Monash University

Wavenet Architectures for Time Series Forecasting

This thesis is presented in partial fulfilment of the requirements for the degree of Master of Data Science at Monash University

Author: Naveen Kaushik
Supervisor: Dr Christoph Bergmeir

Year: 2020

Abstract

Forecasting is a vital part of many industries in a wide, more than ever-growing industry. Traditional forecasting models like ETS and ARIMA are still the goto tools of any forecaster due to their robustness, simplicity, and explainability. But, with the increase in the amount of data, these univariate models fail to scale up to the needs of the industry because of which, the need of having global forecasting models is increasing day by day. Recurrent Neural Networks (RNN) models have become quite popular and competitive in recent times. The RNN models generally take more time to train and needs more preprocessing as well. Convolutional Neural Networks(CNN) are the more preferred choice in the academia and the industry.

We provide an extensive study and a software framework of WaveNet networks, which are a particular type of CNNs, on various time series datasets and compare them against the benchmark methods. We also do an ablation study to understand the effect of various additional components of an open-source implementation of a WaveNet network. From these analyses, we show that there are certain cases where the WaveNet network outperforms the traditional methods but in some, it fails to do so. We also observe that these WaveNet networks also need a lot of tuning of hyperparameters and are very susceptible to the input window that is fed into them. The ablation study shows us that a particular kind of dropout layer, called Concrete Dropout, gives a significant improvement to the model against a simple dropout layer.

Acknowledgements

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the work of others has been acknowledged.

Signed by:

Name:
Date:

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Given the volatile and ever-changing nature of the surrounding, it is critical for businesses and the industries to have an accurate estimate of how the consumers needs change. For example, a retailer needs to know about how many items from a particular department will be sold in the coming week so that appropriate replenishment plans can be made. This makes the field of time series forecasting of paramount importance.

The forecasting field has been dominated by traditional univariate methods and these are still one of the most widely used methods in the industry (Hyndman and Khandakar (2008)). But with the new era of big data where the data is growing exponentially, it is difficult for these univariate methods to scale up. In recent times, with the growth of related time series together, global forecasting models have come into the picture.

In this thesis, we explore a particular kind of global model made of a one-dimensional convolutional network with dilations, which forms a WaveNet architecture (Simonyan et al. (2016)). We explore the WaveNet models on the datasets of different size, seasonality and trends.

1.1 Area of study

The main area of study for this research is Time Series Forecasting. A Time Series is a series of data that can be recorded at every time step in fixed intervals. This fixed interval can be as granular as every second, minute, hour or as aggregated as month, year or decade. Time series forecasting is a science of predicting the future values of such time series data. It is helpful for the industry to have an accurate prediction of what kind of demand is coming from the consumers in the future.

Considering the example of a retail industry, a retailer would want to know how often the shelves of its store replenished so that a balance can be made between the supply and the demand. The demand from time to time varies on the changes of various external factors. Sometimes, these external changes are related to each other and hence make the demands from the customer change in a similar fashion. For example, an external factor like the change in the weather to winter would have an effect on the sale of different kinds of clothes, shoes as well as household equipments. In some way, all these demands are related to each other. In today's world, with the increase in the amount of data, univariate forecasting models are not capable enough to scale up with the scale of the data and also do not have a way to use cross-series information which can be potentially useful in forecasting. Global forecasting model have proven to be useful (Bandara et al. (2017), Sen et al. (2019)).

The approaches of such global forecasting models generally include a special kind of Neural Network called Recurrent Neural Network (RNN) that learns the cross-series information from all

the series. The recently completed M4 forecasting competition (Makridakis (2018)) is an example of such global RNN forecasting models. Convolutional Neural Networks (CNN) are another kind of neural network that is widely used in image recognition. A special kind of 1-dimensional CNN has shown its potential on raw one-dimensional audio data (Simonyan et al. (2016)). These networks are explained in detail in further sections.

There have been some successful usages of a one-dimensional convolutional network on time series data in the past (Borovykh et al. (2017), Toby Bischoff (2019)). This research focuses on using a special one-dimensional network called WaveNet introduced by Simonyan et al. (2016) on various time series datasets and tries to analyse what works better for a time series data and what does not.

1.2 Motivation

As established above, the global forecasting models have the capability to scale up with the big data (Suilin (2017), Bandara et al. (2017), Sen et al. (2019), Smyl (2018)). Bandara et al. (2017) uses a clustering approach to first cluster the set of series into smaller subgroups and then build a model per group. This approach works better than building one model for all the series. Smyl (2018) developed a hybrid model with a combination of Exponential Smoothing and RNN.

The problem with the RNN model is that they require a lot of preprocessing to be done and there are lots of parameters that need to be finely tuned for the network to work. One-dimensional convolutional networks and the WaveNet networks have shown some promises to overcome those challenges (Toby Bischoff (2019), Borovykh et al. (2017)). All these methods use a similar architecture but with small variations to it to fit the need of the data. As part of this research, we wanted to perform analysis on a variety of time series datasets and see how good the WaveNet model works and what specific variations in them help in the forecasting accuracy.

1.3 Significance

As it is stated earlier, forecasting is an indispensable component of the industry. Businesses rely heavily on having an accurate forecast. One of the most common used tool is the *forecast* package(Hyndman and Khandakar (2008)) implemented in R programming language (R Development Core Team (2011)). However, these methods don't scale well because they are univariate and make assumptions about the data that they are forecasting (Taylor and Letham (2017)). The WaveNet models have proven to be quite useful in forecasting and have the potential of being an automated forecasting tool.

Hence, this research is significant in terms of understanding how well do WaveNet networks work on time series datasets of varying length, trend and seasonality. Having a scalable and automated forecasting tool would be highly significant for industries.

1.4 Research aim and Questions

This research is aimed at implementing a WaveNet architecture for a time series dataset and evaluating its performance on a variety of datasets. Based on the above, the primary research question is:

- Can a plain WaveNet model be used for general purpose time series forecasting? How does the performance of WaveNet compare against to benchmark methods?

The above main question can be split into sub-questions as follows:

- What variations on WaveNet architecture is more suitable for which kind of datasets?
- In what kind of datasets are WaveNet models better than the available benchmark methods like ETS(Gardner Jr. (1985)) and ARIMA (Box et al. (2015))?
- What characteristics of a particular data affect the performance of the WaveNet model?

1.5 Outcomes and contributions

Based on the questions defined above, the following would be the outcomes and contributions of the research:

- Forecasting tool: An automatic forecasting tool that is based on a WaveNet architecture which can forecast based on an input data for a specific forecast horizon.
- Evaluation of WaveNet on time series datasets: The research will produce results of the WaveNet network on a variety of datasets and provide the analysis to compare the results.
- Ablation study: An ablation study on the *Deep4Cast* library provided by Toby Bischoff (2019) to understand how various components of WaveNet contribute to the forecasts.

1.6 Structure of the thesis

The structure of this thesis is as follows. Following the introduction from this chapter, a literature review is provided in Chapter 2, that provides an in-depth review of the global forecasting models and the WaveNet architectures. This section also provides a review of some of the applied variations to the WaveNet models.

Chapter 3 provides the methodology taken to forecast and the details of the architecture built. Chapter 4 presents the results based on the experiments conducted. The final chapter, Chapter 5 discusses and analyses the results in order to address the research questions defined in this chapter. To end with, a conclusion of the entire research is presented.

Chapter 2

Literature Review

Before going into understanding how a WaveNet network works, it's important to understand what univariate forecasting is and why do we need a global forecasting model. This is discussed in section 2.1 and section 2.2 respectively. Following that, we discuss convolutional network, causal and dilated convolutional networks in section 2.3. Then a detailed explanation of WaveNet models is given in section 2.4 followed by a brief about concrete dropout layers. These details then set the grounds for the methodology in the next chapter.

2.1 Univariate Forecasting

Univariate forecasting refers to predicting the future steps of a time series based on its own values in the past. In a general form, a time series can be defined as $X = \{x_1, x_2, ..., x_t, ..., x_T\}$. The future time steps of this series can be defined as $Y = \{x_{T+1}, x_{T+2}, ..., x_{T+H}\}$. Here the original series is till time step T and the univariate forecasting mechanism has the forecasting horizon of H. The problem can formally be defined as Hewamalage et al. (2019):

$$\{x_{T+1}, x_{T+2}, ..., x_{T+H}\} = F(x_1, x_2, ..., x_t, ..., x_T) + \epsilon$$

where, ϵ is the error in function approximation

F is the approximated function

H is the forecast horizon

As seen based on the above formulation of the problem, univariate forecasting of time series would need to build one model for each series. This limits the forecasting to be not scalable as the data grows.

2.2 Global Forecasting Models

With the enormous increase of data in the past decades, it becomes infeasible to have one model per time series as in the case of univariate forecasting. It also has the disadvantage of not being able to use cross-series information. These drawbacks make a strong case for global models. The crux of the global model is that instead of training one univariate model per time series, a global model is trained which uses information from all the available series.

