Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 144 (2018) 291–297

Procedia Computer Science

www.elsevier.com/locate/procedia

INNS Conference on Big Data and Deep Learning 2018

Traffic Congestion Detection: Learning from CCTV Monitoring Images using Convolutional Neural Network

Jason Kurniawan[a], Sensa G.S. Syahra[a], Chandra K. Dewa[b], Afiahayati[a,*]

[a]*Department of Computer Science and Electronics, Universitas Gadjah Mada, Yogyakarta, Indonesia*
[b]*Department of Informatics, Universitas Islam Indonesia, Yogyakarta, Indonesia*

Abstract

In this paper, we present an intelligent traffic congestion detection method using image classification approach on CCTV camera image feeds. We use a deep learning architecture, convolutional neural network (CNN) which is currently the state-of-the art for image processing method. We only do minimal image preprocessing steps on the small size image, where the conventional methods require a high quality, handcrafted features need to do manual calculation. The CNN model is trained to do binary classification about road traffic condition using 1000 CCTV monitoring image feeds with balance distribution. The result shows that a CNN with simple, basic architecture that trained on small grayscale images has an average classification accuracy of 89.50%.

Keywords: convolutional neural network; deep learning; image classification; traffic congestion detection

1. Introduction

Road traffic condition is one of major problems in big cities especially in the developing countries, like in Indonesia. There are so many bad effects caused by this traffic problem, like massive delays and the increased fuel wastage and monetary losses [1]. The road traffic could be happened because of limited number of facilities and infrastructures, huge number of people, and inappropriate policy from the government. Based on the data from

* Corresponding author. Tel.: +62-274-546194; fax: +62-274-546194.
E-mail address: afia@ugm.ac.id

10.1016/j.procs.2018.10.530

tomtom traffic index [2], Jakarta is the 3rd city with traffic condition in the world. In order to tackle this problem, Indonesian government, especially Jakarta's government try to implement the concept of smart city technology. The government provides CCTV in selected area to monitor the traffic condition.

Intelligent Traffic System (ITS) has been developed to solve the road traffic condition. It uses supported data, such as airbone optical remote sensing sensor [3], wireless signal communication (i.e probe vehicle-to-vehicle) [4][5][6][7]. Nevertheless, in the developing countries, the first problem is that the data is not available because of the expensive infrastructure and maintenance cost. Another alternative is utilizing data from traffic video [8] and captured image [1] with manually processing by human. Manually processing requires handcrafted features and manual calculation like calculating the distance and the level of the traffic congestion between vehicle. Manually processing depends on the human ability and requires time which is not short. Therefore, in this research, we propose a method to detect the road traffic congestion automaticly by utilizing the data from CCTV camera image feeds. We conduct a series of computaional experiments for the road traffic data in Jakarta. We implement the Convolutional Neural Network and preprocessing of the data.

2. Convolutional Neural Network

Convolutional Neural Network (CNN) is a variant of standard neural network which is specifically designed to process sequence data such as image [9]. It requires minimal data preprocessing and it can automatically detect the invariant and extract important features [10]. There are two main components of CNN, convolutional layer and pooling layer [11]. The components are mainly inspired by mammalian visual cortex that have two basic cell types: complex cells (convolution layer) which have receptive fields and are locally invariant to exact position of the pattern, and simple cells (pooling layer) that respond maximally to specific edge-like patterns within their receptive field. A basic CNN usually consists of one or more convolution-pooling layers and fully-connected layer(s). CNN uses three main idea: local receptive fields, shared weights, and pooling [12].

2.1. Convolutional Layer

Convolution layer is the layer that performs the main operation of CNN. This layer is used to generating feature maps of a spatial input through convolution operations. In the convolution operation, a filter/kernel with certain spatial size along the input features. Suppose we have input feature that represented as a two-dimensional feature map x with size $p_1\ x\ q_1$, convolutional kernel W with size of $m\ x\ n$, and shift s. The convolutive operation can be denoted as $C = X * \mathrm{W}$, where output C is called as feature map. Output C has size of $p_2 \times q_2$, where $p_2 = 1 + (p_1 - m)/s$ and $q_2 = 1 + (q_1 - m)/s$. Each neuron unit in the convolution layer is connected to the receptive field units in the corresponding local area of size $m \times n$.

Suppose $W_{i,j}$ is the weight parameyer that represented as convolutive kernel that connects *i*-th feature map from previous layer C_i to *j*-th feature map C_j and b_j is the corresponding bias. One feature map in the convolutional layer can be computed as:

$$C_j = \sigma(\sum_{i \in S} W_{i,j} * C_i + b_j) \tag{1}$$

where S is the set of the selected feature maps from the previous layer and σ denotes the activation function, which can be a sigmoid $\sigma(x) = 1/(1 + e^{-x})$, hyperbolic tangent $\sigma(x) = (1 - e\text{^}(-2x))/(1 + e\text{^}(-2x))$ or rectified linear units (ReLU) $\sigma(x) = \max(0, x)$.

