Improving Faster R-CNN Based Object Detection By Utilizing Soft-NMS Algorithm In Overlapping Boxes
Final Report - Group 28

Xinyue Wang
4555589
xinyue.wang.77@gmail.com

Ang Li
4593820
shawn19931117@gmail.com

Abstract

Faster R-CNN has tremendous applications in object detection field, such as autonomous driving, video/image indexing, and surveillance. In this paper, we introduces a Soft Non-maximum suppression algorithm integrated in the Faster R-CNN pipeline to further improve the precision of Faster R-CNN without adding more hyper-parameters. Significant improvements have been found in qualitative results for some special cases and mean precision of the detector improves by approximately 1%. Sensitivity analysis demonstrates that choosing appropriate values of the hyperparameters and utilizing exponential re-scoring functions in the Soft-NMS algorithm yields a possibility to integrate Soft-NMS in detector without sacrificing any computational efficiency. It can be expected that this algorithm is applicable in any kind of detector due to its model-independent nature.

1. Introduction

As a representative object detection method with relatively high precision, Faster R-CNN has tremendous applications in autonomous driving, video/image indexing, surveillance, etc.

However, the inferior Input image resolutions and lower feature extractors impact speed contributes to the fact that, any improvement for Faster R-CNN should not create a computational bottleneck or add more complexity.

Therefore, to further improve the performance of Faster R-CNN pipeline, we implemented a soft non-maximum suppression(soft-NMS) algorithm in the post-processing stage under Tensorflow framework without the requirement of re-training the model.

1.1. Problem

Below is our research question:

How to improve the precision of Faster R CNN without introducing additional hyper parameters?

2. Related Work

In the field of object detection, the ultimate objective is to search for the candidate positions/boxes that contain object of interest with high precision and computational efficiency. Since the rapid growth and flourish of Deep learning in 2012, researchers shifted their focus from the "Extract hand crafted features" to "learn features", which yielded a significant improvement in the object detection techniques.

Of the state-of-the art Deep learning approaches in object detection, R-CNN(Regions with Convolutional Neural Network Features) families have played an important role since latest research. R-CNN can be described as a system that combines bottom-up region proposals with rich features computed by a convolutional neural network[4]. It computes the region of interest (RoI) from the input, followed by a warped image region, for each of RoI, and it will be passed forward the Convolutional network. Once each region has been forwarded to the network, bounding box regressors are applied and SVM is then used for classification. However, the training stage of RCNN remains limited in the selection of region proposals,softmax classifier(log loss),training SVM and the regressor(squared loss).

Compared to previous work, Fast R-CNN employs a new training algorithm that fixes the disadvantages to improve training and testing speed while also increasing detection accuracy[3]. Instead of computing a lot of proposed regions for the image invariably overlapped, Fast R-CNN runs the CNN just once per image and then share the computation across the all the proposals.In this approach, the remaining bottleneck in the Fast R-CNN processŁisŁthe region proposer which utilizes an inefficient selective search algorithm. Later in this field.

Faster R-CNN fixed this limitation in the first step of classification, and it determines region proposals depending on extracted features maps that were already calculated

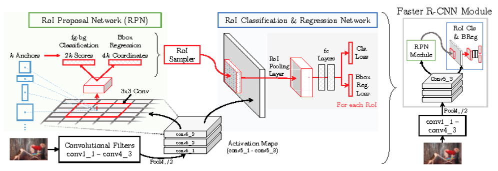


Figure 1. Framework of Faster R-CNN[3]

with the forward pass of the CNN[10]. It is achieving state-of-the-art object detection accuracy on PASCAL VOC 2007 (73.2 % mAP) and 2012 (70.4% mAP) using 300 proposals per image.

