Traffic congestion detection from camera images using deep convolution neural networks

Pranamesh Chakraborty (**Corresponding Author**)
PhD student, Department of Civil, Construction and Environmental Engineering
Iowa State University, Ames, IA 50011
Phone: (515) 357-9076. Email: pranames@iastate.edu

Yaw Okyere Adu-Gyamfi
Data Scientist, Institute for Transportation
2711 South Loop Drive, Suite 4700, Ames, IA 50010
Phone: (302) 883-7460. Email: yaokyere@udel.edu

Subhadipto Poddar
PhD student, Department of Civil, Construction and Environmental Engineering
Iowa State University, Ames, IA 50011
Phone: (515) 598-6807 Email: spoddar@iastate.edu

Vesal Ahsani
PhD student, Department of Civil, Construction and Environmental Engineering
Iowa State University, Ames, IA 50011
Phone: (515) 708-0860 Email: ahsaniv@iastate.edu

Anuj Sharma
Associate Professor, Department of Civil, Construction and Environmental Engineering
Iowa State University, Ames, IA 50011
Phone: (515) 294-3624 Email: anujs@iastate.edu

Soumik Sarkar
Assistant Professor, Department of Mechanical Engineering
207 Lab of Mechanics, Iowa State University, Ames, IA 50011
Phone: (515) 294-5212 Email: soumiks@iastate.edu

Submission Date: 15th November, 2017

Word Count: 5742 words of text + (4 figures + 3 tables) × (250 words each) = 7492 words

ABSTRACT

Recent improvements in machine vision algorithms have led to CCTV cameras emerging as an important data source for determination of the state of traffic congestion. In this study we used two different deep learning techniques, you only look once (YOLO) and deep convolution neural network (DCNN), to detect traffic congestion from camera images. The support vector machine (SVM), a shallow algorithm, also was used as a comparison to determine the improvements obtained using deep learning algorithms. Occupancy data from nearby radar sensors were used to label congested images in the dataset and for training the models. YOLO and DCCN achieved 91.5% and 90.2% accuracy, respectively, whereas SVM's accuracy was 85.2%. Receiver operating characteristic curves were used to determine the sensitivity of the models with regard to different camera configurations, light conditions, etc. Although poor camera conditions at night affected the accuracy of the models, the area under the curve from the deep models were found to be greater than 0.9 for all conditions. This shows that the models can perform well in challenging conditions as well.

INTRODUCTION

The US Department of Transportation (DOT) defined traffic congestion as "one of the single largest threats" to the economic prosperity of the nation (1). The cost of congestion in 2014 was calculated to be $160 billion for the top 471 urban areas in the United States. This included 6.9 billion hours of wasted time and 3.1 billion gallons of wasted fuel (2). Undoubtedly, dissemination of real-time traffic information to road users can significantly improve the efficiency of traffic networks. Hence, estimating real-time traffic states and thereby detecting network anomalies such as congestion and incidents, have been of significant interest to researchers for the last few decades.

Traditionally, traffic-state estimation is conducted using point-based sensors, including inductive loops, piezoelectric sensors, and magnetic loops (3). Recent advances in active infra-red/laser radar sensors have led to these devices gradually replacing the traditional point-based sensors (4). Also, with the increasing usage of navigation-based GPS devices, probe-based data are emerging as a cost-effective way to collect network-wide traffic data (5). Video monitoring and surveillance systems also are used for calculating real-time traffic data (6). Recent advances in image processing techniques have improved vision-based detection accuracy. Deep learning methods, such as convolution neural networks (CNNs), have been able to achieve human-level accuracy in image classification tasks (7). The basic advantage of these methods is that they don't require picking up hand-crafted features and hence can do away with the painstaking calibration tasks needed when using camera images for traffic-state estimation (8).

Studies have also been performed fusing multiple sources of data for traffic state estimation (9–11). Van Lint and Hoogendoorn used extended generalized Treiber-Helbing filter for fusing probe-based and sensor based data (9). Choi and Chung used fuzzy regression and Bayesian pooling technique for estimating link travel times from probe data and sensor data (10). Bachmann et al. investigated several multi-sensor data fusion based techniques to compare their ability to estimate freeway traffic speed (11). State DOTs also traditionally use sensor data and probe vehicle data for traffic state estimation. However, they have also installed a large number of roadside cameras on freeways and arterials for surveillance tasks such as incident detection. These cameras are used by traffic incident managers, who can zoom, tilt, and pan the cameras according to their need. Hence, the use of cameras for traffic-state estimation or congestion detection involves additional challenges due to frequent camera movement, which can alter the default calibrations. However, algorithms shouldn't rely on the exact placement of cameras and should be able to accurately detect traffic conditions for different placement scenarios.

In this study, we used camera images from different locations, orientations, and weather conditions to successfully detect traffic congestion. Three different models were used for congestion detection tasks. Two of these are deep neural networks: deep convolution neural networks (DCNNs) and you only look once (YOLO). Because these models require time-consuming and costly GPU training, the support vector machine (SVM), a shallow learning model, was used as a comparison to determine the advantages of using deep models.

The outline of this article is as follows: The present section provides a brief introduction and the importance of traffic congestion detection, the next section gives a review of previous work done on using cameras for traffic-state estimation, the third section gives an overview of the proposed models used for traffic congestion determination, the fourth section provides description of the data used in this study and the data preprocessing steps adopted for further analyses, the fifth section includes a discussion of the results obtained from the analyses, and the final section provides the conclusion and recommendations for future work.

LITERATURE REVIEW

During the last few decades, significant research efforts have been devoted to using CCTV cameras to determine real-time traffic parameters such as volume, density, and speed (12, 13). These methods can be broadly divided into three categories: (a) detection-based methods, (b) motion-based methods, and (c) holistic approaches.

