PROJECT REPORT: DEEP LEARNING FOR TRAFFIC PREDICTION

Inon Peled

Machine Learning for Mobility Group,
Department of Management Engineering,
DTU, Technical University of Denmark

ABSTRACT

Given traffic data for several roads around Nørre Campus in Copenhagen, Denmark, we compare the prediction accuracy of Neural Networks vs. common baseline models. For Neural Networks, we start from a simple Recurrent Neural Network (RNN) which uses a Fully Connected, Long Short-Term Memory (FC-LSTM) unit. We then gradually enhance the RNN architecture, and eventually obtain the best prediction accuracy when using Convolutional LSTM (Conv-LSTM), noticeably outperforming all baseline models. All our models perform better in middle-of-roads than in junctions.

1. INTRODUCTION

Accurate prediction of traffic conditions enables reliable planning of travel times, early detection of traffic congestion, and effective response by road practitioners. These benefits thus also have positive socio-economic impact.

In this work, we face the challenge of predicting traffic speeds and flows in roads surrounding Nørre Campus – a campus of the University of Copenhagen in Denmark's capital city. To this end, we design and implement several Neural Network architectures, which outperform common baseline models for traffic prediction.

We carry out the implementation in Python, mainly using packages scikit-learn and keras. The code for this work is available in github. This report is accompanied by a synopsis in the form of a Google Slides presentation.

2. DATA DESCRIPTION AND PREPARATION

The dataset consists of traffic information for 294 unique places on roads around Nørre Campus in Copenhagen, Denmark, as described in figure 1. The data was collected by Google from Android devices between 01-Jan-2015 00:00 and 29-Jun-2015 23:59, i.e. the first half of 2015. This time period is partitioned as consecutive, 5 minute long intervals, namely 01-Jan-2015 00:00, 01-Jan-2015 00:05, . . . , 29-Jun-2015 23:55.

For each place p and time interval t, the dataset contains the mean speed in km/hr, and the mean flow decile in $\{0, 1, \dots, 9\}$, as measured in p during t. For example, a flow value of 4 means that the average flow in p during t is greater than at least 40% of the average flows in all other places during t, and smaller than at least 50% of them. The dataset is therefore somewhat noisy: it is aggregated, and some time intervals are missing for various places. Moreover, whereas speed is continuous and pertains locally to only one place, flow decile is discrete and is relative to the global road network.

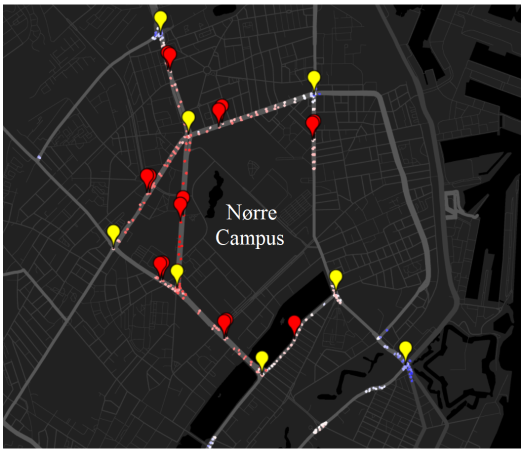


Fig. 1: Mean speed in each of the 294 unique places available in the dataset. The color scale is red for higher values, blue for lower values. Yellow markers indicate selected junctions, and red markers indicate selected middle-of-roads.

From here on, we abbreviate "mean flow decile" and "mean speed" simply as "flow" and "speed", respectively. For each place p, we let S_p denote the time series of speeds for p, and let F_p denote the time series of flows for p. Finally, we partition the data into train and test sets: all our models use January to May data for training, while the prediction quality of all our models is tested on June data.

Because traffic in nearby places is often highly correlated, and in the interest of decreasing computation times, we narrow down the data to several selected places, as also described in figure 1. The selected places are partitioned into two groups, which we study separately: junctions and middle-of-roads (both traffic directions). Because traffic tends to flow more freely in middle-of-roads than in junctions, we may anticipate that prediction models perform better in middle-of-roads than in junctions.

