Spatio-Temporal Traffic Flow Forecasting on a City-Wide Sensor Network

Felix Kunde, Alexander Hartenstein and Petra Sauer

Abstract Intelligent transportation systems (ITS) all around the world are 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 but it can also be very useful for time-series analysis to predict future traffic flow. Still, the acceptance for data-driven forecasting is quiet low in productive systems of the public sector. Without enough probe data from floating cars (FCD) ITS owners feel unable to reach an 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. With this paper we prove that a modern deep learning framework is capable to operate on city-wide sensor data and produces very good results with even simple artificial neural networks (ANN). In order to forecast space-time traffic dynamics we are testing a Feed Forward Neural Network (FFNN) with different geotemporal constraints and can show where and when they have a positive but also a negative effect on the prediction accuracy.

Keywords Traffic forecasting · Time series analysis · Spatio-temporal data mining · Neural network · Deep learning

F. Kunde (✉) · A. Hartenstein · P. Sauer
Beuth University of Applied Sciences Berlin, Luxemburger Str. 10,
13353 Berlin, Germany
e-mail: fkunde@beuth-hochschule.de

A. Hartenstein
e-mail: s58380@beuth-hochschule.de

P. Sauer
e-mail: sauer@beuth-hochschule.de

I. Ivan et al. (eds.), *Dynamics in GIscience*, Lecture Notes in Geoinformation and Cartography, DOI 10.1007/978-3-319-61297-3_18

Introduction

Improvements in the machine learning domain are raising the question if an ITS can be a 100% driven by data from sensors—learning how traffic patterns evolve, which rules to apply for switching traffic lights and how to navigate traffic streams without causing more congestion. The idea is fascinating but also puts higher requirements on the quality of data. Static sensors can be unavailable or report shifted values for hours what urges the use of additional sources like floating car data (FCD) (Graser et al. 2012). On the other hand, it has to be considered if the throughput of FCD is really high enough on each trajectory at any given time to be a representative sample of the real traffic. While it is not expensive to extend fleets of GPS-equipped vehicles or to crowd source location data from smart phones (Apps, Social Media), data privacy remains a critical issue (Jeske 2015).

The ITS of Dresden (called VAMOS) uses data from around 1000 sensors installed along freeways and main roads (induction loops, infrared traffic eyes and cameras). In addition FCD from around 500 taxi cabs is map matched against a generalized routing graph and included in the calculation of travel times. Even though the system incorporates past travel times from every network segment it only generates an image of the current traffic situation. Tests have shown that messages displayed on traffic signs to inform drivers on upstream traffic congestion can get very inaccurate the more distant the sign is located from the incident (Pape and Körner 2016). Therefore, a solid traffic forecasting algorithm that can produces close predictions only on static sensor data would be a great win for the system.

In this paper we present a possible solution which is based on a Feed Forward Neural Network (FFNN). We are taking a subset of 135 double induction loops which are distributed across Dresden. To capture spatio-temporal dependencies of the sensor network we include a spatial weight matrix into our model. In next section we will provide an overview of related work that has been done in past years to improve data-driven traffic forecasting and point out some limitations. After an introduction to related work we will explain our experimental setup followed by the "Results" Section. We evaluate our findings in the Section "Discussion and Future Work" and conclude with an outlook on further research.

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 time lags (Vlahogianni et al. 2014). Lippi et al. (2013) point out that it is often difficult to compare them because of their heterogeneous setup. The applied methods are ranging from classical parametric solutions like autoregressive statistics for time series (ARIMA) (Box, Jenkins 1976) (Williams and Hoel 2003), k-Nearest neighbors on historic data sets (Leonhardt and Steiner 2012), Bayesian networks (Sun et al. 2006) to

non-parametric predictions by support vector machines (SVM) (Lippi et al. 2013) or artificial neural networks (ANN) (Liu et al. 2006).

In recent years ANNs have become popular again due to the hype around the term "Deep Learning". Complex deeply nested architectures are now computable on most modern machines. Thus, a growing research activity in using these networks for traffic forecasting can be noted. Because of their underlying algorithms ANNs are very good in detecting non-linear patterns e.g. like irregular traffic congestion. This is why they are regarded as to be superior against model-driven approaches. Two popular types of ANNs are Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). CNNs are designed to extract features from the input data using kernel functions. They operate well on fuzzy data such as images. RNNs are able to learn sequences in data, which makes them interesting for prediction use cases.

