Deep convolutional neural network architecture for urban traffic flow estimation

Sabbani imad†, Perez-uribe Andres††, Bouattane Omar††, El Moudni Abdellah†††

†School of engineering and management Vaud, Yverdon-les-bains, Switzerland
††Faculty of sciences and techniques, Hassan II University, Mohammedia, Morocco
†††Faculty of sciences and techniques, Franche-Comté University, Besançon, France

Summary
Road traffic density estimation can be very helpful for the successful deployment of intelligent Transportation systems. In this paper, we introduce a deep convolutional neural network (DCNN) based method that learns traffic density from pre-labeled images in order to estimate the traffic flow density in highways. Our method classifies the traffic flow density into three different states: light, medium and heavy. A standard database of real videos from Seattle roads was used to develop our proposed approach. The cross-validation and the class activation mapping techniques were employed in this work, in order to evaluate the performance of our method. The results show that our model outperformed all the existing conventional methods by reaching the highest accuracy of 99,62%.

Key words:
Pattern recognition, Video processing, Road traffic density, Deep convolutional neural network

1. Introduction

Video processing methods are nowadays considered one of the most active research areas for transportation systems in order to manage the traffic flow efficiently. Recently, the increased availability of visual data and the advances in storage devices over the last decade have made the research community focus on the computational capacities for image understanding. The use of cameras for traffic monitoring has enabled gathering useful information in real time including traffic speed, lane occupancy, traffic density, etc., from very large areas and has provided more flexible solutions compared to the traditional magnetic loop radars, microwave, infrared detectors which are limited on a single point, high installation cost, and are difficult to install and maintain [1, 2]. In this context, it has been shown that DCNNs methods outperform traditional methods in pattern recognition [3, 4], due to their powerful ability to adaptively learn complex features for object recognition. Therefore, DCNNs are considered as a powerful tool and showed better performance than other conventional methods in many domains, such as crowd behavior analysis [5], image classification and detection [6], video analysis [7], document recognition [8], and so on. For instance, in [9], authors proposed a multi-column deep neural networks from 25 nets that alternates convolutional traffic with max-pooling layers, and they reached 99,46 % of accuracy in traffic sign recognition, better than the human performance 98,84 % on this task. In recent years, several techniques have been used in the literature for traffic density estimation from traffic surveillance. We can distinguish generally three main types (i) detection based methods, (ii) holistic approaches and (iii) motion based methods.

Holistic approaches perform the analysis on the whole of the image, by avoiding detecting each moving object separately [10]. Some properties such as crowd speed, density and localization can be extracted from crowd behavior analysis [11, 12]. Some related works that apply holistic approach for traffic can be found in [13, 14, 15]. These kind of approaches suffer from low accuracy when the camera has large perspective.

Motion based methods, in this approach, several techniques [16, 17, 18] estimate the traffic density by vehicle tracking. This kind of approaches calculates vehicle trajectories across frames, by partitioning an image into regions. These methods tend to fail in high congestion because of the web-cam video's low resolution, low frame rate and lack of motion information.

Detection based methods aim at identifying and localizing vehicles in each frame. For example in [19], authors proposed a real-time traffic congestion estimation approach, based on image texture feature extraction and texture analysis. The experimental results showed a high accuracy for obtaining the vehicle density. Even if the detection-based model has shown a good results in many fields, it still perform poorly in low resolution and high occlusion videos.

In this paper, we propose a robust traffic-density classification model based on a deep neural network architecture, which overcomes all the issues faced by existing techniques. A DCNN is used to estimate the road traffic density in order to classify the traffic flow into three main classes (light, medium, heavy) in different environmental conditions. The remaining of this paper is

Manuscript received July 5, 2018
Manuscript revised July 20, 2018

organized as follows. Section II presents a background about CNNs. Section III provides a detailed description of our DCNN architecture. Section IV discusses the experimental results by presenting an evaluation of our method performance and a comparison with different approaches used on the same Database. Concluding remarks are given in section V.

2. Proposed approach

In this section, we present a background about CNNs and describe our DCNN in more details and highlight the techniques performed to learn the optimal architecture in order to increase the classification accuracy.

