Post-NMS Training Strategy for Object Detection

Lu Qi *
The Chinese University of Hong Kong

Shuqin Xie *
Shanghai Jiao Tong University

Shu Liu
Youtu Lab, Tencent

Jiaya Jia
Youtu Lab, Tencent

Yue Zhou
Shanghai Jiao Tong University

Abstract

Object detection algorithms have achieved excellent performance while training still does not involve the final non-maximal suppression (NMS) to remove unnecessary proposals. We note intriguingly final proposal removal, albeit seemingly subtle and straightforward, actually finds a lot of room to improve and affects overall performance. In this paper, we propose a *post-NMS training strategy* to directly perform optimization upon results of NMS. Our method, after predicting proposals and their classes, groups proposals into a set of clusters during NMS. We then make the important design to infer the best proposals by integrating all information within clusters. We call it post-NMS learning that models inter- and intra-cluster relationship. Our method makes it possible to incorporate NMS into an end-to-end training framework. Experiments on the challenging MSCOCO object detection tasks yield consistent improvement when using various object detection frameworks. Our method achieves 43.9 mAP even with one single model.

1 Introduction

Object detection is a fundamental task in computer vision and was extensively researched. Proposal-based top-down methods [1, 2, 3, 4, 5, 6, 7] have been dominant in this field. Recently, proposal-free bottom-up methods [8, 9, 10], which directly regress object location, also achieve excellent performance.

Non-maximal suppression (NMS) is an essential step in almost all object detection algorithms to reduce the vast amount of redundant detection results during the course of inference. On the one hand, it is generally not considered during training because the common strategy is to sample a set of foreground and background proposals in all possible locations to train the network, regardless of the NMS step. Though it works well thanks to the carefully designed sampling strategy, there is still much room for improvement since this type of training only optimizes on an intermediate result rather than the final one used for evaluation. On the other hand, some recent work [11, 12] replaces NMS with a neural network. They usually result in more complicated network structures and extra computation, which may slow down object detection.

In this paper, we follow a completely different line to introduce *post-NMS training*, which directly trains the network on NMS output. The major benefit is to integrate NMS into the training stage with rather limited computational overhead. The key is to formulate NMS as a dynamic clustering process, where at each step a most confident proposal is chosen to group a set of proposals into a single cluster. This format perfectly solves the essential problem of training on sparse signals when directly using NMS output. Meanwhile, maintaining the entire cluster provides rich information than keeping just one single proposal, making it possible to further improve result quality by utilizing vastly helpful context information.

*Authors contributed equally to this paper.

Our whole system can built on anchor-based detection frameworks [13, 3] with NMS. We only need to modify NMS by taking resulting proposals as input and outputting a set of clusters. The key module is to recompute these objectness scores for each cluster given categories. This is achieved nicely in our system with a decoupling layer to separate this module from previous steps, a classification network to evaluate objectness scores given the class categories, and a localization network to further refine the box location. A class-encoding feature is proposed to model the condition under a specific class. The detailed cluster-based learning involves an inter-cluster learning branch to enforce integration of information within the entire cluster and merge close clusters, and an intra-cluster learning branch accounting for importance values for different proposals within a cluster and addressing the problem of sparse supervision signals.

Our method finally accomplishes NMS in an end-to-end training framework. Experiments on MSCOCO object detection task demonstrates the surprising effectiveness of our method where we observe consistent improvement when using various popular detection frameworks with different backbones. Extensive ablation studies are provided for detailed analysis of the proposed method.

2 Related Work

Anchor-based Object Detectors Anchor-based top-down methods are popular in object detection [1, 2, 3, 14, 15, 16, 17, 18], while keypoint based methods [8, 19, 9, 10, 20] emerge recently. Contemporary anchor-based approaches mostly fall into two categories, *i.e.*, the two-stage and single-stage paradigms. Improvement of two-stage detectors stem from the development of proposal generation and RoI feature extraction in series of R-CNN [1, 2, 3]. Proposals are first generated from anchors and then sampled for further discrimination and localization. Recent works [21, 15, 18, 4, 22] modify either architecture [21, 15, 18] or pipeline [4, 22] to further improve performance. As the representation of single-stage pipeline, YOLO [7], SSD [6] and RetinaNet [5] directly classify and regress anchors for high inference speed. For all these methods, our method can be easily applied to further boost up performance with little computation overhead.

