Penalty Non-maximum Suppression in Object Detection

Wenqing Zhao(✉) and Hai Yan

North China Electric Power University, Baoding 071003, China
jbzwq@l26.com

Abstract. As a post-processing step, Non-Maximum Suppression (NMS) is always used to obtain final detection boxes. It suppresses all detection boxes which have a higher intersection-over-union (IoU) overlap than threshold T with pre-selected detection box p with the maximum score in each iteration. However, it removes the positive object, if the positive object is adjacent to p with a higher IoU. To overcome these shortages, we propose Penalty-NMS method which according to the different overlap to assign penalty coefficient to decay detections scores. In this process, we will not eliminate any detection boxes. And we keep all detection boxes temporarily until the detections with lower score will be eliminated after many rounds of iteration. Our method obtains significant improvements on standard datasets like PASCAL VOC (1.9% for Faster RCNN) and MS COCO (1.6% for R-FCN and 1.8% for Faster RCNN) without any additional computational and parameters.

Keywords: Non-Maximum suppression · Detection boxes · Penalty coefficient Intersection-over-union

1 Introduction

Object detection has always been a popular research topic in the field of computer vision. In the past decade, traditional methods [26–28] have reached the bottleneck and the accuracy is also not satisfying. However, owing to the image recognition which is based on Convolutional Neural Network, has fulfilled remarkable achievements, thus in the basis of the development of network, [3] has successfully made remarkable progress by introducing Convolutional Neural Network into object detection. Subsequently, many methods [1–3, 7, 32–34] based on the idea of [3] have significantly improved in the part of accuracy and speed. At the meantime, many current object detection pipelines due to the deep learning can be divided into three stages as follows: (1) extracts region proposals, (2) classifies and refines each region proposal, and (3) removes extra detection boxes that might belong to the same object. NMS is frequently used in Stage (3) as an essential part of object detection and obtains impressive effect in [1–3, 7, 8]. However, traditional NMS algorithm suppresses any detection box which exceeds the threshold T. That is, it removes detection boxes which have a IoU of pre-selected detection p exceed the given threshold T. Average precision would drop as a result of the missing positives, thus, traditional NMS could also be named as GreedyNMS. The current state-of-the-art object detection Faster RCNN

J.-H. Lai et al. (Eds.): PRCV 2018, LNCS 11259, pp. 90–102, 2018.
https://doi.org/10.1007/978-3-030-03341-5_8

based on ImageNet model [10–12] extracts feature maps and generates many region proposals by Region Proposal Network(RPN) [2]. Then it uses GreedyNMS to get final detections.

For any IoU threshold, GreedyNMS always tradeoffs between recall and precision. This paper is devoted to solving the problem which leads to a miss due to the setting of the threshold T. As a post-processing step, GreedyNMS aims to remove redundant detections. When we solve its defects, we should not add any additional computation or parameters. [9] proposed Tyrolean network (Tnet) to rescore all detections, then the network can choose the best detection. Although this method improves detection performance, it also adds extra training.

Static images which are addressed by stage (1) and (2) will generate multiple detection boxes (denoted as set P), each detection box attaches a corresponding detection score (denoted as set C). GreedyNMS selects the detection box p with the highest score in the set P, removes p from set P and appends it to set K (final detection boxes set). Then it calculates the IoU of p with the rest detection boxes, suppressing all detections in which the IoU is higher than threshold T. At last, repetitively executing the remaining detection boxes with the same procedures, then T plays an essential role in object detection. When we selected a lower threshold T, the object which has a IoU higher than T would be missed. Instead, when we select a larger threshold T, it would generate more false positives which will lead to a drop in average precision. Hence, GreedyNMS always compromises between recall and precision. For this purpose, we propose Penalty Non-maximum suppression (Penalty-NMS) which penalizes all detection boxes except detection box which has the highest score in each iteration. Owing to the different overlap values, assigning penalty coefficient to reduce the detection boxes score without using threshold T. In comparison with the box in which the GreedyNMS suppression is greater than the threshold T, we'd like to use the quadratic function to calculate the penalty coefficient to reduce detection score which has different overlap values. Penalty-NMS obtains significant improvements in average precision for Faster RCNN and R-FCN on standard datasets like PASCAL VOC and MS COCO.

