Softer-NMS: Rethinking Bounding Box Regression for Accurate Object Detection

Yihui He[1] Xiangyu Zhang[2] Kris Kitani[1] Marios Savvides[1]
[1]Carnegie Mellon University [2]Megvii Inc. (Face++)
{he2,kmkitani,marioss}@andrew.cmu.edu zhangxiangyu@megvii.com

Abstract

*Non-maximum suppression (NMS) is essential for state-of-the-art object detectors to localize object from a set of candidate locations. However, accurate candidate location sometimes is not associated with a high classification score, which leads to object localization failure during NMS. In this paper, we introduce a novel bounding box regression loss for learning bounding box transformation and localization variance together. The resulting localization variance exhibits a strong connection to localization accuracy, which is then utilized in our new non-maximum suppression method to improve localization accuracy for object detection. On MS-COCO, we boost the AP of VGG-16 faster R-CNN from 23.6% to **29.1%** with a single model and nearly no additional computational overhead. More importantly, our method is able to improve the AP of ResNet-50 FPN fast R-CNN from 36.8% to **37.8%**, which achieves the state-of-the-art bounding box refinement result. Our code and models are available at https://github.com/yihui-he/softer-NMS.*

1. Introduction

Object detection is fundamental for many downstream practical computer vision applications like face recognition [59, 53], autonomous driving cars [10, 31], video indexing [51, 41] and video surveillance [2, 37]. Recent advances on deep convolutional neural network greatly improve the performance of object detection [9, 28, 57, 50, 52, 18, 55, 6, 33].

Recent object detection algorithms fall into two categories: efficient one-stage detectors like [43, 35] and high accuracy two-stage detectors like [13, 44]. This work focuses on the latter one. NMS is essential for state-of-the-art two-stage detectors, which generates cluttered bounding boxes. During NMS, candidate bounding boxes are ranked based on classification score. Bounding boxes of lower scores will be suppressed or given a decayed score by

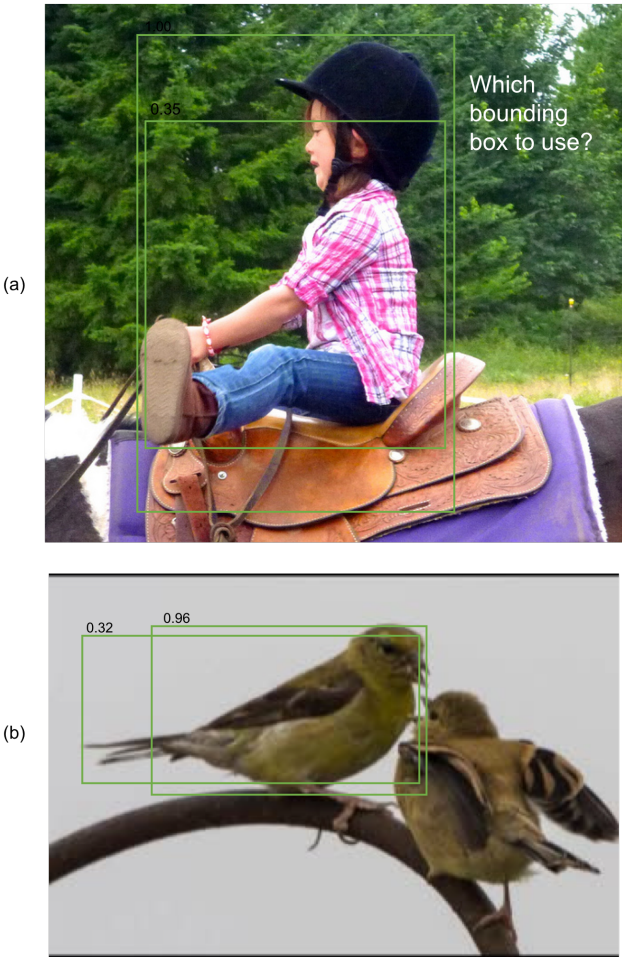

Figure 1. Illustration of failure cases of VGG-16 faster R-CNN on MS-COCO. (a) both candidate boxes are inaccurate in a certain coordinate. (b) high classification scored bounding box is inaccurate on the left boundary. (*better viewed in color*)

soft-NMS. We are interested in two situations that lead to detection failures: one situation occurs when all bounding boxes around an object is inaccurate in some coordinates, illustrated in Figure 1 (a); The other one is classification-

localization confidence inconsistency, in which a bounding box with the accurate location is associated with a low classification score, illustrated in Figure 1 (b). These detection failures suggest that localization confidence is not strongly related to classification confidence.

