August 2017

# Robust classification of city roadway objects for traffic related applications

Niveditha Bhandary
*San Jose State University*

Charles MacKay
*San Jose State University*

Alex Richards
*San Jose State University*

Ji Tong
*San Jose State University*

David Anastasiu
*San Jose State University*, danastasiu@scu.edu

## Recommended Citation

Niveditha Bhandary, Charles MacKay, Alex Richards, Ji Tong, and David Anastasiu. "Robust classification of city roadway objects for traffic related applications" *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)* (2017). https://doi.org/10.1109/UIC-ATC.2017.8397668

# Robust Classification of City Roadway Objects for Traffic Related Applications

Niveditha Bhandary [1], Charles MacKay [2], Alex Richards [1], Ji Tong [1] and David C. Anastasiu [1,*]
{niveditha.bhandary, charles.mackay, alexander.richards, ji.tong, david.anastasiu}@sjsu.edu
[1]College of Engineering, San José State University, San José, CA 95192
[2]College of Science, San José State University, San José, CA 95192
[*]Corresponding author

***Abstract*—The increasing prevalence of video data, particularly from traffic and surveillance cameras, is accompanied by a growing need for improved object detection, tracking, and classification techniques. In order to encourage development in this area, the AI City Challenge, sponsored by IEEE Smart World and NVIDIA, cultivated a competitive environment in which teams from all over the world sought to demonstrate the effectiveness of their models after training and testing on a common dataset of 114,766 unique traffic camera keyframes. Models were constructed for two distinct purposes; track 1 designs addressed object detection, localization and classification, while track 2 designs aimed to produce novel approaches towards traffic related application development.**

**Careful tuning of the Darknet framework's YOLO (You Only Look Once) architecture allowed us to achieve 2nd place scores in track 1 of the competition. Our model was able to achieve inference beyond 50 frames per second (FPS) when performing on the NVIDIA DGX-1's Tesla P100 GPU and up to 37 FPS on a NVIDIA GTX 1070 GPU. However, the NVIDIA Jetson TX2 edge device had a lackluster 2 FPS inference speed. To produce truly competitive automated traffic control systems, either more preferment edge device hardware or revolutionary neural network architectures are required. While our track 2 model approach demonstrated that it is reasonable to obtain useful traffic related metrics without the use of the region proposal networks and classification methods utilized in other models typically associated with traffic control systems.**

***Index Terms*—Object detection, tracking, classification, autonomous traffic control systems, smart city**

## I. Introduction

Video is a powerful medium for conveying information and data is plentiful wherever there are cameras. From dash, body, and traffic cams, to YouTube and other social media sites, there is no shortage of video data. Interesting applications that exploit this are ripe for development. One particularly compelling domain where video analytics has tremendous potential is in automated traffic control and public safety systems. The mere presence of automated traffic control systems, like red-light and speed cameras, has been shown to have a positive effect on the the reduction of traffic violations. A worldwide analysis of 28 studies has concluded that such systems have reduced crashes by 8% to 50% [12]. Carnis and Blais showed in their research that the initial response to France's implementation of their automated speed enforcement program (ASEP) was a 21% reduction in fatal car accidents and a 26% reduction in non-fatal car accidents [1].

In traffic control systems, recall and precision metrics are used to gauge the utility of these systems. Recall, in this case, measures the the number of true traffic infringements that were detected by the system. Improving this metric is crucial as revenue brought in by ticketing must be high enough to justify the cost of installing and maintaining the system. Revenue brought in by ticketing must be high enough to justify the cost of the system. Precision, on the other hand, measures the ability of the system to make correct assessments and must be maximized to ensure customer satisfaction. As part of their study, Carnis and Blais found that only 70% of violations detected were sanctioned. While autonomous violation detection increases public safety, the less than ideal precision of this specific system mandates that humans are still necessary to authorize the ticketing process. Maximizing precision not only reduces the manpower required to validate detected violations but is a requirement in progressing towards fully autonomous traffic control systems.