In the real world, an example of a global model can be considered in the retail domain. A retail store has various products and they can be allocated to a particular stock-keeping unit(SKU). The store owner might be interested in forecasting the sale of each of the SKUs. Given the huge number of SKUs, it would be difficult to build and maintain a huge number of models. A global model in such scenarios is useful to forecast wherein one single model can be used to forecast for all the SKUs. Another problem that the retailer might have, is called a "cold-start" problem for newly introduced SKUs (Daoud et al. (2015)) which do not have any sale history. As the univariate models depend on the historic values of the same series, it cant be used to forecast because there is no history. Whereas, a global model can be used to find similar SKUs and use their history to forecast for the particular SKU under observation. Trapero et al. (2015) uses the exogenous variables like prices and promotions of the products to forecast the sale of the products. Chauhan et al. (2020) proposes a Dynamic Key-Value Memory Network (DKVMN), which are usually used in knowledge tracing for binary tasks, to handle the "cold-start" problem.

Bandara et al. (2017) proposes a sub-global model by first clustering the set of time series into sub-groups using the *anomalous-acm* package(Hyndman (2015)) and then training a model per group to forecast the sale of products. Smyl (2018) also applied the winning solution called ES-RNN(Exponential Smoothing - Recurrent Neural Network) on the M4 forecasting competition. This solution creates a global model using RNN and a local model using ES to model the local patterns of a time series. Salinas et al. (2017) also proposed the idea of creating global models based on the whole set of data and using cross-series information for probabilistic forecasting. Mukherjee et al. (2018) proposes a Neural Network architecture to forecast thousands of products from an online retail chain. Sen et al. (2019) proposes an approach for high-dimensional time series by exploiting the global patterns among them and also calibrating locally for each series.

Benidis et al. (2020) provides an in-depth review of the literature around Neural Networks for forecasting and Hewamalage et al. (2019) provides a direction for them specifically focussing on RNNs.

2.3 Convolutional Network

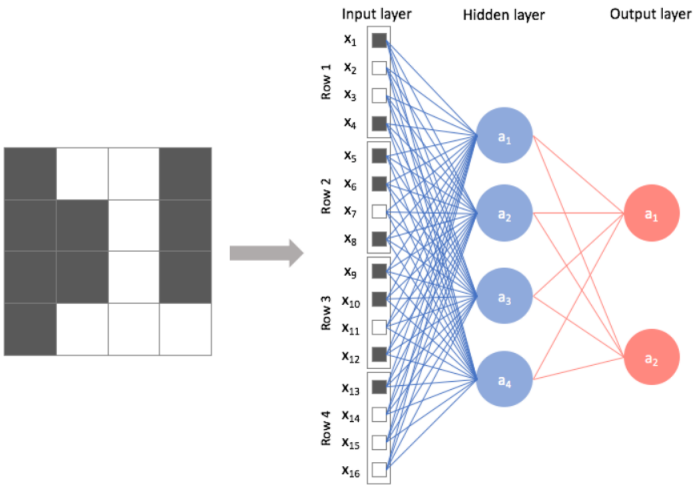

Figure 2.1: Feed forward network for a 4x4 image (Image Source: Jordan (2017))

A feed forward neural network has an architecture where every node in a layer is connected to every other node in the next layer. Considering an example of an image input data, an example is shown in Figure 2.1 where a 4x4 image is processed by a feed forward network.

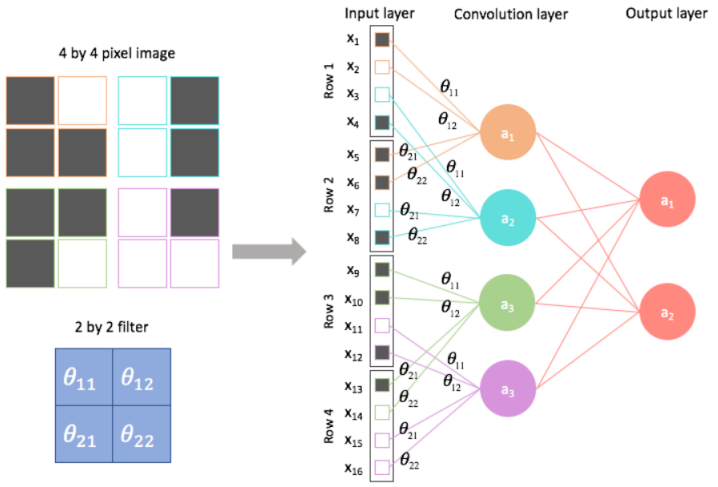

Figure 2.2: Convolutional network for a 4x4 image (Image Source: Jordan (2017))

As seen in the Figure 2.1, in a feed forward network all the nodes of the input layer are connected to all the nodes of the hidden layer which makes the network lose its spatial information.

On the other hand, a convolutional network maps out regions of an image locally as shown in Figure 2.2. In case of a convolutional network, 2x2 filter is applied on the original image and each node in the convolutional layer corresponds with the output for each of the 4 filters. This is displayed in different colors in the Figure 2.2. This helps in preserving the spatial information of the image data. Convolutional networks have shown there benefits on image data in the past (Sultana et al. (2019), Krizhevsky et al. (2012)).

2.3.1 One-dimensional Convolutional Network

One-dimensional convolutional network are convolutional networks that work on one-dimensional data, for example, audio, time series etc. as shown in Figure 2.3.

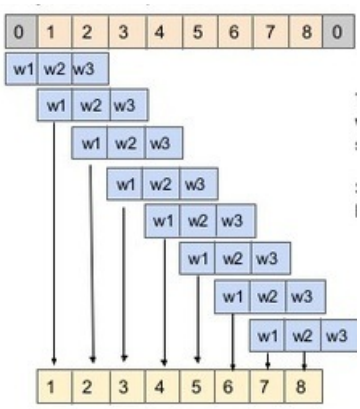

Figure 2.3: One-dimensional convolutional network (Image Source: Ghosh (2017))

As seen in Figure 2.3, a one-dimensional moving filter is applied to the data and the dot product of the two is taken to emit out the output. The filter moves with a stride of 1 and moves from the start to the end. One-dimensional convolutional network have been proven to be useful on one-dimensional data in the past (Kiranyaz et al. (2016), Kiranyaz et al. (2019), Ince et al. (2016)).

2.3.2 Causal Convolutional Network

The word *causal* originates from signal processing (contributors (2017)). It indicates that the output only depends on the current and the past time inputs and it does not depend on future time steps. It is as shown in Figure 2.4.

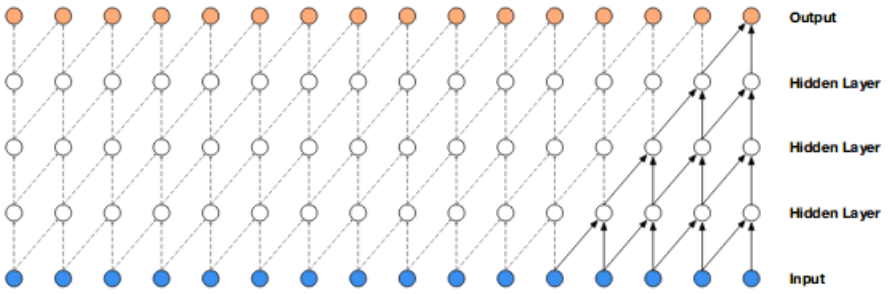

Figure 2.4: Visualising a stack of causal convolutional layers (Image Source: Simonyan et al. (2016))

The architecture of a causal convolutional network ensures that the output emitted by the model at time t, denoted by $p(x_{t+1}|x_1, x_2, ..., x_t)$ only depends on the past time steps i.e. $x_1, x_2, ..., x_t$ and does not depend on any future time steps i.e. $x_{t+1}, x_{t+2}...x_T$.

One of the problems with the causal networks is that they need more and more layers to increase the receptive field of the network. For example, in Figure 2.4 the receptive field size is 5 which is equal to *number of layers + filter length -1*. A dilated convolution network is the solution to improve the receptive field which is discussed in the next section.

2.3.3 Dilated Causal Convolutional Network

A dilated convolutional network also called a convolution network with holes helps in increasing the receptive field of the network (Huszár (2016), Simonyan et al. (2016)). A dilated convolution can be thought of as a large convolution filter and skipping some values systematically in between them as seen in Figure 2.5.

In the default setting, as seen in Figure 2.5, a dilated convolution with dilation factor 1 gives a standard convolution. An example of dilation factor 1, 2, 4, 8 is shown in Figure 2.5. Dilated convolutional networks have been used in the past in various signal processing, image segmentation and time series forecasting contexts (Combes et al. (1990), Yu and Koltun (2015), Chen et al. (2014), Borovykh et al. (2017)).