Detection-based methods use individual video frames to identify and localize vehicles and thereby perform a counting task. Ozkurt and Camci used neural network methods to perform vehicle counting and classification tasks from video records (6). Kalman filter-based background estimation has also been used to estimate vehicle density (14). In addition, faster recurrent convolution neural networks (RCNNs) have been used for traffic density calculation (15); however, they were found to perform poorly for videos with low resolution and high occlusion. Recent achievements in deep learning methods in image recognition tasks have led to several such methods being used for traffic counting tasks. Adu-Gyamfi et al. used DCNNs for vehicle category classification (16). Oñoro-Rubio and López-Sastre used two variations of CNNs, namely counting CNN and hydra CNN, to conduct vehicle counting and predict traffic density (17). Recently, Zhang and Wu used both deep learning and optimization-based methods to perform vehicle counts from low frame-rate, high occlusion videos (12). They mapped the image to a vehicle density map using rank-constrained regression and full convolution networks.

Several motion-based methods have been suggested in the literature to estimate traffic flow utilizing vehicle tracking information. Asmaa et al. used microscopic parameters extracted using motion detection in a video sequence (18). They also analyzed the global motion in the video scene to extract the macroscopic parameters. However, these methods tend to fail due to lack of motion information and low frame rates of videos; some vehicles appear only once in a video, and hence, it becomes difficult to estimate their trajectories.

Holistic approaches avoid the segmentation of each object. Rather, an analysis is performed on the whole image to estimate the overall traffic state. Gonclaves et al. classified traffic videos into different congestion types using spatiotemporal Gabor filters (19). Lempitsky and Zisserman performed a linear transformation on each pixel feature to estimate the object density in an image (20); however, this approach was found to perform poorly in videos with a large perspective. Further, both these methods require manual annotation of each object in the images to perform the training of the counting task.

Overall, significant studies have been conducted in the past using various deep and shallow learning models to implement vehicle counting tasks and thereby determine congestion states. In this study, we adopted the holistic approach to label an image as either congested or non-congested. We also did away with counting each vehicle to determine the congestion state. Rather, we assigned labels to the images based on nearby benchmark sensors and then conducted the classification task. The next section provides detailed description of the methods used in our study.

EXPERIMENTAL APPROACH

Traffic congestion detection from camera images can be conducted in two broad ways. With the first approach, the input image can be fed into an object recognition model to determine the number of vehicles in the image and, when the number of vehicles exceeds a threshold, the image can be labeled as congested. With the second approach, the entire image can be classified as either congested or non-congested. In our study, we used the second approach, as it is much simpler and also doesn't require time-consuming manual annotation of individual vehicles.

We used three different algorithms for the traffic congestion detection task: two based on deep neural networks, which require time-consuming GPU training, and one from the shallow learning algorithm class, which doesn't require GPU training. The shallow algorithm was adopted primarily to determine the advantages, if any, for using GPU for this classification task. The three algorithms used in this study were:

1. Traditional Deep Convolution Neural Network (DCNN)
2. You Look Only Once (YOLO)
3. Support Vector Machine (SVM)

A detailed description of each of these algorithms is provided next.

Deep convolutional neural networks (DCCNs)

Collectively, DCNNs are a state-of-art technique for object detection and image classification. We used a traditional ConvNet architecture consisting of convolution and pooling layers. The convolution architecture used in this study is shown in Table 1. Because images from different cameras were used in this study, the input images were of different sizes, the majority being 800×450 pixels. The images were then resized to 400×225 pixels to prevent memory allocation issues during the training of the model. Next, these images were fed into the model as two consecutive convolution layers 32×3×3 in size followed by a max pooling layer 2×2 in size. This was followed by two additional convolution layers 64×3×3 in size and then again max pooling with a 2×2 filter. Each max pooling layer was followed by dropout with a probability of 0.25 to prevent overfitting. Finally, two fully connected layers (dense) were used, the first one with 512 neurons and the final one with two neurons corresponding to the binary classes (congested and non-congested). A batch size of 32 was used throughout the model and Leaky-ReLU was used as an activation function.

TABLE 1 DCNN model architecture used

Layer	Kernel	Stride	Output Shape
Input			[400, 225, 3]
Convolution	3×3	1	[400, 225, 32]
Convolution	3×3	1	[398, 223, 32]
Max Pooling	2×2	2	[199, 111, 32]
Dropout			[199, 111, 32]
Convolution	3×3	1	[199, 111, 64]
Convolution	3×3	1	[197, 109, 64]
Max Pooling	2×2	2	[98, 54, 64]
Dropout			[98, 54, 64]
Dense			512
Dropout			512
Dense			2

DCNN models are computationally expensive and usually require millions of images to train the model to prevent overfitting. However, in our study, we had only 2400 images, of which 1400 were used for training. So, to prevent overfitting, along with dropout, we also used data augmentation, similar to Ahmed et al. (21). Here, we randomly flipped images horizontally with a probability 0.5 and also performed horizontal and vertical shifts in the range of 10% of the total height and width. It took 25 minutes to train the model on a NVIDIA Tesla K20m GPU with 4 GB RAM memory. Keras (22), a deep learning library, was used to run the script in GPU.

YOLO (You Only Look Once)

We adopted the YOLO model (23) for general purpose congestion detection and localization from CCTV video feeds. Current object detection systems repurpose powerful CNN classifiers to perform detection. For example, to detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in the test image. YOLO reframes object detection; instead of looking at a single image 1000 times to accomplish detection, it looks at an image only once (but in a clever way) to perform the full detection pipeline. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. This makes YOLO extremely fast and easy to generalize to difference scenes. YOLO is also a DCNN classifier, however in the rest of the analyses, we will denote it as YOLO and the traditional DCNN explained before as DCNN.

