Exploiting Recurrent Neural Networks for Gate Traffic Prediction

Fabio Fumarola and Pasqua Fabiana Lanotte

Abstract Traffic information plays a significant role in everyday activities. It can be used in the context of *smart traffic management* for detecting traffic congestions, incidents and other critical events. While there are numerous ways for drivers to find out where there is a traffic jam at a given moment, the estimation of the future traffic is not used for proactive activities such as ensuring a smoother traffic flow and to be prepared for critical situations. Therefore traffic prediction is focal both for public administrations and for the Police Force in order to do resource management, network security and to improve transportation infrastructure planning. A number of models and algorithms were applied to traffic prediction and achieved good results. Many of them require the length of past data to be predefined and static, do not take into account dynamic time lags and temporal autocorrelation. To address these issues in this paper we explore the usage of Artificial Neural Networks. We show how Long Short-Term Memory (LSTM), a particular type of Recurrent Neural Network (RNN), can overcome the above described issues. We compare LSTM with a standard Feed Forward Neural Network (FDNN), showing that the proposed model achieves higher accuracy and generalises well.

1 Introduction

With the increasing amount of traffic information collected through automatic number plate reading systems (NPRS), it is highly desirable for public administrations and the *Police Force* to have surveillance tools to estimate traffic parameters such as the number of car passing through a plate or their average speed to detect traffic congestions, incidents and other critical events. For example, commercial traffic data providers, such as Bing maps Microsoft Research [13], rely on traffic flow data, and

F. Fumarola (✉) · P. F. Lanotte
University of Bari Aldo Moro, Bari, Italy
e-mail: fabio.fumarola@uniba.it

P. F. Lanotte
e-mail: pasqua.lanotte@uniba.it

F. Leuzzi and S. Ferilli (eds.), *Traffic Mining Applied to Police Activities*, Advances in Intelligent Systems and Computing 728,
https://doi.org/10.1007/978-3-319-75608-0_11

machine learning to predict speeds given road segment. Real-time (15–40 min) forecasting gives travellers the ability to choose better routes and authorities the ability to manage the transportation system. While, for the Police Force it can be useful to plan surveillance activities. The predictability of network traffic parameters is mainly determined by their statistical characteristics and by the fact that they present a strong correlation between chronologically ordered values. Network traffic is identified by: self-similarity, long-range dependence and a highly nonlinear nature (insufficiently modelled by Poisson and Gaussian models).

Several methods have been proposed in the literature for the task of *traffic forecasting*, which can be categorised into: linear prediction and nonlinear prediction. The most widely used traditional linear prediction methods are: a) the ARMA/ARIMA model [4, 5, 8]. The most common nonlinear forecasting methods involve Artificial Neural Networks (ANNs) [1, 4, 6]. The experimental results from [3] show that nonlinear traffic prediction based on ANNs outperforms linear forecasting models (e.g. ARMA, ARAR, HW) which cannot meet the accuracy requirements.

Also if it has been proved that ANNs achieve the best results, deciding the best ANN architecture for the task is still a daunting task. The best architecture can be a compromise between the complexity of the solution, characteristics of the data and the desired prediction accuracy. Most of the times a simple Feed Forward Deep Neural Network (FFDNN) can achieve good results. But, we are going to show that by taking into account the implicit temporal characteristics of the analysed problem we can achieve better results.

Unlike feed forward deep neural networks (FFNNs), Recurrent Neural Networks (RNNs) have cyclic connections over time. The activations from each time step are stored in the internal state of the network to provide a temporal memory. This capability makes RNNs better suited for sequence modelling tasks such as time series prediction and sequence labeling tasks. However, RNNs suffer the well know problem of the *vanishing gradient* [7], that happens when multiplying several small values from the temporal memory. This makes RNNs not able to learn long temporal dependencies. On the contrary, Long Short-Term Memory (LSTM) [7] addresses the vanishing gradient problem of conventional RNNs by using an internal memory, a *carry* and a *forget* gate to decide when keep/forget the information stored in its internal memory. RNNs and LSTMs have been successfully used for handwriting recognition [10], language modelling [12], speech to voice [2] and other classification and prediction tasks.