Motivated by this, we propose a novel bounding box regression loss, namely KL Loss, for learning bounding box transformation and localization confidence at the same time. Specifically, we first model the bounding box prediction and ground truth bounding box as Gaussian distribution and Dirac delta function respectively. Then, we train the detection network to minimize our new bounding box regression loss derived from the KL divergence of these two distributions. At last, we propose to weighted-average a candidate box and its neighbors with the predicted standard deviations during soft NMS, such that the bounding boxes with higher confidence get heavier weights regardless of the classification confidence.

To demonstrate the broad and general applicability of KL Loss and softer-NMS, we evaluate on both PASCAL VOC 2007 and MS-COCO using various neural networks including ZF, VGG-M, VGG-16, and ResNet-50 RPN. Our experiments suggest that our approach offers better object localization accuracy for CNN object detectors. For VGG-16 faster R-CNN on MS-COCO, we improve the AP from 23.6% to **29.1%**, with only 2ms increased latency on the GPU (GTX 1080ti). Furthermore, we apply this pipeline to ResNet-50 FPN fast R-CNN and achieve **1.0%** better AP than the baseline, which outperforms the previous state-of-the-art bounding box refinement algorithm [26].

2. Related Work

There has been a significant amount of works on object detection.

Two-stage Detectors: Although one-stage detection algorithms [35, 43, 29] are efficient, state-of-the-art object detectors are based on two-stage, proposal-driven mechanism [44, 7, 8, 17, 30, 3]. Two-stage detectors generate cluttered object proposals, which result in a large number of duplicate bounding boxes. However, during standard NMS procedure, bounding boxes with lower classification scores will be discarded even though their locations are accurate. Our softer-NMS tries to utilize these boxes based on localization confidence for better localization of the selected boxes.

Object Detection Loss: To better learn object detection, different kind of losses were proposed. UnitBox [56] introduced an Intersection over Union (IoU) loss function for bounding box prediction. Focal Loss [33] deals with the class imbalance by changing the standard cross entropy loss such that well-classified examples are assigned lower weights. [42] optimizes for the mAP via policy gradient for learning globally optimized object detector. [27] proposes to weight multi-task loss by considering the uncertainty of each task, whereas we consider uncertainty in the localization level.

Non-Maximum Suppression: NMS has been an essential part of computer vision for many decades. It is widely used in edge detection [46], feature point detection [36] and objection detection [14, 13, 44, 47].

Recently, soft NMS and learning NMS [1, 21] are proposed for improving NMS results. Instead of eliminating all lower scored surrounding bounding boxes, soft-NMS [1] decays the detection scores of all other neighbors as a continuous function of their overlap with the higher scored bounding box. Learning NMS [21] proposed to learn a new neural network to perform NMS using only boxes and their scores. Our softer-NMS averages the selected boxes in a "softer" way different from selecting boxes or changing scores, which can work with soft-NMS well (Table 2 and Table 4).

Bounding Box Refinement: Box voting [12] is first proposed as a "by-product" of its iterative localization scheme, which merges candidate boxes according to classification scores. Relation network [23] proposes to learn the relation between bounding boxes. Recently, IoU-Net [26] proposes to learn the IoU between the predicted bounding box and the ground truth bounding box. IoU-NMS is then applied to the detection boxes, guided by the learned IoU. Different from IoU-Net, we propose to learn the localization variance from a probabilistic perspective. It enables us to learn variances for the four coordinates of a predicted bounding box separately instead of only IoU. From our experiment, we found that the simple combination of soft-NMS and box voting [12] is comparable to IoU-Net and IoU-NMS, whereas our approach improves the baseline by a large margin. (Table 4).

3. Approach

In this section, we first propose KL Loss for training detection network with localization confidence. Then a new NMS approach is introduced for improving localization accuracy with our confidence estimation.

3.1. Bounding Box Regression with KL Loss

Figure 2 illustrates the head part of our network architecture for object detection. We aim to estimate the localization confidence along with location, which will be helpful for NMS. Specifically, our network predict a Gaussian distribution instead of only bounding box location:

$$P_\Theta(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_e)^2}{2\sigma^2}} \tag{1}$$

where x_e is the estimated bounding box location. Standard deviation σ measures uncertainty of the estimation. It is

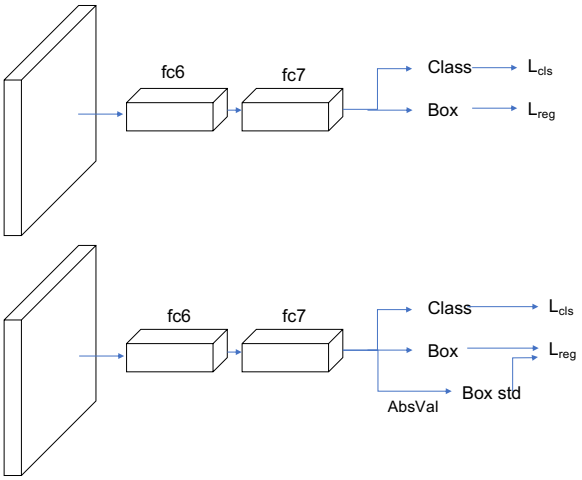

Figure 2. Our network architecture for estimating localization confidence. Up: standard fast R-CNN head part of a two stage detection network, Down: the head part of our network). The estimated standard deviations along with bounding box locations are considered in regression loss L_{reg}