YOLO uses a simple CNN architecture shown in Table 2. This neural network uses only standard layer types: convolution with a 3×3 kernel and max pooling with a 2×2 kernel. The very last convolutional layer has a 1×1 kernel, which serves to reduce the data to the shape 13×13×125. This 13×13 shape is the size of the grid into which the image gets divided. There are 35 channels for every grid cell. These 35 numbers represent the data for the bounding boxes and the class predictions, as each grid cell predicts five bounding boxes and a bounding box is described by seven data elements:

- x, y, width, and height for the bounding box's rectangle;
- the confidence score; and
- the probability distribution over the two classes (congested and non-congested)

The key implementation steps for YOLO are as follows:

1. Resize the input image to 416×416 pixels.
2. Pass the image through a CNN in a single pass. The architecture of the CNN is described in the following section.
3. The CNN outputs a 13×13×k tensor describing the bounding boxes for the grid cells. The value of k is related to the number of classes as follows: k = (number of classes + 5)*5.
4. Compute the confidence scores for all bounding boxes and reject all boxes that fall below a predefined threshold.

Because there are 13×13 = 169 grid cells and each cell predicts five bounding boxes, there are 845 bounding boxes in total. Ideally, the majority of these boxes would have very low confidence scores. In this study, a confidence threshold of 45% was used for congestion detection.

TABLE 2 YOLO model architecture used

Layer	Kernel	Stride	Output Shape
Input			[416, 416, 3]
Convolution	3×3	1	[416, 416, 16]
Max Pooling	2×2	2	[208, 208, 16]
Convolution	3×3	1	[208, 208, 32]
Max Pooling	2×2	2	[104, 104, 32]
Convolution	3×3	1	[104, 104, 64]
Max Pooling	2×2	2	[52, 52, 64]
Convolution	3×3	1	[52, 52, 128]
Max Pooling	2×2	2	[26, 26, 128]
Convolution	3×3	1	[26, 26, 256]
Max Pooling	2×2	2	[13, 13, 256]
Convolution	3×3	1	[13, 13, 512]
Max Pooling	2×2	1	[13, 13, 512]
Convolution	3×3	1	[13, 13, 1024]
Convolution	3×3	1	[13, 13, 1024]
Convolution	1×1	1	[13, 13, 35]

Support Vector Machine (SVM)

A SVM is one of the most widely used shallow algorithms for image classification task. It solves a constrained quadratic optimization problem to classify data into different categories. The resulting optimal hyperplane is determined by maximizing the largest minimum distance to training examples to make it least sensitive to noise. We utilized the Oriented FAST and Rotated BRIEF (ORB) (24) feature detector to detect the key points in each image, whereby the FAST (features from accelerated segment test) algorithm was used to extract the key points and the Harris corner distance was used to determine the top N points. The algorithm was run on the training data set with 10-fold cross-validation to determine the optimal penalty parameter and kernel. This algorithm was run on Windows 7 with Intel Core i7-4790 CPU with 8GB of RAM.

DATA DESCRIPTION

Two different data sources were used in this study: camera images and radar-based Wavetronix sensors. Camera images were obtained from 121 cameras from the Iowa DOT CCTV camera database spread across the interstates and highways of Iowa. The database covered the major cities of Iowa: e.g., Des Moines, Sioux City, Cedar Rapids, Council Bluffs, Davenport, and Iowa City. Images were extracted from the cameras at 5-minute intervals from October 2016 to March 2017, resulting in a total of 3.5 million images during the study period. The task of assigning a label to an image (congested or non-congested) consisted of four sub-tasks:

1. Associating each camera with a nearby wavetronix sensor pair,
2. Smoothening the wavetronix data,
3. Extracting the details (camera name, timestamp) of each image, and
4. Assigning the label of the image based on sensor data.

Details of each of these tasks are discussed next.

Each camera was first associated with the two nearest Wavetronix sensor pairs covering both directions of the freeway on which camera was placed. If the sensor pair was located more than 0.5 miles away from the camera, then the particular camera was removed from analysis. Here, the assumption was made that if sensors are located more than 0.5 miles away from the camera, then the observation made from the camera might not match up with the sensor observations.

The next step was to assign the traffic data from the sensor to each image. However, sensor data obtained from Wavetronix in 20-second intervals included too much noise; so, we used Wavelet smoothing to remove the noise. In this study, among the several families of wavelets that could be used, such as Haar, Daubechies, Biorthogonal, Symlets, Coiflets, Morlet, Mexican Hat, Meyer, etc., we used Daubechies extremal phase wavelets. Daubechies family wavelets are also known "dbN," where N refers to the number of vanishing moments. The higher the value of N, the longer the wavelet filter and the smoother the wavelet. Based on our data, we used db2 with level 6 to achieve a smooth curve-like filter that followed most of the variations of the original signal. A sample of the original and smoothed data is shown in Figure 1.

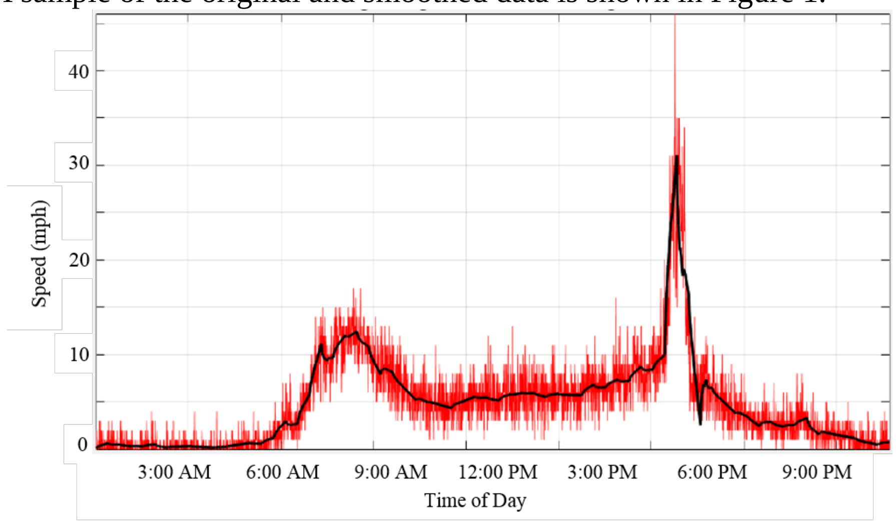


Figure 1 Original and smoothed occupancy using Wavelet transform (db2 level 6)

