Traffic Forecasting Using PaddlePaddle[⋆]

Elena Akimova[1,2], Alexander Chernoskutov[2], Rostislav Kosivets[2], and Alexander Volkanin[2]

[1] Krasovskii Institute of Mathematics and Mechanics, Yekaterinburg, Russia
[2] Ural Federal University, Yekaterinburg, Russia
aen15@yandex.ru

Abstract. Traffic forecasting problem is considered. A new traffic prediction algorithm is designed. The algorithm based on an original deep neural network model is implemented with PaddlePaddle deep learning framework using a long-short-term memory layer to improve the prediction accuracy. All experiments have been performed on Ural Federal University cluster with Nvidia Tesla K20 GPUs.

Keywords: forecasting · deep learning · LSTM · PaddlePaddle · GPU

1 Introduction

Nowadays, real-time high-fidelity spatio-temporal data on transportation networks of overwhelming major cities have become available. People can obtain datasets from loop detectors, mobile phones, traffic cameras, and self-driving cars. This gold mine of the data can be utilized to learn the traffic behavior at different times and locations. Solving the traffic congestion prediction problem can potentially result in major saving of time and fuel. But it is still a challenge to construct such a prediction basing on the spatio-temporal relations.

In April 2017, the Asia Supercomputer Community and Inspur Group hosted annual international event: “ASC Student Supercomputer Challenge”, which main purpose was [1] “to challenge and inspire the next generation of HPC scientists and engineers to deliver innovative solutions”. Every year one of the tasks of the challenge is designed to address a specific applied problem, in which some collaborating organization or sponsor is interested. In 2017 such a task was the Traffic Prediction problem that should have been solved on the basis of PaddlePaddle framework.[1] It is implied that the approaches proposed by the participants will be further used or somehow integrated in commercial or other products by the parties involved in organization of the event. In order to do that, the organizers are motivated to provide participants with the real-life (or as close to it as possible) input data.

⋆ This work was partly supported by the Center of Excellence “Geoinformation technologies and geophisical data complex interpretation” of the Ural Federal University Program.

[1] The authors were awarded the Application Innovation Prize for the best solution of the problem.

In this paper, we describe the problem as it was stated and present our approach and results.

1.1 Existing Solutions Overview

Among the parametric methods, one of the most successful is ARIMA[2], which generated a whole class of methods (ARIMA with own subset, seasonal ARIMA, ARIMA with exogenous factors, ARIMA with Kohonen maps, vector ARIMA). All these methods are based on the assumption of stationary dispersion and mean of time series. The ARIMA method shows better accuracy than predecessors in predicting short-term traffic changes on highways.

Parametric models have a number of advantages. First, such models are easy to build and understand. Second, the solution is simpler and takes small amount of computational time. However, due to the nonlinearity and stochastic nature of the traffic, the parametric models are not able to take into account the uniqueness of data of this nature in the whole and have a large prediction error in comparison with nonparametric models.

Recently, ITS[3] have started to utilize full-connected architectures of deep training models for predicting short-term traffic flow. Researchers of this field have built a DNN[4] to capture the space-time features of the transport stream and developed a multi-tasking architecture for forecasting stationary and dynamic road traffic [2, 3]. Other researchers suggested using the SAE[5] model based on predictions of short-term traffic flow [4]. These approaches allowed one to accurately predict the future transport flow to some extent, however, they did not use the local topology of the road network and long-term data on the transport flow, which significantly reduced their predicting capabilities.

A graph-based neural network model was also developed and showed an improvement in predicting long-term dependencies while taking into account spatial data features [5]. However, such a model gave low accuracy in forecasting short term traffic.

In recent studies, a model was developed that combines the architectures of a convolutional neural network and LSTM[6], showing a slight improvement in accuracy with regard to spatial features [6]. The convolutional neural network layer processed spatial features, and several layers of LSTM processed short-term variations and the frequency of the transport stream.

[2] Autoregressive integrated moving average
[3] Intelligent transportation systems
[4] Deep neural network
[5] Stacked autoencoder
[6] Long short-term memory recurrent neural network

2 Problem Statement

2.1 Data Samples Representation

A city can be viewed as a set of connected roads, each road at any given time has a numerical congestion characteristic $X_{u_i,t} \in \{0, 1, 2, 3, 4\}$, i.e. a number which represents how "severe" the congestion on the current road is (see Table 1).

