Deep learning approaches to short term traffic forecasting

Capturing the spatial temporal relation in historic traffic data

Thomas Kuiper

Supervisor: Elena Congeduti

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Thomas Kuiper
E-mail of the student: T.I.Kuiper@student.tudelft.nl
Final project course: CSE3000 Research Project
Thesis committee: Elena Congeduti, George Losifidis

Abstract

The amount of cars on the roads is increasing at a rapid pace, causing traffic jams to become commonplace. One way to decrease the amount of traffic congestion is by building an Intelligent Transportation System (ITS) which helps traffic flow optimally. An important tool for an ITS is short term traffic forecasting. Better forecasts will enable the ITS to proactively prevent congestion. Recent years have seen a great increase in the availability of traffic data. As a result deep learning approaches have begun to emerge as models of choice in the short term traffic forecasting domain. Among deep learning approaches Long Short Term Memory (LSTM) and Temporal Convolutional Networks (TCN) have both shown state-of-the-art performance in general forecasting tasks as well as promising results in traffic forecasting. This work has compared both of these approaches in terms of capturing the temporal spatial correlation and scalability. The LSTM showed more ability to capture the temporal spatial correlation while both architectures seemed equally scalable.

1 Introduction

The rapid urbanisation of the world has increasingly brought more people together, causing many roads around the world to face increasing numbers of vehicles. As a result traffic jams are becoming common place. Traffic congestion can lead to economical damages, increased road accidents, more pollution etc. Therefore it would be greatly beneficial to reduce the amount of traffic congestion.

Building more infrastructure is an effective solution, however it can cost a lot of time and money while also being limited by space. Instead it is often more viable to use control strategies to reduce congestion in traffic. By adjusting speeds or rerouting in real time, a lot of traffic congestion can be avoided. These control strategies are part of what became known as Intelligent Transportation Systems(ITS), which try to use innovative solutions to achieve maximum usage out of the available road network. At the basis of any ITS is traffic forecasting which enables proactive decisions to prevent congestion.

Plenty of research into traffic forecasting has already been done over the last decades. Several flow-theory models [4] and statistical based models [5], have been proposed and used successfully for a long time. In recent years the amount of available data has become abundant due to the deployment of traffic sensors, most previously used models however cannot take advantage of this. The curse of dimensionality prevents these models from learning the deeper correlation hidden within the data leaving them with a shallow understanding. The correlation in traffic data can be more challenging to learn than a typical forecasting problem because traffic prediction involves temporal and spatial correlations. A model that can capture both the temporal and the spatial dependencies in this large amount of data is needed.

Neural network (NN) approaches have shown to be able to overcome the curse of dimensionality due to distributed calculations. Although conventional NN's are not the best fit to capture temporal-spatial dependencies, many deep learning architectures have shown to be capable of capturing these dependencies. In [12] and [1] long-term-short-memory(LSTM) models are proposed for traffic forecasting. The authors of [8] used a deep convolutional NN (CNN) to build a traffic prediction model. Both of these types of models are known for their state-of-the-art performance on time-series forecasting problems[7], and have shown good performance in traffic forecasting.

This work aims to compare LSTM and CNN architectures in the domain of traffic forecasting. Whilst both show good performance, it is not yet known what architecture is best at capturing the temporal-spatial correlation. By comparing these two relatively simple architectures deductions can be made about the capabilities of more complex architectures that are based on LSTM or CNN features.

2 Methods

This section explains the key concepts that are used.

2.1 The dataset

The data used is a month of signals from 130 traffic sensors in the city center of The Hague. The original signals are magnetic field readings which are converted into a number of cars passing over said sensor every 15 minutes. In this form the data can be used for analysis.

2.2 Spatial-Temporal Correlation

Traffic prediction is a time series prediction problem. This means that given a time sequence $[x_1, .., x_t]$ the goal is to predict x_{t+1} where t is the last recorded data point. So an architecture suited for time series problems should be able to approximate the dependency between $(x_{t+1},\ [x_{t-b}, .., x_t])$ where b is how far back in time the model looks to predict $(x_{t+1}$.

