Tampere University

Eetu Pulkkinen

FORECASTING EMERGENCY DEPARTMENT ARRIVALS

With Neural Networks

Bachelor of Science Thesis
Faculty of Information Technology and Communication Sciences
Examiners: MsC Francesco Lomio
Prof. Heikki Huttunen
December 2020

ABSTRACT

Eetu Pulkkinen: Forecasting Emergency Department Arrivals
Bachelor of Science Thesis
Tampere University
Information technology
December 2020

Emergency departments often suffer from chronic overloading as well as seasonal spikes in number of arrivals. In this study three different Deep learning based models are used to try to predict the next days arrivals to the Pirkanmaa Hospital Districts emergency department. First one is a simple Recurrent Neural Network (RNN) which uses Long short-term memory (LSTM) cells for temporal relations. The latter two use a newly developed Temporal Fusion Transformer (TFT) architecture specifically made for multi-horizon time series forecasting.

The dataset used to train the three different models contains a mix of variables thought to have effect on the end results predictions. The dataset is split into a training set and a validation set. The models learn from the data in the training set and their performance is measured using the validation set. The LSTM model is implemented using TensorFlow python library and the TFT model with PyTorch Forecasting.

The results show that all three models perform better than the measured baseline. Although much faster to train, The LSTM model falls behind of the TFT architecture in terms of prediction accuracy. From the two TFT based models the hourly frequency model performs the best as it has access to more data. The TFT model achieves this performance advantage due to its more sophisticated network architecture and from the usage of temporal self-attention layers for learning long term dependencies. Due to the nature of its architecture the TFT model has better interpretability. From the Variable Selection Network we can deduce which variables contribute the most to the end results prediction and possibly even remove those that do not bring much value. The weights from the self-attention layer tells us which parts of the time series the model is focusing on.

The use of machine learning, especially Neural Networks, is still quite a new phenomena in applications like in this study. The results from this study are promising and indicate that further research of the subject matter is warranted. Experimenting with different datasets and a wider range of ML methods could shed more light into the future of ED forecasting.

Keywords: machine learning, deep learning, Neural Network, time series forecasting, Temporal Fusion Transformer

TIIVISTELMÄ

Eetu Pulkkinen: Ensiapupoliklinikalle saapuvien aikasarjaennustaminen
Kandidaatintutkinto
Tampereen yliopisto
Tietotekniikka
Joulukuu 2020

Ensiapupoliklinikat kärsivät usein kroonisesta ylikuormituksesta sekä kausittaisista piikeistä ensiapuun saapuvien määrissä. Tässä tutkimuksessa käytetään kolmea erilaista Syväoppimiseen pohjautuvaa mallia, joilla ennustetaan seuraavan päivän saapuvien määrää Pirkanmaan sairaanhoitopiirin ensiapupoliklinikalle. Ensimmäinen malleista on yksinkertainen takaisinkytkeytyvä neuroverkko (engl. Recurrent Neural Network, RNN), joka käyttää pitkä-lyhytkestomuistisoluja (engl. long short-term memory, LSTM) aikasuhteisiin. Jälkimmäiset kaksi mallia käyttävät vastikään keksittyä Temporal Fusion Transformer (TFT) -arkkitehtuuria, joka on kehitetty varta vasten aikasarjaennustamiseen.

Koneoppimismallien kouluttamiseen käytetty tietoaineisto sisältää joukon muuttujia, joilla uskotaan olevan merkitystä loppuennustuksien suhteen. Tietoaineisto jaetaan erilliseen mallien kouluttamisaineistoon ja validointiaineistoon. Mallit oppivat kouluttamisaineistosta ja niiden suorituskykyä mitataan validointiaineiston avulla. Takaisinkytkeytyvä neuroverkko toteutettiin TensorFlow Python kirjaston avulla ja TFT taas PyTorch Forecasting kirjastolla.

Tulokset osoittavat, että kaikki kolme mallia tuottavat parempia tuloksia vertailumalliin verrattuna. Takaisinkytkeytyvä neuroverkko on nopea kouluttaa, mutta se häviää ennustustarkkuudessa TFT -mallille. Kahdesta TFT -arkkitehtuuria käytettävästä mallista tuntitarkkuudella toimiva saavuttaa parhaan ennustustarkkuuden, johtuen sen saamasta suuremmasta aineistosta. TFT -mallit saavuttavat paremman ennustustarkkuuden arkkitehtuurin hienostuneempien ominaisuuksien myötä. Yksi tällainen on TFT:n Temporal Self-Attention -kerrokset, joiden avulla malli oppii pitkäaikaisia aikasuhteita. Hienostuneemman arkkitehtuurinsa ansiosta TFT -mallia on helpompi tulkita. Muuttujavalintaverkosta voidaan päätellä, mitkä tietoaineiston muuttujat vaikuttavat eniten loppuennustukseen. Sen myötä lopputuloksen kannalta turhia muuttujia voidaan jopa poistaa aineistosta. Self-Attention -kerroksen painoista voidaan myös tulkita, mihinkä kohtiin aikasarjassa mallin huomio kohdentuu.

Koneoppimisen käyttö, varsinkin Neuroverkkojen osalta, on varsin uusi ilmiö tämän tutkimuksen kaltaisissa sovelluksissa. Tämän tutkimuksen tulokset ovat lupaavia ja osoittavat, että jatkotutkimus aiheesta on perusteltua. Kokeet erilaisilla aineistoilla ja laajemmalla valikoimalla koneoppimismetodeja voisi tuoda enemmän valoa ensiapuennusteiden tulevaisuuten.

Avainsanat: koneoppiminen, syväoppiminen, neuroverkko, aikasarjaennustaminen, Temporal Fusion Transformer

PREFACE

This thesis was done as part of an ongoing multidisciplinary project carried out by researchers from Tampere University, Pirkanmaa Hospital District and Kanta-Häme Hospital District. I would like to thank my supervisors Francesco Lomio and Heikki Huttunen for their advice and support for the thesis. Special thanks to (MD) Jalmari Tuominen and (PhD) Antti Roine (from Tampere University) for their continued interest and guidance. I would also like to thank (Prof) Niku Oksala and (PhD) Satu-Liisa Pauniaho (from Pirkanmaa Hospital District) for their effort and involvement in producing the dataset used in this study.

Tampereella, 18th December 2020

Eetu Pulkkinen

CONTENTS

1 INTRODUCTION

Emergency Departments serve a crucial purpose in society providing critical care to people in need. Being able to provide high quality care to patients presenting with different conditions varying from non-severe to life threatening in a timely manner is at the utmost importance. However, most emergency departments all over the world suffer from chronic overcrowding as well as regular and irregular seasonal spikes in number of arrivals. The objective of this thesis is to predict the number of visits to emergency departments using data from the Pirkanmaa hospital district (PHSP).