2.1. Data Cleaning

Before processing the data, we clean it as following. We remove speed outliers above 110 km/hr, as the speed limit in the roads around Nørre Campus is much lower. Then, as is common for time series data, we detrend the speeds. That is, for each place p, day $d \in \{\text{Sunday}, \text{Monday}, \dots, \text{Saturday}\}$, and time of day $h \in \{00{:}00, 00{:}05, \dots, 23{:}55\}$, we compute the average speed $\mu_{p,d,h}$ and the standard deviation $\sigma_{p,d,h}$ in the train set, and then transform S_p as

$$S'_p = (S_p - \mu_{p,d,h}) / \sigma_{p,d,h} \qquad (2.1)$$

Thus whereas speeds in S_p are non-negative, detrended values in S'_p can be negative. To complete the cleaning, we interpolate missing time intervals in both S'_p and F_p, and round all interpolated flows to the nearest integer in $[0..9]$.

Finally, we wish to convert the clean data into input for the prediction models. To this end, we convert each time series to a matrix of feature vectors, so that the i'th row vector corresponds to the i'th time interval. The vector for time interval t_i consists of the target variable – i.e. the speed or the flow during t_i – followed by the values for $t_{i-1}, t_{i-2}, \dots, t_{i-12}$, namely the values for the past 60 minutes.

2.2. From Output to Prediction

Our models for speed regression take as input detrended speeds, hence they also output detrended speeds. For measuring speed prediction accuracy, we thus first re-trend the output – namely apply transformation 2.1 in reverse – and remove all interpolated time intervals.

Our models for flow regression treat flow as a continuous variable. Thus for measuring flow prediction accuracy, we first round the output to the nearest integer, and remove all interpolated time intervals. In all our experiments, every such rounded value ended in $\{0, 1, \dots, 9\}$, i.e. the original domain for flow.

In fact, because the categories for flow are ordinal – namely $0 < 1 < \dots < 9$ – we also checked whether flow prediction improves when ordinal regression is used. For Neural Networks, we used the ordinal regression scheme described in [1], so that: input $k \in \{0, \dots, 9\}$ is provided as a binary vector comprising of $k+1$ ones followed by $9-k$ zeros ; the output layer comprises of 10 sigmoid neurons ; and output $[o_0, \dots, o_9]$ is interpreted as $\max\{i | o_0, \dots, o_i > 0.5\}$. However, the difference in prediction accuracy compared to rounding turned out to be of order $E-3$, hence we decided stick with the simpler scheme of rounding.

3. BASELINE

As baseline for comparison of prediction quality, we build three candidate models: Naive Copy, Historical Average, and Linear Regression. We next describe how each of these models predicts the target variable v_i (speed or flow) at time interval t_i, for any place.

1. *Naive Copy* simply predicts $\tilde{v}_i = a_{i-1}$.
2. *Hisotrical Average* predicts $\tilde{v}_i$ as the average of all $a_{i-1}, a_{i-2}, \dots \in A^{(p)}$.
3. *Linear Regression* (LR) predicts

   $$\tilde{v}_i = \beta_0 + \beta_1 v_{i-1} + \dots + \beta_{i-12} v_{i-12}$$

   Where the β's are real-valued parameters, which LR learns during training.

The only candidate baseline model which requires training is thus LR. The quality of each candidate model is measured for each place group $G \in \{\text{middle-of-roads}, \text{junctions}\}$, as following. For $p \in G$, let $V_1^{(p)}, \dots, V_k^{(p)}$ denote the vectors in the test set for p, and let $A^{(p)} = a_1^{(p)}, \dots, a_k^{(p)}$ denote the actual values of the target variable, respectively. Furthermore, let $\tilde{V}^{(p)} = \tilde{v}_1^{(p)}, \dots, \tilde{v}_k^{(p)}$ denote the respective predictions of the model for the target variable. We summarize the prediction quality of the model in four well-known measurements:

1. Root Mean Squared Error (lower is better):

   $$RMSE = \frac{1}{|G|} \sum_{p \in G} \left(\sqrt{\frac{1}{k} \sum_{i=1}^{k} \left(a_i^{(p)} - \tilde{v}_i^{(p)} \right)^2} \right)$$

2. Mean Absolute Error (lower is better):

   $$MAE = \frac{1}{|G|} \sum_{p \in G} \left(\frac{1}{k} \sum_{i=1}^{k} \left| a_i^{(p)} - \tilde{v}_i^{(p)} \right| \right)$$

3. Pearson Correlation Coefficient (higher is better):

   $$\rho = \frac{1}{|G|} \sum_{p \in G} \left(\frac{\text{Cov}\left(A^{(p)}, \tilde{V}^{(p)} \right)}{\sigma\left(A^{(p)} \right) \sigma\left(\tilde{V}^{(p)} \right)} \right)$$

   Where Cov denotes covariance, and σ denotes standard deviation.