2.1 Background

CNNs are a special kind of neural network formed by a certain number of convolutional and subsampling layers stacked on top of each other, depending on the particular application or on the complexity of the data. The architecture of a typical CNN is composed of one or more pairs of convolution and pooling layers and finally followed by fully connected layers as seen in Fig. 1. A convolutional layer, which represents the core building block of a CNN, is composed of a certain number of convolutional filters that are applied to the whole input image in order to detect local features. The output of each convolution is called an activation map.

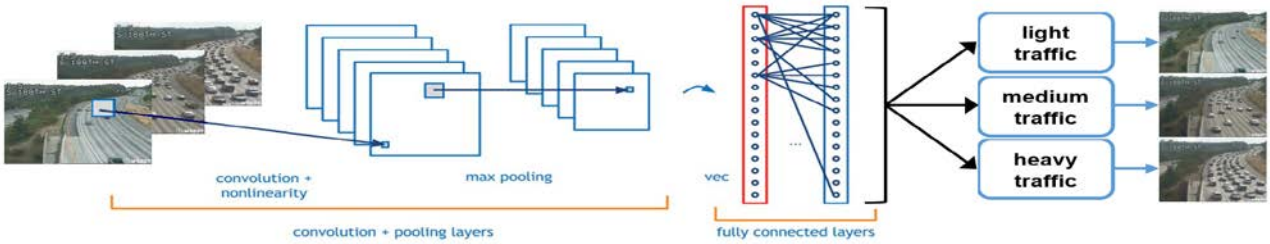

Fig. 1 View of a general CNN architecture which classifies the traffic density as heavy traffic, medium traffic and light traffic.

After each convolutional layer, there may be a pooling layer. The pooling layers are a form of down-sampling, i.e. reducing the resolution of the activation maps at a given point in the network. There are three main non-linear functions to produce pooling, sum pooling, mean pooling and max pooling. The conceptual difference between these approaches lies in the sort of invariance, which they are able to catch. For example, the max-pooling which is the most used, takes the maximum input from a region of the convolutional layer. The sum and mean pooling are set exactly the same as max-pooling but instead using a max function, we use the sum or the mean function. This technique is used to reduce the number of parameters within the model, in order to simplify the computational load first, and secondly to minimize the chance of over-fitting. Finally, after several convolutional and max pooling layers, the high-level features found in the images are processed via fully connected layers with the aim of activating only one output per class in the recognition task. In fact, it creates a stochastic likelihood representation of each class based on the activation maps generated by the concatenation of the previous layers.

2.2 Model description

Our model architecture is composed of first 5 repeatedly stacked blocks excluding the input layer. Each block contains a convolution module, followed by a max pooling module and a normalization module. To regularize our model, we introduced a dropout “layer” in order to prevent over-fitting. This technique randomly drops out weight connections, which are ignored during training. The last part in our CNN is constituted by two dense layers, which are called classifiers too. These layers are fully connected layers, which represent a matrix vector multiplication and need a feature vector as an input. Therefore, we used the flattening operation, in order to convert the output of the convolutional part of the CNN into 1D feature vector. In the following Fig. 2, we show our model architecture in details. For the input layer we introduced a crop images of [238*198*3] to avoid to process non relevant data by our algorithm, with three color channels RGB. The receptive field (or the filter size) is 3*3, then each neuron in the convolutional layer has weights to a [3*3*3] in the input volume, for a total of 3*3*3 = 27 weights (and +1 bias parameter). For example, the first convolutional layer has 8 filters so we get (27+1) * 8 = 224 parameters. Similarly, we can calculate for 2nd convolutional layer where the number of filter from the previous layer become the number of channels for current layer’s input. In a fully-connected layer, all input units have a separate weight to each output unit. For n inputs and m outputs, the number of weights is (n+1) *m, which explain 163904 parameters for the dense layer where n=2560 and m=64.