The PHSP dataset contains a comprehensive mix of inputs from exogenous time series that are only observed in the past (e.g. weather) to known future inputs (e.g. day of the week). Time series prediction can be done with numerous different methods from the more traditional statistical methods to newer, often more complex, Deep learning solutions. This thesis proposes the usage of Temporal Fusion Transformer (TFT) architecture which uses Deep learning for local processing and self-attention layers for long term dependencies.

The research questions can be formulated as three separate questions.

1. To what accuracy can future arrivals be predicted using neural networks?
2. Does a more complex predefined network architecture provide better results than a simpler model?
3. Which input features are the most important for the end result prediction?

This thesis is structured followingly. Chapter 2 presents the theoretical background behind the study and reviews other publications related to time series forecasting and emergency department forecasting. The third chapter goes over the methodology and machine learning details as well as the metrics used to measure model performance. The fourth chapter details the experiments conducted, reviews the dataset used and presents the used hyperparameter optimization methods. Chapter 5 presents the results and findings from the experiments. The final chapter summarizes the results and their meaning for the study.

2 BACKGROUND

In the classical statistical handling of time series data making predictions about the future is called extrapolation. In more modern context it is often referred as time series forecasting. Time series forecasting models fit on historical data and use it to predict future observations. The crucial component in time series data is an explicit order of dependence between observations: a time dimension. A time series can be decomposed into 4 constituent parts described in [1] as:

1. **Level**. "The baseline value for the series if it were a straight line."
2. **Trend**. "The optional and often linear increasing or decreasing behavior of the series over time."
3. **Seasonality**. "The optional repeating patterns or cycles of behavior over time."
4. **Noise**. "The optional variability in the observations that cannot be explained by the model."

Not all time series have trend or seasonality. Most time series have noise and all have a level. [1] A time series can be though of as a combination of these 4 constituent components. As an example a model can be formed as follows:

$$\mathbf{y} = level + trend + seasonality + noise, \tag{2.1}$$

where **y** is the output of the model. An important part of time series are the trends and seasonal variations. Another important feature is that often times observations close together in time tend to be correlated. [2]

Usually time series forecasts have access to multiple sources of data as shown in figure 2.1. Known inputs are known information about the future (e.g. dates of upcoming holidays). Observed inputs are other exogenous time series (e.g. past weather). Static covariates do not have a time component and instead denote some constant (e.g. location). These different inputs are suspected to have impact on the end results. Later in the process it can sometimes be determined that a certain input in fact does not contribute to the end result in any meaningful way. In this case often times the input in question is better left out of the equation as unhelpful data creates more noise and ultimately hinders the end results. The prediction model is often application specific as different models

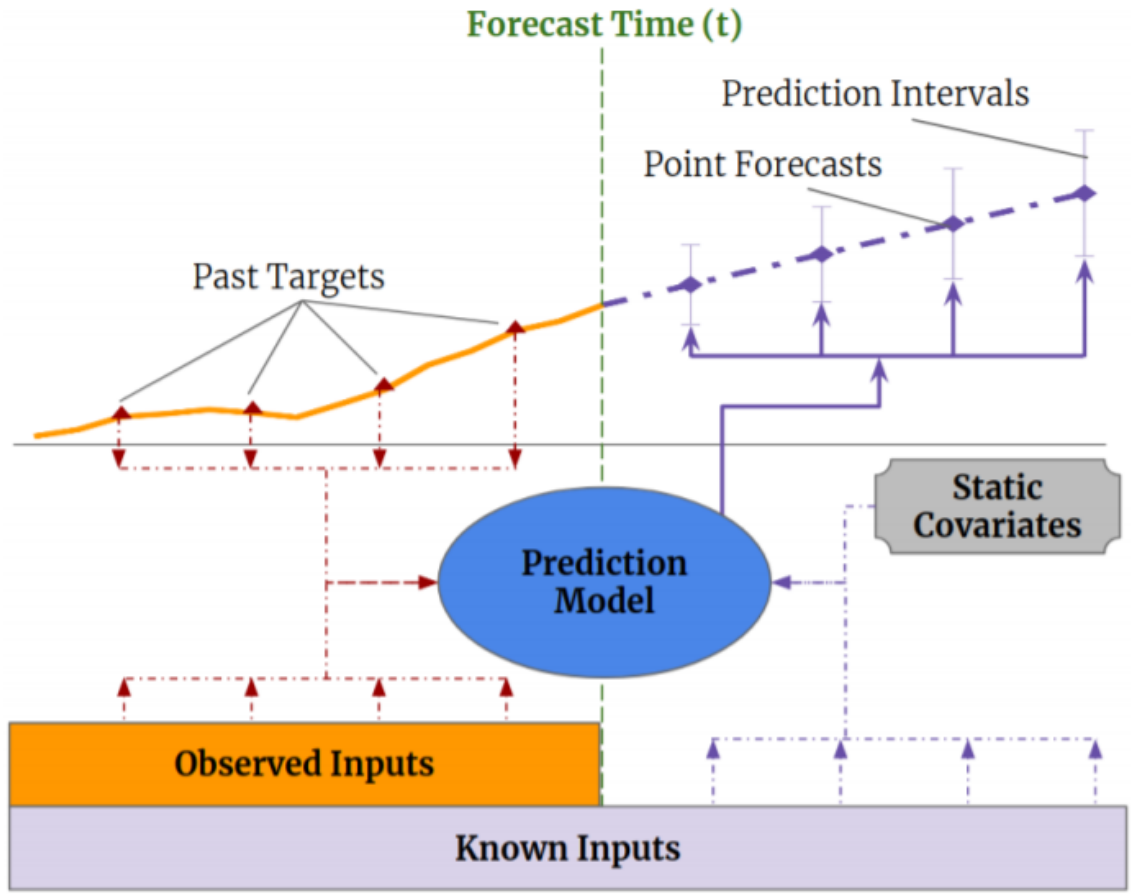

Figure 2.1. *Multi-horizon forecasting illustration with multiple data sources [3].*

work well with different problems.

2.1 Machine learning

Machine learning is a subset of artificial intelligence where algorithms learn from data and improve their performance in the tasks they are designed for. In the context of this thesis the object is to most accurately predict the future arrivals to the emergency department using a mix of machine learning techniques. A wide array of different methodologies exist for time series forecasting. The classical methods use a more of a statistical approach while the newer machine learning methods often rely on deep learning. It is debatable which of these methods suits time series forecasting the best as results and opinions vary from study to study. A consistent finding among different forecasting competitions is that combining these different methodologies yields better results instead of using a singular one. In general combining several methods/models reduces the amount of random errors resulting in more accurate forecasts. [4]

The models used in this study are based on Deep learning. One of the models is a basic Recurrent Neural Network (RNN) which uses long short-term memory (LSTM) cells for temporal relations. The more advanced Machine learning model used in this study

expands this further and combines the LSTM architecture with a wide range of machine learning methods to make predictions more accurate. The models used are explained more thoroughly in the methodologies chapter.