4. Coefficient of Determination (higher is better):

   $$R^2 = \frac{1}{|G|} \sum_{p \in G} \left(1 - \frac{\sum_{i=1}^{k} \left(a_i^{(p)} - \tilde{v}_i^{(p)} \right)^2}{\sum_{i=1}^{k} \left(a_i^{(p)} - \bar{A}^{(p)} \right)^2} \right)$$

   Where $\bar{A}^{(p)}$ is the mean of $A^{(p)}$.

	Place Group	HistAvg	LR	Copy
$RMSE$	Middle	7.913	**4.758**	5.241
	Junction	10.267	**6.719**	7.501
MAE	Middle	5.370	**2.477**	3.0178
	Junction	7.348	**3.893**	4.359
ρ	Middle	0.332	**0.829**	0.808
	Junction	0.173	**0.768**	0.749
R^2	Middle	0.099	**0.674**	0.606
	Junction	0.017	**0.579**	0.476

Table 3.1: Performance of candidate baseline models for speed only. The best value in each row is highlighted in bold.

	Place Group	HistAvg	LR	Copy
$RMSE$	Middle	1.741	**0.568**	0.583
	Junction	1.941	**0.772**	0.802
MAE	Middle	1.393	**0.274**	0.282
	Junction	1.572	**0.468**	0.486
ρ	Middle	0.024	**0.943**	0.941
	Junction	0.0153	**0.913**	0.908
R^2	Middle	negative	**0.888**	0.882
	Junction	negative	**0.831**	0.816

Table 3.2: Performance of candidate baseline models for flow only. The best value in each row is highlighted in bold.

Tables 3.1 and 3.2 summarize the prediction quality of all three candidate baseline model, while figure 2 visually compares their RMSE. All candidate baseline models perform worse in junctions than in middle-of-roads, as we expected. Linear Regression performs consistently the best, while Naive Copy is nearly as good as LR, which suggests that traffic flows and speeds tend to repeat themselves over consecutive, short time intervals. Finally, Historical Average turns out to be a rather unreliable prediction model for our traffic data.

We thus keep only Linear Regression as the baseline for later comparison with the Neural Networks. Next, we check whether the prediction quality of LR improves if it is trained on speeds and flows together. It makes sense to try so, because speeds and flows are usually correlated, e.g. high congestion leads to low flow and low speed. However, table 3.3 shows that LR performs essentially the same whether or not it learns speeds and flows are together.

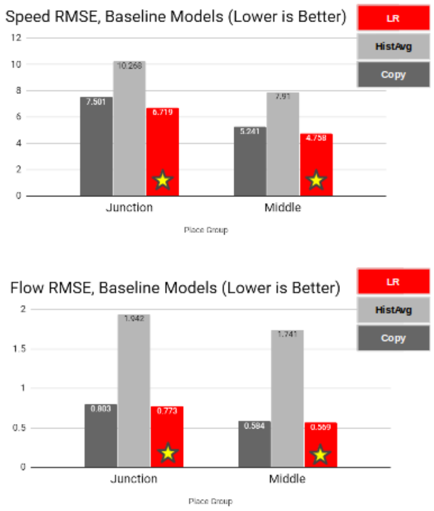


Fig. 2: Visual comparison of RMSE of the three baseline candidate models. A star marks the best value in each case.

	Place Group	Speed	Flow
$RMSE$	Middle	4.757 (-0.001)	0.546 (-0.022)
	Junction	6.714 (-0.005)	0.735 (-0.043)
MAE	Middle	2.477 (0)	0.325 (+0.051)
	Junction	3.893 (0)	0.519 (+0.051)
ρ	Middle	0.829 (0)	0.947 (+0.003)
	Junction	0.769 (+0.001)	0.921 (+0.008)
R^2	Middle	0.674 (0)	0.896 (+0.008)
	Junction	0.579 (0)	0.846 (+0.015)

Table 3.3: Performance of Linear Regression when speeds and flows are provided together. In parenthesis is the difference from the performance of LR when speeds and flows are provided separately.