2.2. Pooling Layer

Pooling layer is another main component of CNN, which perform a non-linear down-sampling. Max pooling is the most commonly non-linear functions used on the pooling operation. This layer is composed a down-sampled feature maps generated by applying pooling operation on the local area of the corresponding feature maps in the convolution layer. Hence, the number of feature maps is the same but in smaller size. The purpose of this layer is to

reduce the resolution of feature maps, so then the number of parameters and amount of computation can be reduced. The size of the local area is determined by a parameter called pooling size. A local area may have overlapped area with the adjacent area according to the area shift parameter called pooling stride.

3. Research Method

3.1. Dataset

The dataset used in this experiment is the road traffic condition images from CCTV camera in Jakarta during 29 April – 5 May 2017 that can be obtained from lewatmana.com [13]. We choose 14 different locations on various times, then manually captured and labeled the CCTV camera images. The label for this dataset is binary: "jammed" which is indicated a traffic jam condition, and "not jammed" which is the otherwise. The dataset contains 1000 images with balanced label distribution. The example of original images for each condition can be seen on Figure 1. The captured images were originally 640x480 pixels colored images, but then we converted it into 100x100 pixels grayscale images. We did this conversion since the light intensity and color difference between daytime and night time would not give useful information to our model and higher image resolution would require higher computational resource. We then used these converted images as the input to train our CNN model.

Fig. 1. Some images in the dataset

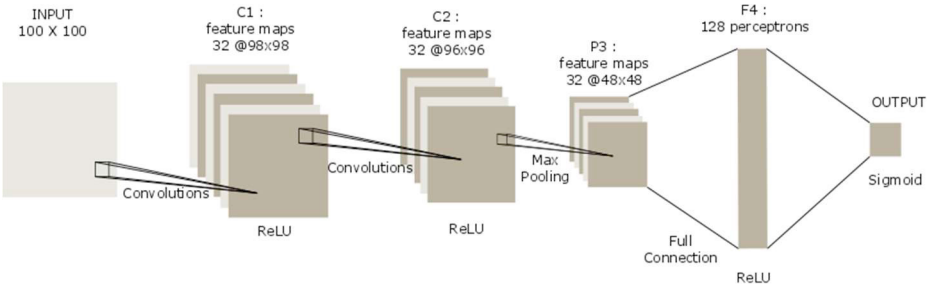

Fig. 2. The architecture of our CNN model

3.2. CNN Architecture

For the CNN architecture used in our experiments, we used two convolutional layers, a max pooling layer, and a fully connected layer. The illustration of our CNN architecture is shown in Figure 2. The first layer C1 is convolution layer that uses 3x3 filters and 32 feature maps, therefore since the input size is 100x100, the size of each

feature map is 98x98. The second layer C2 is another convolution layer that also uses 3x3 filters and 32 feature maps with 93x96 size each. The third layer P3 is 2x2 max pooling layer which used for down-sampling each feature map into 48x48 size. The last hidden layer is fully connected layer consist of 128 perceptrons, each perceptron is fully connected with each unit of the feature maps from P3. We used Rectified Linear Units (ReLU) activation function in both the convolutional and fully connected layers. On the output layer, we used a perceptron with sigmoid activation function.

3.3. Training and Evaluation

To train our CNN model, we used mini-batch gradient descent with Adam [14] optimization. We implemented our CNN model using Python with Keras library [15] that running on top of Theano library [16]. The weight initialization were set as suggested by [17], we also use Dropout [18] with probability of 0.5 for the reguralization. For the mini-batch gradient descent we used 250 batch size and 100 epoch. Hence, it took (number of train sample)/(batch size) iterations to complete an epoch, and for each iteration, a batch of 250 images was presented to the CNN model and then the weights were updated by backpropagation. Binary cross entropy is used for the objective function.

For the CNN model validation, we used stratified k-fold cross validation with k=10. Stratified 10-fold cross validation randomly shuffled and splitted the complete dataset into 10 subsets, each subset has the same class distribution as the complete dataset. We then evaluated the accuracy of each fold, and the overall measure of accuracy is the mean of accuracy of all folds combined.

