Electronic Preprint for Journal of Information Processing Vol.27

Regular Paper

Two-stage Deep Neural Network for General Object Detection

Tran Duy Linh[1,a)] Masayuki Arai[1,b)]

Received: June 5, 2018, Accepted: December 4, 2018

Abstract: In the present paper, we propose a deep network architecture in order to improve the accuracy of general object detection. The proposed method contains a proposal network and a classification network, which are trained separately. The proposal network is trained to extract a set of object candidates. These object candidates cover not only most object ground truths but also a number of false positives. In order to make the detector more robust, we train these object candidates using a secondary classifier. We propose combination methods and prove that a combination of two networks is more accurate than a single network. Moreover, we determine a new method by which to optimize the final combination results. We evaluate the proposed model using several object detection datasets (Caltech pedestrian, Pascal VOC, and COCO) and present results for comparison.

Keywords: object detection, deep learning, two-stage detector

1. Introduction

Object detection is a key problem in computer vision, and the performance of detectors has rapidly improved recently. Given an input image, these methods output a set of detections, where each detection includes an object label, an object location, and a score indicating the confidence of the detection. Most conventional object detection methods use deep learning based on strong baseline architectures, such as Fast/Faster R-CNN [11], [26] and a single-shot detector (SSD) [22].

However, the detection results usually contain numerous false positive detections. There are two kinds of false positive errors: poor localization and misclassification. For instance, in the SSD [22] method, more than half of false positive detections are due to misclassification. **Figure 1** shows examples of misclassification detection because of ambiguity with background objects, similar objects, or dissimilar objects. This issue motivates the use of an additional classifier to reduce the number of misclassification detections.

The idea of using two stages to detect objects has been considered in numerous studies [11], [18], [26]. These models rely on an external or an internal region proposal generator (as the first stage) to predict class independent box proposals. Thus, the box proposals can be separated into objects and non-objects (the background). The second stage is used to predict the class and offset the shifting proposal location to fit the ground-truth bounding box.

In the proposed approach, we also use an object detector as a proposal generator. However, the proposals extracted using this detector contain not only box locations but also class labels and scores, we refer to this detector as the proposal network. Following this proposal network, we add another convolutional neural network, called the classification network, to re-classify the extracted proposals. The output of the two networks allows us to choose either the first stage scores or the second stage scores as the final scores. However, we observed that, in general, the second stage scores after re-classification do not boost detection performance. There are two reasons for this unsatisfactory accuracy: (i) the classification network focuses on only classification objects based on their similarities and differences but lack localization support, and (ii) there are no connections between proposal confidence and classification confidence.

In the present paper, we investigate how to use the secondary network as an object proposal classifier to improve the object detection performance for various datasets. We present a simple but effective network model for a two-stage detector. In the proposed method, we connect the first scores of the proposal network and the second scores of the classification network using a combination function. The proposal network overcomes the limitation of using single scores as final detection scores, thus improving the performance of a given object detector.

Since we primarily discuss the performance of the two-stage object detection model, we do not apply enhanced methods such as multi-crop, horizontal flipping, or multi-scale during testing.

The main contributions of the present paper can be summarized as follows:

(1) We prove that training the secondary classifier improves the detection performance without additional training datasets. We evaluate the proposed method on several benchmarks, including Caltech [6], Pascal VOC [9], and COCO [19].

(2) We discuss how to combine two networks by introducing various combination procedures.

[1] Graduate School of Science and Engineering, Teikyo University, Utsunomiya, Tochigi 320–8551, Japan
[a)] duylinh161287@gmail.com
[b)] arai@ics.teikyo-u.ac.jp

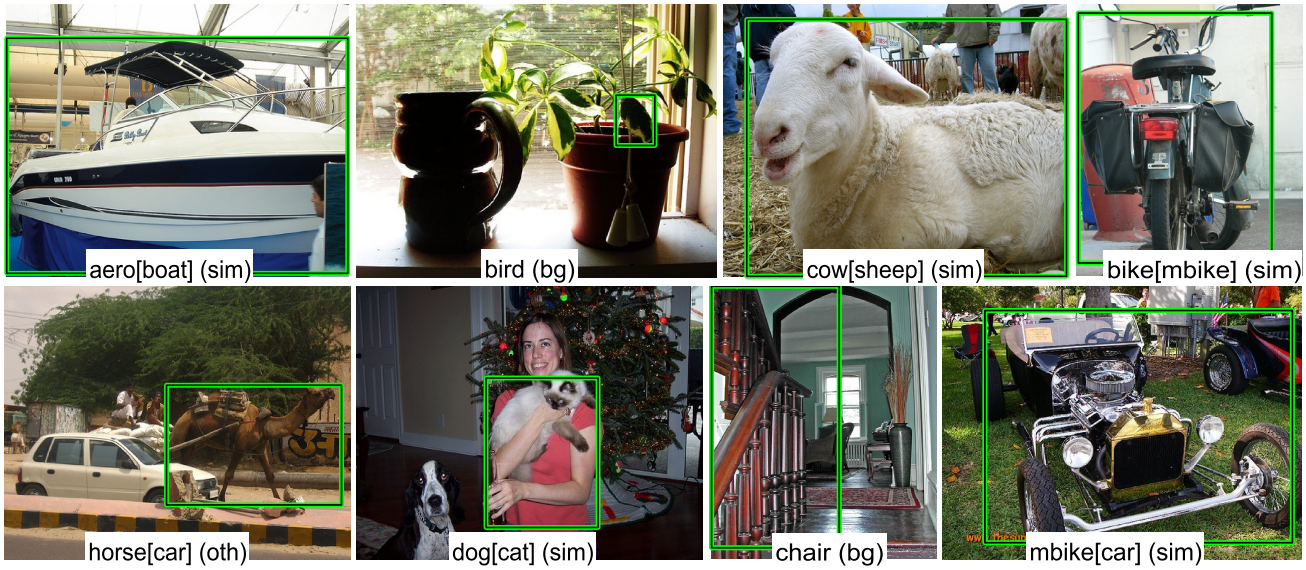


Fig. 1 False positive detections on the Pascal VOC dataset by the Faster R-CNN object detection model. The green bounding boxes depict the object location inside an image. The text under the image indicates the detected class, the type of misclassification (sim: confusion with similar objects, oth: confusion with other objects (dissimilar objects), bg: confusion with background). "Confusion" occurs when a false positive detection matches with an object from the non-target category with IOU at least 0.1. We show the ground-truth class for sim and oth cases in the square brackets.