2.4 WaveNet

WaveNet is a deep generating neural network that operates on the raw audio waveforms. Its probabilistic and autoregressive in nature. The joint probability for a waveform $x = \{x_1, x_2 \dots x_T\}$

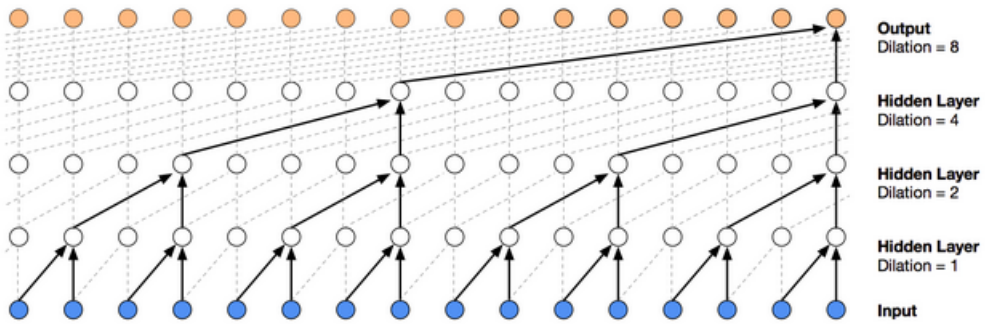

Figure 2.5: Visualising a stack of dilated causal convolutional layers (Image Source: Simonyan et al. (2016))

can be factorised as a product of conditional probabilities as shown in Equation 2.1 (Simonyan et al. (2016)):

$$p(x) = \prod_{t=1}^{T} p(x_t|x_1, x_2, ..., x_{t-1}) \tag{2.1}$$

This means that each of the sample x_t is conditional on all its previous time steps. This is a similar situation for forecasting where the forecast of a future time step is dependent on all the past time values.

Van Den Oord et al. (2016) also proposed a similar conditional probability distribution that was modelled using a stack of causal convolutional layers. WaveNet also uses a similar idea without having any pooling layers. The key component of WaveNet model is the causal convolutions which ensures that the ordering of the data is maintained while we model the data, i.e. the prediction $p(x_{t+1}|x_1, x_2, ..., x_t)$ cannot depend on any of the future time steps $x_{t+1}, x_{t+2}...x_T$.

The original WaveNet architecture is as shown in Figure 2.6 which was proposed for the raw audio data. In this original architecture, Simonyan et al. (2016) proposed that, since the prediction is done one time step at a time, the training phase can be parallelised. But during actual prediction, it would be done one step at a time and then the output would be fed back into input for the prediction of the next time step.

As discussed in section 2.3.2, causal convolutional layer struggle to get a large receptive field without increasing the number of layers significantly. This problem is solved by using dilated networks. In WaveNet, the dilation factor is doubled on each layer and then repeated as multiple blocks. For example,

1,2,4,8,.....,256,512,1,2,4,8,.....,256,512,1,2,4,8,.....,256,512

The reason for such a configuration is two-fold. First, increasing the dilation factor exponentially increases the receptive field growth exponentially (Yu and Koltun (2015)). Second, such stacking of layers increases the capacity of the model and receptive field size.

Although the audio data is a 16-bit integer values and would have one value per time step, which amounts to 65,536 probabilities per time step, a softmax distribution is used because it works better for an implicit continuous data (Van Den Oord et al. (2016)). One other reason to support a categorical distribution is its flexibility and the fact that it does not make assumption about the data (Simonyan et al. (2016)). To make the problem of predicting 65,536 probabilities more controllable, a μ-law (ITU-T (2017)) is applied and then it is quantized to 256 discreet values

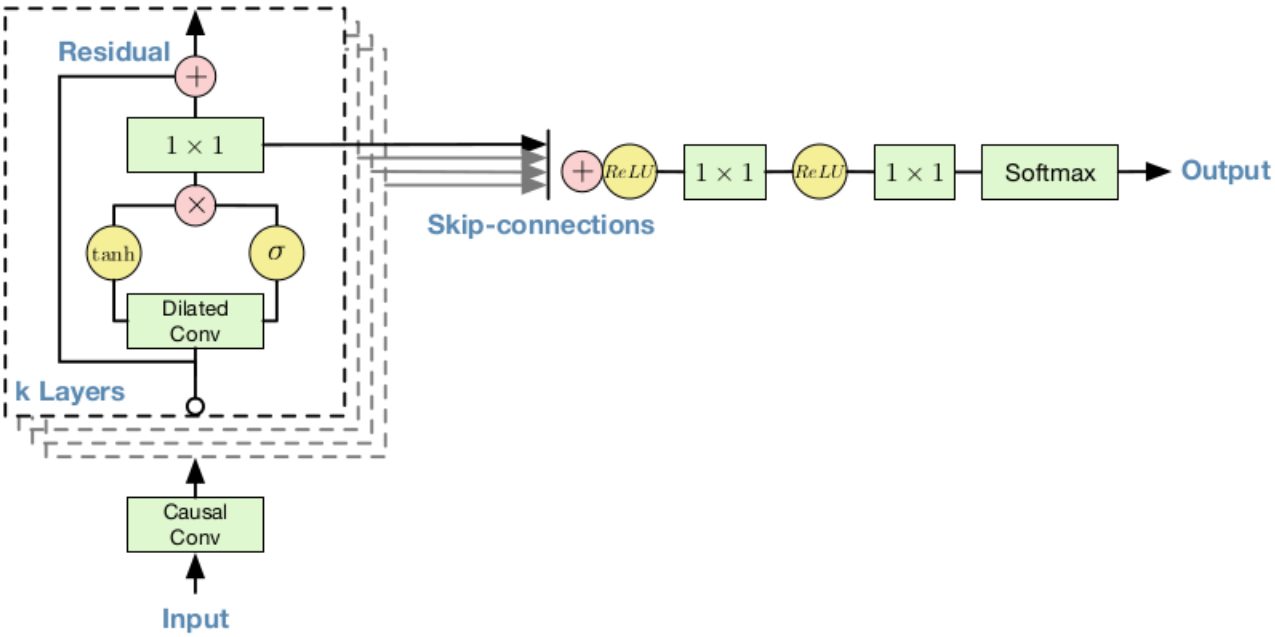

Figure 2.6: WaveNet architecture (Image Source: Simonyan et al. (2016))

as shown in the below equation:

$$f(x) = sign(x_t)\frac{ln(1+\mu \mid x_t \mid)}{ln(1+\mu)}$$

$$\text{where, } -1 < x_t < 1$$
$$\text{and } \mu = 255$$

WaveNet uses the same gated activation units as suggested by van den Oord et al. (2016) and is defined as follows:

$$z = tanh(W_{f,k} * x) \odot \sigma(W_{g,k} * x)$$

where, $*$ denotes a convolutional operator
$\odot$ denotes an element-wise multiplication operator
σ is a sigmoid function
k is the layer index
f and g denote filter and gate

WaveNet uses Residual (He et al. (2016)) and skip connections throughout the network in order to speed up convergence and train deeper models. Figure 2.6 shows one block of the WaveNet model which would then be stacked multiple times to complete the architecture.

2.5 Concrete Dropout

Dropouts in a neural network are a way to avoid overfitting (Srivastava et al. (2014)). Labach et al. (2019) does an in-depth review of various dropout techniques. Gal et al. (2017) proposes a method "concrete dropout" wherein based on Bayesian deep learning, a continuous relaxation of dropout mask is done to calibrate the probability of a dropout layer. This helps in automatic tuning of dropout probability and faster experimentation. Toby Bischoff (2019) uses concrete dropout on WaveNet models for time series forecasting.

2.6 Literature Review Conclusion

Univariate methods have been used for a long time because of their simplicity and explainability in the time series forecasting space. Because of the enormous increase of the data, it has become challenging for the univariate models to scale up with the data. Global models, especially RNNs have shown great potential in achieving state-of-the-art results.

WaveNet models have proven to be effective on 1-dimensional data. The causal nature of the WaveNet makes it suitable for time series forecasting problems. They have also shown the potential to work well on time series forecasting problems. There have been a lot of variations that have been tried on WaveNet to make them perform better on time series data.

The examples in the literature shown have proven to be working on a specific dataset. There doesn't exist a comprehensive review of vanilla WaveNet models on time series datasets. This research tries to provide a comprehensive study of WaveNet and experiment with its variations via an ablation study to understand what kind of variations work better for a WaveNet network.

Chapter 3

Methodology

In this chapter, we discuss the experimental framework designed for WaveNet models. It also includes datasets used, pre-processing techniques experimented on the datasets and the ablation study.

3.1 Model Architecture

The architecture of the model is as shown in Figure 3.1. This is based on the WaveNet setup as explained by Simonyan et al. (2016) for a raw audio input data. The input to the model is based on a moving window fashion of the original time series which is explained in the further sections.

The input is first passed through a convolutional filter, the output of which is passed through a set of layers which make the WaveNet "core" blocks. Each of WaveNet core block looks as shown in the figure with a varying dilation factor. Each block basically consists of a dilated convolutional filter with a Rectified Linear Units(Relu) activation layer as introduced by Nair and Hinton (2010). As shown in the figure, the residual connection is achieved by applying another layer of convolutional filter on the Relu output. This is then added to the raw input to the WaveNet block as indicated by the $\oplus$ in the figure. Each WaveNet block emits two outputs as indicated by out1 and out2.