The next step was to extract the details of each image. The top of each image showed the details of the image (direction, camera name, and timestamp). Optical character recognition (OCR) was used to extract the details from each image, which were then matched with the corresponding sensor data based on the camera's name and timestamp.

After obtaining the smoothed Wavetronix data, timestamp, and camera name for each image, we assigned the traffic data obtained from the sensor to the image. The traffic data comprised speed, volume, and occupancy observed at 20-second intervals. To assign the congested or non-congested label to the image, we used occupancy values, which are denoted by the percentage of the time the sensor is occupied by vehicles and have one-to-one mapping to traffic density or the number of vehicles in the unit distance. Persaud and Hall suggested that occupancy of 20% or more should be considered congested, whereas occupancy below that should be considered non-congested (25). Thus, if no congestion (occupancy <20%) was observed in either direction of the wavetronix pair, then the image was classified as "non-congested"; if congestion was visible in any particular direction or in both directions, then it was labeled as "congested." We adopted this approach to do away with manual labeling of congested and non-congested images and to follow a uniform methodology for assigning labels to the images.

Finally, we obtained 1218 congested images and more than 3 million non-congested images. Due to class imbalance, we randomly chose 1200 non-congested images out of the 3 million images. This dataset consisting of a total 2418 images was then subdivided into a training set and a test set. The training set consisted of 1400 images with equal proportions of congested and non-congested images. However, as will be discussed later, the YOLO approach of congestion detection requires manually annotating the region of congestion. For this purpose, 100 congested images were extracted from the training set and manually annotated with the congested region. The test set consisted of 1018 images out of which 518 were congested and the rest were uncongested. Because sensor errors can occasionally cause misclassification of images, test set images were manually cross-checked and then the final labels were assigned. However, no manual cross-checking of labels was performed for the training set, as it was assumed that the algorithm itself should be able to determine the misclassifications, if any, in the training set.

RESULTS

The performance of each of the three algorithms were trained on 1400 images and tested on 1018 test set images (518 congested and 500 non-congested). YOLO was trained and tested on NVIDIA GTX 1080 Ti 8 GB RAM GPU while for DCNN, NVIDIA Tesla K20m GPU with 4 GB RAM was used. Intel Core i7-4790 with 8 GB RAM CPU was used for training and testing of SVM. The training times for YOLO, DCNN and SVM were 22 hours, 26 minutes, and 50.4 seconds respectively. The testing times for the 3 algorithms were 0.01, 0.01, and 0.03 seconds/frame respectively. The testing time does not include the time required for the model; rather, it includes only the time required to predict the class for each image. Because YOLO and DCNN are deep models, they had to be trained and tested using GPUs and which involved time-consuming and costly training compared to its shallow counterpart, SVM. The testing times for DCNN and YOLO were lower, but they required GPU during testing time as well.

The performance of the algorithms was evaluated using the standard performance metrics of precision, recall, and accuracy (Equations 1, 2 and 3, respectively). When a congested image was correctly labeled (i.e., the predicted label was also “congested”), it was classified as true positive (TP). Similarly, if a non-congested image was correctly labeled as “non-congested,” then it was classified as true negative (TN). However, if the actual label was “congested” and the predicted label was “non-congested,” it was classified as false negative (FN). And finally, if the actual label was “non-congested” and the predicted label was “congested,” it was classified as false positive (FP).

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$Precision = \frac{TP + TN}{TP + FP + TN + FN} \quad (3)$$

Some examples of true classifications and misclassifications obtained from each algorithm (YOLO, DCNN, and SVM) are shown in Figure 2. Examples of true positives, for which each algorithm correctly labeled congested images, are shown in Figures 2a, 2b, and 2c (YOLO also gives the bounding box for the congested region). On the other hand, examples of false positives, for which the algorithms misclassified non-congested images as congested, are shown in Figures 2d, 2e, and 2f. It can be seen that YOLO misclassified an image as a congested region because of a group of vehicles located far away from the camera during nighttime (Figure 2d) and vehicles on a bridge led to misclassification by DCNN (Figure 2e). SVM, on the other hand, had misclassifications in adverse weather conditions (Figure 2f) because snow particles were detected as corners, which caused the image to be labeled as congested. Examples of false negatives, for which the algorithms failed to detect congested images correctly, are shown in Figures 2g, 2h and 2i. Congestion quite distant from the camera led to misclassification by YOLO (Figure 2g), whereas DCNN failed to detect congestion in a single lane when the other lane was closed and hence empty (Figure 2h). Glare issues resulted in SVM misclassifications (Figure 2i). Finally, examples of true negatives, for which the algorithms correctly labeled non-congested images, are shown in Figures 2j, 2k, and 2l.