In this paper we present a deep neural network (DNN) architecture based on LSTM to forecast hour by hour the number of vehicles passing through a gate. We build a general model that is capable of predicting the next 24 hours of traffic basing on the past observed data. Moreover, the model is capable of abstracting on the gate number and on the temporal component. Thus, it can do predictions for variable time ranges. We show the advantages of the proposed architecture in modelling temporal correlations with respect to a FFDNN in term of mean square errors. In remainder of the paper we discuss: RNN and LSTM, the proposed architecture for the task, the experimental results and we draw conclusions and discuss on future works.

2 RNN and LSTM

Recurrent Neural Network. Feed Forward Neural Networks treat each input as independent with respect the previous ones. They start for scratch for every input not taking into account possible correlation stored in consecutive observations of the input. On the contrary, RNNs address this issue, by using a loop (see Fig. 1) in them that allows information to flow.

Given an input-length l, the loop in the RNN unrolls the network *l-times* and creates multiple copies of the same network, each passing its computed state value to a successor (see Fig. 2).

The simple architecture of RNN has an input layer **x**, hidden layer **h** and output layer **y**. At each time step t, the values of each layer are computed as follows:

$$h_t = f(U_{x_t} + W_{h_{t-1}}) \tag{1}$$

$$y_t = g(Vh_t) \tag{2}$$

where **U**, **W** and **V** are the connection weight matrices in RNN, and $f(z)$ and $g(z)$ are *sigmoid* and *softmax* activation functions.

Long Short-Term Memory. Long short-term memory (LSTM) [7] is a variant of RNN which is designed to deal with the gradient vanishing and exploding problem when learning with long-range sequences. LSTM networks are the same as RNN, except that the hidden layer updates are replaced by memory cells. Basically, a memory cell unit is composed of three multiplicative gates that control the proportions of information to forget and to pass on to the next time step. As a result, it is better for exploiting long-range dependency data. The memory cell is computed as follows:

$$i_t = \sigma(W_i h_{t_1} + U_i x_t + b_i) \tag{3}$$

$$f_t = \sigma(W_f h_{t_1} + U_f x_t + b_f) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_c h_{t-1} + U_c x_t + b_c) \tag{5}$$

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o) \tag{6}$$

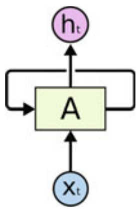

Fig. 1 Loops in Recurrent Neural Networks

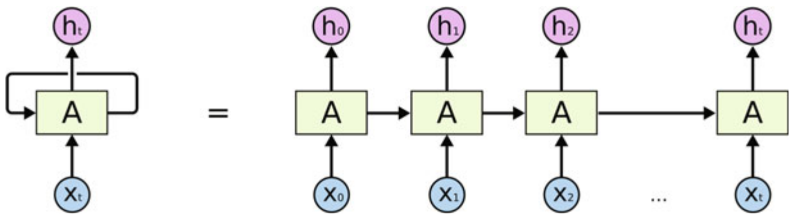

Fig. 2 Unrolled Recurrent Neural Networks

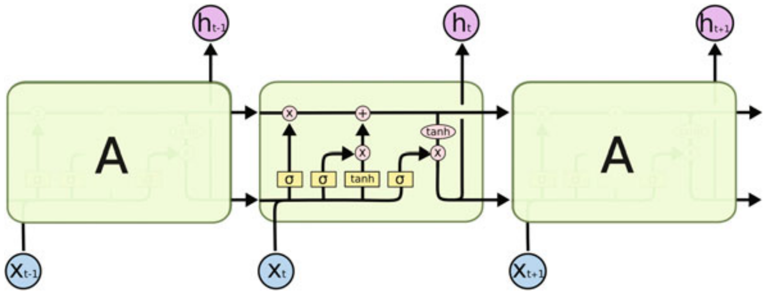

Fig. 3 The repeating module in an LSTM contains four interacting layers

$$h_t = o_t \odot tanh(c_t) \tag{7}$$

where σ is the element-wise sigmoid function and $\odot$ is the element-wise product, **i, f, o** and **c** are the input gate, forget gate, output gate and cell vector respectively. W_i, W_f, W_c, W_o are connection weight matrices between input **x** and gates, and U_i, U_f, U_c, U_o are connection weight matrices between gates and hidden state h. While, b_i, b_f, b_c, b_o are the bias vectors. Figure 3 shows how the information carried in the memory cell are modified and passed out the next LSTM cell.