RNNs seem like a good fit for traffic forecasting and have been used by Liu et al. (2006) and Zeng and Zhang (2013). Because RNNs can only operate on a rather small sequence Hochreiter and Schmidhuber (1997) created the Long Short Term Memory (LSTM) ANN which uses internal filters (called "gates") to reduce the learning overhead that occurs in "simple" RNNs. So far, not many efforts have been made to use LSTMs for traffic forecasting (Ma et al. 2015). In the work of Zhang et al. (2017) an LSTM has been coupled with a CNN. The CNN would extract patterns such as traffic congestion from heat maps and the LSTM would learn how the patterns evolve.

Many studies have also proven the relevance of a spatial dimension to improve the accuracy of the predictions. The idea is to not only look at the historic values of the target sensor but also to incorporate data from all other sensors filtered by a spatial weight matrix to strengthen the relation between neighbours (Kamarianakis and Prastacos 2005; Cheng et al. 2014). However, in ANN-driven research the spatio-temporal models are often fallen short in terms of complexity (e.g. freeway setting, low number of sensors) compared to ARIMA-based approaches. Therefore, we combine our experiments on FFNNs with ideas from Cheng et al. (2014).

Experimental Setup

Neural Network Architecture

For further description of the network architecture we are using the same naming convention as in Lipton et al. (2015). We have implemented a FFNN using Google's Tensor Flow framework (tensorflow.org). The network consists of one output layer o and one hidden layer h. The number of input nodes i and output nodes k is defined by the number of sensors we consider as a valid input source (98% availability of measurements in the training data set). The number of hidden nodes j is set to 60 in every training session. For each layer we are using a sigmoid

activation function l (1) to produce the value v of every neuron (2). The sigmoid function $\sigma(z)$ is a classical nonlinear function and a good choice if we want to detect nonlinear patterns in our data.

$$l = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

$$v_j = l_j\left(\sum_{j'} w_{jj'}.v_{j'}\right) \tag{2}$$

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 on each iteration using backpropagation (Rumelhart et al. 1985). The weights between the neurons are adjusted by stochastic gradient descent (SGD) with a mini batch size of 20 and a learning rate of 0.01. The backpropagation algorithm subsequently calculates the derivatives of L from output nodes k (3) to hidden nodes j with respect to their corresponding activation function (4).

$$\delta_k = \frac{\partial L(\hat{y}_k, y_k)}{\partial \hat{y}_k} \cdot l'_k\left(\sum_j w_{kj} \cdot v_j\right) \tag{3}$$

$$\delta_j = l'\left(\sum_{j'} w_{jj'} \cdot v_{j'}\right) \sum_k \delta_k \cdot w_{kj} \tag{4}$$

Data Preparation

From the VAMOS data archive we have extracted three month of measurements from all double induction loops (July to September 2015). We choose this sensor group as it is most representative for the traffic in Dresden. The loops are installed only on main roads and with enough distance to traffic lights to avoid noise from waiting queues. They are detecting the speed, number of cars (occupancy), time on sensor, length of cars and gaps between cars. All variables are minutely aggregated and can be zero if there are no cars in one minute. Therefore, we are taking a 50 min moving average (MA) of the time series as input. As for now, we omit time steps where the input or output vectors contain missing values. In our test case we had to reduce the number of double induction loops to 59. Still, we got a good spatial distribution across the city (see Fig. 1).