4. DEEP NEURAL NETWORKS

In this section, we'll study Deep Neural Networks (NN) for prediction of speeds and flows. We're interested in two research questions: 1) Are NN also sensitive to the differences between middle-of-roads and junctions? 2) how much better can NN outperform the baseline?

Our basic architecture of choice is Recurrent Neural Network (RNN), which is a natural choice for sequential data ([2]). However, some online resources (e.g. [3]) suggest that RNN may not be easily applicable to time series data. Nevertheless, we will start from a simple RNN architecture and gradually enhance it, in an attempt to improve the prediction quality. Tables 4.1 and 4.2 summarize all our attempts, which we detail next.

4.1. Selection of Hyper-Parameters

In order to select the best hyper-parameters for use in all NN, we perform exhaustive search over all the following options: LSTM state size in $\{10, 20, 30\}$; number of past lags in $\{3, 6, 12\}$; mini-batch size in $\{256, 512\}$; number of epochs in $\{50, 100, 150\}$. For each combination of these options, we measure the performance of a simple RNN, illustrated in figure 4a on speeds in 6 places: 3 middle of roads and 3 junctions, all chosen arbitrarily.

Figure 3 summarizes the results of the exhaustive search. We conclude that LSTM state size has the most consistent effect on performance, so that more cells yield better prediction quality. Also, the smaller mini-batch size yields better results, and better performance is gained by learning less past lags. Intuitively, the speed in the immediate future is more

Model	Place Group	$RMSE$	MAE	ρ	R^2
Baseline LR	Middle Junction	4.757 6.714	2.477 3.893	0.829 0.769	0.674 0.579
LSTM Separated	Middle Junction	4.521 (-0.236) 6.436 (-0.278)	2.383 (-0.094) 3.735 (-0.158)	0.846 (+0.017) 0.787 (+0.018)	0.614 (+0.060) 0.614 (+0.035)
LSTM Combined	Middle Junction	4.263 (-0.494) 6.212 (-0.502)	2.313 (-0.164) 3.634 (-0.259)	0.862 (+0.033) 0.802 (+0.033)	0.740 (+0.066) 0.642 (+0.063)
LSTM Mixture$_{15}$	Middle Junction	4.528 (-0.229) 6.425 (-0.289)	2.389 (-0.088) 3.72 (-0.173)	0.845 (+0.016) 0.788 (+0.019)	0.706 (+0.032) 0.615 (+0.036)
LSTM Mixture$_{30}$	Middle Junction	4.517 (-0.240) 6.422 (-0.292)	2.378 (-0.099) 3.713 (-0.180)	0.846 (+0.017) 0.789 (+0.020)	0.708 (+0.034) 0.616 (+0.037)
LSTM Grouped$_{1}$	Middle Junction	5.729 (+0.972) 6.537 (-0.177)	3.511 (+1.034) 3.956 (+0.063)	0.729 (-0.100) 0.778 (+0.009)	0.524 (-0.150) 0.601 (+0.022)
LSTM Grouped$_{2}$	Middle Junction	4.543 (-0.214) 6.376 (-0.338)	2.357 (-0.120) 3.782 (-0.111)	0.847 (+0.018) 0.792 (+0.023)	0.704 (+0.030) 0.622 (+0.043)
LSTM Conv1D$_{RND}$	Middle Junction	4.22 (-0.537) 6.163 (-0.551)	2.314 (-0.163) 3.6 (-0.293)	0.867 (+0.038) 0.807 (+0.038)	0.746 (+0.072) 0.648 (+0.069)
LSTM Conv1D$_{LAT}$	Middle Junction	4.209 (-0.548) 6.161 (-0.553)	2.294 (-0.183) 3.593 (-0.300)	0.866 (+0.037) 0.807 (+0.038)	0.747 (+0.073) 0.648 (+0.069)
LSTM Conv1D$_{LNG}$	Middle Junction	**4.184 (-0.573)** **6.141 (-0.573)**	2.293 (-0.184) 3.607 (-0.286)	0.867 (+0.038) 0.807 (+0.038)	0.75 (+0.076) 0.651 (+0.072)

Table 4.1: Speed prediction quality for all models. Improvement over baseline is in parenthesis. Best RMSE is in bold.

