ResNet-based Vehicle Classification and Localization in Traffic Surveillance Systems

Heechul Jung† Min-Kook Choi‡ Jihun Jung Jin-Hee Lee Soon Kwon Woo Young Jung*
DGIST, Daegu, Republic of Korea
{heechul, mkchoi, jihun.jung, jhlee07, soonyk, wyjung}@dgist.ac.kr

Abstract

In this paper, we present deep residual network (ResNet)-based vehicle classification and localization methods using real traffic surveillance recordings. We utilize a MIOvision traffic dataset, which comprises 11 categories including a variety of vehicles, such as bicycle, bus, car, motorcycle, and so on. To improve the classification performance, we exploit a technique called joint fine-tuning (JF). In addition, we propose a dropping convolutional neural network (DropCNN) method to create a synergy effect with the JF. For the localization, we implement basic concepts of state-of-the-art region based detector combined with a backbone convolutional feature extractor using 50 and 101 layers of residual networks and ensemble them into a single model. Finally, we achieved the highest accuracy in both classification and localization tasks using the dataset among several state-of-the-art methods, including VGG16, AlexNet, and ResNet50 for the classification, and YOLO Faster R-CNN, and SSD for the localization reported on the website.

1. Introduction

The development of intelligent traffic surveillance systems has emerged as an important issue in recent years. One of the most important functions in intelligent traffic surveillance systems is analyzing and extracting useful information from recordings. In particular, a technique that classifies and localizes vehicles or pedestrians can represent a basic procedure for traffic surveillance analysis.

The MIOvision traffic camera dataset (MIO-TCD) that was recently released is a representative large-scale image dataset for vehicle classification and localization. The images in the dataset have been taken in real traffic surveillance environments at different times of the day and different periods of the year by thousands of traffic cameras. Our goal in this research is to develop classification and localization algorithms that surpass the accuracy of state-of-the-art methods for the MIO-TCD dataset.

To achieve our goal, we adopt a deep residual network (ResNet), which is the most popular deep learning technique; it allows training of an ultra deep convolutional neural network (CNN) as a baseline network using identity mapping [4]. Thanks to the ResNet, practitioners can easily achieve good accuracy in image classification and localization tasks by simply stacking many layers [4]. Based on the technique, we obtain a 97.95% classification rate on average for the vehicle classification task and 79.24% in terms of mean average precision (mAP) for the vehicle localization task. Our methods show the best accuracy for each task among the other state-of-the-art approaches reported on the MIO-TCD website. [1]

The rest of our paper is organized as follows: We present training details of our approach for the vehicle classification task and propose a joint fine-tuning (JF) method with a dropping CNN (DropCNN) for the model ensemble in Section 2. For the vehicle localization task, we give implementation details of region-based, fully convolutional networks (R-FCN) [1] in Section 3. Finally, we show our experimental results in Section 4, and concluding remarks are given in in Section 5.

2. Classification

2.1. The training phase

Our network for the vehicle classification task is based on a pre-activation ResNet, which has 18 layers applying shortcut connections [5]. The network receives $224 \times 224 \times 3$ images as input. The sizes of training images vary, so we resize each training image such that the shorter side is

† and ‡ equally contributed to this work. Specifically, † and ‡ mainly contributed in terms of the classification and localization tasks, respectively.

* Corresponding author.

[1] http://tcd.miovision.com/results/classification, http://tcd.miovision.com/results/localization