Layer (type)	Output Shape	Param #	Connected to
convolution2d_6 (Convolution2D)	(None, 8, 238, 198)	224	convolution2d_input_2[0][0]
maxpooling2d_6 (MaxPooling2D)	(None, 8, 119, 99)	0	convolution2d_6[0][0]
convolution2d_7 (Convolution2D)	(None, 16, 117, 97)	1168	maxpooling2d_6[0][0]
maxpooling2d_7 (MaxPooling2D)	(None, 16, 58, 48)	0	convolution2d_7[0][0]
convolution2d_8 (Convolution2D)	(None, 32, 56, 46)	4640	maxpooling2d_7[0][0]
maxpooling2d_8 (MaxPooling2D)	(None, 32, 28, 23)	0	convolution2d_8[0][0]
convolution2d_9 (Convolution2D)	(None, 64, 26, 21)	18496	maxpooling2d_8[0][0]
maxpooling2d_9 (MaxPooling2D)	(None, 64, 13, 10)	0	convolution2d_9[0][0]
convolution2d_10 (Convolution2D)	(None, 128, 11, 8)	73856	maxpooling2d_9[0][0]
maxpooling2d_10 (MaxPooling2D)	(None, 128, 5, 4)	0	convolution2d_10[0][0]
dropout_3 (Dropout)	(None, 128, 5, 4)	0	maxpooling2d_10[0][0]
flatten_2 (Flatten)	(None, 2560)	0	dropout_3[0][0]
dense_3 (Dense)	(None, 64)	163904	flatten_2[0][0]
activation_2 (Activation)	(None, 64)	0	dense_3[0][0]
dropout_4 (Dropout)	(None, 64)	0	activation_2[0][0]
dense_4 (Dense)	(None, 3)	195	dropout_4[0][0]

Total params: 262483

Fig. 2 Our CNN architecture

We developed our architecture by varying one of three parameters: number and sizes of kernels, number of layers and pooling strategies. The variation of all these parameters impacts the network performance in different aspects over multiple runs. The increase of convolutional kernels enhanced the network's capacity of detecting and extracting patterns and co-occurrence of patterns. The increase of convolution and max-pooling layers make the network 'deeper' to learn more sophisticated data representations. The insertion of pooling layers between successive convolutional layers periodically decreases the spatial size of representations and the amount of parameters and computation in the network.

The second key ingredient is the loss function used in our model, which can be seen as a differentiable objective that measures the compatibility between a prediction and the ground truth label in the training data (also called a "scoring function"). Intuitively, we want the correct class to have a higher score than the other classes. When this is the case, the loss should be low and otherwise the loss should be high. There are many ways to quantify this intuition, but when using a DCNN to perform classification and prediction on more than two classes, it's usually better to use the cross-entropy loss associated with the Softmax classifier. With this combination, the output prediction is always between zero and one, and is interpreted as a discrete probability distribution over the classes. the Softmax activation where the function mapping is $Z(x_i; W) = W x_i$ always unchanged of the ith output unit with W is the filter weights is defined as

$$\text{Softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_i)} \quad (1)$$

and the cross entropy error function for multi-class output is

$$L_i = -\log\left(\frac{\exp(z_i)}{\sum_j \exp(z_i)}\right) \quad (2)$$

2.3 Visualizing features detectors

Deep learning algorithms are known to be very complicated to interpret, that's why they are usually treated as black boxes. However DCNN algorithms are actually different, and we can visualize various components. This will give us an in depth look into their internal workings and help us understand them better. We presented four main visualizations for each layer to help readers understand how our DCNN learns features in intermediate layers. We passed an input picture through our DCNN and record the intermediate activations as shown in Fig. 3. These visualizations served as supporting information to help us assess hypotheses about the cause

of certain types of errors and understand the relation between the different classes (i.e. heavy, medium, light).