4. Results and Analysis

The result of each fold from the 10-fold cross validation is shown in Table 1. The highest accuracy that our CNN model can achieve is 93% and the lowest is 82%. From all these results, our trained CNN model achieved an average accuracy of 89.50% (+/- 3.67%). This shows that our simple, basic CNN model can be used to classify CCTV camera traffic condition images with minimal preprocessing steps and it can produce an acceptable result on small size greyscale images.

Table 1. The accuracy for each fold from the 10-fold cross validation

Fold	Accuracy(%)
I	91
II	91
III	82
IV	93
V	83
VI	92
VII	89
VIII	90
IX	92
X	91.91

The train and validation accuracy for each fold is shown in Fig. 3. The accuracy on the validation data is increased significantly until the 20th epoch.

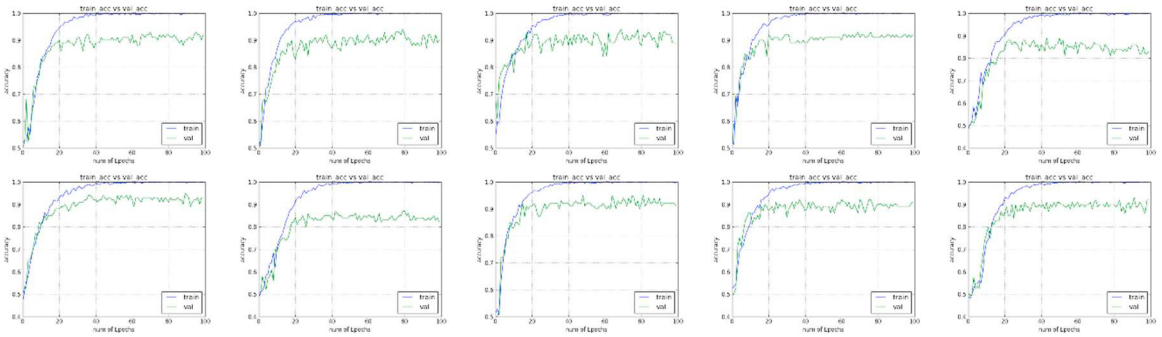

Fig. 3. The train (blue) and validation (green) accuracy for each fold

The train and validation loss for each fold is shown in Fig. 4. The loss on the validation data is significantly decreased until the 20th epoch, but on the next epochs it trend is increased. This mean that our model is converged fast until 20th epoch and the greater epoch would not give any significant change on the classification result. For each model from each fold, we only take the best epoch which has the lowest validation loss.

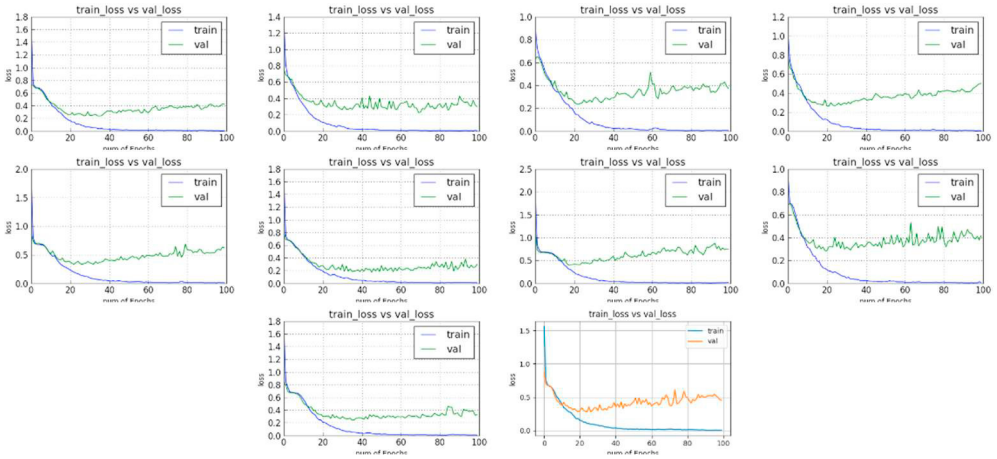

Fig. 4. The train (blue) and validation (green) loss for each fold

5. Conclusion and Future Work

This paper has presented a convolutional neural network for traffic congestion detection by using image classification approach. The CNN model has achieved an average accuracy of 89.50% on the CCTV camera monitoring captured images dataset. We only resize and convert the images into 100x100 grayscale images, and did not use any handcrafted features on the preprocessing steps. For the future work, we plan to enhance our CNN architecture and use higher resolution images to improve our model classification performance. The model also can be implemented to a system, so we can detect traffic congestion automatically using real time captured CCTV camera captured image on specific location and/or integrate the detection result with map/navigation application to prevent further traffic congestion. The proposed system is shown in Fig. 5.