Duplicate Removal Duplicate removal aims to eliminate highly overlapped detection results and only retain the most accurate bounding box for each object. NMS [23] is the most popular method due to its simplicity and efficiency. Regarding its variants, instead of suppressing all non-maximum bounding boxes, SoftNMS [24] keeps them according to the degree of overlap with "key" boxes. Different from SoftNMS, box-voting [13, 25] renew the "key" boxes by grouping all their highly-overlapped neighborhood. These methods only consider the score and location of each bounding box, which do not provide strong information to determine the final bounding boxes.

Recent works on duplicate removal uses convolutional neural network to help final selection among the large number of proposals. In [11, 12], learning NMS is achieved with elaborate network design. In [26, 27], NMS and box-voting are provided with an extra localization confidence score for further refinement. Relation network [17] is the first one to completely abandon NMS but still encounters problem of extreme imbalance between positive and negative samples.

Refinement Module Some recent works improve performance by cascading modules [4, 28] or changing sampling strategies [29, 30, 31]. They all follow the 'coarse-to-fine' detection pipeline to filter out most of the simple background windows and then process those more difficult proposals [12]. In contrast, our method pays more attention to proposals used for final evaluation and uses the entire cluster for training.

3 Our Approach

Our method is built on object detection framework with a novel post-NMS stage, illustrated in Figure 1. We start with brief introduction of the pre-NMS operations, followed by viewing NMS as dynamic clustering. Then we explain the post-NMS stage, which is our focus.

3.1 Before NMS

Before NMS, similar to most anchor-based object detection algorithms [2, 3, 15] we use a classification branch to evaluate existence of objects and a localization regression branch to predict location of the object. During training, it randomly samples a set of foreground and background proposals and

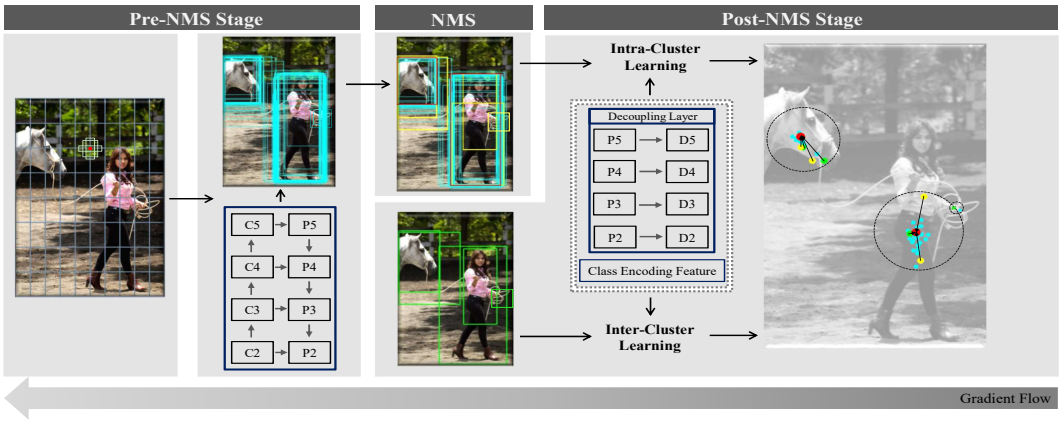

Figure 1: Illustration of our end-to-end object detection framework. **(a)** Prior to NMS stage, proposals are generated by Faster R-CNN with FPN backbone. C and P denote different feature map levels in ResNet and FPN. **(b)** NMS stage uses the most confident boxes (yellow ones) to group close proposals (blue ones) into clusters. Representative boxes (green ones) are generated to represent clusters. **(c)** Post-NMS stage uses the decoupling feature D and class-encoding features to perform classification and localization. Intra- and inter-cluster learning are introduced to make the cluster more compact and merge close clusters. The dots in the right most image represent centers of boxes where red ones are ground truth and the other colors (yellow, blue and green) corresponds to boxes described before.

computes classification and localization loss on these proposals. The classification loss is computed by $L_{cls} = \sum_i^N -\log p_i$ where p_i is the probability on the true class label. The localization loss is the commonly used smoothed-L1 loss in [2]. It is also possible to cascade multiple stages by gradually changing the foreground threshold [4].