3 Gate Traffic Prediction

In this section we describe the use of a LSTM to extract the temporal feature of the network traffic and predict hour by hour the number of vehicles passing through gates. This architecture can deeply model mutual dependence among gate measure entries in various time-slots.

3.1 Problem Statement

Let G be the list of the gates in a roads network, and T the time dimension, we can collect data from the gates at each timestamp t_i. Each measure contains information related to the gate number, the number of cars pass through, and the actual timestamp.

We can represent the data as a matrix of shape $L \times M$, where L is the number of measures and M is the number of features recorded. Given a list of consecutive measures for a gate we can model them as a time-series. The main challenge here is how to sample properly the batches of data to feed the LSTM for the training phase.

3.2 Modelling the Input for the LSTM

As input for the LSTM we draw a sliding window of measures extracted from the same gate. Each batch has:

- a **batch size**: the number of observations to generate
- a **sample size**: the number of consecutive observations to use as features
- a **predict step**: the position of the observation to use as target value for the prediction task.

Figure 4 shows an example of measure with: batch size equal to 3, sample size equal to 10 and predict step with value 1.

During the training we feed the LSTM with batches of *sample size* larger than the one showed in the example above. In particular, we are interested to predict the traffic state for the next 24 hours, thus we set this as fixed value for the evaluation.

3.3 Performance Metric

To quantitatively assess the overall performance of our LSTM model, Mean Absolute Error (MAE) is used to estimate the prediction accuracy. MAE is a scale dependent metric which quantifies the difference between the forecasted values and the actual values of the quantity being predicted by computing the average sum of absolute errors:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}(|y_i - \hat{y}|) \tag{8}$$

where y_i is the observed value, $\hat{y}_i$ is the predicted value and N represents the total number of predictions.

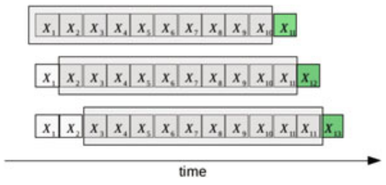

Fig. 4 Example of sliding window batch

4 Experimental Results

In order to evaluate the proposed approach, we performed an experimental evaluation on **Trap-2017**[1] dataset. This dataset contains transits recorded using several gates. It stores measures from the transits of a limited area of Italy, in which gates are homogeneously distributed. The plates are coherently anonymised (a plate is always referred using the same ID). Data records contains: *plate id, gate number, street line number, timestamp, nation of the car*. All The data are split into a file per day and the whole dataset contains measured sampled for the 2016, which are in total 155.586.309 observations.

Since our goal to create a model to predict the number of cars flowing through a specific gate hour by hour, we preprocessed the data to obtain the following features: *gate number, hour, day, month, week-day, and the count of transit in 60 min range*. After this data transformation step the size of the dataset was reduced to 222.115 observations. This is mostly because we aggregate the measures to 60 min, if data was aggregated each 15 min we would have a size four times larger.

We compare results for the proposed LSTM with respect a Feed Forward Deep Neural Network architecture (FFDNN). *Mean Square Error* (see Eq. 9) is used as performance measure. To implement both the models we used Keras[2] Deep Learning Library. The first architecture is composed by 4 LSTM layers with 64, 128, 64 and 1 neurons per layer, while the second uses 4 Dense Layers with 64, 128, 64 and 1 neurons. As activation functions we used Rectified Linear Unit (ReLU) [9], while as optimizer RMSprop [11]. The dataset is split in 70%, 10%, 20% for training, validation and test.