Traffic prediction however, is not just a time series problem, it is a spatial-temporal problem. For each sensor the correlation between other sensors can also be considered. More formally this means that given several time sequences $[X_1, .., X_s]$ where X is a time sequence and s is the number of sensors being considered, the correlation between $(X_i,\ X_j)$ also needs to be considered for all $i < s\ \&\ j < s$.

2.3 Deep Learning Methods

Deep neural networks are well suited to learning predictive relationships. A neural network uses a sequence of non-linear hidden layers followed by an output layer to give the predictions. The hidden layers will learn intermediate features that can be put together by the output layer into critical features used for prediction [3]. There are many different deep learning architectures, the ones discussed in this section have shown great performance in time series predictions as well as promising traffic prediction results. First Recurrent Neural Networks (RNN) are introduced, followed by a variant of a

RNN: a Long Term Short Memory (LSTM) architecture, after which Temporal Convolutional Networks (TCN) will be discussed.

RNN

A RNN is a Deep learning architecture specialised for sequence tasks.

It can take a sequence of inputs $[x_1, .., x_t]$, after each input x_i it will update its hidden layer h_i to h_{i+1} using an update function. When the last input x_t is processed, the final output of the model will be produced. This recurrent structure enables the output to be effectively based on all inputs. The reccurent structure can take a variable amount of inputs making it ideal for language modelling tasks. Using the language modelling task as an example, a normal feed forward network would have to learn for each input all the rules of the language separately whereas a RNN can learn the rules once and apply them to all inputs [6].

Though great, RNN's are not the fit all solution for sequential problems. Due to the vanishing and exploding gradient problems, RNN's accuracy degrades when modelling larger timespans [12]. A different version of a RNN known as a LSTM, does a bit better in this regard using gated units that have a more complex update function.

LSTM

Similar to the RNN, a LSTM also has a recurring structure that takes a sequence of inputs and after each input it updates it's structure. A LSTM treats its hidden layer as a memory cell that is updated with the gates that connect to it. There is an input, output and forget gate. This effectively enables the LSTM to decide whether something is worth keeping in memory, allowing it to forget irrelevant information. This way the LSTM architecture can more effectively capture relationships in time than the RNN structure.

TCN

Convolutional networks are typically used for image analysis. These networks use two-dimensional convolutions to capture relations along the width and length of an image. One-dimensional convolutions however can be used to capture relations along the time dimension. This works well for small inputs however for larger inputs it can be hard for a normal convolutional network to find the relations between timesteps that are far apart. The TCN architecture has a key feature to solve this which is dilated convolutions. The effect of dilation is illustrated in Figure 1, by increasing the dilation rate throughout several layers a wide reach can be achieved without needing to do more calculations. Using dilated convolutions a deep temporal memory can be built up and used to achieve accurate predictions. Authors of [2] have shown that in many cases a TCN can perform sequence tasks better than the commonly used recurrent architectures.

3 Experimental Setup and Results

This section outlines how the experiments are performed, it also describes what the exact setup is of all the parts and lastly the results are shown.

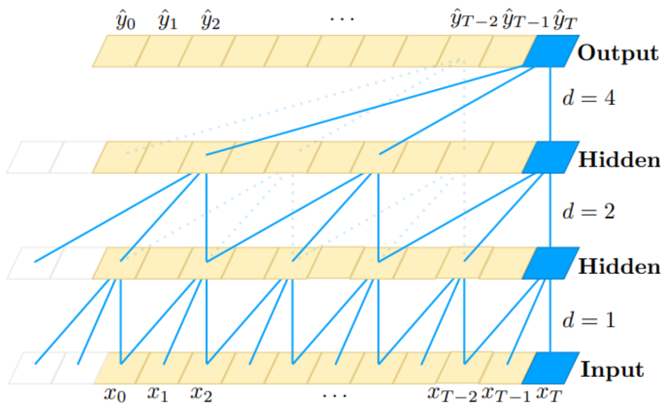
$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ $\cdots$ $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 $\cdots$ x_{T-2} x_{T-1} x_T

Figure 1: Made by authors of [2]. This figure shows how dilation can widen the reach of convolutions. A kernel size of 3 is used and the dilation rate can be seen on the side. Note that a dilation rate of 1 is the same as no dilation.

3.1 Measures

To evaluate two models some measures to compare them by need to be established.

