Traffic prediction using a Deep Learning paradigm

Felix Kunde Alexander Hartenstein Stephan Pieper Petra Sauer

Beuth University of Applied Sciences
Luxemburger Strasse 10
13353 Berlin, Germany
{fkunde, s58380, spieper, sauer}@beuth-hochschule.de

ABSTRACT

For many years intelligent transportation systems (ITS) have been collecting and processing huge amounts of data from numerous sensors to generate a ground truth of urban traffic. Such data has set the foundation of traffic theory, planning and simulation to create rule-based systems. It has also been used in many different studies in data-driven short-term traffic flow forecasting with promising results.

Still, the acceptance for data-driven predictions is quiet low in productive systems of the public sector. Without enough probe data from floating cars (FCD) ITS owners feel unable to reach accuracy like private telecommunication or car manufacturing companies. On the other hand, investigating into FCD requires a thoughtful treatment of user privacy and a close look on data quality which can also be very time consuming.

Recent progress in hardware and deep learning software has lowered the bar to handle machine learning algorithms what urges the field of traffic forecasting to continue exploring the predictive power of artificial intelligence. With this paper we present our first approach of feeding sensor data to an Artificial Neural Network (ANN). We train the ANN with different spatial and temporal lags to find an optimal setup for an entire city.

Categories and Subject Descriptors

• Information systems→Information systems applications→ Spatial-temporal systems→Sensor networks • Computing methodologies→Machine learning→Machine learning approaches→ Neural networks.

Keywords

Traffic forecasting; spatio-temporal data mining; deep learning; neural networks; Tensor Flow

1. INTRODUCTION

Detecting macroscopic traffic parameters such as travel time (time needed per trajectory) or traffic density (number of cars per trajectory) is crucial for managing and monitoring an intelligent transportation system (ITS). Such a system must adapt to different traffic scenarios and provide guidance to drivers to reduce traffic congestion and road collisions. Plus, it produces input data for traffic simulation programs which are used for traffic planning.

Apart from static sensor data, probe data from driving cars (Floating Car Data - FCD) is a very valuable resource as it can deliver trajectory-based data with a greater yet more realistic accuracy. Some cities have contracts with companies that own a great fleet of vehicles to deliver probe data, e.g. public transport or taxi cab companies. However, such datasets are mostly biased as the driving behavior can be bound to certain tasks e.g. busses have a fixed route and schedule what might cause waiting times or intentional delaying while driving.

From the very precise traffic information of the routing engines from Google Maps[1] or Here Maps[2] we can see the benefit of private transport data which is produced from the GNSS units of cars or smartphones. Usually, such data is not available to an ITS of the public sector. A city might fear an investment in such a (potentially huge) data set because of the hardware and software requirements it takes to process and store it. Depending on the level of detail of the recorded tracks, guarantees would have to be made that user privacy is treated carefully.

2. PROBLEM DEFINITION

We want to engage cities to have more faith in the data, that they are already collecting. Many ITS only take the detected data to monitor the current state of the traffic to react to traffic congestion e.g. by switching signs or blocking roads. The time series analysis methods used on the historic data sets are mostly very simple, e.g. moving average or exponential smoothing. Traffic predictions for future temporal horizons longer than 15 minutes are neither applied nor trusted although there is plenty of sample data available to run against modern algorithms.

One of these algorithms could be artificial neural networks (ANN), which got a lot attention recently under the buzz word "Deep Learning". The basic idea of ANNs is not new and many researchers have already adopted them for traffic flow forecasting [1]. But, with grown CPU and GPU power plus newly available deep learning frameworks like Google's Tensor Flow, Facebook's Torch or SkyMind's DeepLearning4J we see a potential for powerful additions in terms of usability and scalability. It is easier than ever to train an ANN with numerous input and target data setups and optimize its hyperparameter settings. On the other hand, the possible number of different combinations can make it hard to find a solution that provides a solid prediction for most scenarios.

This short paper will present our initial results with a Feed Forward Neural Network (FFNN) which we have trained with univariate inputs of different spatially correlated sets of double inductive loops and different time lags. We kept the setup very simple in order to see what the network can learn by itself and to make better statements about the prediction accuracy in future tests where we would change certain parameters.

[1] https://www.google.de/maps

[2] https://wego.here.com/

The next chapter will introduce related work in the field of short-term traffic forecasting. We will then describe the FFNN model that we have used followed by the experimental setup for analyzing the predictive power under different spatial and temporal dimensions. In chapter 6 we will present our results and conclude with a future outlook.

