IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 2, 2021, accepted March 3, 2021, date of publication March 9, 2021, date of current version March 18, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3064603*

# Multi-Regime Analysis for Computer Vision-Based Traffic Surveillance Using a Change-Point Detection Algorithm

**SEUNGYUN JEONG, SEUNGBIN ROH, AND KEEMIN SOHN**
Department of Urban Engineering, Chung-Ang University, Seoul 06974, South Korea

Corresponding author: Keemin Sohn (kmsohn@cau.ac.kr)

This work was supported in part by the Chung-Ang University Research Scholarship Grants in 2020, in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government under Grant 2021R1A2C2003842, and in part by the Korea Agency for Infrastructure Technology Advancement (KAIA) Grant funded by the Ministry of Land, Infrastructure and Transport, under Grant 20TLRP-B148676-03.

**ABSTRACT** As a result of significant advances in deep learning, computer vision technology has been widely adopted in the field of traffic surveillance. Nonetheless, it is difficult to find a universal model that can measure traffic parameters irrespective of ambient conditions such as times of the day, weather, or shadows. These conditions vary recurrently, but the exact points of change are inconsistent and unpredictable. Thus, the application of a multi-regime method would be problematic, even when separate sets of model parameters are prepared in advance. In the present study we devised a robust approach that facilitates multi-regime analysis. This approach employs an online parametric algorithm to determine the change-points for ambient conditions. An autoencoder was used to reduce the dimensions of input images, and reduced feature vectors were used to implement the online change-point algorithm. Seven separate periods were tagged with typical times in a given day. Multi-regime analysis was then performed so that the traffic density could be separately measured for each period. To train and test models for vehicle counting, 1,100 video images were randomly chosen for each period and labeled with traffic counts. The measurement accuracy of multi-regime analysis was much higher than that of an integrated model trained on all data.

**INDEX TERMS** Change-point algorithm, autoencoder, traffic surveillance, multi-regime model, traffic density.

## I. INTRODUCTION

Computer vision technology has been widely adopted to measure traffic parameters such as traffic volumes, densities and speeds [1]–[5]. This trend was accelerated when deep-learning models could recognize objects within an image similar to the way a human does [6]–[14]. In particular, vehicle detection is an easier task than recognizing other objects, because a vehicle takes a simple shape and the road environment for traffic surveillance is not complex. Many researchers are trying to devise a robust vehicle detector for the use of traffic surveillance based on deep learning [15], [16]. For such a detector to be installed in the field for traffic surveillance, however, some limitations must be resolved. The current level of detection accuracy varies according to ambient conditions such as the time of day, weather and shadows. It remains difficult to develop a single vehicle detector with consistent performance both at daytime and nighttime. To recognize vehicles at nighttime, some researchers have utilized headlight beams, which is a feature that is unique to vehicles in nighttime [17]. Switching detection models, however, cannot be automatically implemented on a real-time basis.

The present study provides a robust way to determine the switching times based on consecutive video frames under the assumption that the video frames fully reflect ambient conditions. Although it is easily understood that video images include hidden features of ambient conditions such as the time of day, how to extract them and to reduce their dimensions into a tractable level is a different story. We decided to use an autoencoder to elicit ambient conditions from video images. An autoencoder can extract factors based on nonlinear correlations between variables. The hidden features that

The associate editor coordinating the review of this manuscript and approving it for publication was Yun Zhang.

an autoencoder extracts are then used as input for an algorithm to determine the change-points. A novel algorithm for finding change-points was adopted in the present study [18]. This algorithm depends on a rigorous statistical test unlike other rule-based naïve methods. This algorithm also offers the advantage of field application, because it recognizes the changes in ambient conditions on a real-time basis. In addition, the algorithm has been successfully adopted to find the change-points for signals from many engineering and learning tasks [19]–[21].

We set up three different vehicle counting models. First, a deep-learning-based regression model for vehicle counting to validate the multi-regime analysis. This method is more convenient to prepare labeled images for training, because it does not require the drawing of a bounding box for each vehicle; rather, it counts the number of vehicles in an image. Second, there have been many attempts to measure traffic parameters such as volume, speed, and density using ''state-of-the-art'' object detectors [22]–[24]. It is appropriate to adopt these object detectors to validate the proposed multi-regime approach to measuring traffic density. Two ''state-of-the-art'' object detectors were employed to validate the proposed multi-regime measurement. A YOLO v3 [11], which is a representative anchor-based model, was adopted to measure the traffic density. A CenterNet [25], which is a non-anchor-based object detector, was also adopted since it recorded the best accuracy in vehicle counting. Both models were pretrained on an open dataset (UA-DETRAC) that includes only vehicle images with labels [26].

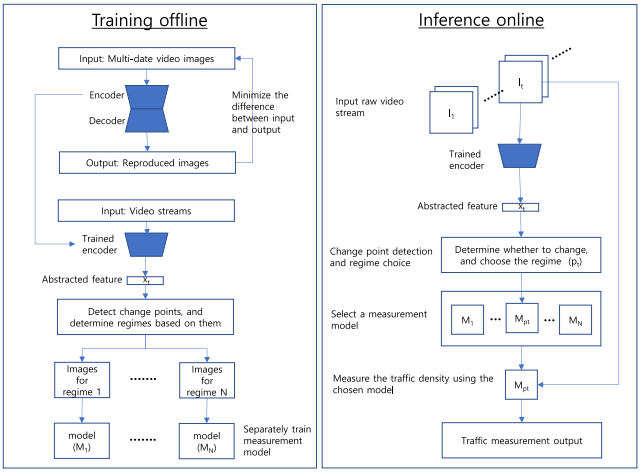



**FIGURE 1.** Modeling framework.

As a result of offline training of the proposed approach, a day was divided into 7 different periods, each of which had unique ambient conditions, and a set of model parameters was separately trained for each period in advance [see Fig. 1]. For the regression-based vehicle counting model, the same amount of image data (1,100 images) was sampled for each period and used for training and testing tasks. For the object detector-based vehicle counting, only 200 images were labeled with bounding boxes for training and testing. It should be noted that such a small number of labeled images cannot lead to an acceptable level of accuracy, but it can verify that a separate training is better than an integrated one. All these models were, therefore, trained and tested on the entire dataset for all periods under the expectation that the multi-regime analysis would outperform the integrated analysis.

The main contribution of this research is that we proposed a robust multi-regime approach for traffic surveillance that combines three existing technologies: 1) an autoencoder to abstract features, 2) a change-point detection algorithm, and 3) a convolutional neural network (CNN) model for measuring traffic density. We verified that the proposed multi-regime approach improves measurement accuracy compared with the use of a conventional overall model when the same training data are used. Another advantage of the multi-regime approach is that the measurement model can be downsized. For example, a model would require a large number of parameters to function well during both daytime and nighttime. Thus, more time is required for the model to learn from the data due to the increased number of parameters. However, if a model accommodates a short period, the number of parameters could dwindle.

The next section establishes an overall modeling framework that includes the introduction of three technologies: 1) an autoencoder to abstract images, 2) a change-point detection algorithm to separate multiple regimes, and 3) traffic density measurement models. The third section describes how to choose testbeds and shows how the ground truth change-points were determined for them. The fourth section lists the results of implementing a change-point detection algorithm when several different input dimensions are processed by an autoencoder. The section also verifies the validity of the multi-regime approach to count vehicles for traffic surveillance. The last section draws conclusions and suggests further studies.

## II. MODELING FRAMEWORK

The modeling framework combines three different models to enhance traffic surveillance performance: an autoencoder to abstract video images, a change point detection algorithm to set up multi-regimes of ambient conditions, and a measurement model for traffic density. Fig. 1 summarizes both how to train the models offline and how to implement them for inference online.

### A. THE AUTOENCODER REDUCES THE DIMENSIONS OF IMAGES

A video image is represented by three values of RGB for all pixels that comprise it. If these values are used as an input variable to find change-points, the computation time increases exorbitantly. The most popular method to reduce the variable dimension is to use a PCA [27], [28], but this algorithm has a handicap whereby only a linear relationship between variables can be accommodated when

deriving factors with a reduced dimension. On the other hand, an autoencoder can consider non-linear correlations between original input features while reducing their dimension. For this reason, autoencoders have been widely utilized in machine learning for dimension reduction [29], [30]. The present study devised an autoencoder to change a 128 × 512 × 3 pixel image into only 1 to 10 features. Fig. 2 shows the model architecture of the autoencoder established in the present study. The encoding part is composed of 6 hidden layers. After flattening the last convolutional layers, 5 fully connected (FC) layers are attached to reduce the feature dimension. The number of nodes in the last FC layer of the encoding part is decreased to six target dimensions (1 to 5 and 10). The last layer of the encoding part is amplified again to reproduce the original input dimension through the decoding part. part.

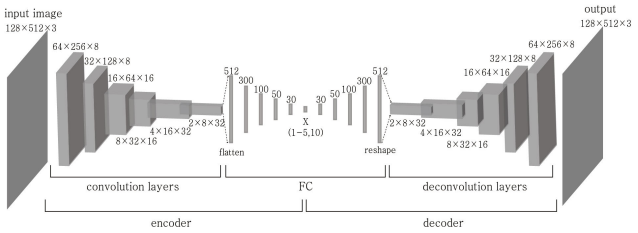



**FIGURE 2. Architecture of the proposed autoencoder used to reduce a feature dimension.**

Training an autoencoder requires no supervision with labeled data. The output image that an autoencoder evaluates from an input image is refitted to the input image. Such a self-fitting algorithm guarantees obtainment of the reduced features in the last layer of the encoding part, and only node values of the middle layer of an autoencoder are used as variables for the subsequent change-point detection algorithm once all parameters are completely trained. A change-point algorithm that used the middle node values elicited from a trained autoencoder was used to determine variations in the ambient conditions.