2.2 Related work

Time series forecasting has found variety of use cases in many fields. Stock market prediction [5][6] in its many forms continues to be of interest to many for its lucrative applications. Other use case examples are Web traffic prediction [7], health care expenditure prediction [8] as well as earthquake prediction [9]. Different forecasting methods are used for different purposes ranging from the tried and tested linear methods (e.g. ARIMA [10]) to the newer yet unproven non-linear deep learning solutions.

Even though there have been many studies in ED arrival prediction [11][12] and patient flow forecasting [13][14], opportunities remain for future improvement. In a more recent study (2019) by Whitt et. al. [15] 5 different prediction models were used including rolling averages, highly structured time series models and a neural network model. In this study they found that a SARIMAX model had the best predicting power.

Compared to Whitt et. al. the dataset used in this thesis contains many more exogenous variables which could help a neural network produce better results. The Neural Network in Whitt et. al. is simple multilayer Perceptron (MLP). Compared to the architecture used in this thesis it lacks many of the advanced features that could make the prediction prowess of a Deep Learning model the better solution. Whitt et. al. attributes the shortcomings of the Neural Network to low dimension of the problem and the sample size being small.

Many of the older studies did not have the computational power or the newly invented methodologies to consider a Neural Network as a viable solution. With the help of modern computational prowess (especially the use of GPUs and cloud computing) and the newly invented models, like the Temporal Fusion Transformer used in this thesis, could give merit to Deep Learning as a viable solution.

3 METHODOLOGY

Deep learning has been extensively used in many areas of machine learning like image recognition and natural language processing etc. However in the realm of time series forecasting Deep learning has not yet proved itself as the prevalent solution. In December 2019 a new type of architecture, called Temporal Fusion Transformer, was introduced by Lim et al.[16]

Temporal Fusion Transformer (TFT) is an attention based Deep Neural Network (DNN) architecture for time series forecasting. It enables high performance and provides new forms interpretability. TFT has the ability to learn both long- and short-term temporal relationships (e.g. seasonality) from both observed and known time-varying inputs. [3]

The architecture of the Temporal Fusion Transformer is relatively complex as seen in section 3.3. To begin understanding the network as a whole it needs to be broken down in to a more digestible format. However this thesis won't go into all the details and mathematics behind each component and merely presents more of a general overview of the architecture used.

3.1 Neural Networks

Neural networks, a beautiful biologically-inspired programming paradigm which enables a computer to learn from observational data [17].

The base unit of a Neural Network is called a perceptron. A perceptron can be thought as an artificial neuron. A perceptron takes several inputs and produces a single output. Each input has a *weight* which express the importance of the input in respect to the output. It is a basic mathematical model which can be thought of as a device that makes decision by weighing up evidence. The structure of a single neuron is presented in figure 3.1.

Many perceptons organized into layers formulate a Multilayer percepton (MLP) which is a class of feedforward Artificial Neural Network. Commonly MLPs are structured followingly:

1. Input layer
2. Hidden layer(s)
3. Output.

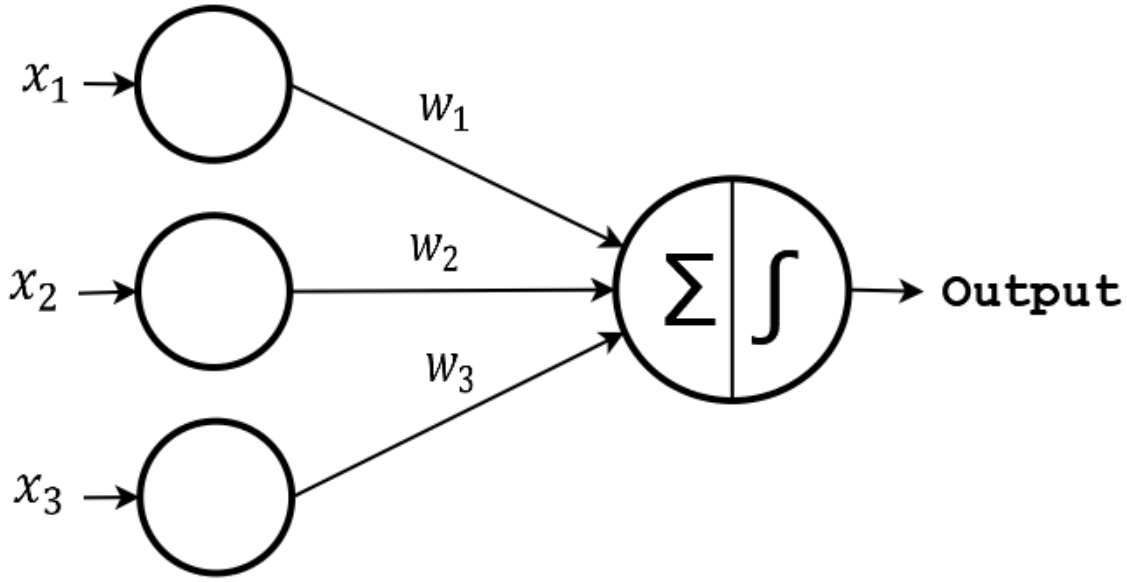

Figure 3.1. *Example of a perceptron - the base unit of a Neural Network. $\sum$ is the weighted sum of the inputs. This is then fed to the activation function $\int$ which decides the output for the perceptron.*

Example structure of a MLP network with 2 hidden layers is presented in figure 3.2. Here the weights can be thought as connections between the neurons. In each neuron these weighted inputs are summed and passed through an activation function (e.g. sigmoid). An activation function simply maps the inputs into an output of the neuron [18]. The activation function resides in every neuron except the input layer. In this example the final output layer is a singular neuron. In the context of time series forecasting this could be the prediction our model makes about the future.

A neural network needs to be trained. In this case a technique is used called supervised learning. In supervised learning the algorithm learns by example. This learning occurs in the perceptron by changing the weights after each piece of data is processed. Changing of the weights is based on the error produced when comparing the output to the expected result. [19]

3.2 Long short-term memory

Traditional Neural Networks suffer from a problem - they lack context and memory. This is addressed with a type of Neural Network called Recurrent Neural Network (RNN). RNNs allow information to persists across the network through the usage of loops. These cycles feed the network activations of previous time steps as context for the current time step. This internal state of the network can in principle hold long-term temporal contextual information. [20]

Long short-term memory (LSTM) is a type of a cell that can offer solutions to problems