Mean Square Error (MSE) (Eq. 9) is used as loss function for the optimizer. MSE is a scale dependent metric which quantifies the difference between the forecasted values and the actual values of the quantity being predicted by computing the average sum of squared errors:

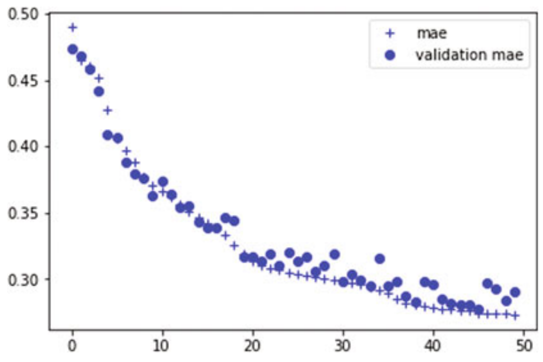

Fig. 5 Training and validation mean absolute error for the FFDNN

[1]https://trap2017.poliziadistato.it/.

[2]http://keras.io/.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2 \tag{9}$$

where y_i is the observed value, $\hat{y}_i$ is the predicted value and N represents the total number of predictions.

The FFDNN and the LSTM models were trained for 50 epochs. Figures 5 and 6 present training and validation MAE for the FFDNN. From the results is interesting to see that the architecture converge nicely without overfitting. After the training the minimum MAE for the validation set is 0.29, while for the test set is 0.287.

On the contrary, Figs. 7 and 8 present the results for the LSTM. Here we can note that the convergence rate is faster than the one observed for the FFDNN network. Finally, it achieves 0.121 and 0.134 of MAE score for the validation and test set respectively, which is half the of the value obtained for the FFDNN.

In Figs. 9 and 10 we can see an example of prediction done by the two models. Here we can note that the prediction made by the LSTM is smoother than the one made by the FFDNN.

