Traffic State Estimation Using Speed Profiles and Convolutional Neural Networks

* Faculty of Transport and Traffic Sciences/Department for Intelligent Transport Systems, University of Zagreb, Croatia
ltisljaric@fpz.hr

Abstract – Determining the traffic state is one of the most attractive problems for experts in the field of Intelligent Transport Systems (ITS). In this paper, a deep learning model for determining the traffic state is presented. Model is based on Convolutional Neural Networks (CNN) and uses weekly speed profiles as input data. The proposed model consists of input and output layer with an addition to four convolutional layers, two pooling layers and two fully connected layers that are extracting important features and classifying intersections as congested or not congested. We analyze data and predict traffic state for the most relevant road segments in the City of Zagreb which is the capital and largest city in Croatia. Speed profiles from included road segments are represented as one traffic image and used to train CNN. In that way traffic state for all sequentially connected road segments is estimated. The proposed method achieves a classification accuracy of more than 90% on three analyzed types of road topologies. The results show that CNN trained with traffic images can be used as a tool for traffic state estimation.

Keywords – traffic state estimation, convolutional neural networks, speed profiles, intelligent transport systems

I. INTRODUCTION

Traffic state estimation is commonly referred as a process of the determining traffic state parameters on one or many road segments using collected traffic data. Most often used traffic state parameters are traffic flow, density and speed, but other parameters can be used such as acceleration, spacing and other equivalent variables [1]. High quality estimation of traffic parameters has an important role in many ITS applications. Applications like ramp metering, speed limit control, toll collection and signal plan control at intersections require accurate traffic state estimation in order to reduce congestion and obtain fluent traffic flow.

In this paper we are using GPS speed data aggregated into speed profiles to estimate traffic state. Speed profile is a discrete function which represents the speed change over observed time. Useful information can be extracted from a speed profiles such as: the congested areas, occurrence and duration of rush hours, congestion propagation patterns etc. For that reason, historical speed profiles are chosen for the estimation of a traffic state on an urban road network. CNN was trained with speed profiles represented as images in order to predict the traffic state as congested or not congested. Therefore, traffic state estimation is modeled as a binary classification problem.

The remainder of this paper is organized as follows. In section 2 related work on recent developments related to the estimation and prediction of traffic state parameters based on the deep learning methods is presented. Section 3 describes the method for collecting and processing GPS data to extract speed profiles and represent them as traffic images. In section 4 method for traffic state estimation based on CNN is presented. In section 5 main research results are presented and analyzed. The conclusion of the paper is given in section 6.

II. RELATED WORK

According to [2] two approaches in traffic state estimation and prediction can be identified: non-parametric models and parametric models. Non-parametric models like k-nearest neighbors [3] and support vector regression [4] have been applied to predict traffic speed and traffic flow. The parametric models are represented with the autoregressive integrated moving average model (ARIMA) [5] such as seasonal ARIMA, Kohonen ARIMA and ARIMA with Kalman filter. The Bayesian network [6] and Markov chain models [7] are also applied for traffic state estimation and prediction tasks.

Speed profiles are common input data for the estimation and prediction of traffic state parameters. Authors in [8] are using speed profiles to estimate time dependent traffic states and use them in routing application. Highway capacity manual [9] classifies the level of service on road segments into six groups based on speed data. Classes ranges are estimated by comparing the measured speed value to the Free Flow Speed (FFS). In [10] authors are using speed profiles to predict traffic speed for highway operations.

Recently, a great number of deep learning models based on neural networks were developed for traffic state parameters estimation and prediction. Authors in [11] are using deep learning models to predict traffic flow. Deep learning traffic state prediction model that takes into account spatio-temporal relation is proposed in [12]. Deep learning model is applied using autoencoders as building blocks that represent traffic flow features for prediction. Authors in [2] propose a CNN based method to predict large-scale, network wide traffic speed. As input, the authors used speed profiles computed into two minutes intervals. Speed profiles were then represented as traffic images. Traffic images are constructed as a spatio temporal matrices. Then, traffic images are used to train CNN to
extract and predict traffic parameters. In [13] authors are using CNN and recurrent neural networks to predict and estimate traffic state. The authors classify traffic state as fluent, slow, congested and extremely congested.