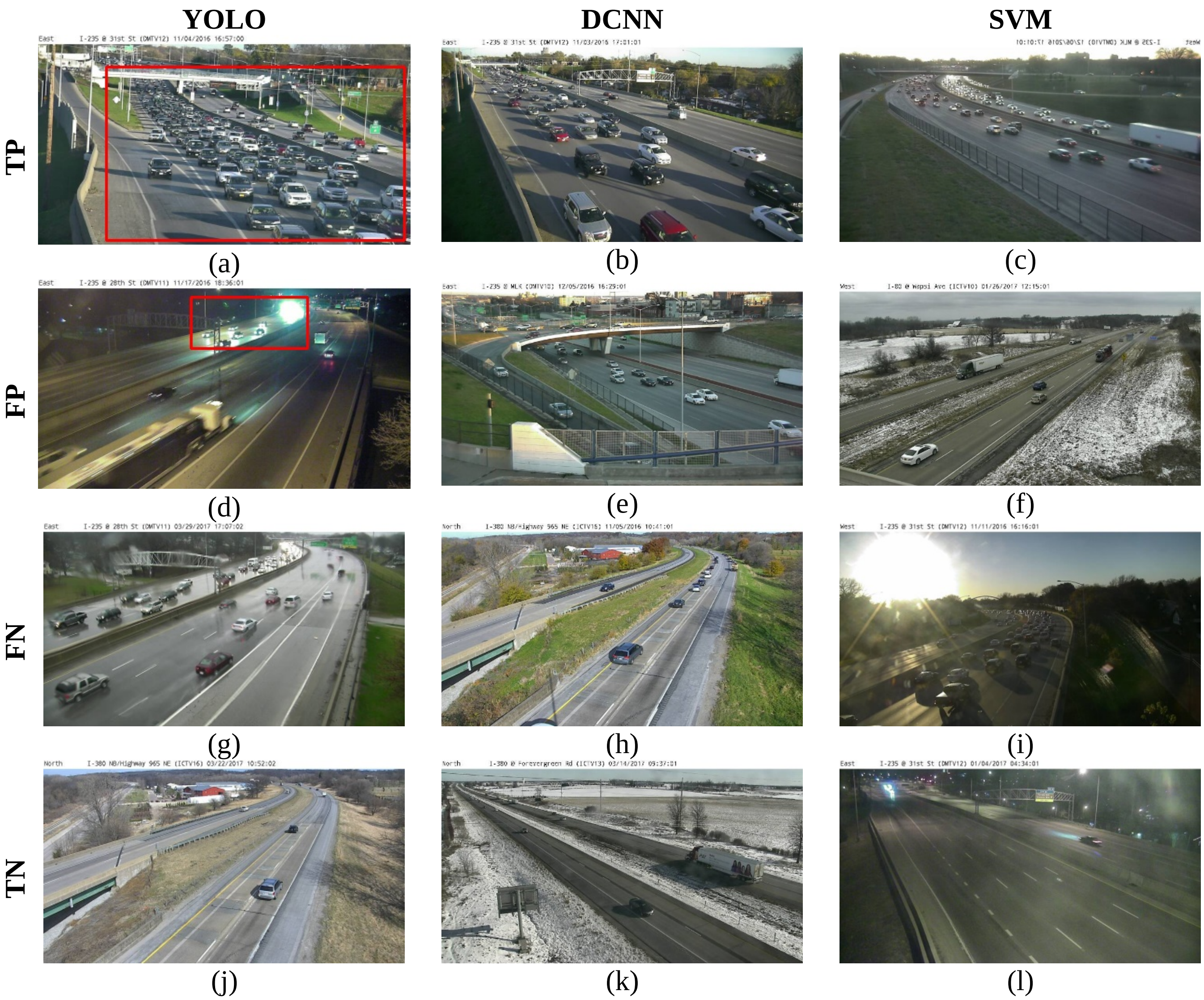


Figure 2 Congestion detection classifcation examples: (a-c) true positives, (d-f) false positives, (g-h) false negatives, (j-l): true negatives

The precision, recall, and accuracy values obtained from each algorithm are shown in Table 3. YOLO achieved the highest precision, recall, and accuracy followed closely by DCNN.

Because YOLO achieved better accuracy compared to DCNN, we didn't performed region-based CNN separately to determine the congested region of the results obtained from DCNN. YOLO, on the other hand, being a region-based classifier gives the congested region of the image by default (see Figures 2a and 2d). The accuracy obtained by SVM was comparatively lower (85.2%) than expected given the lower computation costs involved in such a shallow algorithm. In this context, it should be mentioned that a separate analysis, reported in a separate paper (24), using an ensemble of shallow learning algorithms (SVM with ORB, Shi-Tomasi, and Structured Edge Toolbox feature detector) gave an accuracy of 86.7% with the same dataset. Here, we used only SVM with ORB for comparison of deep learning models with a standard shallow model.

Table 3 Precision, recall and accuracy values obtained from the three algorithms

Method	Precision (%)	Recall (%)	Accuracy (%)
YOLO	88.6	94.3	91.4
DCNN	86.9	93.9	90.2
SVM	82.8	88.5	85.7

Sensitivity Analysis

Sensitivity analysis was also performed to determine which factors might affect the performance of the congestion detection system developed here. We evaluated two factors that could influence the classification task: first, the time of day the image was captured (daytime versus nighttime) and, second, camera resolution (blurring, rain, snow, and glare). The test database was then divided into four subgroups according to the combination of the two factors, as follows:

1. D-G: daytime, good resolution (436 images);
2. N-G: nighttime, good resolution (147 images);
3. D-P: daytime, poor resolution (190 images); and
4. N-P: nighttime, poor resolution (245 images).