The video image of a testbed includes both ambient and traffic conditions. We adopted a two-step process to train models to count vehicles in the testbed. In the first step, only the features for ambient conditions are extracted from images, and the times of day are divided based on these. In the second step, a model is set up to count vehicles in an image, and model parameters are separately trained and tested for each period. The features derived in the first step, however, should include both ambient and traffic conditions. To suppress the features of traffic conditions other than ambient conditions, we tried several different numbers (1 to 5, and 10) of final features (= numbers of middle nodes in the proposed autoencoder) and examined how each option performed in the counting of vehicles.

### B. CHANGE-POINT DETECTION ALGORITHM

Change-point detection algorithms are widely used in science and engineering fields. Applications include signal processing, dynamic social networks, and online marketing. In similar manner, various change-point detection algorithms have been developed such as the segment neighborhood model [31], piecewise IID data models [32], the structural change model [33], PELT [32], and the Group-Fused Lasso (GFL) [34]. Whereas a univariate model prevails in the field [32]–[34], few models can successfully accommodate multivariate problems [35]–[37]. Other researchers recently developed a robust multivariate algorithm under the assumption that target data follows a parametric probability density [18]. This assumption makes it possible to use a statistical test to judge the validity of a change-point. The present study applied the algorithm to separating different regimes of ambient conditions in video images for traffic surveillance.

Once an image is represented in a smaller dimension after being processed by an autoencoder, the abstracted features constitute multivariate time-series data. First, the data are regarded as random variables with a Gaussian probabilistic density. Parameters of the Gaussian distribution include the means, variances and covariances. The number of parameters to be estimated varies according to the feature dimension. Second, the abstracted features can be assumed to follow a Dirichlet probabilistic distribution. To use the Dirichlet distribution, the features need to be compositional data that take a proportional value and should be summed to 1. Such data encompasses the percentage contribution of goods sold every month, the proportion of time spent by each task that comprises a whole production process in a given time period, and probabilities from a discrete distribution, etc. Data need to be preprocessed so that the Dirichlet probabilistic distribution can be applied. Raw data must be placed on a simplex in a hyperspace, and to do so the raw signal data of each interval should be changed to a compositional form via the use of a multi-dimensional Expit function (inverse of multinomial logit function), the details of which are described later in this section.

The theory behind the change-point detection algorithm in reference [18] is described as follows. The symbols $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$ denote the sequence of feature variables where $\boldsymbol{x}_i \in R^d$ is a feature vector at interval $i \in \{1, \ldots, T\}$, $T$ denotes the total number of time periods, and $d$ is the dimension of the feature vector that corresponds to the size of the middle layer of an autoencoder in the present study. The change-point detection algorithm determines $\{\tau_1, \ldots, \tau_{k-1}\}$, where $k$ is the number of regimes to be differentiated and $\tau_i$ is the change-point that separates the $(i+1)^{th}$ period from the $i^{th}$ period ($i \in \{1, \ldots, k-1\}$). These multiple change-points are detected by repeatedly implementing a single change-point detection algorithm. Once an active window size (I) is determined, the portion of the sequence data that is covered by the window can then be investigated from the start point. The initial search period then becomes [1, I]. If a change-point is identified within this time window, it becomes the first change-point, $\tau_1$, and the next time window is moved to $[\tau_1, \tau_1 + \mathrm{I}]$. When finding a change-point within the current range fails, the next search

range is increased to $[1, I + B]$ by a predefined update interval ($B$). This process continues until the end-point of the incumbent range reaches the final interval, $T$. An online implementation of the process is possible once a small $B$ is selected.

The change-point detection algorithm is dependent on the assumption that signals at different time steps would be independently distributed according to a parametric probability density. We mobilized two parametric probability densities, a Gaussian distribution and a Dirichlet distribution, to test whether a change-point could divide a period into two different sub-periods, and a null hypothesis ($H_0$) was set up whereby data within a range ($A = [1, t]$) would come from a single probabilistic distribution. Under the null hypothesis, the maximum log-likelihood could then be computed by Eq. (1) such that the probability that the data were collected could be maximized.

$$\begin{aligned} LL_0 &= \max_{\theta} LL(\theta) \\ &= \max_{\theta} \left[\log(f_\theta(\boldsymbol{x}_1)) + \ldots + \log(f_\theta(\boldsymbol{x}_t))\right] \end{aligned} \tag{1}$$

In Eq. (1), $f_\theta$ is a multivariate probability density function of $d$-dimensional feature variables, and $\theta$ is the vector of function parameters to be estimated.

The log-likelihood under the null hypothesis is a baseline that computes a test statistic to reject the hypothesis and to conclude that a change-point exists within the range. To compute the statistic, another log-likelihood is necessary. The log-likelihood value should be computed under an alternative hypothesis ($H_\tau$). The alternative hypothesis says that a given range would be divided into two sub-ranges, each of which has a different set of parameters while sharing a probabilistic density. Since there are $(t - 2)$ potential change-points in a range (A $=$ [1, t]), the most plausible change-point ($\tau^*$) should be chosen to obtain the maximum log-likelihood ($LL_{\tau^*}$). To do so, for every candidate change-point ($\tau$) from 2 to $(t - 1)$, two maximum log-likelihoods are computed for two sub-ranges [see Eq. (2)], and then an optimal change-point ($\tau^*$) is chosen such that the sum of the two log-likelihoods is maximized [see Eq. (3)].

$$\begin{aligned} LL_\tau &= \max_{\theta_L} \left[\log(f_{\theta_L}(\boldsymbol{x}_1)) + \ldots + \log(f_{\theta_L}(\boldsymbol{x}_\tau))\right] \\ &\quad + \max_{\theta_R} \left[\log(f_{\theta_R}(\boldsymbol{x}_{\tau+1}) + \ldots + \log(f_{\theta_R}(\boldsymbol{x}_t))\right] \end{aligned} \tag{2}$$

In Eq. (2), $\theta_L$ and $\theta_R$ are different parameter sets of a probabilistic density function for the left and right ranges separated by a change-point ($\tau$). The statistic ($z^*$) that is used to reject the null hypothesis is the difference between the log-likelihood ($LL_0$) under the null hypothesis and that ($LL_{\tau^*}$) under the alternative hypothesis with the optimal change-point ($\tau^*$). The test statistic represents the ratio between two different likelihoods [see Eqs. (3) and (4)].

$$\tau^* = \underset{\tau}{argmax}\ LL_\tau \tag{3}$$

$$z^* = LL_{\tau^*} - LL_0 \tag{4}$$

Every statistical test is conducted under the assumption that a null hypothesis is true. A statistic is computed using a sample and measures how far the sample deviates from the population under the null hypothesis. Thus, the statistic is a random variable that varies from sample to sample. If a statistic follows well-known probabilistic distributions such as a normal distribution, a t-distribution, or a chi-square distribution, the probability of being more extreme than a given statistic can be computed based on the probabilistic density function. The null hypothesis can be rejected if the probability (p-value) is less than a predefined threshold (5%), which is called the significance level. The p-value means the risk of a type I error that is committed when the null hypothesis is rejected even though it is true.

There was a problem, however, when the statistical test was applied to the change point detection. The chosen statistic ($z^*$) was not distributed as any well-known probabilistic distribution. Therefore, setting up an empirical distribution was necessary to obtain a threshold to reject the null hypothesis. In principle, the full permutations of data in a given sequence should be enumerated to derive the distribution. The computing time for this, however, is exorbitant. As a practical solution, a sampling scheme was employed in the original paper [18]. A random sample of an arbitrarily chosen size was extracted from the original sequence and regarded as the left sub-sequence. The remainders of the variables were then regarded as the right sub-sequence. This scheme was appropriate because variables in different time intervals were assumed to be independent. A test statistic was computed from the two sub-sequences using Eqs. (2 - 4). Once this sampling task was sufficiently repeated, an empirical distribution of the test statistic could be obtained.

If an optimal test statistic ($z^*$) ranks in the top 5% of the distribution, the null hypothesis can be rejected, and the corresponding change-point is accepted. This statistical test can confirm the change-point ($\tau^*$) found by the algorithm for every time window. Thus, when considering $M$ random permutations, the computational complexity of the statistical test is $O(MI)$. In the present study, a 5% significance level was chosen because it is a market standard, and the sampling task was repeated 100 times to derive an empirical distribution for every time window.

The change-point detection algorithm was applied to features extracted from video images. The algorithm adopted both Gaussian and Dirichlet distributions. For each distribution density function, six different feature dimensions ($d = 1$ to 5, and 10) were tested. Features extracted by an autoencoder were assumed to be distributed as either of the probabilistic distributions. Eq (1) shows how to derive the log-likelihood ($LL$) based on a chosen probabilistic density [$f_\theta(\boldsymbol{x}_i)$].