For our test case we have started with using only the occupancy as input value which is later normalized to a range between 0 and 1. We generate an input matrix and a target matrix with the following structure, where each line represents an input vector to the neural network:

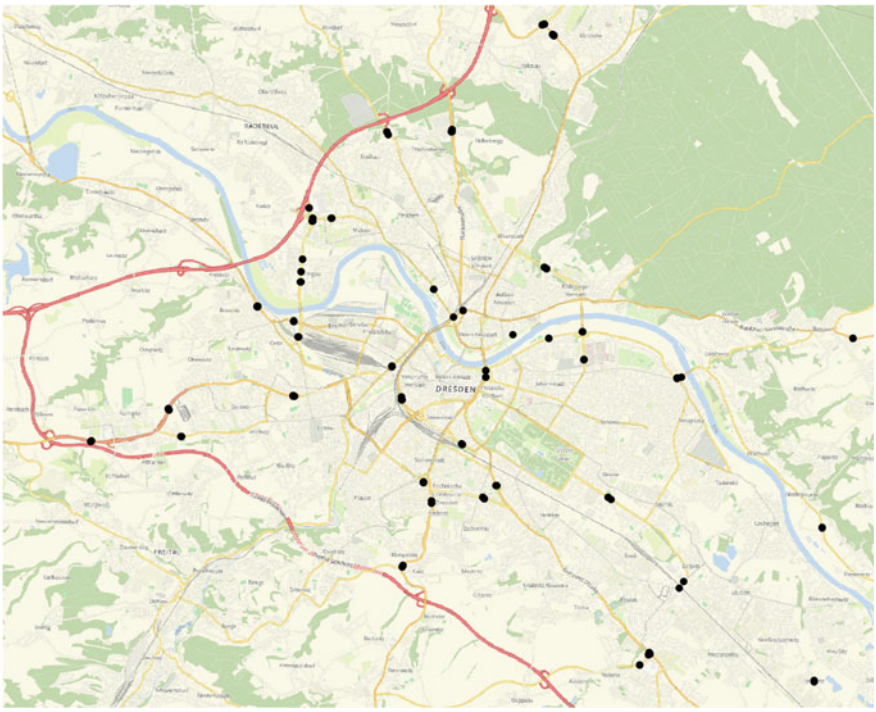

Fig. 1 Location of sensors selected for training

$$\begin{bmatrix} x_{s_1,t_0} & \cdots & x_{s_n,t_0} \\ \vdots & \ddots & \vdots \\ x_{s_1,t_n} & \cdots & x_{s_n,t_n} \end{bmatrix} \begin{bmatrix} x_{s_1,t_0+offset} & \cdots & x_{s_n,t_0+offset} \\ \vdots & \ddots & \vdots \\ x_{s_1,t_n+offset} & \cdots & x_{s_n,t_n+offset} \end{bmatrix} \tag{5}$$

Here x stands for the smoothened and normalized occupancy value detected at a sensor s at time step t where offset can be the occupancy value in the next 5, 10, 15, 30 and 45 min. Our strategy to define neighbours of each sensor is inspired by the work of Cheng et al. (2013). 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 (graphhopper.com/api/1/docs/isochrone/) which produces a reverse flow isochrone polygon for a given time lag for each sensor. The intersecting sensors are the neighbours. We do not apply any further weighting yet as our ANN should be able to learn by itself which neighbours are having a higher impact for a future state at the target sensor. We make an exception and take only isochrones for 5 min even for bigger prediction horizons because 10-minute isochrones can already cover great areas of Dresden and, thus, include many sensors. Moreover, our isochrones are fixed and not dynamic as in Cheng

et al. (2014). The resulting adjacency matrix has to be applied against the input matrix.

In the end, we came up with four different input settings to analyse the effect of including other sensors into our predictions:

- $\text{FFNN}_{\text{simple}}$: Only historic values of the target sensor to predict a future value
- FFNN_{NN}: Only historic values from nearest neighbours excluding the target sensor
- $\text{FFNN}_{\text{NN+}}$: Only historic values from nearest neighbours including the target sensor
- FFNN_{all}: Historic values from all sensors

We also want to measure possible improvements by including temporal sequences in the input matrix. Polson et al. (2016) have shown that sequential information can also be passed to a FFNN by appending the time lags to the matrix to mimic a RNN. We are also applying this strategy in four additional tests ($\text{mFFNN}_{\text{simple/NN/NN+/all}}$) using a sequence of m time steps. In our case we choose a sequence of 5 min, because it correlates with our isochrone radius.