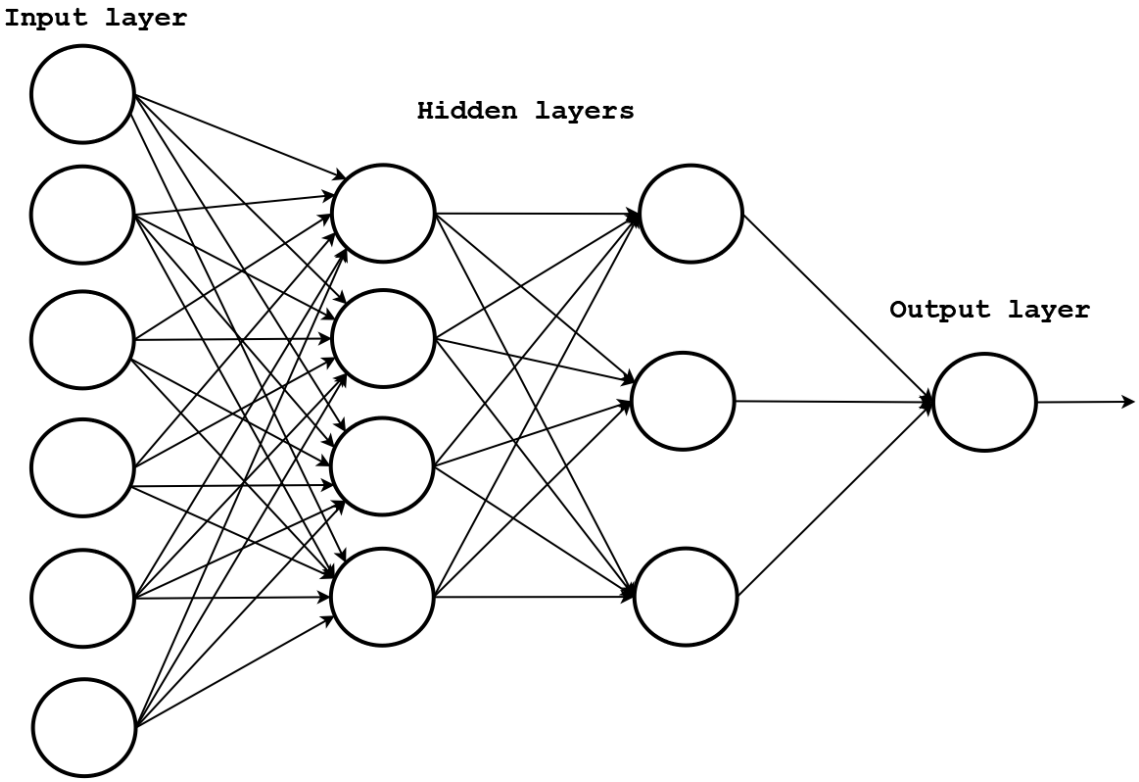

Figure 3.2. *Example structure of a Multilayer percepton with 2 hidden layers.*

a traditional RNN has. Standard RNNs often fail to take into account time steps that lag more than 5-10 steps behind [20]. A model which uses these LSTM cells does not suffer from this problem. Through its internal mechanisms the LSTM cell learns which past information it keeps and which it forgets. [21] LSTM is an important component of the TFT architecture as seen in figure 3.3. It allows the network to create and keep long and short-term dependencies in temporal data.

In the TFT architecture LSTMs are used a bit differently. Here the LSTM's job is to identify points of interest in relation to their surrounding values. The output of the LSTM encoder-decoder is not the "final answer". Instead it generates a set of uniform temporal features from each input and feeds them onwards into the temporal fusion decoder. The long-term dependencies between the different sets of the time series data is done with a multi-head attention layer.

3.3 Temporal Self-Attention

The self-attention layer allows TFT to detect long-range dependencies which are challenging for RNN based architectures. The attention layer prioritizes patterns which provide the most predicative power [22]. For example in this thesis the model comes to a conclusion that focusing on the peak hours of arrivals yields the best results. The self-attention layer has multiple heads where each of them can focus on learning different

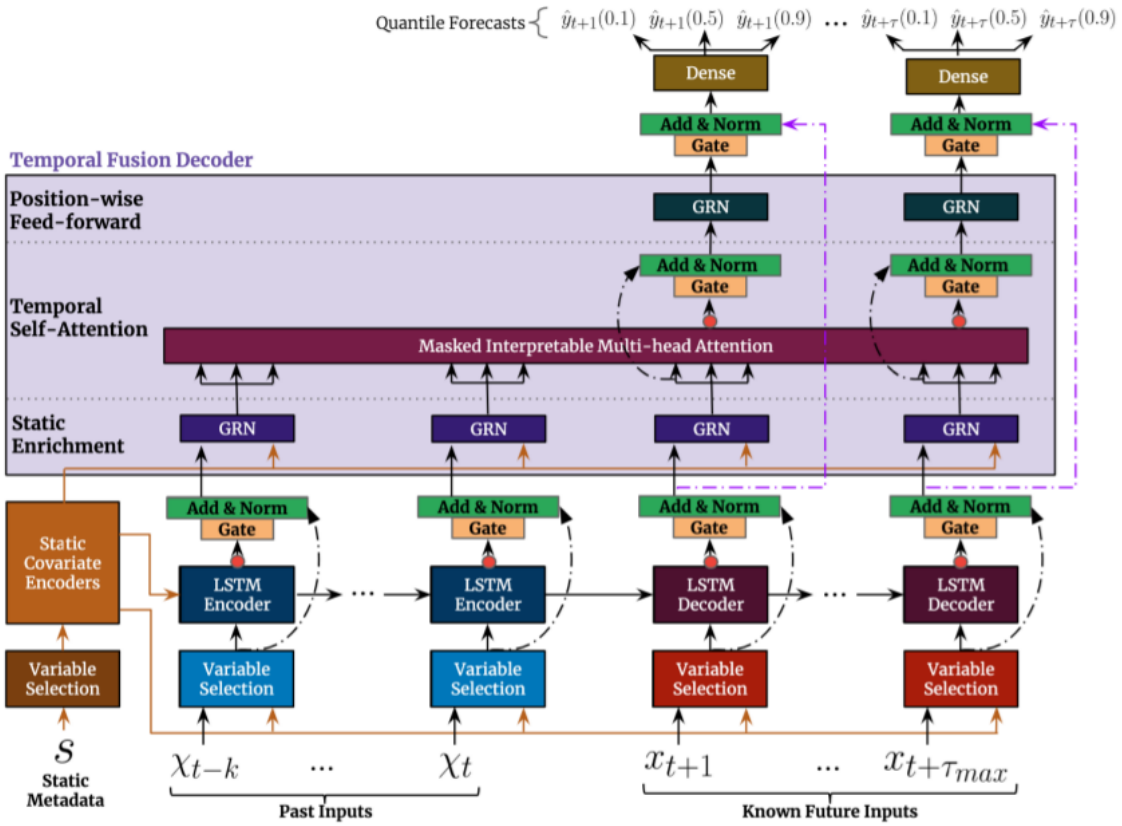

Figure 3.3. *High level architecture of Temporal Fusion Transformer (TFT). Originally proposed by Lim et al. Adapted from source [3].*

temporal patterns [3].

An attention mechanism gives the model new forms of interpretability. Input variables can be ranked according to their magnitude of attitude weights. [3] This allows us to compare the importance of inputs and detect underperforming variables. Removing them reduces noise which increases prediction accuracy and reduces computational complexity [23]. From the attention mechanism we can also see which parts of the times series the model is focusing on.