2 Related Work

Although the NMS algorithm is a core part of object detection, Numerous studies have focused on feature design, classifier design, and object proposals in the past. Surprisingly, Few studies on the NMS algorithms exist.

NMS was first employed in edge detection which is performed edge thinning to remove spurious responses [35]. Subsequently, it has been applied to face detection [29] and object detection [16]. [16] demonstrated that a greedy NMS algorithm, where a detection box with the maximum score is selected and its neighboring boxes are suppressed using a threshold improves performance over the approach used for face detection [29]. Since then, greedy NMS has been the de-facto algorithm used in object detection [1–3, 7, 24, 32].

For clustering detections, principled clustering formulation has been proposed in [15, 16] which obtain good performance for object class detection. Several other

clustering algorithms have been explored for the task of NMS: mean-shift clustering [16], agglomerative clustering [17] and affinity propagation clustering [18]. However, they have yet to surpass the performance of GreedyNMS. To link convnets and NMS [26], directly generates a sparse set of detections by training an LSTM that NMS is unnecessary. [21] designs a convnet that combines decisions of GreedyNMS with different overlap thresholds, allowing the network to choose the GreedyNMS operating point locally. [20] proposed a proposal subset optimization algorithm for detecting salient objects as an alternative to NMS. [22] has proposed a true end-to-end learning algorithm which makes the classifier aware of the NMS procedure at test time by including GreedyNMS at training time. [23] operates on graphs, but requires a pre-processing that defines a node ordering. [4, 30] tend to produce fewer spread-out double detections and improve overall detection quality. [19, 24] propose to detect pairs of objects instead of each individual objects in order to handle strong occlusion. But there is a problem that single and double detections need to be handled.

In summary, most of the proposed algorithms can replace GreedyNMS. However, we find that GreedyNMS still obtain the greatest performance for generic object detection. [21] has obtain better results which is capable of performing NMS without being given a set of suppression alternatives to choose from and without having another final suppression step. But an additional deep network with vast parameters and Computation is required. [25] proposed Soft-NMS that Our algorithm is similar to it but does not add any parameters.

Although the traditional NMS algorithm can obtain better performance in several generations of detector [1–3, 13, 14], it is still a greedy algorithm with obvious defects. This paper aims to improve the NMS algorithm without any additional parameters or computation.

3 Penalty-NMS

This section is divided into two parts. First of all, we review the details of the traditional NMS (i.e. GreedyNMS) and analyze the shortcomings of GreedyNMS as a post-processing in object detection. Then we introduce Penalty-NMS which is proposed in this paper in detail.

3.1 GreedyNMS

When a test image passes the detection system without post-processing, multiple detections are generated around each object. However, each object only needs one detection box. Therefore, GreedyNMS is used as a post-processing to eliminate redundant detection boxes. GreedyNMS process is as follows:

1. Detection boxes set $p\{p_1, p_2 \cdots p_n\}$ and the corresponding scores set $C\{c_1, c_2 \ldots c_n\}$.
2. Choose the detection box p_{max} which has a maximum score and merge it into set K (the final detection boxes set). Then remove p_{max} from set P and calculate the IOU overlap of the remaining detections with p_{max}.

3. Set the threshold T. Remove all detections which have a IoU overlap that is greater than T from set P.
4. Repeat steps 2 and 3 until set P is empty.

Threshold T in step 3 has a decisive influence on the final result of the entire detection system. However, it's hard for us to find a suitable value.

3.2 Penalty-NMS

The neighbor detection boxes have a higher likelihood of detecting the same object. However, GreedyNMS always falls into local optimal, since it removes all the detection boxes which have a IoU overlap that is greater than threshold T. Unlike GreedyNMS, we penalize all detection boxes that should be suppressed in GreedyNMS in **Piecewise Penalty-NMS** or penalize all detection boxes regardless of whether they have an overlap with detection box p in **Continuous Penalty-NMS**. The detection boxes with lower scores are removed after multiple penalties.

(Piecewise Penalty-NMS). As it is shown in Eq. (1), Piecewise Penalty-NMS penalizes any detection which has a higher IoU than T and the detection boxes with a IoU less than T keep its original score. The penalty coefficient is $\beta\left(1-\text{overlap}^2\right)$.

$$\mu_i = \begin{cases} 1, & overlap < T \\ \beta\left(1-\text{overlap}^2\right), & overlap \geq T \end{cases} \tag{1}$$

Here μ_i is the penalty coefficient for detection box i, overlap ($0 \leq \text{overlap} \leq 1$) is the IoU overlap of the detection box p_i with the pre-selected detection box, β ($\beta > 0$) is the regulatory factor.