1. Accuracy - Root Mean Squared Error (RMSE) score of predictions as in Formula 1
2. Ability to capture spatial-temporal correlation - How much can the model improve when increasing the amount of sensors as input data
3. Scalability - How much does performance degrade when predicting multiple sensors at the same time

$$RMSE = \sqrt{\frac{1}{t}\sum_{i=1}^{t}(x_i - \hat{x}_i)^2} \tag{1}$$

$$MAE = \frac{1}{t}\sum_{i=1}^{t}|x_i - \hat{x}_i| \tag{2}$$

3.2 Experiment

To perform an experiment comparing two different neural network architectures there needs to be some common ground. So to make sure both networks have the same learning capabilities they are made to have around the same amount of trainable parameters.

Spatial-temporal correlation

To find how well the model can capture the spatial-temporal correlation a single sensor will be predicted. When using only the sensor itself as input there is only a temporal correlation in the data. The accuracy score of this simple model can then be used as a baseline for the rest of the experiment. When adding more sensors to the input a spatial-temporal correlation is introduced in the data. The experiment will gradually introduce more sensors to the input, and measure the accuracy. The additional sensors that are added are selected based on how correlated they are to the sensor being predicted. This

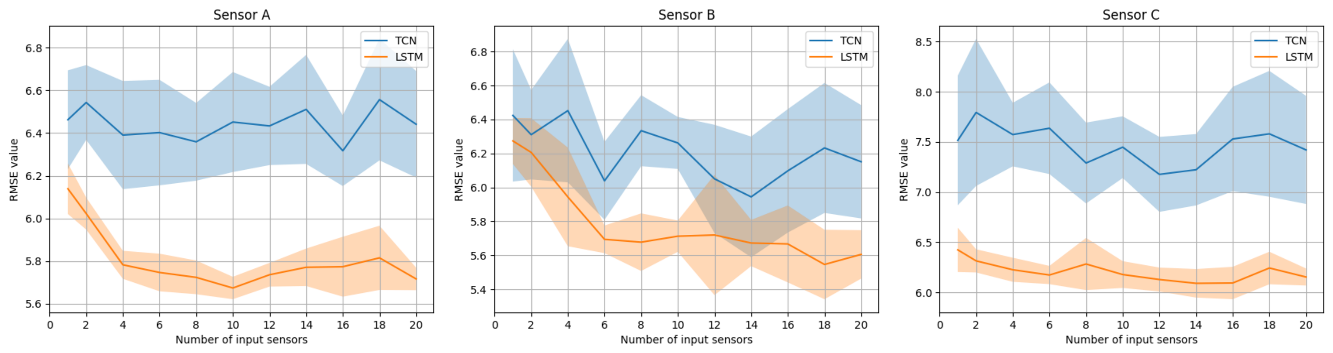

Figure 2: Three graphs showing the accuracy of predictions in RMSE as amount of input sensors increase. Each graph shows the prediction of a different sensor, with the TCN as blue and LSTM as orange.

is defined by the Pearsons correlation between the output sensor and all other sensors. There are only a limited amount of correlated sensors, so we assume that eventually adding sensors to the input does not increase accuracy anymore, so the experiment is only performed up until 20 input sensors. When the accuracy scores are obtained, they are compared to the previously established baselines to find the improvement. The ability to capture the spatial-temporal correlation is then decided by how big the improvement is.

The output sensor chosen to perform the experiment could affect the outcome. Some sensors have different patterns than others which could cause a bias towards a certain acrhitecture. For that reason this experiment is performed with 3 different output sensors that will be referred to as A, B and C.

Scalability

To test scalability a similar experiment can be performed, instead of varying the amount of inputs, the amount of outputs are varied. Gradually increasing the amount of outputs while having all sensors as inputs. As a baseline the average accuracy of prediciting each sensor individually is used. Then the accuracy scores of the experiment are compared with the baseline to measure how well the model stays accurate when predicting multiple sensors.