3.4 Other features of the Temporal Fusion Transformer

Gating mechanism is done with a Gated Residual Network (GRN) shown in figure 3.4. GRN provides flexibility to skip parts of the architecture that are not required for a given dataset. Often when a dataset is small or noisy this is necessary for increased performance. GRN also has a dropout mechanism where some of the neurons in the network are randomly ignored or "dropped". This is used to prevent overfitting.

Overfitting means that the model learns the training data too well and does not generalize resulting in poor out-of-sample performance in a "real world" scenario as new data is introduced. Often in medical applications (like this thesis) datasets tend to be on the

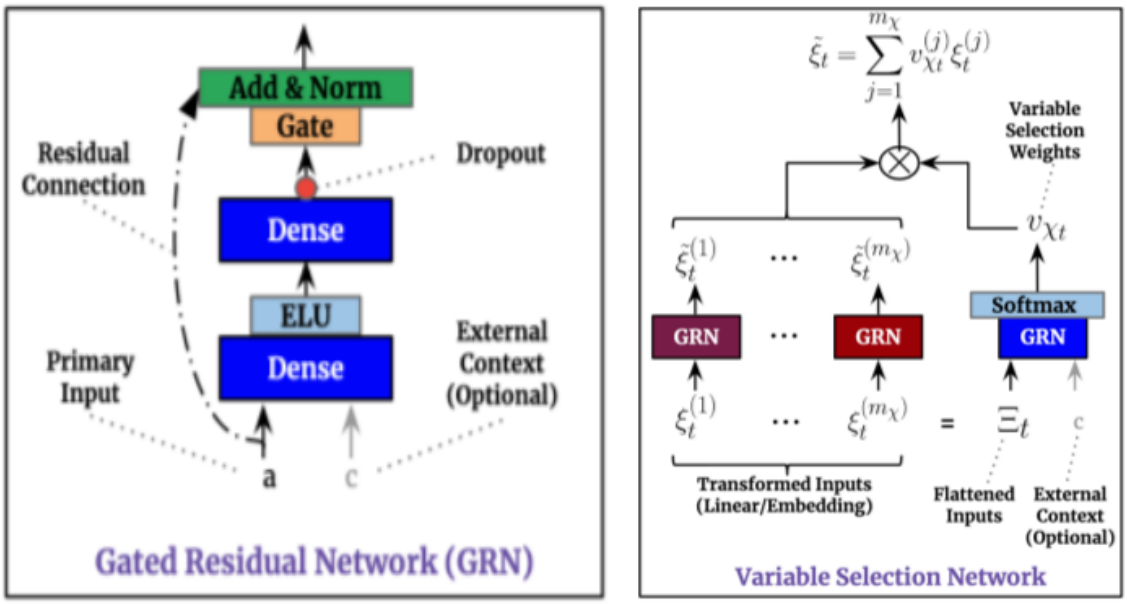

Figure 3.4. *Closer look into the Gated Residual Network (GRN) and the Variable Selection Network. Adapted from source [3].*

smaller side. Both GRN and dropout are needed to combat problems that occur when huge datasets are not available.

A variable selection network as seen in figure 3.4 allows the selection of relevant variables at each time step. Noisy or otherwise unnecessary inputs can be given less importance or removed all together from the equation. This also improves interpretability compared to other DNN solutions which often tend to be black boxes.

Static covariate encoders integrate static (i.e. Time-invariant) metadata into various different parts of the TFT architecture. Special GRN encoders produce context vectors c that are then wired into various locations in the TFT. For example static variables can provide context for better variable selection or improvements in local processing of temporal features.

3.5 Metrics

Mean absolute percentage error (MAPE) is a commonly used metric in time series forecasting. It measures the percentage error of the forecast in relation to the actual values. [24] MAPE is calculated as follows:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| * 100\%, \tag{3.1}$$

Where A_t is the actual value, F_t the prediction and n is the number of samples. Mean average absolute error (MAE) expresses average model prediction error in units [25]. This

is useful as the results are absolute differences and not percentages. MAE is defined as:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|F_t - A_t|. \tag{3.2}$$

Root mean squared error (RMSE) is similar to MAE where they are both positive and indifferent to the direction of errors. RMSE however gives higher weight to large errors. [25]

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(F_t - A_t)^2}. \tag{3.3}$$

All three metrics are negatively oriented scores, which means lower values are better.

4 EXPERIMENTS

The main experiment is to take a slice of history (e.g. the past 60 days) and try to predict the the total arrivals to the ED in the next 24 hours. The dataset used is divided into a training set (the first ~ 1200 days) and a test set (last ~ 400 days). The training set is used to train the models and the test set to measure and validate their performance. It is highly imperative that the training and test set are kept separate and the samples are not shuffled as the point of the test set is to simulate real world performance on data the models have not seen and learned from before.

The same experiment is done with three different implementations. First implementation is a Recurrent Neural Network relying purely on LSTM for temporal relations. The model is built using TensorFlow [26] - a Python library by Google. This is a very basic model as it lacks most of the advanced features (e.g. temporal self-attention) that the TFT model has.

The second and third implementations are done using the Temporal Fusion Transformer. PyTorch forecasting [27] is a newly released Python library which utilizes the TFT architecture. Both PyTorch implementations also predict the next days arrivals. Implementation 2 uses the same daily frequency as the pure LSTM model. The third implementation instead uses hourly data and outputs a prediction for each hour for the next 24 hours. These hourly predictions are then summed up to be comparable with the other 2 implementations.

All three models are compared to a baseline model and to each other using few metrics (e.g mean absolute percentage error). The baseline model uses the last known target value to make a prediction. All three models should at the very least perform better than this baseline.

Model training is done using a single NVIDIA GTX 1080 GPU. For pure LSTM network using TensorFlow and optimizing for GPU usage the training process for 50 epochs takes under a minute. Similar run on the TFT architecture requires significantly more resources as it takes 15 minutes for a 50 epoch run. Changing to hourly frequency requires the network to handle much more data which transfers to higher costs. a 50 epoch run with hourly frequency takes upwards of 3 hours to complete. For the hourly model some restrictions are also needed to ensure a mid range GPU can handle the increased data

rates and higher memory requirements. However, for all 3 of the models, once the training is complete making new predictions can be done virtually instantaneously.

4.1 Data

The dataset contains hourly data from the Pirkanmaa hospital district (PHSP). In total there is over 35 000 hours of data from May 2015 to September 2019. From this dataset 30 different variables are used to train the models. These include calendar variables (e.g. holidays), past weather and a collection of internet variables (e.g. website visits).