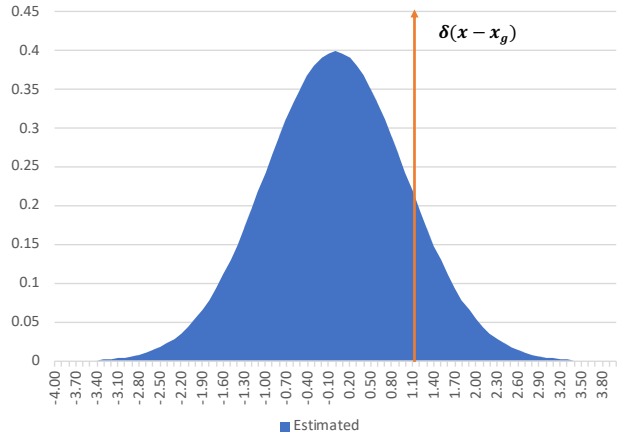

Figure 3. The Gaussian distribution in blue is our estimation. The Dirac delta function in orange is the distribution of the ground truth bounding box. When the location x_e is estimated inaccurately, we expect the variance σ^2 be smaller so that L_{reg} will be lower. (*better viewed in color*)

produced by a fully-connected layers on top of the fast R-CNN head (`fc7`) and an absolute value layer. We use absolute value instead of ReLU [38] to avoid most σ from being 0. When $\sigma \rightarrow 0$, it means our network is extremely confident about estimated bounding box location.

The ground truth bounding box can also be formulated as a Gaussian distribution, with $\sigma \rightarrow 0$, which is actually a Dirac delta function:

$$P_D(x) = \delta(x - x_g) \tag{2}$$

where x_g is the ground truth bounding box location.

The goal of object localization in our circumstance is to estimate $\hat{\Theta}$ that minimize the KL-Divergence between $P_\Theta(x)$ and $P_D(x)$ [45]:

$$\hat{\Theta} = \arg\min_{\Theta} D_{KL}(P_D(x)||P_\Theta(x)) \tag{3}$$

We use the KL-Divergence as the loss function L_{reg} for bounding box regression. The classification loss L_{cls} remains the same.

$$\begin{aligned} L_{reg} &= D_{KL}(P_D(x)||P_\Theta(x)) \\ &= \int P_D(x) log \frac{P_D(x)}{P_\Theta(x)} dx \\ &= -\int P_D(x) log P_\Theta(x) dx + \int P_D(x) log P_D(x) dx \\ &= -\int P_D(x) log P_\Theta(x) dx + H(P_D(x)) \\ &= -log P_\Theta(x_g) + H(P_D(x)) \\ &= \frac{(x_g - x_e)^2}{2\sigma^2} + \frac{1}{2} log(\sigma^2) + \frac{1}{2} log(2\pi) + H(P_D(x)) \end{aligned} \tag{4}$$

Shown in Figure 3, when the location x_e is estimated inaccurately, we expect the variance σ^2 to be smaller so that L_{reg} will be lower. $\frac{1}{2}log(2\pi) + H(P_D(x))$ has nothing to do with our decision, since it does not depend on the estimated parameters Θ. Therefore:

$$D_{KL}(P_D(x)||P_\Theta(x)) \propto \frac{(x_g - x_e)^2}{2\sigma^2} + \frac{1}{2} log(\sigma^2) \tag{5}$$

When $\sigma = 1$, KL Loss degenerates to the standard Euclidean loss:

$$D_{KL}(P_D(x)||P_\Theta(x)) \propto \frac{(x_g - x_e)^2}{2} \tag{6}$$

The loss is differentiable w.r.t location estimation x_e and location confidence σ:

$$\begin{aligned} \frac{d}{dx_e} D_{KL}(P_D(x)||P_\Theta(x)) &= \frac{x_e - x_g}{\sigma^2} \\ \frac{d}{d\sigma} D_{KL}(P_D(x)||P_\Theta(x)) &= -\frac{(x_e - x_g)^2}{\sigma^{-3}} - \frac{1}{\sigma} \end{aligned} \tag{7}$$

However, since σ is in the denominators, the gradient sometimes can explode at the beginning of training. To avoid this, our network predicts $\alpha = 1/\sigma^2$ instead of σ in practice. Following batch normalization [24], we also add a small constant $\epsilon = 0.0001$ to the log term to avoid it from being negative infinite.