3.2 NMS as Dynamic Clustering Process

Prior to delving into the detail of our main post-NMS stage, we discuss how the NMS algorithm can be explained as a dynamic clustering process. The most common understanding of NMS is that at each step, the algorithm greedily chooses the most confident proposal and eliminates boxes close to it. From another view, at each step, the algorithm uses the most confident proposal to group a set of proposals into a cluster. The cluster here is defined as a set of proposals whose IoU with its most confident proposal are larger than a threshold. The clustering understanding here allows us to successfully address the crucial problem of sparse training signals in the following important post-NMS stage.

3.3 Post-NMS Stage

This stage is new to directly perform optimization on output of NMS. It takes as input a set of clusters and outputs a set of proposals. The basic operation unit is a cluster. We detail our design regarding modeling, learning strategy and inference in what follows.

3.3.1 Modeling

Modeling post-NMS consists of three components: a **decoupling layer** to separate this module from previous proposal generation, a **classification network** to compute the objectness score given class K, and a **localization network** to refine box location.

Decoupling Layer The objectives for our post-NMS stage and operations before it are different. The proposal generation is to optimize results regarding each proposal while our post-NMS is to update results in terms of each cluster. Since the mechanisms are different, it is beneficial to separate them, as validated in our experiments. We let features generated before NMS be F_{pre} and features in our post-NMS stage be F_{post}. F_{post} can be represented as $F_{post} = \Gamma(F_{pre})$, where Γ is a function

parameterized by the decoupling layer. In this paper, we take the backbone feature (*e.g.* FPN feature) as F_{pre} and a 3×3 convolution layer with 256 channels for the decoupling layer. Its output is F_{post}.

Classification and Localization Networks After decoupling, we introduce small-scale classification and localization networks to re-compute objectness score and refine localization for each cluster.

Recall that in common proposal generation frameworks of object detection by [2, 3, 4], the class probability for a proposal is computed by a softmax function as

$$p(K = k) = \frac{e^{z_k}}{\sum_i^C e^{z_i}}, \tag{1}$$

where C is the total number of classes and z_k is the logit for class k. Also in the NMS stage, for a class K, the algorithm maintains proposals whose class probabilities on K are larger than a threshold.

We note it is valid for one proposal to be assigned to multiple categories in our configuration because one position could have multiple objects when heavy occlusion arises. However, the above $softmax$ function makes different categories compete and lower some scores, which does not allow multi-category high scores to happen. In addition, decrease of scores also influences mAP, since proposals with low scores are put to the latter position regarding the evaluation metric. These facts indicate that *the softmax function is not suitable* in our post-NMS framework. We instead propose a binary classification network to compute the conditional probability of objectness given a class k, denoted as

$$p(obj = 1|K = k).$$

Inspired by the thought in [32, 17, 12], the condition of $K = k$ is encoded into a class-encoding vector F_k that has the same number of channels as F_{post}. Each dimension of F_k is computed as

$$F_{(k,2i)} = \sin(k/1000^{2i/d})$$

$$F_{(k,2i+1)} = \cos(k/1000^{2i/d})$$

where k is the class, i is the dimension and d is a constant value. Each dimension of class-encoding F_k corresponds to a sinusoid. This formulation ensures each class is encoded into a unique vector. Therefore, the conditional probability of objectness given class k is computed by

$$p(obj = 1|F_r, F_k; W), \tag{2}$$

where F_r is RoI feature extracted from F_{post} and F_k is class-encoding feature. W is the weight of the network.

3.3.2 Learning Strategy

We discuss the learning strategy in this section. Since this stage takes clusters as input, we consider learning based on clusters. Specifically, we use inter-cluster learning to differentiate among clusters and intra-cluster learning to model various importance of proposals within a cluster.

Inter-cluster Learning We consider the relationship between different clusters. Intuitively, at the beginning of training phase, there are a lot of clusters and only a few proposals for each cluster. During the course of training, the number of clusters reduces and more proposals for each cluster are generated. Put differently, clusters gradually merge. This is an inherent property since the learning procedure is still based on each individual proposal.