(3) We also propose an optimized combination method to improve the performance of final detection scores. Remarkably, the proposed combination method can be used to efficiently train the classification network. The final output has higher accuracy than the stand-alone proposal network as an object detector. The experimental results obtained using the proposed method are comparable to those obtained using state-of-the-art object detection methods.

2. Related Research

2.1 Fast/Faster R-CNN

Fast R-CNN [11] is a deep learning object detector that combines an object proposal method (e.g., Selective Search [32]) and a convolutional neural network (CNN) classifier. Fast R-CNN introduces a Region Of Interest (ROI) pooling mechanism and multi-task losses by minimizing the loss function of both class confidences and bounding box regression.

The improved version of Fast R-CNN, i.e., Faster R-CNN [26], replaces the region proposal component with a deep network. Faster R-CNN contains two components: a Region Proposal Network (RPN) and a Fast R-CNN [11] object detection network. The first component, the RPN, is a fully CNN network that predicts class-agnostic box proposals. These networks can be trained separately or end-to-end and share the extracted image features with the object detection network.

The shared features are fed to the remaining layers of the features extractor. The second component uses the box proposals to crop features and outputs the class-specific and box offset for each proposal.

2.2 Single Shot Detector (SSD)

The SSD [22] is a single-shot feed forward network based on a pre-trained network for object detection. Basically, the SSD model is very similar to the RPN component of the Faster R-CNN model, except that the SSD model directly predicts class-specific and box offsets and does not require a second object detection network. Instead of using a CNN module to extract the set of RPN, the SSD uses a fixed set of default boxes (anchors) for prediction, and thus the SSD can avoid merging the RPN module with Fast R-CNN. The SSD model can generate a large pool of possible box shapes by discrete output into a set of default boxes of different scales and aspect ratios at several feature map locations. This approach allows the SSD model to achieve slightly better speed than Faster R-CNN-like detectors.

2.3 One-stage and Two-stage Object Detectors

We consider two approaches to object detection modeling. In the first approach, one-stage detectors (unified detectors), such as an SSD [22] or YOLO [24], [25], use a single CNN to predict object location in an entire image. Since one-stage detectors require only single-network computation, their speed is better than other CNN-based detectors. However, the limitation of YOLO is that detecting small objects is difficult and does not work well with unusual object aspect ratios. Although the SSD provides better object localization than YOLO, because of location sharing for multiple categories, the SSD method involves increased confusion with similar object categories. Moreover, the SSD and MS-CNN [3] independently predict objects at multi-feature maps locations, because there are no combinations of features or scores.

In the second approach, two-stage detectors, such as Fast/Faster R-CNN [11], [26] and R-FCN [5], require the first stage to extract object proposals. By refining the object proposals twice (when refining the anchors to class-agnostic box proposals and when refining the RPN output to class-specific boxes), Fast/Faster R-CNN-like methods can obtain better detection results than one-stage detectors. However, training these detectors

requires significant effort because it is difficult to optimize each network component.

The proposed model uses a two-stage approach in which the proposals are class-specific boxes. Without combining the first and second stages, the proposed model is equivalent to a non-box-regression Fast R-CNN detector with a pre-computed RPN. The proposed method is also different from Fast/Faster R-CNN in terms of sampling. The extracted RPN proposals of Faster R-CNN are in the same image or the same image batch, whereas the second stage can freely shuffle the training samples in the entire training dataset. Following Ref. [2], it is efficient for optimizing the classifier that the order of samples is changed for each epoch, and each sample is sampled independently.

3. Proposed Method

Figure 2 shows the proposed architectures. Each model contains two stages: the proposal network and the classification network. An input image is fed to the proposal network (e.g., Faster R-CNN) to generate a set of object proposals inside the image. The output of the first stage is the candidate boxes and the corresponding scores for each category of these boxes. The boxes are again used for sample images to fine-tune the classification network (e.g., Inception-V3 [31]). In the second stage, we introduce two combination strategies: post-combination and during-training combination (referred to hereinafter as trained combination). The details of combination methods are discussed in Section 3.3.

3.1 Proposal Network

In the first stage of the proposed model, we choose a proposal network to extract the set of object candidates. We primarily focus on Fast/Faster R-CNN and an SSD because many recent methods [3], [12], [13], [18] are based on these architectures.

3.1.1 Single-shot Detector Meta Architecture

With the SSD model, we use VGG16 [28] pre-trained by the ImageNet [27] dataset as the baseline network, which has a simple design and is widely used. The VGG16 baseline network is modified by converting the fc6 and fc7 layers to convolutional layers, which also reduce the number of outputs to 1,024. All dropout layers and the fc8 layer are removed from the original VGG16 network. We add five additional output layers (conv6_2, conv7_2, conv8_2, conv9_2, and conv10_2) to predict the locations and scores of objects. The model is then finely tuned using the COCO dataset [19] for the COCO test-dev detection task.

3.1.2 Faster R-CNN Meta Architecture

We use a high-quality features extractor, such as Resnet-101 [13] or Inception-Resnet-V2 [29], which are used in state-of-the-art ImageNet ILSVRC 2012 classification and detection tasks, as a baseline.

In the Resnet-101 network, we extract features from the last layer of the conv4 block. The Region Of Interest (ROI) is cropped to 14×14, and a maxpooling layer is then used to reduce the feature size to 7×7.

Inception-Resnet-V2 is the mixing residual design with Inception networks [30]. We implement Inception-Resnet-V2 on TensorFlow [1] and extract features from the Mixed_6a layer. The features are cropped by ROI and resized to 17×17 and max-pooled with a stride of 1.