3.3 Data Preparation

To make the data suitable for a time series problem it needs to be split into a sequences of chronological steps, where the last step serves as the desired output and all steps before as the input. This can be done using a sliding window transformation. Using a specified look-back window size b and starting the time-step t as b, out of the data a slice as follows can be taken $[t-b, ..., t-1, t]$. Repeatedly increasing t by one until reaching the last recorded time of the data, will give a data set suitable for time series analysis. Before inputting the data into the models, a normalisation is performed to keep the weights of the network similarly sized. Then lastly a train/validation/test split of 60%/15%/25% is used. Using the train set to train the model, the validation set to tune hyper parameters and the test set to evaluate the performance.

3.4 Setup

To perform the experiment both the LSTM and TCN model need to be trained many times for the experiments to reduce the effects of the inherent randomness of neural networks. The sections below will detail the consistent hyper parameters and structure used in the models. Hyper parameters were found by trying a range of common and sensible values, and selecting the optimal values among them. Both models are built using the Tensorflow library Keras.

LSTM setup

The LSTM model consists of 3 layers. The first two layers are both LSTM layers with 128 units, followed by a fully connected layer that connects the 128 outputs of the second LSTM with the amount of nodes equal to the number of sensors that are being predicted. This final layer then produces an output for each sensor predicted. The input data has a look-back window of 96 timesteps. Meaning that the input to predict the next step is the previous 24 hours of data. Lastly, for the training specific parameters a learning rate of 0.001 is used with a batch size of 32 while running for 500 epochs with early stopping based on the validation set.

TCN setup

The TCN uses 6 residual blocks which consist of two one-dimensional convolution layers, each followed by a RelU actiavtion and a dropout layer of 0.2. Every block has some parameters for its convolutions. The amount of filters (or output channels) each block has is structured as follows: $[64, 128, 128, 128, 64, no\_outputs]$ where the no outputs is the amount of sensors being predicted. Each layer has a dilation rate 2^i where i starts at 0 and for each subsequent layer i gets incremented by 1. Both the residual block structure and dilation rate is modelled after the TCN built by authors of [2]. Across all blocks a kernel size of 3 is used in combination with causal padding to ensure the predictions are only based on the past. The input data once again is 96 timesteps (or 24 hours), with the same run time parameters as the LSTM model.

3.5 Results

Each result data point in the following section is an average of 10 training cycles. Meaning all parameters were initialized, trained and evaluated 10 times. The results are also shown in two metrics RMSE and Mean Absolute Error (MAE) as in formula 2. RMSE is the metric the models are trained on, MAE gives a view that is not biased to larger absolute errors.

	LSTM		TCN		Improvement	
Inputs	RMSE	MAE	RMSE	MAE	LSTM	TCN
1	6.14	4.13	6.46	4.16	Baseline	
2	6.02	4.04	6.54	4.11	-0.12	+0.08
4	5.78	3.88	6.39	4.04	-0.36	-0.07
6	5.74	3.84	6.40	**4.02**	-0.39	-0.06
8	5.72	3.84	**6.36**	4.03	-0.42	**-0.10**
10	**5.67**	**3.83**	6.45	4.03	**-0.47**	-0.01

Table 1: Results of the first experiment from sensor A. One sensor is predicted with different amount of input sensors. Shows the accuracy of both models as well as the improvement in RMSE compared to the baseline is shown.

Spatial-Temporal Correlation Experiment

In Table 1 the results of the first experiment with sensor A can be found. Across the board the LSTM model has a better accuracy. Furthermore compared to the baseline the LSTM shows more improvement with an increase of inputs, showing that it is more capable of learning the spatial-temporal correlation. The results of all 3 experiments have been graphed in figure 2, from which similar conclusions can be drawn. The LSTM model significantly improves as inputs increase for all sensors. For sensors A and C the TCN struggles to improve its score, but for sensor B it shows improvements. However the accuracy varies a lot even after averaging over 10 cycles, making the TCN seem less capable of finding the effective information and filtering out the useless information. Whereas the LSTM stays fairly consistent even when the added data does not provide new insights. So it is fair to say these results point to the LSTM architecture being more accurate and also more effective at learning the spatial-temporal correlation.

Scalability Experiment

The results of this experiment can be found in Figure 3. The results see both the LSTM and TCN increase by about the same amount of RMSE. It seems both models experience the same accuracy loss trend when tasked with predicting an increasing amount of sensors at the same time. Although it is possible to build a seperate model for each or for a small amount of sensors, on a large scale it would be a lot less costly and a lot more efficient if this could be done in one model.