Receiver operating characteristics (ROC) curves were then used to compare the performance of each algorithm for each subgroup based on the true positive and false positive rates (TPR and FPR, respectively), as defined in Equations (4) and (5), respectively:

$$TPR = \frac{TP}{TP + FN} \tag{4}$$

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

For an efficient image classification model, the TPR should be higher than the corresponding FPR. On the other hand, for a poor-vision system model, when the sensitivity (TPR) increases, it loses the ability to discriminate between congested and non-congested images, which makes the TPR directly proportional to the FPR. The ROC curves for each subgroup obtained from the three models—YOLO, DCNN, and SVM—are shown in Figures 3a, 3b, and 3c, respectively. The overall ROC curve for the three algorithms is shown in Figure 3d. Also, the area under each curve (AUC) is provided for each case. For all three algorithms, TPRs were higher than the corresponding FPRs irrespective of the prevailing conditions (daytime or nighttime, poor or good resolution). All of the algorithms performed well during the daytime, irrespective of the camera resolution. However, AUCs were found to be lowest for poor resolution images at night (N-P). Moreover, irrespective of the conditions, the AUCs from all algorithms for each subgroup were found to be mostly higher than 0.90, except for N-P conditions from SVM. This shows that the system works well even under challenging conditions.

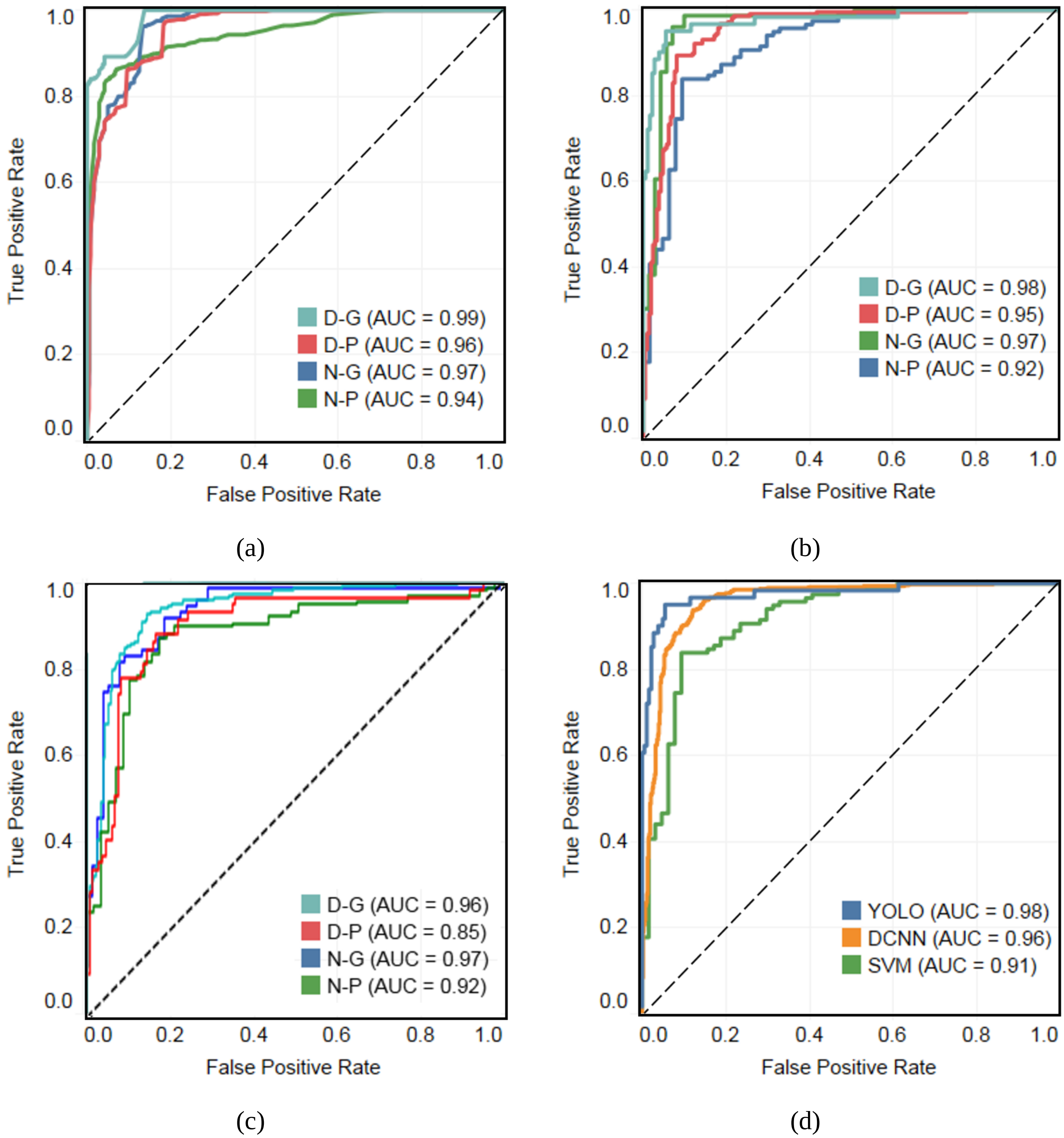


Figure 3 ROC curves under different prevalent conditions obtained from a) YOLO b) DCNN c) SVM and (d) all conditions combined for each algorithm

In addition, ROC curves can be used by traffic management centers (TMCs) to choose an optimal threshold between TPR and FPR. Previous studies have shown that too many false calls is a major reason for limited integration of automatic incident detection algorithms in the TMC. Hence, it is important for TMC personnel to know the accuracy that be achieved given a particular FPR. For example, if a TMC wants to restrict the FPR to lower than 0.1, then the TPR obtained by YOLO, DCNN, and SVM will be 0.92, 0.96, and 0.82, respectively, during good daytime (D-G) conditions. Obviously, the accuracy would be lower with poor camera conditions at night. Hence, TMC personnel can use the ROC curves to set the optimal threshold of TPR and FPR based on their specific needs.