In this paper, we will provide an overview of techniques used to address the problems of object detection, tracking, and classification, and describe methods methods we developed for these tasks as part of the 2017 IEEE Smart World NVIDIA AI City Challenge [7]. Our group constructed models using the Darknet [8] and Keras [2] frameworks utilizing pre-trained weights for convolutional layers and fitting the remaining parameters to perform optimally on two independent tasks. Our track 1 models were designed to perform under the task of object localization and classification while our track 2 models were designed to predict traffic density using a regression network postfixed to a feature map. The remainder of our article is organized as follows. In Section II, we discuss related works. Section III introduces our approaches towards pre-processing, architecture design, and hyper-parameter tuning of deep-learning models for the challenge tasks. We present results of our models in Section IV and make considerations for future work in developing improved models and novel applications that utilize the state-of-the-art in object detection in Section V. Finally, we conclude the article in Section VI.

## II. Related Works

There are a multitude of approaches used to detect objects in an image. Early methods focused on feature descriptors like histogram of oriented gradients (HOG) [6] and scale-invariant feature transform (SIFT) [5]. HOG is one of the most popular computer vision techniques; put simply, it groups together pixels into cells which are grouped into adjacent blocks. Each cell votes (bins) on a direction of the pixel gradient and the bin magnitude according to the votes of the adjacent cells in a block are pooled to determine the final gradient. The pattern of gradients across an entire image is used to localize objects. SIFT works similarly to HOG, but gradient detection is more localized and does not perform contrast normalization as it does in HOG. Relative to newer deep learning-based techniques, these approaches are slow and do not perform as well.

One of the first deep-learning approaches to object detection was introduced by Ross Girshick et al. in the form of Regions with Convolutional Neural Networks (R-CNN) [3]. This early design details a *region proposal* system in which images along with suggested bounding boxes, provided via an external process like selective search, are fed into a convolutional neural network. A support vector machine (SVM) uses the activated features detected by the CNN and the bounding boxes to classify whether or not the subjects that may be contained within a given bounding box are indeed objects of interest. Finally, a regression layer tightens the bounding boxes around identified objects. R-CNN outperforms HOG with a mean average precision (mAP) of 54% on the VOC2010 dataset compared to HOG's mAP of 33%. The drawbacks of this method include the need of an external system to propose bounding boxes and the bottleneck introduced by the need for repeated forward passes through the CNN for each proposed bounding box. In addition, the system is difficult to train due to the use of disparate components.

Girshick et al. improved upon R-CNN with Fast R-CNN [4] by introducing RoiPool (Region of Interest Pooling) which pools together features shared across multiple region proposals. This advancement greatly speeds up inference by removing the necessity to make more than one forward pass through the network for each image. In addition, the external SVM classifier and regression model are integrated with the CNN in the form of a softmax layer and regression network, respectively. This consolidation of components makes it possible to train the network from beginning to end in one simple process. On the VOC07 dataset, Fast R-CNN trains 9 times faster than R-CNN and is 213 times faster at test time. Fast R-CNN also shows a 5.9% increase over R-CNN achieving a mAP of 68.8% on the VOC2010 dataset. However, region proposal still depends on an external selective search process, making this the glaring bottleneck of Fast R-CNN.

A final improvement, Faster R-CNN [10] from Girshick et al., uses the insight that the CNN of Fast R-CNN discovers all the features captured by selective search. A region proposal network placed after the CNN's feature map eliminates the need for proposed regions to be calculated in advance. Faster R-CNN continues the trend of incremental improvements on training and testing speeds and mAP scores. Most importantly, Faster R-CNN can be trained in an end to end fashion without the need for externally computed region proposals.

The most recent and preferment models that operate in this domain simplify the architecture advances that Girshick et al. made in order to produce models that execute forward passes extremely efficiently. Perhaps the best performing such model is YOLO [9]. This model's architecture utilizes a modified version of the GoogLeNet [13] model, called Extraction, which features 21 convolutional layers with 1x1 and 3x3 filters and max pooling. Although the Extraction model's predictions are less accurate than VGG-16 [11], a forward pass through the Extraction network requires only 8.52 billion floating point operations instead of VGG-16's 30.69 billion operations. This is in part responsible for YOLO's increased efficiency. YOLO's weakest point is that it predicts using a 13x13 feature map, which may be too large of a grid for the detection of small objects or objects in low resolution images.