$$\begin{aligned} D_{KL}(P_D(x)||P_\Theta(x)) &\propto \frac{\alpha}{2}(x_g - x_e)^2 - \frac{1}{2} log(\alpha + \epsilon) \\ \frac{d}{dx_e} D_{KL}(P_D(x)||P_\Theta(x)) &= \alpha(x_e - x_g) \\ \frac{d}{d\alpha} D_{KL}(P_D(x)||P_\Theta(x)) &= \frac{(x_e - x_g)^2}{2} - \frac{1}{2(\alpha + \epsilon)} \end{aligned} \tag{8}$$

For $|x_g - x_e| > 1$, we adopt a loss similar to the smooth L_1 loss defined in Fast R-CNN [13]:

$$L_{reg} = \alpha(|x_g - x_e| - \frac{1}{2}) - \frac{1}{2}log(\alpha + \epsilon) \quad (9)$$

To fully utilize our loss, we adopt parameterizations of the $x1, y1, x2, y2$ coordinates instead of the x, y, w, h coordinates used by R-CNN [14].

$$\begin{aligned} t_{x1} &= \frac{x1 - x1_a}{w_a}, t_{x2} = \frac{x2 - x2_a}{w_a} \\ t_{y1} &= \frac{y1 - y1_a}{h_a}, t_{y2} = \frac{y2 - y2_a}{w_a} \\ t^*_{x1} &= \frac{x1^* - x1_a}{w_a}, t^*_{x2} = \frac{x2^* - x2_a}{w_a} \\ t^*_{y1} &= \frac{y1^* - y1_a}{h_a}, t^*_{y2} = \frac{y2^* - y2_a}{w_a} \end{aligned} \quad (10)$$

where $t_{x1}, t_{y1}, t_{x2}, t_{y2}$ are the predicted locations (*i.e.*, x_e in Equation 1). $t^*_{x1}, t^*_{y1}, t^*_{x2}, t^*_{y2}$ are the ground truth locations (*i.e.*, x_g in Equation 2). $x1_a, x2_a, y1_a, y2_a, w_a, h_a$ are from the anchor box. $x1, y1, x2, y2$ are from the predicted box. Note that the standard deviation σ is predicted after boxes are normalized.

3.2. Softer-NMS

After we obtain the standard deviation of predicted location, merging the bounding boxes via the weighted mean is intuitive. Shown in Algorithm 1, we change NMS with two lines of code:

Algorithm 1 softer-NMS

$\mathcal{B}$ is $N \times 4$ matrix of initial detection boxes. $\mathcal{S}$ contains corresponding detection scores. $\mathcal{C}$ is $N \times 4$ matrix of corresponding variances. N_t is the softer NMS threshold. The lines in blue and in green are soft-NMS and softer-NMS respectively.

$\mathcal{B} = \{b_1, .., b_N\}, \mathcal{S} = \{s_1, .., s_N\}, \mathcal{C} = \{\sigma^2_1, .., \sigma^2_N\}, N_t$
$\mathcal{D} \leftarrow \{\}$
$\mathcal{T} \leftarrow \mathcal{B}$
while $\mathcal{T} \neq$ empty **do**
  $m \leftarrow$ argmax $\mathcal{S}$
  $\mathcal{M} \leftarrow b_m$
  $\mathcal{T} \leftarrow \mathcal{T} - \mathcal{M}$
  $\mathcal{S} \leftarrow \mathcal{S}f(IoU(\mathcal{M}, T))$ ▷ soft-NMS
  $idx \leftarrow IoU(\mathcal{M}, B) \geq N_t$ ▷ softer-NMS
  $\mathcal{M} \leftarrow \mathcal{B}[idx]/\mathcal{C}[idx]/sum(1/\mathcal{C}[idx])$
  $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}$
end while
return $\mathcal{D}, \mathcal{S}$

First, standard NMS or soft-NMS [1] is applied to candidate bounding boxes $\mathcal{T}$ and produce $\mathcal{M}$, $\{x1_i, y1_i, x2_i, y2_i, s_i, \sigma_{x1,i}, \sigma_{y1,i}, \sigma_{x2,i}, \sigma_{y2,i}\}$. For each box $\mathcal{M}$, we then compute its new location based on weighted mean of its neighbor bounding boxes and itself. Formally, for example, new $x1$ coordinate for ith box $x1_i$ is computed as follow:

$$\begin{aligned} x1_i &:= \frac{\sum_j x1_j/\sigma^2_{x1,j}}{\sum_j 1/\sigma^2_{x1,j}} \\ &\text{subject to } IoU(x1_j, x1_i) > N_t \end{aligned} \quad (11)$$

Bounding boxes with overlap threshold over N_t will be considered into the weighted mean. We don't set a classification score threshold, since lower scored boxes may have higher localization score. In Figure 4, we provide a visual illustration of softer-NMS. With softer-NMS, the two aforementioned situations that lead to detection failure can sometimes be avoided.