Before training the models some preprocessing of the data is needed. With the PyTorch library most of the preprocessing is done to get the dataset from hourly frequency into daily format. Doing this also reduces the computational complexity as training time for the 2 daily frequency implementations is significantly lower than when done with hourly data. The PyTorch forecasting has many normalization and variable transformation features that the LSTM implementation lacks where they need to be done manually. For the LSTM model all variables are converted into float values (decimal numbers) and scaled to be between 0.0 and 1.0. For example true - false variables are transformed to correspond 0.0 for false and 1.0 for true. This step is not needed in the TFT implementation as it can handle categorical values natively. Full code for all 3 of the models can be found from the reference GitHub repository [28].

4.2 Hyperparameter optimization

Hyperparameters are properties of the model that govern the entire training process and largely define the model structure. Finding a good combination of hyperparameters for the network is crucial for the success of the model. [29] Following list introduces the hyperparameters for the LSTM and TFT models. Values used for these hyperparameters are presented in table 4.1.

1. **Hidden layers**. Number of hidden layers in the network.
2. **Hidden units**. Number of hidden units (i.e. neurons) in each layer.
3. **Loss function**. Output of a loss function represents how well the model is performing during training. The network seeks to minimize the error produced by the loss function.
4. **Dropout**. Percentage of ignored neurons.
5. **Epochs**. Number of times the model sees the whole dataset from start to finish.
6. **Window size**. How much history the network is given when it makes a prediction. Larger window size might increase accuracy while always increasing the computational complexity (i.e. time and/or money spent on training the model).

7. **Learning rate**. How quickly the network updates its parameters (e.g. weights). Often a decaying learning rate works best meaning the model makes big adjustments at start and slows down towards the end of the training.
8. **Batch size**. The dataset is divided into batches and each batch is processed as a seperate instance. Between each batch the model makes adjustments to the network according to the error produced by the loss function.
9. **Attention head size**. Number of attention heads. TFT specific.
10. **Hidden continuous size**. Size of the network for processing continuous variables. TFT specific.

Hyperparameter	LSTM	TFT daily	TFT hourly
Hidden layers	1	2	2
Hidden units	4	15	20
Loss function	MAE	QuantileLoss	QuantileLoss
Dropout	0.18	0.27	0.13
Epochs	50	50	30
Window size	60	60	168
Learning rate	0.003	0.0024	0.0042
Batch size	64	64	128
Attention head size	x	1	9
Hidden continuous size	x	14	4

Table 4.1. *Hyperparameters for LSTM and TFT models.*

PyTorch forecasting has an inbuilt implementation for optimizing hyperparameters (optuna) [30]. It uses a bayesian optimization process [31] to find the best combination of values for the hyperparameters. Compared to grid search or random search it finds proper values much faster as it has the ability to learn each time a different set of hyperparameter values are used. The LSTM model uses a random search for hyperparameter optimization. For each parameter a minimum and a maximum value is set and before training a value is chosen in-between for that run. This is a common and valid approach as with time it also finds better and better values for the parameters albeit more inefficiently.

5 RESULTS

Table 5.1 presents the values of the 3 different metrics used to measure the performance of each model. All 3 models perform better than the baseline. This shows that a neural network approach could be used for this type of a problem. These results also show that using a more sophisticated network architecture tuned specifically for time series forecasting improves the prediction capabilities by a significant margin. Figure 5.1 presents these results in a more visual format.

It is clear that the pure LSTM network cannot match the TFTs performance. During traning the LSTM model begins to overfit to the training data much faster than the TFT models. Overfitting to training data is a concern in any machine learning solution and its presence in this experiment is prevalent. Even with the many tricks used in the LSTM model to combat overfitting, due to the relatively small size of the dataset, overfitting happens regardless. The smaller size of the dataset is not such a problem for the TFT models as seen from the results. The TFT model with its internal mechanisms (e.g. gating layer) manages overfitting better with smaller datasets.

The dataset used has a sudden shift in trend in 2018 as seen in figure 5.1. This is due to an institutional reorganization of the emergency department. A similar event is not present in the training data. The LSTM model fails to accommodate and the results are worse as a result. The TFT model in comparison quite quickly sees this change and tunes itself accordingly. This is even more obvious in the hourly model. Even though the LSTM model performs better than the baseline its usefulness in this application is questionable.

Method	MAPE	MAE	RMSE
Baseline	8.50%	21.59	26.81
Pure LSTM	7.96%	20.27	25.10
TFT Daily	6.68%	17.10	21.43
TFT Hourly	6.37%	16.28	20.37

Table 5.1. *Comparison table of best measured metric (validation set) for each model used. The metrics used (introduced in chapter 3.5) are negatively oriented meaning lower values are better.*