3. RELATED WORK

Short-term traffic forecasting based on sensor data has seen many different approaches in the last decades, be it for freeways or arterial road networks, with univariate or multivariate inputs and for different temporal lags [1]. The applied methods are ranging from classical parametric solutions like autoregressive statistics for time series (ARIMA) [2], k-Nearest neighbors on historic data sets [3], Bayesian networks [4] to non-parametric predictions by support vector machines (SVM) [5] and ANNs [6]. [5] have pointed out that it is often difficult to compare them because of their heterogeneous setup. Usually, one method is engineered exhaustively and compared to only simple variants of other algorithms paradigms (base lines). For the future we plan to provide a comprehensive comparison of different forecasting methods incl. spatial-temporal ARIMA [7] and SVMs.

Due to the hype on Deep Learning a growing number of papers on traffic forecasting can be noted that use modern ANN architectures such as convolutional networks (CNN) (good for learning on fuzzy data such as images and audio streams) or variants of recurrent networks (RNN) (good for learning on sequence-based data). RNNs seem to be very suitable to mine on time series of traffic sensors [6][8]. To learn on long sequences, the Long Short Term Memory network (LSTM) can be used [9][10]. Also, a combination seems reasonable when a recorded traffic state is regarded as an image. The CNN would extract patterns such as traffic congestion and the LSTM would learn how the patterns evolve [11].

Generally, we are interested in the impact of location on the prediction to find spatio-temporal correlations in traffic. Many studies have also proven the relevance of a spatial dimension to improve the accuracy of the predictions e.g. [7] or [8]. One technique to filter the input data against spatial dependencies is to apply a spatial weight matrix to strengthen the relations between neighbors. This has mostly been done for parametric approaches [12]. ANN-driven research for traffic prediction is often lacking a complex spatial weighting model. Either the number of sensors is very low [10] or a freeway setting is used [8], where spatial relation between upstream and downstream is already given by the road network itself. Therefore, we combine our experiments on FFNNs with ideas from [12].

4. MODEL SETUP

For further description of the network architecture we are using the same convention as in [13]. We have implemented a FFNN using Google's Tensor Flow framework. The network consists of an input layer, one hidden layer h and one output layer o. The number of input nodes i and hidden nodes j is bound to the number of sensors we consider as a valid source to go into our model (98% availability of measurements in the training data set). The number of output nodes k is limited to the number of prediction horizons we choose for one sensor (see next chapter). We have tested the network with higher numbers of neurons (up to 150) which let the overall error increase. For each layer we are using a sigmoid activation function l to produce the value v of every neuron. The sigmoid function $\sigma(z)$ is a classical nonlinear function and a good choice if we want to detect nonlinear patterns in our data.

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad v_j = l_j\left(\sum_{j'} w_{jj'} . v_{j'}\right) \tag{1}$$

Here j' stands for nodes that are connected with the hidden neuron j and w is the weight of edges jj'. The root mean square error (RMSE) is our loss function $L(\hat{y}_k, y_k)$ which modifies the network at each iteration using backpropagation [14]. The weights between the neurons are adjusted by stochastic gradient descent (SGD) with a batch size of 20 iterations. The backpropagation algorithm subsequently calculates the derivatives of L from output nodes k (2) to hidden nodes j with respect to their corresponding activation function (3).

$$\delta_k = \frac{\partial\, L(\hat{y}_k, y_k)}{\partial\, \hat{y}_k} . \acute{l}_k\left(\sum_j w_{kj} . v_j\right) \tag{2}$$

$$\delta_j = l'\left(\sum_{j'} w_{jj'} . v_{j'}\right) \sum_k \delta_k . w_{kj} \tag{3}$$

5. EXPERIMENTAL SETUP

We train our FFNN using 1 month of data from July 2015 produced by 59 double induction loops which are spread across the city of Dresden. The data is coming from the Dresden ITS called VAMOS[3] and aggregated to minutely values. We choose double inductive loops as they are capable in capturing the speed of cars accurately. This group of loop detectors is usually installed on main roads and with enough distance to intersections. Therefore, the measurements do not get affected by waiting queues. Nevertheless, the time series are still quiet noisy because of traffic lights intervals. Even during rush hours there can be minutes with no detected cars (see Figure 1).