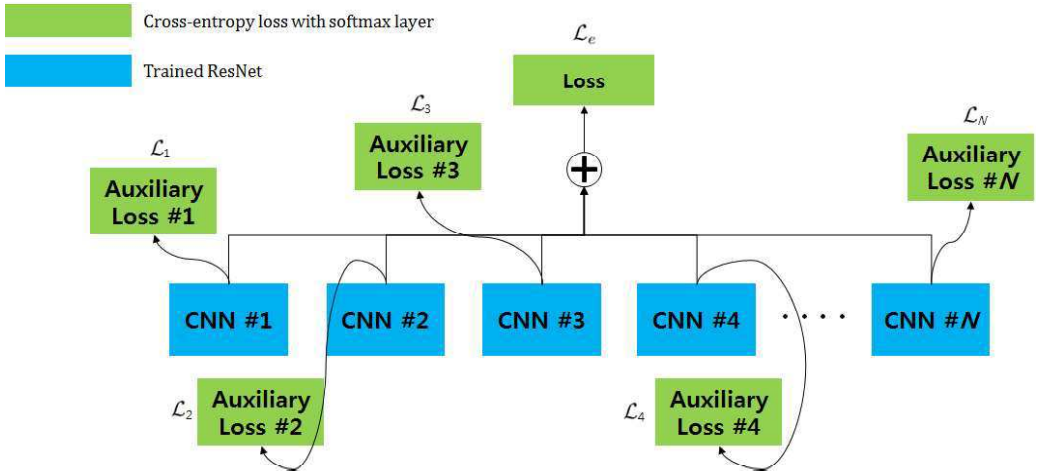

Figure 1. **A conceptual sketch of JF for the model ensemble.** Outputs of all CNNs are element-wise added, and they are used for the final prediction. Each CNN has an auxiliary loss for preventing loss of previously learned information.

in [192, 341.33], and then we randomly crop it to $224 \times 224 \times 3$. the RGB pixel values of each training image are normalized as follows:

$$\bar{\mathbf{x}}(i,j) = \mathbf{x}(i,j)/127.5 - 1, \tag{1}$$

where $\mathbf{x}(i,j)$ is a vector that has RGB pixel values at the coordinate of (i,j).

Photometric distortions [6] and color augmentation [9] are applied to obtain good results. We use rectified linear units (ReLU) as activation functions, and batch normalization is used for each layer. The number of outputs of the network is 11, which is the same as the number of categories of the dataset. For the final output, we utilize the softmax layer.

2.2. Fully joint fine-tuning for model ensemble

The original JF method was first proposed by Jung et. al. [8] The method shows better classification rates than traditional ensemble methods, such as averaging several models. The original JF method only fine-tunes the final classification layer, but we use fully JF for fine-tuning whole layers of CNN models.

The total loss $\mathcal{L}_t$ is defined as follows (For the sake of simplicity of notation, we assume the mini batch size is 1.):

$$\mathcal{L}_t(\mathbf{o}, \mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_N) = \mathcal{L}_e(\mathbf{o}) + \sum_{i=1}^{N} \mathcal{L}_i(\mathbf{o}_i), \tag{2}$$

where N is the number of CNN models, and $\mathcal{L}_e$ is a loss function for averaging models. Furthermore, $\mathcal{L}_i$ is an auxiliary loss for the i-th CNN, and the loss is only used in the training phase. $\mathbf{o}$ is defined as follows:

$$\mathbf{o}(\bar{\mathbf{x}}; \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{o}_i(\bar{\mathbf{x}}; \mathbf{w}_i), \tag{3}$$

where $\mathbf{o}_i(\bar{\mathbf{x}}; \mathbf{w}_i)$ is the output of the i-th CNN with a trained weight parameter of $\mathbf{w}_i$ and $\bar{\mathbf{x}}$ denotes input data normalized by equation 1. Each network is fine-tuned as using a very small value of learning rate η to minimize $\mathcal{L}_t$ (e.g., $\eta = 1e^{-5}$ in this paper). Figure 1 shows a conceptual sketch of JF for the model ensemble.