A Gaussian probabilistic density function has three groups of parameters: means, variances, and covariances. The number of parameters increases proportional to the feature dimension [$= d(d+3)/2$]. The parameters are estimated every time a log-likelihood is computed. Eq. (5) denotes the multivariate

Gaussian density function.

$$f(\boldsymbol{x}_i) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(\boldsymbol{x}_i - \mu)^T \Sigma^{-1}(\boldsymbol{x}_i - \mu)\right) \tag{5}$$

In Eq. (5), $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{id})^T$ is a feature vector in time step $i$, $\mu \in R^d$ is a vector that contains the mean of each feature component, and $\Sigma$ is a variance and covariance matrix for feature components.

A Dirichlet distribution density function is the best fit when dealing with compositional data. Features for the distribution are constrained such that their components should sum to 1. The support for a Dirichlet distribution on the order of $d$ is the ($d$-1)-simplex ($x_{ik} > 0$, $\sum_{k=1}^{d} x_{ik} = 1$). In other words, each point of a sequence should lie on the ($d$-1)-simplex. The number of parameters to be estimated equals the feature dimension, which are fewer than that for a Gaussian distribution density. The computing time for evaluation when using a Dirichlet distribution, on the other hand, is onerous. Eq. (6) denotes the probabilistic density function for a Dirichlet distribution, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)^T$ is a set of parameters to be estimated.

$$f(\boldsymbol{x}_i) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{d} {x_{ik}}^{\alpha_k - 1} \tag{6}$$

In Eq. 6, $x_{ik} > 0$, $\sum_{k=1}^{d} x_{ik} = 1$, and $B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^{d} \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{d} \alpha_k)}$.

It should be noted that each datapoint must be standardized and normalized prior to applying a Dirichlet distribution to detecting a change-point. Data are shifted with the global mean and scaled by the global standard deviation, and then normalized with an Expit function. An Expit function is the inverse of a multinomial logit function that transforms a point in the $d$-dimensional feature space into a point on the $d$-simplex [see Eq. (7)]. The original paper [18] verified how the transformation of data does no harm to the statistical test established above.

$$g(\boldsymbol{x}_i) = \left[\frac{e^{x_{i1}}}{1 + \sum_k^d e^{x_{ik}}}, \ldots, \frac{e^{x_{id}}}{1 + \sum_k^d e^{x_{ik}}}, \frac{1}{1 + \sum_k^d e^{x_{ik}}}\right]^T \tag{7}$$

### C. VEHICLE-COUNTING MODELS FOR MEASURING TRAFFIC DENSITY

Our previous work [38] developed a CNN to count the number of vehicles in the image of a road segment. The CNN took the form of a regression model that could collectively count vehicles rather than adopting a counting-after-detection scheme that depends on a robust object-detection model. The latter model is more accurate, since individual vehicles in an image can be detected and tracked. However, in order to obtain such accuracy, much human effort should be exerted in securing training images labeled with bounding boxes. The architecture of the regression-based CNN model is simple. Former convolutional layers of the model abstract features from an image, and the subsequent dense layers feed a final single node to collectively count the number of vehicles in the image. Labeling images for the model is also much easier than "state-of-the-art" detection models that require drawing a bounding box for every object in an image. Only the number of vehicles in an image can be counted to provide the present CNN model with labeled images.

An abbreviated CNN was devised by removing some of the convolutional layers from the regression-based CNN developed in our previous study [38]. This version of a CNN was applied to counting vehicles in the first testbed. The model estimated vehicle counts separately for each period that the change-point detection algorithm divided. Each period shared the same model architecture but had a separate set of model parameters, because the model was trained and tested on a separate dataset for each period. Fig. 3 shows the proposed architecture of a CNN to count vehicles.

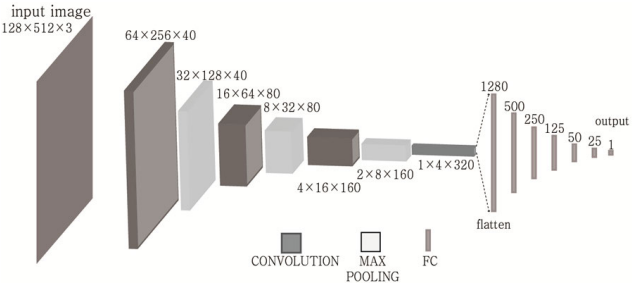
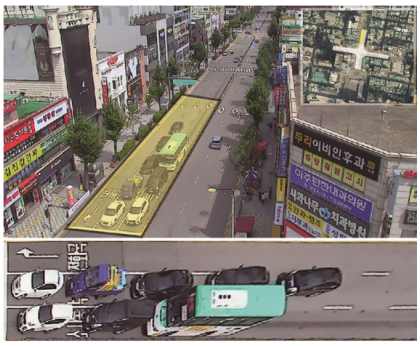



**FIGURE 3.** The CNN architecture used to predict the number of vehicles in an image.

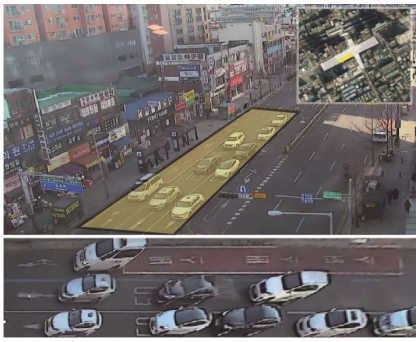
As references, we applied the two representative object detection models in the present study for comparison. A YOLO v3 was chosen because many forms of traffic surveillance have been studied based on this [11]. A key-point-based detection model (CenterNet) [25] developed without anchor boxes was also tested for comparison, since it recorded the best mAP (mean average precision) score when applied to detecting vehicles within images. We pretrained the two models using an open dataset (UA-DETRAC) that included only labeled vehicles [26]. The architectures of both models are featured in the original papers [11] and [25].

## III. TESTBED AND DATA COLLECTION

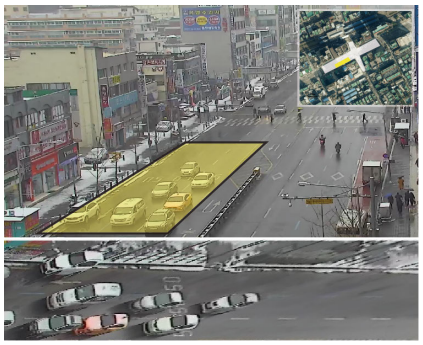
The purpose of detecting change-points is to increase the measurement accuracy of multi-regime analysis for traffic surveillance. Computer vision-based traffic surveillance is largely affected by ambient conditions. Typically, video images vary significantly according to the time of day. Multi-regime analysis can be accomplished by identifying change-points online if model parameters are separately prepared for each recurrent regime. Three testbeds were chosen in an effort to reflect real-world conditions. Three intersection approaches located in Buchon city, Korea, were selected, as shown in Fig. 4, and video images were taken for two days for the former two testbeds. The video in the first testbed was taken in summertime, whereas that in the second testbed was taken in wintertime. In particular, the third testbed was



(a) The first testbed with video taken in summer (August)



(b) The second testbed with video taken in winter (December)



(c) The third testbed with video taken on a snowy day

**FIGURE 4.** **Three testbeds located in Buchon city, Korea.**

selected to confirm if the proposed change point detection algorithm could detect the weather change. A video stream taken on a snowy day was obtained to detect the beginning of snowfall.

For the first and second testbeds, the video frames taken on the first day were used to train an autoencoder to reduce the feature dimensions. The video frames of the second day were divided into times of day based on manual inspection. As a result, 7 different periods were considered: dawn, morning, daytime, afternoon 1, afternoon 2, evening, and nighttime. The criterion used to divide the afternoon periods was based on shadow patterns at the site. Representative images for each period are shown in Fig. 5 along with threshold times. The next section describes how the proposed algorithm was used to detect these thresholds in time and describes the detection accuracy. For the third testbed, a video stream of only a short period was acquired. The video stream includes frames before and after snowfall. Three different regimes were manually identified such as cloudy weather, transient conditions, and heavy snowfall.

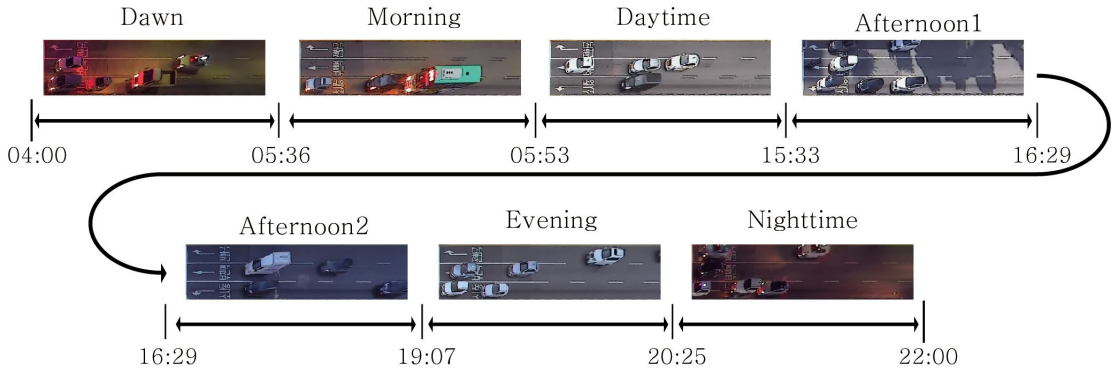
For models used to measure traffic density, only video from the first testbed was utilized because of the burden to manually label images. The regression-based measurement model utilized 1,100 video images that were randomly chosen from each period of the second day. For short periods from which the predefined number of images could not be chosen, video frames taken for the same period in both dates were summed. For each period, 100 images were reserved for testing, and the remaining images were used for training. The overall model was trained and tested on the total amount of data (7,000 images for training and 700 images for testing). The "state-of-the-art" object detectors (YOLO v3 and CenterNet) were fine tuned and tested using 200 images that were randomly chosen for each period and manually labeled with bounding boxes. Half of the images were used for training, and the other half for testing. No less than 1,400 images were labeled with bounding boxes.