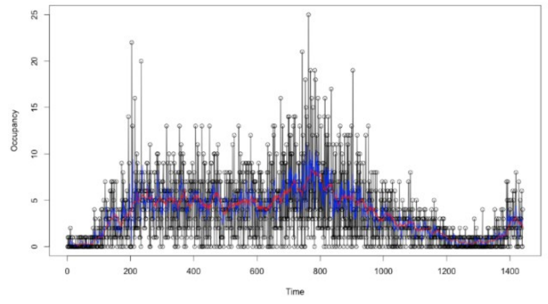

Figure 1 Occupancy at one loop detector for one day in contrast to moving averages of 25 and 50 minutes

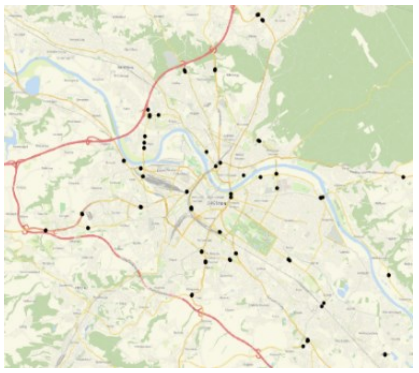

Figure 2 Locations of double inductive loops

[3] https://tu-dresden.de/bu/verkehr/vis/vlp/forschung/forschungs-projekte/verkehrsmanagementsystem-vamos-dresden

Unfortunately, we had to exclude many sensors because of missing data, but we are currently working on repair mechanisms to include more detectors. However, we still got a good spatial distribution across the city (see figure 2).

As for the input of the neural network we are generating a matrix which consists of data from all valid sensors S and measurements T of variable x (occupancy in our tests). The input values are smoothened by a rolling mean of 50 minutes to ease the prediction and normalized to a range between 0 and 1 to fasten the computation. Our target is following the same structure but with shifted values of a given temporal offset for 5, 10, 15, 30 and 45 minutes. Every line of the input matrix represents an input vector, see (4). As for now, pairs of input and target vectors containing NULL elements are removed before the training.

$$\begin{bmatrix} x_{s_1,t_0} & \cdots & x_{s_n,t_0} \\ \vdots & \ddots & \vdots \\ x_{s_1,t_n} & \cdots & x_{s_n,t_n} \end{bmatrix} \qquad (4)$$

To define the neighbors of each sensor we are following the idea from [12]. Only sensors from where traffic can get to the target within a given time lag are considered. We are using the Isochrone API of the Open Source routing engine GraphHopper[4] which produces a reverse flow isochrone polygon for a given time lag for each sensor. The intersecting sensors are the neighbors. We do not apply any further weighting yet as our ANN should be able to learn by itself which neighbors are having a higher impact for a future measurement at the target sensor. The resulting adjacency matrix has to be applied against every input matrix in individual trainings for each target sensor. We are using one neighbor setting where the target sensor is included and one where it is excluded.

We made an exception and took only isochrones for 5 minutes even for bigger prediction horizons because 10 minutes isochrones can already cover great areas of Dresden and, thus, include many sensors. Moreover, our isochrones are fixed and not dynamic as in [12]. The resulting adjacency matrix has to be applied against the input matrix. In the end, we came up with four different input settings to analyze the effect of including other sensors into our prediction:

FFNN$_{\text{simple}}$: Only historic values of the target sensor to predict a future value

FFNN$_{\text{NN}}$: Only historic values from nearest neighbors excluding the target sensor

FFNN$_{\text{NN+}}$: Only historic values from nearest neighbors including the target sensor

FFNN$_{\text{all}}$: Historic values from all sensors

[15] have shown that sequential information can also be passed to a FFNN by appending the temporal lags to the matrix to mimic a RNN. We are also applying this strategy in our tests using a sequence of 5 time steps (**mFFNN**) as illustrated in (5).

$$\begin{bmatrix} x_{s_1,t_0} & \cdots & x_{s_n,t_0} & x_{s_1,t_1} & \cdots & x_{s_n,t_1} & \cdots & x_{s_n,t_m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{s_1,t_n} & \cdots & x_{s_n,t_n} & x_{s_1,t_{n+1}} & \cdots & x_{s_n,t_{n+1}} & \cdots & x_{s_n,t_{n+m}} \end{bmatrix} \qquad (5)$$

We run the network with 20000 iterations and tested it with data from the two following months – August and September 2015.