2.3. DropCNN

The traditional dropout method is used for preventing overfitting of deep neural networks [13]. Inspired by the dropout technique, we invent a dropping CNN method

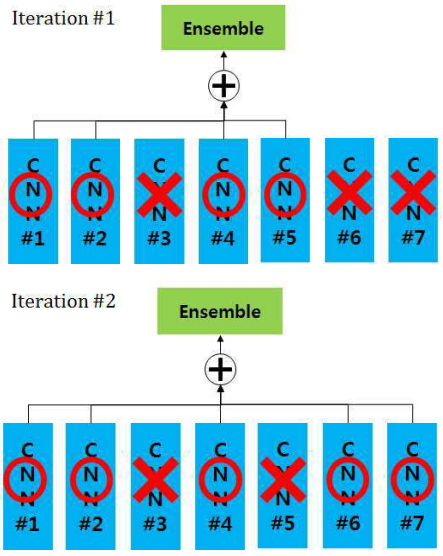

Figure 2. **An illustration of DropCNN.** CNNs are randomly chosen at each iteration during the JF training.

called by DropCNN. The method is used for maximizing the effect of JF, and it improves performances in terms of accuracy. The DropCNN is illustrated in Figure 2. CNNs are selected uniformly and randomly with a probability of $1 - p_d$, where p_d denotes a drop rate. (We use $p_d = 0.1$ in this paper) Only selected CNNs are averaged for the prediction in the training phase. This helps to make good predictions even with a small number of CNNs in the training phase. Finally, this process gives better results when performing predictions using all models in the test phase as shown in Table 1. The metrics used in the experiment are introduced in Section 4.

Table 1. Effectiveness of JF with DropCNN using a center crop image.

Method	Ensemble	JF	JF DropCNN
Cohen Kappa	0.9567	0.9576	**0.9578**
Accuracy	0.9721	0.9727	**0.9728**
Mean Precision	0.9293	**0.9314**	0.9307
Mean Recall	**0.9018**	0.8985	0.8991

2.4. The test phase

For a more accurate prediction, we use a multi-crop test. We change the fully connected layer at the top of the network to the fully convolutional form as in [3, 12], and we average the scores at multiple scales (the shorter side of input image is in 224, 256, 384, 480). In addition, we average the softmax outputs of the trained models after finishing JF with DropCNN, as mentioned in Sections 2.2 and 2.3.

3. Localization

Region-based convolutional detectors [2, 10] are the most promising approaches for object localization tasks compared to other detection methods. A branch of the region-based CNN methods achieved the highest accuracy in terms of mAP with a given intersection on union (IoU). A different version of the region based CNN was proposed by Jifeng et al. [1] combining an R-FCN with a position-sensitive score map and position-sensitive RoI pooling. For convenience, we devided the R-FCN's sublayers into two parts, depending on their purpose and, labelled them as follows: the convolutional feature extraction (CFE) Layer and the RoI feature localization (RFL) layer (see Figure 3. (b)).

We use the same architecture as in [1] with the position-sensitive score map: $r_c(i,j|\mathbf{w}) = \sum_{(x,y)\in B(i,j)} a_{i,j,c}(x + l, y + m|\mathbf{w})/n$. $r_c(i,j|\mathbf{w})$ is the response value estimated by the pooled region in the (i,j)th bin $B(i,j)$ for the c-th category, $a_{i,j,c}$ is a score map output of the $k^2(C+1)$ score map with k size of the position-sensitive RoI-pool layer and C number of classes, (l,m) denotes the top-left corner index of a RoI, n is the number of pixels in the B, and $\mathbf{w}$ denotes every learnable parameters of the model. We apply two types of ResNets with 50 and 101 layers [4] as a backbone CNN network for convolutional feature extraction, and a region proposal network (RPN) [10] is used for generating region candidates during the training process. Figure 3 (a) shows a sampled image and its ground truth annotation boxes from the MIO-TCD training dataset, and Figure 3 (b) shows a illustration of the overall structure of the R-FCN.