III. DATA PROCESSING

For the computation of speed profiles, we used GPS data acquired from the vehicles equipped with the GPS tracking devices. Each record contains a time-stamp, geographical longitude and latitude, speed, heading etc. Due to the storage limitation most of the data are sampled in the following way: sampling rate of 100 m/s for vehicles in driving mode and every 5 min for turned off vehicles. Raw data are map matched to the road segments in a digital map based on the measured latitude, longitude and heading. Data which could not be matched to appropriate road-segment due to the GPS error caused by tunnels, high building concentration or other causes were filtered out [14], [15]. GPS data for the road network of Croatia were recorded during a five-year period between August 2009 and October 2014 by approximately 4200 tracked vehicles. The tracked vehicle fleet is versatile and consists mostly of delivery vehicles (vans, caddies, small trucks) and taxi cars. The historical tracked GPS data which consists of 6.55 billion GPS records were provided by the company Mireo Inc., as a part of SORDITO project [16].

A. Speed profiles

To construct a speed profile, speed data from GPS records are aggregated into 5 min intervals. In that way, a day is divided into 288 intervals with corresponding speed values. To cope with different speed limits and driving behavior on road segments, relative speed profiles are used. Speed is computed relative to the FFS on the road segment. FFS can be defined as the speed of a single vehicle on an empty road segment or as the speed limit on the observed road segment [3]. Every speed is represented with in interval [0, 100] relative to FFS on the road segment. For FFS value, maximal value of recorded speed was used. Fig. 1 shows a difference between the relative and absolute speed profile for a road segment with two typical rush hour periods during a day. In such way similar driving patterns on different road segments can be observed (grouped).

![Figure 1. Absolute and relative speed profile for one road segment in city of Zagreb.](image)

B. Traffic images

To predict traffic speed, time and space dimensions should be jointly considered. We represent speed profiles for road segments in the road network as 2D matrices (Fig. 2). Two dimensions of the matrix are m and n where n represents the number of road segments under consideration (space dimension) and m represents the number of consecutive discrete time intervals (time dimension).

The time dimension is usually represented as intervals that extend throughout the day and are divided into 2 or 5 min intervals. In this paper, 5 min interval was used. Space-time matrix $M$ is presented in (1), where $v_{ij}$ is the average traffic speed recorded on road segment $i = 1, ..., n$ in the time interval $j = 1, ..., m$.

$$M = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & \cdots & v_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nm} \end{bmatrix}$$ (1)

For every week of the year one traffic image was generated. To lower the deviation, summer months, July and August, are excluded from the model, because they significantly influence the results due to the different traffic flows caused by vacations [17]. Considered seasonal differences resulted in 47 weeks of data. Data are further divided into two groups: working days and weekend days. Working days speed profiles from Monday to Friday are different than the weekend speed profiles for Saturday and Sunday mostly due to the daily commuters. To train a binary classification model, we used speed profiles for working and weekend days as a train and test data for classifying road segments as congested and not congested, respectfully.

IV. MODEL

The CNN is a type of a feed forward Artificial Neural Network (ANN). CNN architecture makes the assumption that the input is in form of an image. In comparison to classical neural networks architecture, CNN reduces the number of parameters using convolutional layers which allows a network to be deeper. Common CNN consists of a combination of the three types of layers: convolutional, pooling and fully connected layers.
A. CNN layers

Convolution layers (CL) consists of one or more filters (kernels) which weights are adapted through the learning process. Every filter has a small width and height, but it extends through the full depth (all channels) of the input image. CL takes several feature maps as input and produces \( n_f \) feature maps as output, where \( n_f \) is a number of filters in the CL. Feature maps are then passed as input data to the next layer of the network.

Train data for CNN can be denoted as \( \{x_i, y_i\}_{i=1}^{nr} \), where \( x_i \) is input data described as a matrix with three dimensions \( x_i \in \mathbb{R}^{h \times w \times d} \). \( h \times w \times d \) represents image height, width and depth, output data \( y_i \in \mathbb{R} \) and \( nr \) number of training examples. Then, the convolution process can be described by (2), where \( f_m \) are a feature maps obtained after convolution operation on image \( x_i \) with the set of weights \( w_j \) and bias \( b_j \).

\[
    f_m = \text{Conv}(x_i) = \sum_{j=1}^{n_f} w_j \times x_i + b_j 
\]

Pooling layers are commonly applied after one or more convolution layers. Their main purpose is to downgrade an input image by reducing its spatial dimensions. The most common pooling filter size is 2x2, which reduces dimensions by half with one of pooling methods like pooling by maximal value or average pooling.

Fully connected layers are very similar to hidden layers in the multi-layer perceptron network. The main usage of fully connected layers is to learn weights to classify given image. In contrast to the convolution layer, a fully connected layer has only one dimension. To use it, the output of the last convolution or pooling layer has to be flattened in form of a vector.

B. CNN Hyperparameters

Regarding to architecture of the CNN, three important factors must be considered: (I) determining the hyperparameters of the CNN layers such as kernel size, pooling size, pooling method, activation functions, (II) depth of the network (number of filters used) and (III) number of layers in the network. For determining all mentioned hyperparameters there are no strict rules to follow. Every problem is different, and most of the hyperparameters have to be determined experimentally [2].