## IV. ANALYSIS RESULTS

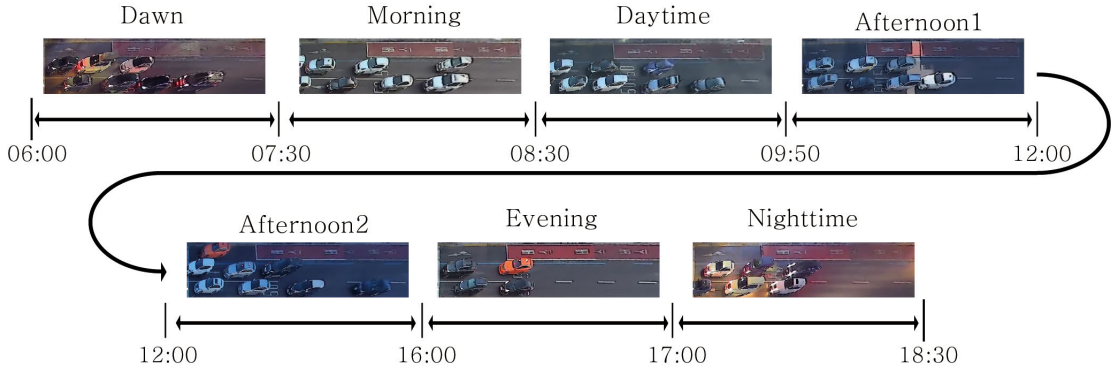
### A. REDUCING THE DIMENSION OF IMAGES

Training an Autoencoder is relatively easy by comparison with training other types of supervised deep-learning models. When training an autoencoder, once an input image is processed by an encoder and a decoder, then the output image is refitted against the input image. This mechanism is unsupervised so that no human effort is required to tag images with labels. In other words, there is no need to secure labeled images because images are simultaneously used as both input and target when training an autoencoder. That is, the mean squared error (MSE) is minimized between input images and the estimated output images. This simple scheme guarantees convergence when training an autoencoder. A trained model is utilized to reduce the dimension of $128 \times 512 \times 3$ pixel images into a vector of a smaller size (1-5, and 10).

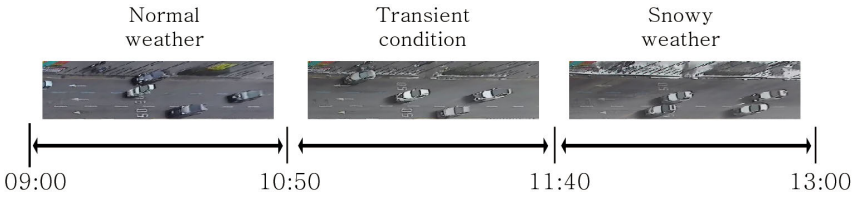
Figs. 6 and 7 depict profiles of reduced features extracted by an autoencoder from video images. Each color indicates a specific feature extracted by an autoencoder. It is impressive that the profile can show different patterns according to specific times of the day. The dotted vertical lines in each graph indicate the predicted change-points, whereas small arrows in the top timeline mark the ground-truth change-points that were determined by manually examining the video images. Regarding the first testbed, in cases where the Gaussian density was adopted for 3-dimensional abstracted features, changing patterns in the features are exactly consistent with those from the ground-truth change-points [see Fig. 6 (a)]. There were slightly inferior results from the second testbed. No case was found with a perfect match. A false positive was shown even for the best cases of both the Gaussian and Dirichlet probabilistic densities where 3-dimensional features were adopted [see Fig. 7 (a) and (b)]. The proposed algorithm




(a) The first testbed with video taken in summer (August)




(b) The second testbed with video taken in winter (December)




(c) The third testbed with video taken on a snowy day

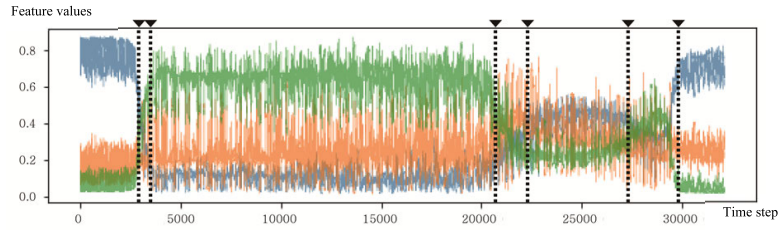
**FIGURE 5.** Different ambient conditions by times of day in testbeds. it is clear that the shadow length and the sunlight intensity are quite different among the three daytime periods (daytime, afternoon1, and afternoon2).

falsely detected a redundant change point in the fifth ground truth period. Nonetheless, all ground truth change-points were found exactly.

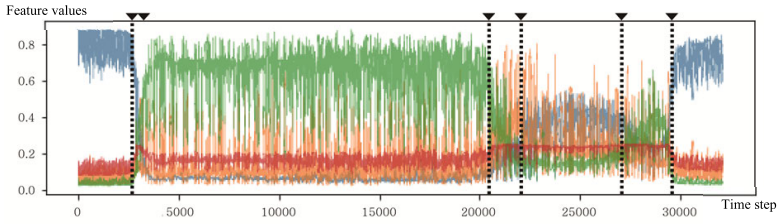
The objective to choose the third testbed is different from the former two testbeds. About four hours of video stream, which includes conditions before and after snowfall, was used for detecting weather change. With the naked eye we recognized a transient period between cloudy weather and heavy snowfall. The proposed change-point detection algorithm exactly found the beginning and end of the transient period. Fig. 8 shows the search cases of weather changes from cloudy conditions to heavy snowfall according to the choice of probabilistic density function or window sizes. With the exception of the first case [Fig. 8 (a)] with a false positive, the remaining searches were perfect. It should be noted that the main objective of the present study was not to recognize weather change but to find homogeneous periods that are recurrently found daily. The latter detections would be much more difficult for the algorithm to carry out.

### B. PERFORMANCE OF THE PROPOSED CHANGE-POINT DETECTION ALGORITHM

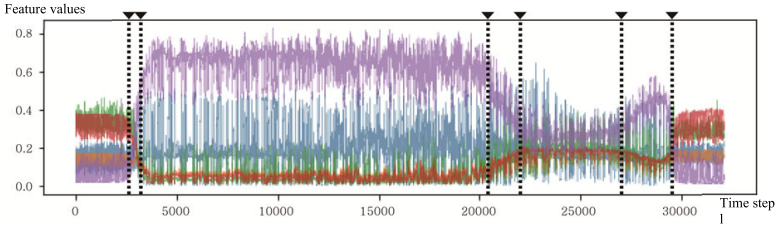
The proposed change-point detection algorithm was applied to an image sequence of 18 hours (04:00-22:00) on the second day for the first testbed, and to another image sequence of 12.5 hours (06:00-18:30) on the second day for the second testbed. The sequence used intervals of 2.0 seconds, and thus the total number of periods was 32,123 and 20,900 for the first and second testbed, respectively. There were some missing video frames for the first and second testbeds. The missing frames were ignored in the present analysis. The algorithm employed the two probabilistic densities with 6 different



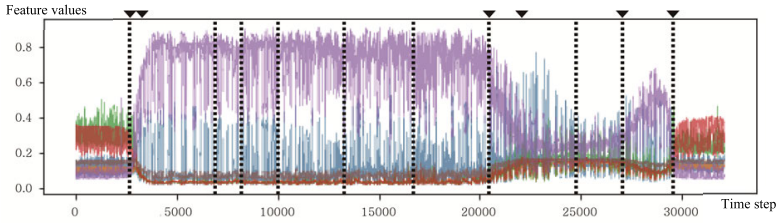

(a) Gaussian distribution for 3-dimensional features with a window size of 6,000



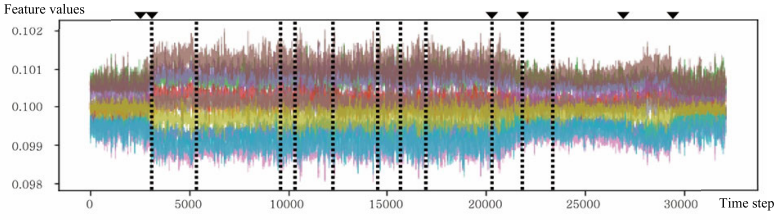

(b) Dirichlet distribution for 3-dimensional features with a window size of 5,000



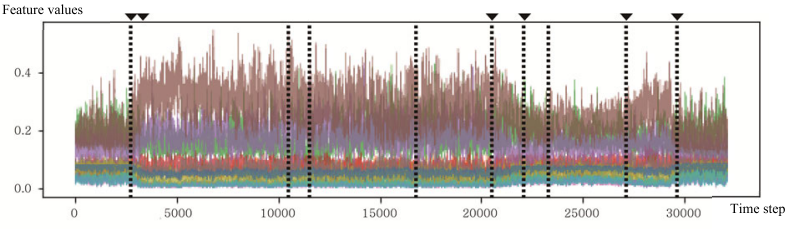

(c) Gaussian distribution for 5-dimensional features with a window size of 6,000




(d) Dirichlet distribution for 5-dimensional features with a window size of 5,000