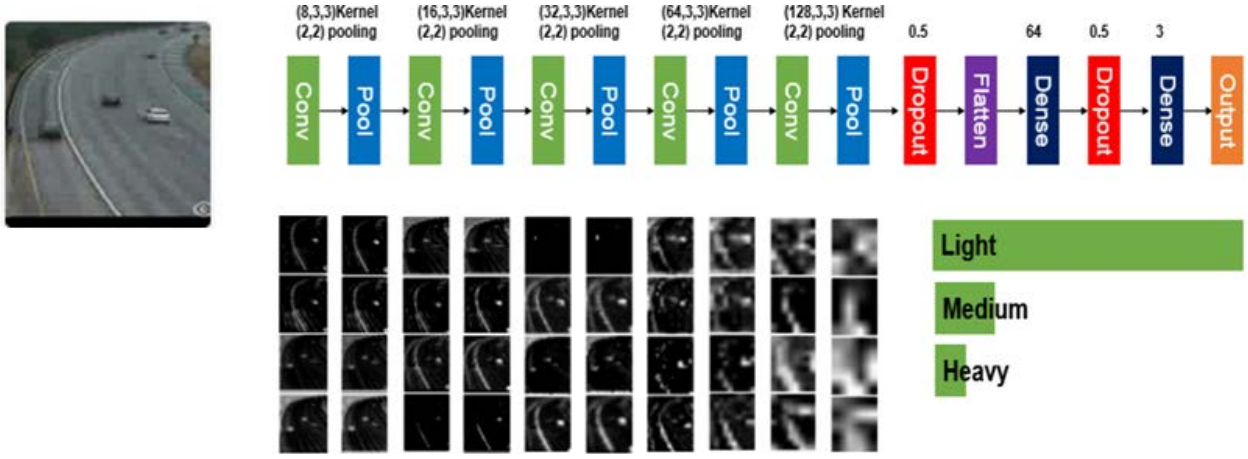

Fig. 3 The structure and intermediate processing layer results of our CNN model

Each filter in the DCNN is associated with a feature detector or neuron that learns during training to be particularly active when presented with a specific sequence of input images. Therefore, the algorithm is able to perform image classification by looking for low level and easy interpretable features such as edges and curves, and then building up to more abstract concepts through a series of convolutional layers. It will be more interesting to visualize multiple feature maps from each convolutional layer. Fig. 4 shows a direct comparison between the results of each convolutional layer applied on the same original image used in Fig. 3.

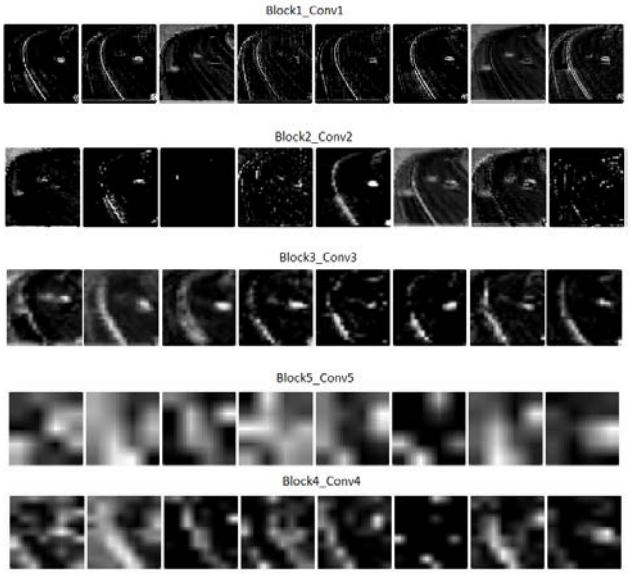

Fig. 4 Visualization of the features maps

As we go deeper through the network, the feature maps look more like an abstract representation and less similar to the original image. As we can see in block3_conv3 which contains 32 filters, the cars are somewhat visible, but after that it becomes unrecognizable. The reason is that deeper feature maps encode high level concepts like “headlights” or “Wheels” while lower level feature maps detect basic texture and borders. That’s why deeper feature maps contain less information about the image and more about the class of the image and they still encode useful features, but they are less visually interpretable by us.

3. Experimental results and evaluation

In this section, we present the results obtained by our approach. Firstly, the data set is described and then the classification results.

