AIC2018 Report: Traffic Surveillance Research

Tingyu Mao, Wei Zhang, Haoyu He, Yanjun Lin, Vinay Kale, Alexander Stein, Zoran Kostic

Columbia University

Abstract

Traffic surveillance and management technologies are some of the most intriguing aspects of smart city applications. In this paper, we investigate and present the methods for vehicle detections, tracking, speed estimation and anomaly detection for NVIDIA AI City Challenge 2018 (AIC2018). We applied Mask-RCNN and deep-sort for vehicle detection and tracking in track 1, and optical flow based method in track 2. In track 1, we achieve 100% detection rate and 7.97 mile/hour estimation error for speed estimation.

Track 1: Vehicle Speed Estimation

In track 1, it is required to estimate multiple vehicle's real-time velocity under different scenarios which comprise 2 highway scenes and 2 intersection scene. To achieve this goal, our system consists of three parts, as shown in Figure 2: (a) Vehicle detection by Mask RCNN network. Mask RCNN[1] is two-stage detector and predicts accurate detection bounding boxes as well as the segmentation simultaneously. Additional segmentation information will be helpful in having more accurate estimation of vehicle physical coordinate. (b) Associate the detected vehicles across different frames by deep appearance descriptor. (c) Speed estimation by track-lets. To reconstruct 3D coordinate from image, we adopt the conventional projective matrix method. Landmark points are manually selected from the videos and their corresponding physical location are attained from Google Map. Then, we consider the central position of bottom contour points as the vehicle position and calculate velocity based on a series of smoothed 3D position coordinates.