## III. Methods

### A. Materials

Our models were fit using keyframes taken at one second intervals from several cameras mounted at traffic intersections. Cameras were positioned at multiple intersections in the following three areas: Lincoln, NE, Virginia Beach, VA, and Silicon Valley, CA. More than 75 hours of footage was captured during daytime as well as nighttime. Videos were taken at 30 frames per second at either a resolution of 480x720 or 1080x1920. Then, keyframes were extracted at 1 second intervals from each video. Over 150 volunteers, including the participants of this competition, collaboratively annotated these keyframes over the course of two weeks. While more than the 150,000 keyframes were annotated with over 1.4M annotations, quality filtering left a total of 114,760 annotated keyframes across two datasets, including 100,372 taken from the 1080p footage and 14,388 from the 480p footage. The two datasets were split into training, validation, and testing subsets by the challenge organizers. The 1080p footage contained 59,482 training keyframes, 19,272 validation keyframes, and 21,618 test keyframes. The 480p dataset had 7,640 training keyframes, 3,376 validation keyframes, and 3,372 test keyframes. Additionally, a downsampled version of the 1080p dataset was provided at a resolution of 540x960. We performed no data pre-processing prior to training our track 1 models. However, for the track 2 models we normalized pixel values to fall in the range between 0 and 1 inclusive.

### B. Procedure

**Track 1**

Training accurate models on the AI City challenge datasets was straightforward due to the simplicity of using Darknet YOLO "out of the box". This allowed us to quickly begin training and producing inference results on the dataset. YOLO object detection is framed as a regression problem. It divides the image into an $SxS$ grid. We adjusted the filters of the last convolutional layer to accommodate for the number of classes. Specifically, $num\_filters = (num\_classes + num\_coords[x, y, h, w] + 1) \times num = (15 + 5) \times 5$. If the center of an object falls into a grid cell that cell is responsible for detecting this object. Each grid cell predicts $B$ bounding

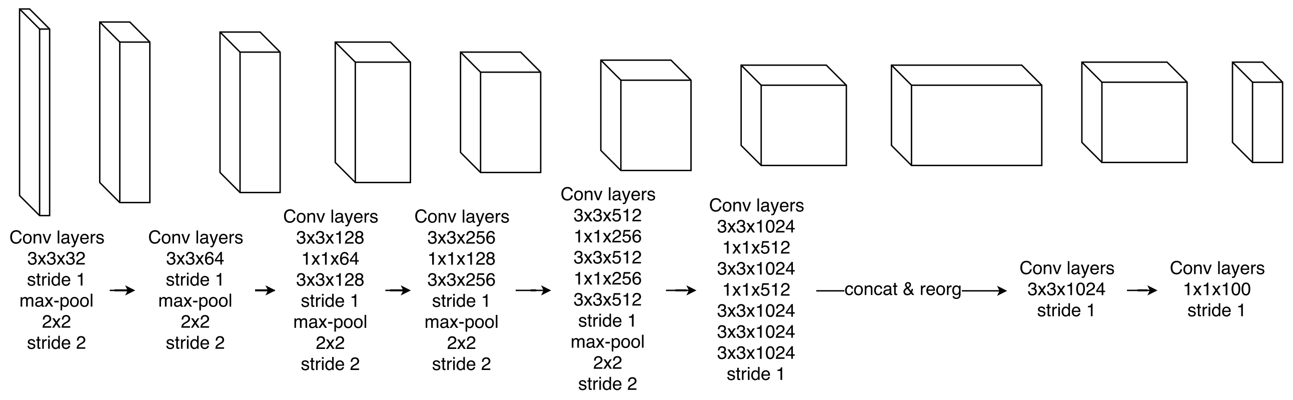
Conv layers 3x3x32 stride 1 max-pool 2x2 stride 2
Conv layers 3x3x64 stride 1 max-pool 2x2 stride 2
Conv layers 3x3x128 1x1x64 3x3x128 stride 1 max-pool 2x2 stride 2
Conv layers 3x3x256 1x1x128 3x3x256 stride 1 max-pool 2x2 stride 2
Conv layers 3x3x512 1x1x256 3x3x512 1x1x256 3x3x512 stride 1 max-pool 2x2 stride 2
Conv layers 3x3x1024 1x1x512 3x3x1024 1x1x512 3x3x1024 3x3x1024 3x3x1024 stride 1
concat & reorg
Conv layers 3x3x1024 stride 1
Conv layers 1x1x100 stride 1