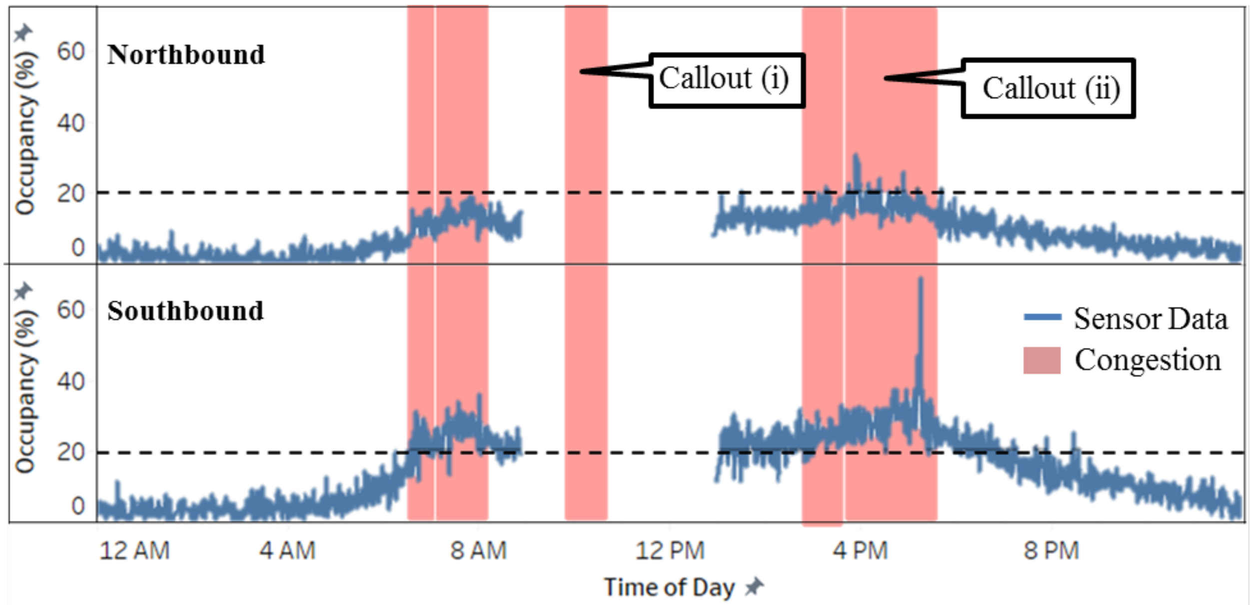
Real-time Implementation

The congestion detection algorithms can also be implemented online easily. With a test time of 0.01 seconds per image, the algorithms can be adopted to detect traffic congestion of approximately 1000 cameras in every 10 seconds interval using a single GPU. Figure 4 shows an example of congestion detection by DCNN algorithm on images extracted from a camera on a single day (27th October, 2017) at every 10 seconds interval. The congestion alarm occurrences from camera is shown in the background of Figure 4 along with the occupancy data obtained from nearest radar sensors (both directions). However, due to sensor issues, sensor data were missing from 8:51 AM to 12:57 PM. So, a 2-vehicle crash reported at around 10:30 AM was missed by sensor, but was detected successfully by the camera. Thus, this example also shows that using multiple data sources (cameras, sensors, etc.) can increase the reliability in traffic state estimation. Callouts (i) and (ii) are also provided in Figure 4 to show samples of camera images when congestion alarms were triggered. To eliminate false alarms, alarms are triggered only when congestion is detected on 3 consecutive 10-seconds interval frames (persistency test). Also, alarms triggered within 5 minutes interval are combined together to form a single continuous alarm. These “signal smoothing” techniques help in decreasing false alarm rates (FAR) and increasing detection rates (DR). Future studies can be done implementing better smoothing techniques like Fourier Transforms or Wavelet smoothing and determining the DR and FAR on a network of cameras.

CONCLUSIONS

Recent advancements in machine-vision algorithms and high performance computing have improved image classification accuracy to a great extent. In this study, two such deep learning techniques, the traditional DCNN and YOLO models, were used to detect traffic congestion from camera images. SVM also was used for comparison and to determine what improvements were obtained while using costly GPU techniques. To eliminate the time-consuming task of manual labeling and to maintain uniformity in congestion labeling, we used nearby Wavetronix sensors to correctly identify congested images. For testing purposes, we also labeled each image manually to remove misclassifications due to sensor errors.

The YOLO model achieved the highest accuracy of 91.2% followed by DCNN with an accuracy of 90.2%; 85% of images were correctly classified by SVM. Congestion regions located far away from the camera, single-lane blockages, and glare issues were found to affect the accuracy of the models. To determine the sensitivity of the models to different camera configurations and light conditions, ROC curves were used. All the algorithms were found to perform well in daytime conditions, but night conditions were found to affect the accuracy of the vision system. However, for all conditions, the AUCs were found to be greater than 0.9 for the deep models. This shows that the models perform well in challenging conditions as well.



(a)

(b) (c)

Figure 4 (a) Sensor occupancy data and congestion alarm from a camera on a particular date; (b-c) Camera images of Callouts (i-ii) shown in Part a

An example of the real-time implementation of congestion detection using DCNN algorithm is also performed using a continuous set of images extracted from a camera. Simple persistence test methods were applied to reduce the false alarms and smoothen the output signal. Future studies can look into different smoothing techniques (Fourier Transform, Wavelets) to denoise the output obtained from the algorithm and determine the overall detection rate and false alarm rates on a network of cameras. Future studies can also be done using different model architectural designs to improve detection accuracies. Such models can also be used for determining different levels of congestion (high, medium, or low) and also more accurate traffic state determination (speed, volume and occupancy). Congestion status obtained from the cameras can also be stored as historical data and used to determine traffic anomalies such as incidents.