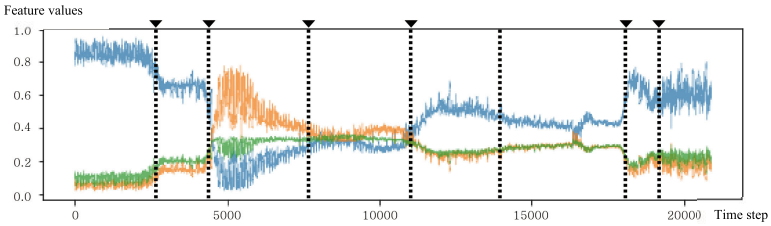



(e) Gaussian distribution for 10-dimensional features with a window size of 5,000

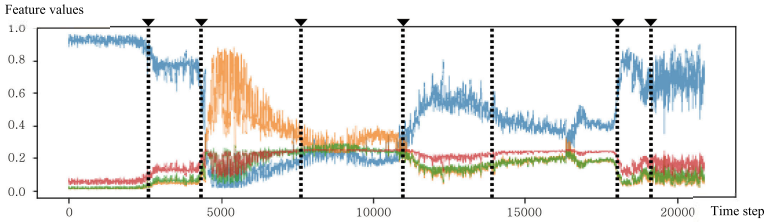



(f) Dirichlet distribution for 10-dimensional features with a window size of 5,000

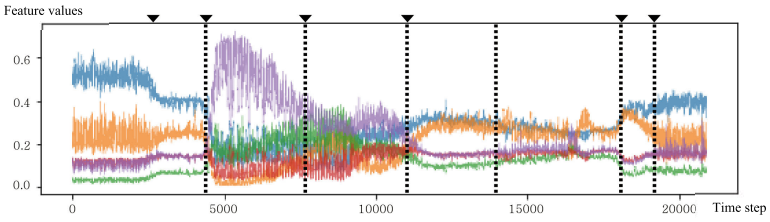
**FIGURE 6.** **The profile of reduced features for 128 × 512 × 3 pixel images (1st testbed).**



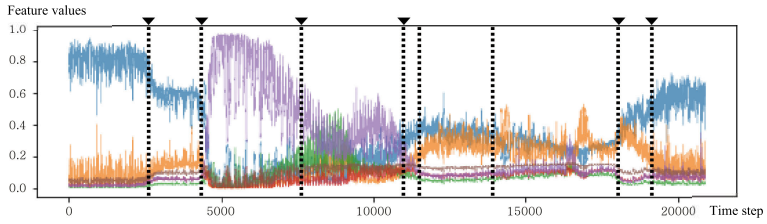

(a) Gaussian distribution for 3-dimensional features with a window size of 5,000



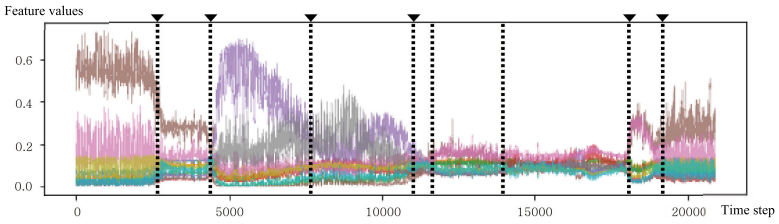

(b) Dirichlet distribution for 3-dimensional features with a window size of 5,000



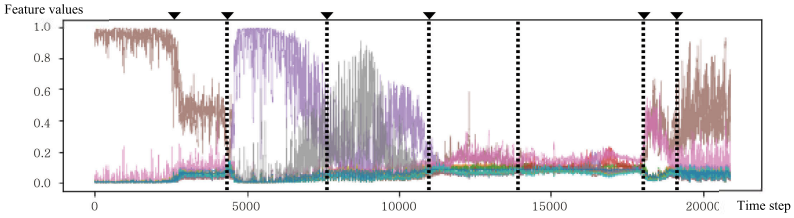

(c) Gaussian distribution for 5-dimensional features with a window size of 5,000




(d) Dirichlet distribution for 5-dimensional features with a window size of 5,000

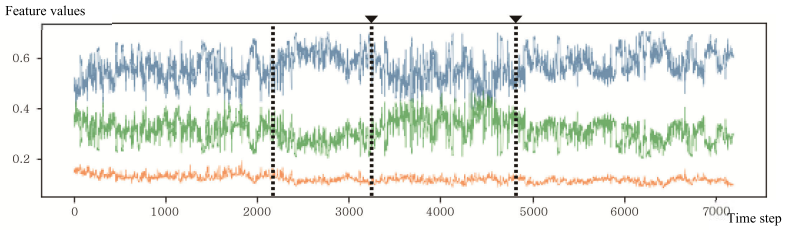



(e) Gaussian distribution for 10-dimensional features with a window size of 5,000

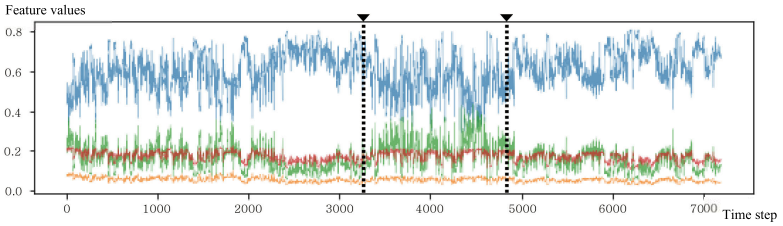



(f) Dirichlet distribution for 10-dimensional features with a window size of 5,000

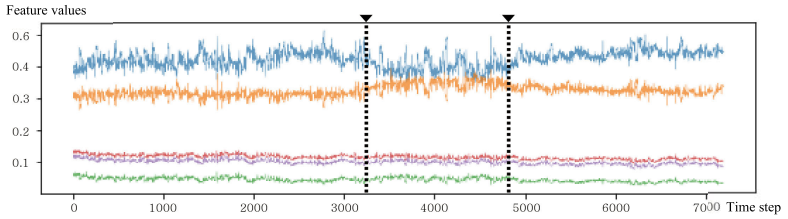
**FIGURE 7.** The profile of reduced features for 128 × 512 × 3 pixel images (2nd testbed).



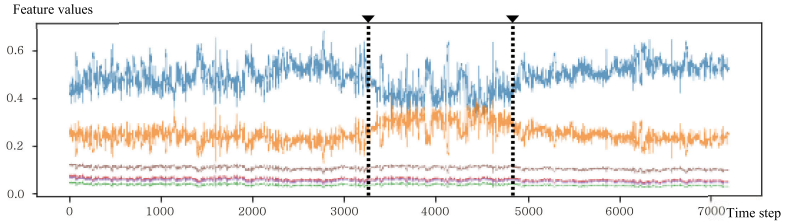

(a) Gaussian distribution for 3-dimensional features with a window size of 3,000



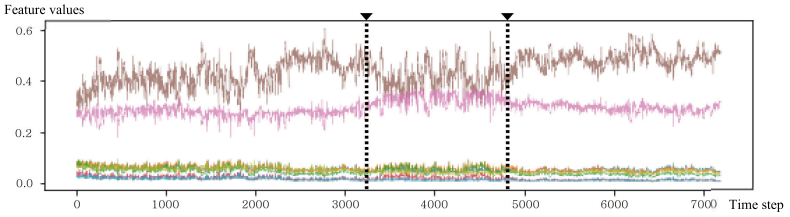

(b) Dirichlet distribution for 3-dimensional features with a window size of 5,000



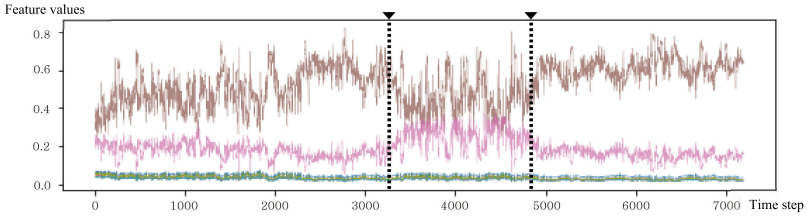

(c) Gaussian distribution for 5-dimensional features with a window size of 5,000




(d) Dirichlet distribution for 5-dimensional features with a window size of 5,000




(e) Gaussian distribution for 10-dimensional features with a window size of 5,000




(f) Dirichlet distribution for 10-dimensional features with a window size of 4,000

**FIGURE 8.** The profile of reduced features for 128 × 512 × 3 pixel images (3rd testbed).

**TABLE 1.** Precision and recall when detecting change-points for the first testbed.

(a) Test results from 1-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.429 | 0.5 | 0.462 | | 3,000 | 0.211 | 0.667 | 0.32 |
| | 4,000 | 0.286 | 0.333 | 0.308 | | 4,000 | 0.188 | 0.5 | 0.273 |
| | 5,000 | 0.667 | 0.333 | 0.444 | | 5,000 | 0.25 | 0.5 | 0.333 |
| | 6,000 | 0.286 | 0.333 | 0.308 | | 6,000 | 0.25 | 0.5 | 0.333 |