In Eq. (1) we change the score of the detection boxes which have a IoU overlap is greater than threshold T in GreedyNMS from 0 to $\beta\left(1-\text{overlap}^2\right) \cdot$ confidence(detection boxes' score). Therefore, we can regard GreedyNMS as a special form of Piecewise Penalty-NMS. As we have analyzed in Sect. 3.1, there are always some positive objects which are missed in GreedyNMS. So penalizing detections to decay the scores seems to be a better approach. Eq. (1) decreases the scores of the detection boxes whose IoU overlap is greater than threshold T. However, the Piecewise Penalty-NMS still needs to set threshold T manually. The performance of the algorithm is still limited by the threshold T.

(Continuous Penalty-NMS). Continuous Penalty-NMS algorithm is shown in Fig. 3. The penalty coefficient is shown in Eqs. (2) and (3). In Continuous Penalty-NMS, we no longer use threshold T, but directly penalize all detection boxes. In Eqs. (2) and (3), the growth of penalty has the opposite change. In the following sections, Continuous Penalty-NMS_1 and Continuous Penalty-NMS_2 correspond to Eqs. (2) and (3) respectively.

$$\mu_i = \beta\left(1 - overlap^2\right) \tag{2}$$

$$\mu_i = \beta(overlap - 1)^2 \tag{3}$$

In Sect. 3.1, we have analyzed the performance of the traditional NMS which is strictly limited by the threshold T. Piecewise Penalty-NMS is a better way to improve the performance of GreedyNMS algorithm, but it's still unable to get rid of the threshold T. Therefore, we further propose Continuous Penalty-NMS which is different from [25] that does not penalize detection boxes with no overlap. Instead, Continuous Penalty-NMS penalizes all detections except detection box with maximum score in each iteration.

Quadratic functions are used as the penalty coefficients in this paper. The first derivative of Eqs. (1) and (2) is $-2\beta \cdot \text{overlap}$ and the first derivative of Eq. (3) is $2\beta(\text{overlap} - 1)$. All of them are less than 0. Therefore, the penalty coefficient becomes smaller as the value of IoU becomes higher to decay the score of detection box. The second derivative of Eqs. (1) and (2) are both less than 0 that are convex function. Equation (3) is opposite to it but is the same as Gaussian function which is presented in [25]. The growth proportion of Penalty in Eq. (2) increases as the IoU overlap increases. However, the growth proportion of Penalty in Eq. (3) decreases as the IoU overlap increases.

Penalty-NMS algorithm is used in each iteration and removes detections whose scores are less than σ. Compared with the influence of the threshold T in GreedyNMS, the parameter β and σ in Penalty-NMS is less sensitive to the performance of the algorithm. We can also set β to 1.0 which means that our algorithm does not add any parameters. Besides, the computational complexity for Penalty-NMS is $O(n^2)$ which is the same as GreedyNMS, the n is the number of detection boxes. Although Penalty-NMS algorithm does not get the global optimal solution, compared with GreedyNMS algorithm, a better sub-optimal global solution is obtained without any additional computation and can be easily embedded in any detection system (Fig. 1).

4 Experiment

4.1 Experiment on PASCAL VOC

Our experiment is based on Faster RCNN on PASCAL VOC [4] that has 20 object categories and the basic network is VGG16 [11]. We train the models on the union set of VOC 2007 *trainval* and evaluate on VOC 2007 *test* set. Object detection accuracy is measured by mean Average Precision (mAP). In this section, we analyze and compare the performance of Piecewise Penalty-NMS, Continuous Penalty-NMS_1, Continuous Penalty-NMS_2 and traditional NMS algorithm. We also analyze the sensitivity of parameters β and σ.

(Penalty-NMS Performance Analysis). In Table 1, we analyze the changes of mAP values at different threshold T by GreedyNMS and Piecewise Penalty-NMS. We also compare the performance of Piecewise Penalty-NMS at different values of β.