To further enhance it, we propose inter-cluster learning that accumulates all information within a cluster so that the algorithm is forced to consider a cluster. We compute a representative box b_r for cluster C using function f to accumulate all proposals within the cluster, that is, $b_r = f(C)$. f is flexible as long as it summarizes the cluster information. For simplicity, we use *union* for f and b_r is computed by

$$b_r^{tl} = \min_{b \in C} b^{tl}, \quad b_r^{br} = \max_{b \in C} b^{br}, \tag{3}$$

where b^{tl} and b^{br} are the bounding box top-left and bottom-right corner locations. Then b_r is fed into the classification and localization networks to further merge close clusters and decrease scores of non-object clusters, reducing false positives. A cluster C with category K is considered positive if $\max_{b \in C} \text{IoU}(b, gt_K) > 0.5$ where gt_K is the ground truth annotation on class K. Otherwise, a negative

label is assigned. The box regression target is only assigned when C is considered positive and is computed following the method of [2].

Intra-cluster Learning We further consider varying importance of proposals within a cluster. We expect the most confident and precisely localized proposals are more important than the rest. To this end, we propose an importance sampling strategy to assign different weights to proposals. The importance for the i-th proposal is defined as

$$w^i_{cls} = \frac{\text{IoU[i]}}{\max(\text{IoU})}, \quad w^i_{loc} = \frac{\text{score[i]}}{\max(\text{score})},$$

where subscripts cls and loc indicate the classification and the localization branches respectively. The final losses for cluster C are given by

$$L_{cls} = \sum_i w^i_{cls} L^i_{cls}, \quad L_{loc} = \sum_i w^i_{loc} L^i_{loc},$$

where L^i_{cls} is the binary cross entropy loss and L^i_{loc} is the smoothed-L1 loss on the i-th proposal. The class label and matched ground truth for each proposal are identical to the cluster's.

This sampling strategy assigns larger weights to proposals with higher IoUs, forcing the classification branch to focus on them and increase their scores. So does the localization branch where more attention is drawn to those with higher scores. Another benefit is that it unifies classification and localization. The weights for classification branch depends on the regression result and vice versa. In this way, we connect them and make proposal suppression performs more accurately, leading to superior performance.

Note that our sampling strategy is different from those used in pre-NMS methods [2, 3] where the latter only considers proposals with IoU > 0.5 as positive samples. Our cluster-based sampling strategy contrarily assigns positive labels to proposals with IoU < 0.5, as long as their cluster is considered positive. This manifests that our method makes use of richer supervision signals.

Finally, this sampling strategy does not have the problem of sparse supervision signals when training on NMS output. Note that the conventional understanding of NMS will only keep a few proposals and that number is too small to train a network. Adopting the clustering explanation addresses this problem by keeping all proposals within a cluster and using them to train the network. In this way, much more supervision signals are provided and the problem of sparse training signal is addressed. T

3.4 End-to-End Framework

Training The design of our post-NMS module makes it easy for joint training the whole system for object detection. We define the overall loss for our framework as

$$L = \alpha L_{pre} + \beta L_{post},$$

where α is the weight for loss L_{pre} before proposal suppression and β is the weight for post-NMS stage loss L_{post}. Note that L_{post} is computed by $L_{post} = \frac{1}{N}(L_{cls} + L_{loc})$ and N is the number of proposals for this stage. Since NMS has eliminated most of background proposals, L_{post} mainly focuses on foreground proposals and is more difficult to optimize, resulting in much larger loss than L_{pre}. To address this issue, we set α to 1 and β to $\frac{num(b_k)}{num(b_a)}$ where b_k denotes the boxes kept after suppression and b_a denotes the boxes before it. This term balances the loss in the two stages.

Inference At inference stage, for each cluster, we choose the most confident proposal and re-compute its objectness score and location. The final score is computed as

$$s = w_{ori} s_{ori} + w_{clu} s_{clu} + w_{obj} s_{obj},$$

where $(w_{ori}, w_{clu}, w_{obj})$ are the weights for its original score s_{ori}, its cluster score s_{clu} and objectness score s_{obj}. In our experiments, we set $(w_{ori}, w_{clu}, w_{obj})$ to (0.8, 0.1, 0.1). The final box localization is computed using localization network's prediction.

4 Experiments

We conduct experiments on the challenging MSCOCO [33] object detection task to verify the effectiveness of our method. We apply our method to several popular detection frameworks, including both

two-stage methods (Faster R-CNN [3] and Cascade R-CNN [4]) and one-stage method (RetinaNet [5]). We also make comparison with other NMS methods regarding the way to suppress proposals.