(b) Test results from 2-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.417 | 0.833 | 0.556 | | 3,000 | 0.25 | 0.833 | 0.385 |
| | 4,000 | 0.333 | 0.667 | 0.444 | | 4,000 | 0.25 | 0.667 | 0.364 |
| | 5,000 | 0.429 | 0.5 | 0.462 | | 5,000 | 0.333 | 0.667 | 0.444 |
| | 6,000 | 0.6 | 0.5 | 0.545 | | 6,000 | 0.6 | 1.0 | 0.75 |

(c) Test results from 3-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.417 | 0.833 | 0.556 | | 3,000 | 0.5 | 0.833 | 0.625 |
| | 4,000 | 0.375 | 0.5 | 0.429 | | 4,000 | 0.462 | 1.0 | 0.632 |
| | 5,000 | 0.75 | 1.0 | 0.857 | | 5,000 | 1.0 | 0.833 | 0.909 |
| | 6,000 | 1.0 | 1.0 | 1.0 | | 6,000 | 0.833 | 0.833 | 0.833 |

(d) Test results from 4-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.4 | 0.667 | 0.5 | | 3,000 | 0.211 | 0.667 | 0.32 |
| | 4,000 | 0.5 | 0.667 | 0.572 | | 4,000 | 0.333 | 0.833 | 0.476 |
| | 5,000 | 0.625 | 0.833 | 0.714 | | 5,000 | 0.364 | 0.667 | 0.471 |
| | 6,000 | 0.667 | 0.667 | 0.667 | | 6,000 | 0.417 | 0.833 | 0.556 |

(e) Test results from 5-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.667 | 1.0 | 0.8 | | 3,000 | 0.235 | 0.667 | 0.348 |
| | 4,000 | 0.714 | 0.833 | 0.769 | | 4,000 | 0.308 | 0.667 | 0.421 |
| | 5,000 | 0.833 | 0.833 | 0.833 | | 5,000 | 0.4 | 0.667 | 0.5 |
| | 6,000 | 1.0 | 1.0 | 1.0 | | 6,000 | 0.2 | 0.333 | 0.25 |

(f) Test results from 10-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.222 | 0.667 | 0.333 | | 3,000 | 0.263 | 0.833 | 0.4 |
| | 4,000 | 0.154 | 0.333 | 0.211 | | 4,000 | 0.286 | 0.667 | 0.4 |
| | 5,000 | 0.273 | 0.5 | 0.353 | | 5,000 | 0.556 | 0.833 | 0.667 |
| | 6,000 | 0.111 | 0.167 | 0.133 | | 6,000 | 0.4 | 0.667 | 0.5 |

feature dimensions. For each density function, 4 different window sizes were examined. We tested window sizes from 3,000 to 6,000 in increments of 1,000. The optimal window size was determined when a maximum precision and recall was found. The total number of intervals was tantamount to 32,123 for the first testbed, so the proportion of the candidate window sizes ranged from 9.3 to 18.7%. The precision and recall of the change-point detection algorithm were computed for a total of 48 different cases across the former two testbeds. When a predicted change-point approximated the ground truth within the predefined margin of error (= 420 intervals), the detection was regarded as successful.

Tables 1 and 2 show the test results of detecting change-points for the feature sequence of images. For the first testbed, when using Gaussian density, a window size of 6,000 recorded the best performance with the smaller feature dimensions (= 3 and 5). That is, both performance indices were 1.0, which meant that every ground truth change-point was detected with neither false positives nor false negatives. We also confirmed that a quite large window size (= 6000) is necessary for the perfect detection of change-points. It is interesting that the detection failed when the number of features was increased to 10 [see Fig. 6 (e)]. This implies that the use of many features to accommodate more conditions in an image may be ineffective in finding change-points attributable only to the time of day. More concretely, some of the 10 features may contain traffic conditions that impede the recognition of ambient conditions. This result proved that using 3 to 5 abstracted features is sufficient to extract the necessary information from an image when using a Gaussian probabilistic density. Regarding the Dirichlet density, the change-point detection algorithm was implemented after using the Expit function [Eq. (7)] to transform abstracted features into compositional data. Detection was never perfect

**TABLE 2.** Precision and recall when detecting change-points for the second testbed.

(a) Test results from 1-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.417 | 0.833 | 0.556 | | 3,000 | 0.417 | 0.833 | 0.556 |
| | 4,000 | 0.667 | 1.0 | 0.8 | | 4,000 | 0.667 | 1.0 | 0.8 |
| | 5,000 | 0.833 | 0.833 | 0.833 | | 5,000 | 0.857 | 1.0 | 0.923 |
| | 6,000 | 0.833 | 0.833 | 0.833 | | 6,000 | 0.833 | 0.833 | 0.833 |

(b) Test results from 2-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.444 | 0.667 | 0.533 | | 3,000 | 0.455 | 0.833 | 0.588 |
| | 4,000 | 0.556 | 0.833 | 0.667 | | 4,000 | 0.6 | 1.0 | 0.75 |
| | 5,000 | 0.667 | 0.667 | 0.667 | | 5,000 | 0.667 | 0.667 | 0.667 |
| | 6,000 | 0.667 | 0.667 | 0.667 | | 6,000 | 0.667 | 0.667 | 0.667 |

(c) Test results from 3-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.5 | 0.833 | 0.625 | | 3,000 | 0.455 | 0.833 | 0.588 |
| | 4,000 | 0.556 | 0.833 | 0.667 | | 4,000 | 0.6 | 1.0 | 0.75 |
| | 5,000 | 0.857 | 1.0 | 0.923 | | 5,000 | 0.857 | 1.0 | 0.923 |
| | 6,000 | 0.833 | 0.833 | 0.833 | | 6,000 | 0.833 | 0.833 | 0.833 |

(d) Test results from 4-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.545 | 1.0 | 0.706 | | 3,000 | 0.5 | 1.0 | 0.667 |
| | 4,000 | 0.667 | 1.0 | 0.8 | | 4,000 | 0.6 | 1.0 | 0.75 |
| | 5,000 | 0.857 | 1.0 | 0.923 | | 5,000 | 0.833 | 0.833 | 0.833 |
| | 6,000 | 0.857 | 1.0 | 0.923 | | 6,000 | 0.833 | 0.833 | 0.833 |

(e) Test results from 5-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.5 | 1.0 | 0.667 | | 3,000 | 0.545 | 1.0 | 0.706 |
| | 4,000 | 0.667 | 1.0 | 0.8 | | 4,000 | 0.667 | 1.0 | 0.8 |
| | 5,000 | 0.833 | 0.833 | 0.833 | | 5,000 | 0.75 | 1.0 | 0.857 |
| | 6,000 | 0.833 | 0.833 | 0.833 | | 6,000 | 0.833 | 0.833 | 0.833 |

(f) Test results from 10-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.556 | 0.833 | 0.667 | | 3,000 | 0.6 | 1.0 | 0.75 |
| | 4,000 | 0.556 | 0.833 | 0.667 | | 4,000 | 0.667 | 1.0 | 0.8 |
| | 5,000 | 0.75 | 1.0 | 0.857 | | 5,000 | 0.833 | 0.833 | 0.833 |
| | 6,000 | 0.833 | 0.833 | 0.833 | | 6,000 | 0.833 | 0.833 | 0.833 |

when using Dirichlet density. The best performance was recorded when 3 abstracted features were used with a window size of 5,000. The precision was 1.0, but the recall was 0.833, since only 5 out of 6 ground truth change-points were properly detected and the remaining one was missed. In terms of the results above, the Gaussian distribution was more robust than the Dirichlet in finding change-points for the times of day.

Results from the second testbed were similar except for a slight deterioration in performance. Fortunately, there was no missing point among the ground truth change-points. As mentioned earlier, a single false positive was found in the second longest ground truth period, but this is not a serious problem because an additional calibration of a further vehicle counting model might do little harm to the overall performance of traffic surveillance. A more important finding is that the proposed change-point detection algorithm worked well for different circumstances (summertime or wintertime). As shown in Figs 6 and 7, although the signals from two testbeds follow quite different patterns, the proposed change-point algorithm exactly detected real multi-regimes. For the second testbed, no performance gap was found between the Gaussian and Dirichlet distributions. For the Dirichlet distribution, using only a single feature yielded performance equivalent to employing 3 features. This implies that an autoencoder played a more significant role in abstracting ambient

**TABLE 3.** Precision and recall when detecting change-points for the third testbed.

(a) Test results from 1-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.5 | 0.5 | 0.5 | | 3,000 | 0.5 | 0.5 | 0.5 |
| | 4,000 | 0.5 | 0.5 | 0.5 | | 4,000 | 0.5 | 0.5 | 0.5 |
| | 5,000 | 0.5 | 0.5 | 0.5 | | 5,000 | 0.5 | 0.5 | 0.5 |
| | 6,000 | 1.0 | 0.5 | 0.667 | | 6,000 | 1.0 | 0.5 | 0.667 |

(b) Test results from 2-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.667 | 1.0 | 0.8 | | 3,000 | 0.667 | 1.0 | 0.8 |
| | 4,000 | 1.0 | 0.5 | 0.667 | | 4,000 | 1.0 | 1.0 | 1.0 |
| | 5,000 | 1.0 | 0.5 | 0.667 | | 5,000 | 1.0 | 1.0 | 1.0 |
| | 6,000 | 1.0 | 0.5 | 0.667 | | 6,000 | 1.0 | 0.5 | 0.667 |