$$\begin{bmatrix} x_{s_1,t_0} & \cdots & x_{s_n,t_0} & x_{s_1,t_1} & \cdots & x_{s_n,t_1} & \cdots & x_{s_n,t_m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{s_1,t_n} & \cdots & x_{s_n,t_n} & x_{s_1,t_{n+1}} & \cdots & x_{s_n,t_{n+1}} & \cdots & x_{s_n,t_{n+m}} \end{bmatrix} \begin{bmatrix} x_{s_1,t_{m+offset}} & \cdots & x_{s_n,t_{m+offset}} \\ \vdots & \ddots & \vdots \\ x_{s_1,t_{n+m+offset}} & \cdots & x_{s_n,t_{m+offset}} \end{bmatrix} \tag{6}$$

Training

We train our FFNN with one month of data for July 2015 and test it against the data sets from August and September 2015. For all eight FFNN architectures we train each sensor with five different temporal offsets as described above. Theoretically, around 44,600 time steps (1440 min for 31 days minus the offset) can be fed in as input. The inputs are selected randomly from the time series.

Results

For evaluating our results, we are using the mean absolute error (MAE) as defined in (7), which is a common measure in research:

$$MAE = \frac{1}{N}\sum_{k=1}^{N}|\hat{y}_k - y_k| \tag{7}$$

$\hat{y}_k$ stands for the predicted value. Table 1 shows an exemplary result for one sensor. 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 targets nearest neighbours incl. historic values of the target itself.

To fully understand the differences between given time lags, FFNN model and sensor location we feed our results into a spatial database and visualize them in a set of maps (see Fig. 2). In Dresden many double induction loops are installed in a group of four—a sensor on each lane for both sides of the road. Therefore, we style the layer with the Point Displacement feature of the open source geographic information system QGIS (qgis.org). In general, the results of all different FFNN setups are very good. But, we are colouring the sensor locations from green (low MAE) to red (slightly higher MAE) in order to better spot the differences of the trained models.

Temporal Dimension

Again, we can see that including nearest neighbours and the target produces the highest MAEs in our tests. Like the results from Table 1 adding sequence information to the input matrix does not have a great beneficial impact on the predictions. Especially for small time lags the result of simple FFNNs are comparable to the ones with sequences. On the other hand, temporally extended FFNNs are better for longer prediction horizons.

Spatial Dimension

In contrast to the results of many other papers in the field of spatio-temporal traffic forecasting, adding neighbourhood information decreased the accuracy in many cases. It's 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 ($FFNN_{single}$). Nevertheless, the $FFNN_{NN}$ setup seems to be a superior choice for greater time lags (>15 min).

Table 1 MAE for different prediction horizons for one sensor

FFNN type	t5	t10	t15	t30	t45
$FFNN_{single}$	0.5042	0.5743	0.6679	0.9698	1.2970
$FFNN_{NN}$	0.6255	0.6660	0.7248	0.9831	1.0299
$FFNN_{NN+}$	1.7177	1.7157	1.7236	1.7813	1.8928
$FFNN_{all}$	0.4975	0.4933	0.5262	0.7474	1.0299
$mFFNN_{single}$	0.4324	0.5213	0.6137	0.9360	1.2771
$mFFNN_{NN}$	0.5616	0.6080	0.6667	0.9231	1.2519
$mFFNN_{NN+}$	1.6420	1.6543	1.6862	1.8126	1.9612
$mFFNN_{all}$	0.3998	0.4086	0.4521	0.6405	0.8684

