaluation methodology used to suggest better result portability.

5.4.1.1 Dataset

The experiments were conducted on an open-source dataset [197] containing aggregated hourly loads (in kW) for 20 zones from a US utility company. Therefore, 20 different data streams were used. Each of these streams contains hourly data for 487 days or 11,688 consecutive readings. The dataset provided the usage, hour, year, month and day of the month for each reading. The following temporal contextual features were also obtained: the day of the year, day of the week, weekend indicator, weekday indicator and season. During the training phase, the contextual information and the load value were normalized using standardization.

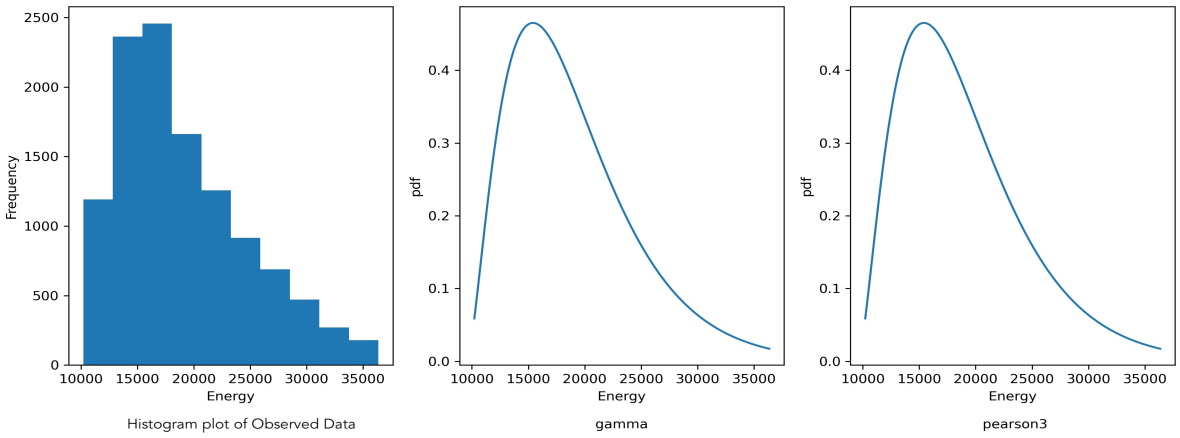
This dataset was chosen for two main reasons: firstly to compare our results more accurately to those obtained by Tian et al. [175] who also used this set. Secondly, the large number of included streams enables us to assess the portability of results. We suggest that evaluating an algorithm on a single data source, regardless of its size, may impact the repeatability of the results since the underlying patterns and distribution found within the data may change significantly between sources. Although the impact of such changes has not been formally quantified, we suggest that repeating the evaluation over data sources from various underlying distributions can better ensure the portability of our results. Often times in the field of load forecasting, algorithms are evaluated using data coming from a single source and although there is a large amount of data, the data may not be varied enough to know if the solution could be used effectively with a different dataset.

5.4.1.2 Result Analysis

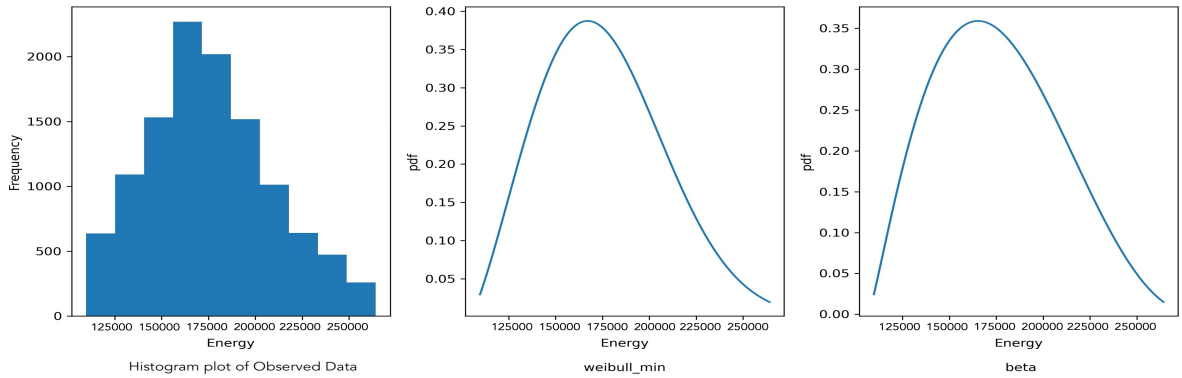
In order to ensure the portability of our result, each zone of the dataset was analyzed to see if it contained different data distributions. The energy dimension of each zone was standardized in order to enable better comparison across each data subset while retaining the overall scale of the energy readings. Once standardized, the data was separated into 10 bins and a histogram of each zone was created. The binning technique is used to treat our continuous values as categorical. From there, each set was fitted through with the following continuous distributions: Weibull Minimum Extreme Value, Normal, Weibull Maximum Extreme Value, Beta, Inverse Gaussian, Uniform, Gamma, Exponential, Lognormal, Pearson Type III and Triangular. For each of the distributions, the parameters were stored and the chi-squared metric was computed. An approximation of the goodness of fit test was performed and the two distributions with the lowest chi-squared were selected as the best fit. This is the best approximation and also, given that all were evaluated with the same candidate distributions, it enables us to see if the underlying distributions vary across the zones. Because the objective of comparing the data distribution is to examine the similarities and differences between the zones, we found that an approximation of the distribution was appropriate.

Figure 5.2 shows the histograms and the two best-fitted approximations for each zone.

