Training Multi-Object Detector by Estimating Bounding Box Distribution for Input Image – Supplimentary Meterial –

1. Details of MDOD

MDOD with static size input: For training MDOD with static size input, we use the stochastic gradient descent optimization with a learning rate of 0.005 and a momentum factor of 0.9. The learning rate is decayed at epochs 120 and 150 with a decay rate of 0.1, and the network is trained up to 160 epochs. Here, the batch size is 32. Gradient clipping [3] is applied with a cutoff threshold of 7.0. We perform the generally used data augmentation process: the expansion, cropping and the horizontal flip described in [2]. These are the same processes used in EfficientDet.

MDOD with variable size input: For training MDOD with variable size input (short-800), we use the ADAM optimizer [1] with a learning rate of 0.001. The learning rate is decayed at epoch 60 and 65 with a decay rate of 0.1. MDOD is trained up to 70 epoch and the batch size is 20. Unlike static size input setting, Gradient clipping is not applied. We apply the same augmentation strategies used in RetinaNet, and FCOS: the horizontal flip, and the scale jitter. For a fair comparison with the methods using 'short-800' such as RetinaNet, and FCOS, MDOD network uses the 10 convolution layers that have 256 channels, when using 'short-800' input. This is the same number of convolutions with RetinaNet, FCOS, and etc.

2. Considerations for fair comparison with Baselines

We came up with our own baseline detector model because the variables of each detector such as batch size, augmentation methods, network architecture and so on are so much different from detector to detector. We found it very hard to conduct fair and proper comparisons between various detectors due to these variables. Therefore, we designed our own baseline model with our own controlled variables so that we could perform fair experiments for comparison. The baseline model and MDOD share completely the same batch size, augmentation strategy, and network architecture except the output layer. Then, we tuned our baseline model by trying different hyper-parameters (positive-negative ratio, loss weight between regression and classification, weight-decay and learning rate) and tried to find the best parameters that show the best results for the baseline model (30.1 AP when using ResNet50 and 320x320 size input image).

3. Future Works and Broader Impacts

Modeling the distribution of bounding boxes: Which density model is used to estimate the probability distribution is an important thing in terms of density estimation. Changing the density model may cause practical and theoretical differences. For example, in our paper, the mixture model using the Cauchy distribution as a component shows higher detection performance (AP) than the Gaussian mixture model. Like this, in our framework, detection performance may vary depending on which model the distribution of the bounding box is estimated through, and there is a large room for performance improvement, depending on the design of the model.

Likelihood for any bounding box: In the previous methods, we can only know the class probability of the predicted bounding boxes. But we cannot obtain the statistical information of any bounding box that is not included in the set of predicted bounding boxes. However, in MDOD, the probability density function is cleary defined. We can get the likelihood for any bounding boxes from this probability density function of the mixture model estimated by MDOD. That is, we can obtain the likelihood of the bounding box for an arbitrary object by calculating the probability density function. This unique characteristic helps the analysis of the object detector, and opens up the possibility of applications to other fields, such as knowledge distillation in object detection. Also, by itself, this characteristic can be utilized as a function that receives a query bounding box from the user and returns the statistical information of it.

4. Pseudo-code

```
def mdod_training_procedure(mu, gamma, prob, pi, gt_box, gt_label, fg_th=0.5, roi_ratio=3):
    # {mu: (K, 4), gamma: (K, 4), prob: (K, #class), pi: (K, 1)}: network predictions.
    # {gt_box: (#GT, 4), gt_label: (#GT, #class)}: ground truth annotations.
    # fg_th: foreground threshold.
    # roi_ratio: the number of RoI per ground truth annotation.

    # calculate moc loss
    p_cau = cauchy_pdf(gt_box, mu, gamma) # (#GT, K)
    p_moc = sum(pi * p_cau, dim=1) # (#GT)
    moc_loss = -log(p_moc)

    # roi sampling
    n_roi = n_gt * roi_ratio
    roi_box = sampling_from_mu_pi(mu, pi, n=n_roi) # (#RoI)
    iou_pair = calculate_iou(roi_box, gt_box) # (#RoI, #GT)
    max_iou, max_idx = max(iou_pair, dim=1) # (#RoI), (#RoI)
    bg_idx = get_background_idx(max_iou < fg_threhsold) # (#RoI)
    roi_label = gt_label[max_idx] # (#RoI)
    roi_label[bg_idx] = 0.0

    # calculate mm loss
    p_cau_roi = cauchy_pdf(roi_box, mu, gamma) # (#RoI, K)
    p_cat_roi = category_pmf(roi_label, prob) # (#RoI, K)
    p_mm = sum(pi.detach() * p_cau_roi.detach() * p_cat_roi, dim=1) # (#RoI)
    mm_loss = -log(p_mm)
    return moc_loss, mm_loss
```

Figure 1: Pytorch-like pseudocode for the training procedure of MDOD.

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[2] 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

[3] 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. 1