ACKNOWLEDGMENT

Our research results are based upon work jointly supported by the National Science Foundation Partnerships for Innovation: Building Innovation Capacity (PFI: BIC) program under Grant No. 1632116, National Science Foundation under Grant No. CNS-1464279, and Iowa DOT Office of Traffic Operations Support Grant. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

1. Owens, N., Armstrong, A., Sullivan, P., Mitchell, C., Newton, D., Brewster, R., et al. Traffic incident management handbook. *Public Roads*, Vol. 172, 2010, pp. 116.
2. Schrank., D., Eisele., B., Lomax., T. and Bak., J. 2015 Urban Mobility Scorecard. 2015. http://d2dtl5nnlpfr0r.cloudfront.net/tti.tamu.edu/documents/mobility-scorecard-2015.pdf.
3. Kotzenmacher, B.J., Minge, E.D. and Hao, B. Evaluation of Portable Non-Intrusive Traffic Detection System. 2004.
4. Zhong, M. and Liu, G. Establishing and Managing Jurisdiction-wide Traffic Monitoring Systems: North American Experiences. *Journal of Transportation Systems Engineering and Information Technology*, Vol. 7, No. 6, 2007, pp. 25–38.
5. Feng, Y., Hourdos, J. and Davis, G.A. Probe vehicle based real-time traffic monitoring on urban roadways. *Transportation Research Part C: Emerging Technologies*, Vol. 40, 2014, pp. 160–178.
6. Ozkurt, C. and Camci, F. Automatic traffic density estimation and vehicle classification for traffic surveillance systems using neural networks. *Mathematical and Computational Applications*, Vol. 14, No. 3, 2009, pp. 187–196.
7. He, K., Zhang, X., Ren, S. and Sun, J. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385v1*, Vol. 7, No. 3, 2015, pp. 171–180. http://arxiv.org/pdf/1512.03385v1.pdf.
8. Bauza, R., Gozalvez, J. and Sanchez-Soriano, J. Road traffic congestion detection through cooperative Vehicle-to-Vehicle communications. In Proceedings - Conference on Local Computer Networks, LCN. pp. 606–6122010.
9. Van Lint, J.W.C. and Hoogendoorn, S.P. A Robust and Efficient Method for Fusing Heterogeneous Data from Traffic Sensors on Freeways. *Computer-Aided Civil and Infrastructure Engineering*, Vol. 25, No. 8, 2010, pp. 596–612.
10. Choi, K. and Chung, Y. A Data Fusion Algorithm for Estimating Link Travel Time. *Journal of Intelligent Transportation Systems*, Vol. 7, No. 3–4, 2010, pp. 235–260. http://www-tandfonline-com.ezproxy.ucn.cl/doi/abs/10.1080/714040818#aHR0cDovL3d3dy10YW5kZm9ubGluZS1jb20uZXpwcm94eS51Y24uY2wvZG9pL3BkZi8xMC4xMDgwLzcxNDA0MDgxOEBAQDA=.
11. Bachmann, C., Abdulhai, B., Roorda, M.J. and Moshiri, B. A comparative assessment of multi-sensor data fusion techniques for freeway traffic speed estimation using microsimulation modeling. *Transportation Research Part C: Emerging Technologies*, Vol. 26, 2013, pp. 33–48.
12. Zhang, S. and Wu, G. Understanding Traffic Density from Large-Scale Web Camera Data. 2015.
13. Darwish, T. and Abu Bakar, K. Traffic density estimation in vehicular ad hoc networks: A

review. *Ad Hoc Networks*, Vol. 24, No. PA, 2015, pp. 337–351. http://dx.doi.org/10.1016/j.adhoc.2014.09.007.

14. Balcilar, M. and Sönmez, A.C. Extracting vehicle density from background estimation using Kalman filter. In 2008 23rd International Symposium on Computer and Information Sciences, ISCIS 2008. 2008.
15. Ren, S., He, K., Girshick, R. and Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 6, 2017, pp. 1137–1149.
16. Adu-Gyamfi, Y., Asare, S., Sharma, A. and Titus, T. Automated Vehicle Recognition with Deep Convolutional Neural Networks. *Transportation Research Record Journal of the Transportation Research Board*, 2017: 10.3141/2645-13.
17. Oñoro-Rubio, D. and López-Sastre, R.J. Towards perspective-free object counting with deep learning. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Volume 9911 LNCS. pp. 615–6292016.
18. Asmaa, O., Mokhtar, K. and Abdelaziz, O. Road traffic density estimation using microscopic and macroscopic parameters. *Image and Vision Computing*, Vol. 31, No. 11, 2013, pp. 887–894.
19. Gonçalves, W.N., Machado, B.B. and Bruno, O.M. Spatiotemporal Gabor filters: a new method for dynamic texture recognition. *arXiv preprint arXiv*, 2012. http://arxiv.org/abs/1201.3612.
20. Lempitsky, V. and Zisserman, A. Learning To Count Objects in Images. *Advances in Neural Information Processing Systems*, 2010. http://papers.nips.cc/paper/4043-learning-to-count-objects-in-images.pdf.
21. Ahmed, E., Jones, M. and Marks, T.K. An Improved Deep Learning Architecture for Person Re-Identification. *Computer Vision and Pattern Recognition (CVPR)*, 2015: 10.1109/CVPR.2015.7299016.
22. Chollet, F. Keras: Deep Learning library for Theano and TensorFlow. *GitHub Repository*, 2015.
23. Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *Cvpr 2016*, 2016: 10.1016/j.nima.2015.05.028.
24. Rublee, E., Rabaud, V., Konolige, K. and Bradski, G. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the IEEE International Conference on Computer Vision. pp. 2564–25712011.
25. Persaud, B.N. and Hall, F.L. Catastrophe theory and patterns in 30-second freeway traffic data- Implications for incident detection. *Transportation Research Part A: General*, Vol. 23, No. 2, 1989, pp. 103–113.