3.3. Joint Training

From our early experiments, we observe that training from scratch with KL Loss is unstable. We hypothesize that the instability is introduced by the second order term $\alpha(x_g - x_e)^2/2$ in our loss [54] (α and x_e are both from `fc7` feature and element-wisely multiplied). Therefore, we first warm up the network without location confidence estimation in the first stage of training. Then we incorporate location confidence estimation and follow the same training procedure as faster R-CNN [44] (*e.g.*, for VGG-16 faster R-CNN on COCO, we train for 350k iterations without σ estimation and then train for 490k iterations with σ estimation). The learning rate is divided by 10 after each training stage.

4. Experiments

We conduct rich experiments on both PASCAL VOC 2007 [11] and MS-COCO [34] datasets. We use four GPUs for our experiments. The training schedule and batch size are adjusted according to the linear scaling rule [16]. For ZF Net [57], VGG-M Net and VGG-16 Net [50], our implementation is based on Caffe [25]. For ResNet-50 FPN [18, 32], our implementation is based on Detectron [15, 40].

We initialize the weights of the FC layer for bounding box confidence prediction with random Gaussian distribution. The standard deviation and mean are set to 0.0001 and 1 respectively, so that the loss will be similar to the standard smooth L1 loss at the beginning (Equation 6).

4.1. Experiments on MS-COCO

For VGG-16 [50] faster R-CNN, following `py-faster-rcnn`[1], we train on `train2014` and

[1] https://github.com/rbgirshick/py-faster-rcnn

Figure 4. Results of softer NMS with VGG-16 faster R-CNN on MS-COCO. The blue number in the middle of each boundary is the corresponding standard deviation σ we predicted (Equation 1). Two failure situations corresponding to Figure 1 that can be improved by softer-NMS: (a) When each candidate bounding box is inaccurate in some coordinates (women on the right), our softer-NMS can incorporate their localization confidence and produce better boxes. (b) The bounding box with a higher classification score (train 0.99) actually has lower localization confidence than the bounding box with a lower classification score (train 0.35). After softer-NMS, the box scored 0.99 moves towards the correct location. (*better viewed in color*)

test on `val2014`. For FPN [32] ResNet-50 [18], we train and test on the newly defined `train2017` and `val2017` respectively. Unless specified, all hyper-parameters are set to default.

4.1.1 Softer-NMS

First, we analyze the sensitivity of the overlap threshold we introduced in Section 3.2, which can affect the performance. We train a VGG-16 faster R-CNN network with KL Loss. Then we apply softer-NMS with different overlap thresholds and soft-NMS. In Table 1, our network's results with overlap threshold from 0.0 to 1.0 are shown (threshold 1.0 is the same as no softer-NMS). We found that the range of acceptable overlap threshold is normal (around $0.5 \sim 0.8$). In the following experiments, we set the overlap threshold to 0.7 unless specified.

4.1.2 Ablation Study

We evaluate the contribution of each element in our detection pipeline: `x1, y1, x2, y2` coordinates, KL Loss, soft-NMS and softer-NMS with VGG-16 faster R-CNN. The detailed results are shown in Table 2.

x1, y1, x2, y2 Coordinates: Unexpectedly, simply training with `x1, y1, x2, y2` coordinates instead of the standard `x, y, w, h` coordinates improves the AP by 0.7%. We found that the AP^{50} is not increased, however AP^{50},

IoU threshold	AP	AP^{50}	AP^{75}	AP^S	AP^M	AP^L	AR^1	AR^{10}	AR^{100}	AR^S	AR^M	AR^L
0.0	20.8	38.4	20.2	7.2	23.7	30.9	24.2	35.2	38.7	14.3	42.7	60.6
0.1	24.6	44.7	24.2	8.3	28.2	36.8	25.1	38.2	41.7	15.8	47.2	63.8
0.2	26.5	47.2	26.4	8.6	30.2	39.8	25.6	39.8	43.1	16.3	49.0	65.7
0.3	27.5	48.3	27.7	8.7	31.2	41.4	25.8	40.7	43.9	16.5	49.9	67.0
0.4	28.2	48.9	28.7	8.9	31.9	42.5	26.0	41.4	44.4	16.7	50.6	67.9
0.5	28.7	**49.1**	29.6	**9.0**	32.3	43.4	26.1	41.9	44.9	16.9	51.1	68.6
0.6	29.0	**49.1**	30.3	**9.0**	**32.7**	44.0	**26.2**	42.3	45.3	**17.1**	51.6	68.9
0.7	**29.1**	**49.1**	**30.4**	8.7	**32.7**	44.3	**26.2**	**42.5**	**45.5**	**17.1**	**51.9**	69.2
0.8	28.8	49.0	29.7	8.4	32.0	**44.4**	26.0	42.4	**45.5**	**17.1**	**51.9**	**69.3**
0.9	28.1	49.0	28.7	8.4	31.0	43.3	25.5	41.9	45.2	17.0	51.4	68.7
1.0	27.8	49.0	28.5	8.4	30.9	42.7	25.3	41.7	44.9	17.0	51.3	68.1