Fig. 1. YOLO network architecture.

boxes and confidence scores for those boxes. If no object exists in that cell, the confidence score for that class is 0. We only predict one set of class probabilities per grid cell, regardless of the number of boxes $B$. We modified the input and output layers of the YOLOv2 network architecture to accommodate the input size of the AI City datasets, and left the middle layers unchanged. The architecture of the neural net is shown in Figure 1. We utilized transfer learning to speed up the training process by applying pre-trained convolutional weights from the Darknet19_448 model available in YOLOv2. This is an offshoot of the GoogLeNet model, trained on ImageNet with top-5 accuracy of 93.5% [9]. The ImageNet dataset contains a significant number of vehicle classes, and loading these pre-trained weights allowed us to quickly produce an accurate model for the AI city challenge datasets. We retrained the network from end-to-end after loading the convolutional weights for $30,000$ to $45,000$ iterations. Snapshots of weights were taken every $5,000$ iterations and we periodically evaluated the performance of the models.

The AIC480 model converged to a desirable loss after $30,000$ iterations. Evaluation of this model showed good localization and accuracy. Due to the low resolution of AIC480 dataset images, the $SxS$ grid ($13 \times 13$) was too large of a division to properly evaluate the objects in the center of the grid. This caused objects such as bicyclists and pedestrians to be undetected. This is a downfall of YOLO, as other groups using this framework have noticed. The mAP of the bicycle class for all teams was 0.098 while our score was 0.08. The AIC540 model was trained in the same way as the AIC480 model, only changing the first layer of the network to adjust for the resolution change. This model was trained for $45,000$ iterations and, after evaluation, was found to have good performance. The default threshold value for the minimum confidence of a bounding box in YOLO is 24%. Lowering this threshold allowed us to increase our recall score significantly and without many false positives. Eventually taking the threshold down to 3-4% allowed us to maximize our recall and we achieved beyond 90% recall of all classes and up to 95% recall for the SUV class. We successfully trained a model for the AIC480 and AIC540 datasets. Training a model for the AIC1080 dataset was challenging due to the sheer size of the images. We had to reduce the batch size to 1 image during training. This caused our model's training time to be significantly slower and our loss rate unable to converge. We fell back on our AIC540 model to run inference on the AIC1080 dataset since they are identical besides having different resolutions. However, since we did not re-train the first layer for the AIC1080 model, our scores were much lower across the board. We developed a work-around to this problem by downsizing the AIC1080 images to 540x960, and then running inference on the downsampled image with our AIC540 model. Finally, we upsampled the resulting bounding boxes to fit the AIC1080 image. This step adds to the real-time processing requirements by having to downscale the input source image, but performed well in practice with regards to effectiveness. In particular, using this model increased the AIC1080 mAP from 0.28 to 0.470.

**Track 2**

The proposed utility of our track 2 model was to predict the traffic density at a given keyframe. Instead of proposing regions, regressing on object localizations, and then classifying these bounding boxes, our model simply takes in the activated features from the convolutional feature map and feeds them into a simple regression network. This model was constructed using the Keras framework. Pre-trained VGG-16 weights comprised the feature detecting convolutional layers of our model. VGG-16 has 13 convolutional layers (arranged in 5 blocks) with 3x3 and 2x2 filters and max pooling. A diagram of the network's architecture is visible in Figure 2.

We attempted several architectures that were postfixed to the VGG-16 feature map. All configurations utilized either one or two hidden layers, rectified linear unit (RELU) activation functions, and 30%, 50%, or 75% dropout applied to these hidden layers. The output layer consisted of either a single linear unit or 14 linear units. The architectures with 14 output units regressed on the counts for all 14 classes in our dataset while the single output unit architectures were trained on labels produced by aggregating all vehicle classes into a single count.