We trained the classifiers with the provided training data and an objective function consisting of the summation of the cross-entropy loss and the box regression loss: $\mathcal{L}(o, t_b) = \mathcal{L}_{cls}(o_{c^*}) + \lambda I(c^*)\mathcal{L}_{reg}(t_b, t_{b^*})$. Here, c^* is the RoIs ground truth label, which has a background label when c^* equals zero, and $I(c^*)$ is an indicator that equals to 1 except when c^* has the background label ($c^* = 0$). $\mathcal{L}_{cls}(o_{c^*}) = -\log o_{c^*}$ is the cross-entropy loss for classification under a given class output o_{c^*}, $\mathcal{L}_{reg}(t_b, t_{b^*})$ is the bounding box regression loss with $t_b = (t(x), t(y), t(w), t(h))$ as defined in [10], and λ is balance weight $\lambda = 1$ as in [2].

4. Experiments

4.1. Classification

The MIO-TCD-classification dataset has $648,959$ color images, such that $519,164$ can be used for training data and $129,795$ for test data. The dataset can be downloaded on the website.[2] Metrics for the evaluation of the classification task are as follows:

$$Accuracy = \frac{TP}{\#\ of\ Testing\ Images}, \tag{4}$$

where TP is the number of true positive images. The mean precision ($mPre$) and mean recall ($mRec$) of each category are defined as follows:

$$mPre = \text{mean}(Pre_c), \tag{5}$$

$$mRec = \text{mean}(Rec_c), \tag{6}$$

where $Pre_c = \frac{TP_c}{TP_c + FP_c}$, $Rec_c = \frac{TP_c}{TP_c + FN_c}$ of category c. $mean(\cdot)$ denotes averaging all values over c.

Our method shows the best performance among other state-of-the-art methods, as illustrated in Table 2. The single model of ResNet-18 with a center crop achieves 96.99%, and the accuracy of the the eight-model ensemble (average) is 97.93%. Surprisingly, our ResNet-18 is better than ResNet-50, which is a deeper model than our network is. Our JF with DropCNN model achieved 97.95% accuracy, and this is superior to the averaging ensemble model.

Table 3 gives a confusion matrix of our JF with DropCNN. Our method shows high accuracies in the bus,

[2] http://podoce.dinf.usherbrooke.ca/challenge/dataset/

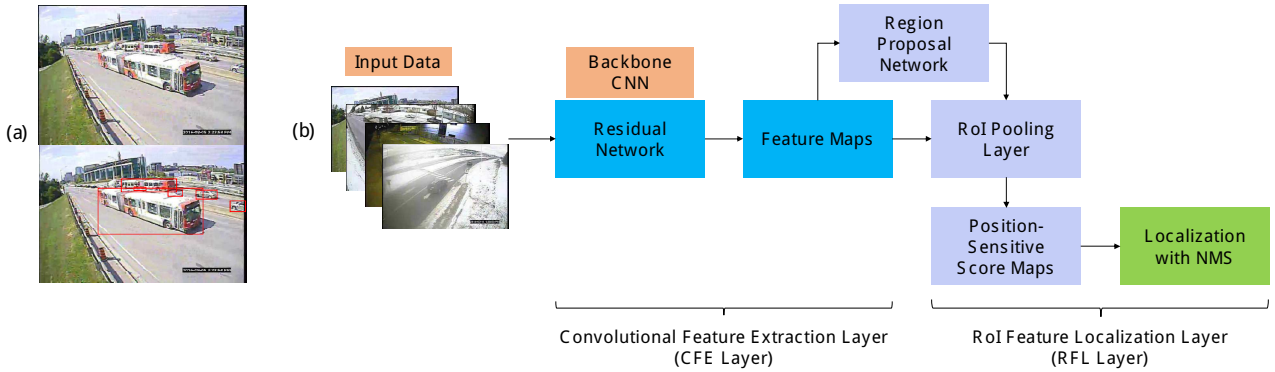

Figure 3. **Overview of the localization task.** (a) A sampled image and its ground truth annotations from MIO-TCD. (b) Overall architecture of the region based fully convolutional network (R-FCN). The R-FCN can be divided into two parts such that the convolutional feature extraction (CFE) layer extracts CNN features from the backbone architecture and the RoI feature localization (RFL) layer localizes RoI features and bounding boxes with NMS (non-maximum suppression).