Table 1. Numerical congestion characteristic

Congestion characteristic	Description
0	No data
1	Healthy traffic, vehicles can drive freely within the speed limit
2	Minor congestion
3	Medium congestion
4	Drastic congestion, the traffic is almost completely stopped

It may look like the traffic characteristic has been simplified too much, but in this case we find it more suitable than some real physical quantity like average speed because of the next reasons:

– the traffic forecasting results in this particular case is targeted for human use (road users themselves). We find a short-scale congestion characteristic is much more intuitive for people because it is easy to understand and, most importantly, easy to compare current road condition to a "normal" traffic or to what it was like before;
– the congestion characteristic incorporates roads' parameters such as speed limits and road quality. For example, average speed of 40km/h can be considered good in busy downtown or on a field road, but it is absolutely inadequate for a highway. So in the first case the congestion value can be defined as 1 and in the second as 3, even though the average speed is the same. So, users do not need to take into consideration any additional parameters; they can understand how "good" or "bad" the traffic on the particular road is right away;
– the collected data samples are usually not evenly distributed throughout the time, which can introduce instability to the system. For example, if speed data is acquired through the drivers' cellphones, amount of collected data is proportional to the amount of drivers that decided to drive through the particular road. Coarsening the data, we are getting rid of its fluctuations and making it easy to interpolate in the case of insufficient data.

Additionally to the collected time-dependent traffic data, we also consider road connectivity information, which is represented by the oriented graph $G(V, A)$, where V is a road set, A is a set of ordered pairs of vertices $u_i, u_j \in V$ denoting an intersections of roads.

2.2 Metric

In order to be able to compare different prediction results and reduce the task to a minimization problem, a representative metric is to be chosen. In this case, the results were evaluated by RMSE[7]. The RMSE is very common choice for many minimization problems. While its main advantages are continuousness and differentiability, we also find it very intuitive at representing of how "good" the result is. Simply analyzing the construct of the problem, we can determine a few things about RMSE: in the worst case scenario, when the prediction and target as far away from each other as possible, RMSE = 3 (since $X_{actual,i} \in \{1, 2, 3, 4\}$ and $X_{model,i} \in \{1, 2, 3, 4\}$); in the best case scenario, RMSE = 0.
Now the forecasting problem can be reduced to the minimization problem of finding m traffic states of u_i node in V using n previous states

$$argmin \left(\sqrt{\frac{\sum_{t=1}^{n} (X_{u_i,t} - X^*_{u_i,t})^2}{m}} \right), \tag{1}$$

where $X_{u_i,t}$ is observed value of u_i node at the instant t, while $X^*_{u_i,t}$ is predicted value of u_i node at the instant t.

3 Initial Data Analysis

In the course of our work, we had only one data source for all the experiments; but its spatial resolution was sufficient to conduct a number of independent tests (by splitting it to several non-overlapping training and testing samples). The size of the whole provided data relates to the size of the prediction as 400 to 1.

3.1 Data Format

The data is aggregated in 5-min intervals each from 00:00 a.m. on March 1st to 8:00 a.m. on May 25th, 2016. Every measurement is denoted by four states, as was described earlier. A traffic intensity map is shown in Fig. 1. Our task was to predict the traffic in the following 2 hours from 8:05 a.m. to 10:00 on May 25th.

3.2 Data Analysis

Initial data contain several anomaly regions. There are: periodic absences of data (white regions) from 5:00 a.m. Saturday to 5:00 a.m. on Monday (Fig. 1), stochastic anomalies and nonuniformness of values (Fig. 2). Small anomalies were approximated by neighbor values, but big regions were just removed from the training dataset.