Training on our models used only the 540x960 resolution dataset. Though we did experiment with training several

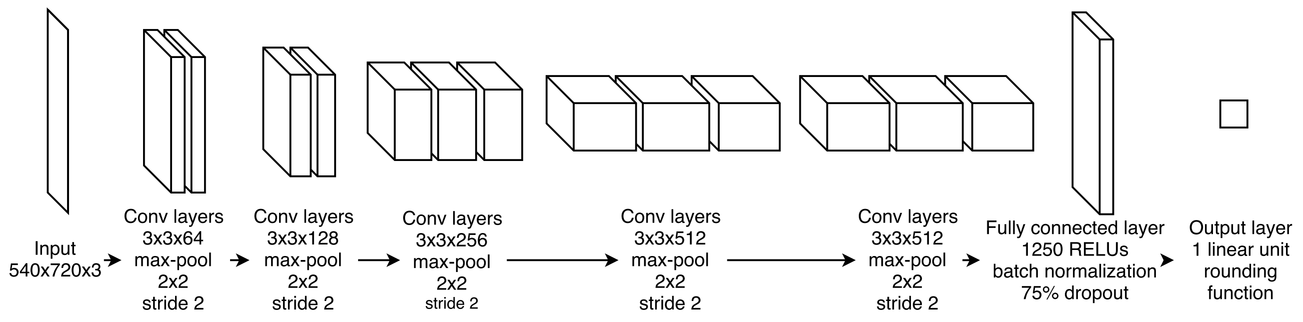

Fig. 2. VGG-16 + regression network architecture.

variants of our models on the 480x720, this was mainly done to configure the network architecture initially and test our methods. We selected a mean-squared-error loss (MSE) function to train our models and evaluated their performance with a custom mean-absolute-error (MAE) function which rounded the real values output by the model to the nearest integer. In retrospect, the loss and evaluation functions we chose were not well suited for the regression task. These functions underemphasize the significance of differences in true and predicted counts for low density images while overemphasizing this significance in high density images. Our multi-class models had deceptively good MAE scores; despite having low error, there was high bias towards zero as most classes in most images have a zero count. Time constraints prevented our group from pursuing proper pre-processing techniques for the multi-class models. Instead, we focused most of our time on the single-class models. We found that our models badly overfit if we used more than one hidden layer after the feature map. Our model achieved the best evaluation scores with relatively high dropout and more units in the hidden layer. Due to hardware limitations, we were unable to evaluate a model with more than 1250 hidden units.

## IV. Results

### Track 1

We placed 2nd overall in track 1 of the challenge by having the 2nd and 3rd best mAP scores on the AIC540 and AIC480 datasets, respectively. The challenge organizers evaluated track 1 by averaging the maximum mAP of the AIC540 and AIC1080 datasets with the mAP of the AIC480 dataset, since the AIC540 and AIC1080 datasets were in fact the same videos with different resolutions. Teams were given an additional two weeks after the conference to further improve their models. We used this time to fine tune our thresholding and filtering parameters. Specifically we lowered the threshold criteria for minimum confidence of the detected bounding boxes, as well as their associated class probabilities. This resulted in an increased recall score, with the trade off of an increased number of false positives. However, this improved the model's performance overall, and we were able to maintain our second place rank. Table I shows the overall and per-class mAP scores for the best models we trained on each of the three datasets.

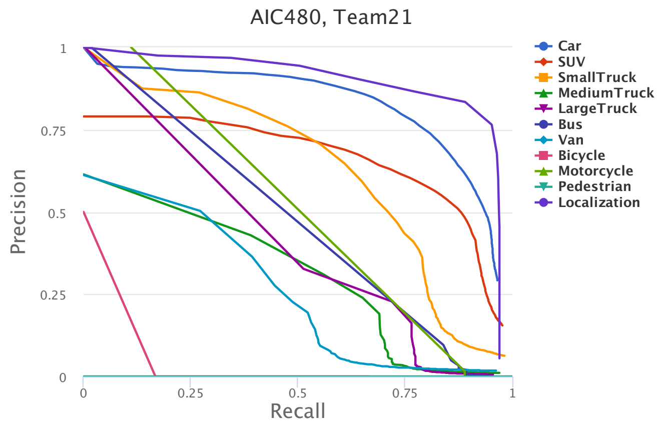

Fig. 3. Track 1 model performance on AIC480 dataset.