Penalty-NMS: $\mathcal{P}$ is the set of detections
K is the set of the final detections
S is the set of the scores
μ_i is the penalty coefficient of detection box i
c_i is the score of detection box i, σ is the lowest score.

while $\mathcal{P} \neq \emptyset$:
$p_m = maximum(\mathcal{P})$
K = K ∪ p_m
$\mathcal{P} = \mathcal{P} - p_m$
for p_i in $\mathcal{P}$:
$c_i = \mu_i c_i$ ---------- Continuous Penalty-NMS
if overlap > T:
$c_i = \mu_i c_i$ ---------- Piecewise Penalty-NMS
else:
$c_i = c_i$
if $c_i < \sigma$
$\mathcal{P} = \mathcal{P} - p_i$
return K, S

Fig. 1. The pseudo code for Penalty-NMS algorithm

Table 1 shows that when the threshold T is 0.3 or 0.4, compared with GreedyNMS, Piecewise Penalty-NMS has a significant improvement up to 1.9%(higher than [25]) in mAP. Although Piecewise Penalty-NMS adds a parameter β, analyzing the red box in Table 1, we can see that the different values of β has no significant distinction in mAP. The fluctuation range is only 0–0.7%. The blue box in Table 1 shows that Piecewise Penalty-NMS does not obtain any performance improvement when the threshold T $\geq$ 0.5. We analyze the main reason for this situation is that the positive detections which should be suppressed is directly kept. The effect is similar to our algorithm when we set higher threshold. However, this growth of positive is much less than the growth of false positive, which leads a drop in average precision.

Table 1. Results on PASCAL VOC 2007 test set, PPenalty-NMS denotes Piecewise Penalty-NMS. 0.5, 0.6, 0.7, 0.8 denotes the value of β

NMS	T				
	0.3	0.4	0.5	0.6	0.7
GreedyNMS	69.6	**70.0**	69.2	64.6	56.4
PPenalty-NMS$_{0.5}$	71.5	71.1	69.0	64.7	56.5
PPenalty-NMS$_{0.6}$	**71.9**	71.2	68.8	64.1	56.6
PPenalty-NMS$_{0.7}$	71.5	71.4	68.6	64.8	56.3
PPenalty-NMS$_{0.8}$	71.2	71.4	68.9	64.7	56.5

Table 2 shows that Continuous Penalty-NMS$_1$ and Continuous Penalty-NMS$_2$ algorithm still obtain significant improvement without using threshold T. The highest growth reached 1.9% and 1.8% respectively which are Equal to Piecewise Penalty-NMS. Empirically, the detection boxes with higher overlap should be penalized more. The growth proportion of Penalty in Eq. (2) increases as the IoU increases. However, the growth proportion of Penalty in Eq. (3) decreases as the IoU increases. As parameter β changes, although Penalty-NMS$_1$ and Continuous Penalty-NMS$_2$ are two different modes of penal growth, both of them obtain similar improvement. Therefore, the growth mode of penalty is not the key factor which can affect the performance of the algorithm. In Table 2, row 4 and 5 show that the value of mAP decreases continuously as the threshold T increases. Compared with Continuous Penalty-NMS1 and Continuous Penalty-NMS2 whose ranges of variation are from 0.3% to 0.8%, the performance of the traditional NMS is totally depended on the setting of threshold T.

Table 2. Results on PASCAL VOC 2007 test set, CPenalty-NMS$_1$ denotes Continuous Penalty-NMS$_1$, CPenalty-NMS$_2$ denotes Continuous Penalty-NMS$_2$.

NMS	β							
	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
CPenalty-NMS$_1$	71.6	71.5	71.3	**71.9**	71.4	71.2	71.4	71.1
CPenalty-NMS$_2$	71.4	71.1	71.1	71.5	71.5	71.4	71.1	**71.8**
T	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
GreedyNMS	69.6	**70.0**	69.2	64.6	56.4	42.9	–	–

(Sensitivity Analysis). The function of the parameter σ is the same as the threshold T in GreedyNMS. But its effect on the performance of the algorithm is far less than T. As shown in Fig. 4, Piecewise Penalty-NMS$_1$ and Continuous Penalty-NMS$_2$ obtain better performance from a range between 0.001 to 0.004. The mAP for Piecewise Penalty-NMS $(\beta = 0.6, \mathrm{T} = 0.3)$ are maintained at 71.4%–71.9%. Likewise, Continuous Penalty-NMS$_1$ $(\beta = 0.6)$ also stays at 71.6%–71.9%. The fluctuation range is kept within 0.5% and less than the GreedyNMS (Fig. 2).