Model	Place Group	$RMSE$	MAE	ρ	R^2
Baseline LR	Middle Junction	0.546 0.735	0.325 0.519	0.947 0.921	0.896 0.846
LSTM Separated	Middle Junction	0.557 (+0.011) 0.753 (+0.018)	0.270 (-0.055) 0.463 (-0.056)	0.945 (-0.002) 0.917 (-0.004)	0.893 (-0.003) 0.840 (-0.006)
LSTM Combined	Middle Junction	0.539 (-0.007) 0.718 (-0.017)	0.327 (+0.002) 0.518 (-0.001)	0.949 (+0.002) 0.925 (+0.004)	0.900 (+0.004) 0.855 (+0.009)
LSTM Mixture$_{15}$	Middle Junction	0.539 (-0.007) 0.718 (-0.017)	0.327 (+0.002) 0.516 (-0.003)	0.948 (+0.001) 0.924 (+0.003)	0.899 (+0.003) 0.855 (+0.009)
LSTM Mixture$_{30}$	Middle Junction	0.539 (-0.007) 0.717 (-0.018)	0.326 (+0.001) 0.514 (-0.005)	0.949 (+0.002) 0.925 (+0.004)	0.9 (+0.004) 0.855 (+0.009)
LSTM Grouped$_{1}$	Middle Junction	0.66 (+0.114) 0.726 (-0.009)	0.461 (+0.136) 0.542 (+0.023)	0.922 (-0.025) 0.924 (+0.003)	0.848 (-0.048) 0.851 (+0.005)
LSTM Grouped$_{2}$	Middle Junction	0.537 (-0.009) 0.718 (-0.017)	0.364 (+0.039) 0.541 (+0.022)	0.95 (+0.003) 0.926 (+0.005)	0.9 (+0.004) 0.854 (+0.008)
LSTM Conv1D$_{RND}$	Middle Junction	0.517 (-0.029) 0.693 (-0.042)	0.334 (+0.009) 0.51 (-0.009)	0.953 (+0.006) 0.93 (+0.009)	0.907 (+0.011) 0.864 (+0.018)
LSTM Conv1D$_{LAT}$	Middle Junction	0.517 (-0.029) 0.693 (-0.042)	0.332 (+0.007) 0.509 (-0.010)	0.953 (+0.006) 0.93 (+0.009)	0.907 (+0.011) 0.864 (+0.018)
LSTM Conv1D$_{LNG}$	Middle Junction	**0.515 (-0.031)** **0.692 (-0.043)**	0.327 (+0.002) 0.504 (-0.015)	0.953 (+0.006) 0.93 (+0.009)	0.908 (+0.012) 0.865 (+0.019)

Table 4.2: Flow prediction quality for all models. Improvement over baseline is in parenthesis. Best RMSE is in bold.

affected by speeds in the immediate past, rather than speeds in the distant past.

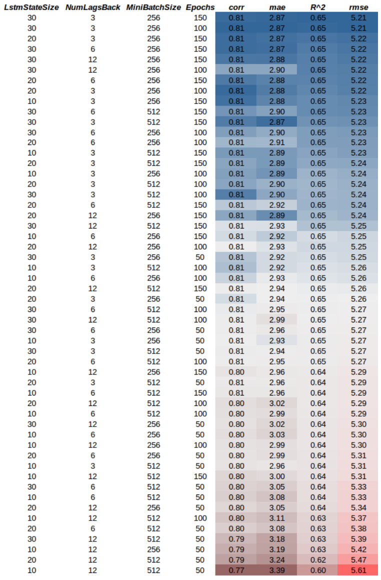
Thus from here on, all our NNs will be built with the following hyper-parameters, unless otherwise stated: LSTM state size = 30 ; past lags = 3 ; mini-batch size = 256 ; epochs = 100. We could otherwise choose 150 epochs for the same performance, but having more epochs increases the risk of over-fitting to train data.

4.2. Neural Networks with FC-LSTM

We begin by constructing RNN's where the recurrent units are "classic", Fully Connected Long Short-Term Memory (FC-LSTM) units. As mentioned before, tables 4.1 and 4.2 summarize the performance of these models, and compares them to the baseline LR. Furthermore, figures 5a and 5b compare the RMSE of all models visually. These are the tables and figures we refer to in the remainder of this section.