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel size</td>
<td>3x3</td>
</tr>
<tr>
<td>Pooling size</td>
<td>2x2</td>
</tr>
<tr>
<td>Pooling method</td>
<td>Maximal pooling</td>
</tr>
<tr>
<td>Activation function (convolutional layers)</td>
<td>Rectified linear unit</td>
</tr>
<tr>
<td>Activation function (classification layer)</td>
<td>Sigmoid</td>
</tr>
</tbody>
</table>

To determine CNN hyperparameters, well known CNN architectures can be referenced. First successful architecture was LeNet, based on the work of Yann LeCun in 1990’s where 32x32 pixelated images are used as input. The LeNet has two convolution layers, two pooling layers followed by two fully connected layers [18]. Later on, deeper and larger CNN architectures were developed as a part of popular image classification competition called ImageNet Large Scale Visual Recognition Competition (ILSVRC). In 2010. AlexNet was introduced with the architecture based on five convolutional layers with filter sizes 1x11 and 3x3, three fully connected layers and three maximal pooling methods. For input, the network is using images with the size of 224x224 pixels [19]. VGGNet was introduced in 2014. This network is using 3x3 filters in 13 or 16 convolution layers and three fully connected layers [20]. Similar architecture was used in papers [2] for traffic state estimation and predictions. Table 1 shows chosen hyperparameters based on the AlexNet and VGGNet.

The number of the convolutional filters and the number of layers of the CNN were estimated experimentally. Structure of the CNN was tested on three different depth levels and different number of layers, presented in Table 2. Depth 1 represents only two fully connected layers that perform classification on flattened data. Depth 2 and 3 are using convolutional and pooling layers for feature extraction and then perform classification through fully connected layers. Every two convolutional layers were followed by a pooling layer. Layers were labeled with a notation \( h \times w \times d \), where \( d \) is a number of filters in a convolution layer (depth), \( w \) represents filter width and \( h \) filter height. In the experiment, for every traffic state (work week and weekend) 47 images was generated, which resulted with 94 images in total. Each image size is 100x288 pixels. Where the values \([0, 100]\) represents the relative values of the speed in a given interval, and 288 is the number of time intervals used to represent the speed profile. Data were further divided into train and test dataset. To keep the track of the train process and to detect possible overfitting, the 20% of the train dataset is used to validate the training process.

Fig. 3 shows the result of testing the depth of CNN. As expected, depth 1 gives the worst loss results as a model is
not able to learn features from the presented images. Adding the depth to CNN increases the model capacity and significantly reduces loss on train and test dataset. As a result, depth 3 with two convolution layers 5x5x128, 2x2 max pooling layer, two convolutional layers 5x5x64 and 2x2 max pooling layer gives the best results and is used for further experiments in this paper.

Table 3 represents the hyperparameters of depth 3 CNN used in this paper. The first layer indicates input image size which consists of three dimensions with size 100x288x3. The first dimension is image height, second image width and third dimension is number of image channels. Convolutional layers transform the input image into 128 and 64 channels, without changing the width and height. Pooling layers are downsampling the image to 50x144 and 25x72 pixels. After the last pooling layer with a dimension 25x72x64 image is flattened into a vector with dimension of 115200. Flattening is followed by two fully connected layers with a size of 128. Last layer represents the model output with a size of 1. The last layer in binary classification task gives the result \( r \) in \( \{0, 1\} \) which shows the class affiliation.

To prevent the model from an overfitting, a dropout layer is applied. Dropout is a method that randomly drops units (neurons) and their weights from a neural network layer during training. This prevents units from co-adapting to feature detectors [21]. It is common to use a dropout layer after a fully connected layer in the CNN structure [22].

---

**Table 2. Tested structures of the CNN classification model**

<table>
<thead>
<tr>
<th>Depth</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth 1</td>
<td>Two fully connected layers</td>
</tr>
<tr>
<td>Depth 2</td>
<td>2(5x5x64) convolutional layers 2x2 maximal pooling</td>
</tr>
<tr>
<td>Depth 3</td>
<td>2(5x5x128) convolutional layers 2x2 maximal pooling 2(5x5x64) convolutional layers 2x2 maximal pooling</td>
</tr>
</tbody>
</table>