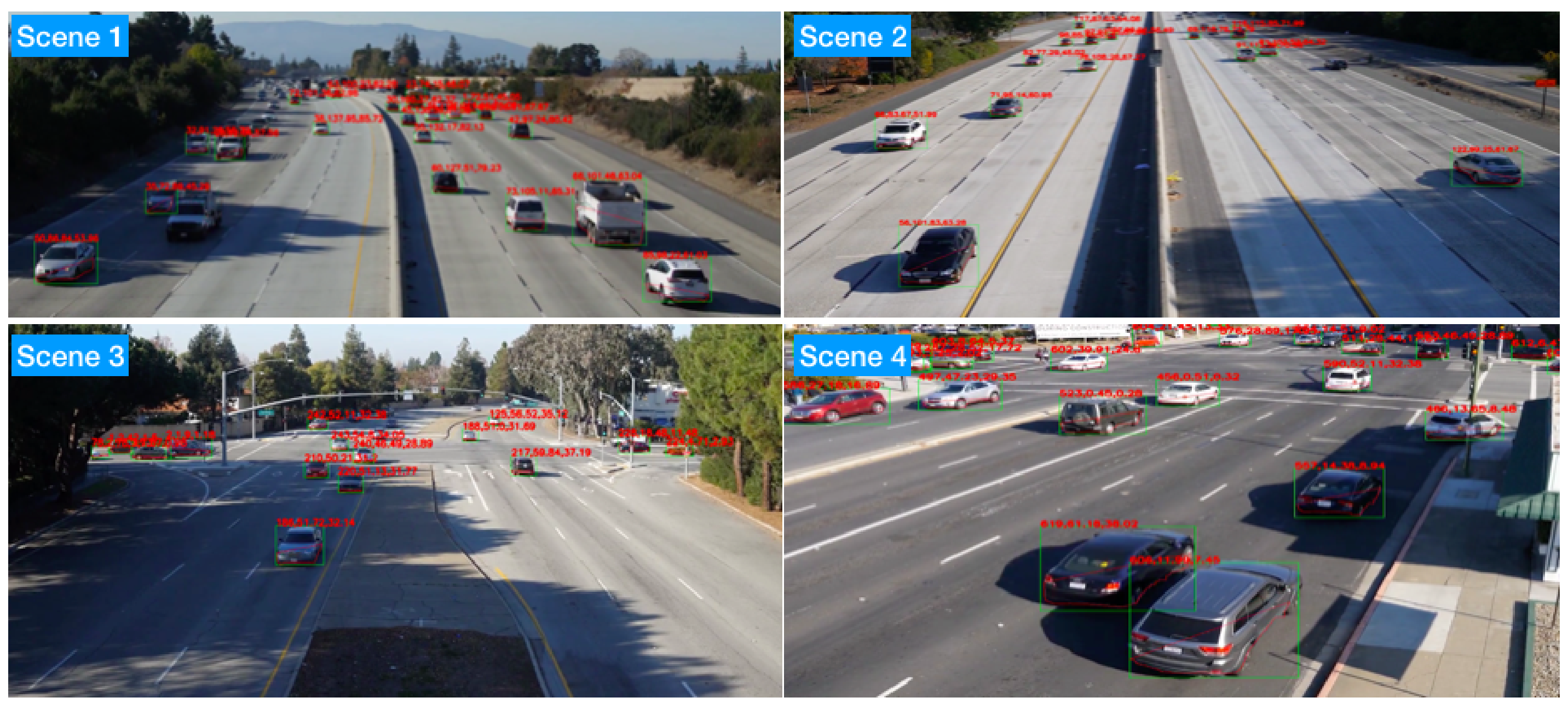

Figure 1: Four scenes in track1

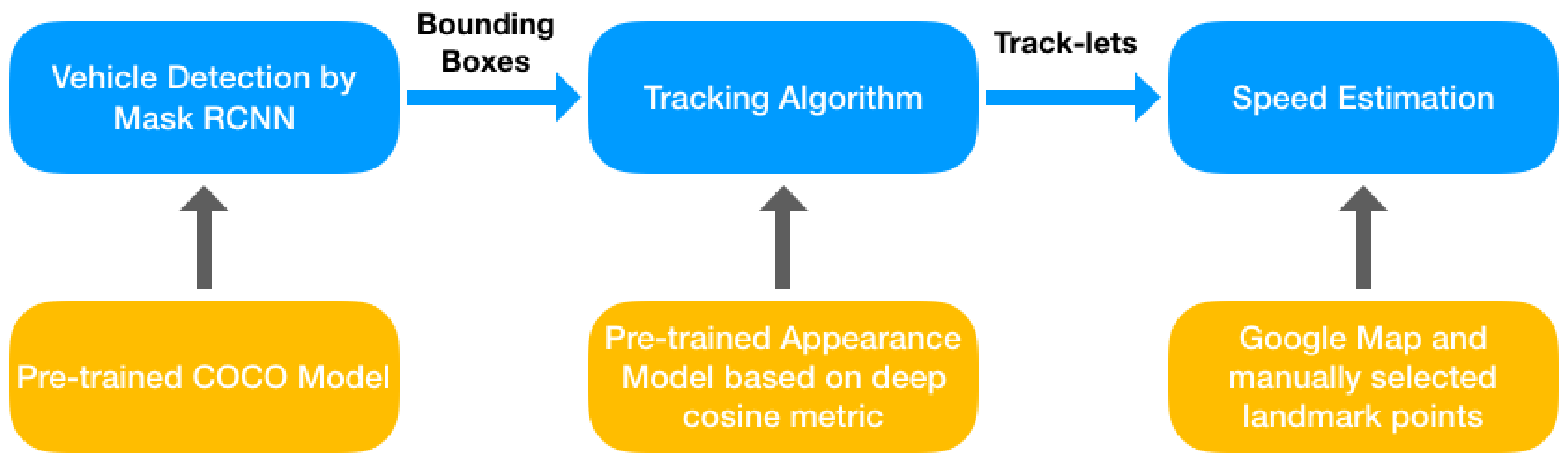
Vehicle Detection by Mask RCNN
Bounding Boxes
Tracking Algorithm
Track-lets
Speed Estimation
Pre-trained COCO Model
Pre-trained Appearance Model based on deep cosine metric
Google Map and manually selected landmark points

Figure 2: The system for real-time speed estimation

Simple Online and Realtime Tracking with a Deep Association Metric

The tracking part adopts the conventional single hypothesis tracking methodology with recursive Kalman filtering and frame-by-frame data association [2]. Figure 3 visualize the process of a single frame association between track-lets and detections.