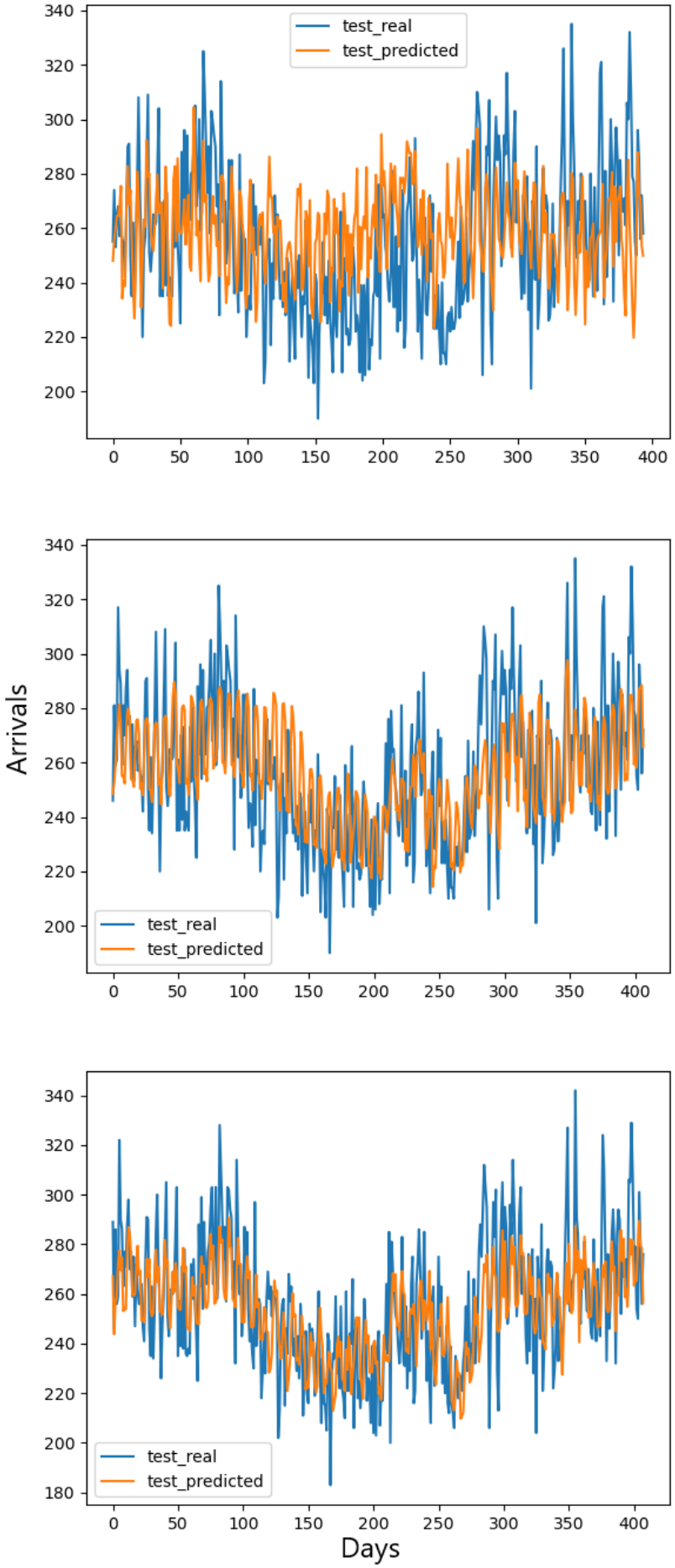

Figure 5.1. *Sample predictions for the validation set. Pure LSTM at the top. TFT daily in the middle and TFT hourly summed into daily format for comparison at the bottom.*

All 3 models can accurately detect the seasonality present in the dataset. However predicting the absolute peaks in either direction becomes problematic. If the model is not confident in its ability to predict accurately it tends to make only small adjustments to each prediction. This model confidence can be built in many ways. Easiest is just to train it more. With more training it starts to make more volatile shifts in its predictions. Unless the training dataset is grown, at some point the model validation performance begins to worsen with more training. The model starts to confidently making more and more incorrect predictions based on its findings from the training data. Again the better solution is to introduce more reliable data. Both longer history and more variables could build up model confidence without the downsides of overfitting.

5.1 Variable importance and model attention

The TFT model provides better interpretability compared to the LSTM model. Figure 5.2 presents the encoder and decoder variable importance from an example run. What the model thinks is important varies a bit between each run as it might focus on different things each time. This means that definite conclusion can't be made from just a singular example. In this example the model came to the conclusion that visits to the hospital website the day before has great relevance. Other notable variables for this example run were some weather variables (e.g. temperature), calendar variables (e.g. weekday/holiday) as well as the historical arrivals to the emergency department. The decoder has access only to the variables that we know in advance (e.g. the day of the week tomorrow). Here the day of the week played an important role in the prediction as well.

The model attention graph gives us a bit of an idea what the model is focusing on. For both the hourly and daily frequency model the attention, after some time of training, interprets that the peak values in the dataset are what it should focus on. The attention graph in figure 5.2 resembles quite closely to what the hourly arrivals to the emergency department are. This makes sense as focusing on the peak hours where most people arrive to the ED has the most significance to the end result prediction. The attention graph also often shows that events closer to the date we're prediction carry more significance.

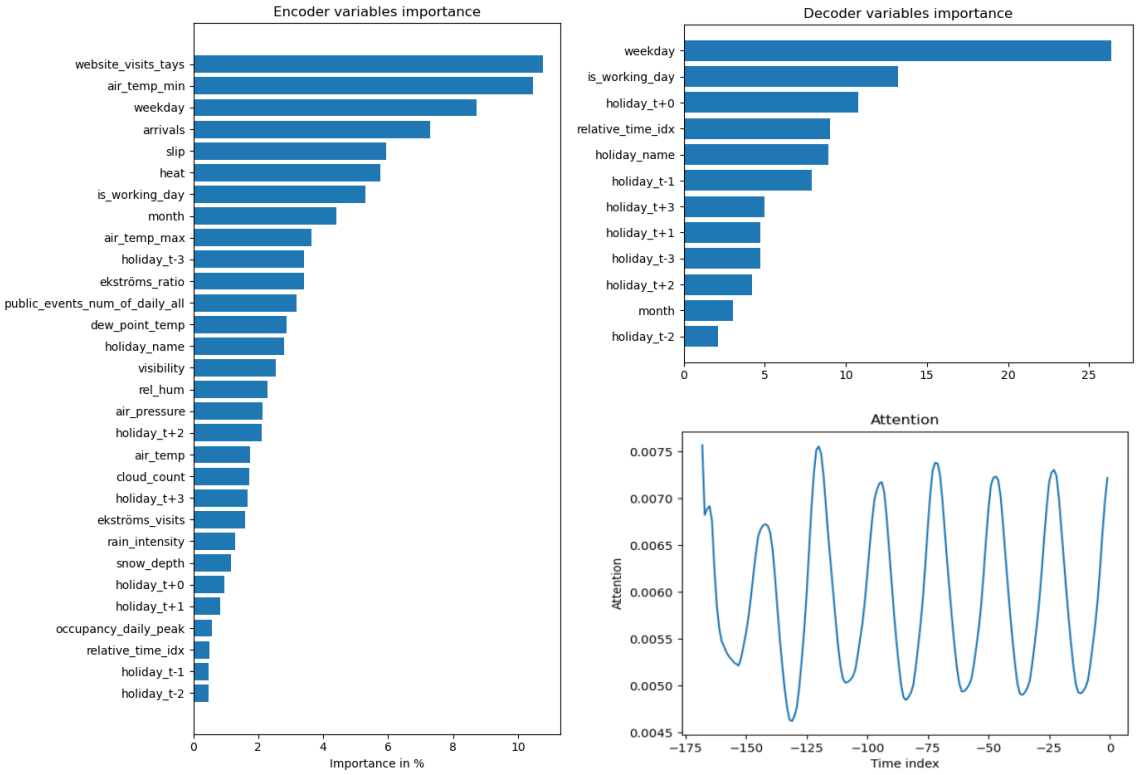

Figure 5.2. *Encoder (left) and decoder (top right) variable importance from the TFTs variable selection network. On bottom right the model attention graph from the Temporal Self-Attention layer (hourly frequency). Attention is focused on the daily peak hours for each day.*

6 CONCLUSIONS

Emergency departments suffer from chronic overloading. Being able to predict and respond to seasonal spikes in arrivals means better care for those in need. This study attempted to provide a possible solution for emergency department forecasting. The dataset used in this study is provided by the Pirkanmaa hospital district and it contains over 35 000 hours of data from 2015 to 2019.

Time series forecasting is a complex problem with a long history of different methodologies. This study used a newly invented machine learning model called the Temporal Fusion Transformer and compared its performance to a more traditional Neural Network. The Temporal Fusion Transformer itself is an attention based Deep Neural Network architecture made specifically for Multi-horizon time series forecasting. This architecture also provides new forms of interpretability which shed light on how the model makes decisions.

With the experiments conducted all three of the models used to predict the next days arrivals provided better results than the measured baseline. The more traditional Neural Network using LSTM cells for detecting temporal relations fell short compared to the more advanced capabilities of the Temporal Fusion Transformer. From the two different data frequencies tested the hourly model provided better results overall. With the Variable Selection Network and the Temporal Self-Attention layer we can deduce which of the many inputs proved to be more useful than others when making predictions.