Table 1. Analysis of the IoU threshold N_t sensitivity with VGG-16 faster R-CNN on MS-COCO. (KL Loss and soft-NMS are already applied)

xyxy	KL Loss	soft-NMS	softer-NMS	AP	AP^{50}	AP^{75}	AP^S	AP^M	AP^L	AR^1	AR^{10}	AR^{100}
				23.6	44.6	22.8	6.7	25.9	36.3	23.3	33.6	34.3
		✓		24.8	45.6	24.6	7.6	27.2	37.6	23.4	39.2	42.2
✓				24.3	44.6	24.0	6.9	26.5	37.3	24.0	34.6	35.2
✓	✓			26.4	47.9	26.4	7.4	29.3	41.2	25.2	36.1	36.9
✓	✓	✓		27.8	49.0	28.5	8.4	30.9	42.7	25.3	41.7	44.9
✓	✓	✓	✓	**29.1**	**49.1**	**30.4**	**8.7**	**32.7**	**44.3**	**26.2**	**42.5**	**45.5**

Table 2. The contribution of each element in our detection pipeline. The baseline model is VGG-16 Faster R-CNN. (xyxy denotes using x1, y1, x2, y2 coordinates for training and testing)

method	latency (ms)
baseline	99
softer-NMS	101

Table 3. Inference time comparison on MS-COCO with VGG-16 Faster R-CNN on a GTX 1080ti GPU, CUDA 8 and CUDNN 6

AP^M and AP^L are improved a lot. This indicates that the improvement is gained from more accurate object localization. We hypothesize that learning location transformation (x, y) is easier than learning scale transformation (w, h), since learning (w, h) is actually learning $(log(w), log(h))$.

KL Loss: Training with KL Loss further improves the AP by 2.1%, which is also observed on ResNet-50 FPN (Table 4). On the one hand, KL Loss incorporates learning localization confidence which can potentially help the network to learn more discriminative features. On the other hand, this may be caused by the longer training schedule, since nearly no additional computation complexity is introduced into our network. (training 61% more iterations improved the AP by 1.4% in faster R-CNN implementation report [4]).

Soft-NMS: As expected, soft-NMS performs consistently on both baseline and our network trained with KL Loss. It improves the AP by 1.2% and 1.4% on the baseline and our network respectively.

Softer-NMS: Finally, with softer-NMS, the AP is further improved to **29.1%**. Notice that the AP^{50} is only improved by 0.1%. However, AP^{75}, AP^M and AP^L are improved by 1.8%, 1.8%, and 1.6% respectively. We can safely draw the conclusion that the contribution to improvement mainly comes from the more accurate localization. We also found that softer-NMS and soft-NMS can work well with each other. Soft-NMS is good at scoring candidate bounding boxes which improves overall performance, whereas softer-NMS is good at refining those selected bounding boxes for more accurate object localization.

4.1.3 Inference Latency

We also evaluate the inference time of our improved VGG-16 faster R-CNN on a single GTX 1080ti GPU with CUDA 8 and CUDNN 6, as it is crucial for detection applications [22, 58, 20, 19]. Shown in Table 3, we only increase 2ms latency on GPU because only a 4096×324 fully-connected layer and an AbsVal layer are added for the localization confidence prediction. As for NMS processing, our implementation based on single thread Python is slow. However, it can be significantly optimized by implementing

type	iter	method	AP	AP^{50}	AP^{75}	AP^S	AP^M	AP^L
fast	90k	-	36.4	58.4	39.3	20.3	39.8	48.1
		voting	36.8	**58.5**	39.5	20.3	40.1	48.6
		soft-NMS	37.1	**58.5**	40.6	**20.7**	40.5	48.8
		soft-NMS+voting	37.2	58.4	40.7	20.6	**40.6**	49.2
	180k	-	36.8	58.4	39.5	19.8	39.5	49.5
	270k	-	36.8	58.3	39.6	20.0	39.5	49.6
	270k	KL Loss (ours)	37.0	57.2	39.9	19.8	39.7	50.1
	180k	voting	37.1	58.4	40.0	20.0	40.0	50.0
	90k+?	IoU-NMS	37.3	56.0	-	-	-	-
	180k	soft-NMS	37.4	58.2	41.0	20.3	40.2	50.1
	180k	soft-NMS+voting	37.5	58.3	41.0	20.3	40.3	50.4
	270k	**soft-NMS+KL Loss+softer-NMS (ours)**	**37.8**	57.1	**41.3**	20.3	40.4	**51.2**
end-to-end	90k	-	36.7	58.4	39.6	21.1	39.8	48.1
		voting	37.1	58.5	39.9	21.3	40.2	48.8
		soft-NMS	37.4	58.6	40.9	21.6	40.6	48.9
		soft-NMS+voting	37.7	58.6	41.1	21.7	40.9	49.5
	160k	IoU-Net	37.0	58.3	-	-	-	-
	160k	IoU-Net+IoU-NMS	37.6	56.2	-	-	-	-
	180k	-	37.9	59.2	41.1	21.5	41.1	49.9
	160k	IoU-Net+IoU-NMS+Refine	38.1	56.3	-	-	-	-
	160k	voting	38.2	59.3	41.3	21.7	41.3	50.7
	180k	soft-NMS	38.6	**59.3**	42.4	21.9	41.9	50.7
	180k	soft-NMS+voting	**38.9**	**59.3**	**42.6**	**22.0**	**42.0**	**51.4**