3.1 Traffic video databases

The traffic video database consists of 254 video sequences of highway traffic in Seattle, Washington, collected from a single stationary traffic camera over two days [14]. The database contains a variety of traffic patterns and weather conditions (e.g. raining, overcast and sunny). Each video has a resolution of 320*240 pixels and has 42 to 52 frames at 10 frames per second. The database was labeled by hand concerning the level of traffic congestion in each sequence. The database presents 165 sequences of light traffic, 45 of medium traffic and 44 of heavy traffic (very slow), in different environmental conditions as shown in Fig. 5.

Fig. 5 Example frames of the traffic video data set. These examples show various traffic congestion conditions. From the left to right, labeled as heavy traffic, medium (raining) and light (Dark) traffic.

3.2 CNN activation maps

Class activation mapping [24] is a technique used in order to show a visual explanation for our CNN-based model and make it more transparent by obtaining the relevant image regions used by our CNN to make the correct decision. Fig. 6 shows on top of the input image, the sum of filters at each layer of our CNN using a red-blue heat map visualization. The more red a pixel, the more relevant it is for the classification task. This helps us understand what our network is doing with a given input, and somehow allow us to be sure that the network is not using non relevant information from the images at hand.

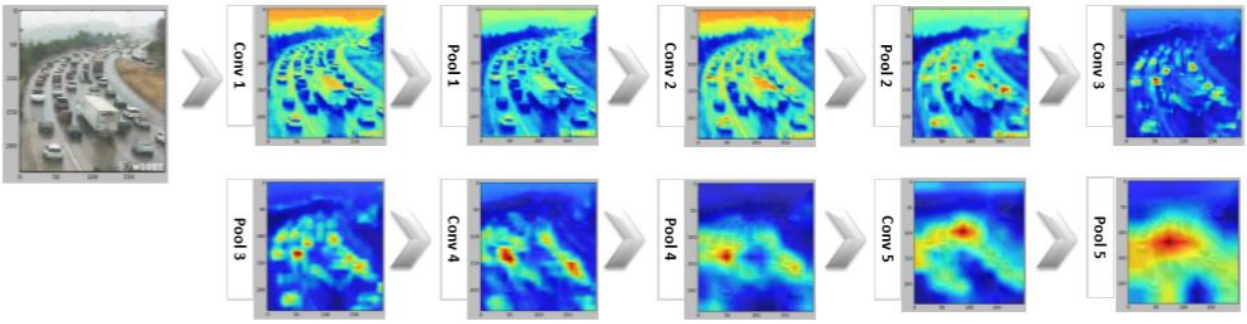

Fig. 6 The sum of activation maps generated to visualize each convolutional layer in a 2D grid of a trained CNN.

Fig. 7 shows the Sum of Activation Maps for two sample input images at the last convolutional layer of our CNN. We can observe that our model is focusing on the cars for estimating high density traffic and on the space between cars (absence of cars) to assess a low density traffic. This shows that our model makes the right decision based on the correct information. Thus proving the robustness of our model and explaining the high accuracy that we get.

3.3 Results and comparison

Our traffic density estimation approach has been applied on the database described before. The results from the proposed system have been compared to many other techniques applied on the same database. Our proposed CNN-based method has reached the best accuracy in this field by reaching 99, 62% as seen in Table 2.

In our approach, we split the database into two random samples (80% and 20 %) for training and testing, which represents 5678 and 1420 frames respectively. In this way, we are sure that our training and testing sets reflect the real properties of the original dataset. Table 1 provides a confusion matrix for the resulting classifier. We can observe that the miss-classifications occur between the medium and heavy classes, due to their close texture pattern. Therefore, we evaluated our model using 5-fold cross validation method. We partitioned our data into 5 equal folds, four parts are used to train our CNN model and rest used to test. The process is repeated five times by exploring the remaining folds for training and testing. The final performance measure, computed as the accuracy across all five models, was around 99, 62%. The inconvenience of this method is that the training algorithm has to be repeated from scratch 5 times, which means it takes 5 times as much computation to make an evaluation.

Fig. 7 Sum of Activation Maps of a high traffic density image (right) and of a low density traffic image (left).

Table 1: Confusion matrix

Actual predicted	Light	Medium	Heavy
Light	458	0	0
Medium	0	493	2
Heavy	0	3	464

Table 2 presents the comparison of the proposed method and other classification techniques, which were evaluated on the same database. Our proposed system achieves the best accuracy comparatively to all the other methods.