[7] Root-mean-square error

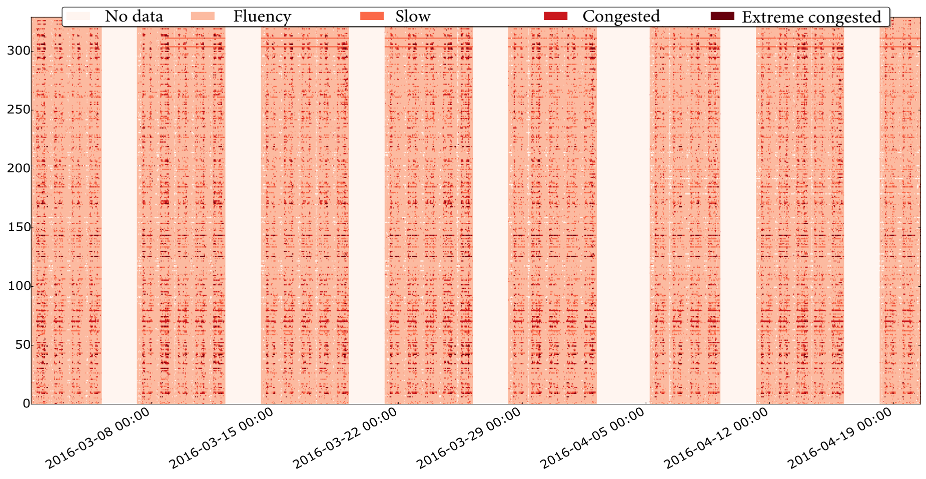


Fig. 1. Initial data

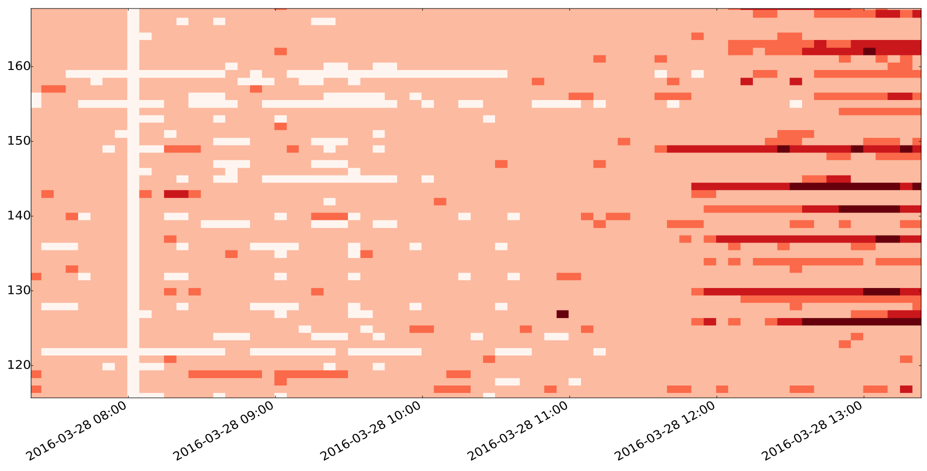


Fig. 2. Anomaly example

Figure 1 shows the initial data for a graph of roads consisting of 349 nodes. White gaps mean the absence of observations at the current time. The colored spots are the values of the traffic congestion (from 1 to 4). The darker the area the worse is the congestion. The figure shows that there are entire blocks of missing data, and these blocks represent periodic. Most of these blocks represent missed traffic information for two days - from 05:00 Saturday to 05:00 Monday. One of these blocks is three days long, from 05:00 on 02/04/2016 to 05:00 on 05/04/2016. In addition to the missing data, there could be a lack of information about the congestion on the road at any time, at arbitrary nodes of the network, and with different time length. Thus, analysis of the initial data shows that they are highly sporadic and inconsistent.

In order to be able to preprocess the initial data and choose the algorithm for solving the problem, we analyzed the average daily traffic on different days of the week. Figure 3 shows the average traffic load graphs for the entire system of roads.