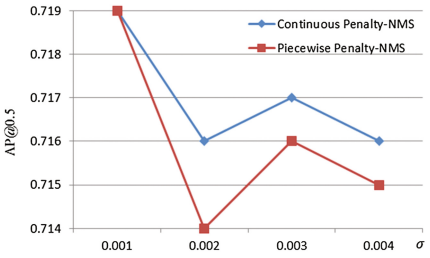

Fig. 2. Sensitivity Analysis to parameter σ

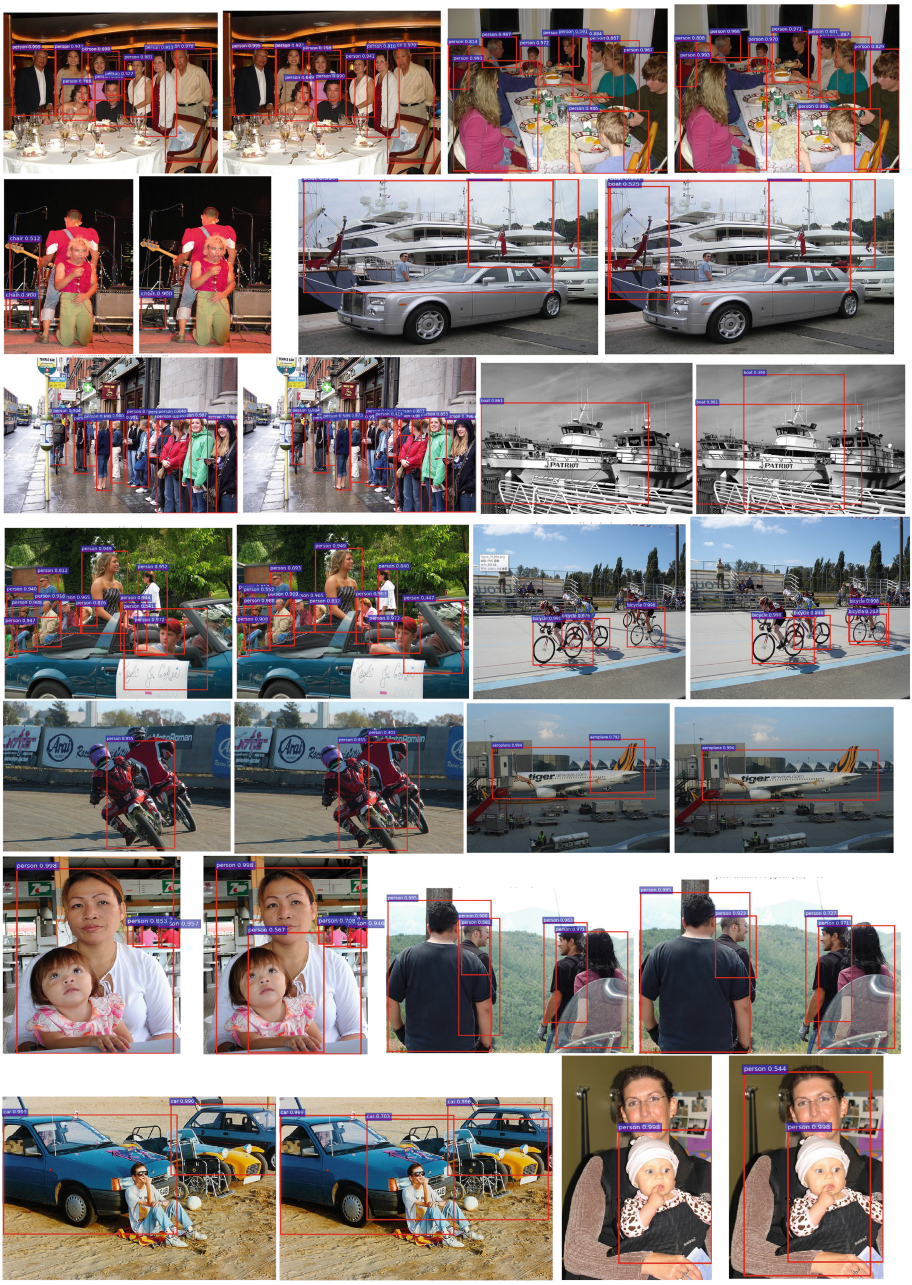

Fig. 3. Selected examples of object detection results on the PASCAL VOC 2007 test set using the Faster R-CNN system. The model is VGG-16 and the training data is 07 trainval. Left for NMS algorithm and right for Continuous Penalty-NMS1 algorithm.