Table 4. Performance comparison with FPN ResNet-50 on MS-COCO (*sorted by AP*). fast: fast R-CNN; end-to-end: faster R-CNN; iter: number of iterations; voting: box voting [12]

in CUDA or C++ [15, 39, 5].

4.1.4 Experiments with ResNet-50 FPN

We further evaluate our approach on the recent feature pyramid network (ResNet-50 FPN) [32, 18]. Improving performance for ResNet-50 FPN is more challenging than for VGG-16 faster R-CNN. Modern networks like ResNet, Xception and ResNeXt [18, 6, 55] are designed for both high performance and efficiency. In addition, the training schedule is improved [15]. By training 180k iterations, fast R-CNN version AP can be improved from 36.4% to 36.8%. By training 180k end-to-end, the AP is improved to 37.9%. Recent state-of-the-art end-to-end bounding box refinement approach IoU-Net [26] only improves the AP by 0.2% over the end-to-end baseline, which is even worse than the simple combination of soft-NMS and bounding box voting [1, 12] in our experiments (Table 4).

Different from VGG-16 faster R-CNN, the overlap threshold N_t is set to 0.9 instead of the default threshold 0.7. From our experiments, the default threshold doesn't work well on ResNet-50 FPN. It does not improve the AP at all when we apply it after soft-NMS. We hypothesize that since ResNet-50 FPN is more accurate than VGG-16 faster R-CNN, the original candidate bounding boxes are already of high quality. So that merging farther boxes may do more harm. Limited by computing resources and time, we have not completed the experiments on the end-to-end version.

For fast R-CNN version, we first train a standard ResNet-50 FPN with `x1,y1,x2,y2` coordinates for 90k iterations as our initialization, which we found helpful for faster convergence. Then we train with KL Loss for 180k iterations, which increases the 180k iterations baseline by **0.2%**. We argue that naively training for 90k+180k iterations will not improve the performance, shown in Table 4. The performance gain is from KL Loss instead of longer training time.

After applying softer-NMS along with soft-NMS, we achieve the state-of-the-art bounding box refinement result with **37.8%** AP. Notice that we improve the performance via more accurate object localization (*i.e.*, the AP^{75}), even though our AP^{50} is lower than other methods. Compared with IoU-NMS [26], our approach is 0.5% better. Compared with voting [12] (weighted mean according to classification score), our approach is 0.3% better. This indicates that classification confidence is not strongly related to localization confidence. Therefore, learning localization confidence apart from classification confidence is important for more accurate object localization.

backbone	method	mAP
ZF	-	60.9
	KL Loss	60.9
	KL Loss+soft-NMS+softer-NMS	**62.3**
VGG-M	-	60.4
	KL Loss	60.4
	KL Loss+softer-NMS	**60.7**
VGG-16	-	68.7
	QUBO (tabu) [48]	60.6
	QUBO (greedy) [48]	61.9
	soft-NMS [1]	70.1
	KL Loss	68.6
	KL Loss+softer-NMS	69.1
	KL Loss+soft-NMS+softer-NMS	**70.3**

Table 5. Comparisons of different approaches on PASCAL VOC 2007 with Faster R-CNN.

4.2. Experiments on PASCAL VOC 2007

We also conduct experiments on PASCAL VOC 2007 with Faster R-CNN [44]. Backbone networks: ZF [57], VGG-M and VGG-16 [50] are tested. We train on `voc_2007_train` and test on `voc_2007_val`.

Shown in Table 5, we compare soft-NMS, quadratic unconstrained binary optimization (QUBO [48]), and our approach. For QUBO, we test both greedy and classical tabu solver (we have already manually tuned the penalty term for both solvers). We observe that it is much worse than standard NMS, though it was reported to be better for pedestrian detection. We hypothesize that QUBO is better at pedestrian detection since there are more occluded bounding boxes [49]. For VGG-M Net, we skip soft-NMS and apply only softer-NMS, which improves the mAP by 0.3%. For ZF and VGG-16, our approach improves the mAP by 1.4% and 1.6% respectively. We notice that softer-NMS could still improve performance even after soft-NMS is applied to the network. This observation is consistent with our experiments on MS-COCO (Table 2).