All in all, the field of machine learning, especially the use of Neural Networks, is still in its very beginnings in many medical applications. With the growing rate of data being produced and the ever increasing computational prowess, Neural Networks, in conjunction with more traditional machine learning methods, could prove to be of significant importance in many applications across the medical field. Specifically in ED forecasting, a more thorough study is needed with more data from different sources and using a wider range of different forecasting methods for best results.

REFERENCES

[1] Brownlee, J. *What Is Time Series Forecasting?* Feb. 12, 2016. URL: https://machinelearningmastery.com/time-series-forecasting/ (visited on 03/11/2020).

[2] Andrew V. Metcalfe, P. S. C. *Introductory Time Series with R*. Springer, 2009.

[3] Bryan Lim, S. O. A. *Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting*. Dec. 19, 2019. URL: https://arxiv.org/abs/1912.09363 (visited on 11/17/2020).

[4] Spyros Makridakis Evangelos Spiliotis, V. A. *The M4 Competition: 100,000 time series and 61 forecasting methods*. Mar. 1, 2020. URL: https://machinelearningmastery.com/time-series-forecasting/ (visited on 03/11/2020).

[5] Alexiei Dingli, K. S. F. *Financial Time Series Forecasting – A Deep Learning Approach*. Oct. 2017. URL: http://www.ijmlc.org/vol7/632-P17.pdf (visited on 11/30/2020).

[6] Nayak, A. *Predicting Stock Price with LSTM*. Mar. 18, 2019. URL: https://towardsdatascience.com/predicting-stock-price-with-lstm-13af86a74944 (visited on 11/30/2020).

[7] Google/Kaggle. *Web Traffic Time Series Forecasting. Forecast future traffic to Wikipedia pages*. Sept. 1, 2017. URL: https://www.kaggle.com/c/web-traffic-time-series-forecasting (visited on 11/30/2020).

[8] al., S. K. et. *AI in Healthcare: Time-Series Forecasting Using Statistical, Neural, and Ensemble Architectures*. Mar. 19, 2017. URL: https://www.frontiersin.org/articles/10.3389/fdata.2020.00004/full (visited on 11/30/2020).

[9] Smart, A. *Artificial Intelligence Takes On Earthquake Prediction*. Sept. 19, 2019. URL: https://www.quantamagazine.org/artificial-intelligence-takes-on-earthquake-prediction-20190919/ (visited on 11/30/2020).

[10] Prabhakaran, S. *ARIMA Model – Complete Guide to Time Series Forecasting in Python*. Feb. 18, 2019. URL: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/ (visited on 11/30/2020).

[11] De Bruin, A. M., Van Rossum, A., Visser, M. and Koole, G. Modeling the emergency cardiac in-patient flow: an application of queuing theory. *Health Care Management Science* 10.2 (2007), 125–137.

[12] Ahmed, M. A. and Alkhamis, T. M. Simulation optimization for an emergency department healthcare unit in Kuwait. *European journal of operational research* 198.3 (2009), 936–942.

[13] Kolker, A. Process modeling of emergency department patient flow: Effect of patient length of stay on ED diversion. *Journal of Medical Systems* 32.5 (2008), 389–401.

[14] Medeiros, D. J., Swenson, E. and DeFlitch, C. Improving patient flow in a hospital emergency department. *2008 Winter Simulation Conference*. IEEE. 2008, 1526–1531.

[15] al., W. W. et. *Forecasting arrivals and occupancy levels in an emergency department*. June 2019. URL: https://www-sciencedirect-com.libproxy.tuni.fi/science/article/pii/S2211692318300407#sec4.3 (visited on 11/30/2020).

[16] Holecek, M. *Speeding up Google's Temporal Fusion Transformer in TensorFlow 2.0*. Sept. 3, 2020. URL: https://medium.com/@ampx/speeding-up-googles-temporal-fusion-transformer-in-tensorflow-2-0-b45721e5663a (visited on 11/17/2020).

[17] Nielsen, M. *Neural Networks and Deep Learning*. Dec. 2019. URL: http://neuralnetworksan com/ (visited on 11/19/2020).

[18] Brownlee, J. *Crash Course On Multi-Layer Perceptron Neural Networks*. May 17, 2019. URL: https://machinelearningmastery.com/neural-networks-crash-course/ (visited on 11/19/2020).

[19] Wilson, A. *A Brief Introduction to Supervised Learning*. Sept. 29, 2019. URL: https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590 (visited on 11/19/2020).

[20] Brownlee, J. *A Gentle Introduction to Long Short-Term Memory Networks by the Experts*. May 24, 2017. URL: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/ (visited on 11/19/2020).

[21] Phi, M. *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. Sept. 24, 2018. URL: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 (visited on 11/19/2020).

[22] SMITH, D. *Interpreting Patterns in Multi-Variate Multi-Horizon Time-Series Forecasts from Google's Temporal Fusion Transformer Model*. May 19, 2020. URL: https://www.theorylane.com/2020/05/19/interpreting-patterns-in-multi-variate-multi-horizon-time-series-forecasts-from-googles-temporal-fusion-transformer-model/ (visited on 11/23/2020).

[23] Guo, G., Wang, H. and Bell, D. Data Reduction and Noise Filtering for Predicting Times Series. Aug. 2002, 421–429. ISBN: 978-3-540-44045-1. DOI: 10.1007/3-540-45703-8_39.

[24] Baeldung. *Understanding Forecast Accuracy: MAPE*. Sept. 12, 2020. URL: https://www.baeldung.com/cs/mape-vs-wape-vs-wmape (visited on 11/25/2020).

[25] JJ. *MAE and RMSE — Which Metric is Better?* Mar. 23, 2016. URL: https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d (visited on 11/25/2020).

[26] Google. *An end-to-end open source machine learning platform*. Nov. 24, 2020. URL: https://www.tensorflow.org/ (visited on 11/24/2020).

[27] al, J. B. et. *Introducing PyTorch Forecasting*. Nov. 24, 2020. URL: https://pytorch-forecasting.readthedocs.io/en/latest/ (visited on 11/24/2020).

[28] Pulkkinen, E. *Git repository for code*. Dec. 17, 2020. URL: https://github.com/eetulauri/ED-Forecasting (visited on 12/17/2020).

[29] Prabhu. *Understanding Hyperparameters and its Optimisation techniques*. July 3, 2018. URL: https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568 (visited on 11/25/2020).

[30] Preferred Networks, I. *An open source hyperparameter optimization framework to automate hyperparameter search*. Dec. 1, 2020. URL: https://optuna.org/#key_features (visited on 12/01/2020).

[31] al., A. A. et. *Exploring Bayesian Optimization*. May 5, 2020. URL: https://distill.pub/2020/bayesian-optimization/ (visited on 12/01/2020).