Table 2: Different traffic density estimation approaches

Method	**Accuracy %**
Our proposed approach	99.62 %
Symbolic features [23] (2013)	96.83 %
Probabilistic Kernels [20] (2005)	96.00 %
Spatiotemporal Orientation [21] (2011)	95,28 %
Motion Vector Statistical Features [16] (2013)	95.28 %
Holistic Approach [12] (2013)	94.50 %
Dynamic Texture Method [14] (2005)	94.50 %

The most important quantity to track while training a classifier is the validation/training accuracy which is used to measure the inconsistency between predicted value and actual label. To determine the accuracy of our model, we calculated the classification error after that the parameters of our algorithm are learned and fixed and no learning process is taking place. We fed our model by the test samples and we recorded the number of mistakes (zero-one loss) that the model makes after comparison to our true targets. Then the percentage of misclassification is calculated. We plot the model accuracy and the model loss over epochs in Fig. 8 and Fig. 9 respectively, from which we can understand that our train process took 5 – 6 iterations to achieve a similar accuracy of 99.62%.

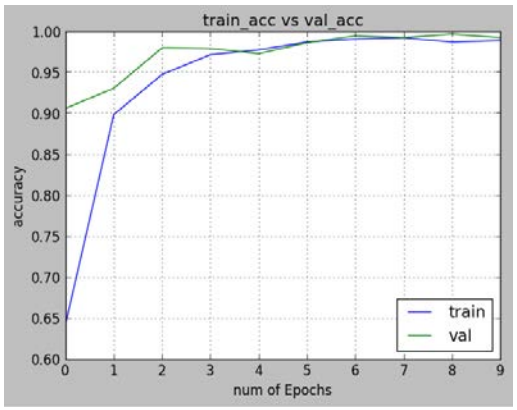

Fig. 8 Accuracy over epochs

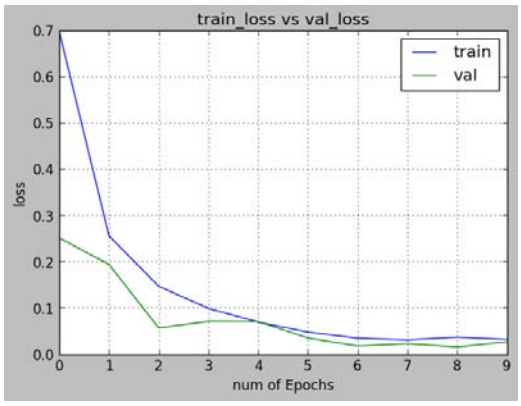

Fig. 9 Loss over epochs

4. Conclusion

In this paper, we presented a DCNN traffic density estimation approach for traffic monitoring systems through video processing. Our method classifies the traffic flow density into three different states: light, medium and heavy. A standard database of real videos from Seattle roads was used to experiment our proposed approach. The

results showed that our model outperformed all the conventional existing methods by reaching 99, 62% of accuracy. For future work, the proposed approach has some possible interesting extensions. For instance, it would be interesting to explore other deep learning algorithms for traffic flow estimation and adapt and apply these algorithms on different data sets to evaluate their potential. It can be also a good issue to combine the CNN with the Ant colony algorithm. Specifically, CNN can extract the traffic features from the traffic network that can be involved into the Ant colony algorithm in order to manage the urban traffic.

Acknowledgment

The authors would like to thank Jérémie Despraz for his technical support and assistance with this project. This research was supported by The School of Business and Engineering Vaud (HEIG-VD), Switzerland.

References

[1] Cucchiara, R., Piccardi, M., & Mello, P. (2000). Image analysis and rule-based reasoning for a traffic monitoring system. IEEE Transactions on Intelligent Transportation Systems, 1(2), 119-130.

[2] Kastrinaki, V., Zervakis, M., & Kalaitzakis, K. (2003). A survey of video processing techniques for traffic applications. Image and vision computing, 21(4), 359-381.

[3] Sun, Y., Wang, X., & Tang, X. (2014). Deep learning face representation from predicting 10,000 classes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1891-1898).