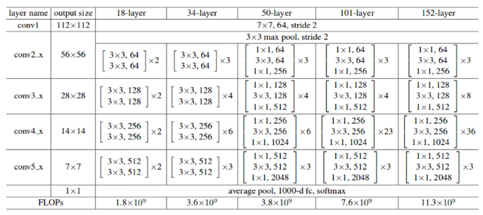
In the latest research, more advanced object detectors, including R-FCN[2], SSD[8], FPN[6], RetinaNet[7] and YOLOv3[9] have been compared with Faster-R-CNN in terms of speed and accuracy Even though the detection speed is outperformed by latest networks, the mean accuracy over different datasets still still remain the best. Therefore, it is significant to continue working on improvements on Faster R-CNN pipelines. To this end, reference[1] proposes a soft-NMS algorithm to improve the basic Non-maximum suppression(NMS) method integrated Faster R-CNN pipelines. This algorithm retains the bounding boxes with highest score and scales all other detection boxes with a significant overlap by using a soft decaying function instead of a hard way which re-scores all neighboring boxes above the threshold as zero. Generally soft-nms method can improve the detection accuracy in cases when objects classified to the same class are stacking in the image. Nevertheless, in some cases Soft-NMS generates failed samples where the bounding boxes that should be eradicated are retained in the image, or even wrong boxes appear. Therefore, the soft-nms algorithm integrated in Faster R-CNN still worths more efforts in further investigations.

3. Faster R-CNN frameworks

The visualization of Faster Region-based convolution neural network(Faster RCNN) frameworks can be found in 1, which combines feature-extraction, proposal-extraction, bounding box regression and classification. Pre-trained models on ImageNet are usually served as the feature maps extractor. Four different losses are simultaneously minimized during the training stage, including classification and regression losses in the RPN and classification and regression losses in the RCNN classifier, respectively.

3.1. Convolution layer architectures

Two different architectures: Vgg16(16 layers Very Deep Convolutional Networks for Large-Scale Image Recognition[12]), and Res101(101 layers Deep residual networks[5]) are utilized in this project.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2. Residual network family[5]

3.1.1 Vgg16

In VGG16, 13 convolution layer, 13 RELU layer, and 4 pooling layer are constructed. Every convolution layer has kernel size=3 and padding=1, and every pooling layer has kernel size=2, stride=2.Therefore, the convolution and RELU layer will not change the size of their input and output, while the output of a pooling layer will be reduced by 50% in width and height compared with inputs. After 4 pooling layer, the original image can be mapped by multiplying the extracted feature maps with 16.

3.1.2 Res101

The deep residual networks are more complicated in that it explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions.The residual network family can be found in Figure2

3.2. Region Proposal Network

Anchors are the most important concepts in a RPN network,as shown in Figure3. An anchor is a vector consisted of [x1,y1,x2,y2], which represents 4 coordinates of the corners of a bounding box. To fully cover the region of an image, these anchors have 3 prescribed sizes and aspect ratios, respectively, leading to k=9 rectangles in total. The RPN randomly samples 256 anchors in an image to compute the loss function of a mini-batch, where the sampled positive and negative anchors have a ratio of up to 1:1. For each such anchor box, we output one bounding box and score per position in the image. We then pass each such bounding box that is likely to be an object into Fast R-CNN to generate a classification and tightened bounding boxes.

Faster R-CNN adds a Fully Convolutional Network on top of the features of the CNN creating whats known as the Region Proposal Network. It classifies the anchors to acquire foreground and background by softmax,and those foreground objects will be passed to next step. It also calculates the transform parameters which map anchors generated by ourselves to Ground Truth box. Simultaneously, the anchors are refined in a regressor after we feed the logit into a softmax/logistic regression activation function, then

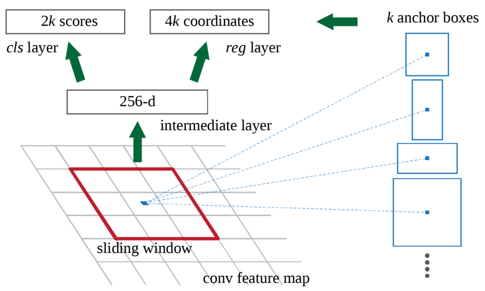


Figure 3. Anchors in a RPN[3]

it will predict the labels. In this way the training data is complete with features and labels. But it should be noted that the class labels are not yet able to be predicted, which will be performed in the final stage.