When scaling up, the TCN architecture could get an advantage over the LSTM with its faster runtime. A recurrent architecture like LSTM can only do 1 step at a time because the next step is dependent on the previous. The TCN however can calculate its convolutions in parallel, meaning if there is enough processing power and memory it can be sped up tremendously.

4 Conclusions and Future Work

This work describes two deep neural network based traffic prediction models. They were implemented and compared with each other in two different experiments. It was found

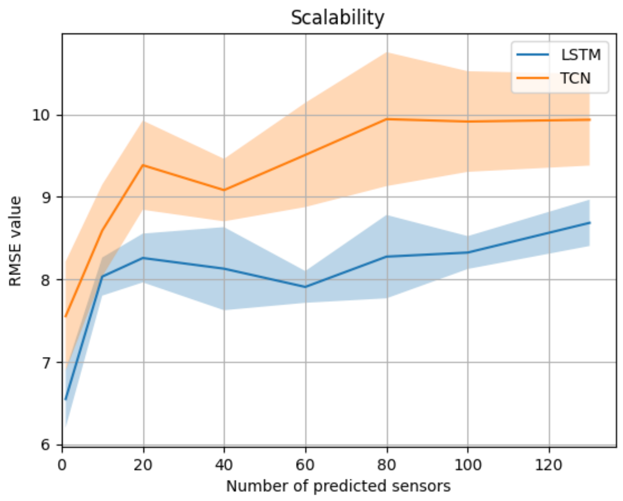

Figure 3: Accuracy of predictions in RMSE while amount of output sensors increase. The amount of input sensors is kept constant at 130. The RMSE calculated is the average of the RMSE of each individual predicted sensor.

that the TCN and LSTM scale similarly with some upside of faster potential training time for the TCN. However the LSTM shows better performance in accuracy and ability to capture the spatial-temporal correlation. So it can be concluded that the LSTM is a better pick for this case study. The results also suggest that for a short-term traffic forecasting problem, a LSTM based architecture is most likely worth trying.

In future work more complex architectures can be compared. Authors of [11] proposed a gated TCN architecture to predict traffic, while authors of [9] propsed a graph convolutional model for traffic prediction. Another interesting model that could be considered is one that combines a LSTM and convolutional architecture, which was found to have good results in [10]. Lastly it would be interesting to see if results could improve from decreasing the timesteps in the data from 15 minutes to 10 or 5.

5 Responsible Research

There are no ethical concerns with the data that is used, there are no privacy concerns involved with it nor can it be used for unethical purposes.

Neural networks inherently have some variability causing them to not always produce reliable or consistent outcomes. To account for this all reported findings are averaged out over several training cycles. This should ensure that when reproduced similar results will be found. Furthermore, a detailed setup is given in the Experiment section for the neural network architectures. This combined with access to the source code should give other developers enough information to be able to reproduce the experiment.

References

[1] Zainab Abbas, Ahmad Al-Shishtawy, Sarunas Girdzijauskas, and Vladimir Vlassov. Short-term traffic pre-

diction using long short-term memory neural networks. In *2018 IEEE International Congress on Big Data (BigData Congress)*, pages 57–65, 2018.

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[4] Carlos F. Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, 28(4):269–287, 1994.

[5] Xiang Fei, Chung-Cheng Lu, and Ke Liu. A bayesian dynamic linear model approach for real-time short-term freeway travel time prediction. *Transportation Research Part C-emerging Technologies - TRANSPORT RES C-EMERG TECHNOL*, 19:1306–1318, 12 2011.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[7] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):202–209, feb 2021.

[8] Xiaolei Ma, Zhuang Dai, Zhengbing He, Jihui Ma, Yong Wang, and Yunpeng Wang. Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, 17(4), 2017.

[9] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, jul 2018.

[10] Haiyang Yu, Zhihai Wu, Shuqin Wang, Yunpeng Wang, and Xiaolei Ma. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks, 2017.

[11] Changxi Zhao. Traffic flow prediction model based on temporal convolutional network. In *2023 17th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, volume 1, pages 1–4, 2023.

[12] Zheng Zhao, Weihai Chen, Xingming Wu, Peter C. Y. Chen, and Jingmeng Liu. Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2):68–75, 2017.