4.3. LSTM with Separated Input

Our first and simplest architecture is illustrated in figure 4a. We name this architecture **LSTM Separated**, because it is applied separately to speeds and to flows. We run this NN once for each selected place, and then aggregate the results as described in the formulae for prediction quality measure-



LstmStateSize	NumLagsBack	MiniBatchSize	Epochs	corr	mae	R^2	rmse
30	3	256	150	0.81	2.87	0.65	5.21
30	3	256	100	0.81	2.87	0.65	5.21
20	3	256	150	0.81	2.87	0.65	5.22
30	6	256	150	0.81	2.87	0.65	5.22
30	12	256	150	0.81	2.88	0.65	5.22
30	12	256	100	0.81	2.90	0.65	5.22
20	6	256	150	0.81	2.88	0.65	5.22
20	3	256	100	0.81	2.88	0.65	5.22
10	3	256	150	0.81	2.88	0.65	5.23
30	6	512	150	0.81	2.90	0.65	5.23
30	3	512	150	0.81	2.87	0.65	5.23
30	6	256	100	0.81	2.90	0.65	5.23
20	6	256	100	0.81	2.91	0.65	5.23
10	3	512	150	0.81	2.89	0.65	5.23
20	3	512	150	0.81	2.89	0.65	5.24
10	3	256	100	0.81	2.89	0.65	5.24
20	3	512	100	0.81	2.90	0.65	5.24
30	3	512	100	0.81	2.90	0.65	5.24
20	6	512	150	0.81	2.92	0.65	5.24
20	12	256	150	0.81	2.89	0.65	5.24
30	12	512	150	0.81	2.93	0.65	5.25
10	6	256	150	0.81	2.92	0.65	5.25
20	12	256	100	0.81	2.93	0.65	5.25
30	3	256	50	0.81	2.92	0.65	5.25
10	3	512	100	0.81	2.92	0.65	5.26
10	6	256	100	0.81	2.93	0.65	5.26
20	12	512	150	0.81	2.94	0.65	5.26
20	3	256	50	0.81	2.94	0.65	5.26
30	6	512	100	0.81	2.95	0.65	5.27
30	12	512	100	0.81	2.99	0.65	5.27
30	6	256	50	0.81	2.96	0.65	5.27
10	3	256	50	0.81	2.93	0.65	5.27
30	3	512	50	0.81	2.94	0.65	5.27
20	6	512	100	0.81	2.95	0.65	5.27
10	12	256	150	0.80	2.96	0.64	5.29
20	3	512	50	0.81	2.96	0.64	5.29
10	6	512	150	0.81	2.96	0.64	5.29
20	12	512	100	0.80	3.02	0.64	5.29
10	6	512	100	0.80	2.99	0.64	5.29
30	12	256	50	0.80	3.02	0.64	5.30
10	6	256	50	0.80	3.03	0.64	5.30
10	12	256	100	0.80	2.99	0.64	5.30
20	6	256	50	0.80	2.99	0.64	5.31
10	3	512	50	0.80	2.96	0.64	5.31
10	12	512	150	0.80	3.00	0.64	5.31
30	6	512	50	0.80	3.05	0.64	5.33
10	6	512	50	0.80	3.08	0.64	5.33
20	12	256	50	0.80	3.05	0.64	5.34
10	12	512	100	0.80	3.11	0.63	5.37
20	6	512	50	0.80	3.08	0.63	5.38
30	12	512	50	0.79	3.18	0.63	5.39
10	12	256	50	0.79	3.19	0.63	5.42
20	12	512	50	0.79	3.24	0.62	5.47
10	12	512	50	0.77	3.39	0.60	5.61

Fig. 3: Exhaustive search for best hyper-parameters. The color scale is blue for better values, red for worse values.

ments. The tables and figures show that LSTM Separated is somewhat better at predicting speeds, but worse at predicting flows, compared to the baseline LR.

4.4. LSTM with Combined Input

We thus advance to the next architecture, illustrated in figure 4b. We name this architecture **LSTM Combined**, because it is applied to speeds and flows together. As before, we run this network on each selected place separately before aggregating the results for each place group. The tables and figures show that LSTM Combined is better than all previous models at predicting both speeds and flows. Hence unlike LR, LSTM can take advantage of processing speeds or flows together rather than separately.