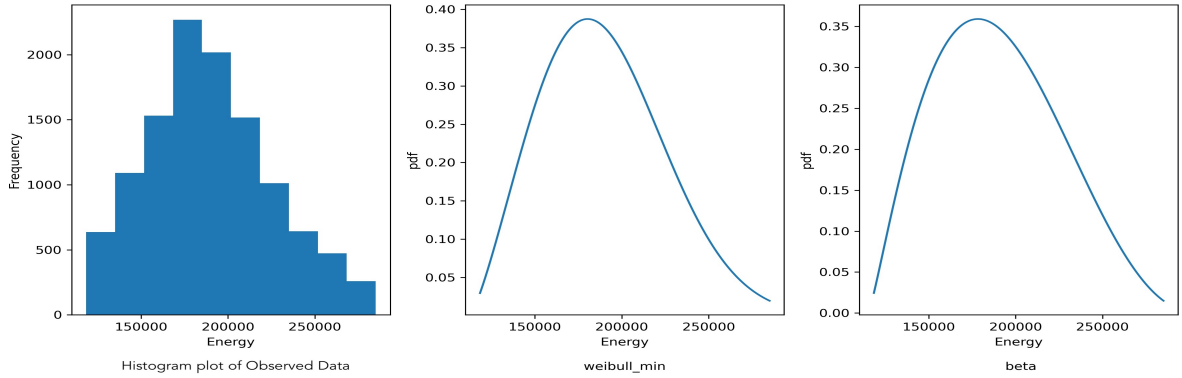
(a) Data Distribution for zone 1



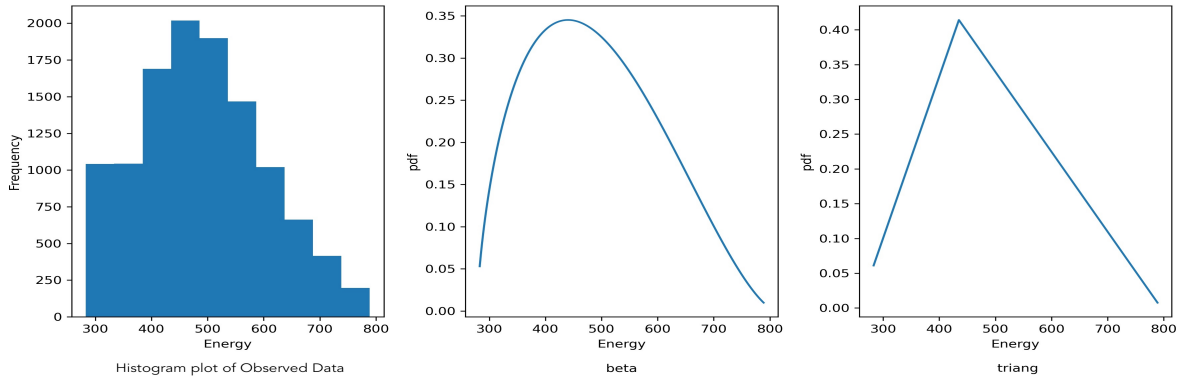
(b) Data Distribution for zone 2



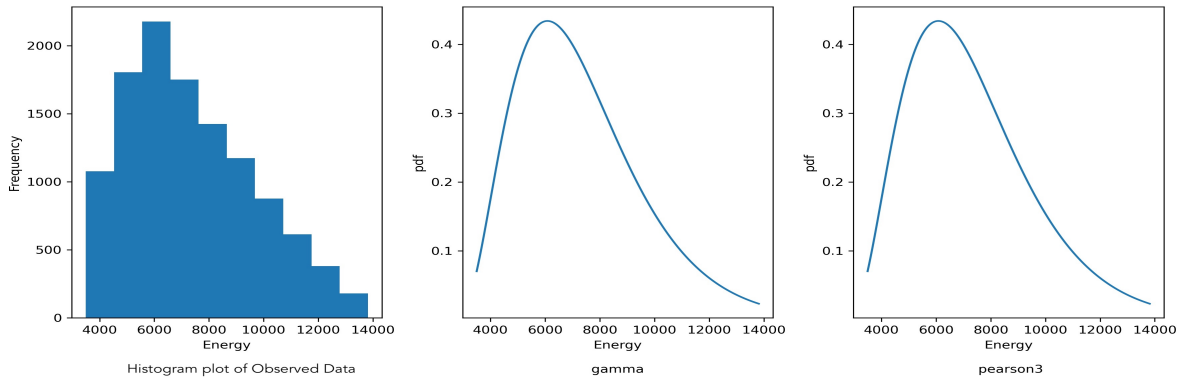
(c) Data Distribution for zone 3



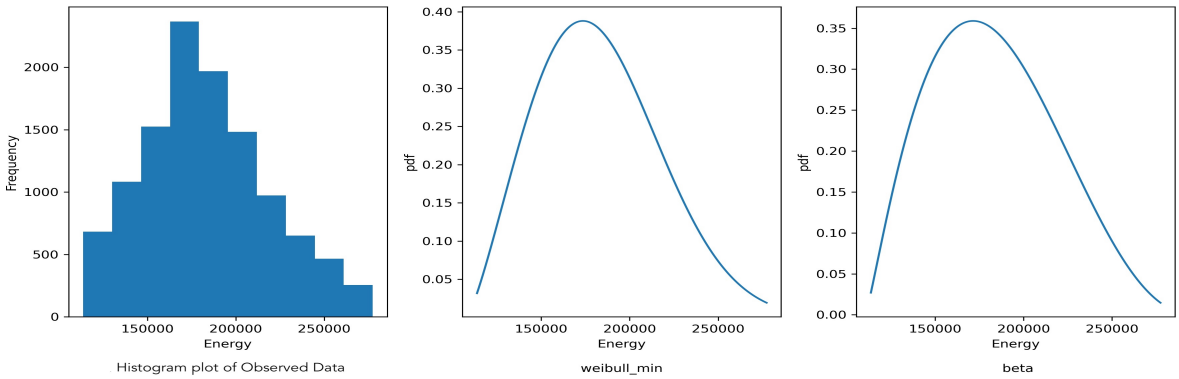
(d) Data Distribution for zone 4



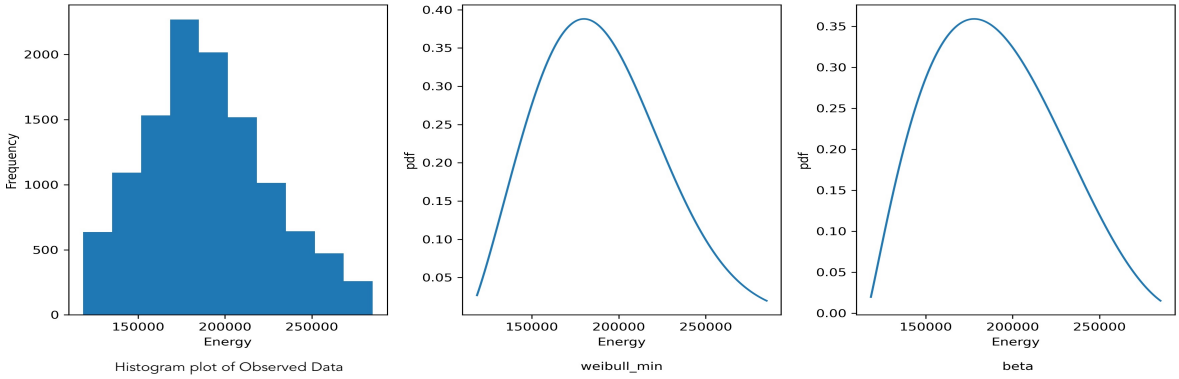
(e) Data Distribution for zone 5



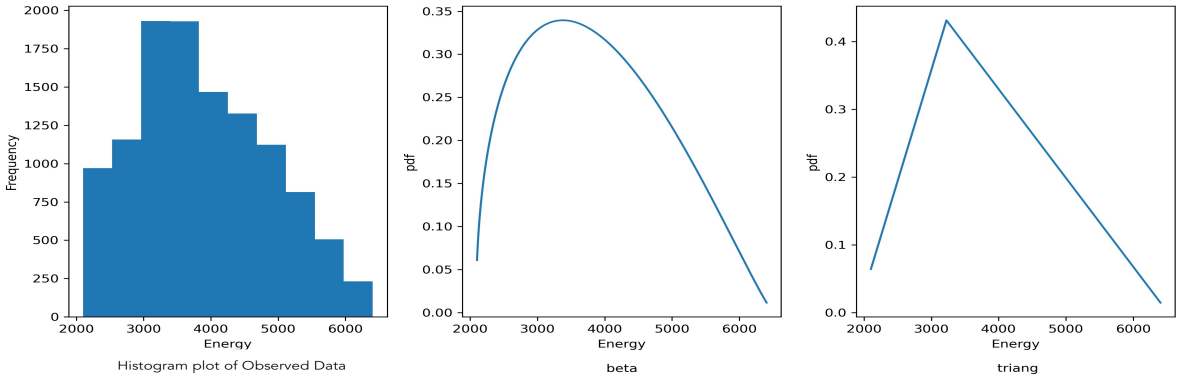
(f) Data Distribution for zone 6



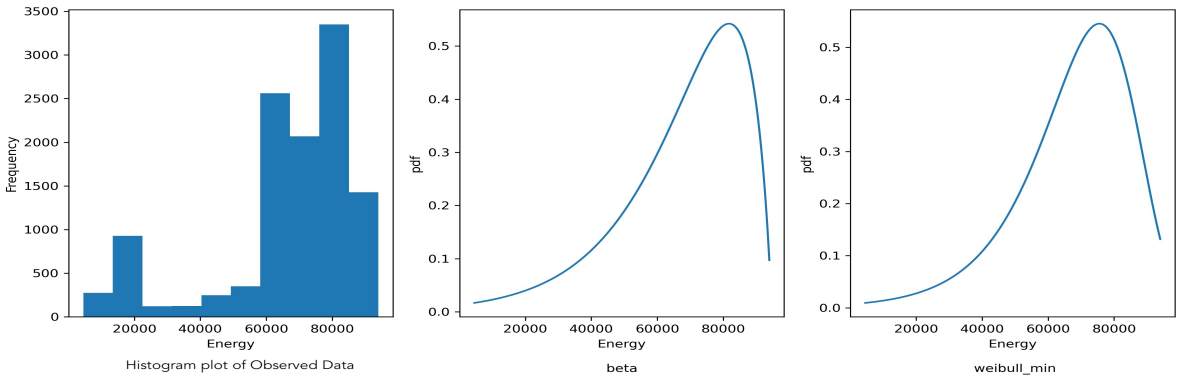
(g) Data Distribution for zone 7



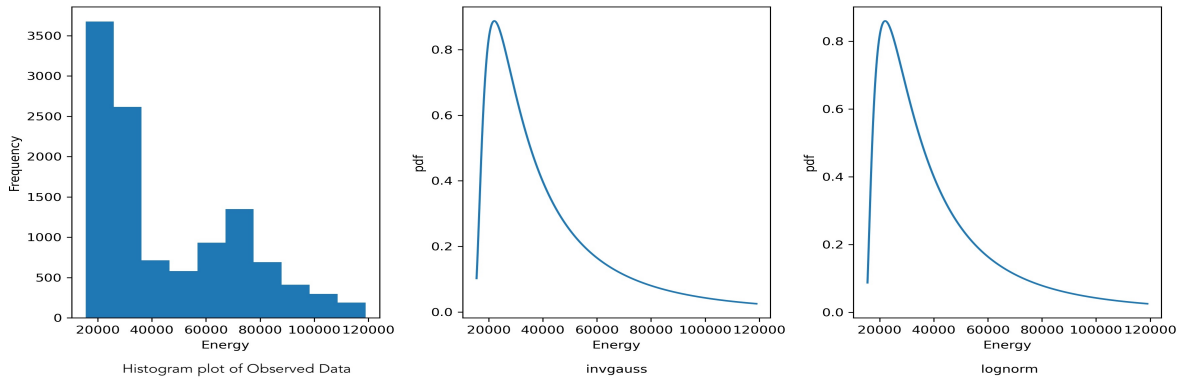
(h) Data Distribution for zone 8



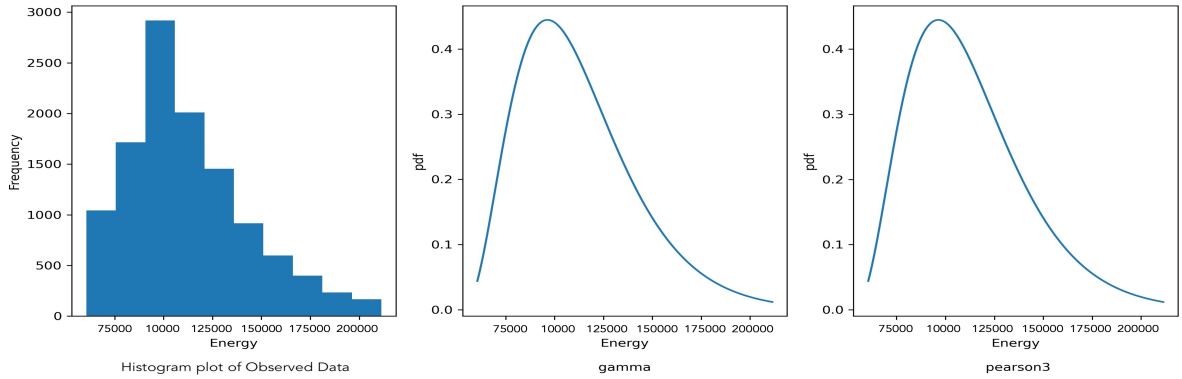
(i) Data Distribution for zone 9



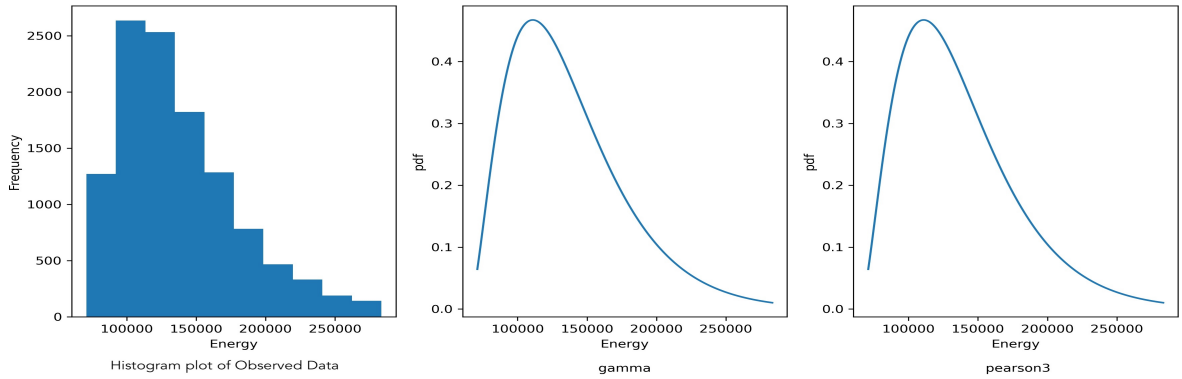
(j) Data Distribution for zone 10



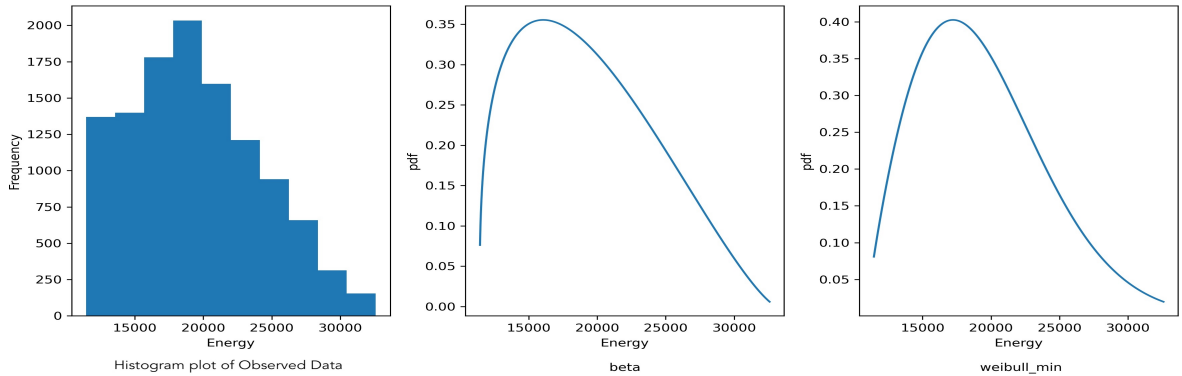
(k) Data Distribution for zone 11



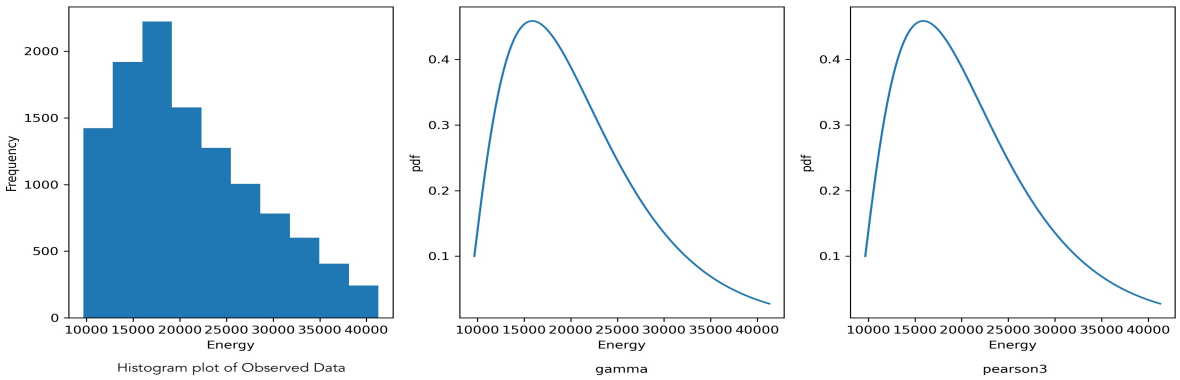
(l) Data Distribution for zone 12



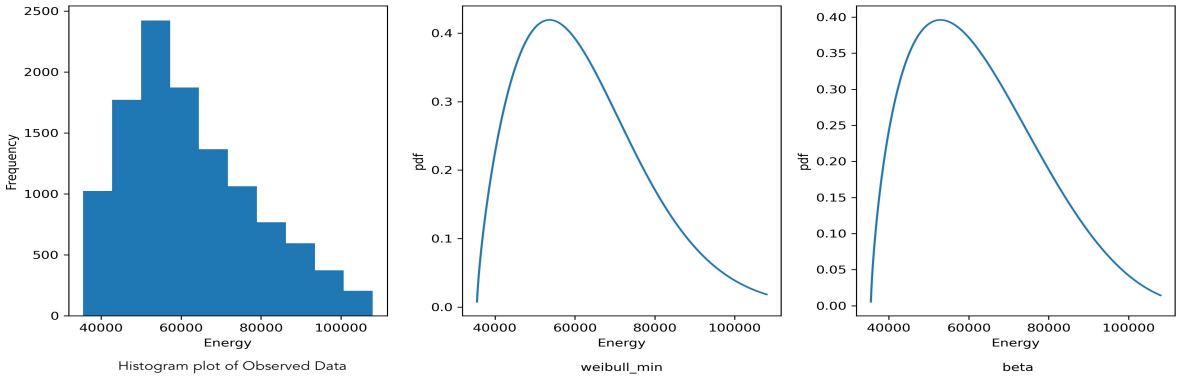
(m) Data Distribution for zone 13



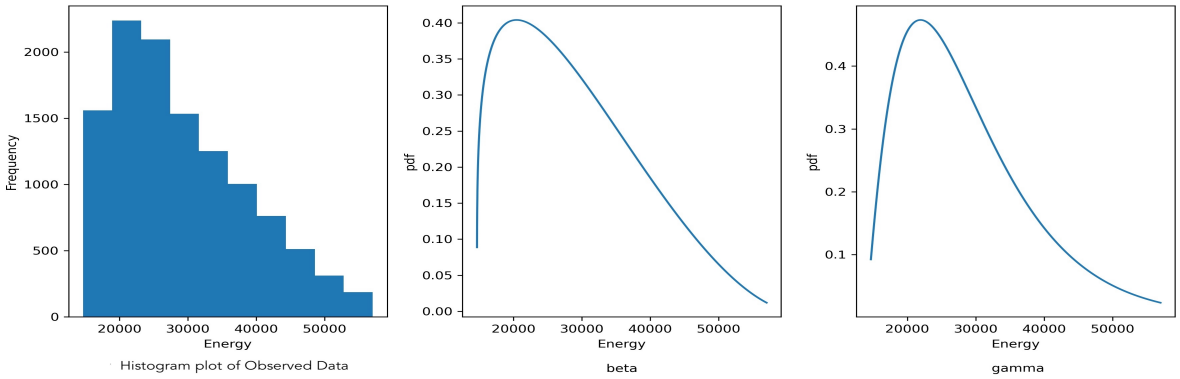
(n) Data Distribution for zone 14



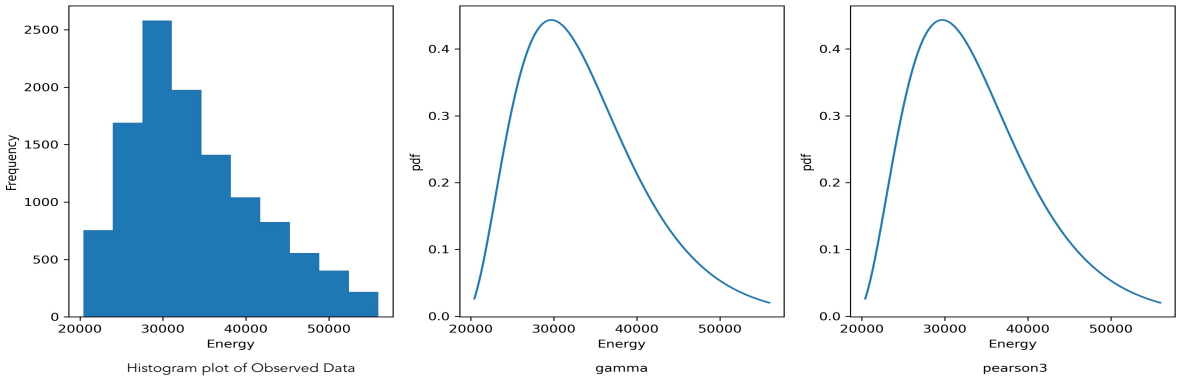
(o) Data Distribution for zone 15



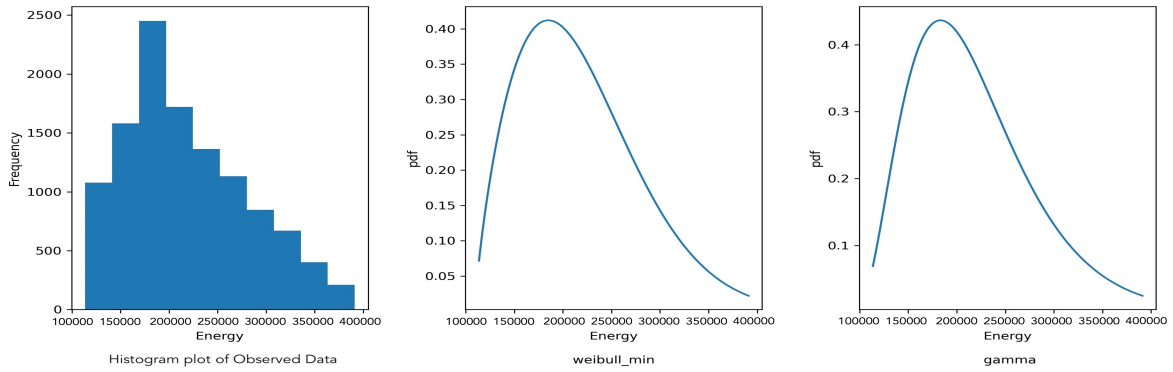
(p) Data Distribution for zone 16



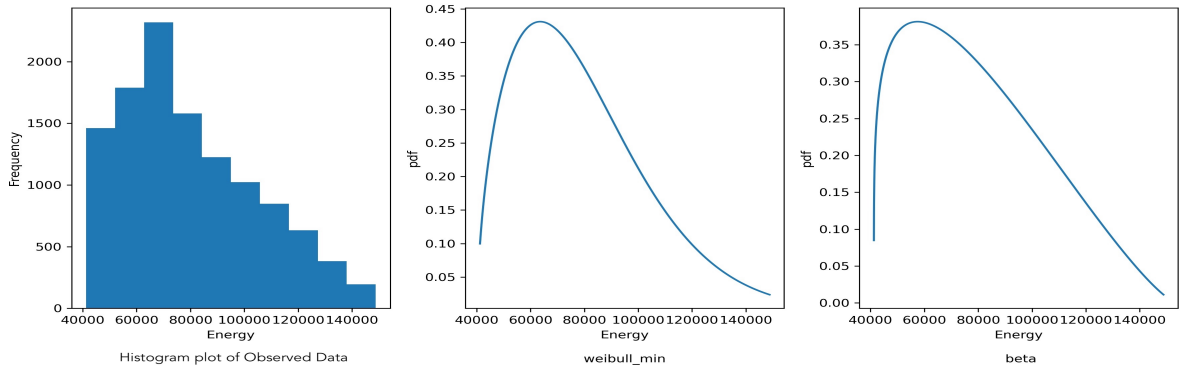
(q) Data Distribution for zone 17



(r) Data Distribution for zone 18



(s) Data Distribution for zone 19



(t) Data Distribution for zone 20

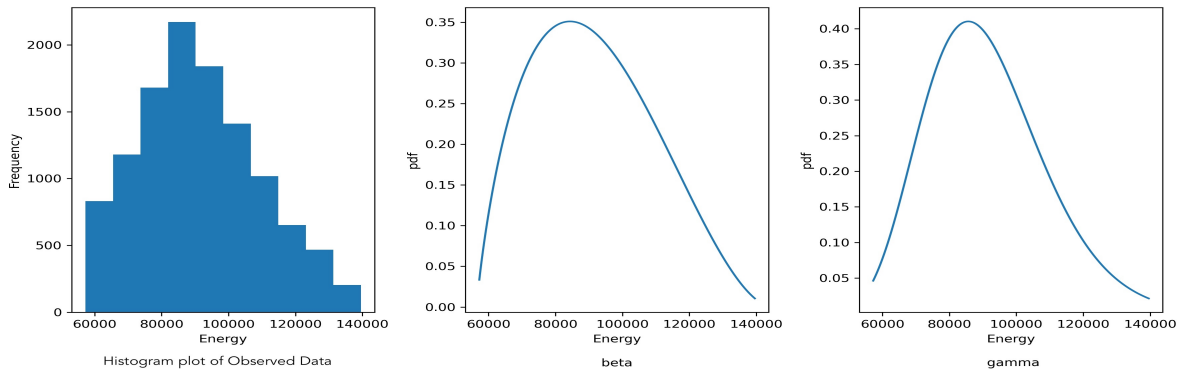


Figure 5.2: Approximated distribution of the electrical load for each zone

It can be observed that a number of underlying distribution was uncovered. Table 5.1 shows the best two fitted distributions for each zone. Based on those results, we suggest that the dataset contains enough variance to ensure result portability. This will be further evaluated

when compared with the model's performance in the following section.

Zone	1st Distribution	2nd Distribution
1	gamma	pearson3
2	weibull_min	beta
3	weibull_min	beta
4	beta	triang
5	gamma	pearson3
6	weibull_min	beta
7	weibull_min	beta
8	beta	triang
9	pearson3	weibull_min
10	invgauss	lognorm

Zone	1st Distribution	2nd Distribution
11	gamma	pearson3
12	gamma	pearson3
13	beta	weibull_min
14	gamma	pearson3
15	weibull_min	beta
16	beta	gamma
17	gamma	pearson3
18	weibull_min	gamma
19	weibull_min	beta
20	beta	gamma

Table 5.1: Each zone and their corresponding best-fitting underlying distribution

5.4.2 Experiments

A number of experiments were conducted to evaluate our proposed solution. These experiments and their purpose are described in this sub-section.

5.4.2.1 Experiment 1

The first experiment we conducted was to compare the performance of the scaling methodology to the published S2S with transfer learning by Tian et al. [175]. A comparison of the following three models was made:

- S2S with transfer learning from literature [175]
- S2S with parametric transfer learning, an implementation based on the work of Tian et al. [175]
- S2S with scaling, our proposed solution

The results for zone 7 and 3 are not present as the author of the transfer method removed them from his results sets as he used them as the initial transfer seed, therefore they were also removed for consistency. The results are presented in Table 5.2.

Figure 5.3 shows the MAPE accuracy, after every training epoch, for each zone for all three of these models. It can be observed that the published results and those obtained by the S2S model with parametric transfer learning are very similar. Although, the scaling method

Zone	MAPE Scaling	MAPE S2S Tr.	MAPE S2S Lit.	Time Scaling	Time S2S Tr.	Time S2S Lit.
1	0.0625	0.0512	0.0465	0.3491	0.5767	0.8707
2	0.0383	0.0357	0.0322	0.4025	0.6239	0.8647
4	0.3947	0.4316	0.4298	0.3527	0.6005	0.8757
5	0.0770	0.0631	0.0620	0.3568	0.6152	0.8557
6	0.0395	0.0369	0.0331	0.3805	0.6253	0.8747
8	0.0591	0.0487	0.0462	0.3351	0.6025	0.8757
9	0.2355	0.2247	0.2187	0.3709	0.6335	0.8727
10	0.0545	0.0393	0.0409	0.3433	0.7013	0.8866
11	0.0514	0.0387	0.0385	0.3510	0.6066	0.8866
12	0.0594	0.0483	0.0471	0.3563	0.6075	0.8757
13	0.0722	0.0722	0.0706	0.3411	0.5737	0.8408
14	0.0928	0.0912	0.0917	0.3559	0.6243	0.8487
15	0.0721	0.0712	0.0716	0.3576	0.5968	0.8727
16	0.0745	0.0741	0.0735	0.3514	0.5852	0.8667
17	0.0459	0.0425	0.0414	0.3434	0.5779	0.8936
18	0.0648	0.0600	0.0604	0.3567	0.6097	0.8777
19	0.0845	0.0803	0.0796	0.3464	0.5819	0.8298
20	0.0489	0.0467	0.0465	0.3351	0.6225	0.8677

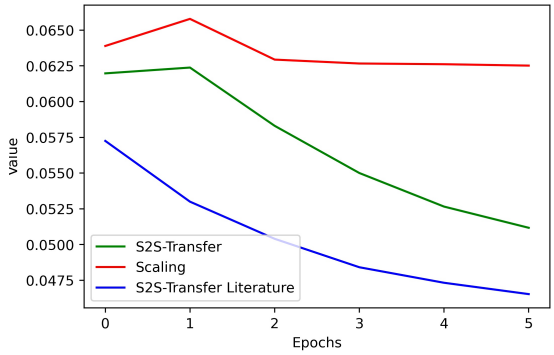
Table 5.2: Results in terms of time and MAPE for all three models

performs with slightly lower accuracy. However, Figure 5.4 shows the time in seconds, at each epoch, used to train the models where it can be seen that the scaling method performs much faster. The models were developed using Python and the PyTorch framework and were conducted on a machine running Ubuntu OS, AMD Ryzen 4.20 GHz processor, and 128 GB DIMM RAM.

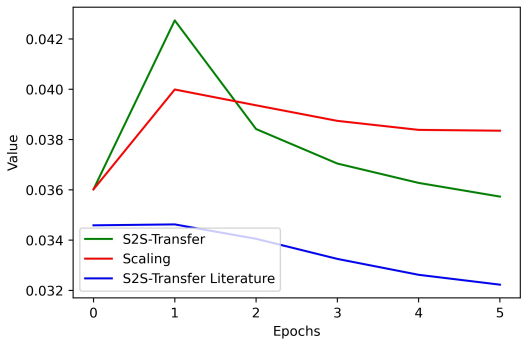
On average, the scaling method took 0.3547 sec to train (per epoch) in comparison to 0.8686 for the literature comparison and 0.6091 for our basic S2S. Therefore, although there was a small loss in accuracy of 0.0054 on average, less than 1%, the new model was trained over 1.7 times faster.

In order to allow for consistency in the comparison with Tian et al. [175], those experiments were performed using a hidden size of 64, a batch size of 256, an input of 8h and a forecasting horizon of 4h.