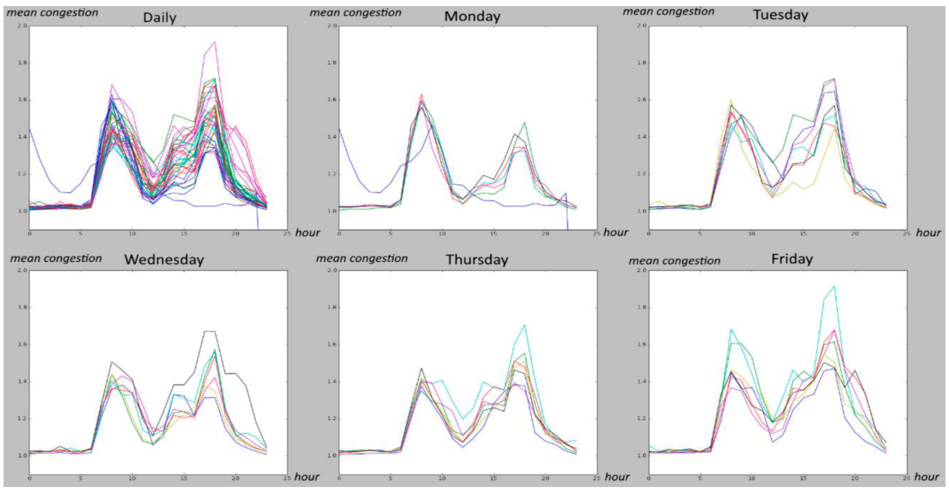


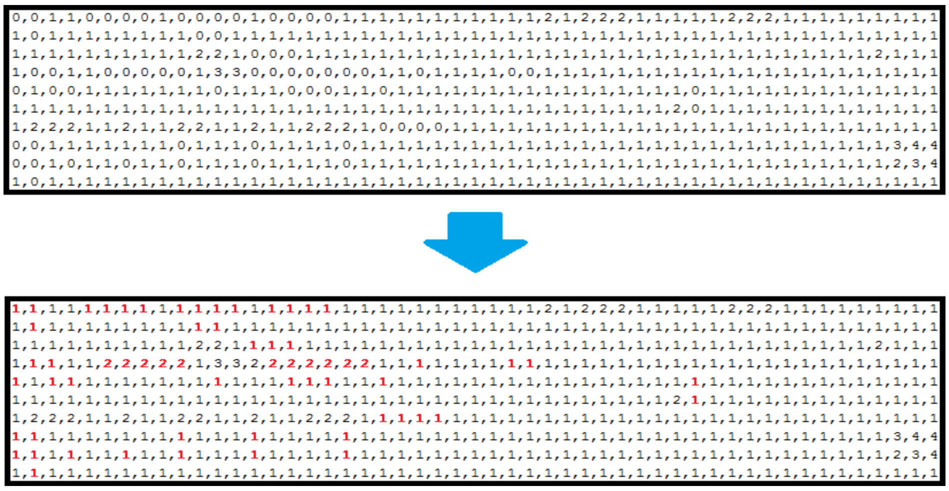
Fig. 3. Traffic intensity by the days of the week on separate roads

3.3 Data Preprocessing

The initial data contain both useful data for training the neural network (traffic congestion values) and the filler values (zeros) denoting the instants when there is no data available. If you feed such data to the neural network input during learning without preprocessing it, a good result is not to be expected, since blocks of missing data will disrupt the learning process.

In order to improve the quality of traffic forecasting, all the data gaps should be eliminated. We can split this task in two stages: the elimination of large periodic groups of gaps and the elimination of relatively-isolated gaps in random places. In the case of periodic blocks, we simply cut these blocks out of the original data and concatenate the remaining parts in such a way that there is no gaps in timestamps of the day. The random data gaps are somewhat more difficult to handle because they can arise at arbitrary places and have an arbitrary length in time. The processing consists in interpolating such intervals with averaged values from several closest surrounding points of known data. At the top of Fig. 4, a part of initial data are shown, at the bottom the data are already preprocessed; interpolated values are marked with red.

After the preprocessing has been applied to the initial data, it has reduced in size from, approximately, 85 days to 61 (due to 24 days of missing data).



0,0,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,2,1,2,2,2,1,1,1,1,1,2,2,2,1,1,1,1,1,1,1,1
1,0,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
1,1,1,1,1,1,1,1,1,1,2,2,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,1,1,1
1,0,0,1,1,0,0,0,0,0,1,3,3,0,0,0,0,0,0,0,1,1,0,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
0,1,0,0,1,1,1,1,1,1,1,0,1,1,1,0,0,0,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,0,1,1,1,1,1,1,1,1,1,1,1,1
1,2,2,2,1,1,2,1,1,2,2,1,1,2,1,1,2,2,2,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
0,0,1,1,1,1,1,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,3,4,4
0,0,1,0,1,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,3,4
1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