3.3. ROI Pooling and final classification

Region of Interest(RoI) Pooling is used to compute proposals feature map which are supposed to have fixed length because RoI pooling uses max pooling to convert the feature of any RoI into a feature map with fixed spatial extent of H*W (7*7 in our case). An RoI is defined by a four tuple(r,c,h,w) that specifies its top-left corner(r,c) and its height and width (h,w). Two inputs are fed in the ROI Pooling, firstly a fixed-size feature map obtained from a deep convolutional network with several convolutions and max pooling layers, secondly a N x 5 matrix of representing a list of regions of interest, where N is a number of RoIs. The first column represents the image index and the remaining four are the coordinates of the top left and bottom right corners of the region.

In the following stage, three different steps are performed in the ROI Pooling, including dividing the region proposal into equal-sized sections, finding the largest value in each section, and copying these max values to the output buffer.The result is that from a list of rectangles with different sizes we can quickly get a list of corresponding feature maps with a fixed size. This approach can save us tremendous time since the convolutional calculations in the early stages are time consuming.

3.4. Training

The training of Faster R-CNN will be continued based upon the pre-trained models on ImageNet. These models have been fine-tuned so no additional tuning is needed for the early stage feature extractions. The remaining training can be divided in 6 steps as follows:

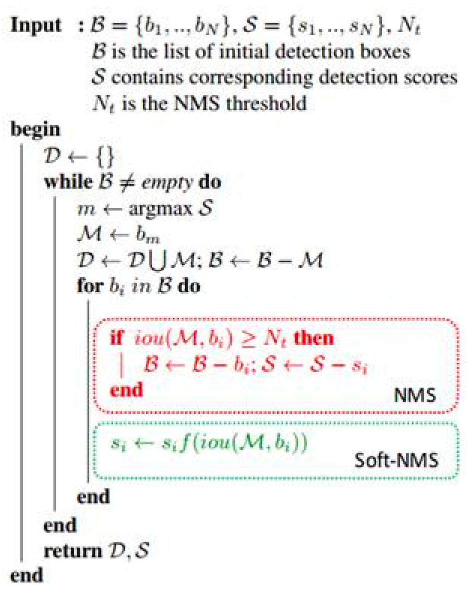
1. Training the RPN based on per-trained models
2. Generating region proposals from RPN
3. First time training the remaining Fast RCNN network
4. Second time training with RPN
5. Generating region proposals from RPN
6. Second time training the remaining Fast RCNN network

The total training loss is the sum of two RPN losses and two Fast RCNN losses[3]. The loss is then optimized by utilizing momentum gradient descent method due to its high efficiency in training Deep Neural Networks[11].

3.5. Soft-NMS algorithm

One of the problem of Faster R-CNN classification network as we have learned so far is that the network may find multiple detections at the same point.Non-maximum-suppression algorithm is an efficient way to assure that the network detects each point only once. More specifically, since there is no constraint in the network which forces it to generate a unique RoI for an object, multiple proposals may correspond to the same object. To alleviate this problem, non-maximum-suppression is performed on detection boxes of each class independently, with a specified overlap threshold. Applying NMS with a low threshold could lead to a drop in average precision when the overlap criterion during evaluation for a true positive is higher, because there could be a detection box which is very close to an object within the threshold, but had a slightly lower score, thus this box with lower score gets suppressed by a low threshold in NMS the algorithm. Furthermore, The increase in false positives would be much higher than the increase in true positives for this case because the number of objects is typically 1We do not replace this non-maximum suppression in the object detection pipeline much smaller than the number of RoIs generated by a detector. Apparently, the misclassification rate would increase if such cases ever happen.