4.5. Mixture of Experts

Because combining speeds and flows improved the prediction quality, we next check whether it is better to first specialize one LSTM for each of the two types of data, and only then combine the learned representations. Figure 4c illustrates this architecture, which we name **LSTM Mixture**, because it employs a mixture of expert LSTM's. We run this network twice for each place: once with LSTM state size 15 – so that the total state size is 30 as before – and once with state size 30 – so that the network has twice as much memory. The tables show that even with twice the memory, the mixture of experts performs worse than LSTM Combined, which is therefore still the best architecture so far. Interestingly also, LSTM Mixture performs virtually the same regardless of the size of LSTM state.

4.6. Dropout and Regularization

As LSTM Combined is still the best performing architecture so far, we next try to improve it by introducing dropout and regularization. These techniques may reduce overfitting on the train set, and so yield better performance on the test set. We try out all combinations of the following parameters: dropout rate in $\{0, 0.25, 0.5, 0.75\}$, applied to the LSTM unit ; kernel regularizer in $\{\text{None}, l_2(E-2), l_2(E-4), l_2(E-6)\}$, applied to the dense layer. The baseline LSTM Combined is obtained for dropout rate zero and no regularization. We obtain that performance only worsens as more dropout and more regularization are applied, and so conclude not to use these techniques.

4.7. LSTM with Entire Place Group Input

For the last RNN architecture that we try with FC-LSTM, we feed an entire place group $G \in \{\text{middle-of-roads}, \text{junctions}\}$ at once, as in figure 4d. This way, perhaps the NN could take advantage of an overview of the road network. We name this architecture **LSTM Grouped**, and run it first with the same hyper-parameters as before. The results appear in the tables under LSTM Grouped$_1$, and show worse performance than LSTM Combined, which runs on one place at a time.

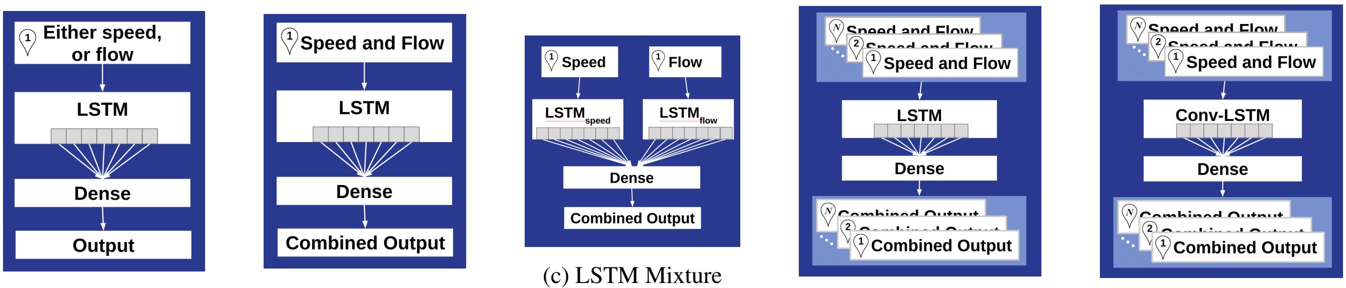
Thereafter, we increase the number of epochs to 150 and the state size to $30\,|G|$, and run again. The results for these larger hyper-parameters appear in the tables under LSTM Grouped$_2$. The performance is similar to LSTM Combined, with a slight improvement in junctions.

4.8. Convolutional LSTM

So far, our models have been oblivious to the structure of the underlying road network in the data. This structure should account for spatio-temporal correlations between different places, e.g. traffic congestion in a junction quickly spills over to the roads surrounding the junction. We now wish to build a model which takes advantage of such spatio-temporal correlations.

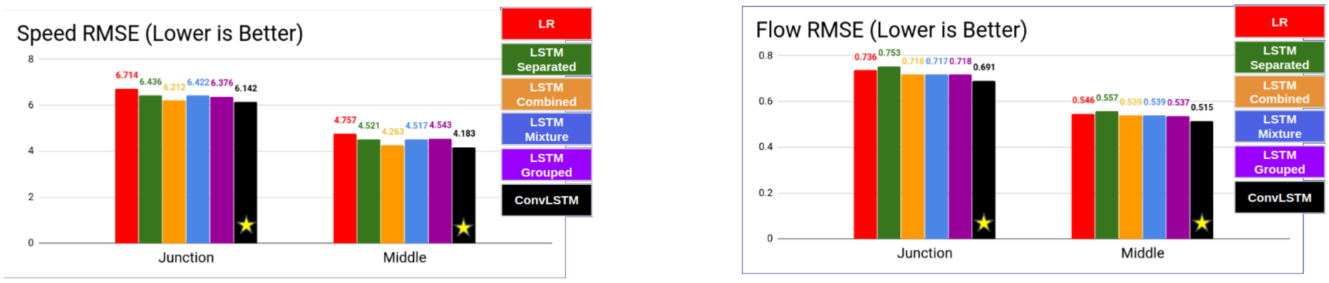
Our inspiration comes from [4], which shows that Convolutional LSTM ourperforms FC-LSTM in short-term prediction of weather over a 2-dimensional grid. In essence, a Convolutional LSTM (Conv-LSTM) unit internally applies convolution instead of matrix product, and so can learn patterns in local neighborhoods. In contrast, Fully-Connected LSTM (FC-LSTM) is equipped to learn global patterns.