(c) Test results from 3-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.667 | 1.0 | 0.8 | | 3,000 | 0.667 | 1.0 | 0.8 |
| | 4,000 | 0.5 | 0.5 | 0.5 | | 4,000 | 0.667 | 1.0 | 0.8 |
| | 5,000 | 0.5 | 0.5 | 0.5 | | 5,000 | 1.0 | 1.0 | 1.0 |
| | 6,000 | 0.5 | 0.5 | 0.5 | | 6,000 | 1.0 | 1.0 | 1.0 |

(d) Test results from 4-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.667 | 1.0 | 0.8 | | 3,000 | 0.667 | 1.0 | 0.8 |
| | 4,000 | 0.5 | 0.5 | 0.5 | | 4,000 | 0.5 | 0.5 | 0.5 |
| | 5,000 | 1.0 | 1.0 | 1.0 | | 5,000 | 0.5 | 0.5 | 0.5 |
| | 6,000 | 1.0 | 1.0 | 1.0 | | 6,000 | 1.0 | 0.5 | 0.667 |

(e) Test results from 5-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.667 | 1.0 | 0.8 | | 3,000 | 0.667 | 1.0 | 0.8 |
| | 4,000 | 1.0 | 1.0 | 1.0 | | 4,000 | 1.0 | 1.0 | 1.0 |
| | 5,000 | 1.0 | 1.0 | 1.0 | | 5,000 | 1.0 | 1.0 | 1.0 |
| | 6,000 | 1.0 | 1.0 | 1.0 | | 6,000 | 1.0 | 1.0 | 1.0 |

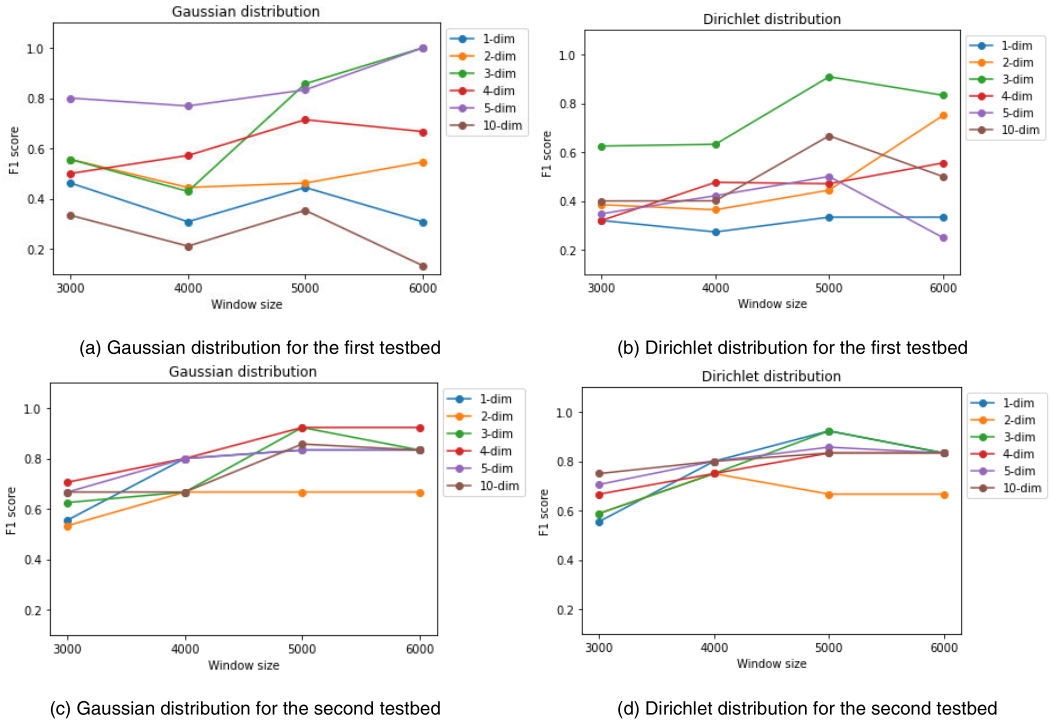
(f) Test results from 10-dimensional features

| Gaussian distribution | Window size | Precision | Recall | F1 score | Dirichlet distribution | Window size | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|---|---|
| | 3,000 | 0.667 | 1.0 | 0.8 | | 3,000 | 0.667 | 1.0 | 0.8 |
| | 4,000 | 0.5 | 0.5 | 0.5 | | 4,000 | 1.0 | 1.0 | 1.0 |
| | 5,000 | 1.0 | 1.0 | 1.0 | | 5,000 | 1.0 | 0.5 | 0.667 |
| | 6,000 | 1.0 | 1.0 | 1.0 | | 6,000 | 1.0 | 0.5 | 0.667 |

conditions in wintertime. Fig. 8 graphically shows the empirical relationships between the number of features and the window size for the Gaussian and Dirichlet distributions. As mentioned earlier, the proposed change point detection algorithm almost perfectly found the transient period between before and after snowfall. Table 3 lists the precision and recall scores of detecting snowy weather. Unlike the search for the former two testbeds, perfect detection was achieved for most cases of feature dimensions when a proper probabilistic density function was chosen. As expected, detecting a special weather condition proved to be easier than discerning ambient conditions that change recurrently on a daily basis.

### C. PERFORMANCE OF THE MULTI-REGIME APPROACH IN MEASURING TRAFFIC DENSITY

A regression-based simple detector was employed as a baseline in the present study. The reason the model was selected as a baseline was because of the ease of labeling the data. As reference models, a YOLO v3 [11] and a CenterNet [25], which are representative object detectors, were applied to supporting the validity of the chosen multi-regime analysis. The two models were pretrained on an open dataset (UA-DETRAC) that provides annotated vehicle images. To fine tune and test the models for each period, only 200 local images were manually labeled with bounding boxes. Due to the difficulty of labeling local images, the test




(a) Gaussian distribution for the first testbed

(b) Dirichlet distribution for the first testbed

(c) Gaussian distribution for the second testbed

(d) Dirichlet distribution for the second testbed

**FIGURE 9.** The performance to detect change-points according to the window size.

**TABLE 4.** Model performance used to measure traffic density (RMSE) for different times of the day.

| RMSE | Dawn (04:00-05:36) 96 min. | Morning (05:36-05:53) 17 min. | Daytime (05:53-15:33) 580 min. | Afternoon1 (15:33-16:29) 56 min. | Afternoon2 (16:29-19:07) 158 min. | Evening (19:07-20:25) 78 min. | Nighttime (20:25-22:00) 95 min. |
|---|---|---|---|---|---|---|---|
| The regression-based model with separate training | 0.345 | 0.917 | 0.895 | 1.184 | 1.150 | 0.712 | 0.615 |
| The regression-based model with integrated training | 0.467 | 0.904 | 1.464 | 0.880 | 1.346 | 0.742 | 0.792 |
| YOLO v3 with separate training | 0.755 | 0.566 | 0.608 | 1.794 | 1.606 | 1.034 | 0.959 |
| YOLO v3 with integrated training | 0.975 | 1.025 | 0.721 | 1.145 | 2.128 | 1.575 | 1.039 |
| CenterNet with separate training | 0.520 | 1.606 | 0.911 | 0.933 | 1.421 | 1.229 | 0.458 |
| CenterNet with integrated training | 0.787 | 0.906 | 1.192 | 0.529 | 1.584 | 1.241 | 0.640 |

for the traffic density measurement was carried out only for the first testbed.

Each model was trained in two different ways to distinguish the utility of the proposed multi-regime measurement. Basically, the models were trained on images from each period divided by the proposed change-point detection algorithm (=separate training). They were also trained using the entire dataset for all time periods (=integrated training). The model performance was measured by the root mean square error (RMSE) between observed and predicted vehicle counts. The test performance was measured for each period. We did not compare the performance across three different models because the performance depends on the amount of training data. Instead, the enhancement in the performance by separate training was the focus for each model.

The test results from the three models are listed in Table 4. Regarding the results from the regression-based measurement model, separated learning outperformed integrated learning for the 5 periods highlighted in the table. The remaining two periods were too short to secure a sufficient amount of

**TABLE 5.** Model performance used to measure traffic density (RMSE) for all time periods.

| | |
|---|---|
| The regression-based model with separate training | 0.863 |
| The regression-based model with integrated training | 1.211 |
| YOLO v3 with separate training | 0.890 |
| YOLO v3 with integrated training | 1.066 |
| CenterNet with separate training | 0.946 |
| CenterNet with integrated training | 1.129 |

different images, although data collected during the two days were used. The performance of the overall model with integrated training was slightly better than that of the separated model for the remaining two periods. If multi-date data are available, the performance would be enhanced because the two periods turned out to be non-homogeneous according to the results of change-point detection.

A YOLO v3 was pretrained on the UA-DETRAC, which is an open dataset that includes only vehicle objects with labels. This ''state-of-the-art'' detector turned out to be somewhat less accurate than the proposed regression-based model (see Table 4). The performance would be improved if the detector was fine tuned with more locally annotated images. However, such a labeling task required too much effort for the present study. Nonetheless, the vehicle counting performance of separate training was superior to integrated training when a YOLO v3 was adopted to count vehicles. This implies that the proposed multi-regime approach would also work well with a ''state-of-the-art'' computer vision algorithm. The reverse results were obtained only for a period of afternoon 1 that was relatively short. When a CenterNet was employed to count vehicles, the results were similar. Separate fine tuning was superior to the integration of training for every period. The integrated training was better for two relatively short periods, but the remaining periods showed the same results as the former two measurement models.