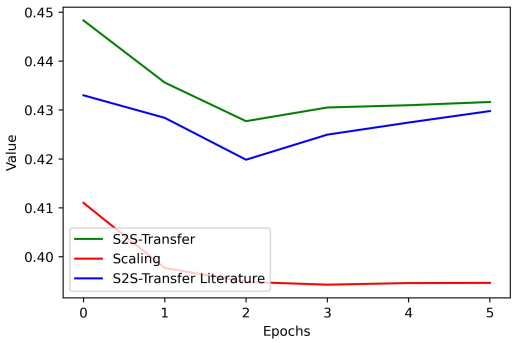
(a) MAPE Comparison for zone 1



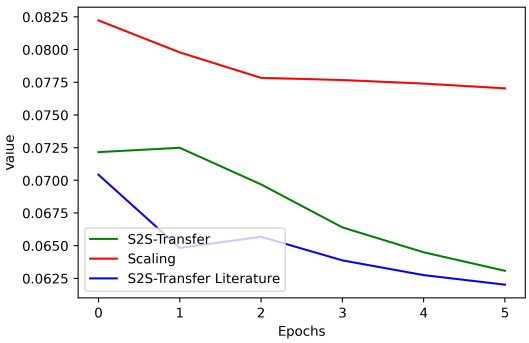
(b) MAPE Comparison for zone 2



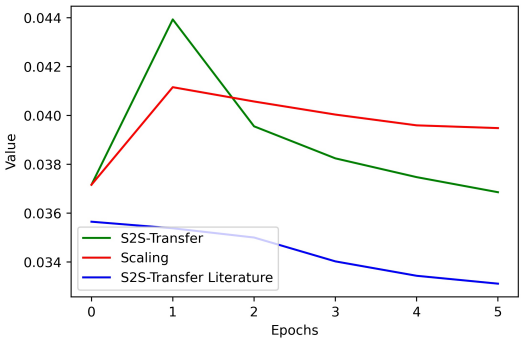
(c) MAPE Comparison for zone 4



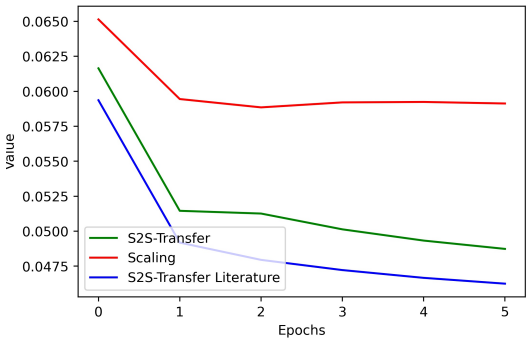
(d) MAPE Comparison for zone 5



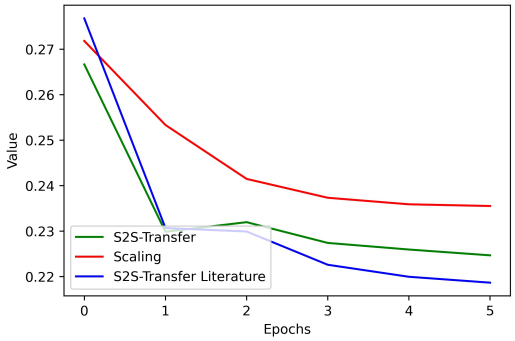
(e) MAPE Comparison for zone 6



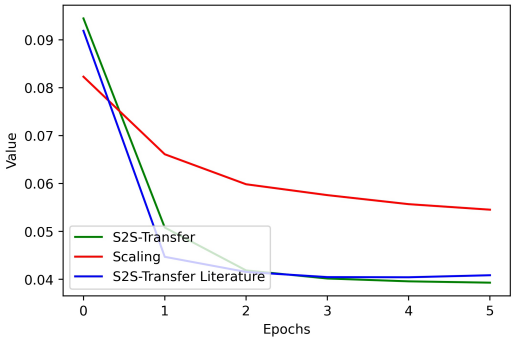
(f) MAPE Comparison for zone 8



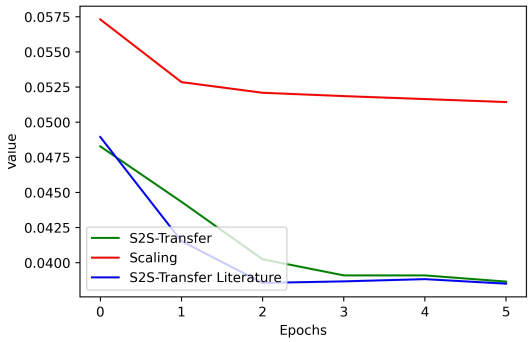
(g) MAPE Comparison for zone 9



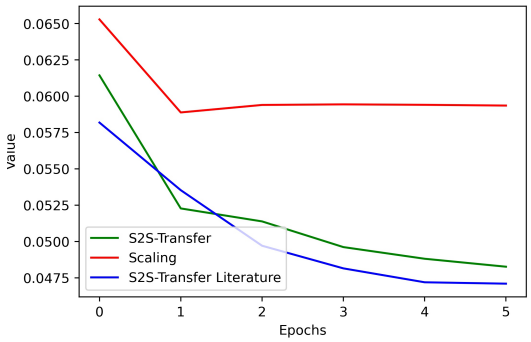
(h) MAPE Comparison for zone 10



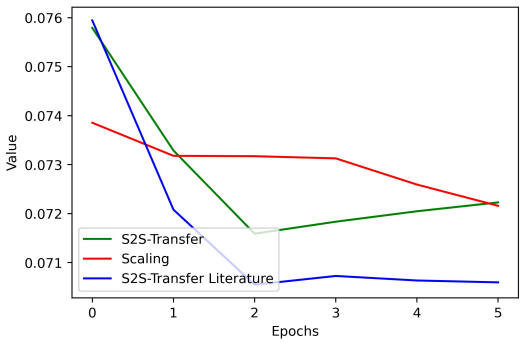
(i) MAPE Comparison for zone 11



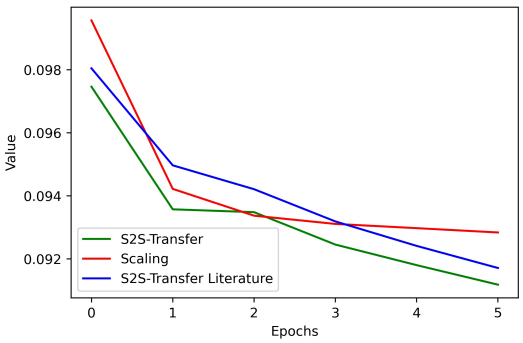
(j) MAPE Comparison for zone 12



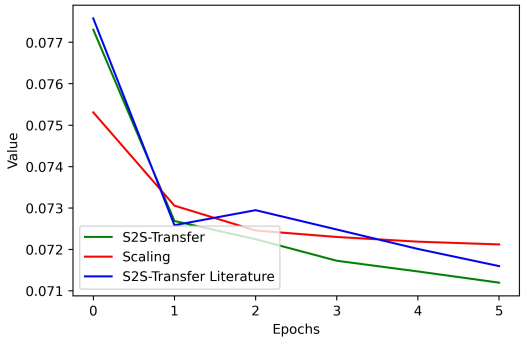
(k) MAPE Comparison for zone 13



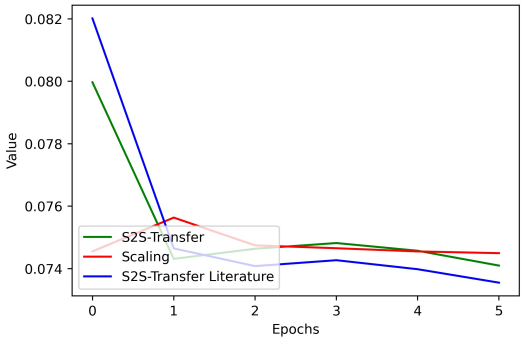
(l) MAPE Comparison for zone 14



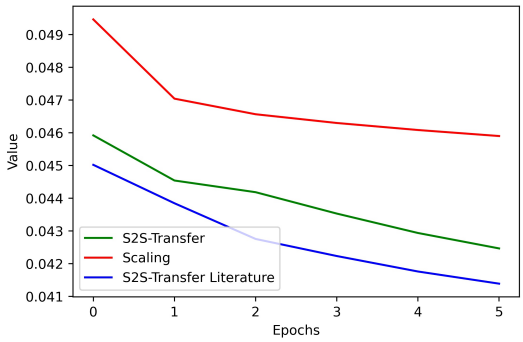
(m) MAPE Comparisonfor zone 15



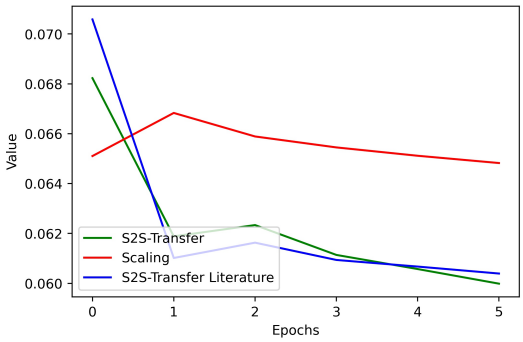
(n) MAPE Comparison for zone 16



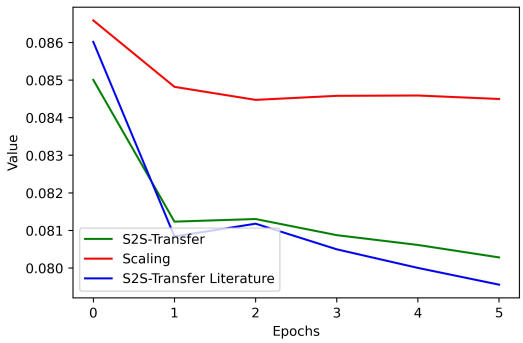
(o) MAPE Comparison for zone 17



(p) MAPE Comparison for zone 18



(q) MAPE Comparison for zone 19



(r) MAPE Comparison for zone 20

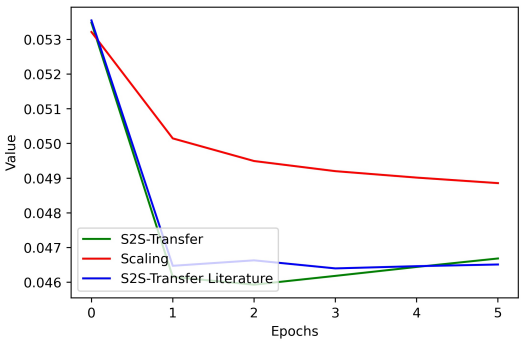
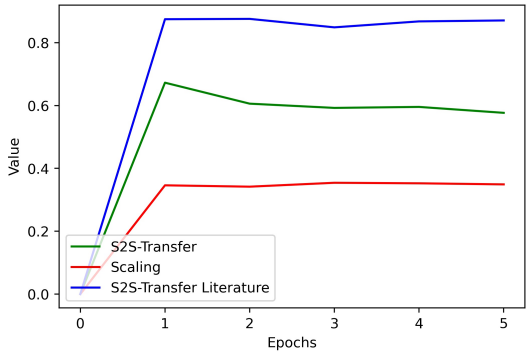
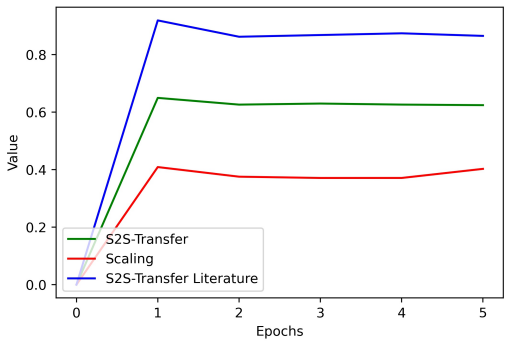


Figure 5.3: Comparison of MAPE accross the 3 tested models for each zone

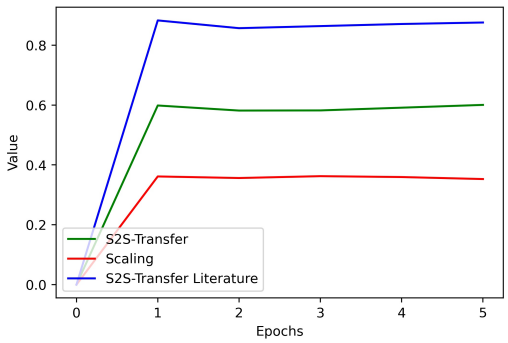
(a) Time to Predict Comparison for zone 1



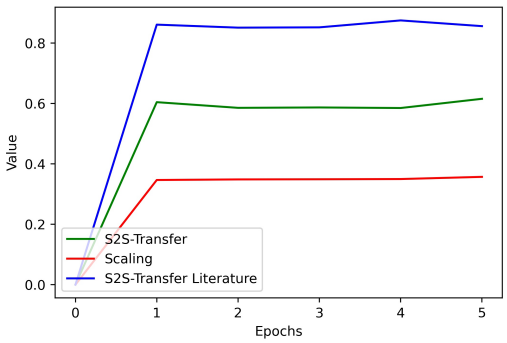
(b) Time to Predict Comparison for zone 2



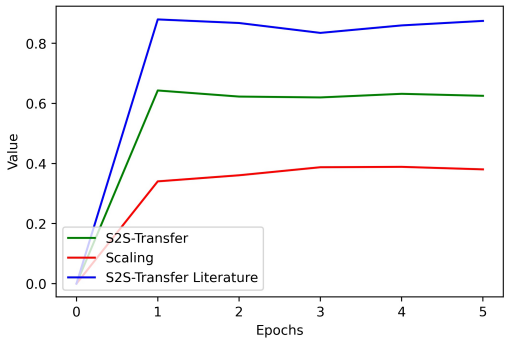
(c) Time to Predict Comparison for zone 4



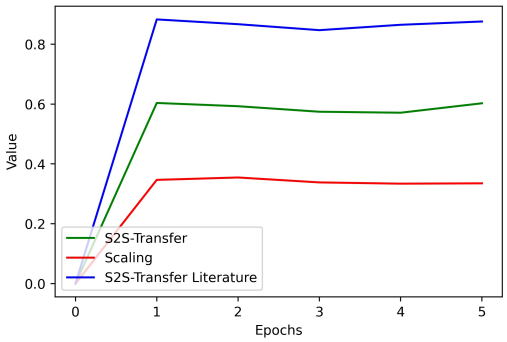
(d) Time to Predict Comparison for zone 5



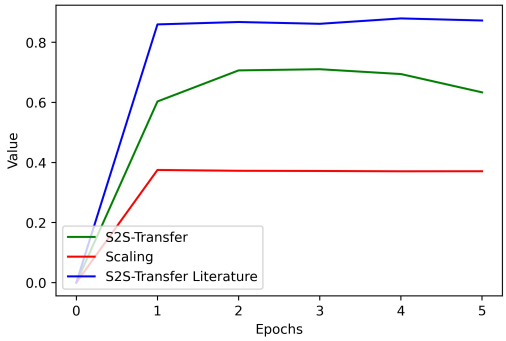
(e) Time to Predict Comparison for zone 6



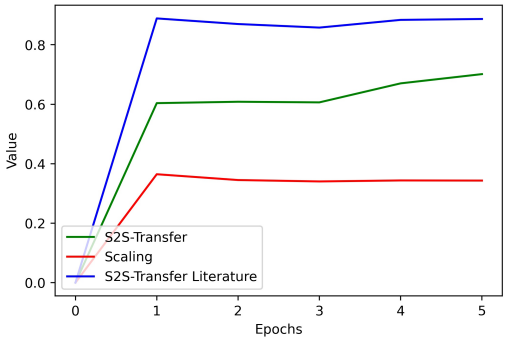
(f) Time to Predict Comparison for zone 8



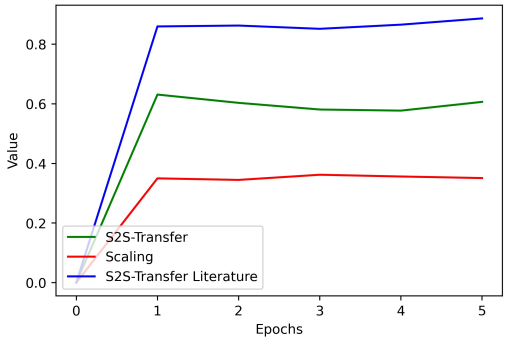
(g) Time to Predict Comparison for zone 9



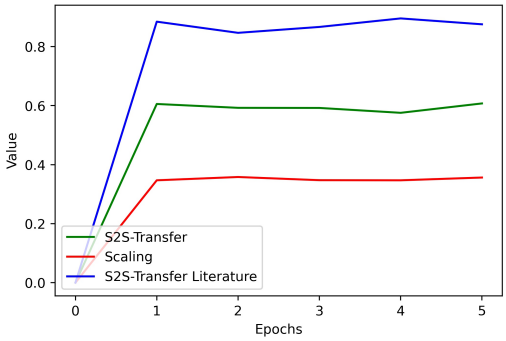
(h) Time to Predict Comparison for zone 10



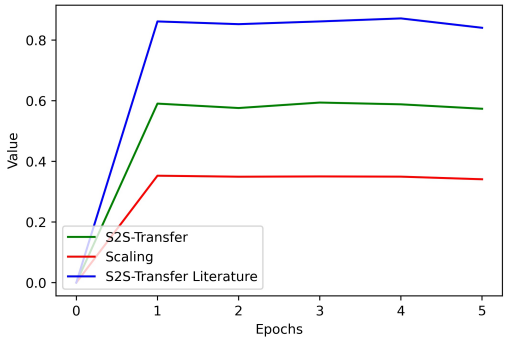
(i) Time to Predict Comparison for zone 11



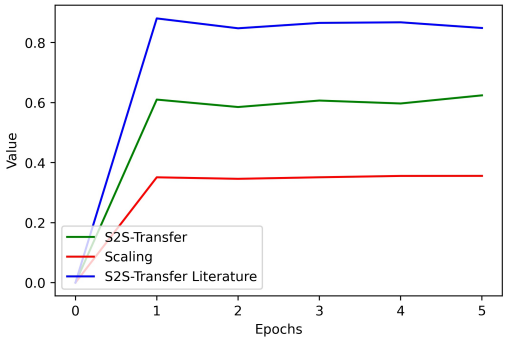
(j) Time to Predict Comparison for zone 12



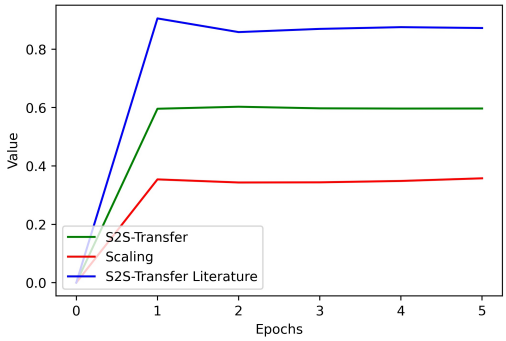
(k) Time to Predict Comparison for zone 13



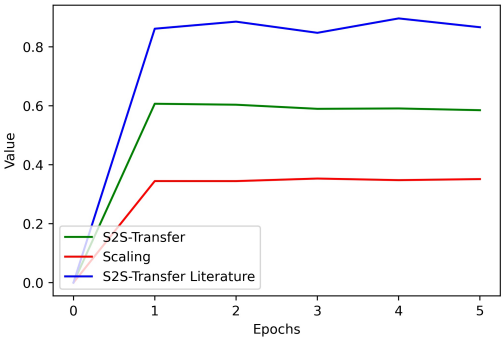
(l) Time to Predict Comparison for zone 14



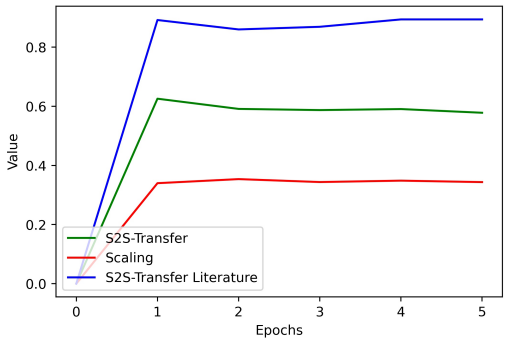
(m) Time to Predict Comparison for zone 15



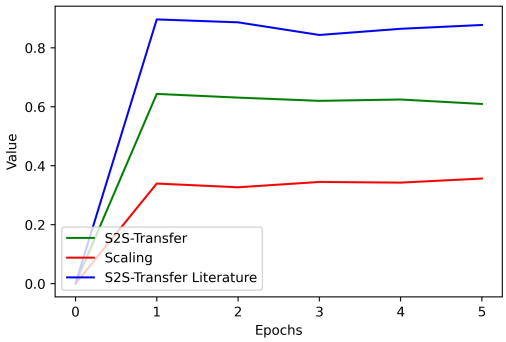
(n) Time to Predict Comparison for zone 16



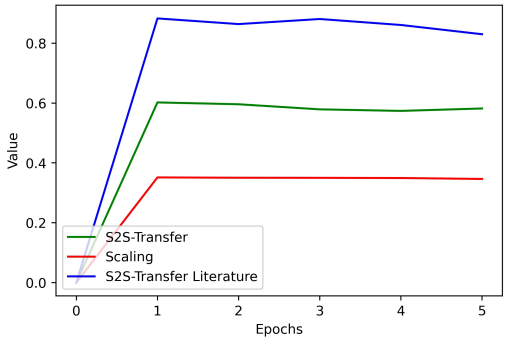
(o) Time to Predict Comparison for zone 17



(p) Time to Predict Comparison for zone 18



(q) Time to Predict Comparison for zone 19



(r) Time to Predict Comparison for zone 20

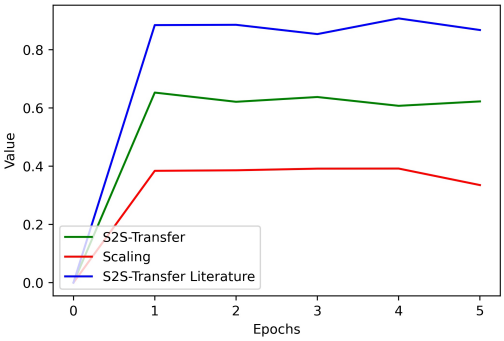


Figure 5.4: Comparison of Time to Train across the 3 tested models for each zone

5.4.2.2 Experiment 2

This experiment compares the MAPE accuracy between the S2S model with scaling and the S2S model with parametric transfer learning (which will henceforth be referred to as the non-scaling method). The results are no longer directly compared to the literature since we established in the previous experiment that our parametric transfer learning implementation of Tian et al. [175] performed very similarly to their published results. Additionally, the performance of our novel solution is compared to the data distribution of each zone.