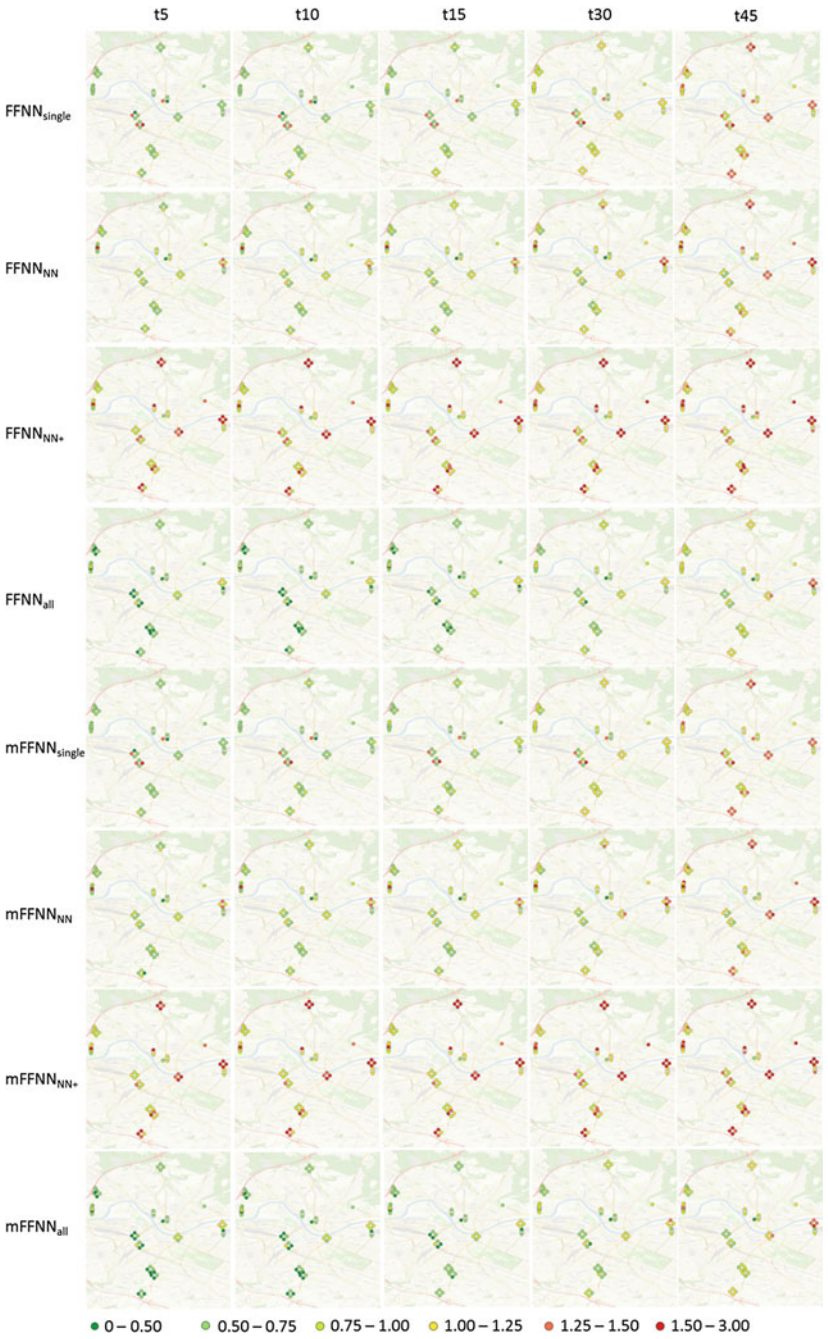

Fig. 2 MAEs of all tested sensors for different prediction horizons and different FFNN architectures

In the map series of the $FFNN_{single}$ tests we notice groups of sensors where two of them have a low MAE and the other two have a considerably higher error. As there is no styling rule about which point represents which lane we have to take a closer look. When zooming in, it can be seen that the predictions for sensors at the inner side of the road are less accurate (see Fig. 3). The pattern also occurs at other locations. It probably depends on the selected variable occupancy. The outer lane of a road has probably a higher and more regular occupancy throughout the day and night than the inner lane.

The MAE does not only vary for single groups but also across the city. For small time lags, predictions for loops on the in- and outgoing main roads of Dresden are more accurate than in the city centre. For longer time windows this pattern is turning to the opposite. In other words, traffic in the centre, which is probably more non-linear than on a main road leading to a freeway, is better predictable for FFNNs trained on bigger temporal offsets. But, these FFNNs work less accurate on linear traffic patterns compared to FFNNs trained on small time lags.