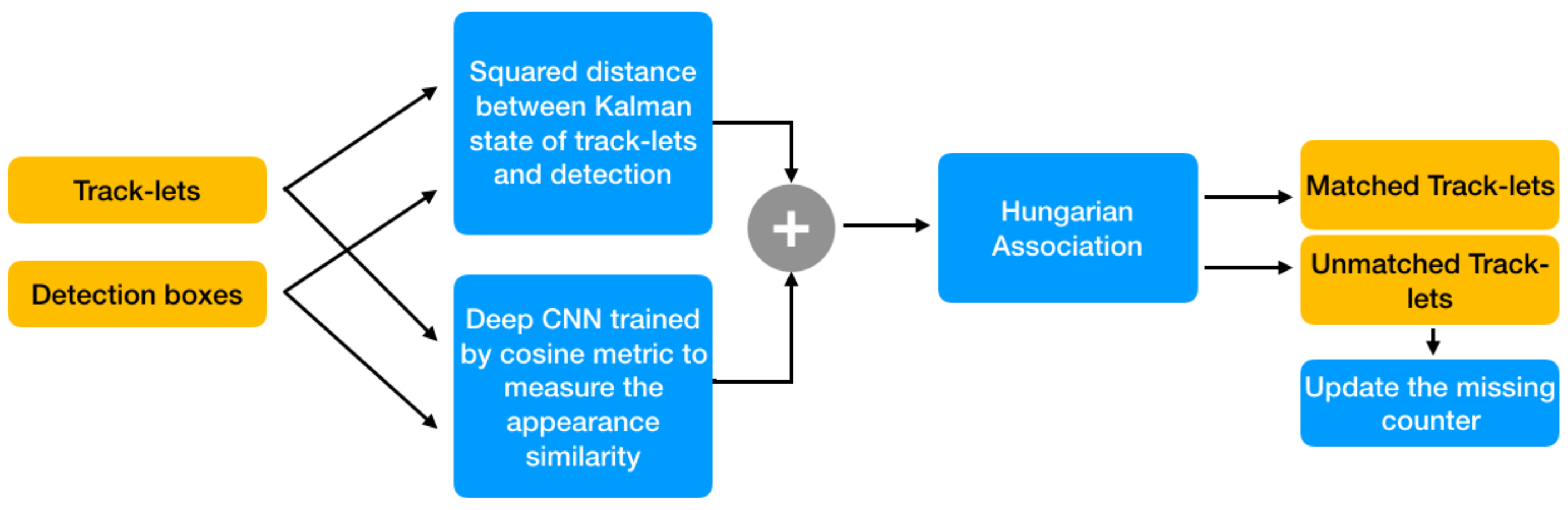
Track-lets
Detection boxes
Squared distance between Kalman state of track-lets and detection
Deep CNN trained by cosine metric to measure the appearance similarity
+
Hungarian Association
Matched Track-lets
Unmatched Track-lets
Update the missing counter

Figure 3: Matching cascade

Results

Track 1 is evaluated on two aspects: detection rate and root of mean square error (RMSE) of speed estimation. According to the submissions, our detection rate is 100% and the overall RMSE is about 7.97 miles/hour. We also evaluate the output of different locations separately. In terms of mean speed value, our result is in a reasonable range. Detailed information is shown in Table 1.

Location	Mean	Median	RMSE
1	66.64	67.31	9.61
2	60.15	61.20	10.20
3	11.54	5.27	6.50
4	9.27	5.48	5.50
Overall	-	-	7.97

Table 1: Track 1 speed estimation (mile/hour).

Track 2: Anomaly Detection

In this challenge, track 2 requires us to detect the initial timestamp of car crashes or stalled cars. To detect anomaly over a whole video can be decomposed into making detection on overlapped short video clips. For each video clip, intensive global features are first extracted and then SVM classification is implemented to identify whether an anomaly is happening in this video clip. The full pipeline is shown in Figure 4 where we combine two different types of video information together to make estimations. The first branch focuses on visual features. It use VGG network to extract 1×4096 vector from each frame and apply PCA to further shrink into a "shorter" 1×256 vector. Then it applies encoding methods to aggregate 50 frame-based vector together and achieve the final global representation of the short video clip. Similarly, in the second branch, it adopts feature-then-encoding methodology but it focuses on temporal/motion features.

It should be noted that the final output of classification can be overlapping, as multiple events can happen simultaneously. The models are trained on a private traffic dataset.