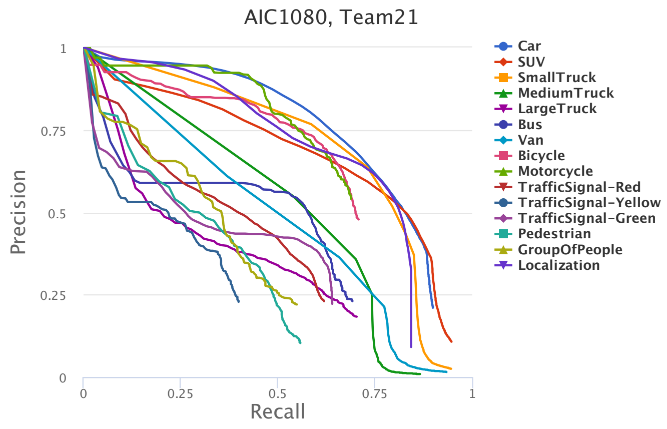

Fig. 4. Track 1 model performance on AIC1080 dataset.

Figures 3 and 4 show our precision/recall curves for our AIC480 and AIC1080 models. Our models were able to detect standard-sized vehicles with ease. Many small objects were not annotated by some of the volunteers. As such, the organizers ignored bounding boxes smaller than 30x30 during evaluation. This caused traffic lights and other small objects to be discarded, which negatively affected our result with respect to these classes. From our own evaluation on the validation

TABLE I
PERFORMANCE ON DATASETS

| **Dataset** | mAP | Car | SUV | SmTruck | MdTruck | LgTruck | Bus | Van | Bicycle | Traffic-R | Traffic-Y | Traffic-G | Ped | Loc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AIC480 | 0.44 | 0.82 | 0.66 | 0.62 | 0.32 | 0.31 | 0.52 | 0.26 | 0.08 | N/A | N/A | N/A | 0 | 0.89 |
| AIC540 | 0.38 | 0.74 | 0.69 | 0.71 | 0.48 | 0.32 | 0.41 | 0.48 | 0.28 | 0 | 0 | 0 | 0.01 | 0.77 |
| AIC1080 | 0.47 | 0.73 | 0.67 | 0.68 | 0.46 | 0.32 | 0.4 | 0.45 | 0.58 | 0.36 | 0.2 | 0.33 | 0.29 | 0.67 |

dataset, we would have achieved an AP of 0.65, 0.57, and 0.60 for Green, Red, and Yellow traffic lights, respectively, for the AIC540 model if these small bounding boxes had not been discarded. Similarly, the bicycle class had a score of 0.65 if considering all objects, as opposed to 0.27 after discarding small boxes.

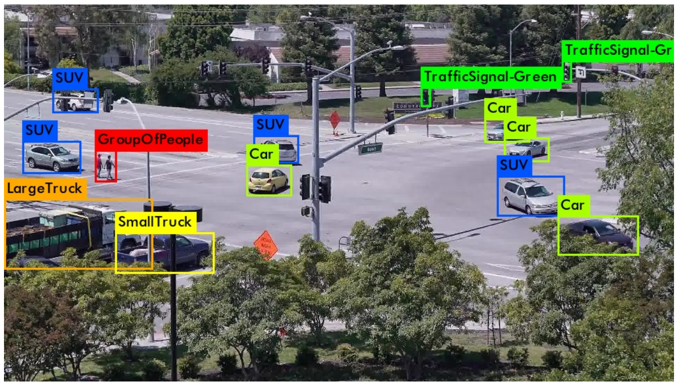

Fig. 5. A variety of different classes detected.