5. Conclusion

To conclude, classification confidence is not strongly related to localization confidence, which needs to be trained separately. In this paper, a novel bounding box regression loss along with a new NMS method is proposed for more accurate object localization. By training with KL Loss, the network learns to predict localization variance for each coordinate. This empowers softer-NMS, which refines the selected bounding boxes via weighted mean. Compelling results are demonstrated for VGG-16 faster R-CNN and ResNet-50 FPN on both MS-COCO and PASCAL VOC 2007. Our code and models are available at https://github.com/yihui-he/softer-NMS.

References

[1] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Soft-nmsimproving object detection with one line of code. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 5562–5570. IEEE, 2017. 2, 4, 7, 8

[2] T. Bouwmans and E. H. Zahzah. Robust pca via principal component pursuit: A review for a comparative evaluation in video surveillance. *Computer Vision and Image Understanding*, 122:22–34, 2014. 1

[3] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. *arXiv preprint arXiv:1712.00726*, 2017. 2

[4] X. Chen and A. Gupta. An implementation of faster rcnn with study for region sampling. *arXiv preprint arXiv:1702.02138*, 2017. 6

[5] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014. 6

[6] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint*, pages 1610–02357, 2017. 1, 7

[7] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 2

[8] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. *CoRR, abs/1703.06211*, 1(2):3, 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 *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 1

[10] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 304–311. IEEE, 2009. 1

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 4

[12] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1134–1142, 2015. 2, 7

[13] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 1, 2, 4

[14] R. Girshick, J. Donahue, T. Darrell, and J. 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, 4

[15] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. `https://github.com/facebookresearch/detectron`, 2018. 4, 6, 7

[16] 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. *arXiv preprint arXiv:1706.02677*, 2017. 4

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017. 2

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 4, 7

[19] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018. 6

[20] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1389–1397, 2017. 6

[21] J. H. Hosang, R. Benenson, and B. Schiele. Learning non-maximum suppression. In *CVPR*, pages 6469–6477, 2017. 2

[22] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 6

[23] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. *arXiv preprint arXiv:1711.11575*, 8, 2017. 2

[24] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[25] 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 *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. 4

[26] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018. 2, 7

[27] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*, 3, 2017. 2

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

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

[30] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun. Light-head r-cnn: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264*, 2017. 2

[31] X. Liang, T. Wang, L. Yang, and E. Xing. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. *arXiv preprint arXiv:1807.03776*, 2018. 1

[32] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, page 3, 2017. 4, 7

[33] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 1, 2

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, Cham, 2014. 4

[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 2

[36] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 2

[37] X. Ma, Y. He, X. Luo, J. Li, M. Zhao, B. An, and X. Guan. Vehicle traffic driven camera placement for better metropolis security surveillance. *IEEE Intelligent Systems*, 2018. 1

[38] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 2

[39] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with cuda. In *ACM SIGGRAPH 2008 classes*, page 16. ACM, 2008. 6

[40] A. Paszke, S. Gross, S. Chintala, and G. Chanan. Pytorch, 2017. 4

[41] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 1

[42] Y. Rao, D. Lin, J. Lu, and J. Zhou. Learning globally optimized object detector via policy gradient. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6190–6198, 2018. 2

[43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 2

[44] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 2, 4, 8

[45] C. Robert. Machine learning, a probabilistic perspective, 2014. 3

[46] A. Rosenfeld and M. Thurston. Edge and curve detection for visual scene analysis. *IEEE Transactions on computers*, (5):562–569, 1971. 2

[47] R. Rothe, M. Guillaumin, and L. Van Gool. Non-maximum suppression for object detection by passing messages between windows. In *Asian Conference on Computer Vision*, pages 290–306. Springer, 2014. 2

[48] S. Rujikietgumjorn and R. T. Collins. Optimized pedestrian detection for multiple and occluded people. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3690–3697, 2013. 8

[49] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*, 2018. 8

[50] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 4, 8

[51] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *null*, page 1470. IEEE, 2003. 1

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1

[53] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014. 1

[54] Y. Wang, L. Xie, C. Liu, S. Qiao, Y. Zhang, W. Zhang, Q. Tian, and A. L. Yuille. Sort: Second-order response transform for visual recognition. In *ICCV*, pages 1368–1377, 2017. 4

[55] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017. 1, 7

[56] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang. Unitbox: An advanced object detection network. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 516–520. ACM, 2016. 2

[57] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 1, 4, 8

[58] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 6

[59] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458, 2003. 1