Zone	Non-Scaling MAPE	Scaling MAPE	MAPE Diff.	First Distribution	Second Distribution
4	0.4316	0.3947	-0.0370	beta	triang
13	0.0722	0.0722	-0.0001	beta	weibull_min
16	0.0741	0.0745	0.0004	beta	gamma
15	0.0712	0.0721	0.0009	weibull_min	beta
14	0.0912	0.0928	0.0017	gamma	pearson3
20	0.0467	0.0489	0.0022	beta	gamma
2	0.0357	0.0383	0.0026	weibull_min	beta
6	0.0369	0.0395	0.0026	weibull_min	beta
17	0.0425	0.0459	0.0034	gamma	pearson3
19	0.0803	0.0845	0.0042	weibull_min	beta
18	0.0600	0.0648	0.0048	weibull_min	gamma
8	0.0487	0.0591	0.0104	beta	triang
9	0.2247	0.2355	0.0108	pearson3	weibull_min
12	0.0483	0.0594	0.0111	gamma	pearson3
1	0.0512	0.0625	0.0113	gamma	pearson3
11	0.0387	0.0514	0.0128	gamma	pearson3
5	0.0631	0.0770	0.0140	gamma	pearson3
10	0.0393	0.0545	0.0152	invgauss	lognorm

Table 5.3: Top 2 distribution for each zone, ordered by the difference in accuracy between scaling and non-scaling

Table 5.3 presents the different zones ordered by how much better in terms of MAPE accuracy the scaling technique performed over the non-scaling approach along with the zone's data distribution. It can be observed that the ordered results appear to be organized mostly in a manner that follows those distributions. The zones can be grouped into three main levels: the

first level where the scaling technique performs its best is dominated by the beta distribution, the second level, where the accuracy is very similar (zones 14, 20, 2, 6, 17, 19, 18, 8 and 9) where the weibull_min distribution predominant and lastly the third level where the scaling technique is least effective, the gamma distribution prevails. However, it is important to note that in all of these scenarios, the difference in accuracy where the non-scaling outperforms the scaling technique does not exceed 1.5%. These results enable us to say that the distribution has some effect on accuracy and suggests the importance to evaluate results in future research on data that contains various distributions in order to validate results replication and portability.

5.4.2.3 Experiment 3

After the efficacy of our proposed solution was shown, the impact of the size of the input sequence and that of the forecasting horizon was investigated, along with the size of the hidden context vector. For each forecasting horizon, an input sequence of 4, 8, 12 and 24h was used along with a hidden size of 64, 128 and 256 layers. Four forecasting horizons were tested: 4h, 8h, 12h and 24h, for a total of 36 experiments per model, per stream or 1,440 experiments. The results shown display the average over all 20 streams.

Table 5.4 and 5.5 both present the accuracy and training time for all experiments as an average for all zones of the specified parameters.

Window input	**Horizon**	**Hidden Size**	**Avg. MAPE**	**Avg. Time**	**Window Input**	**Horizon**	**Hidden Size**	**Avg. MAPE**	**Avg. Time**
4	8	64	12.57%	0.2314	12	4	128	8.23%	0.1779
4	8	128	11.60%	0.2250	12	4	256	7.90%	0.2031
4	8	256	10.52%	0.2685	12	8	64	11.58%	0.2790
4	12	64	13.86%	0.3160	12	8	128	10.61%	0.2809
4	12	128	12.39%	0.3185	12	8	256	10.15%	0.3136
4	12	256	12.23%	0.3590	12	24	64	13.94%	0.6258
4	24	64	14.19%	0.5346	12	24	128	13.23%	0.5854
4	24	128	13.55%	0.5801	12	24	256	12.78%	0.6521
4	24	256	13.06%	0.6173	24	4	64	8.48%	0.2194
8	4	64	9.01%	0.1565	24	4	128	7.87%	0.2234
8	4	128	8.42%	0.1742	24	4	256	7.78%	0.2288
8	4	256	8.33%	0.1870	24	8	64	12.48%	0.3168
8	12	64	13.40%	0.3209	24	8	128	10.22%	0.3080
8	12	128	12.39%	0.3383	24	8	256	9.63%	0.3620
8	12	256	11.65%	0.4021	24	12	64	11.98%	0.4148
8	24	64	14.06%	0.5808	24	12	128	10.90%	0.3811
8	24	128	13.66%	0.5999	24	12	256	10.37%	0.4176
8	24	256	13.09%	0.6401	24	36	64	14.81%	0.8531
12	4	64	8.56%	0.1786	24	36	128	14.20%	0.8674
...	...	...	...	...	24	36	256	13.53%	0.9021

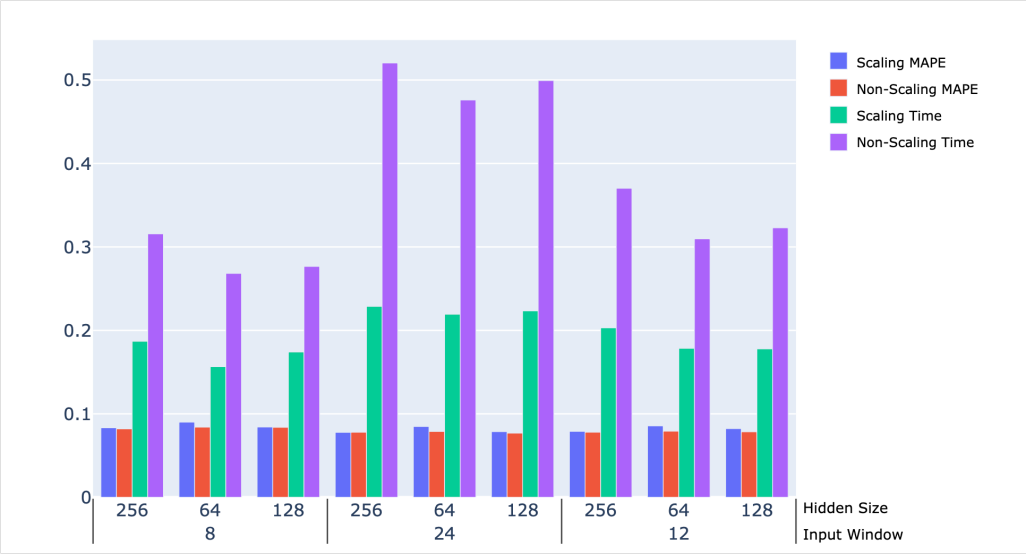
Table 5.4: Scaling model average MAPE accuracy and time for all zones under all parameter variance

Window input	**Horizon**	**Hidden Size**	**Avg. MAPE**	**Avg. Time**	**Window Input**	**Horizon**	**Hidden Size**	**Avg. MAPE**	**Avg. Time**
4	8	64	11.79%	0.3043	12	4	128	7.85%	0.3229
4	8	128	11.01%	0.3096	12	4	256	7.80%	0.3703
4	8	256	10.76%	0.3552	12	8	64	10.42%	0.3797
4	12	64	12.89%	0.3581	12	8	128	10.02%	0.3962
4	12	128	12.49%	0.3802	12	8	256	9.83%	0.4696
4	12	256	12.34%	0.4383	12	24	64	13.15%	0.6949
4	24	64	13.93%	0.6086	12	24	128	12.52%	0.7163
4	24	128	13.35%	0.6228	12	24	256	13.17%	0.7600
4	24	256	13.68%	0.6371	24	4	64	7.89%	0.4761
8	4	64	8.41%	0.2683	24	4	128	7.69%	0.4993
8	4	128	8.39%	0.2767	24	4	256	7.80%	0.5207
8	4	256	8.20%	0.3157	24	8	64	10.11%	0.5463
8	12	64	12.41%	0.4496	24	8	128	9.89%	0.5761
8	12	128	12.64%	0.4423	24	8	256	9.77%	0.6077
8	12	256	12.28%	0.4935	24	12	64	10.97%	0.6796
8	24	64	13.60%	0.6762	24	12	128	10.67%	0.6345
8	24	128	13.72%	0.6547	24	12	256	10.47%	0.6861
8	24	256	14.20%	0.7198	24	36	64	14.19%	1.0073
12	4	64	7.92%	0.3097	24	36	128	14.48%	0.9861
...	...	...	...	...	24	36	256	14.65%	1.1299

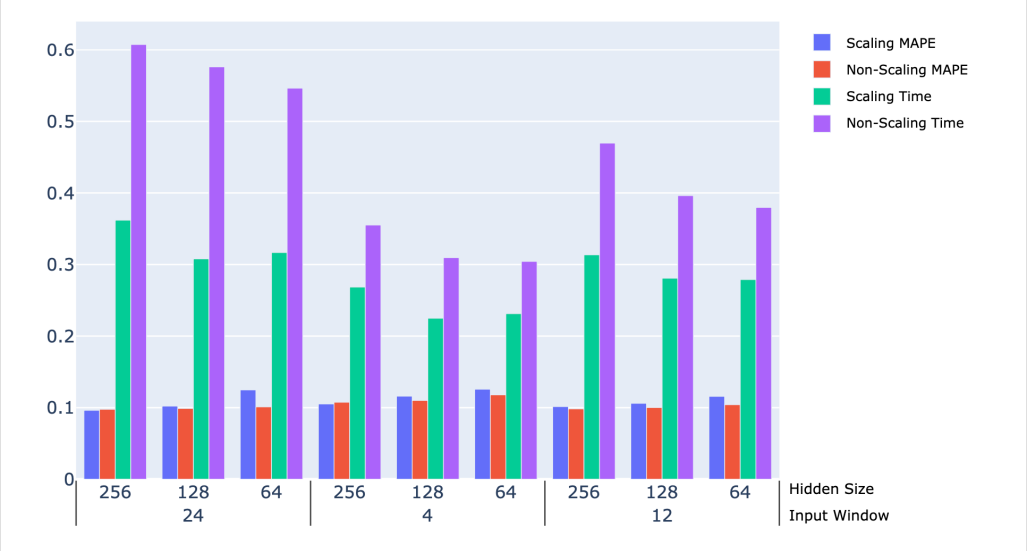
Table 5.5: Non-Scaling model average MAPE accuracy and time for all zones under all parameter variance

The results are also presented graphically in Figure 5.5. This visual enables us to view the result for each forecasting horizon and comprehend how the input window and hidden size affect the accuracy and processing time. The purple and turquoise bars in the graphs are representing the processing time whereas the blue and red bars represent the accuracy.

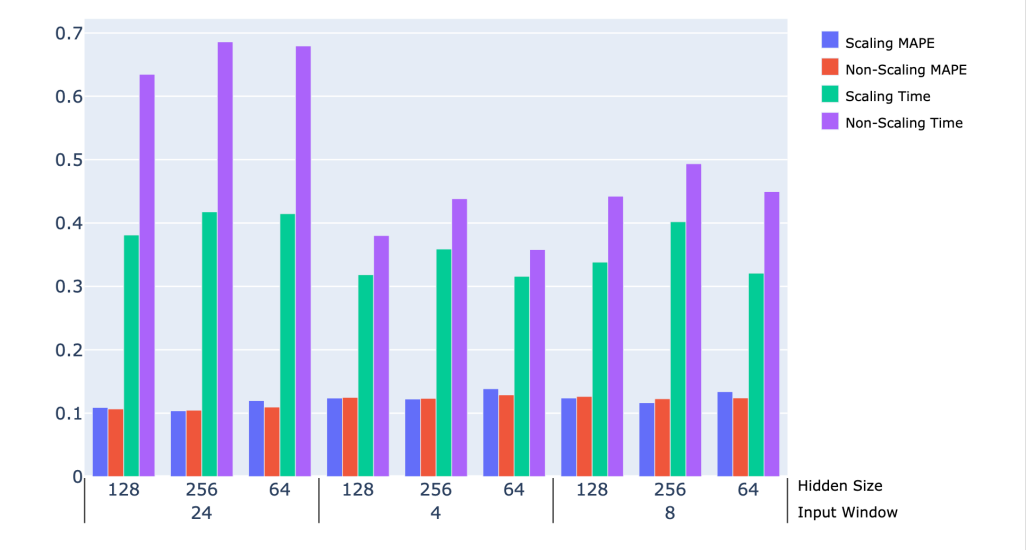
(a) 4h Forecasting Horizon Comparison



(b) 8h Forecasting Horizon Comparison



(c) 12h Forecasting Horizon Comparison



(d) 24h Forecasting Horizon Comparison

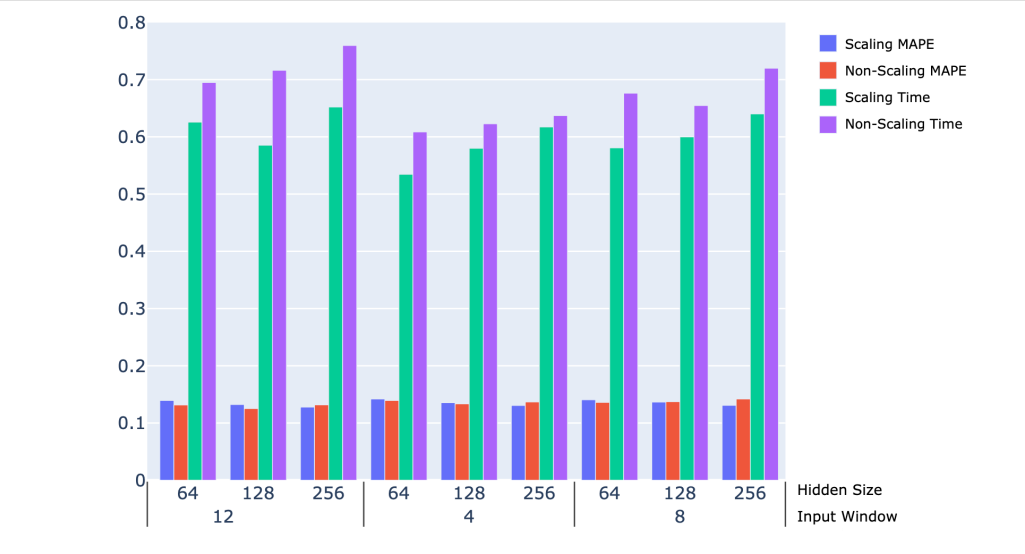


Figure 5.5: Impact of Input Size on Accuracy across the different Forecasting Horizon

Those results show that there are no significant changes in accuracy between both methods but that the scaling method benefits from a larger hidden size. This is in line with the design of our solution, the hidden size serves as the memory of the learning representation, the larger the memory the better the accuracy.