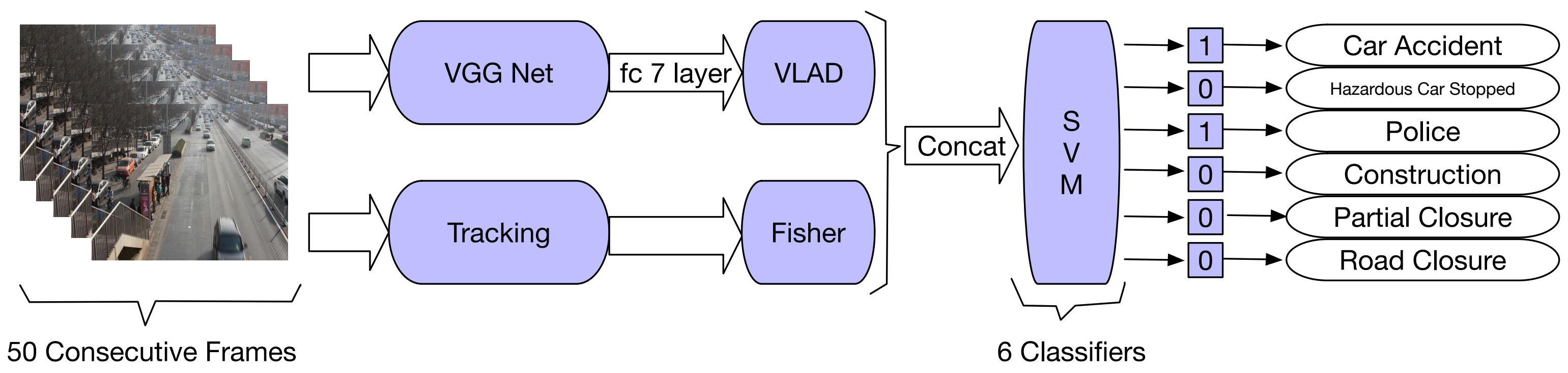

Figure 4: Full pipeline of Anomaly Detection

Tracking Features

To capture the motion information, we apply sparse tracking and improved dense trajectory (iDT) feature descriptors. They are both implemented by comparing the difference between the consecutive two frames. iDT is an aggregation of many motion descriptor and robust to shaky video.

Feature Encoding Methods

The encoding methods which we adopt to find a intensive global representation of the input 50-frame sequence consists of fisher vector (FV) and vector of locally aggregated descriptor (VLAD). The formulation of FV is given in Equation 1 and 2,

$$u_k = \frac{1}{N\sqrt{\pi_k}} \sum_{i=1}^{N} q_{ki}(\frac{x_i - \mu_k}{\alpha_k}) \tag{1}$$

$$v_k = \frac{1}{N\sqrt{2\pi_k}} \sum_{i=1}^{N} q_{ki}[(\frac{x_i - \mu_k}{\alpha_k})^2 - 1] \tag{2}$$

where N is the number of data points (in our case, N = 50 as the input contains 50 frames), q_{ki} is the posterior probability of i-th data in k-th cluster, x_i is the vector we intend to encode. By concatenating v_k and u_k, we form our Fisher vector with the length of $2D'K$ where D' is the reduced dimensionality after applying PCA.

$$u_k = \sum_{i:NN(x_i)=c_k} (x_i - c_k)\alpha_k) \tag{3}$$

VLAD descriptor is given as the concatenation of the sum of the distance between each cluster centroid and the frame vectors which belongs to this cluster. The formulation of VLAD is given in Equation 3. VLAD has been treated as the simplified version FV. The key difference is that VLAD applies a simple K-mean clustering methods while FV uses GMM clustering.

In this challenge, we set the number of cluster $K = 64$ for the first branch in Figure 4, therefore the output vector is $2 \times 64 \times 256$ if using FV or 64×256 if using VLAD. For the second branch, we set $K = 256$. Similarly, the output is $2 \times 256 \times 256$ encoded by FV.

Results

Track 2 is evaluated based on the detection performance which is based on F-1 score and event time difference judged by RMSE. Our goal is trying to predict the timestamp as accurate as possible without compromising too much on F-1 score. It is worth mentioning that we also perform a threshold cutting on the confidence score on the outputs so that we can get as better F-1 score as possible. The final F-1 score and RMSE in seconds can be seen in Table 2.

We think that part of reason of this high RMSE can be that we pay too much attention on the end timestamp of an anomaly event in training. Thus, the start timestamp cannot be fully trained. In addition, the intermediate models are trained on a private traffic dataset and the direct transfer learning is not robust to the potential variations between different datasets.

F1	RMSE (seconds)
0.7692	214.2712

Table 2: Track 2 anomaly detection.

Conclusion

- For track 1, we develop a pipeline to estimate vehicle speed. Due to the good accuracy in vehicle detection/tracking, our method achieves 100% detection rate, and the average estimation error is about 7.97 mile/hour. Our rank in track 1 is 5/13.
- For track 2, we develop an optical flow based method. By using the information from optical flow, we can include the temporal relationship between frames. We obtain 0.7692 F1 score and about 214 second estimation error in the contest. We apply a threshold cut on the confidence score to eliminate the unqualified judgment.

References

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[2] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. *arXiv preprint arXiv:1703.07402*, 2017.

Acknowledgements

We would like to appreciate this opportunity provided by NVIDIA AI City challenge organizing committee.