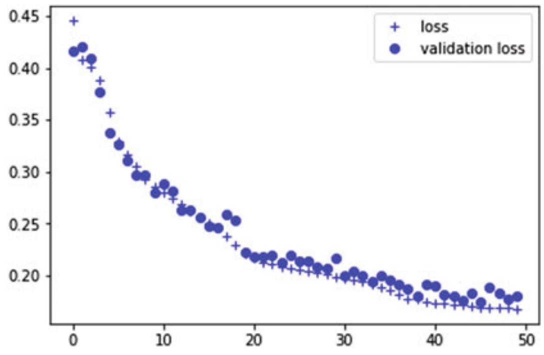

Fig. 6 Training and validation loss for the FFDNN

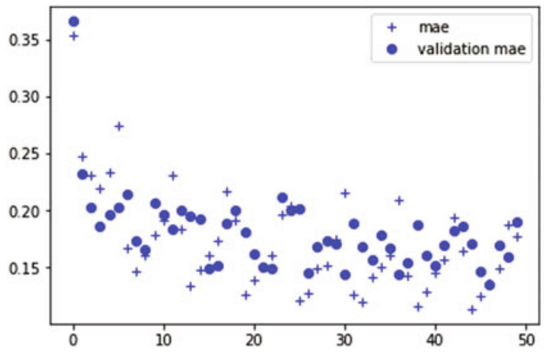

Fig. 7 Training and validation mean absolute error for the LSTM

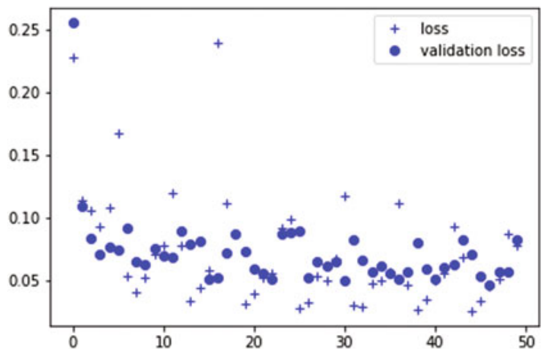

Fig. 8 Training and validation loss for the LSTM

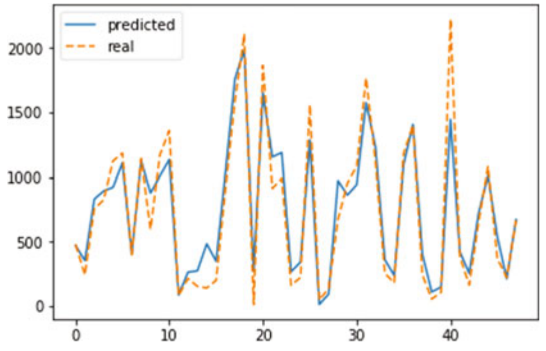

Fig. 9 Prediction for day 20 for the FFDNN

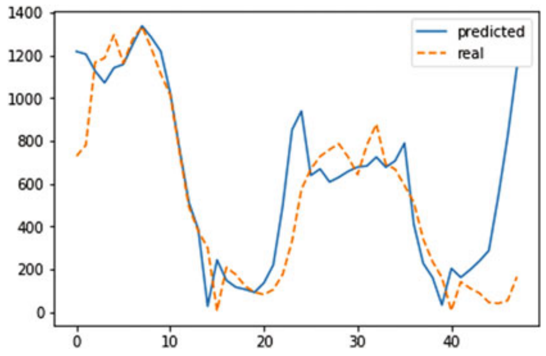

Fig. 10 Prediction for day 20 for the LSTM

5 Conclusions and Future Works

In this paper, we have presented a new neural network architecture based on LSTM for gate traffic prediction. Experiments results show that the proposed architecture performs well on the Trap-2017 dataset in terms of mean absolute error, and that it obtains better result with respect to a classical FFDNN. As future work we will run our algorithm on different datasets and do an extensive search for hyper-parameters tuning.

References

1. Abdennour, A. 2006. Evaluation of neural network architectures for mpeg-4 video traffic prediction. *IEEE Transactions on Broadcasting* 52 (2): 184–192.
2. Arik, Sercan Ömer, Mike Chrzanowski, Adam Coates, Greg Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Jonathan Raiman, Shubho Sengupta, and Mohammad Shoeybi. Deep voice: Real-time neural text-to-speech. *CoRR*, abs/1702.07825, 2017.
3. Barabas, M., G. Boanea, A. B. Rus, V. Dobrota, and J. Domingo-Pascual. 2011. Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition. In *2011 IEEE 7th International Conference on Intelligent Computer Communication and Processing*, 95–102.
4. Cortez, Paulo, Miguel Rio, Pedro Sousa, and Miguel Rocha. 2007. *Topology Aware Internet Traffic Forecasting Using Neural Networks*, 445–454. Berlin, Heidelberg: Springer.
5. Dai, J., and J. Li. 2009. Vbr mpeg video traffic dynamic prediction based on the modeling and forecast of time series. In *2009 Fifth International Joint Conference on INC, IMS and IDC*, 1752–1757.
6. Dharmadhikari, V.B., and J.D. Gavade. 2010. An nn approach for mpeg video traffic prediction. In *2010 2nd International Conference on Software Technology and Engineering*, vol. 1, pp. V1–57–V1–61.
7. Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9 (8): 1735–1780.
8. Joshi, Manish, and Theyazn Hassn Hadi. 2015. A review of network traffic analysis and prediction techniques. *CoRR*, abs/1507.05722.
9. Nair, Vinod, and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, 807–814, USA, Omnipress.
10. Stuner, Bruno, Clément Chatelain, and Thierry Paquet. 2016. Cohort of LSTM and lexicon verification for handwriting recognition with gigantic lexicon. *CoRR*, abs/1612.07528.
11. Tieleman, T., and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.
12. Verwimp, Lyan, Joris Pelemans, Hugo Van hamme, and Patrick Wambacq. 2017. Character-word LSTM language models. *CoRR*, abs/1704.02813.
13. Zhang, Junbo, Yu Zheng, and Dekang Qi. 2016. Deep spatio-temporal residual networks for citywide crowd flows prediction. AAAI 2017.