The pseudo code of soft-NMS can be found in Figure 4. To prevent the occurrence of the aforementioned special cases, the re-scoring function in the NMS algorithm has been softened. This is, instead of directly suppressing the neighboring boxes with lower score, Soft-NMS will recursively re-score them depending on current score levels. When integrated in the Faster R-CNN, this algorithm will not introduce any hyperparameters in the training stage as Soft-NMS is more like the post-processing techniques of the bounding boxes in the images. Therefore, the hyperparameters to fine-tune the Soft-NMS algorithm will only appear in the testing or demonstration stages. This nature can improve the precision of the detector, without affecting the trained model. In this way, the tunning of the training model and Soft-NMS algorithm can be accomplished individually, which significantly reduces the efforts we need to improve the precision.



```
Input  : $\mathcal{B} = \{b_1, .., b_N\}$, $\mathcal{S} = \{s_1, .., s_N\}$, $N_t$
         $\mathcal{B}$ is the list of initial detection boxes
         $\mathcal{S}$ contains corresponding detection scores
         $N_t$ is the NMS threshold
begin
    $\mathcal{D} \leftarrow \{\}$
    while $\mathcal{B} \neq empty$ do
        $m \leftarrow$ argmax $\mathcal{S}$
        $\mathcal{M} \leftarrow b_m$
        $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}$; $\mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
        for $b_i$ in $\mathcal{B}$ do
            if $iou(\mathcal{M}, b_i) \geq N_t$ then
                $\mathcal{B} \leftarrow \mathcal{B} - b_i$; $\mathcal{S} \leftarrow \mathcal{S} - s_i$
            end                                          NMS

            $s_i \leftarrow s_i f(iou(\mathcal{M}, b_i))$
                                                         Soft-NMS
        end
    end
    return $\mathcal{D}, \mathcal{S}$
end
```

Figure 4. The pseudo code in red is replaced with the one in green in Soft-NMS.

The hyperparameters in the Soft-NMS contain the same threshold value as the NMS, as well as the re-scoring functions as shown in the pseudo code. Four different re-scoring functions in the Soft-NMS(1) are investigated, including linear function $S_i = S_i\left(1 - iou\left(M, b_i\right)\right)$, Gaussian function $S_i = S_i e^{-\frac{iou(M,b_i)^2}{\sigma}}$,polynomial function $S_i = S_i\left(1 - iou\left(M, b_i\right)^3\right)$ and exponential function $S_i = S_i e^{-\frac{iou(M,b_i)^3}{\sigma}}$.The plots of these functions are demonstrated in (0,1) as shown in Figure5.The effect of these functions and hyperparameters on the final mean precision will be investigated in this project.

$$S_i = \begin{cases} S_i, iou\left(M, b_i < N_t\right) \\ S_i\left(1 - iou\left(M, b_i\right)\right), iou\left(M, b_i \geq N_t\right) \end{cases} \tag{1}$$

4. Experiment Setup

In order to find out the actual performance of soft-NMS, we setup two experiments using two different datasets. Following introduce the datasets we use and the experiments we design.

4.1. Dataset

4.1.1 Training set

The dataset used for training the model is PASCAL VOC 2007+2012, which provides standardized image data sets for object class recognition.

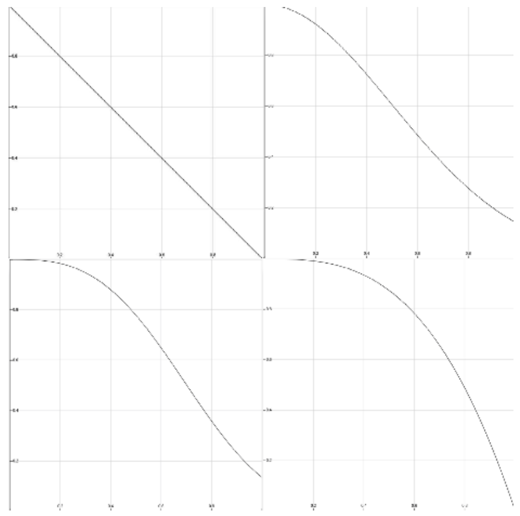


Figure 5. Re-scoring functions:Linear(upper left),Gaussian(upper right),Exponential(lower left) and Polynomial(lower right)

4.1.2 Test set 1

The first dataset includes 5 selected images from PASCAL VOC 2007. These 5 images have a similar feature that the objects belong to the same category some extent of overlapping. The reason for choosing this kind of images is that the traditional NMS couldn't deal with this situation and normally could only detect some or even one of these overlapping objects. The selected images can be find in **Figure 6** and **Figure 7**.