Our models performed well for the majority of the classes. The greatest accuracy for all of our models was shown on the car, small truck, and SUV classes, since these were the most common and easily distinguishable objects. Our AIC540 model achieved average precision scores of 0.74, 0.69, and 0.71 for these classes respectively. The models often mislabeled minivans as SUVs, as shown in Figure 5. Minivans have a similar shape and size; especially those with rounded edges. These similarities made it difficult for the neural net to accurately classify them. Larger freight vans that were boxy in shape were detected with greater accuracy. We suspect precision could be increased with van subclasses. The model performed well on the truck sub-classes because it was divided into 3 specific categories, small, med, and large. A small, medium, and large van class could have increased accuracy.

The network had relatively poor precision on large trucks however, as exemplified in Figure 7. The figure shows ground truth bounding boxes and labels in red and predicted ones in blue. Large trucks spanned multiple grids and the model at times split them into two trucks. Also, when a large truck drove through an intersection, it was at times split into two by a light pole, resulting in the model detecting 2 large trucks. This made the intersection over union of the predicted and ground truth bounding boxes very low.

The labeling of the dataset was sometimes ambiguous. Participants were inconsistent with the labeling of vans, SUVs and trucks. One such example can be seen in the far-left

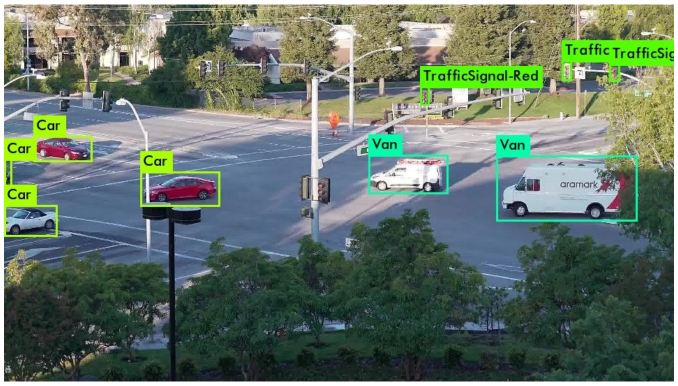

Fig. 6. Two very different but accurately labeled vans.

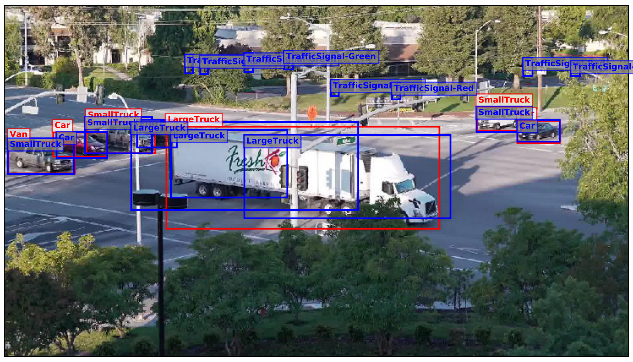

Fig. 7. Model performance on AIC540 dataset; ground truth colored red, predictions colored blue.

section of Figure 7, where an annotator clearly mis-labeled a small truck as a van. Some keyframes were completely skipped because participants were unsure what to call these objects. More consistent and higher quality annotations would have allowed our model to be trained with higher accuracy.

During the conference, we demonstrated the model's performance on the NVIDIA Jetson TX2 with live video from a USB camera and downloaded traffic camera from YouTube clips. This allowed attendees to observe the performance of the models on data vastly different from the training and validation sets, in a real-world setting running on an edge device connected to a live video stream. We noticed the AIC540 model performed brilliantly when shown images of the same view angle and height as the training data, but struggled to detect objects when the camera perspective deviated greatly from the training set. This demonstration showed the viability of an edge device to perform inference on live video despite low

FPS.

**Track 2**

Our best performing track 2 model (without considering the biased multi-class models) had a MAE of 2.39. As mentioned before, this metric is less than ideal as it under-emphasizes the significances of differences between true values and predicted values when the traffic density is low while overemphasizing the difference in high traffic density keyframes. Figure 8 exemplifies good performance with this model. Also notable in this figure is the potential impact that mislabeling may have had on all models trained using this dataset. This particular image shows 10 cars present despite the annotators labeling only 9.