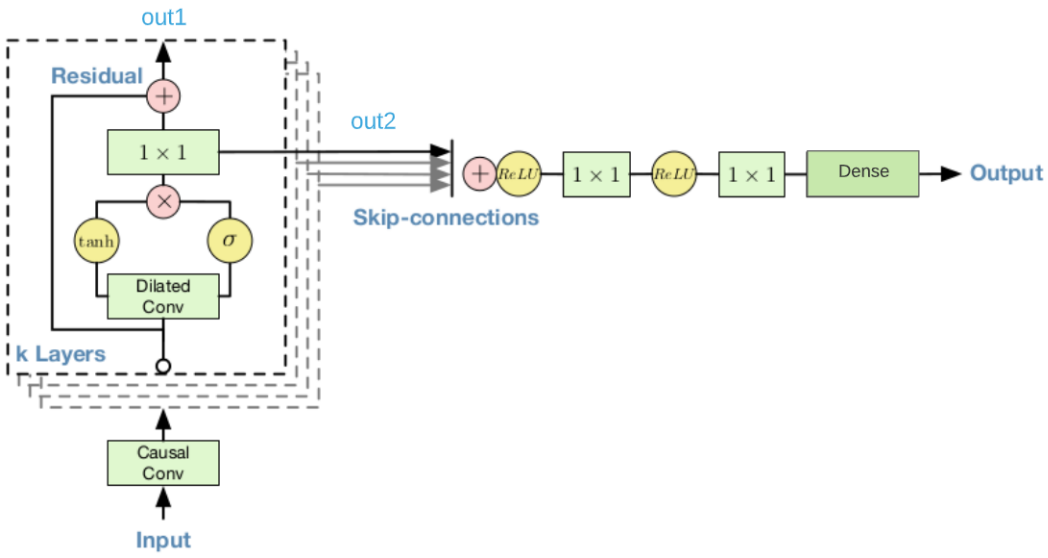

Figure 3.1: Model architecture (Image Source: Simonyan et al. (2016))

All the out2s from the WaveNet blocks are then added together and passed through a Relu activation layer. Finally, a dense layered output is produced based on the forecast horizon needed for the time series. The original WaveNet model, which was built for raw audio data, proposed by Simonyan et al. (2016) uses a softmax distribution instead of a dense layer. A raw audio data is a 16-bit integer value that is stored at every time step and hence the softmax distribution would need a 65,536 probabilities for each time step to model the data. Van Den Oord et al. (2016) argues that using the softmax is a better choice than a dense layer to model such a huge number of possibilities and hence Simonyan et al. (2016) applies a μ-law (ITU-T (2017)) to make the problem more controllable. This is not a viable solution in case of forecasting because in forecasting we try to be as close to the actual value and every single variation accounts for the increase of error. Hence a dense layer was used which was equal to the forecasting horizon of the dataset.

Depending on the dilation or the receptive field for the dataset that needs to be achieved, the WaveNet blocks are added to the model. The dilations for every layer is doubled than to the previous layer which means we would need 5 layers to achieve the receptive field size of 16 (1,2,4,8,16).

3.2 Datasets

The experiments are conducted on various datasets sourced from the forecasting competitions that have happened in the past years. Some of the dataset is processed before it is used and the details are provided in further sections. The major sources of the datasets used are as follows:

1. M4 Forecasting competition (Makridakis (2018))
2. NN5 Forecasting competition (Crone (2008))
3. Wikipedia Web Traffic Time Series Forecasting (Google (2017))

Dataset Name	Number of Time series	Forecasting horizon	Frequency	Maximum length	Minimum length
M4-yearly	23000	6	Yearly	300	13
M4-quaterly	24000	8	Quarterly	866	16
M4-monthly	48000	18	Monthly	2794	42
M4-weekly	359	13	Weekly	2597	80
M4-daily	4227	14	Daily	9919	93
M4-hourly	414	48	Hourly	960	700
NN5-daily	111	56	Daily	735	735
NN5-weekly	111	8	Weekly	105	105
Wikipedia Web traffic - Weekly	1000	8	Weekly	106	106

Table 3.1: Datasets used

There are various strategies that can be applied to the model with regards to predicting the forecasting horizon. Taieb et al. (2012) does an in-depth review on such various strategies. This particular research is focused on a multi-step ahead forecasting which is explained in section 3.4. Of the datasets listed above, the dataset from NN5 and Wikipedia web traffic dataset consists of series which have a 0 value. Various other details about all these datasets are provided in Table 3.1.

The M4 dataset has a total of 100,000 time series with varying seasonality as shown in Table 3.1. The original NN5 dataset consists of 111 daily time series of cash withdrawals across different cash machines in England (Crone (2008)). The NN5 competition was held in 2008. The daily dataset from the NN5 was aggregated to a weekly level as well and used in the experiments. The NN5 dataset also had some missing values in some of the series and they had been imputed as explained in section 3.3.2.

The Wikipedia Web Traffic dataset is a dataset of approximately 145,000 time series denoting the web traffic on the Wikipedia articles of around 2 years (Google (2017)). For this particular research, only a limited top 1000 time series were picked, which were then aggregated on a weekly level making the lengths of time series to be of 106 and the forecasting horizon to be 8. A particular thing to note about these series was that all the series have an integer value, indicating the exact number of hits that happened on a Wikipedia article. For this reason, the forecasts from the models were rounded off to the nearest integer value before evaluating them.

Table 3.1 shows a short overview of all the datasets considered in this research. The series are of varying lengths and seasonality and picked from various domains.

3.3 Data Preprocessing

Before using the time series data, various preprocessing steps are applied to it. This helps in stabilising the variance of the data and transforming to a moving window approach, which is discussed in detail in further sections. Most of the ideas are similar to as used by Hewamalage et al. (2019).

3.3.1 Dataset - Training data

For each series in all the datasets mentioned, training and validation data are generated based on a moving window approach using a Multi-Input Multi-Output(MIMO) strategy as discussed by Taieb et al. (2011).

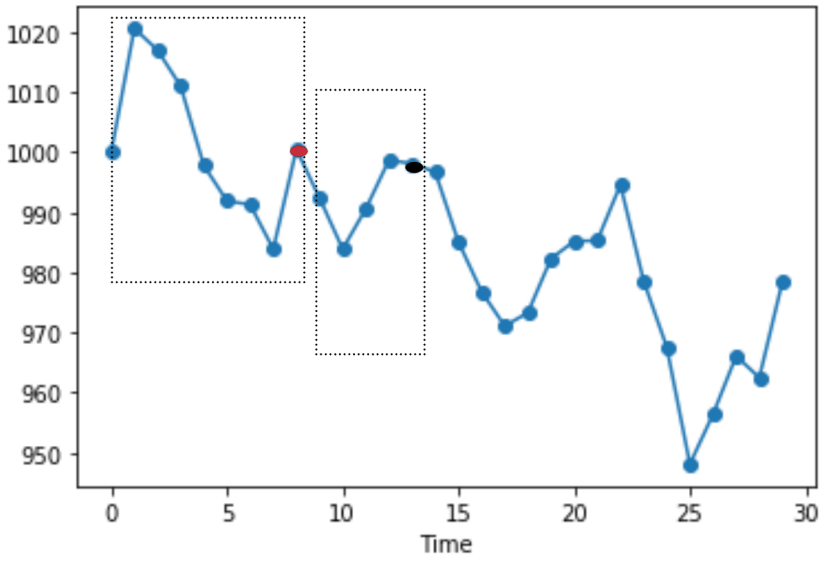

Figure 3.2: Multi-Input Multi-Output(MIMO)

An example of that is shown in Figure 3.2. The first dotted box indicates the training window and the last point in the training window is marked in red. The second box indicates the output window and the last data point in the output window is marked in black. Now to generate the

training data out of the series, the entire window keeps shifting by one step until the last data point in the time series is reached.

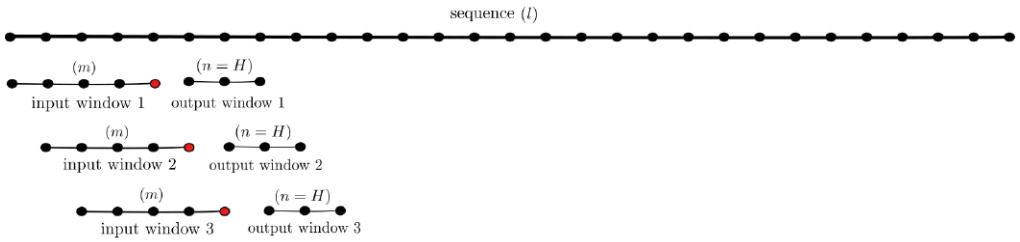

Figure 3.3: Moving window scheme (Image Source: Hewamalage et al. (2019))

An example of a series being split into training and validation for a series of length l and forecasting horizon H is shown in Figure 3.3. Every series is preprocessed based on the moving window approach as shown in the figure and is explained below.

To start with a series of length l which needs to be preprocessed for an input window length of m and output window of length H, it is converted to shorter series of length $m + H$. So there are a total of l - m - H + *1* number of series created. In the creation of training data, the last H number of data points are not included because they are used as a validation set for hyperparameter tuning. This finally results in l - m - H - H + *1* number of shorter series for each series. This technique has a disadvantage that the last H number of data points are never used for training and hence once the hyperparameters are tuned, the model is retrained on the entire dataset including the validation set as well.