4.1.3 Test set 2

The second dataset utilized is collected from the existing large-scale datasets, we choose PASCAL VOC 2007 as our test set, which contains 30% labeled training samples and 70 % test samples.

4.2. Experiments

In the following subsections, the hyperparameters for training and testing configurations will be specified. After fine-tuning these parameters to achieve the optimal performance, two main experiments are designed for comparing the performance of Faster R-CNN with NMS algorithm and with Soft-NMS algorithm, both qualitatively and quantitatively.

4.2.1 Training Configuration

Following shows the configuration used during the training phase of the model:

- LEARNING_RATE = 0.001
- MOMENTUM = 0.9

- WEIGHT_DECAY = 0.0001
- GAMMA = 0.1
- STEPSIZE = [30000]
- BATCH_SIZE = 128
- IMS_PER_BATCH = 1
- FG_FRACTION = 0.25
- FG_THRESH = 0.5
- BG_THRESH_HI = 0.5
- BBOX_THRESH = 0.5
- BBOX_INSIDE_WEIGHTS = (1.0, 1.0, 1.0, 1.0)
- BBOX_NORMALIZE_MEANS = (0.0, 0.0, 0.0, 0.0)
- BBOX_NORMALIZE_STDS = (0.1, 0.1, 0.2, 0.2)
- RPN_POSITIVE_OVERLAP = 0.7
- RPN_NEGATIVE_OVERLAP = 0.3
- RPN_BATCHSIZE = 256
- RPN_NMS_THRESH = 0.7
- RPN_PRE_NMS_TOP_N = 12000
- RPN_POST_NMS_TOP_N = 2000

4.2.2 Test Configuration

Following shows the general default NMS configuration used during testing phase. The configuration for Soft-NMS will be discussed in the following chapters.

- NMS = 0.3
- RPN_NMS_THRESH = 0.7
- RPN_PRE_NMS_TOP_N = 6000
- RPN_POST_NMS_TOP_N = 300

4.2.3 Experiment 1: Visual Inspection

In this experiment we first implement the TensorFlow version of Soft-NMS and then did the visual inspection for both Faster R-CNN + NMS and Faster R-CNN + Soft-NMS on several selected images. We tuned several parameters in Soft-NMS such as the method($LinearandGaussian$), Sigma, Threshold, etc. to find out how Soft-NMS performed under different settings by visually checking the results.

4.2.4 Experiment 2: Detector Precision Measurement

The second experiment is aimed to compare the quantitative performance of the Soft-NMS with different hyperparameters (re-scoring functions, overlap threshold parameter Nt, σ) and architectures(Vgg166,Res101). The mean precision over 20 classes and precision for 10 of these classes are computed and compared in this study. Sensitivity analysis of these parameters are investigated as well.

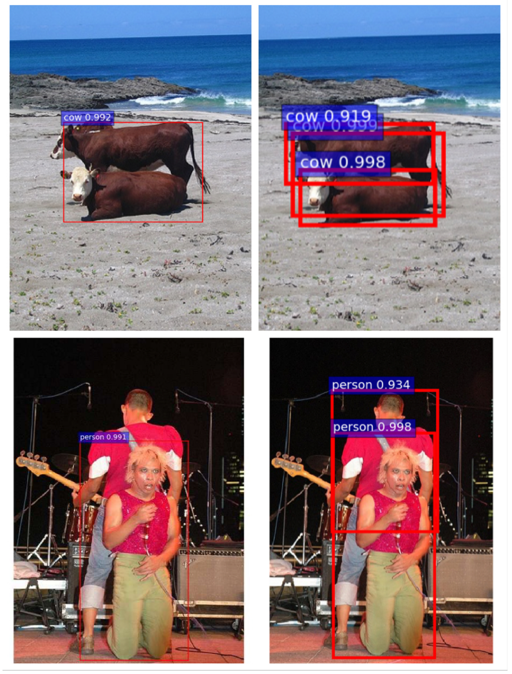


Figure 6. Experiment1: Visual Inspection Result Part I; Left: Faster R-CNN+NMS; Right: Faster R-CNN+Soft-NMS; Method=Linear, Threshold=0.05, Nt=0.7