Discussion and Future Work

Our approach shows the best performance when we apply all available spatial and temporal information to the neural network. This strengthens the general assumption that neural networks are very good at learning spatio-temporal patterns from a huge amount of data. As for now, we have only used a rather simple measure—the MAE—to validate our results. In general, our MAE is very low in all our test runs. When looking at the average traffic distribution of one sensor over one year (see Fig. 4) we can see that it follows a quite regular trend (grey area is the standard deviation). In the morning hours between 5 and 8 o'clock there is a strong increase of traffic volume. Then the graph follows a valley of moderate traffic density during the day with a less steep curve. After the evening rush hour the traffic constantly decreases to a minimum at night. In this example, we can extract five different rather unique temporal traffic patterns which also occur at many other sensors of our sample with varying local maxima. Given a small temporal offset of just a few minutes and the great smoothing of 50 min moving average we produce many

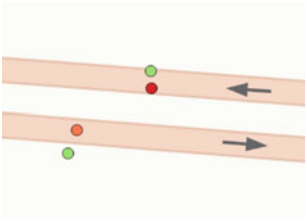

Fig. 3 MAEs of sensors on different street lanes (same colour scale as in Fig. 2)

examples that are very easy to predict. The figure also shows a curve of traffic at January 1st which is, of course, very different from the yearly average. In our future work we will focus on how the predictive power of the model will keep up especially in these situations.

Comparing the prediction with the actual observed value on a time line helps to understand which periods were easier to predict than others. This is done in Fig. 5 for one week of August 2015 (time 0 = Monday midnight). The solid line represents the target, the dotted lines are showing the prediction results of 3 different FFNNs. It can be seen that our predictions come very close to the real value, especially when the amount of traffic is increasing in the morning and decreasing in the evening. The FFNN has only problems to predict zero values during night time and sometimes it over- or underestimates the traffic during the peak periods. It seems that the FFNN can either not distinguish between the two rush hour periods or it relies on the patterns learned from the previous month. This can be solved by adding information about daytime to the input vector.

What is also really interesting to see in Fig. 5 is the ability of the network to learn the differences between weekends and weekdays. This can be a hint that filtering the input by a spatial weight matrix introduces a bias that—while being reasonable in traffic theory—might not be necessary. It would be interesting to see how other ANN architectures behave given the same input than in our experiments. Especially, when using RNNs the model complexity and training time will grow exponentially. Therefore, using a sparse matrix as input can be a mandatory compromise in order to have a scalable solution. Using FFNNs did not cause any performance problems. The training and testing of a given FFNN setup for one sensor only took a few minutes.

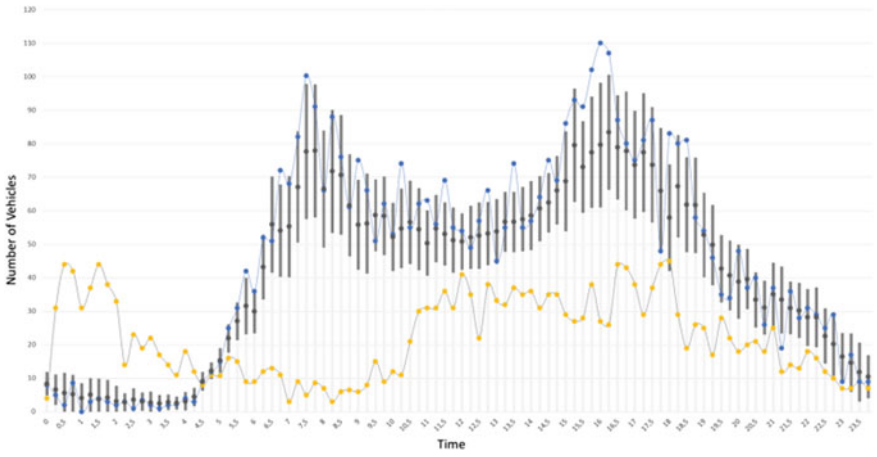

Fig. 4 Number of vehicles for one sensor on two wednesdays in 2014. The *yellow line* depicts January 1st of 2014 and the *blue line* shows a typical Wednesday (April 16th of 2014). Additionally, the *grey rectangles* visualize the standard deviation for number of vehicles in one year (occupancy)