3.3.2 Missing values treatment

Having a missing value in a time series is a very common problem and Moritz et al. (2015) does an in-depth review of various techniques. Out of the datasets selected for this research, NN5 daily and the Wikipedia web traffic dataset had missing values. For the NN5 daily dataset, a median substitution technique was used which is in-line with the technique used by Hewamalage et al. (2019). The NN5 daily data has a strong weekly seasonality and hence the missing values are substituted using the same day's data from the rest of the series. For example, if a particular missing value occurred on a Friday, then the median of all the other available Friday's was taken to substitute the missing value. The NN5 daily data was processed as described above and then the NN5 weekly data was generated using the process described in the next section. The Wikipedia web traffic data did not have a distinction between a '0' and a missing value, because 0 is a valid value in this case which denotes no traffic on that particular day. Because of this, the missing values in Wikipedia web traffic data were replaced with 0.

3.3.3 Conversion to weekly data

The NN5 weekly and the Wikipedia web traffic weekly datasets are aggregated based on their respective daily datasets. The aggregation was performed by summing up the first seven values and then move to the next seven. This resulted in, for example, the NN5 weekly time series length to reduce seven times from 735 to 105. Since the missing value treatment was already done on the daily dataset, the weekly ones did not have any missing values.

3.3.4 Variance of data

A transformation is a useful technique to stabilise a time series if there is a variation that increases or decreases with time and level (Hyndman and Athanasopoulos (2018)). One useful family of transformation is the family of Box-Cox transformations (Box and Cox (1964)). It is defined as:

$$f(x) = \begin{cases} \log(y_t) & \text{if } \lambda = 0 \\ \frac{(y_t^{\lambda} - 1)}{\lambda} & \text{otherwise} \end{cases}$$

The logarithm as defined in a Box-Cox transformation is to the base e (i.e. a natural logarithm). In this way, if $\lambda = 0$, natural logarithm is used and if $\lambda \neq 0$, a power transformation is used which also goes through a simple scaling.

If $\lambda = 1$, then $w_t = y_t - 1$ which basically means that the data is shifted down by 1 unit and but the shape of the time series is not changed. To use a Box-Cox transformation, a tuned λ value needs to be chosen.

Because of the complexity involved in tuning a λ parameter and having a dataset that has all non-negative values, just a logarithmic transformation was used in the experiments conducted which can be thought of as using the Box-Cox transformation with a fixed value of $\lambda = 0$. Since a logarithm is not defined at 0, the time series containing a 0 value was added by a constant factor 1 and then used for training the model. On the trained model, once the forecasts were generated, the same constant factor 1 was subtracted from the series.

3.4 Model training

The model training is done after the preprocessing of the data and a set of hyperparameters are tuned which are described in the next section. The training of the model is done on CPU resources provided by eResearch Centre (2020). The configuration of the machine is shown as below:

- **CPU Model Name:** Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz
- **CPU Architecture:** x86_64
- **CPU Cores:** 12
- **Memory(GB):** 14

3.4.1 Hyperparameters

As part of the model training process, various hyperparameters were tuned using the validation data. Following were the hyperparameters:

1. Batch size
2. Learning rate
3. Dropout rate
4. Number of convolutional filter
5. Filter length
6. Dilation

7. Layer regularizer

The first parameter 'Batch size' denotes the number of training data points that a model would iterate over before updating the weights in the model. It can be thought of as a loop running over all the data points and taking a 'batch size' number of points each time. For each batch, a prediction would be made and the calculated errors would be backpropagated to update the model weights. On one extreme of the batch size, if it is set to 1, the backpropagation would happen for each series. It should be noted that since a moving window scheme is used to create the training data, as explained in section 3.3.1, one data point is not equivalent to one time series in the original dataset that is being used but instead, is a section of the original series.

The Adam optimizer used requires a learning rate parameter to take a step in the direction of minimum loss. Learning rate defines that step and is tuned based on the validation data. There are 2 dropout layers added in the network, of which the dropout probabilities are tuned. The dropout probabilities indicate the probability by which a particular neuron in the network is expected to be dropped. A key thing to note here is that the dropout probabilities are used even while forecasting the test data as well, because of which there are multiple samples taken in for each data point while forecasting in the test data. This is explained in detail in section 3.5.

Convolutional filters in the WaveNet network define the number of filters that are passed over each batch of data. The filter length of each of the filter defines the window size that the filter convolves over. The number of filters and the filter length were kept the same for all the convolutional layers. Dilation factor needed to be changed based on the varying length of the series of different datasets. In order to avoid overfitting of the model, a L2 regularization is applied to all the weights in the convolutional layer. The weights of the layers are initialised using a normal distribution with a mean 0 and a standard deviation of 0.05.

3.4.2 Hyperparameter optimisation

There are various techniques available for tuning the hyperparameters. Yu and Zhu (2020) does an in-depth review of the available techniques along with the open-source software/libraries that can be used as well. The most simple one is to tune all the parameters manually, with the help of some domain knowledge, if available. Although, hand tuning requires a lot of experimentation which is computationally expensive and doesn't guarantee the most optimised solution. In the direction of a more automated solution, grid search and random search are the two most basic solutions. Grid search runs all the possible combinations of the parameter and narrows down on the best one, whereas random search tries a random combination of hyperparameters and selects the best one among them. On one hand, grid search guarantees the best combination but is highly computationally expensive, random search can be much quicker but does not guarantee an optimal result.

3.4.2.1 Automatic hyperparameter optimisation

To deal with the problem of selecting the right hyperparameters for the model, an open source implementation based on Bergstra et al. (2012b) is used. Bergstra et al. (2012b) proposes an approach to tune the hyperparameters based on Tree of Prazen Estimator(TPE) (Bergstra et al. (2011)). Bergstra et al. (2012b) shows based on the results that a TPE based approach converges to the best hyperparameter in much less trials. A python based implementation of the package *hyperopt* is used for these experiments (Bergstra et al. (2012a)).

3.5 Model Forecast

Once the model parameters are tuned based on the above techniques, a forecast is generated per data point. Since the dropout layers are used in the forecasting phase as well, 100 samples are generated for each of the input data points. Each forecast is different because based on the dropout probability, a different dropout mask is applied in the forward pass of the data in the network. Since in each of the 100 samples the dropout mask changes, it results in different forecasts for the same data point. This approach makes the model more robust and reduces the uncertainty (Gal and Ghahramani (2015)).

3.6 Model testing

Based on the model training, once an optimal set of hyperparameters is obtained, that configuration is used to train the final model. Since as mentioned earlier, during the training phase a set of data points equal to the forecasting horizon were kept aside for tuning the hyperparameters, the final forecasts are generated using all the available data points. The issue of a Neural Network initialisation is quite a common one (Skorski et al. (2020)). To address that issue, the model was trained 10 times with 10 different random seeds and the mean of the error measures were calculated which are presented in a further section.

3.6.1 Post processing of data

Based on the preprocessing that was done on the data, the forecasts from the model would also be on a similar scale. To get back the forecasts to its original scale in order to compute the error metrics, the following steps are applied:

1. Exponentiate the forecast to reverse the effect of log transformation.
2. Subtract 1 if the original series contained 0
3. If the data needs to be an integer, round the forecast to its closest integer.

3.6.2 Error metrics

The performance of the models are evaluated based on Symmetric Mean Absolute Percentage Error(SMAPE) which is a quite commonly used measure in practice.

$$SMAPE = \frac{100\%}{N} \sum_{k=1}^{N} \frac{\mid F_k - Y_k \mid}{(\mid Y_k \mid + \mid F_k \mid)/2}$$

where, N is the number of time series

F_k is the forecast

Y_k is the actual values

The SMAPE error is based on percentage errors. Hyndman and Koehler (2006) states that the SMAPE error is unstable for values that are near 0. However, this measure was used in the M4 forecasting competition and is a widely used measure in general practice as well and hence it is used in this research as well.

3.7 Software implementation

As part of this research, an implementation of WaveNet architecture suitable for a time series data was done using Keras (Chollet et al. (2015)) which is named *waveCast* from here on. Another implementation done by Toby Bischoff (2019), called *Deep4Cast*, was used to do an ablation study. *Deep4Cast* contains an implementation in PyTorch (Paszke et al. (2019)) which uses a modified version of WaveNet network which adds a concrete dropout layer (Gal et al. (2017)) and a student-t distribution as the loss function.

3.8 Experimental flow

Based on the methodology described above, this section gives a brief overview on the structure of the code implementation.

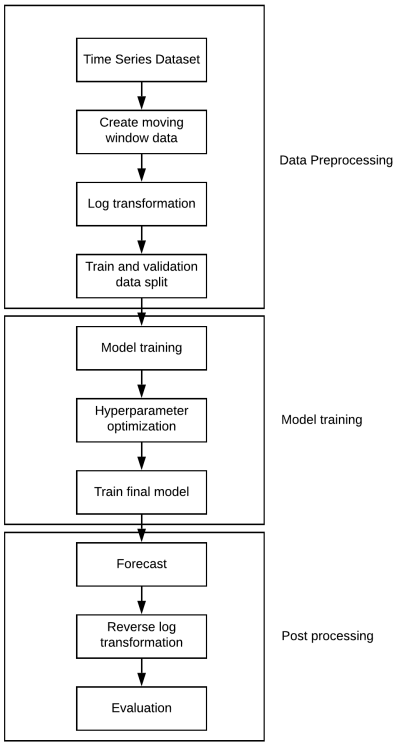

Figure 3.4: Experiment workflow

As seen in Figure 3.4, the entire architecture is divided into 3 pieces, Data preprocessing, model training and the post-processing of the data. The data preprocessing as explained in section 3.3 includes generating the moving window data from the original time series, log transformation and then the split to training and testing data. The model training component as described in section 3.4 includes the tuning of the hyperparameter and getting the final model trained. The last block is post-processing which deals with generating the forecasts for the actual test data and reversing the preprocessing that was done during the training phase.