It can also be seen that although the scaling method is always faster, the larger the input window, the larger the difference in time performance. The time to train is not affected as significantly in the scaling method as it is with non-scaling. Additionally, using smaller amounts of data to predict longer horizons does not result in changes in processing time that are as markedly different as shown in 5.5d . However, it is clear that the scaling method enhances the time to train efficiency without significant loss of accuracy.

5.4.3 Conclusion

In conclusion, the objective of this chapter was manifold. Firstly, we aimed to address the performance challenges associated with transfer learning by reducing the processing time of the algorithm. The work shown in this chapter demonstrated a method used to address the performance challenge linked to transfer learning by successfully scaling a transfer learning method for load forecasting and reducing training time by approximately 40%.

Secondly, we aimed to reduce the training time of S2S models by proposing a modified transfer learning algorithm that would not involve deep design modifications in order to enable manageable changes to already deployed applications. The proposed approach is capable of

faster deployment of new zones within existing application infrastructure as the methodology does not require any structural or hardware-related modification. Existing models based upon the S2S algorithm could easily be adapted and benefit from these improvements.

Lastly, we strived to demonstrate the need for a novel evaluation strategy used to ensure result portability. The distribution analysis of our dataset demonstrated the importance of probability distribution variance for the performance of load forecasting. Having varied data streams is shown to be critical to ensure that the reported results can be adequately replicated.

Chapter 6

Conclusion

Over the last decade, the technological world has been undergoing a number of technological revolutions, for example, the amount of devices connected to the internet has more than doubled, social media accounts have tripled [214] and the change from 3G to 5G has increased mobile network speeds more than ten folds [215]. In other words, our access to technological connections has significantly increased: we have more connected devices, we have better access to the networks and perhaps more importantly we have more ways and opportunities to exchange information. These combined factors have led to a shift in users' approaches to technology. End users want not only faster access to information, but also seek better insights, in other words, consumers have higher expectations. In order to fulfill those expectations and better serve both companies and end-users, data are being collected, stored and analyzed to enable better decision making and knowledge acquisition: we are truly undergoing the Big Data revolution.

Data are being collected, exchanged and stored in every aspect of our lives: health care, transportation, finances, social media, education and utilities. Now that we have the ability to gain access to significant amounts of data through sensors and devices in real-time, we can offer better insights and knowledge in many spheres of our lives. This shift is made possible mainly due to data analytics and artificial intelligence, both of which are centred around a key concept: machine learning.

Therefore, identifying, categorizing and providing direction in regards to the challenges faced by machine learning with Big Data is of significant importance. This thesis has provided such a foundation. Additionally, the field of energy generation is facing a number of challenges of its own given the ongoing climate crisis. Having the ability to make better decisions in regard to the environment and our energy consumption is critical to reducing carbon emissions. In the United States, in 2014, more than 30% of greenhouse emissions came from electricity generation [216]; however, technological changes have enabled us to decrease these numbers

since [217]. More accurate and efficient means of performing electrical load forecasting contribute to such a decrease. The work presented in this thesis not only provides means for more accurate forecasting but also for faster and more efficient deployment of such solutions.

The remainder of this chapter will briefly summarize the contributions within this thesis and presents some future research directions.

6.1 Contributions

This thesis has presented the following contributions toward solving the identified challenges:

- Challenges of Machine Learning with Big Data
  - The challenges of Machine Learning with Big Data were identified and categorized according to the dimensions of Big Data responsible for the issues. By providing such organization, the cause and effect relationship between the data and its challenges was highlighted.
  - By highlighting the causality between data dimensions and their resulting challenges, we provided techniques for the industry to make better-informed design decisions when dealing with known data characteristics. Researchers and developers alike will be able to better anticipate potential challenges based on the expected characteristics of their data.
  - By identifying how each of the various machine learning paradigms was able to handle each of the challenges, we provided resources to assist in the design decisions of complex machine learning applications. Data scientists and engineers can more easily identify which machine learning paradigms to consider based on their data and desired tasks.
  - Created a foundation work for a better understanding of machine learning with Big data.
  - Provided the research community with potential research directions for future work and offered groundwork for potential improvements to the field of machine learning with Big Data by presenting and discussing how each paradigm successfully or unsuccessfully tackled each ML challenge.
- Accurate and Efficient Electrical Load Forecasting
  - By adapting the transformer architecture, accuracy improvements were made for load forecasting tasks.

  - By implementing learning scaling to load forecasting tasks, we supplied a way to more efficiently and quickly deploy models for load forecasting with new data.
  - By adapting and improving upon existing architectures, we provided a blueprint for performance improvements of already deployed solutions. Therefore, addressing challenges without significantly impacting the cost associated with solving performance issues.
  - By ensuring the statistical differences within the data test set, we suggested a novel approach to load forecasting result portability and repeatability.
  - A better measure of significance was suggested for load forecasting in order to address the disconnect between statistical and real-world impact.

- Performance Challenges Associated with Deep Learning

  - By adapting the transformer architecture for load forecasting we provided means to reduce the performance impediment linked to the volume of Big Data typically faced by the Deep learning paradigm. This is achieved due to our solutions' ability to be parallelized.
  - Successfully adapted the transformer architecture from one data type to another to further improve deep learning's ability to handle a variety of data.

- Performance Challenges Associated with Transfer Learning

  - By performing algorithmic and workflow changes to our Sequence-to-Sequence model we were able to address the performance challenges typically associated with transfer learning as identified in Chapter 3.

  Lastly, although the work tackled the task of load forecasting, the obtained results could be repeated with any tasks dealing with time series.

6.2 Future Work

This thesis explored some of the challenges of ML with Big Data for electrical load forecasting and although the contributions of this research are significant, many challenges remain to be investigated. The following subsection will discuss some of the future directions of this work.

6.2.1 Research Opportunities for Machine Learning Challenges for Big Data

The future work related to Machine Learning with Big Data is presented in this subsection. Although this topic of this thesis could be extended in many different ways, we chose to focus on topics related to software engineering as follow:

- The work presented in Chapter 3 identified and categorized the challenges of ML with Big Data in order to highlight how the components of Big Data affected the execution of ML. A future research opportunity will be to create concise software engineering guidelines based on the expected Big Data challenges. Additionally, the knowledge of which ML paradigm can address each challenge will be used to create a number of requirements and design guidelines in order to provide design patterns for common use cases. The field of machine learning lacks a strong software engineering design component and future work will attempt to formalize and bridge this gap.

- Another possible expansion to the work in Chapter 3 is in regards to the Big Data characteristics and corresponding metrics since there is currently no formal means of quantifying Big Data. A future direction of this work will be to identify and define metrics for each of the Big Data characteristics, without creating boundaries within these metrics. The goal will be to use them as a means to compare amongst Big Data and identify how each metric may be able to indicate the likelihood that a specific challenge will be encountered. However, the metrics would not be used to restrict the definition of Big Data. Having a means to measure and assess Big Data to identify the likelihood that a specific challenge may be encountered would enable better software design.

- Lastly, another opportunity for future work will be to investigate which machine learning paradigm best performs on various types of tasks, as opposed to how we identified how each paradigm can address each challenge. This relationship between tasks and paradigms could also be used for better machine learning application design.

6.2.2 Load Forecasting

This subsection will detail the three main possible directions of future work related to load forecasting:

- Impact of Data Similarity

  - The work presented in Chapter 5 has shown that knowledge can be effectively transferred from one data stream to another, regardless of the variation in data distributions. However, the root cause of the transferability have not been explored. A possible next step in our research will be to identify and formalize a means of comparing data stream similarities and determine how these similarities impact our ability to transfer knowledge. Having a better understanding of the versatility of the data could also allow for better transfer learning design.
  - Another future direction will be to develop a framework used to predict the similarity of data streams using known non-energy consumption-related parameters of future data. Having the ability to identify similarities before a large amount of data has been collected would enable faster and more accurate deployment of models for new users.

- Transformer Exploration

  - The work introduced in Chapter 4 provided means to improve performance and accuracy over fixed short-term horizons using a new adapted architecture. Future work direction is to explore the ability of the transformer to predict at much longer horizons.
  - Pre-trained transformers are often used in NLP. A new research opportunity could explore pre-training a transformer on large amounts of data, from various sources, and see how effective those pre-trained transformers may be with unseen data.

- Load Forecasting and Other Challenges

  - Chapters 4 and 5 addressed the Big Data challenge of performance associated with load forecasting and how to improve model accuracy and efficiency. Future work could investigate the ability of models to withstand other challenges identified in Chapter 3 such as concept drift.
  - Lastly, the work presented in this thesis mainly investigated challenges related to the volume and velocity dimensions of Big Data. A future research direction will be to study how data variety can affect load forecasting tasks. This could be achieved by integrating various data sources and types such as occupancy, traffic or financial data in order to identify patterns of consumption.

This thesis made contributions towards ML with Big Data; however, there are many further developments and solutions needed to continue to extract value from Big Data and address the ever changing nature of this field.

Bibliography

[1] Thomas H. Davenport and Randy Bean, "Big Data and AI Executive Survey 2019," NewVantage Partners, Boston, Tech. Rep., 2019. [Online]. Available: www.newvantage.com

[2] "Machine learning challenges 2021," Statista. [Online]. Available: https://www.statista.com/statistics/1111249/machine-learning-challenges/

[3] "Global IoT and non-IoT connections 2010-2025," Statista. [Online]. Available: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/

[4] "Big Data Analytics Market in the Energy Sector - Growth, Trends, COVID-19 Impact, and Forecasts (2022 - 2027)," Mordor Intelligence, Tech. Rep., 2022. [Online]. Available: https://www.mordorintelligence.com/industry-reports/big-data-in-energy-sector-industry

[5] I. G. Y. Bengio and A. Courville, "Deep learning," Tech. Rep., 2016, book in preparation for MIT Press.

[6] "IoT Growth Demands Rethink of Long-Term Storage Strategies, says IDC," IoT Business News. [Online]. Available: https://iotbusinessnews.com/2020/07/29/20898-iot-growth-demands-rethink-of-long-term-storage-strategies-says-idc/

[7] W. Raghupathi and V. Raghupathi, "Big data analytics in healthcare: promise and potential," *Health Information Science and Systems*, vol. 2, no. 1, pp. 1–10, 2014.

[8] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, "Efficient machine learning for big data: A review," *Big Data Research*, vol. 2, no. 3, pp. 87–93, 4 2015.

[9] T. H. Davenport and R. Bean, "Big Data and AI Executive Survey 2020: Executive Summary of Findings," NewVantage, Tech. Rep., 2020. [Online]. Available: www.newvantage.com

[10] D. Drai, "Why AI-Driven Analytics Is Essential For Data-Driven Decision-Making," Forbes, 2021. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2021/12/27/why-ai-driven-analytics-is-essential-for-data-driven-decision-making/?sh=565e43ab73f4

[11] F. Halper, "Best Practices Report — Driving Digital Transformation Using AI and Machine Learning — Transforming Data with Intelligence," TDWI, Tech. Rep., 9 2019. [Online]. Available: https://tdwi.org/research/2019/09/adv-all-best-practices-report-driving-digital-transformation-using-ai-and-machine-learning.aspx?tc=page0&tc=assetpg&tc=page0&tc=assetpg

[12] I. Wladawsky-Berger, *The Current State of AI Adoption*, New York, 2 2019. [Online]. Available: https://www.wsj.com/articles/the-current-state-of-ai-adoption-01549644400

[13] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, 5 2017.

[14] V. H. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi, "Big data and its technical challenges," *Communications of the ACM*, vol. 57, no. 7, pp. 86–94, 2014.

[15] K. Grolinger, M. Hayes, W. A. Higashino, A. L'Heureux, D. S. Allison, and M. A. M. Capretz, "Challenges for mapreduce in big data," 6 2014, pp. 182–189.

[16] A. Ghasempour, "Internet of things in smart grid: Architecture, applications, services, key technologies, and challenges," 2019.

[17] Q. Yang, "Internet of things application in smart grid: A brief overview of challenges, opportunities, and future trends," *Smart Power Distribution Systems: Control, Communication, and Optimization*, pp. 267–283, 1 2019.

[18] "IoT and analytics global revenue by segment 2021," Statista. [Online]. Available: https://www.statista.com/statistics/913299/projected-global-revenue-of-the-internet-of-things-segment/

[19] "How Does Forecasting Enhance Smart Grid Benefits?" SAS, Tech. Rep., 2010.

[20] B. Farsi, M. Amayri, N. Bouguila, and U. Eicker, "On short-term load forecasting using machine learning techniques and a novel parallel deep LSTM-CNN approach," *IEEE Access*, vol. 9, pp. 31 191–31 212, 2021.

[21] Y. Cui, B. Yin, R. Li, Z. Du, and M. Ding, "Short-time Series Load Forecasting by Seq2seq-LSTM Model," pp. 517–521, 2020.

[22] F. X. Diebold, ""Big Data" Dynamic Factor Models for Macroeconomic Measurement and Forecasting: A Discussion of the Papers by Lucrezia Reichlin and by Mark W. Watson," in *Advances in Economics and Econometrics*, M. Dewatripont, L. P. Hansen, and S. J. Turnovsky, Eds. Cambridge: Cambridge University Press, 2003, pp. 115–122. [Online]. Available: https://www.cambridge.org/core/product/identifier/CBO9780511610264A019/type/book_part

[23] Markets and Markets, "Big Data Market with COVID-19 Impact Analysis, by Component, Deployment Mode, Organization Size, Business Function (Finance, Marketing & Sales), Industry Vertical (BFSI, Manufacturing, Healthcare & Life Sciences) and Region - Global Forecast to 2026," Tech. Rep., 2 2022. [Online]. Available: https://www.businesswire.com/news/home/20220216005953/en/The-Worldwide-Big-Data-Industry-is-Expected-to-Reach-273.4-Billion-by-2026---ResearchAndMarkets.com

[24] M. A. Beyer and D. Laney, "The importance of 'big data': a definition," *Gartner Research Report*, 2012.

[25] N. Japkowicz and J. Stefanowski, "A Machine Learning Perspective on Big Data Analysis," pp. 1–31, 2016. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-26989-4_1

[26] J.-P. Dijcks, "Oracle: Big Data for the Enterprise," 2011.

[27] M. Schroeck, R. Shockley, J. Smart, D. Romero-Morales, and P. Tufano, "Analytics: the real-world use of big data," *IBM Global Business Services Saïd Business School at the University of Oxford*, pp. 1–20, 2012.

[28] F. J. Ohlhorst, *Big Data Analytics: Turning Big Data into Big Money*. John Wiley & Sons, 2012, vol. 15.

[29] A. Gandomi and M. Haider, "Beyond the Hype: Big data Concepts, Methods, and Analytics," *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144, 4 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0268401214001066

[30] I. Chebbi, W. Boulila, and I. R. Farah, "Big data: Concepts, challenges and applications," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9330 LNCS, 2015.

[31] U. Department of Energy, "Chapter 3: Enabling Modernization of the Electric Power System Technology Assessments Cyber and Physical Security Designs, Architectures, and Concepts Electric Energy Storage Flexible and Distributed Energy Resources Measurements, Communications, and Controls Transmission and Distribution Components," 2015.

