ME R-CNN: Multi-Expert R-CNN for Object Detection

Hyungtae Lee, *Member, IEEE,* Sungmin Eum, *Member, IEEE,* and Heesung Kwon, *Senior Member, IEEE*

***Abstract*—We introduce Multi-Expert Region-based Convolutional Neural Network (ME R-CNN) which is equipped with multiple experts (ME) where each expert is learned to process a certain type of regions of interest (RoIs). This architecture better captures the appearance variations of the RoIs caused by different shapes, poses, and viewing angles. In order to direct each RoI to the appropriate expert, we devise a novel "learnable" network, which we call, expert assignment network (EAN). EAN automatically learns the optimal RoI-expert relationship even without any supervision of expert assignment. As the major components of ME R-CNN, ME and EAN, are mutually affecting each other while tied to a shared network, neither an alternating nor a naive end-to-end optimization is likely to fail. To address this problem, we introduce a practical training strategy which is tailored to optimize ME, EAN, and the shared network in an end-to-end fashion. We show that both of the architectures provide considerable performance increase over the baselines on PASCAL VOC 07, 12, and MS COCO datasets.**

***Index Terms*—multiple experts, object detection, R-CNN, expert assigner**

I. Introduction

In general, object detection uses distinctive shape patterns as evidence to find the object-of-interest in an image. Object detection models are trained on these shape patterns that are commonly shown within the same object categories yet discriminative among the different categories. However, it is quite burdensome for a single model to accurately identify all the appearances since object appearances greatly vary according to fundamental object shape priors (e.g., airplane vs. person) as well as different object poses and viewing angles (e.g., a person lying down vs. standing upright). Therefore, conventional object detection methods often use mixture of experts, each expert associated only with the corresponding shape patterns, in order to better capture large variations of object appearance [1]–[3].

In this paper, we introduce a novel convolutional neural network (CNN)-based approach for object detection, referred to as ME R-CNN, which adopts multiple experts (ME). The ME R-CNN inherits the architecture of the region-based CNN (R-CNN) [4]–[11] which uses a single stream pipeline for processing each region-of-interest (RoI). However, unlike these approaches, the ME R-CNN is equipped with multiple stream pipelines, where one of the pipelines becomes an "expert" for processing a certain type of RoIs. To designate incoming RoIs to the appropriate experts, we construct a novel network called Expert Assignment Network (EAN). Figure 1 depicts the conceptual mechanism of the ME R-CNN which contains ME and EAN components.

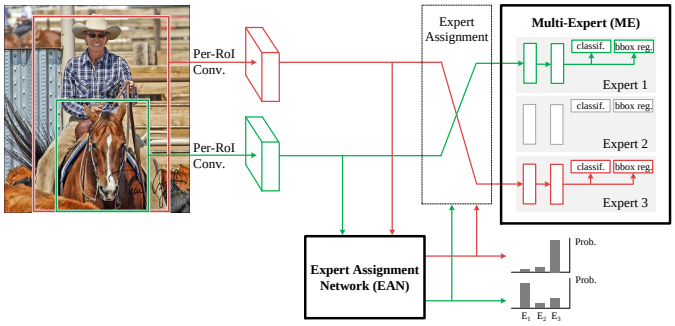


Fig. 1: ME R-CNN. ME R-CNN adopts "multi-expert (ME)" to allow different streamlines for processing different RoIs. The "expert assignment network (EAN)" is built into the architecture to select the optimal streamline for each RoI. EAN is designed to output the probabilities for the RoI-expert relationships which guides the expert assignment.

The EAN is a "learnable" network which is trained to capture the RoI-expert relationship. It is designed to output a vector which indicates the matching probability for selecting one of the experts for a given RoI. The EAN consists of a convolutional, average pooling, and a fully connected layer. In the training scheme, the EAN is learned to choose the expert with minimum expert loss. The expert loss is defined as the sum of the losses for object classification and bounding box regression. These two losses were introduced in [5], [6] to optimize the R-CNNs.

Training ME R-CNN is very challenging because: (1) ME and EAN components mutually affect each other (i.e., EAN training labels are defined based on the expert loss in ME, while EAN distributes the training samples to ME), (2) they are both derived from a shared network. To deal with (1), it is natural to use an alternating optimization strategy to co-train two mutually affecting tasks. However, this approach is likely to fail since the weights in the shared network optimized with respect to ME would no longer be in sync with EAN, and vice

Manuscript received July 9, 2018; revised April 15, 2019 and June 14, 2019; accepted August 27, 2019.

H. Lee and H. Kwon are with the Intelligent Perception Branch, the Computational & Information Sciences Directorate (CISD), Army Research Laboratory, Adelphi, MD, 20783 USA (e-mail: {hyungtae.lee, heesung.kwon}.civ@army.mil).

S. Eum is with Booz Allen Hamilton Inc., McLean, VA, 22102 USA and with the Intelligent Perception Branch, the Computational & Information Sciences Directorate (CISD), Army Research Laboratory, Adelphi, MD, 20783 USA (e-mail: eum_sungmin@bah.com).

versa because of (2). To cope with these structural issues, it is necessary to come up with an end-to-end approach where all three components (ME, EAN, and the shared network) are optimized together.

However, there is a barrier which hinders the joint training of all three components. During the joint training, the RoI-expert relationships which are dependent upon the ME losses will be altered continuously, thus providing inconsistent (i.e., severely fluctuating) expert labels for the training of EAN. Therefore, we have devised an approach to go around this issue by adding a step where EAN and ME are learned separately to gain stability before proceeding into the joint learning stage. In this step, ME weights are firstly learned along with the shared network and then the RoI-expert relationships are learned by the EAN based on the pre-trained ME. Adding this ME/EAN initialization step was found to be useful in providing relatively consistent expert labels for EAN training, thus effectively assisting the joint optimization step which follows the initialization step. Note that, as expert labels for the RoIs do not exist at the time of initializing the ME, we have provided hard-coded labels for the RoIs which will be accounted for in the following paragraph.

On top of advancing the object detection performance via employing multiple experts, ME R-CNN optimization can also be viewed as investigating the RoI-expert relationships in an unsupervised fashion (i.e., clustering) because no expert labels are ever provided as ground truth. Generally, clustering results highly depend on how initial cluster labels are assigned [12]. We have observed that the ME R-CNN learning is also highly sensitive to the presetting of the initial expert labels. When the expert labels are assigned by randomly initialized ME weights, most RoIs resulted in having assigned to one single expert after few training iterations, thus causing a failure in training. To avoid such extremely biased assignment, we also have tried forcing the RoIs to evenly be distributed to the experts for every training iteration, which also was found to be ineffective. Meanwhile, promising results were shown when initial assignments for the RoIs were done according to their aspect ratios.

We use Fast R-CNN [5] and Faster R-CNN [6] for drawing up the baseline architecture of ME R-CNN. The Fast/Faster R-CNN architecture is composed of a network which intakes and processes holistic images (per-image network) which is followed by another network responsible for processing the RoIs (per-RoI network). For Faster R-CNN, the RoIs are generated by the region proposal network (RPN) while Fast R-CNN employs additional selective search process to obtain a set of RoIs. While ME R-CNN directly inherits the per-image network portion (and the RPN from the Faster R-CNN), the latter portion (per-RoI network) is replaced by our novel components: ME and EAN. We verified that ME R-CNN can consistently provide considerable performance boost over the baseline approaches in PASCAL VOC 07, 12, and MS COCO datasets.

The contributions of the proposed ME R-CNN can be summarized as follows:

1) Introduction of ME R-CNN adopting "multiple experts (ME)" to better capture variations of the object appearance.
2) Introduction of the EAN which can "learn" the RoI-expert relationship.
3) Introduction of a practical training strategy to co-learn ME and EAN with a shared network in an end-to-end fashion.
4) Considerable performance boost over the baselines on benchmark datasets.

In Section II, we list out previous relevant literature and briefly introduce the innovative aspects of our approach. ME R-CNN architecture and its optimization strategy are described in Section III and IV, respectively. Evaluation results are provided in Section V. We suggest future works in Section VI and provide the conclusion in Section VII.

II. Related Works

A. Object Detection.

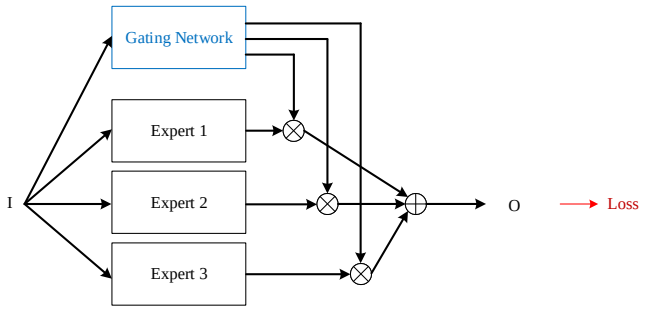
Object detection is one of the most challenging tasks in computer vision. Prior to the introduction of CNNs, non-CNN based object detection approaches, such as HOG-SVM (Histogram of Oriented Gradient - Support Vector Machines), DPM (Deformable Part Models), etc., were widely used for classifying RoIs into corresponding object categories [1]–[3], [13]. Within the past several years, multiple attempts have been made to use CNNs for object detection. Prominent methods among them are R-CNN [4] and its descendants [5], [6], [8], [10], [14] that provided the state-of-the-art performance for both localization accuracy and speed.

Although having achieved the top-notch performance, R-CNNs have not yet exploited some of the effective strategies which conventional object detection methods commonly use for boosting the performance. While the R-CNNs rely on heuristics to select hard negative examples, Shrivastava et al. [15] and Wang et al. [16] used the online hard example mining (OHEM) to automatically select hard examples with high optimization loss in every iteration of training. These approaches were motivated by the offline bootstrapping idea for training a classical object detection method [13].

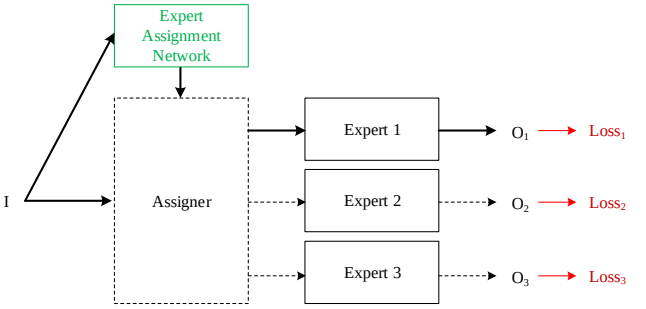
Motivated by their successful practice, we focus on adding another conventional, yet effective, "multi-expert" flavor to the R-CNN architecture. Felzenswalb et al. [3] and Malisiewicz et al. [2] have shown that employing multiple classifiers for the object detection task brings increase in performance.

B. Mixture-of-Experts Models.

Multiple experts embedded in the proposed ME R-CNN is based on the concept of mixture-of-experts models. The mixture-of-experts model is used to better estimate the probability distribution of a composite data with large variation (e.g., Gaussian mixture model [17]). In the image domain, object appearances can also show large variations according to their shapes, poses, and viewing angles. Felzenswalb et al. [3] nicely illustrates the importance of using a mixture of models by presenting two models, each of which captures the appearance of the front and the side view of a bicycle. Accordingly, many approaches [3], [18], [19] have shown



(a) Conventional way of adopting Mixture-of-Expert with CNN



(b) ME R-CNN

Fig. 2: Conventional way of adopting mixture-of-expert with CNN vs. ME R-CNN. Thick lines indicate the computational flow where one input has to go through. Only one of the three experts are activated in (b) whereas all the experts constantly have to be active to process one single example. I and O denote input and output, respectively. Assigner (dashed box) in ME R-CNN is not involved in training.

that using the mixture-of-experts model for advanced object detection is very effective.

Recently, there have been several attempts to adopt the mixture-of-experts model in CNN-based recognition approach [20]–[25]. Figure 2 shows how our way (ME R-CNN) of adopting mixture-of-experts in a CNN architecture is different from the conventional approach. In the conventional approaches (Figure 2 (a)), all the experts are involved in processing each input while the gating network provides different weights to adjust the outputs of the experts. On the other hand, with ME R-CNN (Figure 2 (b)), only one expert assigned by the EAN is activated, which makes it more efficient in terms of computation. Conventionally, all the experts are optimized with a same objective loss, but each expert in the ME R-CNN is optimized separately with its own loss in order to have unique expertise. To ensure that the overall performance does not suffer because of using only one expert at a time (instead of utilizing multiple experts each time), all the experts in the ME R-CNN architecture are trained concurrently in order to promote having experts with complimentary roles.

C. Going Wider with CNN.

One of the major innovations introduced in ME R-CNN is that the network has expanded in width, where the network width refers to the number of nodes in each layer. This is to equip the network with multiple number of specialized experts to better capture variations of object appearance. There have already been several attempts where the width of CNN architecture was expanded. Krizhevsky et al. [26] splits each layer into two parallel layers in order to fully use two GPUs in a parallel fashion. Girshick [5] appended two parallel layers with different functionalities at the end of the network, where the two layers are working for object category classification and bounding box regression, respectively. Szegedy et al. [27] uses the inception module which employs multiple parallel layers in order to make use of dense sets of different sized convolutional filters. Xie et al. [28] modified the residual network structure [29] by replacing each residual module with multiple parallel sets of layers (i.e., branches). They referred to the number of branches as "cardinality" and show that increased cardinality of the network enhances the image classification performance. Several other approaches [30]–[38] also introduced widened networks for the task of co-learning multiple tasks in a single framework. Wang et al. [39] introduces the way of increasing model capacity such as depth or width during finetuning. ME R-CNN also enhances object detection accuracy by increasing the model width during finetuning.

D. Unsupervised Learning (Clustering)

Training ME R-CNN can be viewed as a type of a clustering approach because no expert labels are available at the time of training. For the cases where labels are not provided for a classification problem, many clustering methods are used such as K-means clustering [12], [40], [41], mean-shift clustering [42]–[44], density-based spatial clustering [45]–[47], expectation-maximization (EM) clustering [48]–[50], agglomerative hierarchical clustering [51], [52]. Most clustering methods start off the process with a label initialization step (e.g., random assignment) where initial cluster labels are assigned to each of the samples. How the labels are assigned initially may bring a significant impact in terms of clustering performance. For learning the ME R-CNN, we initially assigned the RoIs to multiple experts according to their aspect ratios. This initialization approach was found to be effective in avoiding the optimization divergence.

III. ME R-CNN

In this section, we first introduce Faster R-CNN [6] as ME R-CNN inherits its structural backbone. Then the architectural components (ME and EAN) unique to ME R-CNN are elaborated in the following subsections. Lastly, we briefly describe how the network performs the task of object detection. The overall architecture of ME R-CNN is depicted in Figure 3.

A. Faster R-CNN

Faster R-CNN consists of a **Per-Image Network** and a **Per-RoI Network**. The **Per-Image Network** can be divided into two parts: a set of convolutional layers (Conv-L) and a region proposal network (RPN). When an input image goes through the Conv-L, a per-image convolutional feature map is generated which is then fed into the RPN. The RPN is used

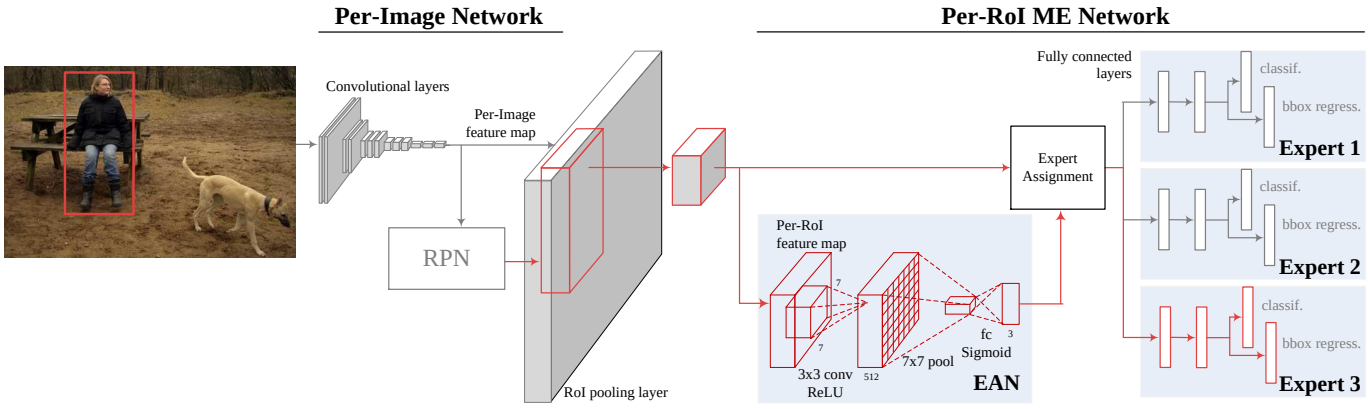


Fig. 3: ME R-CNN architecture. One example of a RoI-to-expert assignment is shown in red arrows.

to provide a set of hypothetical regions of interest (RoIs) for potential object regions.

For all the RoIs from the RPN, RoI pooling layer crops out the corresponding regions from the per-image feature map. Each of these cropped-out feature maps (per-RoI feature maps) are max pooled to have a fixed size output. The output size is set to match the input size of the first fully-connected layer of the predefined CNN (e.g., 7×7 for VGG16 [53]).

All of these per-RoI feature maps are then fed into the **Per-RoI Network**. At the end of this network, two sibling layers are present for object classification and bounding box regression. Object classification and bounding box regression are optimized using softmax classification loss and smooth L_1 loss, respectively.

B. Per-RoI ME Network

ME R-CNN resembles the overall architecture of Faster R-CNN as it contains both **Per-Image Network** and **Per-RoI Network**. **Per-Image Network** of ME R-CNN inherits all the components from that of the Faster R-CNN. However, **Per-RoI Network** portion is redesigned to fit the need of performing multi-expert supported object detection, and therefore, renamed as **Per-RoI ME Network**. There are multiple stream pipelines built into the **Per-RoI ME Network**, and each stream carries equivalent components in the **Per-RoI Network** of Faster R-CNN. Each of the stream, known as an "expert", is responsible for processing a certain type of RoIs. To guide the RoIs to their best matching experts, we have constructed a network called Expert Assignment Network (EAN). Each expert is connected to its two loss functions for object classification and bounding box regression. Although our design does not constrain the number of experts, we have exploited three experts to be used for the following experiments and illustrations.

C. Expert Assignment Network (EAN)

For each RoI, its associated per-RoI feature map is fed into one of the three experts which is assigned by the EAN which is designed to "learn" the RoI-expert relationship. The EAN consists of two learnable layers, one convolutional and one fully connected layer. The input to the EAN are the per-RoI feature maps. When using VGG16 as the baseline architecture, the size of a feature map is 7×7×512. The convolutional layer employs 512 kernels (3×3) with 1 stride and 1 padding. Then ReLU is applied which is followed by 7×7 max-pooling generating a 1×1×512 output. A fully connected layer then takes this and generates 3 dimensional output, where each bin indicates the score for the corresponding expert. To allow the EAN to be able to select more than one expert for training, a binary sigmoid function is applied to the output. Architecture of the EAN is depicted in Figure 3.

Assume that $f(x, W_{EAN})$ is the function which computes the output of the EAN with weight W_{EAN} when given per-RoI feature map x. Note that we denote each entry of the EAN output as $f^{(e)}$ which indicates the assignment probability for the expert e. The expert assignment process which is being carried out by the EAN can then be formularized as:

$$e^* = \arg \max_{e \in \{E_1, E_2, E_3\}} f^{(e)}(x, W_{EAN}). \quad (1)$$

EAN weights are optimized by minimizing the loss $L_{EAN}(\cdot)$ which intakes the EAN output generated by $f(\cdot)$ and the expert label vector $\mathbf{y}$ as shown below:

$$W^*_{EAN} = \arg \min_{W_{EAN}} L_{EAN}(f(x, W_{EAN}), \mathbf{y}) \quad (2)$$

The expert label vector $\mathbf{y}$ is constructed by concatenating the expert labels as $\mathbf{y} = [y_{E_1}, y_{E_2}, y_{E_3}]$. Since the purpose of EAN is to find an expert which best performs in terms of object detection, each expert label y_e is defined based on the expert loss L_e as shown below:

$$y_e = \begin{cases} 1 & \text{if } L_e(\{x, y_{obj}\}, W_e) \leq \tau \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

where y_{obj}, W_e, and τ denote, respectively, object category label, expert weight, and assignment threshold. L_e, which is the expert loss for expert e, is computed as the sum of corresponding softmax classification loss and smooth L_1 bounding box regression loss. The assignment threshold τ is defined by the mean value of all the expert losses for each per-RoI feature map x.

D. Object Detection

In testing, a per-image convolutional map is generated by feeding an input image through the convolutional layers. Using this map and the RoIs provided by the RPN, per-RoI feature maps are acquired which all are then fed into the EAN. Each per-RoI feature map is sent to one of the experts which corresponds to the maximum EAN score. Note that, EAN is trained with the binary labels for each RoI-expert assignment, which allows having each RoI being assigned to more than one expert at a time. This training strategy eventually prevents performance degradation even when a RoI is not assigned to the most desirable expert in the testing phase. This can be intuitively seen as preparing more than one experts which can properly function with confusing RoIs. In testing, each RoI is assigned to only one expert in order to achieve the same level of computational complexity as single expert model.

ME R-CNN outputs three sets of detection results per image, i.e., bounding boxes and their scores, from three different experts. The bounding boxes are refined by incorporating the output of bounding box regression layers. We combine these three sets of detection results and apply non-maximum suppression (NMS) with overlap criteria of 0.3 for each object category.

Note that, computational load which is required to process each RoI using ME R-CNN is comparable to the case when Faster R-CNN is used because only one expert is activated for an RoI in ME R-CNN while EAN adds negligible computational cost. Therefore, as long as the same number of RoIs are used, computational costs for the ME R-CNN and Faster R-CNN are highly similar.

IV. Learning ME R-CNN

We base our training strategy on the pragmatic "4-step alternating optimization" devised by Ren et al. [6] but modify it to fit the need of the components in the ME R-CNN. In the first step, we train the RPN. As RPN takes the output of the Conv-L, we use the ImageNet-pre-trained model to initialize the Conv-L for this step. In the second step, we train ME, EAN, and Conv-L using sets of region proposals generated by the step-1 RPN. Conv-L is again initialized by the ImageNet-pre-trained model. In the third step, we re-train the RPN to be aligned with the newly trained Conv-L from step-2. Finally, we update ME and EAN with step-2 Conv-L and step-3 RPN. In this step, ME and EAN weights are initialized by step-2 ME and EAN, respectively. Note that in step-3 and step-4, Conv-L is fixed.

Overall training strategy for the ME R-CNN is listed out in Algorithm 1. In the following subsections, we elaborate on step-2 and step-4 which are devised to train the components (ME, EAN, and Conv-L) unique to the ME R-CNN. For training the RPN in step-1 and step-3, we have followed the procedures introduced in [6].

Algorithm 1: 4-step alternating algorithm

1	Train RPN & Conv-L
	(Conv-L from 1 is no longer used hereafter.)
2	Train ME, EAN, & Conv-L
2a	Initialize ME & Conv-L
2b	Initialize EAN
2c	Co-train ME, EAN, & Conv-L
3	Train RPN
4	Update ME & EAN
4a	Update EAN
4b	Update ME

A. Step-1: Train RPN & Conv-L

RPN evaluates all spatial locations from the per-image convolutional feature map. Every location in the map is mapped to multiple windows which are predefined in sizes (e.g., 8×16, 16×8, 8×8, etc). Each mapping reference is referred to as an "anchor". For PASCAL VOC, three scales and three aspect ratios, represented with 9 anchors, are considered to obtain region proposals via RPN. These anchors are 16×8, 8×8, 8×16, 32×16, 16×16, 16×32, 64×32, 32×32, and 32×64. For MS COCO, four more anchors (8×4, 4×4, and 4×8) are added to consider objects with extremely small size.

For training RPN, positive/negative examples are chosen from all possible windows according to the IOU (intersection-over-union) between the window bounding boxes and the groundtruth bounding boxes of any objects. Windows with IOUs larger than 0.7, are treated as positive training examples. Windows with IOUs between 0.7 and 0.3, are used as negative examples. RPN is trained using mini-batches, where each mini-batch consists of 128 positive and 128 negative examples.

The trained RPN is used to provide the region proposals in training ME, EAN & Conv-L (step-2). In order to provide more accurate region proposals, Conv-L is also trained in step-1. However, the Conv-L weights learned in this step are no longer used afterwards.

B. Step-2: Train ME, EAN, & Conv-L

In this subsection, we introduce a novel way to simultaneously train the major components (ME, EAN, and Conv-L) in the ME R-CNN which are layed out differently when compared with other types of multi-task learning architectures. Many multi-task learning architectures [30]–[32] are designed so that the tasks are independently derived from the commonly shared network as shown in Figure 4(a). Another stream of multi-task architectures do not require shared networks while having mutually-affecting tasks as depicted in Figure 4(b). The Generative Adversarial Network (GAN) [54] can be considered as a representative example of an architecture which contains mutually-affecting multiple tasks ('Generator' and 'Discriminator'). Shared network-driven multiple task architectures can be trained by enforcing separate loss functions in an end-to-end fashion, while the architecture with mutually-affecting tasks are typically trained using an alternating optimization strategy as the module responsible for one task needs to be fixed to train the other.

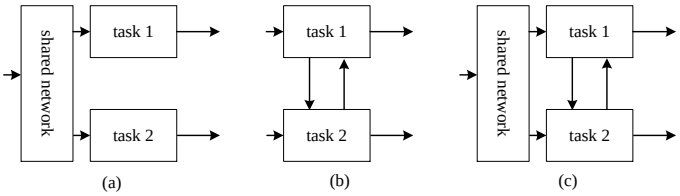


Fig. 4: Network architecture comparison. (a) Shared network-driven multiple task (b) Mutually-affecting multiple task (c) Shared network-driven, mutually-affecting multiple task (In ME R-CNN, shared network: Conv-L, tasks 1 & 2: EAN & ME)

Unlike the two types, the architecture of ME R-CNN can be viewed as having tasks (ME and EAN) derived from the shared network (Conv-L), and at the same time, mutually affecting each other. Figure 4(c) illustrates the schematic architecture of ME, EAN, and Conv-L. This makes the training very challenging as neither of the previously mentioned training approaches can be applied directly.

Attempting to train these modules using an alternating optimization strategy is far from reaching the optimal point and likely to fail. For instance, the shared network portion constantly changes every time task 1 is being optimized, but at the same time loses its sync with respect to task 2. The problem is that previously optimized task 2 is no longer optimized with respect to the shared network but still harmfully affects task 1.

To cope with these structural issues, all three components (Me, EAN, and Conv-L) need to be optimized together. However, when carrying out the joint learning, constantly changing ME provides inconsistent (severely fluctuating) expert labels, which adversely impacts EAN training. To gain stability before proceeding into the joint learning (Step-2c), we add an initialization step where ME weights along with Conv-L are learned (Step-2a) which is then followed by the EAN learning (Step-2b) in which the pre-trained ME weights are fixed. We have observed that these initialization steps were effective in providing relatively consistent expert labels for the EAN training and thus accommodating better grounds for the joint optimization as shown in Table IX.

Initialize ME & Conv-L. When initializing ME and Conv-L independent of EAN (i.e., without any expert assignment information from EAN), temporarily exploiting RoI shape-based assignment criteria in place of EAN was found to be effective. Each RoI is labeled with a shape category chosen among horizontally elongated (**H**), square-like (**S**), or vertically elongated (**V**) according to its aspect ratio.

We denote w and h as the width and the height of an RoI. All RoIs satisfying $w > h$ are assigned to the **H** category. All RoIs satisfying $w < 2h$ and $w > \frac{1}{2}h$ are assigned to the **S** category. Lastly, RoIs with $w < h$ are assigned to the **V** category. Note that, under this RoI assignment criteria, an RoI can be categorized into more than one category. This is done to have multiple experts responsible for the RoIs which can be shared across the different categories while training the network. We trained three different experts using the RoIs assigned to **H**, **S**, and **V** category, respectively.

To optimize the three experts, three batches are prepared for every iteration. Each batch is built from two images, and each image contributes 64 randomly chosen RoIs. For each expert, only the RoIs that match its associated shape category are selected for training. Each RoI is labeled as a positive or negative example according to an IOU overlap criteria between the RoI and the groundtruth bounding box. The RoIs having IOU overlap equal to or bigger than 0.5 are labeled as positive examples and the remaining ones are labeled as negative. For each batch, the ratio between the number of positive and negative examples is fixed as 1:3. Batch preparations for all of the training procedures in this paper (except B in 'Co-learn ME, EAN & Conv-L') are equivalently done as described in 'Initialize ME & Conv-L'.

Conv-L and ME are each finetuned from the convolutional layers and the fully connected layers of the ImageNet-pretrained model, respectively. Three pairs of sibling layers (classification and bounding box regression) appended at the end of the expert streams are initialized as well. The classification layer weights are initialized by randomly selecting them according to Gaussian distribution with the mean of 0 and the standard deviation of 0.01. For the bounding box regression layer, we initialized the weights randomly selected from Gaussian distribution with the mean and the standard deviation of 0 and 0.001, respectively. When finetuning Conv-L, we multiply 1/3 to the base learning rate because optimizing the layers in Conv-L is affected by all three streams of ME at each training iteration when back-propagation takes place. This scheme of multiplying 1/3 to the base learning rate in finetuning Conv-L is also used when we co-learn ME and EAN.

Initialize EAN. The weights in EAN are learned according to Equation 2 while fixing weights in Conv-L and ME. The fixed weights for Conv-L and ME are inherited from the results of 'Initialize ME & Conv-L'. Note that before learning EAN, the weights (both for convolutional and fully connected layers) are initialized by randomly selecting them according to Gaussian distribution with mean and standard deviation of 0 and 0.01, respectively.

Co-learn ME, EAN & Conv-L. The overall protocol of co-training ME, EAN & Conv-L is depicted in Figure 5. Let us assume that the weights for ME (same applies to EAN and Conv-L) learned in the previous and current iteration are denoted as ME_{t-1} and ME_t, respectively. Once RPN generates a set of region proposals, the whole set of corresponding per-RoI feature maps (B) is forward passed into ME_{t-1} and EAN_{t-1}.

Forward passed output from ME_{t-1} is then fed into the EAN Batch Sampler which outputs a downsized batch (B_{EAN}) which contains per-RoI feature maps selected based on the expert loss L_e. Note that L_e, which is the expert loss for expert e, is the sum of corresponding softmax classification loss and smooth L_1 bounding box regression loss. B_{EAN} is then used to train EAN_t. The ground truth labels for the samples in the batch are defined according to Equation 3.

In a similar manner, output of EAN_{t-1} is fed into the

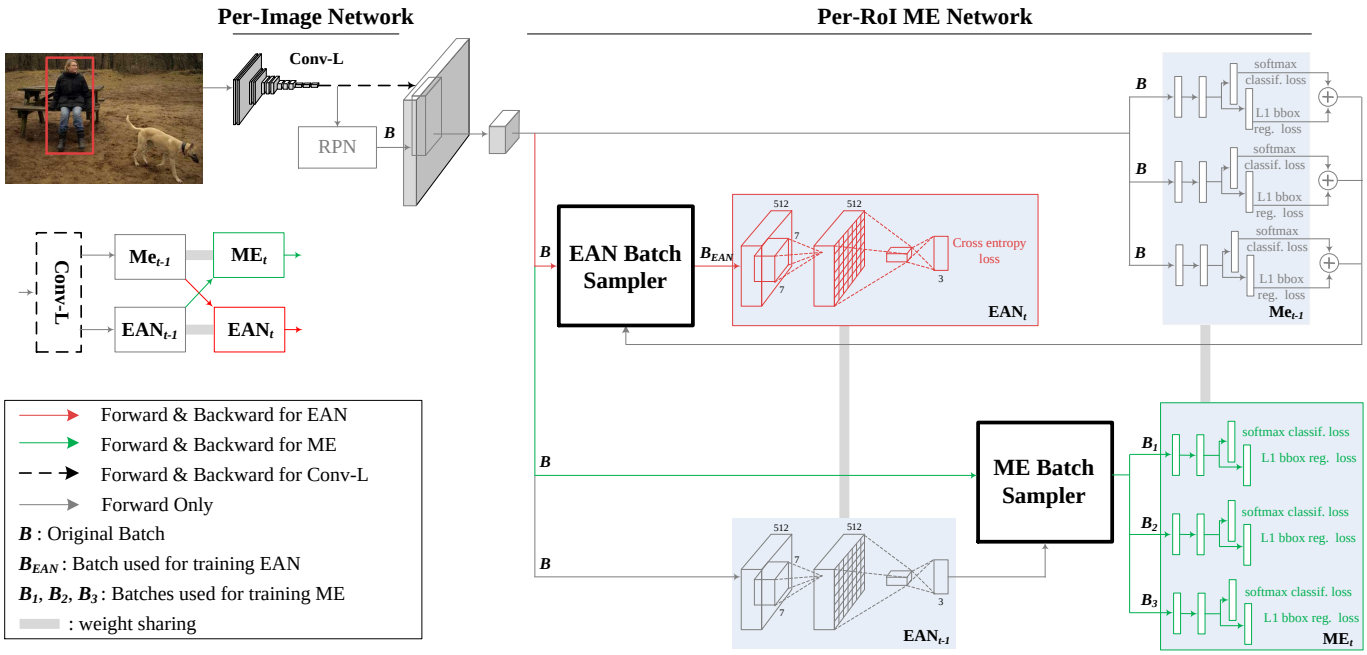


Fig. 5: Protocol of co-training ME, EAN & Conv-L. A conceptual illustration of the entire flowchart is shown below the input image.

ME Batch Sampler which generates equal-sized three sets of batches, B_1, B_2, and B_3 which will be used to train ME$_t$. In the ME Batch Sampler, each batch is generated by collecting the per-RoI feature maps and their associated object category labels which were tagged with 0.8 or higher EAN output probability. EAN output is a 3-dimensional vector where each entry indicates the matching probability for the corresponding expert among three.

Along with the training of EAN$_t$ and ME$_t$, Conv-L is concurrently trained via back-propagation. For the very first iteration of training, ME$_0$ and EAN$_0$ are used in place of ME$_{t-1}$ and EAN$_{t-1}$, respectively, which are acquired by previously mentioned 'Initialize ME & Conv-L' and 'Initialize EAN'.

C. Step-3: Train RPN

In this step, RPN is finetuned from the model trained in step-1. Unlike step-1, conv-L weights are not updated in order to share them among RPN, EAN, and ME. All the details (i.e., hyper-parameters, anchors, training example labeling) are same to those used in step-1. Updated RPN generates the region proposals used in updating EAN and ME in step-4.

D. Step-4: Update EAN & ME

We generate the region proposals using the RPN learned in step-3 to update the weights in EAN and ME. This is carried out to better align EAN and ME with respect to the newly trained RPN and maximize the performance. EAN weights are updated first while fixing ME weights, and then ME is updated by fixing EAN. This alternating update procedure for EAN and ME is feasible for this step since Conv-L is fixated with the previously learned weights.

E. Joint Training of All Components

4-step alternating optimization is a sub-optimal approach. As Ren et al. [9] recently provided a joint training strategy for learning the Faster R-CNN, we also have tried a single-step joint training of all the components (ME, EAN, and Conv-L) for our network. However, the jointly trained ME R-CNN did not perform well in terms of detection accuracy. As previously mentioned (Section I and Section IV-B), jointly optimizing ME R-CNN in an end-to-end fashion is highly sensitive to the presetting of RoI-expert assignment. As our future work, we will seek to develop a more efficient and effective learning strategy for ME R-CNN.

V. Experimental Evaluation

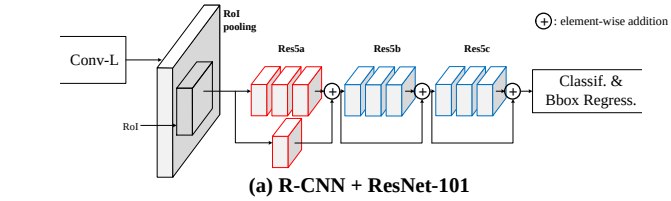
In this section, we evaluate our method on VOC 2007, 2012 [55] as well as MS COCO [56] dataset.

A. Experimental Setup

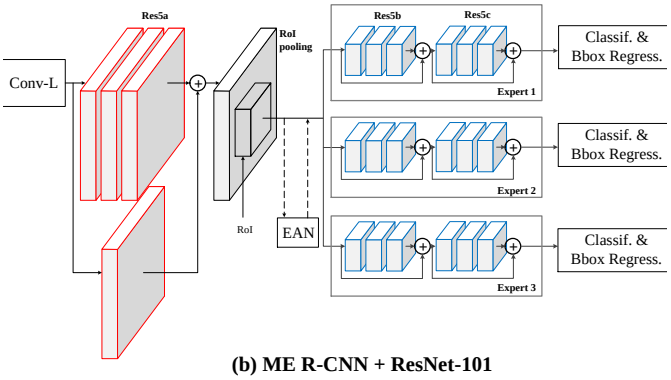
We use VGG16 [53] or ResNet-101 [29] as a predefined CNN for all experiments. We also use Fast R-CNN [5] or Faster R-CNN [6] as our baseline architectures for the ME R-CNN. When Faster R-CNN is chosen as the base architecture for ME R-CNN, all 4-steps are carried out. However, when Fast R-CNN is used, single-step end-to-end optimization (i.e., step 2 of ME R-CNN optimization) is carried out as RPN-related training procedures are not required. For all the methods we have tested, we used stochastic gradient descent with a base learning rate of 0.001 and the weight decay of 0.1. As reported in Table I, the minibatch iterations and the step sizes were varied according to trainsets and training steps. The mini-batch iteration and step size of the ME R-CNN optimization is determined according to each of its baseline architecture.

train set	Fast R-CNN	ME R-CNN	Faster R-CNN				ME R-CNN			
		2a/2b/2c	1	2	3	4	1	2a/2b/2c	3	4a/4b
07	40k/30k	40k/30k	80k/60k	40k/30k	80k/60k	40k/30k	80k/60k	40k/30k	80k/60k	40k/30k
12	40k/30k	40k/30k	80k/60k	40k/30k	80k/60k	40k/30k	80k/60k	40k/30k	80k/60k	40k/30k
07+12	100k/75k	100k/75k	200k/150k	100k/75k	200k/150k	100k/75k	200k/150k	100k/75k	200k/150k	100k/75k
07++12	120k/90k	120k/90k	240k/180k	120k/90k	240k/180k	120k/90k	240k/180k	120k/90k	240k/180k	120k/90k
coco train	·	·	320k/240k	320k/240k	320k/240k	320k/240k	320k/240k	160k/120k	320k/240k	160k/120k
coco trainval	·	·	320k/240k	320k/240k	320k/240k	320k/240k	320k/240k	160k/120k	320k/240k	160k/120k

TABLE I: Minibatch iterations and step sizes of ME R-CNNs and their baseline architectures for different trainsets and training steps. Training set key: **07**: VOC07 trainval, **12**: VOC12 trainval, **07+12**: union of VOC07 trainval and VOC12 trainval, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval, **coco train**: MS COCO train, **coco trainval**: MS COCO train and val.



(a) R-CNN + ResNet-101



(b) ME R-CNN + ResNet-101

Fig. 6: ME R-CNN adopting ResNet-101 architecture. Per-RoI network of ResNet-101 consists of three residual modules (Res5a, Res5b, and Res5c). One can observe that the position of Res5a (red) with respect to the RoI pooling has changed.

For all evaluations, we use single-scale training/testing as in [5], by setting the shorter side of the images to be 600 pixels. We have carried out all the experiments on Caffe framework [57] with a Titan XP GPU.

Adapting Multi-Expert into ResNet-101. To use ResNet-101 as the backbone network for object detection, He et al. [29] exploited the last 10 convolutional layers of ResNet-101 to function as the per-RoI network as depicted in Figure 6(a). Denote three residual modules consisting of the last 10 convolutional layers as Res5a, Res5b, and Res5c, respectively. In our case, as shown in Figure 6(b), the last 6 convolutional layers (Res5b and Res5c) are used as the per-RoI multi-expert network and the first 4 convolutional layers (Res5a) are appended at the end of the per-image convolutional network due to the GPU memory limitation.

In ResNet-101, RoI pooling layer takes a per-image convolutional feature map and outputs 14×14 per-RoI feature maps. Per-RoI feature map resolution is halved to 7×7 during processing in Res5a. For our architecture, in order to maintain the resolution of the per-image feature map (i.e., an input of RoI pooling layer as well as the output of the Res5a), we do not use a resizing function of Res5a. Instead, we modify the output dimension of the RoI pooling layer as 7×7 in order to preserve the following layers (Res5b and Res5c) which are inherited from ResNet-101.

We also reduce the batch size from 128 to 64 for training while taking more training iterations. This change is also considered in order to cope with the GPU memory limitation.

We have made above modifications in order to adopt our ME component into an architecture which consists of Faster R-CNN and ResNet-101. Note that Faster R-CNN + ResNet-101 combination is widely used as the backbone architecture in state-of-the-art object detection approaches. (See Table XII.)

B. VOC 2007 and 2012 Results

PASCAL VOC datasets contain 20 object categories. VOC 07 dataset consists of 5k images in trainval set and 5k images in test set. VOC12 dataset has 10k images in trainval set and 10k images in test set. We use the standard metric for evaluating the detection accuracy for PASCAL VOC which is by taking a mean of average precision (mAP) over all object categories.

Table II shows that, on VOC07, ME R-CNN provides improved detection accuracy in mAP than Fast/Faster RCNN when using VOC07 trainval set for training (69.0% vs. 66.9% and 70.6% vs. 69.9%, respectively). When using **07+12**, ME R-CNN outperforms both Fast R-CNN and Faster R-CNN by 2.2% and 2.6%, respectively (72.2% vs. 70.0% and 75.8% vs. 73.2%). VOC12 results are shown in Table IV where we observe consistent performance boost for ME R-CNN. In both cases of VOC12 trainval and 07++12, ME R-CNN outperforms both Fast/Faster R-CNN by at least 2.1% mAP (2.9% at most).

In table III and V, ME R-CNN shows a consistent performance boost when compared with ResNet-101 with Faster R-CNN on both VOC07 (78.7% vs. 76.4%) and VOC12 (76.1% vs. 73.8%). For this result, ME R-CNN was built on top of the ResNet-101 with Faster R-CNN architecture for fair comparison, also showing that the proposed architecture can effectively be combined with various types of object detection CNNs.

C. MS COCO Results

We evaluate ME R-CNN on MS COCO dataset and show the results in Table VI. The MS COCO dataset contains

method	trainset	mAP (%)	category																			
			aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
Fast R-CNN [5]	**07**	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
ME R-CNN	**07**	**69.0**	70.6	78.9	68.2	55.8	44.3	80.9	78.2	84.6	44.4	76.5	70.4	80.6	81.5	76.6	70.8	35.1	66.4	69.9	76.8	69.5
Faster R-CNN [6]	**07**	69.9	70.0	**80.6**	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
ME R-CNN	**07**	**70.6**	69.9	79.5	68.2	58.5	52.5	77.1	80.1	84.4	52.1	78.6	67.3	81.0	83.7	74.6	77.0	38.2	71.0	66.0	75.2	**74.7**
Fast R-CNN [5]	**07+12**	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
ME R-CNN	**07+12**	**72.2**	**78.1**	78.9	69.4	61.3	44.5	84.8	81.7	87.6	50.7	80.1	**70.6**	85.8	**84.8**	**78.7**	72.3	35.0	71.9	**75.3**	79.9	72.7
Faster R-CNN [6]	**07+12**	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
ME R-CNN	**07+12**	**75.8**	77.2	79.7	**76.3**	**67.0**	**60.3**	**86.0**	**87.1**	**88.6**	**58.3**	**83.8**	70.3	**86.4**	84.7	78.4	**78.4**	**45.1**	**76.0**	73.8	**83.6**	74.6

TABLE II: VOC 2007 detection accuracy. All methods use VGG16.

method	trainset	mAP (%)	category																			
			aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
Faster R-CNN [6]	**07+12**	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	89.8	56.7	87.8	69.4	88.3	88.9	80.9	78.4	41.7	78.6	79.8	**85.3**	72.0
ME R-CNN	**07+12**	**78.7**	**81.2**	**81.9**	**78.0**	**71.8**	**65.0**	**86.0**	**87.5**	**91.3**	**61.0**	**89.2**	**69.9**	**88.4**	**90.1**	**83.9**	**81.4**	**45.2**	**81.0**	**81.7**	**85.3**	**73.9**

TABLE III: VOC 2007 detection accuracy. All methods use ResNet-101.

method	trainset	mAP (%)	category																			
			aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
Fast R-CNN [5]	**12**	65.7	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7
ME R-CNN⋆	**12**	**67.8**	82.6	76.4	69.9	50.3	41.8	75.5	71.1	87.0	42.0	74.3	56.0	86.3	81.5	78.9	72.4	34.1	68.5	62.6	79.6	64.7
Faster R-CNN [6]	**12**	67.0	82.3	76.4	71.0	48.4	45.2	72.1	72.3	87.3	42.2	73.7	50.0	86.8	78.7	78.4	77.4	34.5	70.1	57.1	77.1	58.9
ME R-CNN†	**12**	**69.2**	81.2	75.7	71.2	51.1	47.8	73.3	74.6	88.1	46.9	76.4	52.9	87.1	81.7	81.4	78.8	38.4	72.9	60.0	78.4	66.9
Fast R-CNN [5]	**07++12**	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	**89.3**	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
ME R-CNN‡	**07++12**	**70.7**	84.0	79.8	72.4	54.9	43.3	78.4	74.7	**89.3**	46.6	76.1	**60.6**	**87.8**	**83.6**	82.1	74.8	39.4	70.6	**65.7**	**82.5**	67.9
Faster R-CNN [6]	**07++12**	70.4	84.9	79.8	**74.3**	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
ME R-CNN§	**07++12**	**73.3**	**85.4**	**80.7**	74.0	**58.3**	**55.0**	**79.7**	**78.5**	88.6	**52.9**	**78.2**	57.8	87.7	83.3	**83.7**	**81.9**	**50.6**	**74.8**	62.4	81.8	**69.8**

⋆ http://host.robots.ox.ac.uk:8080/anonymous/69D0YS.html † http://host.robots.ox.ac.uk:8080/anonymous/O3RFBG.html

‡ http://host.robots.ox.ac.uk:8080/anonymous/PLPKPU.html § http://host.robots.ox.ac.uk:8080/anonymous/YTVCEH.html

TABLE IV: VOC 2012 detection accuracy. All methods use VGG16.

method	trainset	mAP (%)	category																			
			aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
Faster R-CNN [6]	**07++12**	73.8	86.5	81.6	**77.2**	58.0	51.0	78.6	76.6	**93.2**	48.6	**80.4**	59.0	**92.1**	**85.3**	84.8	80.7	48.1	77.3	66.5	**84.7**	65.6
ME R-CNN⋆	**07++12**	**76.1**	**87.1**	**82.7**	76.3	**62.5**	**62.6**	**81.7**	**80.8**	90.6	**54.8**	79.1	**63.1**	89.6	84.4	**85.4**	**84.1**	**55.0**	**77.9**	**67.1**	84.3	**71.9**

⋆ http://host.robots.ox.ac.uk:8080/anonymous/M9ZUJK.html

TABLE V: VOC 2012 detection accuracy. All methods use ResNet-101.

80k, 40k and 20k samples in train, val, and test-dev sets, respectively. We compare ME R-CNN with the Faster R-CNN on two different evaluation settings: i) training the network using train set and testing on val set, and ii) training the network on train and val set and testing on test-dev set. We used two different standard metrics, which are mAP@.5 (PASCAL VOC metric) and mAP@[.5,.95] (MS COCO metric). The MS COCO metric (mAP@[.5,.95]) indicates the mAPs averaged for IOU$\in$[0.5:0.05:0.95]. Regardless of the which metric we choose to use, the ME R-CNN achieved consistent performance gain over the Faster R-CNN on both evaluation settings. When trained on train set, ME R-CNN outperforms the Faster R-CNN by 2.1 for mAP@.5 and 0.6 for mAP@[.5,.95]. When using train+val set for training, gains achieved by ME R-CNN over Faster R-CNN are 2.6 and 4.0 for the two metrics, respectively.

D. Ablation Experiments

The ablation experiments are conducted on PASCAL VOC07 [55]. In the ablation experiments, ME R-CNN model with Faster R-CNN and VGG16 is used. The model was trained on VOC07 trainval set and tested on VOC07 test set.

EAN vs. Hard-coded Assignment. We have verified the significance of employing the 'learnable' EAN. Table VII shows the detection accuracy of the ME R-CNN with and without EAN. As the architecture still requires an expert assigner to direct the RoIs to the experts even when EAN

method	trainset	testset	mAP@.5	Gain	mAP@[.5,.95]	Gain
Faster R-CNN [6]	**train**	**val**	41.5	·	21.2	·
ME R-CNN	**train**	**val**	**43.6**	+2.1	**21.8**	+0.6
Faster R-CNN [6]	**trainval**	**test-dev**	42.7	·	21.9	·
ME R-CNN	**trainval**	**test-dev**	**45.3**	+2.6	**25.9**	+4.0

TABLE VI: MS COCO detection accuracy. All methods use VGG16. Set key: **train**: MS COCO train set, **val**: MS COCO val set, **trainval**: MS COCO train and val sets, **test-dev**: MS COCO test-dev set.

Assigner	EAN	Hard-coded assignment
mAP (%)	**70.6**	69.8

TABLE VII: Effectiveness of learning RoI-expert relationship. RoI-expert relationship is "learned" in EAN whereas RoIs are designated to the experts based on a pre-defined criteria in the hard-coded assignment.

is not present, we have used a 'hard-coded assignment' to take its place instead. For the hard-coded assignment case, the RoIs are assigned to the experts based on their aspect ratios which represent one of the three shape categories: horizontally elongated (**H**), square-like (**S**), and vertically elongated (**V**). For training this model, the modified version of the 4-step alternating algorithm is used by updating the ME weights guided by the hard-coded assignments in step 2 and step 4. ME R-CNN with EAN module outperforms the hard-coded assignment case by 0.8 mAP which shows the effectiveness of having a learnable expert assigner in the multi-expert architecture.

Functionality of Experts. Table VIII shows how the functionality of each expert has changed after the overall training has been carried out. As mentioned in 'Initialize ME & Conv-L' in Subsection IV-B, ME weights are initialized by directing the RoIs to the experts by their aspect ratio-based shape categories (**H**, **S**, or **V**). The table shows that the newly trained experts, with the help of 'learnable' EAN, is less biased towards the aspect ratio of the RoIs.

The detection performance of each expert on the whole dataset (VOC07 test set) only reaches to approximately 66% mAP which is lower than the performance of the Faster R-CNN. However, one can notice that when these multiple experts are exploited together, the performance can be boosted up to 70.6% (0.7% better than Faster R-CNN).

We can also observe that the functionality change in the ME, made possible by the co-training of ME and EAN, eventually provided extra room for noticeable performance increase over the hard-coded assignment case (Table VII). Figure 7 depicts example object detection results acquired by different experts in our final version of ME R-CNN.

Co-training of ME, EAN, & Conv-L. We have conducted an experiment to validate the effectiveness of co-training (simultaneous training) the three major components (ME, EAN, and Conv-L) of ME R-CNN. These three components affect each other which makes the training challenging as mentioned in IV-B. We compare our training strategy tailored for ME R-CNN (Algorithm 1) with the two baselines which do not perform any co-training for the three components. The first baseline (denoted as 'Remove 2c' in Table IX) is implemented by simply removing step 2c from Algorithm 1. In this scenario, ME and EAN are trained twice during the entire training process. The second baseline (denoted as 'Replace 2c') is implemented by replacing step 2c by alternating update of ME and EAN. ME and EAN are trained three times for this baseline. For both of the baselines, the last step which Conv-L is being optimized is step 2a in Algorithm 1. This is because, without using the strategy of co-training of the three components, Conv-L needs to be fixed. As can be seen in Table IX, having the three components trained simultaneously outperforms the baseline cases.

EAN Labels Presetting. To understand how EAN label presetting affects object detection accuracy, we compared three RoI-expert assignment presetting cases based on various RoI characteristics: aspect ratio, RoI size, and object category. In each case, three experts were employed to make a fair comparison.

When size is used as the presetting criteria, we manually defined three different categories: *small*, *medium*, and *large*. All the RoIs bigger than 110^2 are assigned as *large*, and the other RoIs are assigned as *small*. All RoIs that fall between 55^2 and 205^2 are assigned as *medium*. Similar to aspect-ratio-based presetting criteria, one RoI can be assigned to either one or two categories.

When the presetting criteria is defined based on object category, the RoIs are grouped to be semantically similar as follows:

- *Vehicl*e: Aeroplane, Bicycle, Boat, Bus, Car, Motorbike, Train
- *Animal*: Bird, Cat, Cow, Dog, Horse, Person, Sheep
- *Other*: Bottle, Chair, Diningtable, Pottedplant, Sofa, TV-monitor

RoIs having less than 0.1 IOU overlap with any object bounding box are not used in EAN training.

Table X compares the object detection accuracy of the three presetting strategies. Presetting based on RoI's aspect ratio shows the highest accuracy, but the differences compared to the other two are marginal ($\sim$1%). We observe that the accuracy is not highly dependent upon the EAN label presetting strategy.

Comparison with CNNs Adopting Mixture-of-Expert in a Conventional Way. We compare ME R-CNN with CNNs that adopt the mixture-of-expert in a conventional manner. This comparison is made to verify the effectiveness of the proposed approach which assigns each RoI into its most appropriate

Expert #	Detection Bounding Boxes			mAP (%)
	$w > \sqrt{2}h$	$\sqrt{2}h \geq w \geq \frac{1}{\sqrt{2}}h$	$w < \frac{1}{\sqrt{2}}h$	
1	46.8% (83.8%)	24.4% (16.2%)	28.8% (0.0%)	66.1
2	33.4% (24.7%)	44.2% (50.2%)	22.4% (25.1%)	66.7
3	7.5% (0.0%)	35.6% (17.7%)	56.9% (82.3%)	66.4

TABLE VIII: The change in the functionality of the experts with EAN. Table shows the RoI-expert distribution after the overall training along with the final detection accuracy (mAP) for each expert. Note that the final distributions have drastically changed from the aspect ratio-based initialization (in parentheses). w and h indicate the width and the height of RoI, respectively.

Optimization Strategy	Co-training?	mAP (%)
Remove 2c	No	68.8
Replace 2c	No	69.3
Algorithm 1	Yes	**70.6**

TABLE IX: Effectiveness of co-training of EAN, ME, & conv-L. '2c' refers to Step 2c in Algorithm 1. This is the only step where co-training of the three components takes place.

Presetting	Aspect Ratio	RoI Size	Object Category
mAP (%)	**70.6**	69.6	69.9

TABLE X: Comparison of three EAN label presetting.

Conventional		mAP (%)	Test time (sec)
ImageNet Pretrain	Gating Net		
		68.7	0.099
	√	68.9	0.106
√		69.6	0.099
√	√	69.7	0.106
ME R-CNN		**70.6**	**0.075**

TABLE XI: Performance comparison of various models adopting mixture-of-expert in a conventional way with respect to object detection accuracy (mAP) and test time. We compared four conventional approaches that differ depending on the use of finetuning from ImageNet pretrained network and the use of the gating network.

expert instead of utilizing all the experts at once. Figure 2 shows the architectural differences between our approach and the conventional approaches when adopting the mixture-of-expert concept into a CNN architecture.

As a representative conventional architecture, we implemented R-CNN with three experts and a gating network tied in a conventional manner as shown in Figure 8. For the gating network, we use the same architecture (single convolutional layer, single pooling layer, and single fully-connected layer) as the EAN in ME R-CNN so that this conventional model has the same number of weights as ME R-CNN. This network is also optimized using 4-step alternating algorithm in exactly the same order. In the second step, each expert is optimized by minimizing its own softmax loss for classification and L1smooth loss for bounding box regression, i.e., six losses are imposed. Accordingly, we prepare three different training batches so that each expert is trained with a different batch. Meanwhile, in the fourth step only one set of losses (classification and bounding box regression) is used to optimize all three experts. The gating network was trained only in the fourth step.

A common goal for having multiple experts is to be able to equip each expert module with its own unique expertise and thus leverage such capability for overall performance increase. However, finetuning from the same ImageNet-pretrained network (i.e., same initialization) is likely to lead all experts to have similar expertise which conflicts with the original goal. Therefore, we have trained this conventional model in two different ways, with and without using the pretraining network. To observe the effectiveness of the gating network, we compared Figure 8 with the model without the gating network. In Table XI, four combinations of the conventional mixture-of-expert approaches are compared with ME R-CNN with respect to mAP and test time. We can observe that the object detection accuracy underperforms when the finetuning strategy was omitted, which shows that our original anticipation to differentiate the experts' expertise was not achieved. It is also shown that the gating network improves performance marginally. Above all, ME R-CNN provides better mAP and speed than any of the conventional mixture-of-expert approaches.

E. Timing

To analyze the computational overhead of exploiting "multiple experts", we compare the train/test time of ME R-CNN with the Faster R-CNN. This analysis was conducted using NVidia Titan XP GPU.

Inference. While Faster R-CNN takes 0.07 sec/image, ME R-CNN takes 0.075 sec/image. Using multiple experts brings almost no overhead because the number of RoIs do not change compared to the Faster R-CNN case, and only one of the experts is being activated for each RoI.

Training. Training ME R-CNN (16.9 hrs) requires almost twice as much time when compared to the case of Faster R-CNN (8.15 hrs). Several recently introduced measures such as multi-GPU parallel computing or enlarging mini-batch size [58], [59] can be taken into consideration to reduce the overall training time.

VI. Future Works

In this paper, we have focused on showing that ME R-CNN architecture can boost the object detection performance when integrated with the baseline R-CNNs. Fast R-CNN and Faster R-CNN, which are two of the most widely used object detection networks, were selected to demonstrate the effectiveness.

Fig. 7: Example object detection results using ME R-CNN. The detection results from the experts 1, 2, and 3 are depicted in red, blue, and green bounding boxes, respectively.

In the future, we will focus on producing the state-of-the art performance in benchmark datasets (PASCAL VOC and MS COCO) by incorporating additional processings (refered to as 'adding bells and whistles' in [15]) such as multi-scale training/testing (MS) [5], [14], online hard example mining (OHEM) [15], iterative bounding box regression [60], global context (CXT) [29], [60], [61], ensemble of classifiers (ENS) [29], [62], integrating with image classification output [63], and feature pyramid network (FPN) [64]. We also plan to incorporate multi-expert into other CNN-based detection architecture such as SSDs [65], [66], YOLOs [67]–[69], R-FCN [8], and RetinaNet [70], which has recently been introduced.

State-of-the-art Object Detection Methods. On the leaderboard of the MS COCO object detection competition [78], the

Rank	Method	Backbone		Adding Bells and Whistles		
		Img. Classif.	Obj. Det.	Plug-In	Boosting Strategy	More Info.
1	MegDet [71]	ResNeXt-152 [28]	Faster	FPN, GCN [72], RoI Align [10]	**Large Batch**, OHEM, CXT, MS, ENS	Segmentation [73]
2	PANet [74]	ResNet-101	Faster	**Adaptive FeatPool**, FPN, RoI Align [10]	MS, ENS	Pixel Label [10]
3	MSRA	Xception [75]	Faster	**Novel RoI Pool [76]**, FPN, SoftNMS [77]	ENS	·
4	Mask R-CNN [10]	ResNeXt-152 [28]	Faster	**RoI Align**, RPN	MS, ENS	Pixel Label [10], ImageNet-5K
⋮	⋮	⋮	⋮	⋮	⋮	⋮

TABLE XII: Leaderboard of MS COCO object detection competition (As of June 14, 2019). The major contribution of each method is shown in bold. For the image classification backbone, ResNeXt and Xception were devised based on the ResNet architecture. All the acronyms in the table have previously been defined.

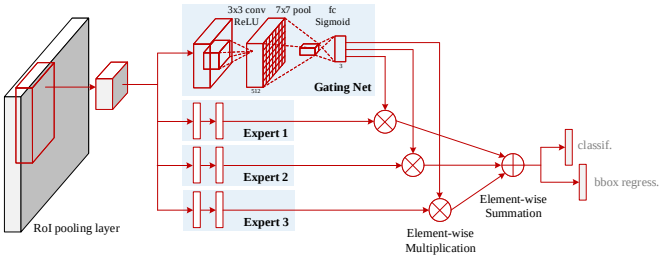


Fig. 8: R-CNN architecture adopting mixture-of-expert in a conventional manner. To provide fair grounds for comparison between this model and ME R-CNN with respect to the network size, the number of experts has been set to three, and the gating network is implemented to match the EAN architecture.

top ranked methods provide much higher object detection accuracy compared to the accuracy achieved with our approach.[1] The top four methods [10], [71], [74], [76] achieve the state-of-the-art accuracy by adding multiple plug-in modules, using performance boosting strategies, and/or providing more information to the backbone combining Faster R-CNN with one of the ResNet variations (ResNet, ResNeXt or Xception). Table XII shows the bells and whistles used to achieve the state-of-the-art accuracy of each method. Following this trend, we also experimented by adding our ME module to the backbone combining Faster R-CNN and ResNet.

VII. Conclusion

We introduced ME R-CNN which uses multiple experts (ME) in place of a conventional single classifier incorporated in CNN-based object detection architecture. Having ME is found to be advantageous as each expert is learned to specialize in a certain type of RoIs, considering the fact that RoIs are manifested in various appearance caused by different shapes, poses, and viewing angles. To optimize the ME usage, we have introduced expert assignment network (EAN) which automatically learns the RoI-expert relationship. We have introduced a practical training strategy to better handle the challenging task of optimizing the complex architecture which contains ME, EAN, and a shared convolutional network. With benefits of the novel components, ME R-CNN proves its effectiveness in consistently enhancing the detection accuracy in PASCAL VOC 07, 12, and MS COCO datasets over the baseline methods.

[1]Details on the architectures used for the top ranked methods in the PASCAL VOC Competition are unavailable.

References

[1] F. Khan, R. Anwer, J. van de Weijer, A. Bagdanov, M. Vanrell, and A. Lopez, "Color attributes for object detection," in *CVPR*, 2012.
[2] T. Malisiewicz, A. Gupta, and A. Efros, "Ensemble of exemplar-SVMs for object detection and beyond," in *ICCV*, 2011.
[3] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 9, pp. 1627–1645, 2010.
[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.
[5] R. Girshick, "Fast R-CNN," in *ICCV*, 2015.
[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.
[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 1, pp. 142–158, 2016.
[8] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *NIPS*, 2016.
[9] 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*, no. 6, pp. 1137–1149, 2017.
[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017.
[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
[12] Y. Sun, Q. Zhu, and Z. Chen, "An iterative initial-points refinement algorithm for categorical data clustering," *Pattern Recognition Letters*, pp. 875–884, 2002.
[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.
[14] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *ECCV*, 2014.
[15] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *CVPR*, 2016.
[16] X. Wang, A. Shrivastava, and A. Gupta, "A-Fast-RCNN: Hard positive generation via adversary for object detection," in *CVPR*, 2017.
[17] K. Yi, K. Yun, S. Kim, H. Chang, and J. Choi, "Detection of moving objects with non-stationary cameras in 5.8ms: Bringing motion detection to your mobile device," in *CVPR Workshop*, 2013.
[18] E. Bernstein and Y. Amit, "Part-based statistical models for object classification and detection," in *CVPR*, 2015.
[19] H. Schneiderman and T. Kanade, "A statistical method for 3D object detection applied to faces and cars," in *CVPR*, 2000.
[20] A. Verma, R. Hebbalaguppe, L. Vig, S. Kumar, and E. Hassan, "Pedestrian detection via mixture of CNN experts and thresholded aggregated channel features," in *ICCVW*, 2015.
[21] R. Rasti, M. Teshnehlab, and S. LamPhung, "Breast cancer diagnosis in DCE-MRI using mixture ensemble of convolutional neural networks," *Pattern Recognition*, pp. 381–390, December 2017.
[22] S. Kumagai, K. Hotta, and T. Kurita, "Mixture of counting CNNs: Adaptive integration of CNNs specialized to specific appearance for crowd counting," in *arXiv:1703.09393*, 2017.
[23] S. Gross, M. Ranzato, and A. Szlam, "Hard mixtures of experts for large scale weakly supervised vision," in *CVPR*, 2017.
[24] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *CVPR*, 2017.

[25] Á. García-Martín, J. C. SanMiguel, and J. M. Martínez, "Coarse-to-fine adaptive people detection for video sequences by maximizing mutual information," *Sensors*, pp. 1–22, December 2018.

[26] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS*, 2012.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[28] S. Xie, R. Girshick, P. Dollàr, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *CVPR*, 2017.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[30] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *CVPR*, 2016.

[31] S. Eum*, H. Lee*, H. Kwon, and D. Doermann, "IOD-CNN: Integrating object detection networks for event recognition," in *ICIP*, 2017, (* indicates equal contribution.).

[32] I. Kokkinos, "UberNet : Training a 'universal' convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *CVPR*, 2017.

[33] H. Lee*, S. Eum*, J. Levis*, H. Kwon, J. Michaelis, and M. Kolodny, "Exploitation of semantic keywords for malicious event classification," in *ICASSP*, 2018, (* indicates equal contribution.).

[34] G. Gkioxari, R. Girshick, P. Dollár, and K. He, "Detecting and recognizing human-object interactions," in *CVPR*, 2018.

[35] H. Lee, S. Eum, and H. Kwon, "Cross-domain CNN for hyperspectral image classification," in *IGARSS*, 2018.

[36] H. Lee, S. Eum, and H. Kwon, "DOD-CNN: Doubly-injecting object information for event recognition," in *ICASSP*, 2019.

[37] H. Lee, S. Eum, and H. Kwon, "S-DOD-CNN: Doubly-injecting spatially-preserved object information for event recognition," in *arXiv:1902.04051*, 2019.

[38] H. Lee, S. Eum, and H. Kwon, "Is pretraining necessary for hyperspectral image classification?" in *IGARSS*, 2019.

[39] Y.-X. Wang, D. Ramanan, and M. Hebert, "Growing a brain: Fine-tuning by increasing model capacity," in *CVPR*, 2017.

[40] H. Ralambondrainy, "A conceptual version of the K-means algorithm," *Pattern Recognition Letters*, pp. 1147–1157, 1995.

[41] A. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, pp. 651–666, 2010.

[42] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 8, pp. 790–799, 1995.

[43] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5, pp. 603–619, 2002.

[44] L. Wang, D. Tang, Y. Guo, and M. Do, "Common visual pattern discovery via nonlinear mean shift clustering," *IEEE Transactions on Image Processing*, no. 12, pp. 5442–5454, 2015.

[45] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, 1996.

[46] J. Hou, H. Gao, and X. Li, "Dsets-DBSCAN: A parameter-free clustering algorithm," *IEEE Transactions on Image Processing*, no. 7, pp. 3182–3193, 2016.

[47] J. Shen, X. Hao, Z. Liang, Y. Liu, W. Wang, and L. Shao, "Real-time superpixel segmentation by DBSCAN clustering algorithm," *IEEE Transactions on Image Processing*, no. 12, pp. 5933–5942, 2016.

[48] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, no. 1, pp. 1–38, 1977.

[49] S. Sanjay-Gopal and T. Hebert, "Bayesian pixel classification using spatially variant finite mixtures and the generalized EM algorithm," *IEEE Transactions on Image Processing*, no. 7, pp. 1014–1028, 1998.

[50] H. Hong and D. Schonfeld, "Attraction-repulsion expectation-maximization algorithm for image reconstruction and sensor field estimation," *IEEE Transactions on Image Processing*, no. 9, pp. 2004–2011, 2009.

[51] P. Fränti, O. Virmajoki, and V. Hautamäki, "Fast agglomerative clustering using a k-nearest neighbor graph," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 11, pp. 1875–1881, 2006.

[52] K. Han, S. Kim, and S. Narayanan, "Strategies to improve the robustness of agglomerative hierarchical clustering under data source variation for speaker diarization," *IEEE Transactions on Audio, Speech, and Language Processing*, no. 8, pp. 1590–1601, 2008.

[53] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[54] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[55] M. Everingham, L. Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes (VOC) challenge," *International Journal on Computer Vision*, no. 2, pp. 303–338, 2015.

[56] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, 2014.

[57] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACMMM*, 2014.

[58] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *CVPR*, 2015.

[59] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: Training ImageNet in 1 hour," in *arXiv preprint arXiv:1706.02677*, 2017.

[60] S. Gidaris and N. Komodakis, "Object detection via a multi-region & semantic segmentation-aware cnn model," in *ICCV*, 2015.

[61] J. Li, Y. Wei, X. Liang, J. Dong, T. Xu, J. Feng, and S. Yan, "Attentive contexts for object detection," *IEEE Transactions on Multimedia*, no. 5, pp. 944–954, 2017.

[62] H. Lee, H. Kwon, R. Robinson, W. Nothwang, and A. Marathe, "Dynamic belief fusion for object detection," in *WACV*, 2016.

[63] Y. Cao*, H. Lee*, and H. Kwon, "Enhanced object detection via fusion with prior beliefs from image classification," in *ICIP*, 2017, (* indicates equal contribution.).

[64] T.-Y. Lin, P. Dollàr, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[65] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *ECCV*, 2016.

[66] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD: Deconvolutional single shot detector," in *arXiv:1701.06659v1*, 2017.

[67] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[68] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *CVPR*, 2017.

[69] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv:1804.02767v1*, 2018.

[70] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollàr, "Focal loss for dense object detection," in *ICCV*, 2017.

[71] C. Peng*, T. Xiao*, Z. Li*, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "MegDet: A large mini-batch object detector," in *CVPR*, 2018, (* indicates equal contribution.).

[72] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters – improve semantic segmentation by global convolutional network," in *CVPR*, 2017.

[73] J. Mao*, T. Xiao*, Y. Jiang, and Z. Cao, "What can help pedestrian detection?" in *CVPR*, 2017, (* indicates equal contribution.).

[74] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *CVPR*, 2018.

[75] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017.

[76] J. Dai*, H. Qi*, Y. Xiong*, Y. Li*, G. Zhang*, H. Hu, and Y. Wei, "Deformable convolutional networks," in *CVPR*, 2017, (* indicates equal contribution.).

[77] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS – improving object detection with one line of code," in *ICCV*, 2017.

[78] http://cocodataset.org/#detection-leaderboard.