3.9 Ablation Study

Ablation studies are a widely used concept in neuroscience to better understand complex biological systems. They have recently been used in the field of machine learning as well to understand how an Artificial Neural Network works and what are the key contributions of each component of the model (Meyes et al. (2019), Mudrakarta et al. (2018)).

3.9.1 Deep4Cast

As mentioned above, *Deep4Cast* is an open-source implementation of the WaveNet architecture done in python using the PyTorch framework. This architecture also includes some variations like the concrete dropout layer and changing the loss function to a student-t distribution. To the best of our knowledge, this was presented at the International Symposium on Forecasting 2019 and has shown to work very well on the M4 datasets. As part of this research, we compare the original method of Toby Bischoff (2019) on some of the datasets and also do some ablation study as explained in the below sections.

Deep4Cast library also uses a moving window approach based on a given time series dataset as explained in section 3.3.1. To stabilise the variance, a log transformation was used. This was also done to make sure that similar preprocessing techniques are applied to both *waveCast* and *Deep4Cast*. Once a model is built, each data point is forecasted 100 times using a sample from a student-t distribution and the mean of the forecast is taken as the final forecast. This again is similar to *waveCast* approach where for each data point multiple forecasts are taken based on the dropout probability and then they are combined together.

3.9.2 Removal of Concrete Dropout layer

Concrete dropout layer help in calibrating the dropout probability rates based on continuously relaxing the masks of the dropout layer (Gal et al. (2017)). In this part of the study, various experiments are done with the concrete dropout layers in the network. Firstly, the dropout layers were completely removed from the network to see how it affects the performance of the model. Secondly, each of the concrete dropout layer is replaced with a normal dropout layer having a fixed dropout rate. This change would help in understanding the effect that a concrete dropout has as against a normal dropout.

Chapter 4

Results

In this section, a complete analysis is provided based on the experiments conducted. As mentioned in the previous sections, the datasets used in these experiments vary quite a lot and their characteristics can be seen in Table 3.1. Based on the variety, it is reasonable to argue that the experiments can be used for some general conclusions that we make in the coming section. The results based on mean SMAPE metric as described in section 3.6.2 can be seen in Table 4.1.

Dataset Name	**ets**	**auto.arima**	**wavecast**	**Deep4Cast**
M4-yearly	16.396	15.1529	15.9	**14.38**
M4-quaterly	10.97	10.57	11.28	**10.19**
M4-monthly	14.83	14.01	15.59	**12.93**
M4-weekly	8.72	8.59	16.91	**6.42**
M4-daily	3.05	3.19	3.69	**2.91**
M4-hourly	16.396	15.1529	15.9	**14.38**
NN5-daily	**21.57**	24.98	24.3	22.1
NN5-weekly	12.29	13.54	**11.65**	12.24
Wikipedia Web traffic - Weekly	33.86	30.38	**27.09**	27.7

Table 4.1: Mean SMAPE results

Based on the Table 4.1 we see that the *Deep4Cast* method has the lowest error in all the M4 datasets. We also notice that *waveCast* method outperforms all the other methods in the NN5-weekly and the Wikipedia web traffic weekly dataset. One thing to note here is that, *Deep4Cast* manages to outperform the standard benchmark results in all the occasions except for the NN5-daily dataset. We further do an ablation study as described in section 3.9 to understand what parts of the *Deep4Cast* contribute more towards a better forecast. The results of the ablation study are discussed in section 4.3.

4.1 Effect of input windows

Its been a topic of debate since the evolution of neural networks that how much data is needed to achieve a good performance on a neural network. For example, Cho et al. (2015) talks about how much data is needed for a model to train on medical images to get high accuracy. Similarly, it is important to understand for a time series forecasting model as well that how much data does it need. Since this research uses an approach to convert the original time series to multiple series using

a moving window approach as explained in section 3.3.1, it would be important to understand how much should be the size of the input windows. Bandara et al. (2017) proposes a heuristic-based approach of taking the window size to be 1.25 times of the forecast horizon. Hewamalage et al. (2019) also discusses the results of varying the input window size of the input data fed into the RNN architecture built. It was concluded by Hewamalage et al. (2019) that larger input window sizes are better than the smaller one. A key thing to notice in this research is that the data before feeding into the network was not deseasonalised, meaning the seasonality of the data was fed to the network as is. This is one way that this approach is different from Bandara et al. (2017), Hewamalage et al. (2019). Since the data was not deseasonalised, longer input windows were experimented with to the model to capture the seasonality and forecast accordingly.

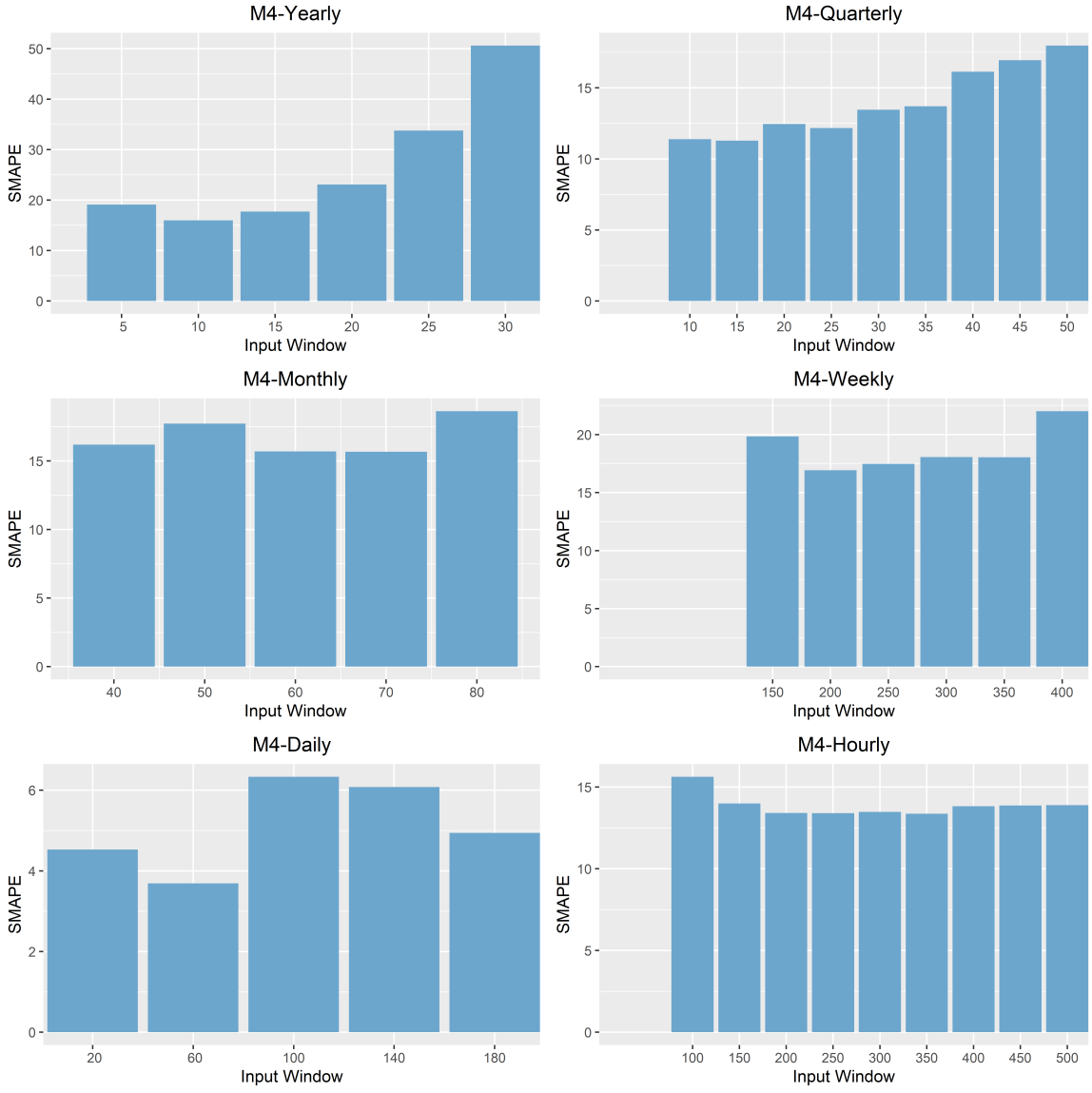

Figure 4.1: Effect of Input window on a WaveNet model

The question of how much input window remains unanswered because it would mostly depend on the dataset used but with the following experiments we can derive the conclusions for the M4

dataset. Figure 4.1 shows the result of varying input window size on all the six different M4 dataset category with respect to mean SMAPE. Yearly data does not have a seasonality and we see that the lowest SMAPE is achieved with an input window of 10. We also notice that the SMAPE increases as we increase the window size indicating that more data is not always better. For the rest of the datasets, since they can be seasonal, an optimal input window size can be related to the number of seasonal patterns needed from the series.