First, user will send a request about traffic condition on certain location on the current time. The system then will capture a real time CCTV camera image from the requested location. After that, the system will convert the captured image into into smaller grayscale image. By using the trained CNN model, the system will detect the traffic congestion condition by using classification approach. If the output of the model is “jammed” it indicates that a traffic jam is occurs on the requested location, and “not jammed” is the otherwise. This output will be sent back to

user as a feedback. The system also can be integrated with navigation application it can detect multiple traffic congestion at the same time so then it can suggest the user several alternative routes.

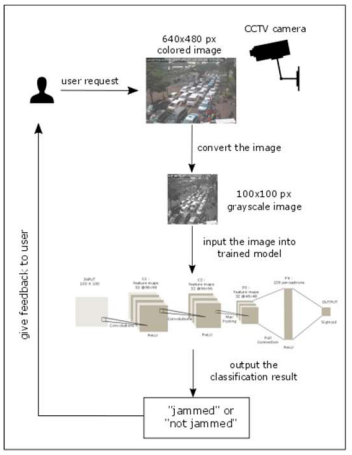

Fig. 5. The proposed traffic congestion detection system

References

[1] Jain V., Sharma A. and Subramanian L. (2012) Road traffic congestion in the developing world. In Proceedings of the 2nd ACM Symposium on Computing for Development 2012 Mar 11 (p. 11). ACM.

[2] TomTom International BV. TomTom Traffic Index [Internet]. 2017 [cited 18 July 2017]. Available from: https://www.tomtom.com/en_gb/trafficindex/list.

[3] Palubinskas, G., Kurz F. and Reinartz, P. (2008) Detection of traffic congestion in optical remote sensing imagery. In Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International 2008 Jul 7 (Vol. 2, pp. II-426). IEEE.

[4] Bauza, R., Gozalvez, J. and Sanchez-Soriano, J. (2010) Road traffic congestion detection through cooperative vehicle-to-vehicle communications. In Local Computer Networks (LCN), 2010 IEEE 35th Conference on 2010 Oct 10 (pp. 606-612). IEEE.

[5] Lakas, A. and Cheqfah, M. (2009) Detection and dissipation of road traffic congestion using vehicular communication. In Microwave Symposium (MMS), 2009 Mediterrannean 2009 Nov 15 (pp. 1-6). IEEE.

[6] Roy, S., Sen, R., Kulkarni, S., Kulkarni, P., Raman, B. and Singh, L.K. (2011) Wireless across road: RF based road traffic congestion detection. In Communication Systems and Networks (COMSNETS), 2011 Third International Conference on 2011 Jan 4 (pp. 1-6). IEEE.

[7] Mandal, K., Sen, A., Chakraborty, A., Roy, S., Batabyal, S. and Bandyopadhyay, S. (2011) Road traffic congestion monitoring and measurement using active RFID and GSM technology. In Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on 2011 Oct 5 (pp. 1375-1379). IEEE.

[8] Pongpaibool, P., Tangamchit, P. and Noodwong, K. (2007) Evaluation of road traffic congestion using fuzzy techniques. In TENCON 2007-2007 IEEE Region 10 Conference 2007 Oct 30 (pp. 1-4). IEEE.

[9] Dewa, C.K., (2016). Javanese vowels sound classification with convolutional neural network, in: Proceedings of International Seminar on Intelligent Technology and Its Applications (ISITIA), pp.123-127.

[10] Rajagede, R.A, Dewa, C.K. and Afiahayati (2017) Recognizing Arabic letter utterance using convolutional neural network, in: 2017 18th IEEE/ACIS International Conference on software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 181-186.

[11] Dewa, C.K., Fadhilah, A.L. and Afiahayati, (2018) "Convolutional neural networks for handwritten Javanese character recognition." *Indonesian Journal of Computing and Cybernetics Systems*, **12**: 83-94.

[12] Michael A. Nielsen. Neural Networks and Deep Learning. 2015 [cited 18 July 2017]. Determination Press. Available from : http://neuralnetworksanddeeplearning.com/.

[13] Lewatmana.com. Live Traffic CCTV. 2017 [cited 18 July 2017]. Available from : http://lewatmana.com

[14] Kingma, D. and Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014 Dec 22.

[15] Fran'cois, C.,et al. Keras [Internet]. GitHub repository. 2015 [cited 18 July 2017]. Available from: https://github.com/fchollet/keras.

[16] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D. and Bengio, Y. Theano (2010) A CPU and GPU math compiler in Python. In Proc. 9th Python in Science Conf 2010 Jun (pp. 1-7).

[17] Glorot, X. and Bengio, Y. (2010) Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics 2010 Mar 31 (pp. 249-256).

[18] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research. 2014 Jan 1;15(1):1929-58.