5. Results

5.1. Results of Visual Inspection

Figure 6 and **Figure 7** shows the object detection results of Faster-RCNN + NMS and Faster R-CNN + Soft-NMS on dataset 1. The images on the left side are the results from Faster-RCNN + NMS and the images on the right side are results of Faster R-CNN + Soft-NMS.

The configuration of Soft-NMS which leads to the results shown in both figures were set as follow: Method=Linear, Nt=0.7, Threshold=0.05. The results of other Soft-NMS configurations are not shown in this report but we found that with the increasing of the threshold and Nt, more objects will be detected, and vice versa.

From these two figures it can be seen that the approach of Faster R-CNN + Soft-NMS outperformed the Faster-RCNN + NMS approach. For the overlapping objects belong to the same category, with Faster-RCNN + NMS only a single object can be detected under this situation. While with Faster R-CNN + Soft-NMS applied, multiple objects can be detected, which is close or even equal to the actual number of objects on the original image.

5.2. Results of Detector Precision Measurement

Table1 demonstrates the results of mean precision and precision for 10 out of 20 classes in PASCAL VOC2007 dataset. The discarding threshold is set as 0.001, Nt as

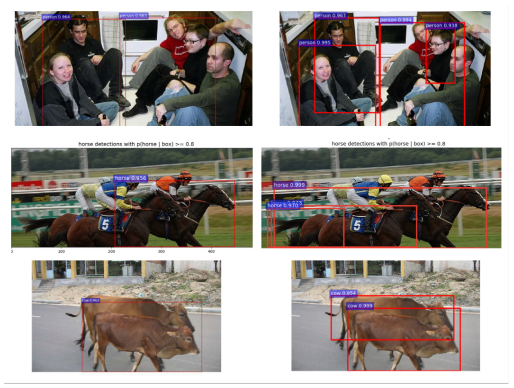


Figure 7. Experiment1: Visual Inspection Result Part II: Visual Inspection Result 2; Left: Faster R-CNN+NMS; Right: Faster R-CNN+Soft-NMS; Method=Linear, Threshold=0.05, Nt=0.7

0.3 and the $/sigma$ is set as 0.5 for linear and Gaussian re-scoring functions. We obtain a 1.2% and 1% improvement in the precision for Vgg16 and Res101 networks respectively, which is significant in PASCAL VOC2007 dataset. This result indicates the effect of architectures and re-scoring functions on the mean precision. It can be found that the Soft-NMS improves the precision of Faster-RCNN for all cases no matter the re-scoring functions, and the exponential functions are slightly better than the other functions. The similar improvements in each architecture lead to the fact that it is independent of the training model. This algorithm does not contribute to the complexity of the model since it does not require re-train the model for fine-tuning. Therefore Soft-NMS is generally applicable in all object detection networks for slight increase in precision with nothing to sacrifice.

The sensitivity analysis has been conducted as well, as shown in Table2 and Table3. In Soft-NMS algorithm, the σ parameter and an overlap threshold parameter Nt are varied and measured in terms of mean precision on the PASCAL VOC2007 dataset. We can find that the effect of σ on mean precision is stable in the range (0.2,0.8) with variance less than 0.5%. σ=0.5 is slightly better than other values. The threshold parameter Nt has similar effect on the precision of the detector. Nt=0.3 is slighly better than Nt=0.7 when σ=0.5 is fixed.

[0]aeroplane
[1]bicycle
[2]bird
[3]boat
[4]bus
[5]car
[6]cat
[7]cow
[8]dog
[9]person

6. Conclusion

Soft-NMS performs better than traditional NMS in object detection, especially for the image that contains multiple overlapped objects belong to the same class. Choosing appropriate values of the hyperparameters and utilizing exponential re-scoring functions in the Soft-NMS algorithm yields a possibility to further improvements in the precision. More importantly, Soft-NMS indeed helps to improve the precision of Faster R-CNN without introducing additional hyper parameters and re-training the Faster R-CNN model. To this end, Soft-NMS is meaningful contribution to the improvement of Faster R-CNN.