Based on LSTM Grouped, we thus create a new architecture by replacing FC-LSTM with Conv-LSTM, as illustrated in figure 4e. Next, we note that placing a 2D grid on the map of places could be problematic in several aspects. First, the selected places would occupy only a few cells in the grid, which would thus be highly sparse. Second, neighboring cells



(a) LSTM Separated (b) LSTM Combined (c) LSTM Mixture (d) LSTM Grouped (e) LSTM Conv1D

Fig. 4: NN architectures in our experiments.



(a) Speed RMSE (b) Flow RMSE

Fig. 5: Visual comparison of RMSE for all models. A star marks the best value in each case.

in the grid would contain places on different traffic directions for the same road, which could hinder learning.

Thus instead of feeding our newest architecture proper 2D input, we first reduce the input to 1D. That is, we order all places linearly, and run the NN separately for each ordering. The ordering we try are: random, by latitude (i.e. horizontally), and by longitude (i.e. vertically) – the results are summarized in the tables under LSTM Conv1D$_{RND}$, LSTM Conv1D$_{LAT}$, and LSTM Conv1D$_{LNG}$, respectively. In all experiments, the convolutional kernel size is 5×1.

The results show that for our data too, for all linear orderings, Conv-LSTM outperforms FC-LSTM in predicting speeds and flows. This suggests that the improvement is brought about by the use of convolution instead of matrix product in the LSTM. Furthermore, ordering by latitude or by longitude yields better performance than ordering randomly, which suggests that our latest architecture indeed takes advantage of structural properties. The best improvement is gained for ordering by longitude ; figure 1 indeed shows that the selected places are more widely scattered vertically than horizontally.

5. CONCLUSIONS AND FUTURE WORK

Starting from a simple RNN based on FC-LSTM, we gradually enhanced the architecture to predict speeds and flows better than a baseline LR model. When we reached a point where performance improvements became marginal, we turned to use Conv-LSTM, which noticeably outperformed FC-LSTM. In comparison with the predictions of the baseline LR, our best RNN achieves 12% better speed RMSE in middle-of-roads, 8.5% better speed RMSE in junctions, 5.7% better flow RMSE in middle-of-roads, and 5.9% better flow RMSE in junctions.

We saw that similarly to LR, all our RNN models yielded more accurate predictions in middle-of-roads than in junction. We also saw that unlike LR, RNN could take advantage of combining speeds and flows at input level.

For future work, Graph Convolutional LSTM [5] appears promising for prediction of traffic in road networks. Our data is naturally structured as a graph, such that places are vertices, and edges exist between places which reside on the same road segment. Moreover, edges may be weighted by e.g. the length of the corresponding road segment. We can also try to improve prediction accuracy by stacking recurrent units – a common technique for adding levels of abstraction in RNN.

Finally, we wish to thank the following for their help and for the ideas they contributed to this work: research group members Filipe Rodrigues, Niklas C. Petersen, and Prof. Francisco C. Pereira (my PhD supervisor) ; and Prof. Ole Winther.

6. REFERENCES

[1] Jianlin Cheng, "A neural network approach to ordinal regression," *CoRR*, vol. abs/0704.1028, 2007.

[2] "The unreasonable effectiveness of recurrent neural networks," http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[3] "On the suitability of long short-term memory networks for time series forecasting," https://machinelearningmastery.com/suitability-long-short-term-memory-networks-time-series-forecasting/.

[4] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *CoRR*, vol. abs/1506.04214, 2015.

[5] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson, "Structured sequence modeling with graph convolutional recurrent networks," *CoRR*, vol. abs/1612.07659, 2016.