[4] Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011, June). Flexible, high performance convolutional neural networks for image classification. In Twenty-Second International Joint Conference on Artificial Intelligence.

[5] S. Jing, K. Kai, L. Chen, Chang, and W. Xiaogang. Deeply learned attributes for crowd scene understanding. In CVPR,2015.S. Jing, K. Kai, L. Chen, Chang, and W. Xiaogang. Deeply learned attributes for crowd scene understanding. In CVPR,2015.

[6] Ciregan, D., Meier, U., & Schmidhuber, J. (2012, June). Multi-column deep neural networks for image classification. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on (pp. 3642-3649). IEEE.

[7] Zou, W. Y., Ng, A. Y., Zhu, S., & Yu, K. (2012, December). Deep Learning of Invariant Features via Simulated Fixations in Video. In NIPS (Vol. 3, p. 6).

[8] Cireşan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. Neural computation, 22(12), 3207-3220.

[9] Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011, June). Flexible, high performance convolutional neural networks for image classification. In Twenty-Second International Joint Conference on Artificial Intelligence.

[10] J. Jacques Junior, S. Musse, and C. Jung. Crowd analysis using computer vision techniques. IEEE Signal Processing Magazine, 27(5):66–77, sept. 2010.

[11] B. Zhan, D. N. Monekosso, P. Remagnino, S. A. Velastin, and L.-Q. Xu. Crowd analysis: a survey. Machine Vision Applications, 19(5-6):345–357, 2008.

[12] Andrews Sobral, L. O., Schnitman, L., & De Souza, F. (2013). Highway traffic congestion classification using holistic properties. In 10th IASTED International Conference on Signal Processing, Pattern Recognition and Applications.

[13] J. Lee and A. Bovik. Estimation and analysis of urban traffic flow. In 16th IEEE International Conference on Image Processing, pages 1157 –1160, nov 2009.

[14] A. Chan and N. Vasconcelos. Classification and retrieval of traffic video using auto-regressive stochastic processes. In IEEE Intelligent Vehicles Symposium (IVS), pages 771–776, june 2005.

[15] F. Porikli and X. Li. Traffic congestion estimation using hmm models without vehicle tracking. In IEEE Intelligent Vehicles Symposium (IVS), pages 188–193, june 2004.

[16] A. Riaz and S. A. Khan. Traffic congestion classification using motion vector statistical features. In Sixth International Conference on Machine Vision (ICMV 13), pages 90671A-7. International Society for Optics and Photonics,2013.

[17] Keck, M., Galup, L., & Stauffer, C. (2013, January). Real-time tracking of low-resolution vehicles for wide-area persistent surveillance. In Applications of Computer Vision (WACV), 2013 IEEE Workshop on (pp. 441-448). IEEE.

[18] Hadi, R. A., Sulong, G., & George, L. E. (2014). Vehicle detection and tracking techniques: a concise review. arXiv preprint arXiv:1410.5894.

[19] Wei, L., & Hong-ying, D. (2016). Real-time road congestion detection based on image texture analysis. Procedia Engineering, 137, 196-201.

[20] Chan, A. B., & Vasconcelos, N. (2005, June). Probabilistic kernels for the classification of auto-regressive visual processes. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (Vol. 1, pp. 846-851). IEEE.

[21] Derpanis, K. G., & Wildes, R. P. (2011, January). Classification of traffic video based on a spatiotemporal orientation analysis. In Applications of Computer Vision (WACV), 2011 IEEE Workshop on (pp. 606-613). IEEE.

[22] Dinani, M. A., Ahmadi, P., & Gholampour, I. (2015, November). Efficient feature extraction for highway traffic density classification. In Machine Vision and Image Processing (MVIP), 2015 9th Iranian Conference on (pp. 14-19). IEEE

[23] E. Dallalzadeh, D. S. Guru, and B. S. Harish. Symbolic Classification of Traffic Video Shots. In Advances in Computational Science, Engineering and Information Technology,pages 11–22. Springer, 2013

[24] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. CVPR'16 (arXiv:1512.04150, 2015).