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] 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.

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

[4] 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.

[5] 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.

[6] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[8] 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.

[9] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[10] 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.

[11] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

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

Table 1. Results of mAP and AP for 10 classes on PASCAL VOC 2007 test set with NMS and Soft-NMS.

Model	Algorithm	S_i	mAP	AP-1[1]	AP-2[2]	AP-3[3]	AP-4[4]	AP-5[5]	AP-6[6]	AP-7[7]	AP-8[8]	AP-9[9]	AP-10[10]
Vgg16	NMS	Hard	0.7543	0.7627	0.8250	0.7549	0.6515	0.8394	0.8742	0.8719	0.8236	0.8336	0.7838
Vgg16	Soft-NMS	Linear	0.7666	0.7815	0.8457	0.7794	0.6745	0.8348	0.8732	0.8727	0.8301	0.8391	0.8251
Res101	NMS	Hard	0.7972	0.8302	0.8695	0.8137	0.7408	0.8772	0.8799	0.8820	0.8679	0.8857	0.8275
Res101	Soft-NMS	Linear	0.8067	0.8552	0.8764	0.8297	0.7584	0.8771	0.8801	0.8826	0.8683	0.8861	0.8497
Res101	Soft-NMS	Gaussian	0.8047	0.8624	0.8761	0.8246	0.7620	0.8749	0.8765	0.8837	0.8667	0.8845	0.8385
Res101	Soft-NMS	Poly	0.8020	0.8401	0.8758	0.8225	0.7569	0.8767	0.8797	0.8801	0.8711	0.8792	0.8439
Res101	Soft-NMS	Exp	0.8077	0.8608	0.8759	0.8349	0.7607	0.8747	0.8777	0.8831	0.8687	0.8846	0.8423

Table 2. Sensitivity analysis of Soft-NMS with Nt=0.3,Gaussian functions

Model	Algorithm	Gaussian	mAP	AP-1[1]	AP-2[2]	AP-3[3]	AP-4[4]	AP-5[5]	AP-6[6]	AP-7[7]	AP-8[8]	AP-9[9]	AP-10[10]
Res101	Soft-NMS	Sigma=0.2	0.8030	0.8670	0.8728	0.8374	0.7563	0.8768	0.8732	0.8817	0.8673	0.8825	0.8304
Res101	Soft-NMS	Sigma=0.4	0.8044	0.8633	0.8766	0.8257	0.7615	0.8737	0.8766	0.8830	0.8684	0.8834	0.8399
Res101	Soft-NMS	Sigma=0.5	0.8067	0.8552	0.8764	0.8297	0.7584	0.8771	0.8801	0.8826	0.8683	0.8861	0.8497
Res101	Soft-NMS	Sigma=0.6	0.8047	0.8624	0.8761	0.8246	0.7620	0.8749	0.8765	0.8837	0.8667	0.8845	0.8385
Res101	Soft-NMS	Sigma=0.8	0.8025	0.8556	0.8769	0.8222	0.7548	0.8727	0.8768	0.8827	0.8680	0.8827	0.8396

Table 3. Sensitivity analysis of Soft-NMS with sigma=0.5,Gaussian functions

Model	Algorithm	Nt	mAP	AP-1[1]	AP-2[2]	AP-3[3]	AP-4[4]	AP-5[5]	AP-6[6]	AP-7[7]	AP-8[8]	AP-9[9]	AP-10[10]
Res101	Soft-NMS	0.3	0.8067	0.8552	0.8764	0.8297	0.7584	0.8771	0.8801	0.8826	0.8683	0.8861	0.8497
Res101	Soft-NMS	0.7	0.8045	0.8624	0.8761	0.8246	0.7620	0.8749	0.8765	0.8837	0.8667	0.8845	0.8385