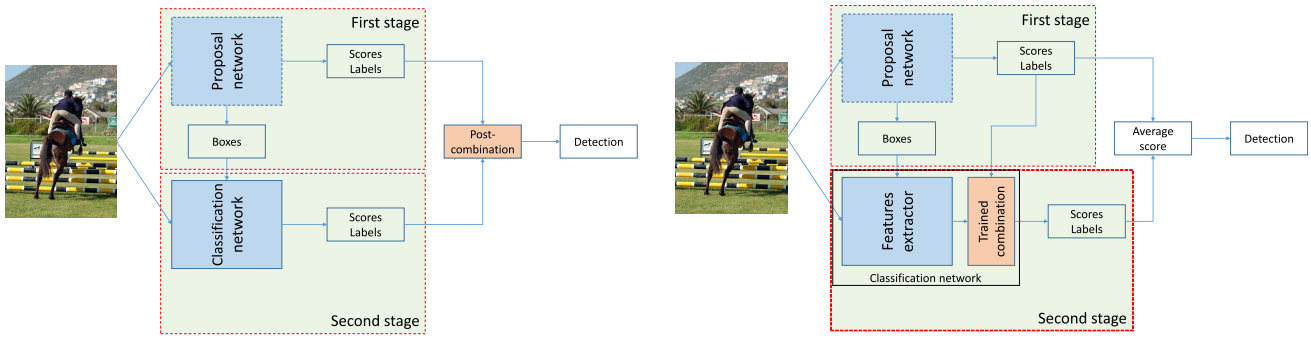
Training details According to Ref. [26], instead of using four-step alternating training, we adopt an end-to-end joint training of the RPN and the Fast R-CNN component for convenience. The multi-task loss on each ROI is the sum of the cross-entropy loss of classification and the box regression loss $L(p, u, t, t^*) = L_{cls}(p, u) + \lambda[u \geq 1]L_{box}(t, t^*)$, where $L_{cls}(p, u) = -\log(p_u)$ is the cross-entropy loss for true class u. The tuple $t = (t_x, t_y, t_w, t_h)$ is predicted bounding box regression offsets, and $t^* = (t_x^*, t_y^*, t_w^*, t_h^*)$ represents the ground-truth. The indicator function $[u \geq 1]$ is equal to 1 when $u \geq 1$, and is 0 otherwise. In our experiment, we set the balance loss weight $\lambda = 1$. We use the non-maximum suppression (nms) with a threshold of 0.7 intersection over union (IOU) and 0.6 IOU for the RPN and Fast R-CNN, respectively.

3.2 Classification Network

3.2.1 Baseline Networks

After extracting the set of object proposals from an image, the number of false positive detections is still large because of the sliding window scheme. We use these proposals to train the second stage as a classifier. In the present paper, we use several classifiers, such as Resnet-101 [13], Inception-V1 [30], or its successor Inception-V3 [31]. All of these classifiers are pre-trained on ImageNet, which contains 1.2 million training images.

In the Inception-V1 baseline, we extract the features from the Mixed_5c layer and add an average pooling layer with kernel size 7×7. The network input size is 224×224. In Inception-V3, we



(a) Model 1. (b) Model 2.

Fig. 2 Models proposed in the present paper.

extract the features from the Mixed_7c layer and add an average pooling layer with kernel size 8×8. The input size of Inception-V3 is 299×299. We do not use auxiliary branches of Inception-V1 and Inception-V3 for fine-tuning the classification network. In the Resnet-101, we faithfully follow the design of Ref. [13] which puts batch normalization after every convolutional layer. We set the input size of Resnet-101 to 224×224.

3.2.2 Data Sampling

Although the second stage is trained using the same dataset as the proposal network (except for the Caltech pedestrian dataset [6], as described in Section 4.1), the method for sampling data is different from the first stage. We first eliminate some "bad" proposals (boxes with very low confidence scores or boxes that are too small). Each proposal is then matched with ground-truth boxes. We calculate the IOU of these object proposals with the ground-truth available in an image. We set a proposal p as a positive sample if $\alpha^* \geq 0.5$, where

$$\alpha^* = \max_{g^i \in G} IOU(p, g^i) \tag{1}$$

and $G = \{(g^i, l^i)\}_{i=1}^{M}$ is the set of ground-truth bounding boxes where $g^i = (g^i_x, g^i_y, g^i_w, g^i_h)$ specifies the top left coordinates of the ground-truth box together with its width and height, l^i is the ground-truth class, M is the number of ground-truth boxes in the given image, otherwise p is a negative sample. In case of a positive sample, the class of proposal p is l^{i^*} where $i^* = \text{argmax}_{i \in [1,M]} IOU(p, g^i)$.

In order to train a more stable classification network, we maintain a constant ratio between the number of positive samples and the number of negative samples (In our experiments, the ratio of positive to negative samples is 1 : 3).

3.2.3 Training

The classification network is fine-tuned from a pre-trained model. The proposal regions from the first stage are used to crop the input images, then are resized using bilinear interpolation to size of 256×256 for Inception-V1 and Resnet-101, and 340×340 for Inception-V3. We then randomly crop to network input size for training (at the test time, we use center cropping). The training samples are randomly flipped horizontally (left to right) with probability of 0.5 and are subtracted from the dataset image means.

3.3 Combination of Two Networks

As shown in Fig. 2, in the proposed model architectures, our combination module uses post-combination and trained combination. These combination procedures are key techniques by which to make a stronger object detector based on an object detector as a proposal network. We use post-combination for model 1 and trained combination for model 2.

Post-combination In this approach, the combination occurs after the training of two stages. The classification network outputs the scores of object proposals and the final detections can use either proposal scores or classification scores. However, we observed that standalone classification scores do not improve detection performance. For large and clear objects, the proposals are usually well detected, and the detection scores are high. On the other hand, deep-learning-based detectors tend to be weak for small objects because of insufficient resolution of feature maps for detecting small instances [34]. In this case, the output detection scores are usually low. In order to address this problem, we propose a method by which to maintain good detections and enhance weak detections by combining two detection scores. For each output of the proposal network $(\mathcal{L}, s_p)$, $\mathcal{L} = (x_1, x_2, y_1, y_2)$ is the location of the box, $s_p = (s_{p_0}, \ldots, s_{p_N})$ is the score of the proposal network, and $s_c = (s_{c_0}, \ldots, s_{c_N})$ is the score of the classification network. $i^* = \text{argmax}_{i \in [0,N]}\, s_{p_i}$ and $j^* = \text{argmax}_{j \in [0,N]}\, s_{c_j}$ are the detected classes of the proposal network and the classification network, respectively. The final detection of the combined network is $(\mathcal{L}, s_{i^*}, i^*)$ where

$$s_{i^*} = f(s_p, s_c) \tag{2}$$

where f is a combination function. We evaluate several combination functions. The combinations are based on the value of two scores and the agreement of two detection classes. We consider a mean function (f_1) and a multiplication function (f_2), which are defined as follows:

$$f_1(s_p, s_c) = \frac{(s_{p_{i^*}} + s_{c_{i^*}})}{2} \tag{3}$$

and

$$f_2(s_p, s_c) = (s_{p_{i^*}} * s_{c_{i^*}}) \tag{4}$$

and we define f_3 as follows:

$$f_3(s_p, s_c) = \begin{cases} f_1(s_p, s_c * c) & \text{if } i^* = j^* \text{ and } s_{p_{i^*}} < d \\ f_1(s_p, s_c) & \text{otherwise} \end{cases} \tag{5}$$

where $c > 1$ is the boosting weight and $d < 1$ is the high-score threshold, respectively.