---

**Table 3. Hyperparameters of the CNN model**

<table>
<thead>
<tr>
<th>Layer number</th>
<th>Type</th>
<th>Parameters</th>
<th>Dimension</th>
<th>Parameter scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>-</td>
<td>-</td>
<td>100x288x3</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>Conv.</td>
<td>5x5x128</td>
<td>100x288x128</td>
<td>9728</td>
</tr>
<tr>
<td>2</td>
<td>Conv.</td>
<td>5x5x128</td>
<td>100x288x128</td>
<td>409728</td>
</tr>
<tr>
<td>3</td>
<td>Pool</td>
<td>2x2</td>
<td>50x144x128</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Conv.</td>
<td>5x5x64</td>
<td>50x144x64</td>
<td>204864</td>
</tr>
<tr>
<td>5</td>
<td>Conv.</td>
<td>5x5x64</td>
<td>50x144x64</td>
<td>102464</td>
</tr>
<tr>
<td>6</td>
<td>Pool</td>
<td>2x2</td>
<td>25x72x64</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Flat</td>
<td>-</td>
<td>115200</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Full</td>
<td>-</td>
<td>128</td>
<td>14745728</td>
</tr>
<tr>
<td>9</td>
<td>Full</td>
<td>-</td>
<td>128</td>
<td>16512</td>
</tr>
<tr>
<td>Output</td>
<td>Output</td>
<td>-</td>
<td>1</td>
<td>129</td>
</tr>
</tbody>
</table>

---

**Figure 4. Example images of train and test dataset**

**Figure 4. Example images of train and test dataset**

---

V. RESULTS

In this paper, traffic state estimation is considered as a binary classification problem. We classify traffic state as congested or not congested. Congested traffic is represented with weekly computed speed profiles when considering only working day and not congested traffic is represented with speed profiles for weekend days. After filtering, as described in section 3, traffic images were extracted for training and testing the CNN network. Fig 4 shows the example of train and test data for classifying congested and not congested traffic state on the road segments. Image a) represents congested road segments, it can be seen that traffic congestion appears in morning and afternoon rush hours between 7 am and 4 pm. Not congested road segments on image b) do not follow the same pattern.

In order to test the performance of the proposed CNN architecture, Long Short-Term Memory (LSTM) neural network with 128 nodes is chosen for comparison. LSTM is a successor of the recurrent neural network architecture and popular solution because it avoids vanishing gradient problem [23]. LSTM architecture is not developed to use images as an input data. In order to train and test the LSTM, data is represented as 2D matrix with dimensions \( m \times n \) described with (1).

CNN model is compared with LSTM model on three different road topologies (Fig. 5): (a) intersection, (b) several consecutive road segments without turnings and (c) several consecutive road segments with turnings. All analyzed road segments have a one-way direction.

Data processing, training and testing of the proposed solution is developed using Python programming language with the addition of deep learning framework TensorFlow [24]. Performance of both networks is shown with Table 4. The results show that CNN architecture achieves better result in every tested topology with an accuracy above 90%.

**Table 4. Performance of the CNN and LSTM**

<table>
<thead>
<tr>
<th>Topology</th>
<th>Loss</th>
<th>Acc[%]</th>
<th>Loss</th>
<th>Acc[%]</th>
<th>Loss</th>
<th>Acc[%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology 1</td>
<td>0.65</td>
<td>92</td>
<td>0.60</td>
<td>93</td>
<td>0.67</td>
<td>90</td>
</tr>
<tr>
<td>Topology 2</td>
<td>0.71</td>
<td>89</td>
<td>0.83</td>
<td>82</td>
<td>0.69</td>
<td>89</td>
</tr>
<tr>
<td>Topology 3</td>
<td>0.65</td>
<td>92</td>
<td>0.60</td>
<td>93</td>
<td>0.67</td>
<td>90</td>
</tr>
</tbody>
</table>


ACKNOWLEDGMENT
This research has been supported by the European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATA CROSS).

REFERENCES

VI. CONCLUSION
In this paper deep learning model for traffic state estimation is presented. The proposed model is based on the CNN and traffic state estimation is represented as a binary classification problem as congested or not congested. Model is trained with weekly speed profiles presented as traffic images. The architecture of the model consists of an input and output layer with an addition of four convolutional layers followed by two pooling layers. For the pooling method, pooling with maximal values is chosen. Pooling layers are followed by two fully connected layers that are performing a classification task.

The method is tested on three types of road network topology: the intersection, sequentially connected road segments without turnings and sequentially connected road segments with turnings. Results are compared with the LSTM based neural network. CNN outperforms the LSTM in every test scenario with an accuracy above 90%.

Results are tested on road segments that have a one-way direction. Possible further research would be developing a model that considers two-way direction roads. A similar problem is researched in [25]. The model can be further extended by combining two or more methods like convolutional LSTM networks or diffusion convolutional RNN networks.

Figure 5. Different test road topologies