Table 5 shows the overall RMSEs that were computed over all periods based on both the separated and the integrated trainings. The overall performance verifies that the proposed multi-regime analysis with separate training is superior to single-regime analysis with integrated training, even though integrated training outperformed separate training for some periods, as shown in Table 4. This superiority was secured regardless of the choice of measurement models.

## V. CONCLUSION

The present study dealt with the problem whereby computer vision-based traffic surveillance technology must account for the effects of different times of the day. A multi-regime analysis was set up and tested for counting vehicles in an image. The adopted change-point detection algorithm was robust in finding changing patterns in images according to times of the day. A statistical test method supported the theoretical background that was used to determine thresholds that separate multiple periods. An autoencoder was used to extract a small number of features from an image containing ambient conditions. Such an abstraction made it possible to implement the change-point detection algorithm within a practical computing time.

The test results from counting vehicles in the testbed verified that the proposed multi-regime analysis outperformed an integrated model that was trained on the entire dataset that encompassed all periods. The test results proved that the proposed change-point detection algorithm could be applied to real-world online traffic surveillance.

In further studies, robust models for measuring traffic volumes and speeds will be tested together using the proposed change-point detection algorithm. In addition, the performance of detecting change-points should be validated for more diverse cases including special weather conditions. It is also necessary to video extended periods of time in order to deal with seasonal changes. The proposed approach is expected to detect such special conditions once such an extended video is available for training the model, because the approach works well for discerning conditions during a normal day, which is much harder than detecting special conditions.

## REFERENCES

[1] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, ''Vehicle detection in satellite images by hybrid deep convolutional neural networks,'' *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 10, pp. 1797–1801, Oct. 2014, doi: 10.1109/LGRS.2014.2309695.
[2] B. Li, ''3D fully convolutional network for vehicle detection in point cloud,'' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vancouver, BC, Canada, Sep. 2017, pp. 1513–1518, doi: 10.1109/IROS.2017.8205955.
[3] L. Wang, Y. Lu, H. Wang, Y. Zheng, H. Ye, and X. Xue, ''Evolving boxes for fast vehicle detection,'' in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Hong Kong, Jul. 2017, pp. 1135–1140, doi: 10.1109/ICME.2017.8019461.
[4] L. W. Sommer, T. Schuchert, and J. Beyerer, ''Fast deep vehicle detection in aerial images,'' in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Santa Rosa, CA, USA, Mar. 2017, pp. 311–319, doi: 10.1109/WACV.2017.41.
[5] N. Audebert, B. Le Saux, and S. Lefèvre, ''Segment-before-detect: Vehicle detection and classification through semantic segmentation of aerial images,'' *Remote Sens.*, vol. 9, no. 4, p. 368, Apr. 2017, doi: 10.3390/rs9040368.
[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, ''Rich feature hierarchies for accurate object detection and semantic segmentation,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, Jun. 2014, pp. 580–587, doi: 10.1109/CVPR.2014.81.
[7] R. Girshick, ''Fast R-CNN,'' in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, CA, USA, Dec. 2015, pp. 1440–1448, doi: 10.1109/ICCV.2015.169.
[8] S. Ren, K. He, R. Girshick, and J. Sun, ''Faster R-CNN: Towards real-time object detection with region proposal networks,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.
[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ''You only look once: Unified, real-time object detection,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[10] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 6517–6525, doi: 10.1109/CVPR.2017.690.
[11] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," Apr. 2018, *arXiv:1804.02767*. Accessed: Sep. 17, 2020. [Online]. Available: http://arxiv.org/abs/1804.02767
[12] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," Apr. 2020, *arXiv:2004.10934*. Accessed: Sep. 17, 2020. [Online]. Available: http://arxiv.org/abs/2004.10934
[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, and C.-Y. Fu, "SSD: Single shot multibox detector," in *Proc. Comput. Vis.-ECCV*, Oct. 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.
[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 2999–3007, doi: 10.1109/ICCV.2017.324.
[15] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu, "UA-DETRAC: A new benchmark and protocol for multi-object tracking," *Comput. Vis. Image Understand.*, vol. 193, Apr. 2020, Art. no. 102907.
[16] L. Wen, D. Du, S. Li, X. Bian, and S. Lyu, "Learning non-uniform hypergraph for multi-object tracking," in *Proc. AAAI*, 2019, pp. 8981–8988.
[17] P. Sevekar and S. B. Dhonde, "Nighttime vehicle detection for intelligent headlight control: A review," in *Proc. 2nd Int. Conf. Appl. Theor. Comput. Commun. Technol. (iCATccT)*, Jul. 2016, pp. 188–190.
[18] K. J. Prabuchandran, N. Singh, P. Dayama, and V. Pandit, "Change point detection for compositional multivariate data," 2019, *arXiv:1901.04935*. [Online]. Available: http://arxiv.org/abs/1901.04935
[19] S. Padakandla, K. J. Prabuchandran, and S. Bhatnagar, "Reinforcement learning algorithm for non-stationary environments," *Appl. Intell.*, vol. 50, no. 11, pp. 3590–3606, Nov. 2020.
[20] A. Naug, M. Quiñones-Grueiro, and G. Biswas, "A relearning approach to reinforcement learning for control of smart buildings," 2020, *arXiv:2008.01879*. [Online]. Available: http://arxiv.org/abs/2008.01879
[21] M. Haliem, V. Aggarwal, and B. Bhargava, "AdaPool: An adaptive model-free ride-sharing approach for dispatching using deep reinforcement learning," in *Proc. 7th ACM Int. Conf. Syst. Energy-Efficient Buildings, Cities, Transp.*, Nov. 2020, pp. 304–305.
[22] X. Chen, Y. Yang, S. Wang, H. Wu, J. Tang, J. Zhao, and Z. Wang, "Ship type recognition via a coarse-to-fine cascaded convolution neural network," *J. Navigat.*, vol. 73, no. 4, pp. 813–832, Jul. 2020.
[23] X. Chen, S. Wang, C. Shi, H. Wu, J. Zhao, and J. Fu, "Robust ship tracking via multi-view learning and sparse representation," *J. Navigat.*, vol. 72, no. 1, pp. 176–192, Jan. 2019.
[24] Q. Zhang and T. Chu, "Structure regularized traffic monitoring for traffic matrix estimation and anomaly detection by link-load measurements," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 12, pp. 2797–2807, Dec. 2016.
[25] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," 2019, *arXiv:1904.07850*. [Online]. Available: http://arxiv.org/abs/1904.07850
[26] *UA-DTRAC Dataset*. Accessed: Mar. 10, 2021. [Online]. Available: http://www.cs.albany.edu/cvml/cvml_downloads.html
[27] T. Zhang and B. Yang, "Big data dimension reduction using PCA," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2016, pp. 152–157.
[28] R. Saraçoğlu, "Hidden Markov model-based classification of heart valve disease with PCA for dimension reduction," *Eng. Appl. Artif. Intell.*, vol. 25, no. 7, pp. 1523–1528, Oct. 2012.
[29] J. Zabalza, J. Ren, J. Zheng, H. Zhao, C. Qing, Z. Yang, P. Du, and S. Marshall, "Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging," *Neurocomputing*, vol. 185, pp. 1–10, Apr. 2016.
[30] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[31] I. Auger and C. Lawrence, "Algorithms for the optimal identification of segment neighborhoods," *Bull. Math. Biol.*, vol. 51, no. 1, pp. 39–54, 1989.
[32] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of change points with a linear computational cost," *J. Amer. Stat. Assoc.*, vol. 107, no. 500, pp. 1590–1598, 2012.
[33] G. D. Montanez, S. Amizadeh, and N. Laptev, "Inertial hidden Markov models: Modeling change in multivariate time series.," in *Proc. AAAI*, 2015, pp. 1819–1825.
[34] K. Bleakley and J.-P. Vert, "The group fused lasso for multiple change-point detection," 2011, *arXiv:1106.4199*. [Online]. Available: http://arxiv.org/abs/1106.4199
[35] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Netw.*, vol. 43, pp. 72–83, Jul. 2013.
[36] D. S. Matteson and N. A. James, "A nonparametric approach for multiple change point analysis of multivariate data," *J. Amer. Stat. Assoc.*, vol. 109, no. 505, pp. 334–345, Jan. 2014.
[37] T. Wang and R. J. Samworth, "High dimensional change point estimation via sparse projection," *J. Roy. Stat. Soc. B, Stat. Methodol.*, vol. 80, no. 1, pp. 57–83, Jan. 2018.
[38] J. Chung and K. Sohn, "Image-based learning to measure traffic density using a deep convolutional neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1670–1675, May 2018.

**SEUNGYUN JEONG** is currently pursuing the degree with the Department of Urban Engineering, Chung-Ang University. His research interests include machine learning and visual recognition.

**SEUNGBIN ROH** is currently with the Laboratory of Big-Data Applications in Public Sector, Chung-Ang University. He is also preparing his thesis for the M.S. degree, the theme of which covers applying an autoencoder to measuring traffic volumes without manually tagging images.

**KEEMIN SOHN** is currently a Professor with the Department of Urban Engineering, Chung-Ang University. His research interests include the applications of artificial intelligence to transportation engineering and planning.