For the M4-quarterly dataset, the lowest SMAPE is achieved at a window size of 15, which would be equivalent to approximately 4 seasons. For the M4-monthly dataset, the minimum value is very close for the input window 60 and 70 which is equivalent to 5-6 seasons. For the daily dataset, the minimum value is at a window size of 60 which, if the weekly seasonality is considered, would be equivalent to almost 8-9 cycles. The M4-hourly dataset can have multiple seasonality, daily and weekly. It has an almost flat line and it is difficult to make a conclusion on the minimum value. The differences is very marginal but if two minimum SMAPE values are considered, they occur at 200 and 350 which can be interpreted to be approximately 8 weekly cycles and 2 yearly cycles respectively.

The above results do not give a concrete answer on how much seasonality to be considered, but it does show that the WaveNet networks varies based on different window sizes. Since the data was not deseasonalised, it was important to feed more seasonal cycles so that the model captures the seasonality and this experiment was needed to understand how many cycles are needed.

4.2 Effect of dilation factor

Once an appropriate input window has been decided upon, the dilation factor decides how much visibility does a particular convolutional filter has which is also called as the receptive field size. An experiment was conducted having the input window fixed and varying the dilation factor from 4 and increasing it by 2 times at every step till 64. The results can be seen as shown in Table 4.2

Dataset Name	Dilation 4	Dilation 8	Dilation 16	Dilation 32	Dilation 64
M4-monthly	14.948902	14.490526	14.051151	13.216181	13.105919
M4-weekly	8.9367075	9.0840845	9.215055	8.480958	8.149735
M4-daily	2.9386694	3.013745	2.9822783	2.9617622	2.9690037
M4-hourly	36.12349	31.542498	22.730383	16.765768	14.658653

Table 4.2: Effect of dilation factor in *Deep4Cast*

We see based on the results that, as the dilation factor increases, the error reduces for the M4-monthly, M4-weekly and M4-hourly datasets. It has a minimal impact on the M4-daily dataset. The Friedman test for statistical significance gives a p-value of 0.06 which is almost on the boundary of being significant. If we look at the table, we can clearly conclude that there is a significant difference with varying dilation factor.

4.3 Ablation study results

As seen in Table 4.1, the *Deep4Cast* implementation manages to outperform the benchmark methods in all the datasets. This experiment particularly does an ablation study by removing specific parts of the model and observe the behaviour on the SMAPE values.

Figure 4.2 shows the result of two different settings done on the original *Deep4Cast* method. In the first case, the concrete dropout layer is completely removed from the network. In the second

Dataset Name	Original Deep4Cast	Without dropout	Fixed dropout
M4-yearly	14.38	**13.72**	16.3
M4-quaterly	10.19	**9.97**	11.05
M4-monthly	**12.93**	13.8	15.9
M4-weekly	**6.424**	6.791	8.299
M4-hourly	14.38	**14.36**	16.2

Table 4.3: Effect of concrete dropout layer in *Deep4Cast*

case, the concrete dropout layer is replaced with a fixed dropout mask. We see in the Figure 4.2 that, when the dropout layer is completely removed, in some cases it helps the model and in others it does not. For example, for M4-Yearly and M4-Quarterly datasets, the SMAPE value reduces by removing the dropout values whereas for the M4-Monthly and M4-weekly datasets the SMAPE increases. Friedman test for statistical significance comparing the mean SMAPE values gives an overall p-value of 0.75. The test indicates that the difference is not statistically significant which is also evident in the Figure 4.2.

When the concrete dropout layer is replaced by a fixed dropout mask, the SMAPE increases in all the cases as seen in the third bar in each graph of Figure 4.2. The Friedman test for statistical significance comparing the mean SMAPE values gives an overall p-value of 0.022. This indicates that the differences are statistically significant. This means that adding a fixed dropout mask to a WaveNet network hampers its performance. The detailed result of this experiment can be seen in Table 4.3.

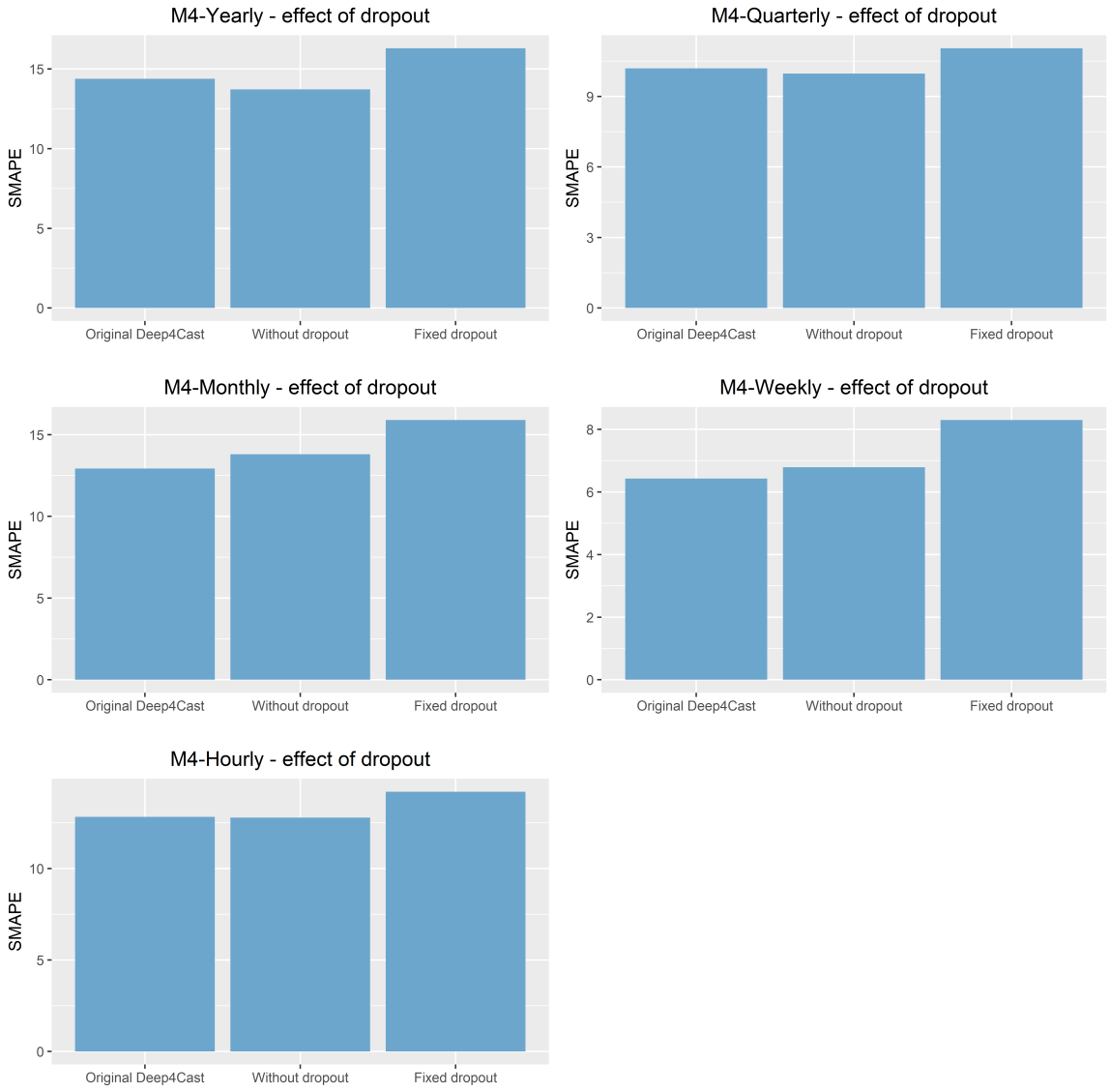

Figure 4.2: Effect of concrete dropout layer in *Deep4Cast*

Chapter 5

Discussion

Based on the research questions defined in the section 1.4, various experiments were conducted and the results were presented in chapter 4. In this chapter, we discuss the findings from our experiments and finally discuss the future work.

5.1 Findings

It is seen that the WaveNet models can be used for time series forecasting but there is no one model that suits all the needs. We see that the *Deep4Cast* works quite well on the M4 datasets. In fact, its performance on the weekly model is outperforming the best result of M4 published at Makridakis (2018). But *Deep4Cast* does fail to beat the plain WaveNet model *wavecast* that was built as part of this research in the NN5-weekly and the Wikipedia weekly datasets.

We also see that since the data fed into the WaveNet models was not deseasonalised, it depends a lot on the input window that is fed into the model. The input window needs to be tuned well based on the dataset because we see from the experiments that if the optimal value is not chosen, the results are affected significantly. It was also found that once an optimal value of the input window has been selected, the change in dilation factor also has a huge impact on the performance of the model as it determines the visibility of each of the convolutional filters.

One of the key take away from this research was to understand the effect of the additional modifications done on the WaveNet network. As we saw, the concrete dropout layer added in the *Deep4Cast* network makes a significant difference in the performance as against to a fixed dropout mask. The results did not change significantly if the dropout layer was completely removed.