Table 2. Experimental results for the classification task.

Method	Cohen Kappa Score	Accuracy	Mean Precision	Mean Recall
VGG16+FineTune [12]	0.9403	0.9616	0.8802	0.8502
ResNet-50+FineTune [4]	0.9326	0.9568	0.8654	0.8239
ResNet50 [4]	0.9276	0.9539	0.8445	0.7802
VGG19 batch norm [12, 7]	0.9175	0.9467	0.8074	0.7971
AlexNet [9]	0.8957	0.9330	0.7729	0.7583
AlexNet+SVM [9]	0.8284	0.8930	0.8076	0.6182
ResNet-50+SVM [4]	0.8271	0.8921	0.8240	0.6305
VGG16+SVM [12]	0.8078	0.8808	0.8102	0.5924
ResNet-18 center crop	0.9535	0.9699	0.9223	**0.9001**
ResNet-18 8 models multi crop	0.9677	0.9793	0.9518	0.8987
ResNet-18 8 models JF + DropCNN	**0.9681**	**0.9795**	**0.9530**	0.8970

car, and background categories. Especially, for the background category, our method achieves near 100% accuracy. We consider that there are not significantly large variations in the background category because the images taken from the traffic surveillance camera have a similar environment.

4.2. Localization

We performed experiments for localization on MIO-TCD, which 110,000 training images and 27,743test images, with eleven object classes, as follows: articulated truck, bicycle, bus, car, motorcycle, non-motorized vehicle, pedestrian, pickup truck, single unit truck, work van, and background. Both train and test datasets contain background and foreground objects with different properties, and some specific regions of these images provide significant occlusion and clutter, inconsistent contrast, various levels of illumination, and pose variation of objects.

We trained four R-FCN models with different random initializations and layer depths (two ResNet-50s and two ResNet-101s), and we tuned hyper-parameters of networks arbitrarily divided into training datasets using 5-fold cross validation. For each model, we set the size of the position-sensitive RoI-pool layer $k = 7$, and then we used $7^2(11 + 1)$ channel convolutional layers to obtain position-sensitive score map. We applied online hard example mining (OHEM) [11] to balance positive and negative samples during training. We set the learning late 0.001 for the initial 80k iterations and then multiplied 10^{-1} for each step until 300k iterations were reached, with a weight decay of 0.0005 and a momentum of 0.9 for every model. Non-maximum suppression (NMS) was applied for post-processing to merge the initially localized bounding boxes with a threshold of 0.3 IoU, as a standard approach.

We also performed experiments to verify the effect of the maximum number of region candidates produced by the RPN, and then we fixed the maximum candidate regions at 700 (see Table 4). Other layer specifications and hyper-parameters followed the original configuration given in [1]. After training each R-FCN model, we ensembled the models into one localization model, as shown in Figure 4. Each model receives an image as an input data, then forwards it and generates initial bounding boxes on an input image.

Table 3. Confusion matrix of our method (%). The labels in the leftmost column and on the top represent the ground truth and prediction results, respectively.

	Articulated Truck	Bicycle	Bus	Car	Motorcycle	Non-motorized Vehicle	Pedestrian	Pickup Truck	Single Unit Truck	Work Van	Background
Articulated Truck	93.24	0.00	0.35	0.93	0.00	0.23	0.00	0.23	4.33	0.15	0.54
Bicycle	0.18	89.49	0.18	2.45	0.70	0.00	6.48	0.00	0.00	0.00	0.53
Bus	0.23	0.00	97.79	1.47	0.00	0.00	0.00	0.35	0.08	0.08	0.00
Car	0.00	0.00	0.00	98.53	0.00	0.00	0.00	1.42	0.00	0.04	0.00
Motorcycle	0.00	1.82	0.00	2.22	91.11	0.00	0.40	0.00	0.00	0.00	4.44
Non-motorized Vehicle	10.73	0.23	0.23	0.91	0.00	52.28	0.68	3.65	13.01	2.05	16.21
Pedestrian	0.00	1.47	0.00	0.26	0.06	0.00	94.06	0.00	0.00	0.00	4.15
Pickup Truck	0.00	0.00	0.02	4.50	0.00	0.00	0.00	95.39	0.09	0.01	0.00
Single Unit Truck	10.31	0.00	0.16	0.47	0.00	0.16	0.00	2.58	83.36	1.09	1.88
Work Van	0.08	0.04	0.08	5.62	0.00	0.04	0.04	0.78	0.78	91.66	0.87
Background	0.04	0.00	0.02	0.07	0.00	0.00	0.00	0.01	0.01	0.02	99.84