1,**1**,1,1,**1**,**1**,**1**,**1**,1,**1**,**1**,**1**,**1**,1,**1**,**1**,**1**,**1**,1,1,1,1,1,1,1,1,1,1,1,2,1,2,2,2,1,1,1,1,1,2,2,2,1,1,1,1,1,1,1,1
1,**1**,1,1,1,1,1,1,1,1,**1**,**1**,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
1,1,1,1,1,1,1,1,1,1,2,2,1,**1**,**1**,**1**,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,1,1,1
1,**1**,**1**,1,1,**2**,**2**,**2**,**2**,**2**,1,3,3,2,**2**,**2**,**2**,**2**,**2**,**2**,1,1,**1**,1,1,1,1,**1**,**1**,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
1,1,**1**,**1**,1,1,1,1,1,1,1,**1**,1,1,1,**1**,**1**,**1**,1,1,**1**,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,**1**,1,1,1,1,1,1,1,1,1,1,1,1
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,**1**,1,1,1,1,1,1,1,1,1,1,1,1
1,2,2,2,1,1,2,1,1,2,2,1,1,2,1,1,2,2,2,1,**1**,**1**,**1**,**1**,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
1,**1**,1,1,1,1,1,1,1,**1**,1,1,1,**1**,1,1,1,1,**1**,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,3,4,4
1,**1**,1,**1**,1,1,**1**,1,1,**1**,1,1,1,**1**,1,1,1,1,**1**,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,3,4
1,**1**,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

Fig. 4. Interpolation of the missing traffic data

4 Implementation

4.1 Design of the Algorithm

Proposed algorithm is based on deep recurrent neural network with long short-term memory layer. As shown in Fig. 5, the model consists of 5 layers: input data (n neurons), full-connected layer (k neurons), LSTM layer (k neurons), full-connected layer (4 neurons), and output data layer (4 neurons). The main idea of the algorithm is that the intensity values of the neighbor nodes affect the current node and, therefore, one should consider those values to predict traffic intensity of the current node.

At each time instant, the neural network input is fed with the traffic intensity values from the neighboring roads or from the entire graph (if computing capabilities are sufficient) at the previous point of time. Training (and prediction) is conducted for the current road at m time points after the time point, from which the data are fed to the input. All m points of time are predicted in parallel. This can be seen in Fig. 5. The final layer of the neural network outputs a set of m values corresponding to each predicted instant. For implementation of the neural network the PaddlePaddle [7] framework has been used.

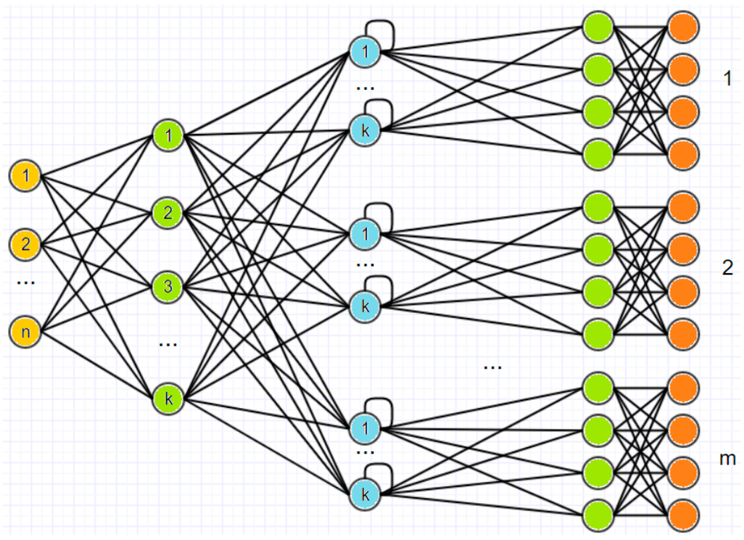


Fig. 5. Neural network architecture

4.2 Model Training

Table 2 shows different model configurations. There are 3 models denoted $LSTM_\lambda$, where λ is mean radius of neighboring nodes. Model $LSTM_v$ uses all the graph nodes for training and predicting the following values; n is number of input values (for each to be predicted node), k is number of hidden neurons (for each to be predicted node), *epoch* is number of epochs, *learning_rate* is learning rate. During training, we used sliding window method to predict the next m values.

Even though our approach is designed and tested using PaddlePaddle, the reader is advised to keep in mind that there is just one of the many implementations of the ANN[8] algorithms, and all the described methods can be adapted for any other ANN implementation without having any effect on the output result whatsoever.

[8] Artificial neural network

Table 2. Model configurations

$model_\lambda$	n	k	$epoch$	$learning\_rate$
$LSTM_0$	1	2	10	10^{-3}
$LSTM_1$	5	6	5	10^{-3}
$LSTM_1$	5	6	10	10^{-3}
$LSTM_2$	12	15	5	10^{-3}
$LSTM_2$	12	15	10	10^{-3}
$LSTM_v$	349	400	10	10^{-4}

5 Experiments

Figure 6 presents the results of the tests. Increasing number of neighboring nodes and epoch after some point does not show any significant reduce of RMSE, the best model is (as expected) $LSTM_v$ that uses all graph nodes.