[32] "U.S. smart grid to cost billions, save trillions," Reuters. [Online]. Available: https://www.reuters.com/article/us-utilities-smartgrid-epri-idUSTRE74N7O420110524

[33] S. Rusitschka and E. Curry, "Big data in the energy and transport sectors," *New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe*, pp. 225–244, 1 2016. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-21569-3_13

[34] M. N. Fekri, K. Grolinger, and S. Mir, "Distributed load forecasting using smart meter data: Federated learning with Recurrent Neural Networks," *International Journal of Electrical Power & Energy Systems*, vol. 137, p. 107669, 5 2022.

[35] "What are Recurrent Neural Networks?" IBM. [Online]. Available: https://www.ibm.com/cloud/learn/recurrent-neural-networks

[36] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, vol. 4, no. January, 2014.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 2017-December, 2017.

[38] R. Krikorian, "Twitter by the numbers," 2010, [Online; accessed 2016-06-20]. [Online]. Available: http://www.slideshare.net/raffikrikorian/twitter-by-the-numbers?ref=http://techcrunch.com/2010/09/17/twitter-seeing-6-billion-api-calls-per-day-70k-per-second/

[39] ABI, "Billion devices will wirelessly connect to the internet of everything in 2020," 2013, [Online; accessed 2016-06-20]. [Online]. Available: https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne/

[40] V. Mayer-Schönberger and K. Cukier, *Big Data: A Revolution that Will Transform how We Live, Work, and Think*. Houghton Mifflin Harcourt, 2013.

[41] M. James, C. Michael, B. Brad, and B. Jacques, "Big data: The next frontier for innovation, competition, and productivity," *The McKinsey Global Institute*, 2011.

[42] M. Rouse, "Machine learning definition," 2011, [Online; accessed 2016-06-20]. [Online]. Available: http://whatis.techtarget.com/definition/machine-learning

[43] ——, "Predictive analytics definition," 2009. [Online]. Available: http://searchcrm.techtarget.com/definition/predictive-analytics

[44] J. Dean and S. Ghemawat, "Mapreduce: Simplied data processing on large clusters," 2004, pp. 137–149.

[45] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," 2010, pp. 1–10.

[46] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, p. 1, 2 2015.

[47] C. Parker, "Unexpected challenges in large scale machine learning." New York, New York, USA: ACM Press, 8 2012, pp. 1–6.

[48] S. R. Sukumar, "Machine learning in the big data era: Are we there yet?" 2014.

[49] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 67, pp. 1–16, 2016.

[50] X. wen Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014.

[51] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144, 4 2015.

[52] D. Singh and C. K. Reddy, "A survey on platforms for big data analytics," *Journal of big data*, vol. 2, no. 1, pp. 1–20, 2015.

[53] d. P. D. C. Almeida and J. Bernardino, "Big data open source platforms," 2015, pp. 268–275.

[54] W. Fan and A. Bifet, "Mining big data: current status, and forecast to the future," *ACM sIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 1–5, 2013.

[55] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97–107, 2014.

[56] R. Narasimhan and T. Bhuvaneshwari, "Big data a brief study," *International Journal of Scientific & Engineering Research*, vol. 5, no. 9, pp. 350–353, 2014.

[57] F. J. Ohlhorst, *Big Data Analytics: Turning Big Data into Big Money*. John Wiley & Sons, 2012, vol. 15.

[58] Y. Demchenko, P. Grosso, C. De Laat, and P. Membrey, "Addressing big data issues in scientific data infrastructure," 2013, pp. 48–55.

[59] M. A. u. d. Khan, M. F. Uddin, and N. Gupta, "Seven v's of big data understanding big data to extract value," 4 2014, pp. 1–5.

[60] R. Kune, P. K. Konugurthi, A. Agarwal, R. R. Chillarige, and R. Buyya, "The anatomy of big data computing," *Software: Practice and Experience*, vol. 46, no. 1, pp. 79–105, 2016.

[61] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast svm training on very large data sets," *Journal of Machine Learning Research*, vol. 6, pp. 363–392, 2005.

[62] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, "Map-reduce for machine learning on multicore," 2006, pp. 281–288.

[63] M. Zaharia, M. Chowdhury, T. Das, and A. Dave, "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," 2012, pp. 2–2.

[64] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, p. 10.

[65] K. A. Kumar, J. Gluck, A. Deshpande, and J. Lin, "Hone: "scaling down" hadoop on shared-memory systems," *Proceedings of the VLDB Endowment*, vol. 6, no. 12, pp. 1354–1357, 2013.

[66] M. Ghanavati, R. K. Wong, F. Chen, Y. Wang, and C.-S. Perng, "An effective integrated method for learning big imbalanced data." IEEE, 6 2014, pp. 691–698.

[67] N. Japkowicz and S. Stephen, "The class imbalance problem: a systematic study," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[68] A. K. Baughman, W. Chuang, K. R. Dixon, Z. Benz, and J. Basilico, "Deepqa jeopardy! gamification: a machine-learning perspective," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 1, pp. 55–66, 2014.

[69] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, p. 78, 2012.

[70] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Transactions on Information Theory*, vol. 14, no. 1, pp. 55–63, 1968.

[71] J. Fan, F. Han, and H. Liu, "Challenges of big data analysis," *National Science Review*, vol. 1, no. 2, pp. 293–314, 2014.

[72] B. Ratner, *Statistical and Machine-Learning Data Mining: Techniques for Better Predictive Modeling and Analysis of Big Data*. CRC Press, 2011.

[73] M. Y. Kiang, "A comparative assessment of classification methods," *Decision Support Systems*, vol. 35, no. 4, pp. 441–454, 2003.

[74] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*. Cambridge University Press, 2014, vol. 13.

[75] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.

[76] C. S. Calude and G. Longo, "The deluge of spurious correlations in big data," *Foundations of Science*, pp. 1–18, 2016.

[77] P. Domingos, "A unified bias-variance decomposition and its applications," 2000, pp. 231–238.

[78] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: Data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.

[79] K. Grolinger, W. a. Higashino, A. Tiwari, and M. A. Capretz, "Data management in cloud environments: Nosql and newsql data stores," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, p. 22, 2013.

[80] Y. Zheng, "Methodologies for cross-domain data fusion: an overview," *Big Data, IEEE Transactions on*, vol. 1, no. 1, pp. 16–34, 2015.

[81] M. Swan, "The quantified self: Fundamental disruption in big data science and biological discovery," *Big Data*, vol. 1, no. 2, pp. 85–99, 2013.

[82] X. Geng and K. Smith-Miles, "Incremental learning," in *Encyclopedia of Biometrics SE - 304*. Springer US, 2009, pp. 731–735.

[83] B. Gu, V. S. Sheng, Z. Wang, D. Ho, S. Osman, and S. Li, "Incremental learning for v-support vector regression." *Neural networks : the official journal of the International Neural Network Society*, vol. 67, pp. 140–50, 2015.

[84] N. Marz, "Apache storm," 2014, [Online; accessed 2016-06-20]. [Online]. Available: http://storm.apache.org

[85] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: Distributed stream computing platform," 2010, pp. 170–177.

[86] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[87] A. Tsymbal, "The problem of concept drift: Definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, 2004.

[88] K. Grolinger, A. L'Heureux, M. Capretz, and L. Seewald, "Energy forecasting for event venues: Big data and prediction accuracy," *Energy and Buildings*, vol. 112, pp. 222–233, 2016.

[89] P. B. Dongre and L. G. Malik, "A review on real time data stream classification and adapting to various concept drift scenarios," 2014, pp. 533–537.

[90] J. D. D. Lavaire, A. Singh, M. Yousef, S. Singh, and X. Yue, "Dimensional scalability of supervised and unsupervised concept drift detection: an empirical study," 10 2015, pp. 2212–2218.

[91] A. Clauset, "A brief primer on probability distributions," 2011.

[92] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[93] M. Dundar, B. Krishnapuram, J. Bi, and R. B. Rao, "Learning classifiers when the training data is not iid," 2007, pp. 756–761.

[94] J. Wang, D. Crawl, S. Purawat, M. Nguyen, and I. Altintas, "Big data provenance: Challenges, state of the art and opportunities," 10 2015, pp. 2509–2516.

[95] P. Buneman, S. Khanna, and W.-C. Tan, "Data provenance: Some basic issues," in *FST TCS 2000: Foundations of software technology and theoretical computer science*. Springer, 2000, pp. 87–93.

[96] H. Park, R. Ikeda, and J. Widom, "Ramp: A system for capturing and tracing provenance in mapreduce workflows." *Proceedings of the VLDB Endowment*, vol. 4, no. 12, pp. 1351–1354, 2011.

[97] N. Cao, L. Lu, Y.-R. Lin, F. Wang, and Z. Wen, "Socialhelix: Visual analysis of sentiment divergence in social media," *Journal of Visualization*, vol. 18, no. 2, pp. 221–235, 2015.

[98] R. Lovelace, M. Birkin, P. Cross, and M. Clarke, "From big noise to big data: Toward the verification of large data sets for understanding regional retail flows," *Geographical Analysis*, vol. 48, no. 1, pp. 59–81, 2016.

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

[100] R. Barga, V. Fontama, and W. H. Tok, "Cortana analytics," in *Predictive Analytics with Microsoft Azure Machine Learning*. Springer, 2015, pp. 279–283.

[101] Google, "Google cloud machine learning," 2016, [Online; accessed 2016-06-20]. [Online]. Available: https://cloud.google.com/products/machine-learning/

[102] A. W. Services, "Amazon machine learning," 2016, [Online; accessed 2016-06-20]. [Online]. Available: https://aws.amazon.com/machine-learning/

[103] IBM, "Ibm watson ecosystem program," 2014, [Online; accessed 2016-06-20]. [Online]. Available: http://www-03.ibm.com/innovation/us/watson/

[104] R. C. Team, *R: A Language and Environment for Statistical Computing*, Vienna, Austria, 2015, vol. 1.

[105] "Matlab," 2016.

[106] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The weka data mining software: an update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[107] A. Labrinidis and V. H. Jagadish, "Challenges and opportunities with big data," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 2032–2033, 2012.

[108] C. L. Philip Chen and C. Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: a survey on big data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[109] E. Bingham and H. Mannila, "Random projection in dimensionality reduction." New York, New York, USA: ACM Press, 8 2001, pp. 245–250.

[110] H. Liu and H. Motoda, *Instance Selection and Construction for Data Mining*. Springer Science & Business Media, 2013, vol. 608.

[111] A. Buades, B. Coll, and J. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 490–530, 2005.

[112] NVIDIA, "Gpu applications: Transforming computational research and engineering," 2016, [Online; accessed 2016-06-20]. [Online]. Available: http://www.nvidia.ca/object/machine-learning.html

[113] G. S. Jedhe, A. Ramamoorthy, and K. Varghese, "A scalable high throughput firewall in fpga," 2008, pp. 43–52.

[114] A. Gesmundo and N. Tomeh, "Hadoopperceptron: A toolkit for distributed perceptron training and prediction with mapreduce," ser. EACL '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 97–101.

[115] A. Ghoting, P. Kambadur, E. Pednault, and R. Kannan, "Nimble: A toolkit for the implementation of parallel data mining and machine learning algorithms on mapreduce," ser. KDD '11. New York, NY, USA: ACM, 2011, pp. 334–342.

[116] Y. Bu, B. Howe, and M. D. Ernst, "Haloop: Efficient iterative data processing on large clusters," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 285–296, 2010.

[117] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister," 2010, pp. 810–818.

[118] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing," 2010, pp. 135–146.

[119] Apache, "Apache giraph," 2016, [Online; accessed 2016-06-20]. [Online]. Available: http://giraph.apache.org

[120] M. Gorawski, A. Gorawska, and K. Pasterak, "A survey of data stream processing tools," in *Information Sciences and Systems 2014*. Springer, 2014, pp. 295–303.

[121] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," *ACM Computing Surveys (CSUR)*, vol. 44, no. 3, p. 15, 2012.

[122] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for svm," *Mathematical Programming*, vol. 127, no. 1, pp. 3–30, 10 2010.

[123] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent." *Journal of statistical software*, vol. 33, no. 1, pp. 1–22, 2010.

[124] Apache, "Apache mahout," 2016, [Online; accessed 2016-06-20]. [Online]. Available: http://mahout.apache.org

[125] 0xdata, "H20," 2016, [Online; accessed 2016-06-20]. [Online]. Available: http://www.h2o.ai

[126] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, "Petuum: A new platform for distributed machine learning on big data," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 49–67, 2015.

[127] J. Langford, L. Li, and A. Strehl, "Vowpal wabbit online learning project," Tech. Rep., 2007.

[128] G. D. F. Morales and A. Bifet, "Samoa: scalable advanced massive online analysis." *Journal of Machine Learning Research*, vol. 16, pp. 149–153, 2015.

[129] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[130] L. Deng and X. Li, "Machine learning paradigms for speech recognition: An overview," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 21, no. 5, pp. 1060–1089, 2013.

[131] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives." *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–828, 8 2013.

[132] V. Q. Le, "A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks," *Google Brain*, 2015.

[133] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," 2009, pp. 448–455.

[134] G. Hinton, "Deep belief nets," in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 267–269.

[135] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR*, vol. abs/1510.0, 2015.

[136] J. Read, F. Perez-Cruz, and A. Bifet, "Deep learning in partially-labeled data streams," ser. SAC '15. New York, NY, USA: ACM, 2015, pp. 954–959.

[137] B. Mirza, Z. Lin, and N. Liu, "Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift," *Neurocomputing*, vol. 149, pp. 316–329, 2 2015.

[138] K. Kanoun and M. van der Schaar, "Big-data streaming applications scheduling with online learning and concept drift detection," ser. DATE '15. San Jose, CA, USA: EDA Consortium, 2015, pp. 1547–1550.

[139] L. Bottou and V. Vapnik, "Local learning algorithms," *Neural Computation*, vol. 4, no. 6, pp. 888–900, 1992.

[140] K. Huang, H. Yang, I. King, and M. Lyu, "Local learning vs. global learning: an introduction to maxi-min margin machine," in *Support vector machines: theory and applications*. Springer Berlin Heidelberg, 2005, pp. 113–131.

[141] T.-N. Do and F. Poulet, "Random local svms for classifying large datasets," in *Future Data and Security Engineering SE - 1*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 9446, pp. 3–15.

[142] E. E. Elattar, J. Goulermas, and Q. H. Wu, "Electric load forecasting based on locally weighted support vector regression," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 4, pp. 438–447, 2010.

[143] K. Grolinger, M. A. M. Capretz, and L. Seewald, "Energy consumption prediction with big data : Balancing prediction accuracy and computational resources," San Francisco, California, 2016.

[144] L. Torrey and J. Shavlik, *Handbook of Research on Machine Learning Applications and Trends*, E. S. Olivas, J. D. M. Guerrero, M. Martinez-Sober, J. R. Magdalena-Benedito, and A. J. Serrano López, Eds. IGI Global, 1 2010.