Table 4. Localization mAP using different backbone CNNs with the maximum number of top scoring boxes with NMS (the-cross validation on training dataset).

Model (# maximum region proposals)	mAP (%)
ResNet-50 (300)	79.53
ResNet-50 (700)	80.45
ResNet-101 (300)	79.71
ResNet-101 (700)	**80.56**

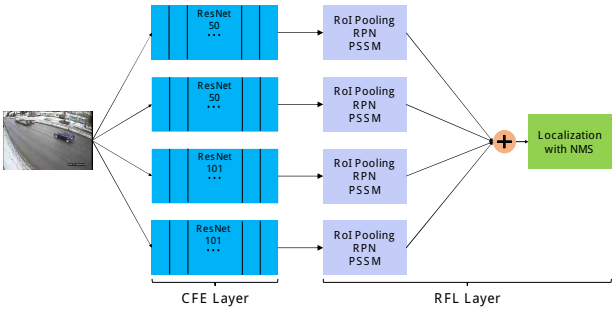

Figure 4. **An illustration of the multi-network ensemble for region-based object detector.** For the an input data, each model generates candidate boxes through the CFE and RFL layers, and then the generated boxes are merged into an image by NMS.

These bounding boxes from each model are merged into finalized one applying NMS. Finally, we achieved 79.24% mAP of the overall classes with the ensemble model using four trained R-FCNs, which showed the highest accuracy among the state-of-the-art models, as illustrated in Table 5. Detailed mAPs in each class of the final ensemble models are shown in Figure 5.

5. Conclusion

We presented methods that classify and localize vehicles in surveillance recordings. For the classification task, we basically used a deep ResNet with 18 layers and ReLU. Furthermore, a new ensemble method called JF with DropCNN was proposed, and it was useful for increasing the model's accuracy. For the localization task, we adopted an R-FCN

Table 5. Comparison of mAP among state-of-the-art models on the test dataset with different detector settings.

Model	mAP (%)
YOLO-v1	62.65
Faster-RCNN	69.98
YOLO-v2-PascalVOC	71.47
YOLO-v2-MIOTCD	71.83
SSD-300	73.97
R-FCN (ResNet-101)	77.76
R-FCN (ResNet-50)	77.99
R-FCN (2-models ensemble)	78.65
R-FCN (3-models ensemble)	78.98
R-FCN (4-models ensemble)	**79.24**

with deep residual models and implemented its ensemble network to achieve more accurate results. Finally, our classification method outperformed other state-of-the-art models, such as AlexNet, ResNet 50+SVM, and VGG16+SVM, and our localization method outperformed state-of-the-art detectors, such as YOLO, F-RCNN, and SSD. In future, we plan to analyze the joint fine-tuning with DropCNN to gain deeper insights into the classification and exploit advanced pre- or post-processing, such as soft NMS, for the localization. We will soon release our models, and source codes are publicly available.

Acknowledgment

This work was partly supported by the DGIST R&D Program of the Ministry of Science of Korea(17-FA-07) and the DGIST R&D Program of the Ministry of Education, Science and Technology of Korea(17-IT-01). Furthermore, this research was also supported in part by Institute for Information &communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (R7117-16-0172, Development of the advanced UI/UX System for Safe Driving using the Convergence of ICT).

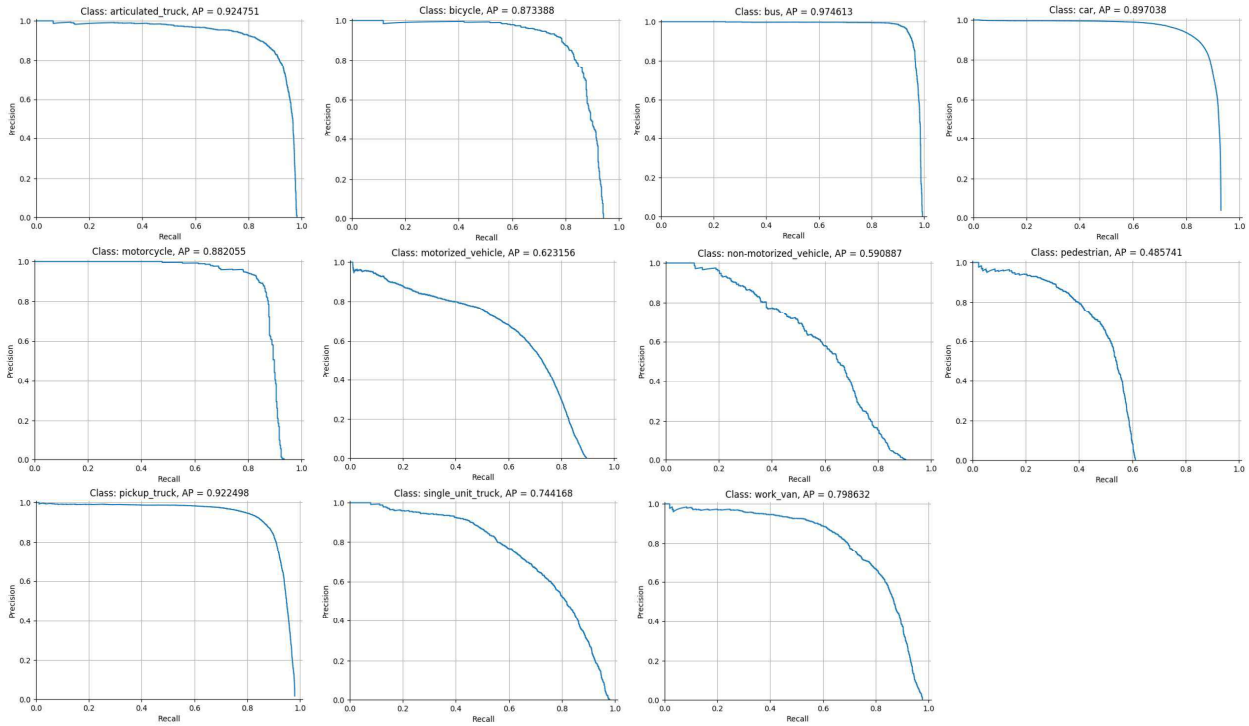

Figure 5. **Per-class precision-recall curves from a four-ensemble model.** Each class name and its average precision (AP) are shown over each precision-recall curve.

References

[1] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, pages 379–387, 2016. 1, 3, 4

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *IIEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, 2016. 3

[3] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 3

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 3, 4

[5] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016. 1

[6] A. G. Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013. 2

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 4

[8] H. Jung, S. Lee, J. Yim, S. Park, and J. Kim. Joint fine-tuning in deep neural networks for facial expression recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2983–2991, 2015. 2

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 2, 4

[10] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IIEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. doi:10.1109/TPAMI.2016.2577031. 3

[11] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3, 4

[13] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 2

Figure 6. **Example detection results for the test dataset from the configuration that achieved 77.99% mAP with a single ResNet-50 model.** Each image was randomly selected, and a score threshold of 0.5 is used for visualization.