The key idea behind the function f_3 is that we encourage detections, which have the same detected classes, that are more confident between the proposal network and the classification network. However, the scores of the proposal network for these cases are not good enough (lower than the threshold score d). Note that to increase the final score, it is possible to boost the first stage score (s_p), but we found that the experimental results of boosting first stage score are worse than that of boosting the second stage score (s_c).

Trained combination In this approach, the combination occurred not only after training of the classification network but also during the training classifier process. Given a proposal $P = (P^1_x, P^1_y, P^2_x, P^2_y)$, the output of the last layer in the classification network is denoted by $\phi_l(P)$. By adding the new combination module in Fig. 2 (b), the output of the classification network becomes $\hat{\mathbf{w}}_\star(\phi_l(P), s_p)$, where s_p is the score vector, and $\hat{\mathbf{w}}_\star$ are trainable parameters. In this case, for true class u the training loss is:

$$L_{cls}(p, u) = -\log(p_u) \tag{6}$$

where $p = (p_0, \ldots, p_N) = f_{cls}(\hat{\mathbf{w}}_\star(\phi_l(P^i), s_p))$ is computed as softmax (f_{cls}) over the $N + 1$ outputs of the last fully connected layer. After training the second stage, we apply the average score

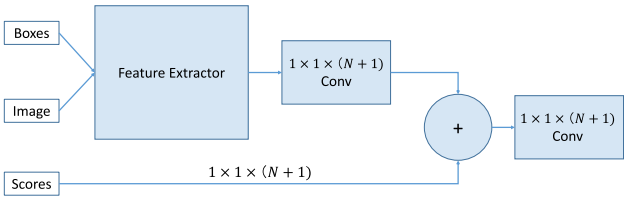


Fig. 3 Classification network with trained combination.

Table 1 Evaluation settings on the Caltech benchmark.

	Height (pixels)	Occlusion level
All	20–*inf*	0–80%
Reasonable	50–*inf*	0–35%
Near scale	80–*inf*	0%
Medium scale	30–80	0%
Far scale	20–30	0%
No occlusion	50–*inf*	0%
Partial occlusion	50–*inf*	1–35%
Heavy occlusion	50–*inf*	35–80%
Small objects	30–50	0–35%

to generate the final score.

The implementation of the combination module is depicted in **Fig. 3**. We construct a $1 \times 1 \times (N+1)$ vector from the score of the proposal network, where N is the number of interest categories and plus 1 for background. This vector is concatenated with extracted features from the last layer of the features extractor network, and then connected to a fully connected layer. The computation of this new trained combination module is nearly cost-free in comparison to training the original classification network.

4. Experiments

In this section, we applied the proposed models to several object detection datasets. These object detection datasets vary in terms of the number of images, the testing conditions, and the evaluation method. While the experiments mainly showed the benefit of using a two-stage network detector, we also obtained results that were comparable to those obtained using other cutting edge detection methods. The description of the two-stage model used in various experiments is given as follows: SSD+Inception-V1+M represents the proposal network is the SSD, the baseline of the classification network is Inception-V1, and the method of combination is multiplicity score f_2 (M). Other notation of the combination method includes the mean function f_1 (A), the combination with threshold function f_3 (T), the trained combination (TC).

4.1 Caltech Pedestrian Detection

We first applied the proposed network for the only one category of dataset, namely, the Caltech pedestrian [6]. The dataset consists of approximately 1,900 individual pedestrians, which are annotated with approximately 350,000 ground-truth annotations. The number of images in the test set is 4,024. The Caltech evaluation benchmark uses the log-average miss rate to summarize the detector performance. The average miss rate is computed by averaging the miss rate for nine false positives per image (FPPI) in log space from 10^{-2} to 10^{0}. The evaluation is performed using several different settings based on the height and occlusion level of pedestrians. The settings are listed in **Table 1**.

In Ref. [34], it is argued that, despite particularly successful general object detection, Faster R-CNN (as a stand-alone detector) has limited success for pedestrian detection. For this reason, we use an SSD [22] with a VGG16 baseline [28] as the proposal network for this experiment. In the network settings, the aspect ratios of the default bounding boxes (the anchors) are $\{1/3, 1/2, 1, 2, 3\}$. For output from the conv4_3 layer, because of the different feature scale, an *L2* normalization layer [23] is added in order to scale down the feature norm to 20.

In order to fine-tune the proposal network, we used a pre-trained SSD from the COCO [19] dataset because the COCO dataset is more similar to the Caltech dataset than ImageNet [27]. Since Caltech is a relatively small dataset, we adopt the data augmentation strategy by adding more training images from other pedestrian datasets: the KITTI dataset [10], the TUD-Brussels dataset [33], and the ETH pedestrian dataset [8]. This addition of data increases the number of images by 26% compared to the Caltech dataset alone. We trained a model with a base learning rate of 10^{-4}, a momentum of 0.9, and a weight decay of 0.0005. The total number of training iterations is 240,000. We used Inception-V1 [30] to classify pedestrian candidates extracted from the proposal network, where the initial learning rate is 10^{-4}.

The performances of the proposal network, the classification network, and the combination of the two networks are compared. **Table 2** shows the experimental results. The classification network is better than the proposal network for the "all", "medium", and "small objects" settings by 2.36%, 4.84%, and 3.79%, respectively. However, the performance of the classification network is inferior for "near" objects. This indicates that Inception-V1 is more robust than the SSD for small pedestrian classification. Moreover, the performances of occluded pedestrians ("partial occ." and "heavy occ.") of the classification network are also worse than the proposal network. The reason might be that the SSD model's classification task is supported by the box prediction task which allows the model can better predict the objects of unusual aspect ratio (occluded objects). The results in Table 2 show that the miss rates for every setting are reduced for all combination functions to compare with the proposal network. With combination function f_3, the performance for the near scale is slightly improved (0.32%) and can be significantly improved for smaller objects. The miss rate reductions for the "small objects" and "medium" setting are 7.30% and 8.81%, respectively. Overall, the mean combination f_1 is slightly better than the multiply combination f_2.

We also compare the proposed model with other state-of-the-art pedestrian detection methods. For overall configuration ("all" setting), our proposed method outperforms MS-CNN [3] method and is close to the result of F-DNN+SS [7] which uses pixel-wise semantic segmentation for reinforced detection. Furthermore, our method is simpler and faster than F-DNN+SS which requires 2.48 s per image [7].

4.2 Pascal VOC Object Detection

We used the Pascal VOC [9] dataset to evaluate the proposed method. The Pascal VOC 2007 test set has 4,952 images belonging to 20 categories for the object detection task. We performed

Table 2 Miss rates of detection using the proposal network (SSD 512) and the classification network, and combination scores for the Caltech test set.