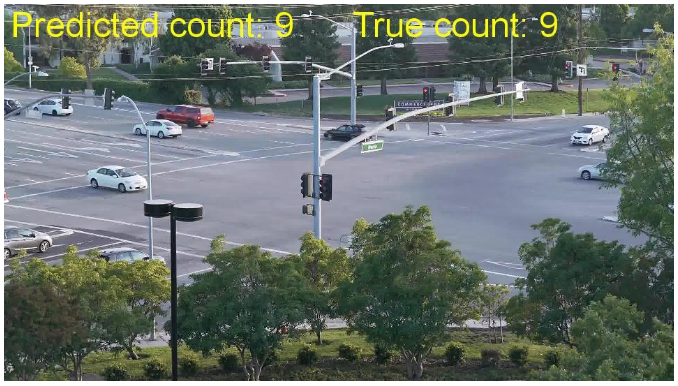

Fig. 8. Example of good performance from track 2 predictions.

The inference speed of this model was a slow 1 FPS on the Titan X Pascal GPU architecture. If inference was performed in batches of 11 frames, the frame rate increased to 3.6. This result stressed the importance of framework selection when designing models for applications that depend upon real-time inference. Keras, which emphasize flexibility and rapid development, is sub-optimal for deployment purposes. Other frameworks may have performed better in this scenario.

## V. Future Work

We will continue to improve our models for use in traffic related applications by training on more diverse datasets with many more hours of video and with special emphasis on high-definition footage. Due to the time constraints of the competition, we did not get the opportunity to really experiment with data pre-processing and tweaking our model training procedures. The need for data generation was evident because our models failed to detect images well when the camera perspective deviated too far from the one in the training images. We will apply image rotations, flips, scaling, as well as hue, saturation, contrast, and brightness manipulations in order to greatly increase the quantity of learned models. Our individual models performed reasonably well when applied to test data of the same resolution as the data the respective model was trained on, but performance fell when trying to mix training and test resolutions. We would like to produce a more robust model that could perform well when used on cameras of varying resolutions. Lastly, we would like to continue exploring a multitude of novel use cases for this technology.

## VI. Conclusion

The richness that video data provides highlights the importance of advancing the state-of-the-art in object detection, classification and tracking for real-time applications. There has been a steady progression of image detection techniques beginning with feature descriptors like HOG and, more recently, deep network-based approaches like Faster R-CNN and YOLO. YOLO provides extremely fast inference speed with slight compromise in accuracy, especially at lower resolutions and with smaller objects. While real-time inference is possible, applications that utilize edge devices still require improvements in either the architecture's design or edge device's hardware. Our track 1 model achieved good performance without much consideration for data pre-processing which could have potentially produced a much more robust model. Also, we did not consider custom convolutional layer training in our track 2 model, which could have greatly reduced the number of parameters and further increased inference efficiency.

## Acknowledgements

The authors would like to thank Viraj Nilakh and Shruti Padmanabhan for their invaluable contribution to this project. We would also like to thank NVIDIA Corporation for the donation of the Titan X Pascal GPU and Jetson TX2 edge computing device, as well as DGX1 resources they made available for the duration of the challenge, all of which were vital for this research.

## References

[1] Laurent Carnis and Etienne Blais. An assessment of the safety effects of the french speed camera program. *Accident Analysis & Prevention*, 51:301–309, 2013.

[2] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, June 2014.

[4] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[5] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[6] R.K. McConnell. Method of and apparatus for pattern recognition, January 28 1986. US Patent 4,567,610.

[7] Milind Naphade, David C. Anastasiu, Anuj Sharma, Vamsi Jagrlamudi, Hyeran Jeon, Kaikai Liu, Ming-Ching Chang, Siwei Lyu, and Zeyu Gao. The nvidia ai city challenge. In *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, SmartWorld'17, Piscataway, NJ, USA, 2017. IEEE.

[8] Joseph Redmon. Darknet, 2013.

[9] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[10] S. Ren, K. He, R. Girshick, and J. Sun. faster r-cnn: towards real-time object detection with region proposal networks. *ieee transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, June 2017.

[11] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[12] American Traffic Solutions, 2014.

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.