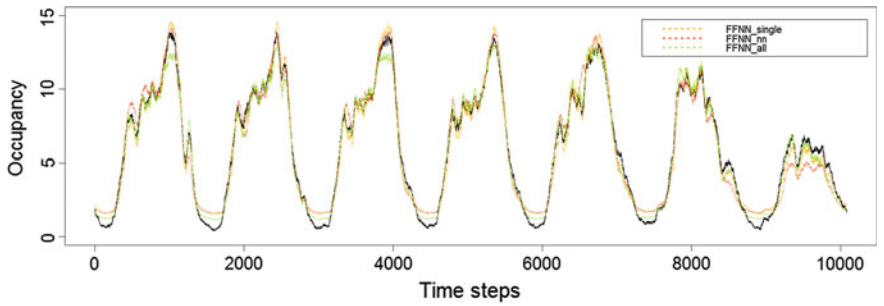

Fig. 5 Predicted occupancy (*dotted line*) versus observed values (*osolid line*)

This question would also be essential when installing a continuous learning where newly detected data is included in cyclic training sessions to adapt to recent changes in the urban traffic. It needs to be evaluated how long a trained model can be used for prediction and how often it needs to be updated. If an ITS would react on behalf of the predictions e.g. by displaying proper messages to drivers through signs or smartphone apps, it could generate new traffic patterns the ANN hasn't see before. This could lead to false predictions and thus frustrated drivers. A regular training from scratch could produce too much overhead when using complex ANNs. Srivastava et al. (2014) have suggested a so called Dropout mechanism, where an existing model is destroyed partly and rearranged in a new training session with updated inputs.

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 are working on a sensor network that is distributed across an entire city. Therefore, we can see that the accuracy of our predictions is varying by location, prediction horizon and model selection.

So far, we can make the following statements: (i) Filtering the input by nearest neighbours puts too much bias into the neural network. Still, such a filtering can be relevant when testing with more complex ANNs. (ii) Including sequence information in the FFNN input generally improves the accuracy, which is why we will work with RNNs (LSTMs) in the future. (iii) For short-term predictions with small offsets simple FFNNs already provide very good results. Bigger gaps between predicted and observed values can sometimes be found for morning peaks and during nights. (iv) FFNNs trained on short time lags produce a higher error for

non-linear traffic patterns than FFNNs trained on longer temporal offsets and vice versa.

It is planned that the developed algorithms (available under: https://github.com/MAGDa-BeuthHS/dlsd) will be tested against real-time data from the ITS of Dresden. We also plan to apply our algorithm on historic travel times of network links. By doing so we should be able to better capture spatio-temporal dynamics. 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. We believe that for data privacy reasons it is important to improve algorithms for analysing time series data of anonymous sensor networks.

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

References

Box, G. E., & Jenkins, G. M. (1976). *Time series analysis: Forecasting and control*. San Francisco: Holden-Day.

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.

Graser, A., Dragaschnig, M., Ponweiser, W., Koller, H., Marcinek, M., & Widhalm, P. (2012). FCD in the real world system capabilities and applications. In *Proceedings of the 19th ITS World Congress*, Vienna, Austria.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Jeske, T. (2015). Sicherheit und Datenschutz in nicht-interaktiven Crowdsourcing Szenarien. Dissertation. Technische Universität Hamburg-Harburg. Available at: https://tubdok.tub.tuhh.de/bitstream/11420/1249/1/Dissertation_Tobias_Jeske.pdf (in german).

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

Leonhardt, A., & Steiner, A. (2012). Instance based learning for estimating and predicting traffic state variables using spatio-temporal traffic patterns. In *TRB 91th Annual Meeting*, Washington, D.C.

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.

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

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*, 99–108.

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. *Transportation Research Part C: Emerging Technologies, 54,* 187–197.

Pape, S., & Körner M. (2016). Verkehrslageprognose unter Berücksichtigung der dynamischen Kapazitäten an LSA-abhängigen Knotenpunkten zur qualitativen Aufwertung der

Verkehrslageinformation im Verkehrsmanagementsystem VAMOS. *25. Verkehrswissenschaftliche Tage 16. und 17. März 2016 in Dresden*. (in german).

Polson, N., & Sokolov, V. (2016). Deep learning predictors for traffic flows. arXiv preprint arXiv:1604.04527

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

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15,* 1929–1958.

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.

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.

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.

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.

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*.