Column 9 of Table 2 shows that Continuous Penalty-NMS_1 and Continuous Penalty-NMS_2 reach 71.1% and 71.8% respectively which also obtain better performance than GreedyNMS. So we have a further experiment that sets β to 1.0, which means that we also no longer use parameter β. In Table 3, we still obtain a better improvement by 1.0%–1.8%, when the range is from 0.001 to 0.004. Although it has brought some loss in average precision, compared with the traditional NMS algorithm, Penalty-NMS not only has significant improvements, but also is freer to select parameter σ, which has a slight influence on accuracy. Table 4 show the detailed numbers.

Table 3. Sensitivity Analysis when β is equal to 1.0. Continuous Penalty-NMS_1 denotes CPenalty-NMS_1 and Continuous Penalty-NMS_2 denotes CPenalty-NMS_2.

NMS	σ			
	0.001	0.002	0.003	0.004
CPenalty-NMS_1	71.1	71.2	**71.0**	71.1
CPenalty-NMS_2	**71.8**	71.6	71.4	71.3

4.2 Experiments on MS COCO

We evaluate Faster RCNN and R-FCN on the MS COCO dataset [5] that has 80 object categories. Our experiments involve the 80 k *train* set and 40 k *val* set for test. We evaluate the mAP average for IoU ∈ [0.5:0.05:0.95] (COCO's standard metric, simply denoted as mAP@[0.5, 0.95]) and mAP@0.5 (PASCAL VOC's metric).

The results are in Table 5. R-FCN [24] with single-scale trained baseline has a *val* result of 48.9%/27.6% and Faster RCNN has a baseline 48.4%/27.2%. R-FCN obtained 1.1% and 1.0% improvement respectively based on MS COCO's standard metric and PASCAL VOC's metric when we adopt Piecewise Penalty-NMS algorithm. We also obtain improvements of 1.4% and 1.3% for Faster RCNN. For MS COCO's standard metric we obtain an improvement of 1.6% and 1.8% respectively for R-FCN and Faster RCNN which are higher than Soft-NMS [25] 's improvements 1.1% for Faster RCNN and 1.3% for R-FCN.

Table 4. Results on PASCAL VOC 2007 test set with Faster R-CNN detectors and VGG-16

method	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
GreedyNMS	70.0	68.8	78.7	69.3	51.5	54.2	81.3	80.1	83.9	50.0	77.1	64.7	83.7	83.4	76.1	77.0	38.4	68.7	64.9	75.1	73.4
Soft-NMS [25] -L	71.2	71.4	81.4	70.5	54.6	52.9	77.6	82.1	83.4	51.8	78.9	66.7	81.2	82.1	78.9	79.7	39.5	71.7	64.4	79.2	75.2
Soft-NMS [25] -G	71.2	71.3	80.7	70.9	54.9	53.0	78.2	81.9	83.8	51.9	78.6	67.1	80.1	81.9	79.4	80.0	39.7	71.8	64.6	78.8	75.6
PPenalty-NMS	71.9	71.8	80.9	73.3	60.0	53.9	79.0	82.5	83.8	51.7	79.9	68.5	79.7	83.7	75.8	79.9	42.5	72.3	66.3	77.9	74.8
CPenalty-NMS_1	71.9	72.3	83.0	72.4	58.6	54.5	78.7	82.8	82.9	51.1	80.0	69.1	81.0	81.7	76.8	79.5	43.6	68.4	68.5	80.1	72.0
CPenalty-NMS_2	71.8	71.4	81.7	70.5	58.5	54.8	79.9	82.1	85.5	51.9	80.3	66.1	81.7	82.6	75.7	79.6	42.7	72.2	67.4	79.4	72.1

Table 5. Results on MS COCO dataset with ResNet-101. G denotes GreedyNMS, P denotes Piecewise Penalty-NMS and C denotes Continuous Penalty-NMS_1.