One of the greatest advantages of the PaddlePaddle framework is that it can utilize the power of modern GPU right "from the box". All experiments have been performed on Ural Federal University cluster with Nvidia Tesla K20 GPUs. Comparison of the training times using CPU and GPU shown in Table 3. It is apparent that small neural networks (with only dozens of neurons) cannot fully take advantage of GPU computational power. But as the size of ANN grows, the GPU learning time becomes up to 10 times smaller that of a CPU. Since there's an isolated instance of ANN for each node of the road graph we can easily spread the learning process over a cluster of computational nodes without the need of internode communications, thus getting a linear computational acceleration.

Table 3. Training times: CPU vs. GPU

$model_\lambda$	CPU time, min.	GPU time, min.
$LSTM_2$	78	460
$LSTM_v$	4688	832

6 Conclusion

A new architecture of neural network and new preprocessing algorithm for short-term traffic forecasting were proposed. Experiments with different types of neural network layers showed that the simple full-connected layers with one LSTM layer yield the best result for the task. The constructed implementation provides the

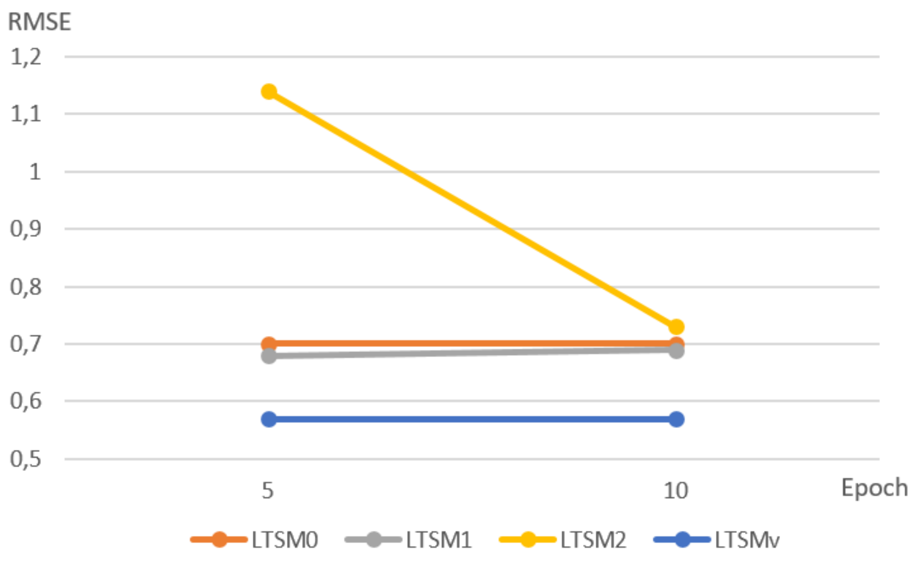


Fig. 6. Training results

task to be easily scaled in number of road graph nodes by limiting the radius of neighboring nodes. The PaddlePaddle framework allowed one to utilize in implementation the power of modern high-performance GPU solutions without modifying the source code.

References

1. ASC Student Supercomputer Challenge. http://www.asc-events.org/
2. Huang W., Song G., Hong H., and Xie K.: Deep architecture for traffic flow prediction: deep belief networks with multitask learning, IEEE Transactions on Intelligent Transportation Systems. Vol. 5, no. 5, p. 2191–2201 (2014).
3. Hinton G. E., Osindero S., and Teh Y.-W.: A fast learning algorithm for deep belief nets, Neural computation. Vol. 7, no. 18, p.1527–1554 (2006).
4. Lv Y., Duan Y., Kang W., Li Z., and Wang F.-Y.: Traffic flow prediction with big data: a deep learning approach, IEEE Transactions on Intelligent Transportation Systems. Vol. 2, no. 16, p. 865–873 (2015).
5. Shahsavari B.: Short-term traffic forecasting: Modeling and learning spatio-temporal relations in transportation networks using graph neural networks: mscs. University of California, Berkeley (2015).
6. Wu Y., Tan H.: Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. https://arxiv.org/pdf/1612.01022
7. PaddlePaddle: parallel distributed deep learning platform. http://doc.paddlepaddle.org/release_doc/0.9.0/doc/