[4] https://graphhopper.com/api/1/docs/isochrone/

6. RESULTS AND DISCUSSION

For evaluating of our results we are using the mean absolute error (MAE) as defined in (6) which is a common measure in research:

$$MAE = \frac{1}{N}\sum_{k=1}^{N}|\hat{y}_k - y_k| \qquad (6)$$

$\hat{y}_k$ stands for the predicted value. Table 1 shows an exemplary result for one sensor. In our case the numbers represent how far we are from the real detected occupancy. Many aspects seen here also apply to other sensors, e.g. lowest MAE when including all sensor and sequence information and highest MAE when filtering the input by the target's nearest neighbors incl. historic values of the target itself. Adding sequence information to the input matrix does not have a great beneficial impact on the predictions. But generally, we are getting very close to the actual values.

Table 1 Results for different neural network configurations and prediction time horizons.

	t5	t10	t15	t30	t45
FFNN$_{\text{single}}$	0,5042	0,5743	0,6679	0,9698	1,2970
FFNN$_{\text{NN}}$	0,6255	0,6660	0,7248	0,9831	1,0299
FFNN$_{\text{NN+}}$	1,7177	1,7157	1,7236	1,7813	1,8928
FFNN$_{\text{all}}$	0,4975	0,4933	0,5262	0,7474	1,0299
mFFNN$_{\text{single}}$	0,4324	0,5213	0,6137	0,9360	1,2771
mFFNN$_{\text{NN}}$	0,5616	0,6080	0,6667	0,9231	1,2519
mFFNN$_{\text{NN+}}$	1,6420	1,6543	1,6862	1,8126	1,9612
mFFNN$_{\text{all}}$	**0,3998**	**0,4086**	**0,4521**	**0,6405**	**0,8684**

In contrast to the results of many other papers in the field of spatio-temporal traffic forecasting, adding neighborhood information decreases the accuracy in many cases. It is obvious that an ANN works best when it is fed with input values of the whole ground truth, but we did not expect it to be worse than taking only values of the target. It would be interesting to see if an additional weighting on the spatial weight matrix could enhance the forecasting. When we will work with LSTMs a reduction of features could be necessary because of the exponential complexity of RNNs.

Figure 3 shows a comparison between predicted and detected values for three different FFNN test setups trained with a 5 minutes temporal offset.

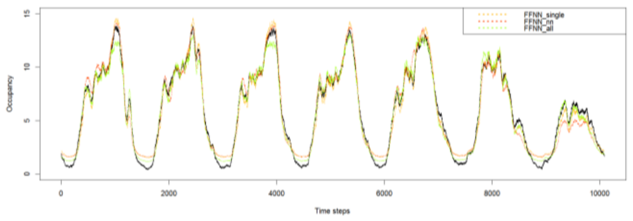

Figure 3 Difference between prediction (dotted lines) and reality (dark solid line) for one week of august

The predictions come very close during the increase and decrease periods in the mornings and evenings. The network can also figure out that traffic is different on a Sunday without having any information about the day. One can argue that our task is relatively easy to solve, as we have smoothed the input and target values. Within a short prediction horizon, the moving average of detected occupancies is not going to change a lot. Thus, we would always receive a low error. As a solution to this issue we should further aggregate more measurements and reduce the rolling mean. When looking at the spatial distribution of the prediction accuracy (see figure 4) we found that it decreases especially on main roads near the highways. This could probably be improved if the network would have information about the time of day to distinguish between the two rush hour periods.

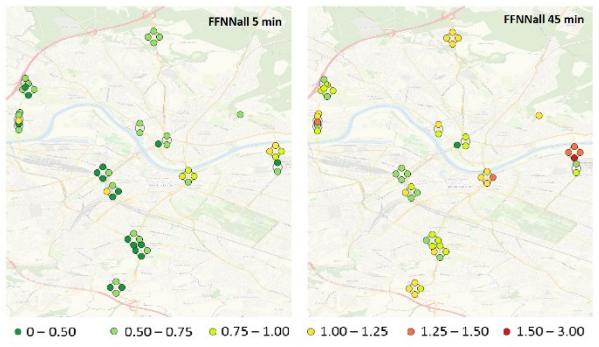

Figure 4 MAE at different sensors for the FFNN$_{all}$ setup trained on 5 minutes and 45 minutes offsets

We have also noticed differences in accuracy for different lanes on the road. Generally, the error has been higher for the inner lane of a road, probably because the occupancy tends to be more nonlinear than on the outer lane.