[145] L. Yang, Y. Chu, J. Zhang, L. Xia, Z. Wang, and K. L. Tan, "Transfer learning over big data," 10 2015, pp. 63–68.

[146] S. Thrun and L. Pratt, *Learning to Learn*, S. Thrun and L. Pratt, Eds. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[147] D. L. Silver, Q. Yang, and L. Li, "Lifelong machine learning systems : Beyond learning algorithms," *AAAI Spring Symposium Series*, no. Solomonoff 1989, pp. 49–55, 2013.

[148] M. T. Khan, M. Durrani, S. Khalid, and F. Aziz, "Lifelong aspect extraction from big data: Knowledge engineering," *Complex Adaptive Systems Modeling*, vol. 4, no. 1, pp. 1–15, 2016.

[149] Z. Chen and B. Liu, "Topic modeling using topics from many domains, lifelong learning and big data," vol. 32, 2014, pp. 703–711.

[150] S. Suthaharan, "Big data classification: Problems and challenges in network intrusion prediction with machine learning," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 4, pp. 70–73, 2014.

[151] T. Dietterich, "Ensemble methods in machine learning," in *Multiple classifier systems*, 2000, vol. 1857, p. 1–15.

[152] M. Sewell, "Ensemble learning," *Research Note UCL Department of Computer Science*, p. 12, 2011.

[153] B. Waske and J. A. Benediktsson, "Fusion of support vector machines for classification of multisensor data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12, pp. 3858–3866, 2007.

[154] Y. Tang, Z. Xu, and Y. Zhuang, "Bayesian network structure learning from big data: A reservoir sampling based ensemble method," in *Database Systems for Advanced Applications: DASFAA 2016 International Workshops: BDMS, BDQM, MoI, and SeCoP, Dallas, TX, USA, April 16-19, 2016, Proceedings*. Cham: Springer International Publishing, 2016, pp. 209–222.

[155] W. Zang, P. Zhang, C. Zhou, and L. Guo, "Comparative study between incremental and ensemble learning on data streams: Case ssudy," *Journal Of Big Data*, vol. 1, no. 1, p. 5, 2014.

[156] R. Alabdulrahman, "A comparative study of ensemble active learning," Ph.D. dissertation, 2014.

[157] S. Gruber, R. W. Logan, I. Jarrín, S. Monge, and M. A. Hernán, "Ensemble learning of inverse probability weights for marginal structural modeling in large observational datasets," *Statistics in Medicine*, vol. 34, no. 1, pp. 106–117, 2015.

[158] D. Bachmann, "Contextual model-based collaborative filtering for recommender systems," 2017.

[159] Y. Wu, M. Schuster, Z. Chen, V. Q. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *ArXiv e-prints*, pp. 1–23, 2016.

[160] R. Saeedi, H. Ghasemzadeh, and A. H. Gebremedhin, "Transfer learning algorithms for autonomous reconfiguration of wearable systems," *2016 IEEE International Conference on Big Data (Big Data)*, pp. 563–569, 2016.

[161] L. Li, K. Ota, and M. Dong, "When Weather Matters: IoT-Based Electrical Load Forecasting for Smart Grid," *IEEE Communications Magazine*, vol. 55, no. 10, 2017.

[162] K. Srinivasan and R. Pronovost, "Short term load forecasting using multiple correlation models," *IEEE Transactions on Power Apparatus and Systems*, vol. 94, no. 5, pp. 1854–1858, 1975.

[163] J. Walther and M. Weigold, "A systematic review on predicting and forecasting the electrical energy consumption in the manufacturing industry," 2021.

[164] I. Energy Agency, "Net Zero by 2050 - A Roadmap for the Global Energy Sector," IEA Publications, Tech. Rep., 5 2021. [Online]. Available: www.iea.org/t&c/

[165] "Frequently Asked Questions (FAQs)," U.S. Energy Information Administration (EIA). [Online]. Available: https://www.eia.gov/tools/faqs/faq.php?id=74&t=11

[166] E. Lo Cascio, L. Girardin, Z. Ma, and F. Maréchal, "How Smart is the Grid?" [Online]. Available: www.frontiersin.org

[167] M. Shabanzadeh and M. P. Moghaddam, "What is the Smart Grid? Definitions, Perspectives, and Ultimate Goals," *Internation Power System Conference*, no. November 2013, 2013.

[168] J. Zheng, C. Xu, Z. Zhang, and X. Li, "Electric load forecasting in smart grids using Long-Short-Term-Memory based Recurrent Neural Network," in *2017 51st Annual Conference on Information Sciences and Systems, CISS 2017*. Institute of Electrical and Electronics Engineers Inc., 5 2017.

[169] Z. Aung, M. Toukhy, J. Williams, A. Sanchez, and S. Herrero, "Towards accurate electricity load forecasting in smart grids," *The Fourth International Conference on Advances in Databases, Knowledge, and Data Applications*, 01 2012.

[170] S. Bouktif, A. Fiaz, A. Ouni, and M. A. Serhani, "Multi-Sequence LSTM-RNN Deep Learning and Metaheuristics for Electric Load Forecasting," *Energies 2020, Vol. 13, Page 391*, vol. 13, no. 2, p. 391, 1 2020. [Online]. Available: https://www.mdpi.com/1996-1073/13/2/391/htmhttps://www.mdpi.com/1996-1073/13/2/391

[171] G. Sun, C. Jiang, X. Wang, and X. Yang, "Short-term building load forecast based on a data-mining feature selection and LSTM-RNN method," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 15, no. 7, 2020.

[172] X. M. Zhang, K. Grolinger, M. A. Capretz, and L. Seewald, "Forecasting Residential Energy Consumption: Single Household Perspective," *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, pp. 110–117, 1 2019.

[173] R. K. Jagait, M. N. Fekri, K. Grolinger, and S. Mir, "Load forecasting under concept drift: Online ensemble learning with recurrent neural network and ARIMA," *IEEE Access*, vol. 9, pp. 98 992–99 008, 2021.

[174] M. N. Fekri, H. Patel, K. Grolinger, and V. Sharma, "Deep learning for load forecasting with smart meter data: Online Adaptive Recurrent Neural Network," *Applied Energy*, vol. 282, p. 116177, 1 2021.

[175] Y. Tian, L. Sehovac, and K. Grolinger, "Similarity-Based Chained Transfer Learning for Energy Forecasting with Big Data," *IEEE Access*, vol. 7, pp. 139 895–139 908, 2019.

[176] M. A. Hammad, B. Jereb, B. Rosi, and D. Dragan, "Methods and Models for Electric Load Forecasting: A Comprehensive Review," *Logistics & Sustainable Transport*, vol. 11, no. 1, pp. 51–76, 2020.

[177] J. F. Chen, W. M. Wang, and C. M. Huang, "Analysis of an adaptive time-series autoregressive moving-average (ARMA) model for short-term load forecasting," *Electric Power Systems Research*, vol. 34, no. 3, 1995.

[178] S. J. Huang and K. R. Shih, "Short-term load forecasting via ARMA model identification including non-Gaussian process considerations," *IEEE Transactions on Power Systems*, vol. 18, no. 2, 2003.

[179] S. S. Pappas, L. Ekonomou, P. Karampelas, D. C. Karamousantas, S. K. Katsikas, G. E. Chatzarakis, and P. D. Skafidas, "Electricity demand load forecasting of the Hellenic power system using an ARMA model," *Electric Power Systems Research*, vol. 80, no. 3, 2010.

[180] J. Contreras, R. Espínola, F. J. Nogales, and A. J. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Transactions on Power Systems*, vol. 18, no. 3, 2003.

[181] B. Nepal, M. Yamaha, A. Yokoe, and T. Yamaji, "Electricity load forecasting using clustering and ARIMA model for energy management in buildings," *Japan Architectural Review*, vol. 3, no. 1, 2020.

[182] G. Scott, "Box-jenkins model definition." [Online]. Available: https://www.investopedia.com/terms/b/box-jenkins-model.asp

[183] H. M. Al-Hamadi and S. A. Soliman, "Short-term electric load forecasting based on Kalman filtering algorithm with moving window weather and load model," *Electric Power Systems Research*, vol. 68, no. 1, 2004.

[184] H. Zhao and S. Guo, "An optimized grey model for annual power load forecasting," *Energy*, vol. 107, 2016.

[185] N. A. Abd Jalil, M. H. Ahmad, and N. Mohamed, "Electricity load demand forecasting using exponential smoothing methods," *World Applied Sciences Journal*, vol. 22, no. 11, 2013.

[186] C. Wang, M. H. Chen, E. Schifano, J. Wu, and J. Yan, "Statistical methods and computing for big data," *Statistics and its Interface*, vol. 9, no. 4, 2016.

[187] M. Pereira Coutinho, C. Inácio de Almeida Costa Germano Lambert-Torres, F. Member, I. Ronaldo Rossi, L. Eduardo Borges da Silva, S. Member, and I. Carlos Henrique Valério de Moraes Maurilio Pereira Coutinho, "Big Data Techniques applied to Load Forecasting." [Online]. Available: https://www.researchgate.net/publication/289117292

[188] M. Cai, M. Pipattanasomporn, and S. Rahman, "Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques," *Applied Energy*, vol. 236, 2019.

[189] K. Nilakanta Singh and K. Robindro Singh, "A Review on Deep Learning Models for Short-Term Load Forecasting," 2021.

[190] D. Xishuang, Q. Lijun, and H. Lei, "Short-term load forecasting in smart grid: A combined CNN and K-means clustering approach," *2017 IEEE International Conference on Big Data and Smart Computing, BigComp 2017*, pp. 119–125, 3 2017.

[191] S. H. Rafi, Nahid-Al-Masood, S. R. Deeba, and E. Hossain, "A Short-Term Load Forecasting Method Using Integrated CNN and LSTM Network," *IEEE Access*, 2021.

[192] R. Sethi and J. Kleissl, "Comparison of Short-Term Load Forecasting Techniques," in *2020 IEEE Conference on Technologies for Sustainability, SusTech 2020*, 2020.

[193] L. Sehovac, C. Nesen, and K. Grolinger, "Forecasting building energy consumption with deep learning: A sequence to sequence approach," *Proceedings - 2019 IEEE International Congress on Internet of Things, ICIOT 2019 - Part of the 2019 IEEE World Congress on Services*, pp. 108–116, 7 2019.

[194] L. Sehovac and K. Grolinger, "Deep Learning for Load Forecasting: Sequence to Sequence Recurrent Neural Networks with Attention," *IEEE Access*, vol. 8, pp. 36 411–36 426, 2020.

[195] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 6 1989.

[196] Y. Peng, Y. Wang, X. Lu, H. Li, D. Shi, Z. Wang, and J. Li, "Short-term Load Forecasting at Different Aggregation Levels with Predictability Analysis."

[197] "Global Energy Forecasting Competition 2012 - Load Forecasting," Kaggle. [Online]. Available: https://www.kaggle.com/c/global-energy-forecasting-competition-2012-load-forecasting/data

[198] D. P. Kingma and J. Lei Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION."

[199] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[200] L. Biewald, "Experiment Tracking with Weights and Biases," 2020. [Online]. Available: https://www.wandb.com/

[201] T. Bachlechner, B. P. Majumder, H. H. Mao, G. W. Cottrell, and J. Mcauley, "ReZero is All You Need: Fast Convergence at Large Depth." [Online]. Available: https://github.com/majumderb/rezero

[202] X. S. Huang, F. Pérez, J. Ba, and M. Volkovs, "Improving Transformer Optimization Through Better Initialization," 2020. [Online]. Available: https://github.

[203] G. Notton and C. Voyant, "Forecasting of Intermittent Solar Energy Resource," *Advances in Renewable Energies and Power Technologies*, vol. 1, pp. 77–114, 1 2018.

[204] "Definition of Scalability," Gartner. [Online]. Available: https://www.gartner.com/en/information-technology/glossary/scalability

[205] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: parallel and distributed approaches*. Oxford: Cambridge University Press, 2011. [Online]. Available: http://dx.doi.org/10.1017/CBO9781139042918

[206] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[207] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11141 LNCS, pp. 270–279, 2018. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-01424-7_27

[208] G. Pinto, Z. Wang, A. Roy, T. Hong, and A. Capozzoli, "Transfer learning for smart buildings: A critical review of algorithms, applications, and future perspectives," *Advances in Applied Energy*, vol. 5, p. 100084, 2 2022.

[209] E. Skomski, J. Y. Lee, W. Kim, V. Chandan, S. Katipamula, and B. Hutchinson, "Sequence-to-sequence neural networks for short-term electrical load forecasting in commercial office buildings," *Energy and Buildings*, vol. 226, 2020.

[210] M. Ribeiro, K. Grolinger, H. F. ElYamany, W. A. Higashino, and M. A. Capretz, "Transfer learning with seasonal and trend adjustment for cross-building energy forecasting," *Energy and Buildings*, vol. 165, pp. 352–363, 4 2018.

[211] A. Cooper Stickland, X. Li, and M. Ghazvininejad, "Recipes for Adapting Pre-trained Monolingual and Multilingual Models to Machine Translation."

[212] F. Zhang, C. Bales, and H. Fleyeh, "From time series to image analysis: A transfer learning approach for night setback identification of district heating substations," *Journal of Building Engineering*, vol. 43, p. 102537, 11 2021. [Online]. Available: http://creativecommons.org/licenses/by/4.0/

[213] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation."

[214] "The rise of social media," Our World in Data. [Online]. Available: https://ourworldindata.org/rise-of-social-media

[215] E. Ezhilarasan and M. Dinakaran, "A Review on Mobile Technologies: 3G, 4G and 5G," *Proceedings - 2017 2nd International Conference on Recent Trends and Challenges in Computational Models, ICRTCCM 2017*, pp. 369–373, 10 2017.

[216] "Greenhouse Gas Inventory Data Explorer," U.S. Environmental Protection Agency. [Online]. Available: https://cfpub.epa.gov/ghgdata/inventoryexplorer/#allsectors/allsectors/allgas/econsect/all

[217] "Canada's Renewable Power Landscape," National Energy Board, Tech. Rep., 2017.

Curriculum Vitae

Name:	Alexandra L'Heureux
Post-Secondary Education and Degrees:	University of Western Ontario London, ON 2013 - 2015 MESc Software Engineering University of Western Ontario London, ON 2009 - 2013 BESc Software Engineering
Honours and Awards:	NSERC CGS D 2016-2019 OGS D 2015-2016 NSERC CGS M 2014-2015 OGS M 2013-2014
Related Work Experience:	Teaching Assistant The University of Western Ontario 2013 - 2022 Part-time Lecturer The University of Western Ontario 2021

Publications:

A. L'Heureux, K. Grolinger, H. F. Elyamany and M. A. M. Capretz, "Machine Learning With Big Data: Challenges and Approaches," in IEEE Access, vol. 5, pp. 7776-7797, 2017, doi: 10.1109/ACCESS.2017.2696365.

A. L'Heureux, K. Grolinger, and M. A. M. Capretz, "Transformer-Based Model for Electrical Load Forecasting," Energies, vol. 15, no. 14, p. 4993, Jul. 2022, doi: 10.3390/en15144993