	Training data	Test data	AP@0.5	mAP@[0.5,0.95]
Faster RCNN-G	Train	Val	48.4	27.2
Faster RCNN-P	Train	Val	49.7	28.6
Faster RCNN-C	Train	Val	49.9	29.0
R-FCN-G	Train	Val	48.9	27.6
R-FCN-P	Train	Val	49.9	28.7
R-FCN-C	Train	Val	50.1	29.2

5 Conclusion

As the greedy algorithm, NMS usually serves as post-processing. In this paper, we analyze in detail the limitations of GreedyNMS, which bring about a miss due to the setting of threshold T. We propose Penalty-NMS algorithm which penalizes detection boxes rather than the direct suppress one. What's more, our method solves the limitations without using any additional computations or parameters. The experimental results indicate that compared to the traditional NMS algorithm, Penalty-NMS achieves salient progress. Notes should be observed that we do not obtain the overall optimal result and Penalty-NMS is still a greedy algorithm. But the penalty idea gets a good effect on applying NMS algorithm. Moreover, other functions are proper to be penalty coefficient. The future work will focus on the way of learning to get penalty coefficient instead of adopting the fixed function.

References

1. Girshick, R.: Fast R-CNN. In: ICCV (2015)
2. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS (2015)
3. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR (2014)
4. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results (2007)
5. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
6. Uijlings, J., van de Sande, K., Gevers, T., Smeulders, A.: Selective search for object recognition. IJCV **104**, 154–171 (2013)
7. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. ArXiv preprint arXiv:1506.02640 (2015)
8. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8691, pp. 346–361. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10578-9_23

9. Hosang, J., Benenson, R., Schiele, B.: A convent for non-maximum suppression. In: GCPR (2016)
10. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
11. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (ICLR) (2015)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv: 1512.03385 (2015)
13. Viola, P., Jones, M.: Robust real-time face detection. IJCV **57**, 137–154 (2004)
14. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. In: PAMI (2010)
15. Rothe, R., Guillaumin, M., Van Gool, L.: Non-maximum suppression for object detection by passing messages between windows. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (eds.) ACCV 2014. LNCS, vol. 9003, pp. 290–306. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16865-4_19
16. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
17. Bourdev, L., Maji, S., Brox, T., Malik, J.: Detecting people using mutually consistent poselet activations. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6316, pp. 168–181. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15567-3_13
18. Mrowca, D., Rohrbach, M., Hoffman, J., Hu, R., Saenko, K., Darrell, T.: Spatial semantic regularisation for large scale object detection. In: ICCV (2015)
19. Stewart. R., Andriluka, M.: End-to-end people detection in crowded scenes. In: CVPR (2016)
20. Zhang, J., Sclaroff, S., Lin, Z., Shen, X., Price, B., Mech, R.: Unconstrained salient object detection via proposal subset optimization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5733–5742 (2016)
21. Hosang, J., Benenson, R., Schiele, B.: Learning non-maximum suppression (2017)
22. Henderson, P., Ferrari, V.: End-to-end training of object class detectors for mean average precision. In: Lai, S.-H., Lepetit, V., Nishino, K., Sato, Y. (eds.) ACCV 2016. LNCS, vol. 10115, pp. 198–213. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54193-8_13
23. Blaschko, M.B.: Branch and bound strategies for non-maximal suppression in object detection. In: Boykov, Y., Kahl, F., Lempitsky, V., Schmidt, Frank R. (eds.) EMMCVPR 2011. LNCS, vol. 6819, pp. 385–398. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23094-3_28
24. Dai, J., Li, Y., He, K., et al.: R-FCN: object detection via region-based fully convolutional networks (2016)
25. Bodla, N., Singh, B., Chellappa, R., et al.: Improving object detection with one line of code (2017)
26. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. In: TPAMI (2010)
27. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, vol. 1, pp. I. IEEE (2001)
28. Felzenszwalb, P., Mcallester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008. IEEE (2008)

29. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: International Conference on Neural Information Processing Systems, pp. 1097–1105. Curran Associates Inc (2012)
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. Comput. Sci. (2014)
31. Szegedy, C., Liu, W., Jia, Y., et al.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition. IEEE (2015)
32. Redmonand, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: CVPR (2017)
33. Liu, W., et al.: SSD: Single Shot MultiBox Detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
34. Shen, Z., Liu, Z., Li, J., et al.: DSOD: Learning deeply supervised object detectors from scratch (2017)
35. Rosenfeld, A., Thurston, M.: Edge and curve detection for visual scene analysis. IEEE Trans. Comput. **100**(5), 562–569 (1971)