7. CONCLUSION

In this paper we have presented the potential of deep learning on traffic sensor data. While the usage of neural networks for short-term traffic forecasting had been used in many different studies most often the spatial dimension is not included or neglected because of a simplistic training scenario with a low number of sensors. We have worked on a sensor network that is distributed across an entire city and got the best results when we included measurements from all sensors. Including a sequence information enhanced the prediction only slightly. Thus, we have to work with RNNs, which should be superior for time series analysis because they enable to learn short and long sequences. We will also do further investigation in how to repair missing or corrupt data values.

We have started with FFNNs for comparison reasons with other ANNs, parametric and non-parametric approaches. But even the rather simple FFNN could provide very good forecasting results with low computational costs. This also raises the question if a proper traffic and travel time prediction really requires more FCD from individuals rather than extending the network of static sensors. For data privacy reasons it is important to improve algorithms for analyzing time series data of anonymous sensor networks. We will also develop strategies for implementing a continuous learning algorithm for an autonomous ITS that is able to apply navigation assistance based on the spatio-temporal findings.

8. ACKNOWLEDGMENTS

The work was supported by the Federal Ministry for Economic Affairs and Energy (BMWi) under grant agreement 01MD15001B (Project: ExCELL).

9. REFERENCES

[1] Vlahogianni, E. I., Karlaftis, M. G., & Golias, J. C. (2014). *Short-term traffic forecasting: Where we are and where we're going.* Transportation Research Part C: Emerging Technologies, 43, 3-19.

[2] Williams, B. M., & Hoel, L. A. (2003). *Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results.* Journal of transportation engineering, 129(6), 664-672.

[3] Leonhardt, A.; Steiner, A. (2012): *Instance Based Learning for Estimating and Predicting Traffic State Variables using Spatio-Temporal Traffic Patterns*. TRB 91th Annual Meeting, Washington D.C..

[4] Sun, S., Zhang, C., & Yu, G. (2006). *A Bayesian network approach to traffic flow forecasting.* IEEE Transactions on Intelligent Transportation Systems, 7(1), 124-132.

[5] Lippi, M., Bertini, M., & Frasconi, P. (2013). *Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning.* IEEE Transactions on Intelligent Transportation Systems, 14(2), 871-882.

[6] Liu, H., van Zuylen, H., van Lint, H., & Salomons, M. (2006). *Predicting urban arterial travel time with state-space neural networks and Kalman filters.* Transportation Research Record: Journal of the Transportation Research Board 1968, 99-108.

[7] Kamarianakis, Y., & Prastacos, P. (2005). *Space–time modeling of traffic flow.* Computers & Geosciences, 31(2), 119-133.

[8] Zeng, X., & Zhang, Y. (2013). *Development of Recurrent Neural Network Considering Temporal‐Spatial Input Dynamics for Freeway Travel Time Modeling.* Computer-Aided Civil and Infrastructure Engineering, 28(5), 359-371.

[9] Sepp Hochreiter und Jürgen Schmidhuber: *Long short-term memory*. In: Neural Computation. 9, Nr. 8, 1997, 1735–1780.

[10] Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015). *Long short-term memory neural network for traffic speed prediction using remote microwave sensor data.* In: Transportation Research Part C: Emerging Technologies 54, 187–197.

[11] Zhang, J., Zheng, Y., Qi, D. (2017). *Deep spatio-temporal residual networks for citywide crowd flows prediction.* In: Thirty-First AAAI Conference on Artificial Intelligence.

[12] Cheng, T., Wang, J., Haworth, J., Heydecker, B., & Chow, A. (2014). *A dynamic spatial weight matrix and localized space–time autoregressive integrated moving average for network modeling.* Geographical Analysis, 46(1), 75-97.

[13] Lipton, Z.C., Berkowitz, J., & Elkan, C. (2015). *A critical review of recurrent neural networks for sequence learning.* arXiv preprint arXiv:1506.00019.

[14] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation* (No. ICS-8506). California University San Diego La Jolla. Institute for cognitive science.

[15] Polson, N., & Sokolov, V. (2016). *Deep Learning Predictors for Traffic Flows.* arXiv preprint arXiv:160.