	All	Reasonable	Near	Medium	Far	No occ.	Partial occ.	Heavy occ.	Small objects
Proposal network (SSD [22])	57.74	17.56	1.47	43.11	79.61	15.59	27.89	69.02	48.86
Classification network (Inception-V1)	55.38	17.63	1.59	38.27	77.71	15.53	30.34	70.08	45.07
SSD+Inception-V1+A	52.02	14.59	1.44	34.77	77.26	12.74	26.01	68.34	42.05
SSD+Inception-V1+M	52.30	14.97	1.44	35.33	77.59	13.11	**25.66**	67.85	42.21
SSD+Inception-V1+T	**51.70**	**13.89**	**1.15**	**34.30**	**74.92**	**11.75**	27.29	**66.59**	**41.56**
SSD+Inception-V1+TC	55.11	15.28	1.33	38.64	79.67	13.08	28.19	68.62	47.06
MS-CNN [3]	60.95	9.95	2.60	49.13	97.23	8.15	19.24	59.94	70.34
F-DNN+SS [7]	**50.29**	**8.18**	2.82	**33.15**	77.37	**6.74**	**15.11**	**53.76**	45.14

Table 3 Individual average precision (%) on the Pascal VOC 2007 test set. All models are trained with the COCO data set and then fine-tuned with the Pascal VOC 2007+2012 training set. We then retrain Faster R-CNN [26] using for the fine-tuning classification network (note: Faster R-CNN[1] uses Resnet-101 as a baseline, and Faster R-CNN[2] uses Inception-Resnet-V2 as a baseline).

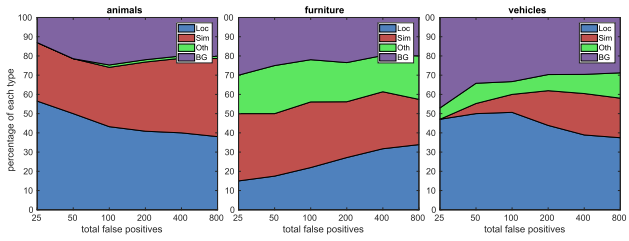
	vehicles							animals							furniture						
	aero	bike	boat	bus	car	mbike	train	bird	cat	cow	dog	horse	sheep	person	bottle	chair	table	plant	sofa	tv	mAP
Proposal Faster R-CNN[1] [26]	80.0	80.2	67.7	79.7	86.6	79.3	78.6	77.2	87.0	83.5	84.6	86.8	75.7	77.1	65.4	62.4	73.3	40.3	77.9	69.6	75.6
Faster R-CNN+Resnet-101+A	81.0	80.2	70.5	81.1	88.7	79.9	78.7	78.3	87.5	86.3	86.0	87.3	78.9	78.6	68.3	63.5	74.0	43.5	80.8	71.0	77.2
Faster R-CNN+Resnet-101+M	81.0	80.2	70.6	81.2	88.7	79.8	78.7	77.9	**88.4**	85.8	86.3	87.6	78.6	78.7	68.2	63.9	73.9	43.5	80.5	71.2	77.2
Faster R-CNN+Resnet-101+T	81.0	80.3	70.7	81.3	88.6	80.2	78.7	78.3	87.4	86.2	86.0	87.6	78.8	78.8	67.9	63.9	74.2	43.7	81.3	71.1	77.3
Faster R-CNN+Resnet-101+TC	81.3	80.6	72.8	81.2	88.6	80.1	79.2	78.8	87.0	86.7	86.9	88.3	78.9	79.9	69.1	64.7	74.8	46.5	**82.6**	71.4	78.0
Proposal Faster R-CNN[2] [26]	87.1	88.6	74.4	87.5	88.7	86.8	87.3	86.4	78.2	87.6	88.0	89.3	78.5	86.6	74.7	65.3	76.1	57.9	78.6	78.6	81.3
Faster R-CNN+Inception-V3+A	87.5	88.9	76.5	88.5	89.2	87.4	87.1	86.6	79.6	88.5	86.4	89.6	79.6	86.9	**77.3**	66.9	75.5	57.4	78.5	79.6	81.9
Faster R-CNN+Inception-V3+M	87.4	88.9	76.4	88.4	89.2	87.4	87.1	86.7	79.6	88.5	86.4	89.6	79.6	86.9	77.2	66.9	75.6	57.2	78.4	79.7	81.8
Faster R-CNN+Inception-V3+T	87.7	88.8	76.8	88.7	**89.3**	87.4	87.2	86.7	79.5	**88.9**	87.0	**89.9**	**79.9**	86.9	77.2	**67.1**	75.7	58.1	79.0	79.9	82.1
Faster R-CNN+Inception-V3+TC	**88.3**	**89.1**	**77.8**	**88.8**	**89.3**	**87.5**	**87.5**	**87.2**	79.0	88.7	**88.4**	89.8	79.3	**87.2**	77.2	**67.1**	**76.1**	**60.3**	80.5	**80.3**	**82.5**

the data augmentation by adding the Pascal VOC 2007 trainval set and the Pascal VOC 2012 trainval set (referred to herein as Pascal VOC 2007+2012), resulting in approximately 16k images for training. We chose the Faster R-CNN architecture to extract the set of object proposals. We first fine-tuned the proposal network on the COCO [19] dataset for 80 categories of the object detection task. The detector was then fine-tuned on Pascal VOC 2007+2012 and tested on the Pascal VOC 2007 test set.

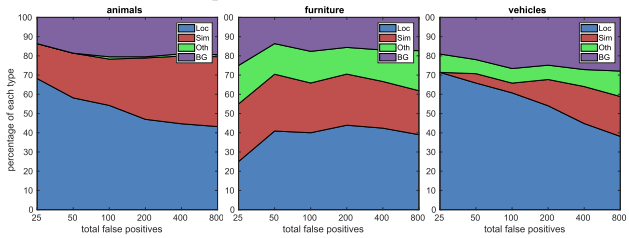
During training, we used an SGD [16] optimizer with a batch size of 1, a momentum of 0.9, and a weight decay of 0.0005. The images were resized to 600 × 1,024, and other preprocessing steps, such as means subtraction and random flipping, were also applied. With a base learning rate of 0.003, we trained the proposed model for 200,000 iterations.

Table 3 shows the results for the Pascal VOC 2007 test set. We use Faster R-CNN as the proposal network. In the first test (the first five rows of Table 3), the baseline is Resnet-101 [13] (pre-trained on ImageNet), and the second stage uses Resnet-101 as the classification network. The two-stage model with trained combination has a better mAP than the proposal network (Faster R-CNN) by 2.4%. In the second test (the last five rows of Table 3) with Faster R-CNN, which has a better baseline Inception-Resnet-V2, and the second stage network is Inception-V3. With trained combination, the mAP surpasses Faster R-CNN by 1.2%. Some categories are greatly improved (e.g., plant: 2.4%, bottle: 2.5%, boat: 3.4%).

Error analysis In order to scrutinize the effect of the second stage on the detection results, we used a tool from Ref. [14]. **Figure 4** shows the distribution of the top-ranked false positive types. We classify objects into three groups: animals (all animals + person), furniture, and vehicles. Compared with the proposal network results shown in Fig. 4 (a), the proportion of the error results from poor localization (a duplicate detection or a detection with



(a) Proposal network (Faster R-CNN).



(b) Two-stage detector using trained combination.

Fig. 4 Distribution of top-ranked false positive (FP) types for three groups, i.e., animals, furniture, and vehicles from the Pascal VOC 2007 test set. The top row shows the distribution of FPs on the proposal network results, and bottom row shows the distribution of FPs on the trained combination results. The FPs are divided into four error types: poor localization (Loc), confusion with a similar category (Sim), confusion with a dissimilar object category (Oth), confusion with the background (BG). False positives are sorted in descending order by the confidence score.

IOU overlapping the correct class between 0.1 and 0.5) of the two-stage detector in Fig. 4 (b) is higher. Since we do not modify the proposal localization, the result indicates that the combination results are better than the proposal network results for reducing misclassification (Sim, Oth, and BG). The vehicles group shows the significant improvement of the combination method for reducing misclassification errors. On the other hand, the animals group shows only a slight improvement.

Further analysis As discussed in Section 3.3, we proposed

Table 5 COCO results on the COCO_2014_minival set (bounding box AP). The subscript number indicates the IOU value, and AP without a subscript number indicates the AP at IOU from 50% to 95%. Moreover, S, M, and L indicate small, medium, and large objects, respectively.

	Baseline	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN [12]	Resnet-101-FPN [18]	37.5	60.6	39.9	17.7	41	55.4
Faster R-CNN	Resnet-101	37.8	55.7	42.5	15.5	43.5	57.6
Faster R-CNN+Inception-V3+A	Resnet-101	37.9	55.8	42.6	15.6	43.6	57.7
Faster R-CNN	Inception-Resnet-V2	43	60.7	48.4	19.1	49.2	64.7
Faster R-CNN+Inception-V3+A	Inception-Resnet-V2	**43.1**	60.8	**48.6**	**19.2**	49.2	64.7
Faster R-CNN+Inception-V3+T	Inception-Resnet-V2	**43.1**	**60.9**	**48.6**	19.1	**49.4**	**64.8**
Faster R-CNN+Inception-V3+TC	Inception-Resnet-V2	43	60.8	**48.6**	**19.2**	49.3	64.7

Table 4 Experimental results for different high score threshold settings. The results are evaluated on the Pascal VOC 2007 test set.

d	0.4	0.5	0.6	0.7	0.8	0.9
mAP	82.2	82.3	82.5	82.4	82.0	81.5

Eq. (5) to encourage detections that have matched labels between proposal and classification. In our experiment, we set $c = 1.4$ as a constant boosting weight. The parameter d depicts the high-score threshold. **Table 4** shows the effect of choosing the threshold. We try different high-score thresholds from 0.4 to 0.9 and find that the performance peaks at 0.6, indicating that beyond that value, the first stage confidence is sufficiently good and the final confidence does not require further boosting. On the other side, if the high-score threshold is below that value, the matching labels between two networks are not reliable enough to correctly predict labels.

4.3 MS COCO Object Detection

In this section, we performed experiments on the COCO dataset. COCO is a large-scale object detection dataset that has 80 object categories. The training dataset contains 118,287 images, including all training images and a subset of valuation images (coco_2014_train and coco_2014_valminusminival). We use COCO API [20] to evaluate our results, which are measured by mAP over IOU in various thresholds. The model results are tested on the minivaluation test (coco_2014_minival), which contains 5,000 images. Compared with Pascal VOC, objects in the COCO dataset tend to be smaller, and the number of objects is higher. Thus, we fine-tuned the COCO dataset with more iterations. For more details, we trained the proposal network with 1.2M iterations with a base learning rate of 0.0003 and reduce the learning rate by a factor of 10 at 900,000 iterations with a momentum of 0.9. We fine-tuned the classification network for 800,000 iterations with a base learning rate of 0.001.

The results are summarized in **Table 5**. Although the performance of the two-stage model depends on which proposal network is used, we observed that the performance is slightly improved in all settings. This indicates that the two-stage model still improves the proposal network performance on the COCO dataset.

5. Discussions

5.1 How to Choose between Model 1 and Model 2

The differences between model 1 and model 2 are the method of training the second stage and the combination method. We propose several combination methods and observe that the mean combination (f_1) and the multiply combination (f_2) are approximately equal. The combination with the threshold function (f_3) is better than f_1 or f_2 alone. However, the f_3 function depends on the high-score threshold and the boosting weight values, and thus is more difficult to optimize. The trained combination is better than f_3 on the Pascal VOC dataset, but is almost equal to f_3 on the COCO dataset, because the scores of the first stage affect the classification results of the second stage. The trained combination only works well on "high" accuracy proposals. For instance, at IOU = 0.5, the mAP on Pascal VOC, which is approximately 84% is much higher than the mAP on COCO, which is approximately 60%. (For reasons of comparison, we evaluated the Pascal VOC test set using the COCO evaluation tool.).

In the trained combination model (model 2), the second stage uses the output scores of the first stage as additional inputs. We conclude that model 2 needs a reliable proposal (high mAP) network to outperform the post-combination model (model 1). The selection between model 1 and model 2 depends on the specific dataset. As shown in the experiment section, both proposed models are better than given proposal networks. For high mAP proposal model (e.g., mAP > 75%) model 2 should be chosen. For not high enough mAP model (e.g., mAP < 60%), model 1 should be chosen. Otherwise, we need more experiments to choose between the two models.

5.2 How to Decide the Network for the First and Second Stage

Although the first stage has a strong impact on the final performance, choosing the second stage is also important. We analyze the method by which to decide the network for the first and second stage of the proposed model.

First, following Ref. [15], the baseline of the first stage is chosen based on its classification performance. However, the SSD meta-architecture appears to be less affected by its baseline's classification accuracy, while it significantly affects the Faster R-CNN meta-architecture performance. Second, because the second stage of the proposed model does not change the proposal location, we should use strong-localization-type detectors, rather than strong-classification-type detectors. For general object detection tasks, the Faster R-CNN is a considerably better choice than the SSD.

Finally, the second stage is chosen by balancing the performance and the model complexity. **Table 6** shows the properties of the feature extractors used in the second stage. We explore the relationship between the performance (mAP) on Pascal VOC 2007 and image classification accuracies on ImageNet, and the number of parameters of the feature extractors used to initialize

Table 9 Detection speed (s) and memory usage (GB) of the proposed models.

	Proposal network				Classification network			Total	
Dataset	model	baseline	time	mem.	baseline	time	mem.	time	mem.
Caltech ped.	SSD	VGG16	0.087	0.9	Inception-V1	0.014	0.9	0.10	1.8
Pascal VOC	Faster R-CNN	Resnet-101	0.106	2.2	Resnet-101	0.018	4.0	0.12	6.2
Pascal VOC	Faster R-CNN	Inception-Resnet-V2	0.602	10.6	Inception-V3	0.036	3.1	0.64	13.7
COCO	Faster R-CNN	Resnet-101	0.115	2.0	Inception-V3	0.036	3.1	0.15	5.1
COCO	Faster R-CNN	Inception-Resnet-V2	0.602	10.7	Inception-V3	0.036	3.1	0.64	13.8

Table 6 Feature extractor properties used in the second stage. The top-1 and top-5 accuracy (%) are the classification accuracies on ImageNet. The bounding box mAP is evaluated on Pascal VOC 2007 using the same proposal network (Faster R-CNN) and the same combination method (f_3).

Feature extractor	Num. Params.	Top-1 Acc.	Top-5 Acc.	mAP
Resnet-101	43M	76.4	92.9	81.9
Inception-V3	22M	78.0	95.2	82.4
Inception-Resnet-V2	54M	80.4	95.3	82.2

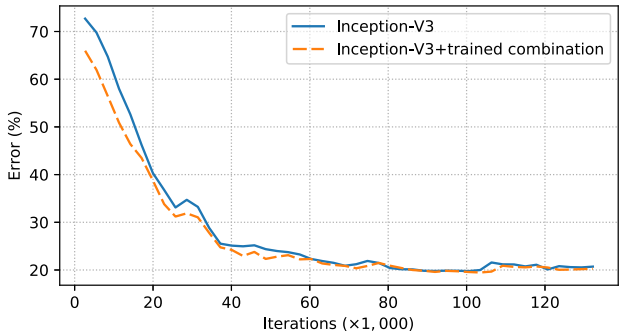


Fig. 5 Top-1 error evolution during training of Inception-V3 vs. an Inception-V3+trained combination module. The computational costs of the two models are approximately equal. The evaluation set is extracted from the Pascal VOC 2007 test set.

the second stage. Remarkably, although Inception-Resnet-V2 is the most accurate model on ImageNet, the two-stage model using Inception-Resnet-V2 has a lower performance than the two-stage model using Inception-V3. The Inception-V3 model also has less complexity than the Inception-Resnet-V2 model. Thus, the Inception-V3 appears to be the most relevant model for use in the second stage.

5.3 Efficiency of the Trained Combination

Since we analyzed how to use trained combination above, if the first stage is "good" enough, the trained combination is better than other combination methods. For this reason, we continue to explore the roles of the trained combination module in the second stage as a classifier on the Pascal VOC 2007 dataset. We train the Inception-V3 and Inception-V3+trained combination module with the same training set and learning rate, and then evaluate two models on the Pascal VOC 2007 test set. In **Fig. 5**, we compare the training errors of two classification models. The error drops more quickly on the Inception-V3+trained combination network, which indicates that the first stage confidence helps speed up the training classification network process.

5.4 Effectiveness of Small Object Detection

Deep learning performance suffers for small object detection. We believe that the reason for this is that low-resolution feature maps are used to handle small objects. The proposed model

Table 7 Miss rate comparison with state-of-the-art pedestrian detection methods on the Caltech test set for the detection of small objects.

Method	Small objects (30-50)	Far scale (20-30)
RPN+BF [34]	79.83	100
CompACT-Deep [4]	75.86	100
SA-FastRCNN [17]	74.49	100
MS-CNN [3]	70.34	97.23
F-DNN [7]	44.86	77.47
F-DNN+SS [7]	45.14	77.37
SmallDeep [21]	42.05	77.26
Proposed model	**41.56**	**74.92**

Table 8 Top improvement performance (AP) for small object detection. The results are evaluated on COCO minival using Faster R-CNN as the proposal network and Inception-V3 as the classification network.

Category	Faster R-CNN	Faster R-CNN+Inception-V3+A	Gap (%)
suitcase	11.5	13.0	1.5
stop sign	23.5	25.2	1.6
carrot	10.3	12.0	1.7
fire hydrant	25.2	27.4	2.2
sheep	21.1	23.5	2.4
toilet	8.6	11.5	2.9

works well for some individual object categories on small scale.

For the Caltech pedestrian task, we consider the "small objects" setting (or small pedestrian setting), because the miss rate increases drastically in the range of from 30 to 50 pixels for most pedestrian detectors [6]. The proposed method achieves state-of-the-art performance for small pedestrian detection, as shown in **Table 7**.

Likewise, in the COCO dataset, the largest gaps between the single model detector and the proposed two-stage network are in small object detection. The top six improvement categories for small object evaluation are listed in **Table 8**.

5.5 Time and Memory Analysis

We trained and tested the proposed model on a machine with 32 GB of RAM and a single GPU TITAN X. The models are implemented on a TensorFlow framework [1]. The proposed method comprises two stages, and thus the running time of the proposed model is the total running time of two stages. **Table 9** summarizes the speed of the proposed model. The running time of the SSD model is faster than that of Faster R-CNN. The running time of the trained combination method (for model 2) is the same as that of the post-combination method (for model 1) with the same baseline. Since the number of proposals is large, we perform non-maximum suppression (nms) to keep the 100 maximum suppression proposals per image. These proposals then are efficiently forwarded to the classification network with batch sizes of 32. The second stage running time is less than 24% of total running time; thus, most of the running time is expended on object proposal extraction. Note that the slowest second stage running time

is just 0.036 s per image, but the slowest proposed model is still far from real-time (30 frames per second or better).

Since the proposed model requires two networks, the memory usage of the model is the total memory usage of the proposal network and the classification network. Overall, the memory of each stage depends on the size of the feature extractors and whether meta architectures are used. In Table 9, we also report the memory usage for each stage of the proposed model.