5.2 Future Work

This particular study is limited to univariate time series and hence multivariate time series forecasting using WaveNet networks would certainly be a future direction to progress on. Since in the era of big data, a lot of series evolve together in groups, and hence using a clustering approach before forecasting them could be a good option. Work done by Bandara et al. (2017) is a great step in that direction.

It could also be useful to cluster the series based on their seasonalities and create a model per group because since the data fed in the models were not deseasonalised, it would be difficult for the model to adjust the weights for a seasonal and a non-seasonal series together. Work done by Lai et al. (2017) is a great step in the direction of combining the RNN and convolutional network to model the long term and short term dependencies.

On many occasions, the lack of dataset result in poor forecasts. Transfer learning is a great way to make use of other datasets and solve the problem of insufficient training data. Experimenting with a WaveNet model could be a possible research to see how well they compare against the RNN networks.

Most of the literature in the global forecasting model tries to build one single model for the whole dataset. It could be useful to decompose the series in some way and build sub-global models and then calibrate the result based on a local attention scheme. Work done by Sen et al. (2019) is a great start in that direction and can be used as one way to proceed.

Chapter 6

Conclusion

It's highly critical for businesses to get an accurate forecast as it is vital for their planning. RNNs have dominated this field with their state-of-the-art results. In this research, two different implementations of WaveNet models were explored in-depth on a variety of datasets and compared against the benchmark results. An ablation study was done as well to understand the improvements that can be done on a plain WaveNet and how much do all these improvements contribute towards the forecasting accuracy.

Based on the results, we observed that both the implementations, *Deep4Cast* and *wavecast* give competitive results on the datasets. NN5-daily dataset was the only dataset where both the implementation could not outperform the benchmark methods, which makes a strong statement that sometimes simple methods work better than complicated ones. We also observed that adding variation like concrete dropout layers to the WaveNet networks have a significant impact on the performance of the model. The size of input windows of the WaveNet model also need to be tuned well based on the datasets.

As part of this work, we conclude that WaveNet models work well on time series data and are a viable choice for forecasting. Certain variations on the architecture, like the concrete dropout layer, boost their performance and it would be good to explore what other variations help a forecasting model in the future.

Bibliography

Bandara, K., Bergmeir, C., and Smyl, S. (2017). Forecasting Across Time Series Databases using Long Short-Term Memory Networks on Groups of Similar Series. *CoRR*.

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, B., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Callot, L., and Januschowski, T. (2020). Neural forecasting: Introduction and literature overview.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 2546–2554, Red Hook, NY, USA. Curran Associates Inc.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2012a). hyperopt. `https://github.com/hyperopt/hyperopt`. Accessed: 2020-05-02.

Bergstra, J., Yamins, D., and Cox, D. D. (2012b). Making a science of model search. *CoRR*, abs/1209.5111.

Borovykh, A., Bohte, S., and Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Box, G., Jenkins, G., Reinsel, G., and Ljung, G. (2015). *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley.

Box, G. E. P. and Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243.

Chauhan, A., Prasad, A., Gupta, P., Prashanth Reddy, A., and Kumar Saini, S. (2020). Time series forecasting for cold-start items by learning from related items using memory networks. In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 120–121, New York, NY, USA. Association for Computing Machinery.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs.

Cho, J., Lee, K., Shin, E., Choy, G., and Do, S. (2015). How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?

Chollet, F. et al. (2015). Keras. `https://github.com/fchollet/keras`.

Combes, J., Grossmann, A., and Tchamitchian, P. (1990). *Wavelets: Time-frequency Methods and Phase Space : Proceedings of the International Conference, Marseille, France, December 14-18, 1987*. Inverse Pro. Springer-Verlag.

contributors, W. (2017). Causal filter. https://en.wikipedia.org/wiki/Causal_filter.

Crone, S. F. (2008). Nn5 competition. http://www.neural-forecasting-competition.com/NN5/. Accessed: 2020-05-02.

Daoud, M., Naqvi, S. K., and Siddiqi, T. (2015). An item-oriented algorithm on cold-start problem in recommendation system. *International Journal of Computer Applications*, 116:19–24.

eResearch Centre (2020). M3 user guide. https://docs.massive.org.au/index.html. Accessed: 2020-06-02.

Gal, Y. and Ghahramani, Z. (2015). Dropout as a bayesian approximation: Representing model uncertainty in deep learning.

Gal, Y., Hron, J., and Kendall, A. (2017). Concrete dropout. In *Advances in Neural Information Processing Systems*.

Gardner Jr., E. S. (1985). Exponential smoothing: The state of the art. *Journal of Forecasting*, 4(1):1–28.

Ghosh, T. (2017). 1 dimensional convolutional neural networks. https://www.quora.com/What-does-it-mean-by-1D-convolutional-neural-network. Accessed: 2019-10-21.

Google (2017). Wikipedia web traffic data. https://www.kaggle.com/c/web-traffic-time-series-forecasting. Accessed: 2020-05-02.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Hewamalage, H., Bergmeir, C., and Bandara, K. (2019). Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions.

Huszár, F. (2016). Dilated convolutions and kronecker factored convolutions. https://www.inference.vc/dilated-convolutions-and-kronecker-factorisation/.

Hyndman, R. and Koehler, A. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22:679–688.

Hyndman, R. J. (2015). anomalous-acm. https://github.com/robjhyndman/anomalous-acm.

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*.

Hyndman, R. J. and Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*.

Ince, T., Kiranyaz, S., Eren, L., Askar, M., and Gabbouj, M. (2016). Real-time motor fault detection by 1-d convolutional neural networks. *IEEE Transactions on Industrial Electronics*, 63:7067–7075.

ITU-T (2017). G.711 : Pulse code modulation (pcm) of voice frequencies. https://www.itu.int/rec/T-REC-G.711-198811-I/en.

Jordan, J. (2017). Convolutional neural networks. https://www.jeremyjordan.me/convolutional-neural-networks/.

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., and Inman, D. J. (2019). 1d convolutional neural networks and applications: A survey.

Kiranyaz, S., Ince, T., and Gabbouj, M. (2016). Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks. *IEEE Transactions on Biomedical Engineering*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.

Labach, A., Salehinejad, H., and Valaee, S. (2019). Survey of dropout methods for deep neural networks.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. (2017). Modeling long- and short-term temporal patterns with deep neural networks.

Makridakis, S. (2018). M4 competition. `https://github.com/M4Competition/M4-methods`. Accessed: 2019-09-02.

Meyes, R., Lu, M., de Puiseau, C. W., and Meisen, T. (2019). Ablation studies in artificial neural networks.

Moritz, S., Sardá, A., Bartz-Beielstein, T., Zaefferer, M., and Stork, J. (2015). Comparison of different methods for univariate time series imputation in r.

Mudrakarta, P. K., Taly, A., Sundararajan, M., and Dhamdhere, K. (2018). Did the model understand the question?

Mukherjee, S., Shankar, D., Ghosh, A., Tathawadekar, N., Kompalli, P., Sarawagi, S., and Chaudhury, K. (2018). Armdn: Associative and recurrent mixture density networks for eretail demand forecasting.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA. Omnipress.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

R Development Core Team, R. (2011). *R: A Language and Environment for Statistical Computing*.

Salinas, D., Flunkert, V., and Gasthaus, J. (2017). DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks.

Sen, R., Yu, H.-F., and Dhillon, I. (2019). Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting.

Simonyan, K., Dieleman, S., Senior, A., and Graves, A. (2016). WaveNet. *arXiv preprint arXiv:1609.03499v2*.

Skorski, M., Temperoni, A., and Theobald, M. (2020). Revisiting initialization of neural networks.

Smyl, S. (2018). Exponential smoothing - recurrent neural network (es-rnn) hybrid models. https://github.com/M4Competition/M4-methods/blob/master/118%20-%20slaweks17/ES_RNN_SlawekSmyl.pdf. Accessed: 2019-09-02.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Suilin, A. (2017). kaggle-web-traffic. https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md.

Sultana, F., Sufian, A., and Dutta, P. (2019). Advancements in image classification using convolutional neural network. *Proceedings - 2018 4th IEEE International Conference on Research in Computational Intelligence and Communication Networks, ICRCICN 2018*, pages 122–129.

Taieb, S. B., Bontempi, G., Atiya, A., and Sorjamaa, A. (2011). A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition.

Taieb, S. B., Bontempi, G., Atiya, A. F., and Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert systems with applications*, 39(8):7067–7083.

Taylor, S. J. and Letham, B. (2017). Forecasting at scale. *PeerJ PrePrints*, 5:e3190.

Toby Bischoff, Austin Gross, K. T. (2019). deep4cast. https://github.com/MSRDL/Deep4Cast. Accessed: 2019-12-02.

Trapero, J. R., Kourentzes, N., and Fildes, R. (2015). On the identification of sales forecasting models in the presence of promotions. *Journal of the Operational Research Society*, 66(2):299–307.

Van Den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *33rd International Conference on Machine Learning, ICML 2016*.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016). Conditional image generation with pixelcnn decoders.

Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions.

Yu, T. and Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications.