Mixture-Model-based Bounding Box Density Estimation for Object Detection

Jaeyoung Yoo Geonseok Seo Nojun Kwak
Seoul National University
{yoojy31|geonseoks|nojunk}@snu.ac.kr

Abstract

In this paper, we propose a new object detection model, Mixture-Model-based Object Detector (MMOD), that performs multi-object detection using a mixture model. Unlike previous studies, we use density estimation to deal with the multi-object detection task. MMOD captures the conditional distribution of bounding boxes for a given input image using a mixture model consisting of Gaussian and categorical distributions. For this purpose, we propose a method to extract object bounding boxes from a trained mixture model. In doing so, we also propose a new network structure and objective function for the MMOD. Our proposed method is not trained by assigning a ground truth bounding box to a specific location on the network's output. Instead, the mixture components are automatically learned to represent the distribution of the bounding box through density estimation. Therefore, MMOD does not require a large number of anchors and does not incur the positive-negative imbalance problem. This not only benefits the detection performance but also enhances the inference speed without requiring additional processing. We applied MMOD to Pascal VOC and MS COCO datasets, and outperform the detection performance with inference speed of other state-of-the-art fast object detection methods. (38.7 AP with 39ms per image on MS COCO without bells and whistles.) Code will be available.

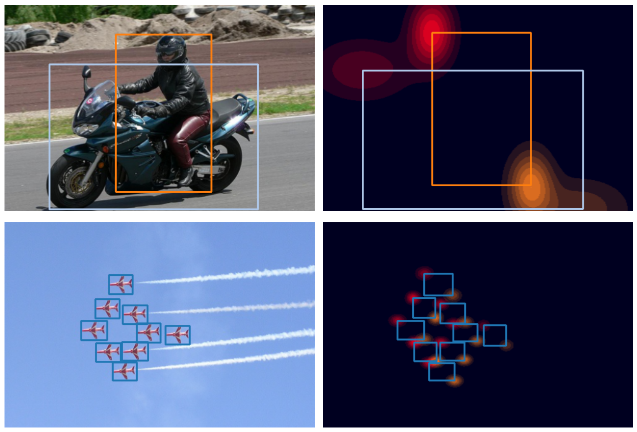

Figure 1. Visualization of the likelihood of the estimated mixture model by our method. Left images are input images and ground truth bounding boxes. In right images, red and orange contours show the likelihood of left-top and right-bottom corners of bounding boxes, respectively.

1. Introduction

Multi-object detection is the task of finding multiple objects through bounding boxes with class information. Since the breakthrough of the deep neural networks (DNN), multi-object detection has been extensively developed in terms of computational efficiency and performance and is now at a level that can be used in real life and industry.

There are several approaches for multi-object detection using DNN. Most methods perform regression of the bounding box coordinates and estimate the class probability by assigning a ground truth bounding box to specific anchors which are used as references of output bounding boxes. These methods, based on ground truth assignment, have become the mainstream of multi-object detection [41, 7, 29, 17, 26, 27]. However, there are some problems in these methods. A ground truth bounding box must be assigned to the anchors at the specific location in the network's output, depending on its location, scale, and aspect ratio. Therefore, a large number of anchors having various scales and aspect ratios are needed to cover the four-dimensional (xy position, height, width, or top-left, bottom-right corners) space in which a box can exist on an image. Besides, the scale and aspect ratio of the anchor affect detection performance. Thus, an appropriate design of the anchor is required, which may be determined heuristically or require a separate process [40, 39]. Also, a large number of anchors can cause the so-called positive-negative imbalance problem, where negative background samples outnumber the positive ones, making training difficult.

Recently, keypoint-based object detection methods have been proposed inspired by bottom-up pose estimation. Instead of learning the bounding box coordinates directly, these methods learn the heat-maps for the points constituting bounding boxes and obtain the resultant bounding boxes from them. The keypoint-based methods solve many of the drawbacks of the ground-truth-assignment-based methods. Since the keypoint-based methods do not rely on anchors, the design of the network is simpler and the number of hyper-parameters is reduced. Currently, keypoint-based detectors show state-of-the-art detection performances. However, these methods have other drawbacks. It is necessary to extract the bounding box coordinates from the estimated heat-maps since they learn the heat-maps rather than directly learn the coordinates of the bounding boxes. Moreover, keypoint-based methods use hourglass-like networks [33] that require a relatively large amount of computation. When using a different structure such as FPN, which requires relatively a small amount of computation, the detection performance becomes much lower than that by an hourglass-like structure [23]. This is a disadvantage that prohibits keypoint-based detection methods from being practical for real applications.

In this paper, we approach the multi-object detection task through density estimation. We propose a Mixture-Model-based Object Detector (MMOD) that captures the distribution of bounding boxes for an input image using a mixture model of components consisting of Gaussian and categorical distributions. For each component of the mixture model, the Gaussian represents the distribution of the bounding box coordinates, and the categorical distribution represents the class probability of that box. Also, we proposed the process to sample the Region of Interests (RoIs) from the estimated mixture model to learn the class probability considering the background. In the training phase, the network is trained to maximize the log-likelihood of the mixture model for the ground truth bounding boxes and sampled RoIs.

Through density estimation using a mixture model, our MMOD has the following advantage. First, unlike the ground-truth-assignment-based methods, the mixture components learn the location and class of objects through the density estimation of bounding boxes without groud truth assignment. Second, since the RoIs are sampled from the estimated mixture model that captures the ground truth bounding boxes, our MMOD is free from the positive-negative imbalance problem. Third, unlike the keypoint-based methods, our method does not need the extra process of extracting bounding boxes from the heat-map. It is also more friendly to the feature-pyramid-style networks that are computationally more efficient. Finally, our proposed method achieves the state-of-the-art detection performance among the object detection methods with similar speed.

2. Related Works

Ground-truth-assignment-based methods: Many kinds of research have been conducted on object detection using DNN [15, 18, 14]. Earlier studies such as Faster R-CNN [41] and SSD [29] attempted to represent the space of bounding boxes as much as possible by using anchors in training. The problem is that the scale and aspect ratio of anchors had a significant impact on detection performance [27, 41]. To resolve it, YOLOv2 [39] and YOLOv3 [40] found the optimal anchor types through k-means clustering. After that, studies not using anchors [43, 22, 52], generating anchor functions [46], and predicting anchor types [45, 49] were conducted. In addition, defining multiple anchors in every possible locations showed the positive-negative imbalance problem. In the early days, OHEM [42] tackled this by constructing a mini-batch with a high-order loss example. Focal Loss [27] tackled the problem by concentrating on the loss of hard examples. Other examples include [25] which suggested the gradient harmonizing mechanism, [34] which presented an effective hard mining method with Intersection over Union balanced sampling, and [5] which used AP-loss by redefining classification as a ranking task.

Keypoint-based methods: Recently, studies on approaching object detection with a keypoint-based method used in pose estimation [44, 38, 4] without using anchors have been conducted. CornerNet [23] used corner pooling to detect corners in the heat-maps and matched them using Associative Embedding [32]. After that, there are studies such as [10, 50, 51] in keypoint-based object detection methods to boost the performance. However, they all show their best performance when using a specific backbone called hourglass network [33] and are relatively slow due to a large amount of computation. There is also a top-down approach [47] that uses deformable convolution [8] to find finer representative points.

Unlike previous methods, we perform multi-object detection by learning the distribution of bounding boxes for an image using a mixture model of Gaussian and categorical distributions. In the proposed method, the heuristic design of anchors and ground truth assignment are not needed. Also, the positive-negative imbalance problem is removed. Finally, our detection performance is superior to any other methods with similar inference speed.

3. Mixture Model for Object Detection

The bounding box b can be represented as a vector consisting of four coordinates b_{ltrb} representing the location (left-top and right-bottom corners) and an one-hot vector b_c representing the corresponding class. It has an uncertainty of its coordinates due to occlusion, inaccurate labeling, and ambiguity of object boundary [20]. Thus, the distribution of b_{ltrb} can be considered as a continuous distribution rather

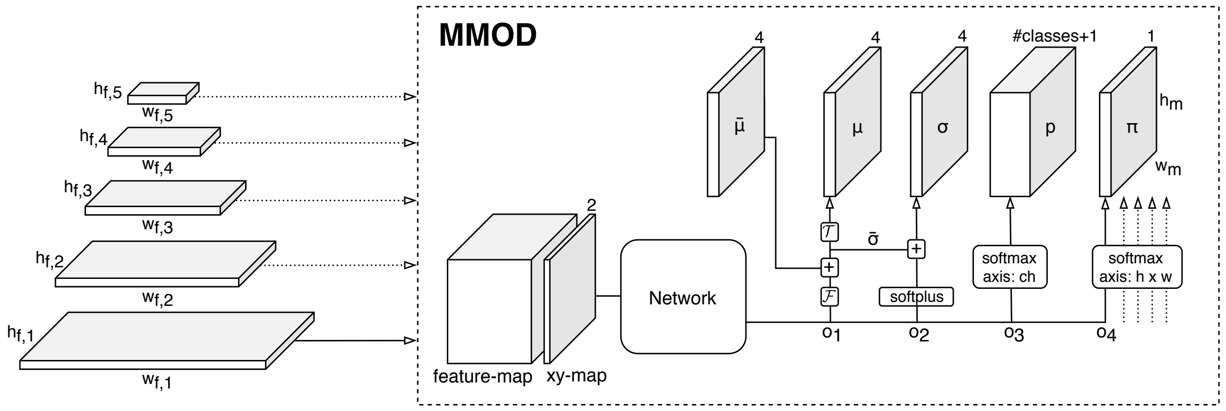

Figure 2. The architecture of MMOD. The parameters of the mixture model (μ, σ, p, and π) are predicted by MMOD. The network produces its intermediate output (o_1 - o_4) from each feature-map of the feature-pyramid (5 levels of feature-maps are used in our implementation). The parameters of the mixture model are obtained from the network's output and the default coordinate ($\bar{\mu}$). The mixed model estimated by MOOD represents the distribution of bounding boxes for an input image.

than a point mass. In the problem of multi-object detection, the conditional distribution of b for an image I may be multi-modal, depending on the number of objects on the input image. Therefore, our object detection network must be able to capture the multi-modal distribution. In this paper, we propose a MMOD that can estimate the multi-modal distribution by extending the mixture density network [1] for object detection. Our proposed network MMOD models the conditional distribution of b for an I using a mixture model whose component consist of Gaussian and categorical distribution. Gaussian and categorical distributions represent the distribution of bounding box coordinates and the distribution for the class probability, respectively. The probability density function of this mixture model is defined by the estimated parameters of the mixture model as follows:

$$p(b|I) = \sum_{k=1}^{K} \pi_k \mathcal{N}(b_{ltrb}; \mu_k, \text{diag}(\sigma_k^2)) \mathcal{P}(b_c; p_k). \qquad (1)$$

Here, $\mathcal{N}$ and $\mathcal{P}$ denote the probability density function of Gaussian distribution and the probability mass function of categorical distribution, respectively. The parameters μ_k, σ_k, and π_k are the mean, standard deviation, and, mixing coefficient of the k-th component among K components. The C-dimensional vector p_k is the probability for C classes. We assume that the covariance matrix of each Gaussian is diagonal to prevent the model from being overly complicated. Each component is a four-dimensional multivariate Gaussian for the coordinates representing the bounding box $b_{ltrb} = \{b_l, b_t, b_r, b_b\}$. Thus, the multivariate Gaussian probability density function of each component of the mixture model can be factorized as follows:

$$\mathcal{N}(b_{ltrb}|I) = \prod_{d \in D} \mathcal{N}(b_d; \mu_{k,d}, \sigma_{k,d}^2), \quad D = \{l, t, r, b\}. \qquad (2)$$

The objective of the MMOD is to accurately estimate the paramters of the mixture model by maximizing the log-likelihood of the ground truth bounding box b, as follows:

$$\theta = \arg\max_{\theta} \mathbb{E}_{b \sim p_{data}(b|I)} \log p(b|I; \theta). \qquad (3)$$

Here, $p_{data}(b|I)$ is the empirical distribution of b for a given I and θ is the parameter vector that includes mixture parameters including the class probability p_k.

4. Mixture-Model-based Object Detector

4.1. Architecture

Figure 2 shows the architecture of MMOD. The network outputs four types of results o_1, o_2, o_3, and o_4 from the input feature-map concatenated by the coordinate embedding (xy-map). The xy-map is the x and y coordinate for the spatial axis of the feature-map. The parameter maps of our mixture model, μ-map, σ-map, p-map, and π-map are obtained from o_1, o_2, o_3, and o_4, respectively. The mixture component is represented at each position on the spatial axis of the paramter-maps.

The μ-map is calculated from $o_1 \in \mathbb{R}^{h_m \times w_m \times 4}$ by the following procedure: $\mu = \mathcal{T}(\bar{\mu} + \mathcal{F}(o_1))$. Here, $\mathcal{F}$, $\bar{\mu}$ and $\mathcal{T}$ represent the xy-limit operation, the default coordinate, and the transformation function, respectively. The xy-limit operation, $\mathcal{F}$, illustrated in Fig. 3 plays a role of limiting the center coordinate of the bounding box not to deviate much from the default coordinate $\bar{\mu}$. This operation is implemented by applying $\tanh$ to the offset value of the center coordinates (xy) in o_1 and multiplying the limit factor f_{lim}. The default coordinates, $\bar{\mu}$, which are similar to conventional anchors, represent the default center position and scale of the bounding box. Note that the first two channels (xy) of the $\bar{\mu}$ are

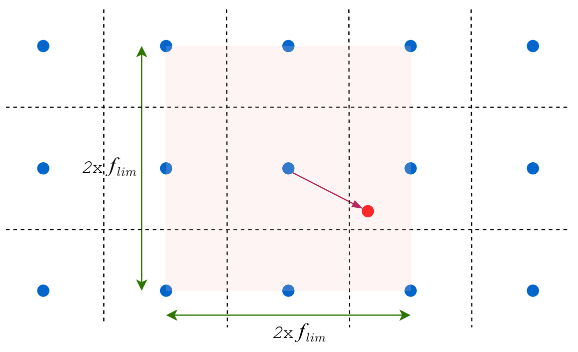

Figure 3. Illustration of xy-limit operation $\mathcal{F}$. The blue points denote the default coordinates $\bar{\mu}$. After $\mathcal{F}$, the $\bar{\mu}$ is moved to the red point μ_k by adding $\tanh(o_{1,xy}) \times f_{lim}$. This operation limits μ_k within the colored rectangular area.

xy-map of the corresponding size, and the last two channels (wh) are filled with a constant depending on the size of the input feature from the feature pyramid. The transformation $\mathcal{T}$ converts coordinates represented by the center, width, and height $(xywh)$ to the left-top and right-bottom corners $(ltrb)$. In this paper, we set f_{lim} equal to the spacing between adjacent $\bar{\mu}$'s on the μ-map (see Fig 3). We used one aspect ratio and five scales of bounding boxes as our $\bar{\mu}$, depending on the layer in the feature pyramid (see Fig. 2). The width and height of our $\bar{\mu}$ are calculated as $S \times (i/6)^2$ for all $i \in \{1, \cdots, 5\}$, where S is the coordinate range, which is defined by width and height of input image. The σ-map is obtained by applying the softplus [11] activation to o_2 and then adding the default std $\bar{\sigma}$. The $\bar{\sigma}$ is calculated as the width and height of μ multiplied by a predefined std-factor f_{std}. The $\bar{\sigma}$ prevents the mixture model from being sharp and σ becoming zero. Note that σ-map is for the $(ltrb)$-coordinate, not for the $(xywh)$-coordinate. The p-map is obtained by applying the softmax function along the channel axis to $o_3 \in \mathbb{R}^{h_m \times w_m \times (C+1)}$, and the π-map is obtained by applying the softmax to the entire five spatial maps of $o_4 \in \mathbb{R}^{h_m \times w_m \times 1}$.

Our network consists of a convolution layer of 3×3 kernel and three convolution layers of 1×1 kernel. Leaky ReLU [31] with a negative slope of 0.2 is used for the activation function of the 3×3 convolution layer. In this paper, we use RetinaNet's Feature Pyramid Network (FPN) [27, 26] as the backbone network. The MMOD estimates one mixture model from all levels of feature-maps outputted from the FPN. Thus, the number of components K is the summation of the number of components of each parameter-map corresponding to the feature-map. Here, each feature-map and parameter-map at the same layer have the same spatial dimension. The height of the first feature-map and the i-th feature-maps are calculated as $\lceil \frac{H}{8} \rceil$ and $\lceil \frac{h_{f,i-1}}{2} \rceil$, respectively, where H is height of the input image and $h_{f,i}$ is the height of the i-th feature-map. The widths of the feature-maps are also calculated in the same way.

4.2. Training

Likelihood compensation: The MMOD is trained using the negative log-likelihood of the estimated mixture model as a loss function, without any procedure such as ground truth (gt) assignment. Because of the nature of the mixture model that the sum of all the components' probabilities equals 1, i.e. $\sum_{k=1}^{K} \pi_k = 1$, the estimated probability of the i-th gt bounding box for a given image I, $p(b^i|I)$, would decrease by a factor of number of objects, N_{gt}, in the image. However, since object detection should be performed based on each object regardless of the number of objects in an image, this phenomenon of lower gt bounding box probability for a large N_{gt} is undesirable. In this paper, we alleviate this problem through *Likelihood compensation*, which multiplies the number of objects in the image, N_{gt}, to $p(b^i|I)$. Therefore, the loss function of the MMOD for the i-th gt object becomes

$$\mathcal{L}_{MM} = -\log \left(N_{gt} \times p(b^i|I) \right). \tag{4}$$

Confidence score through RoI sampling: Note that $\mathcal{L}_{MM}$ is calculated only by the gt bounding boxes, b_{gt}. Since b_{gt} generally does not include the background class, in our model, we cannot obtain the confidence score of a bounding box that takes background probability into account. Instead, we can consider the likelihood of an arbitrary bounding box as a confidence score. However, a likelihood only expresses the density of a bounding box, not its probability whether it is foreground or background or whether it belongs to a certain class. In addition, our model is difficult to evaluate the performance by a metric such as mean Average Precision (mAP) because only relative likelihood comparison of boxes can be performed in an image due to the hardness of assigning a likelihood threshold universal to all the images. Unlike the likelihood, π_k represents the probability of the corresponding mixture component, but likewise, only comparisons between bounding boxes on the same image are possible. Class probability considering background is generally used as a confidence of a bounding box, and it does not suffer from problems like likelihood and π mentioned above. To obtain the class probability that includes background class, we perform an additional sampling and labeling process. We sample bounding boxes from the mixture of Gaussian (MoG) ignoring the class probability from our mixture model. If the IoU between a sampled bounding box and a gt bounding boxes is above a threshold, we label it as the class of the gt with the highest IoU, otherwise, we label it as the background class. Through this sampling and labeling process, we create the region of interest (RoI) set $\{b_{roi}\}$. Since b_{roi} is stochastically acquired from the estimated MoG by the MMOD, we do not suffer from class

Training	Metric	$p(b\|I)$	$p(b\|I)'$	$\pi \times p(c)$	$\pi' \times p(c)$	$p(c)$
$\mathcal{L}_{MM}$ - $p(c)$	F_1 score (N_{gt})	21.6	21.6	52.8	52.8	27.5
wo/ bg.	*AP*	5.0	6.8	28.2	29.6	9.7
$\mathcal{L}_{Mod}$ - $p(c)$	F_1 score (N_{gt})	36.1	36.1	53.2	53.2	53.2
w/ bg.	*AP*	8.6	13.8	30.3	31.7	32.1

Table 1. Comparison of F_1 score and AP for serveral types of confidence score.

imbalance problems, and we can train the class probability by focusing more on the location where objects are more likely to exist.

Modified loss function: In order to train the network to represent the background probability using the $\{b_{roi}\}$, we re-define the loss function of MMOD into two terms. The first loss term is the negative log-likelihood of the MoG:

$$\mathcal{L}_{MoG} = -\log\left(N_{gt} \times \sum_{k=1}^{K} \pi_k \mathcal{N}(b^i_{ltrb}|I)\right). \quad (5)$$

The MMOD learns only the distribution of the coordinates of the ground truth bounding box $\{b_{ltrb}\}$, excluding class information using the MoG parameters (μ, σ and π) through $\mathcal{L}_{MoG}$. The second loss function is a complete form of the MMOD that includes class probability and is calculated as:

$$\mathcal{L}_{Cat} = -\log\left(N_{gt} \times p(b^j_{roi}|I)\right). \quad (6)$$

$\mathcal{L}_{Cat}$ is used to learn the class probability of the estimated mixture model. Note that $\mathcal{L}_{Cat}$ is identical to (4) except the fact that it is calculated on the different sets of bounding box candidates, i.e., it is trained using $b^j_{roi} \in \{b^1_{roi}, \cdots, b^{N_{roi}}_{roi}\}$ sampled from the estimated MoG. Also, it is trained such that the mixture of Gaussian is not relearned by itself. To this end, the error is not propagated to other parameters of mixture models except class probabilities. The final loss function is defined as:

$$\mathcal{L}_{Mod} = \mathcal{L}_{MoG} + \alpha \mathcal{L}_{Cat} \quad (7)$$

Here, α is a hyper-parameter controlling the balance between the two terms.

4.3. Inference

In the inference phase, we choose μ's of mixture components as coordinates of the predicted bounding boxes. We assume that these μ's have a high possibility to be near to the local maxima (modes) of the estimated mixture model by MMOD. In the aspect of MoG-based clustering, we consider the μ's as representative values for the corresponding Gaussian clusters. Before performing the non-maximum suppression (NMS), we filter out the mixture components with relatively low $p(c)$ or π values. Since the scale of π depends on the input image, we filter the mixture component through normalized-π (π'), which is calculated through min-max normalization where min value is zero: $\pi' = \pi / \max(\pi)$.

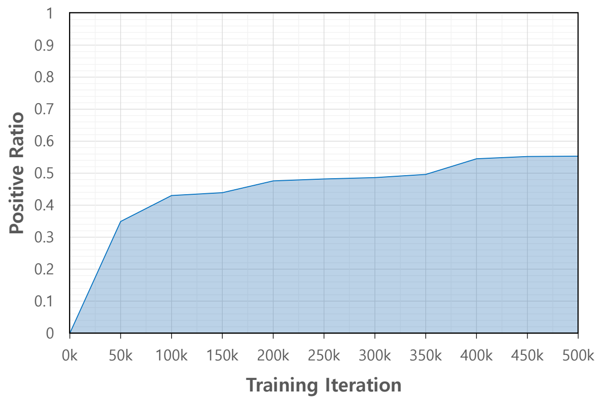

Figure 4. The ratio of positive and negative samples in the Mixture of Gaussian at each training iteration. Results obtained by inference of 100 mini-batches in each iteration on the training set.

5. Experiments

5.1. Details of Experiments

In our experiments, the MS COCO [28] and Pascal VOC [12] datasets are used for training and evaluation. For training our network, we use the stochastic gradient descent (SGD) optimizer with a momentum factor of 0.9. Gradient clipping [35] is applied with a cutoff threshold of 7.0. An ImageNet [9] pretrained ResNet-34, ResNet-50, or ResNet-101 [19] is used for our backbone network, and the remaining layers of the network are initailized with the Xavier-uniform intializer [16]. In order to generalize our network for various inputs, we augment the data with the following process. First we adjust the contrast and brightness of the image, then perform the expanding and cropping process and flip the data horizontally. This augmentation process is applied randomly as specified in [29, 13]. Unless otherwise specified in this section, we apply the *Likelihood compensation*, and the network is trained by the loss in (7) using RoIs sampled from MoG. The size of $\{b_{roi}\}$, N_{roi}, is five-times of N_{gt}. Thus, the class probability that includes background class is used as the confidence score of a bounding box. We set the S and α to 10.0 and 2.0. In inference phase, we perform NMS with the IoU threshold of 0.5 after filtering the bounding boxes with the class probability threshold of 0.001 and the π' threshold of 0.001. We implement MMOD in Pytorch [36]. We basically trained the network with a single GPU, and use 6-GPU only for the network using ResNet-101. Here, in order to reduce the effects of Batch Normalization [21] statistics, we use Cross-GPU Batch Normalization [37].

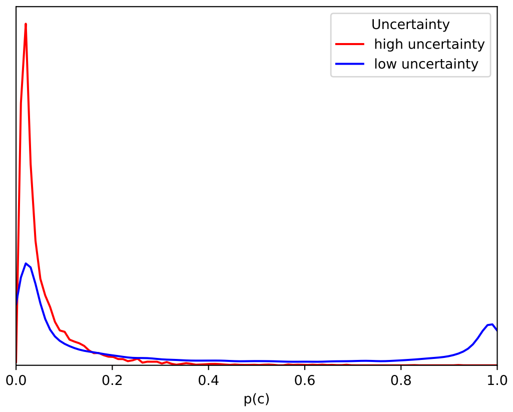

Figure 5. The distribution of the confidence score $p(c)$ according to uncertainty. In the graph, the red and blue line represents the high and low uncertainty, respectively.

5.2. Analysis of MMOD

For the analysis of our MMOD, we use the MS COCO 'train2017' as the training-set and 'val2017' dataset as the test set. Input images are resized to 320×320, and ResNet-50 is used for the backbone network. The initial learning rate is 0.005. The learning rate is decayed at iteration 350k, 430k and 470k with a decay rate 0.1, and the network is trained up to 500k iterations.

Confidence measure: In section 4.2, we considered serveral types of confidence measure of a bounding box. We perform the quantitative comparison for the following confidence measures: $p(b|I)$, $p(b|I)'$, $\pi \times p(c)$, $\pi' \times p(c)$ and $p(c)$. Here, $p(b|I)'$ and π' are normalized $p(b|I)$ and π, respectively. They are calculated by min-max normalization with zero min-value for the bounding boxes predicted on the same image. The results are measured from the network trained either by $\mathcal{L}_{MM}$ or $\mathcal{L}_{Mod}$. The $p(c)$ of the network trained by $\mathcal{L}_{Mod}$ contains the background probability. We compare the confidence measures through F1 score and AP (primary metric of MS COCO). The F1 score is calculated with IoU threshold of 0.5, using the N_{gt} predicted bounding boxes of top confidence in each image. Thus, unlike AP where alignment between all predicted bounding boxes of a dataset is important, only alignment between predicted bounding boxes in an image is required. Table 1 shows F1 scores and APs for different confidence measures. Compared with $\mathcal{L}_{MM}$, $\mathcal{L}_{Mod}$ shows better results for all kinds of confidence measures. All confidences based on $p(b|I)$ show low F1 score and AP, and are considered inappropriate criteria for object detection. Confidences based on π show better results than $p(b|I)$-based confidences. And, $\pi' \times p(c)$ shows improved AP results over $\pi \times p(c)$. In the netwotk trained by L_{Mod}, the F1 score is on the same level as $p(c)$ with backgorund probability. However, $\pi' \times p(c)$ shows a lower AP result compared to $p(c)$. This is conjectured as the result of the difficulty of comparison between π's on different images. The $p(c)$ without background shows a relatively low F1 score and AP, while $p(c)$ with background shows the best results among all the confidence measures.

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
without LC	32.0	52.8	33.4	15.2	34.7	48.5
with LC	32.1	53.0	33.7	15.7	34.5	49.3

Table 2. AP results with or without likelihood compensation on MS COCO val2017. 'LC' denote likelihood compensation.

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
5-scales	32.1	53.0	33.7	15.7	34.5	49.3
1-scale	31.7	52.6	33.1	14.7	34.4	48.9

Table 3. AP results for various pre-defined default coordinate $\bar{\mu}$ of MMOD on MS COCO val2017.

f_{std}	0	0.05	0.1	0.15
AP_{50}	49.3	50.4	53.0	52.5
AP	30.4	30.8	32.1	30.4

Table 4. Results of different std-factor (f_{std}) on MS COCO val2017.

Component	MMOD			
def-std ($\bar{\sigma}$)	✓	✓	✓	✓
xy-map	✓	✓	✓	
def-coor ($\bar{\mu}$)	✓	✓		
xy-limit ($\mathcal{F}$)	✓			
AP_{50}	53.0	52.2	51.0	44.6
AP	32.1	31.7	30.4	26.6

Table 5. The effect of the network component of MMOD on MS COCO val2017.

Positive-negative balance: Since we perform sampling from the estimated MoG, the sampled set $\{b_{roi}\}$ contains both positive and negative bounding boxes. In order to check the balance of positive and negative boxes in $\{b_{roi}\}$, we measure the positive ratio of $\{b_{roi}\}$ in 100 mini-batch of the tranining set at every 50k iterations. The positive ratio is calculated for each image, except for those with zero ground truth bounding box. In Figure 4, the positive ratio, which is initially low, increases as training progresses and converges to certain value. This shows that no positive-negative imbalance problem occurs while training. From this, the MMOD can be trained with stable positive-negative ratio without any special processing.

Relation of uncertainty (σ) and confidence $p(c)$: The σ estimates the uncertainty of the coordinate of a bounding box. In object detection problems, the confidence should reflect not only class probability but also uncertainty for

Method	Backbone	Input size	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	Speed/GPU
Faster R-CNN [26]	ResNet-101 FPN	short-800	36.2	59.1	39.0	18.2	39.0	48.2	172ms/M
Cascade R-CNN [3]	ResNet-101 FPN+	-	42.8	62.1	46.3	23.7	45.5	55.2	140ms/P
Grid R-CNN [30]	ResNet-101 FPN	short-800	41.5	60.9	44.5	23.3	44.9	53.1	-
YOLOv3 [40, 24]	DarkNet-53	608x608	33.0	57.9	34.4	18.3	35.4	41.9	39ms/P
SSD321 [27]	ResNet-101	321x321	28.0	45.4	29.3	6.2	28.3	49.3	61ms/M
SSD513 [27]	ResNet-101	513x513	31.2	50.4	33.3	10.2	34.5	49.8	125ms/M
DSSD321 [27]	ResNet-101	321x321	28.0	46.1	29.2	7.4	28.1	47.6	85ms/M
DSSD513 [27]	ResNet-101	513x513	33.2	53.3	35.2	13.0	35.4	51.1	156ms/M
RefineDet320 [48]	ResNet-101 TCB	320x320	32.0	51.4	34.2	10.5	34.7	50.4	-
RefineDet512 [48]	ResNet-101 TCB	512x512	36.4	57.5	39.5	16.6	39.9	51.4	-
RetinaNet800 [27, 6]	ResNet-101 FPN	short-800	39.1	59.1	42.3	21.8	42.7	50.2	104ms/P
FCOS [43]	ResNet-101 FPN	short-800	41.5	60.7	45.0	24.4	44.8	51.6	-
ScratchDet [53]	Root-ResNet-34	300x300	32.7	52.0	34.9	13.0	35.6	49.0	-
CornerNet$\circ\star$ [23]	Hourglass-104	511x511 (ori.)	40.6	56.4	43.2	19.1	42.8	54.3	244ms/P
ExtremeNe$\circ\star$ [51]	Hourglass-104	511x511 (ori.)	40.2	55.5	43.2	20.4	43.2	53.1	322ms/P
CenterNet$\circ\star$ [10]	Hourglass-104	511x511 (ori.)	44.9	62.4	48.1	25.6	47.4	57.4	340ms/P
MMOD320	ResNet-50 FPN	320x320	32.2	53.3	33.8	13.7	34.0	47.5	28ms/P
MMOD320	ResNet-101 FPN	320x320	33.7	54.9	35.6	13.9	36.1	50.0	32ms/P
MMOD512	ResNet-101 FPN	512x512	38.7	61.2	41.7	20.5	42.0	52.2	39ms/P

Table 6. Comparison of various results with MMOD on MS COCO dataset. '$\circ$' and '$\star$' denote soft-nms [2] and flip test [23], respectively. The 'short-x' means to use an image that shorter side is resized as x while maintaining the aspect ratio, and the 'ori.' means using the original size input image in test. The 'M' and the 'P' represents Maxwell and Pascal architecture based GPU, respectively.

bounding box coordinate. In our method, class probability, $p(c)$, is trained through the probability density function of the mixture model using $\{b_{roi}\}$ sampled from the estimated MoG, thus σ affects the traninig of $p(c)$. To show the change in confidence score $p(c)$ for different uncertainty, σ, we compare the distribution of the confidence for high and low uncertainty. We use $\frac{1}{4}\Sigma_d \frac{\sigma_d}{S_b}$ as the uncertainty score considering the scale of σ with respect to the size of the bounding box S_b (width or height), where $d \in \{l, t, r, b\}$. We set the bounding boxes which have the top 30% uncertainty as high uncertainty set and bounding boxes of the bottom 30% uncertainty as low uncertainty set, in the predicted bounding boxes obtained from the randomly sampled 300 images. As shown in Figure 5, for low uncertainty, $p(c)$ has both high and low values. On the other hand, $p(c)$ is mostly distributed at very low values, for the high uncertainty set. In this experiments, it is shown that the confidence score $p(c)$ considers not only the class probability but also the uncertainty for the bounding box coordinate.

Likelihood compensation: In table 2, we compare the object detection results according to whether likelihood compensation is performed or not on MS COCO evaluation metric. The 'without LC' and 'with LC' mean that the MMOD is trained with likelihood compensation and without likelihood compensation, respectively. In this table, the results 'with LC' are mostly better than the those 'without LC'. Especially, in the metric for small object AP_S and large object AP_L, the likelihood compensation shows noticeable improvement.

Flexibility of the MMOD: The role of the default coordinate $\bar{\mu}$ in our MMOD is similar to that of the anchors in that it represents pre-defined forms of bounding boxes. But, it differs in that the MMOD learns flexibly without assigning gt bounding box to a specific mixture component in the training process. Because of this flexibility, the shape of predicted bounding boxes by the MMOD is not limited by pre-defined forms. Table 3 shows the APs for two-pre-defined $\bar{\mu}$ with different scales. '1-scale' is defined as the half size of the coordinate range S and '5-scales' follows the description of the $\bar{\mu}$ in section 4.1. As can be seen in this table, the AP does not drop significantly even if fewer number of scales is used. In this experiment, MMOD shows relatively robust detection results in different settings of $\bar{\mu}$, compared to ground-truth-assignment-based methods that use the anchors. [41, 39]

Default std ($\bar{\sigma}$): The default-std $\bar{\sigma}$, which is determined by the hyper-parameter std-factor f_{std}, smooths the distribution of the mixture model, preventing the mixture model from being sharp for the given gt bounding boxes. To check how this affects network performance, we changed the f_{std}. Table 4 shows the AP for various f_{std}. Setting f_{std} to 0.1 yields the highest AP. In addition, when the f_{std} is zero, training becomes unstable by increasing possibility of zero σ, thus resulting in a more spiky shape of likelihood.

Method	Backbone	Input size	#Boxes	mAP	FPS
Faster R-CNN [41, 29]	VGG-16	short-600	300	73.2	7
CoupleNet [54]	ResNet-101	short-600	300	82.7	8.2
R-FCN [7]	ResNet-101	short-600	300	80.5	9
YOLOv2 [39]	DarkNet-19	544x544	845	78.6	40
SSD300* [29]	VGG-16	300x300	8732	77.2	46
SSD512* [29]	VGG-16	512x512	24564	79.8	19
DSSD321 [13]	ResNet-101	321x321	17080	78.6	9.5
DSSD513 [13]	ResNet-101	513x513	43688	81.5	5.5
RefineDet [48]	VGG-16 TCB	320x320	6375	80.0	40.3
RefineDet [48]	VGG-16 TCB	512x512	16320	81.8	24.1
ScratchDet [53]	Root-ResNet-34	300x300	8732	80.4	17.8
MMOD320	ResNet-34 FPN	320x320	2134	79.9	42.7
MMOD320	ResNet-50 FPN	320x320	2134	80.8	32.8
MMOD512	ResNet-50 FPN	512x512	5456	83.0	20.2

Table 7. Comparison of various results with MMOD on Pascal VOC dataset. The training and validation set of Pascal VOC 2007 and 2012 is used for training. And the test set of Pascal VOC 2007 is used for evaluation.

Ablation study: The MMOD has components that play a specific role in the intermidate feature-map. In this experiment, we changed each component of the MMOD architecture one by one to see the effect. The results is shown in Table 5. The MMOD that uses all architecture components shows the best performance. It is noteworthy that performance is significantly degraded when neither xy-map nor default coordinate $\bar{\mu}$ with spatial information is used.

5.3. Evaluation result comparison

MS COCO: For the evaluation of the MMOD, we perform the comparison with other object detection methods on MS COCO dataset. We use the MS COCO 'train2017' as the training-set and 'test-dev2017' dataset is used for evaluation of the MMOD. The network is trained according to the details specified in section 5.2. The processing time of MMOD is measured on a single nvidia Geforce 1080Ti GPU. Table 6 reports the MS COCO evaluation results and inference time for the MMOD and other methods. In this table, all of our proposed models show inference time of less than 40ms. Especially, the MMOD320 with ResNet-50 shows the fastest inference speed. The MMOD320 models outperform the detection performance of Refiendet320 even with lighter backbones. The MMOD512 produces the lower detection performances than keypoint-based methods (CornerNet, ExtremeNet and CenterNet), but its inference speed is more than six-times faster. These results show that our method is more competitive than other methods in terms of speed and performance trade-offs.

Pascal VOC: In addition, we also performed evaluation on PascalVOC, another representative object detection dataset. In experiments on the Pascal VOC, we use the PascalVOC '0712trainval' (union of PascalVOC 07 and 12 trainval-set) as the training-set. PascalVOC '07test' dataset is used for evaluation. the initial learning rate is 0.003. The learning rate is decayed at 40k and 70k with decay rate 0.1, and the maximum training iteration is set to 100K. For a fair comparison, the processing time of MMOD is measured on a single nvidia Titan X (Maxwell) GPU. Table 7 shows the mAP results and FPS for the MMOD and other object detection methods. In this table, the MMOD320 with ResNet-34 backbone is the fastest model and the MMOD512 produces the best mAP results. When considering both mAP and FPS, our models show the better mAP and FPS results than other methods. Also, our method predicts a relatively small number of bounding boxes, except for RPN (Region Proposal Network) based methods and YOLO v2 which finds anchors through clustering.

6. Conclusion

In this paper, we proposed a new multi-object detector named as Mixture-Model-based Object Detector (MMOD). Unlike previous multi-object detection methods, our MMOD estimates the density of bounding boxes for an input image using a mixture model. To capture this distribution correctly, we also proposed the mixture model whose components consist of Gaussian and categorical distributions. MMOD does not need the ground truth assignment process, and relatively a small number of mixture components are needed compared to most object detection methods. Also, the positive-negative imbalance problem does not occur due to our RoI sampling process. MMOD not only has the advantages mentioned above but also shows the state-of-the-art detection performance among the fast object detectors. Our method is a new approach to object detection and has a high potential for further research and development.

References

[1] Christopher M Bishop. Mixture density networks. Technical report, Citeseer, 1994. 3

[2] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569, 2017. 7

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. 7

[4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017. 2

[5] Kean Chen, Jianguo Li, Weiyao Lin, John See, Ji Wang, Lingyu Duan, Zhibo Chen, Changwei He, and Junni Zou. Towards accurate one-stage object detection with ap-loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5119–5127, 2019. 2

[6] Yuntao Chen, Chenxia Han, Naiyan Wang, and Zhaoxiang Zhang. Revisiting feature alignment for one-stage object detection. *arXiv preprint arXiv:1908.01570*, 2019. 7

[7] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 1, 8

[8] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 2

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 5

[10] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Object detection with keypoint triplets. *arXiv preprint arXiv:1904.08189*, 2019. 2, 7

[11] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. In *Advances in neural information processing systems*, pages 472–478, 2001. 4

[12] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015. 5

[13] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017. 5, 8

[14] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 2

[16] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 5

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015. 2

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5

[20] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2888–2897, 2019. 2

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 5

[22] Tao Kong, Fuchun Sun, Huaping Liu, Yuning Jiang, and Jianbo Shi. Foveabox: Beyond anchor-based object detector. *arXiv preprint arXiv:1904.03797*, 2019. 2

[23] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018. 2, 7

[24] Hei Law, Yun Teng, Olga Russakovsky, and Jia Deng. Cornernet-lite: Efficient keypoint based object detection. *arXiv preprint arXiv:1904.08900*, 2019. 7

[25] Buyu Li, Yu Liu, and Xiaogang Wang. Gradient harmonized single-stage detector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8577–8584, 2019. 2

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1, 4, 7

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 2, 4, 7

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 2, 5, 8

[30] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and Junjie Yan. Grid r-cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7363–7372, 2019. 7

[31] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. 4

[32] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287, 2017. 2

[33] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016. 2

[34] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 821–830, 2019. 2

[35] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013. 5

[36] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[37] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6181–6189, 2018. 5

[38] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4929–4937, 2016. 2

[39] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 1, 2, 7, 8

[40] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1, 2, 7

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 1, 2, 7, 8

[42] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016. 2

[43] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. *arXiv preprint arXiv:1904.01355*, 2019. 2, 7

[44] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014. 2

[45] Jiaqi Wang, Kai Chen, Shuo Yang, Chen Change Loy, and Dahua Lin. Region proposal by guided anchoring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2965–2974, 2019. 2

[46] Tong Yang, Xiangyu Zhang, Zeming Li, Wenqiang Zhang, and Jian Sun. Metaanchor: Learning to detect objects with customized anchors. In *Advances in Neural Information Processing Systems*, pages 320–330, 2018. 2

[47] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. *arXiv preprint arXiv:1904.11490*, 2019. 2

[48] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4203–4212, 2018. 7, 8

[49] Yuanyi Zhong, Jianfeng Wang, Jian Peng, and Lei Zhang. Anchor box optimization for object detection. *arXiv preprint arXiv:1812.00469*, 2018. 2

[50] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 2

[51] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 850–859, 2019. 2, 7

[52] Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. *arXiv preprint arXiv:1903.00621*, 2019. 2

[53] Rui Zhu, Shifeng Zhang, Xiaobo Wang, Longyin Wen, Hailin Shi, Liefeng Bo, and Tao Mei. Scratchdet: Training single-shot object detectors from scratch. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2268–2277, 2019. 7, 8

[54] Yousong Zhu, Chaoyang Zhao, Jinqiao Wang, Xu Zhao, Yi Wu, and Hanqing Lu. Couplenet: Coupling global structure with local parts for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4126–4134, 2017. 8