6. Conclusion

In the present paper, we present a simple but effective two-stage deep neural network model for object detection. We use CNNs for object proposal extraction and classification as well as to reduce misclassification. The experimental results reveal that the connection between the detection confidence score and the classification confidence score is a key component to improving accuracy. We introduce several combination methods and prove the advantages of using a two-stage model for object detection. The trained combination method can improve the classification performance and detection performance of the overall model. We also discuss how to choose an appropriate first stage, second stage, and combination method in order to construct a strong two-stage detector for a specific application.

One drawback of the proposed method is time and space complexity, and, in the future, we hope to reduce the running time and memory usage of the model and make deployment of the model easier.

References

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: TensorFlow: A System for Large-Scale Machine Learning, *OSDI*, Vol.16, pp.265–283 (2016).

[2] Bengio, Y.: Practical recommendations for gradient-based training of deep architectures, *Neural Networks: Tricks of the Trade*, pp.437–478, Springer (2012).

[3] Cai, Z., Fan, Q., Feris, R.S. and Vasconcelos, N.: A unified multi-scale deep convolutional neural network for fast object detection, *European Conference on Computer Vision*, pp.354–370, Springer (2016).

[4] Cai, Z., Saberian, M. and Vasconcelos, N.: Learning complexity-aware cascades for deep pedestrian detection, *Proc. IEEE International Conference on Computer Vision*, pp.3361–3369 (2015).

[5] Dai, J., Li, Y., He, K. and Sun, J.: R-FCN: Object detection via region-based fully convolutional networks, *Advances in Neural Information Processing Systems*, pp.379–387 (2016).

[6] Dollar, P., Wojek, C., Schiele, B. and Perona, P.: Pedestrian detection: An evaluation of the state of the art, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.34, No.4, pp.743–761 (2012).

[7] Du, X., El-Khamy, M., Lee, J. and Davis, L.: Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection, *2017 IEEE Winter Conference on Applications of Computer Vision* (*WACV*), pp.953–961, IEEE (2017).

[8] Ess, A., Leibe, B., Schindler, K. and Van Gool, L.: A mobile vision system for robust multi-person tracking, *IEEE Conference on Computer Vision and Pattern Recognition*, *CVPR 2008*, pp.1–8, IEEE (2008).

[9] Everingham, M., Van Gool, L., Williams, C.K., Winn, J. and Zisserman, A.: The pascal visual object classes (VOC) challenge, *International Journal of Computer Vision*, Vol.88, No.2, pp.303–338 (2010).

[10] Geiger, A., Lenz, P. and Urtasun, R.: Are we ready for autonomous driving? The kitti vision benchmark suite, *2012 IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), pp.3354–3361, IEEE (2012).

[11] Girshick, R.: Fast R-CNN, arXiv preprint arXiv:1504.08083 (2015).

[12] He, K., Gkioxari, G., Dollár, P. and Girshick, R.: Mask R-CNN, *2017 IEEE International Conference on Computer Vision* (*ICCV*), pp.2980–2988, IEEE (2017).

[13] He, K., Zhang, X., Ren, S. and Sun, J.: Deep residual learning for image recognition, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp.770–778 (2016).

[14] Hoiem, D., Chodpathumwan, Y. and Dai, Q.: Diagnosing error in object detectors, *European Conference on Computer Vision*, pp.340–353, Springer (2012).

[15] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors, *IEEE CVPR* (2017).

[16] Krizhevsky, A., Sutskever, I. and Hinton, G.E.: Imagenet classification with deep convolutional neural networks, *Advances in Neural Information Processing Systems*, pp.1097–1105 (2012).

[17] Li, J., Liang, X., Shen, S., Xu, T., Feng, J. and Yan, S.: Scale-aware fast R-CNN for pedestrian detection, *IEEE Trans. Multimedia*, Vol.20, No.4, pp.985–996 (2018).

[18] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S.: Feature pyramid networks for object detection, *CVPR*, Vol.1, No.2, p.4 (2017).

[19] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L.: Microsoft COCO: Common objects in context, *European Conference on Computer Vision*, pp.740–755, Springer (2014).

[20] Lin, T. and Dollar, P.: MS COCO API (2016).

[21] Linh, T.D. and Masayuki, A.: A two-stage training deep neural network for small pedestrian detection, *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing* (*MLSP*), pp.1–6, IEEE (2017).

[22] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A.C.: SSD: Single shot multibox detector, *European Conference on Computer Vision*, pp.21–37, Springer (2016).

[23] Liu, W., Rabinovich, A. and Berg, A.C.: Parsenet: Looking wider to see better, arXiv preprint arXiv:1506.04579 (2015).

[24] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A.: You only look once: Unified, real-time object detection, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp.779–788 (2016).

[25] Redmon, J. and Farhadi, A.: YOLO9000: Better, faster, stronger, arXiv preprint (2017).

[26] Ren, S., He, K., Girshick, R. and Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks, *Advances in Neural Information Processing Systems*, pp.91–99 (2015).

[27] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge, *International Journal of Computer Vision*, Vol.115, No.3, pp.211–252 (2015).

[28] Simonyan, K. and Zisserman, A.: Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[29] Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning, *AAAI*, Vol.4, p.12 (2017).

[30] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al.: Going deeper with convolutions, *CVPR* (2015).

[31] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z.: Rethinking the inception architecture for computer vision, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp.2818–2826 (2016).

[32] Uijlings, J.R., Van De Sande, K.E., Gevers, T. and Smeulders, A.W.: Selective search for object recognition, *International Journal of Computer Vision*, Vol.104, No.2, pp.154–171 (2013).

[33] Wojek, C., Walk, S. and Schiele, B.: Multi-cue onboard pedestrian detection, *IEEE Conference on Computer Vision and Pattern Recognition*, *CVPR 2009*, pp.794–801, IEEE (2009).

[34] Zhang, L., Lin, L., Liang, X. and He, K.: Is faster R-CNN doing well for pedestrian detection?, *European Conference on Computer Vision*, pp.443–457, Springer (2016).

Tran Duy Linh is a Ph.D. student at Graduate School of Science and Engineering, Teikyo University, Japan. He received his B.E. degree in Information System from the Hanoi University of Science and Technology, Hanoi, Viet Nam in 2010 and received his M.S. degree in Computer Science from Ho Chi Minh City University of Technology, Ho Chi Minh City, Viet Nam, in 2015. His research interests include image and video processing. He is presently engaged in research on object detection using deep learning. He is a member of the Information Processing Society of Japan.

Masayuki Arai is a professor in the Graduate School of Sciences and Engineering at Teikyo University. He received his B.E. degree from Tokyo University of Science in 1981 and Dr.Eng. degree from Utsunomiya University in 1995. His research interests include pattern recognition, natural language processing and information visualization. He is a member of the Information Processing Society of Japan and IEEE.