Dataset MSCOCO [33] dataset is a popular large-scale benchmark in object detection. Heavy occlusion and complex scenes make it very challenging. It labels 80 object categories and contains 7.7 instances per image. The training set *train-2017* split contains 115k images and the validation set *val-2017* split has 5k images. Ground truth annotation on these two subsets are available. The test set *test-dev* split consists of 20k images with no annotation provided. Results are reported under standard COCO-style AP metric, which averages mAP across different IoU thresholds (ranging from 0.5 to 0.95 with interval 0.05).

Implementation Details We use the *train-2017* subset to train models and test them on the *test-dev* subset. Without specially noted, we train the network with 90k iterations on 8 NVIDIA TITAN X Pascal GPUs. The batchsize is 16 and the learning rate starts from 0.02 and decays by a factor of 0.1 on iterations 60k and 80k. The momentum is set to 0.9.

Framework	Backbone	AP	AP^{50}	AP^{75}	AP^S	AP^M	AP^L
Faster R-CNN [3]	ResNet-50-FPN	$\mathbf{38.4}(+\mathbf{1.9})$	58.3	41.8	21.6	41.4	51.2
	ResNet-101-FPN	$\mathbf{40.6}(+\mathbf{1.7})$	60.3	44.2	23.0	43.9	53.9
	ResNeXt-101-FPN	$\mathbf{42.5}(+\mathbf{1.5})$	63.2	46.7	25.2	46.3	55.6
RetinaNet [5]	ResNet-50-FPN	$\mathbf{37.2}(+\mathbf{1.3})$	55.7	40.5	20.3	40.2	49.3
	ResNet-101-FPN	$\mathbf{38.9}(+\mathbf{1.3})$	59.8	41.7	22.1	42.4	50.7
	ResNeXt-101-FPN	$\mathbf{40.2}(+\mathbf{1.2})$	59.8	43.6	22.7	43.4	53.3
Cascade R-CNN [4]	ResNet-101-FPN (3)	$\mathbf{42.9}(+\mathbf{0.8})$	61.1	46.2	23.6	45.5	55.3
	ResNet-101-FPN ($1 \sim 3$)	$\mathbf{43.4}(+\mathbf{0.6})$	62.0	46.5	23.9	45.9	56.5

Table 1: Performance comparison with state-of-the-art methods after applying post-NMS stage for training. Results on *test-dev* split are reported. The content in bracket denotes performance gains over their baseline counterparts.

Results on Detection We report results on the *test-dev* split in Table 1, where we achieve consistent improvement over different detection frameworks with various backbones. For two-stage methods, We improve Faster R-CNN with 1.9, 1.7 and 1.5 mAP respectively for ResNet-50-FPN, ResNet-101-FPN and ResNeXt-101-FPN backbones. For one-stage methods, we boost RetinaNet by 1.3, 1.2 and 1.3 mAP respectively for the same three backbones.

We also improve the very strong baseline Cascade R-CNN by 0.6 mAP. We note that Cascade R-CNN reach its performance ceiling after stacking 3 stages, as pointed out in [4]. So the improvement here does not stem from using an extra stage, and instead comes from our cluster-based sampling strategy. Using all information within the entire cluster for inference also helps.

Method	AP	AP^{50}	AP^{75}	AP^S	AP^M	AP^L	AR^1	AR^{10}	AR^{100}
NMS [23]	36.7	58.4	39.5	21.0	40.1	48.0	30.5	48.0	50.4
Box-Voting [13]	37.1	58.4	40.3	21.1	39.9	48.4	30.8	48.4	50.9
Soft-NMS [24]	37.5	58.4	41.3	21.4	40.1	48.9	30.5	**51.9**	**57.2**
IoU-NMS [26]	37.6	56.2	40.9	20.5	41.8	47.4	31.5	50.2	51.8
Softer-NMS [27]	38.0	57.8	41.3	21.2	42.3	47.9	31.1	50.4	52.3
SCE [12]	37.9	58.0	41.2	20.9	**42.5**	47.9	31.3	49.0	52.1
Ours	**38.6**	**58.2**	**41.9**	**21.6**	41.6	**51.4**	**32.1**	**50.5**	**53.1**

Table 2: Comparison with other NMS methods on the *val-2017* subset. All use ResNet-50-FPN as the backbone.

Comparison on Suppression Strategies We compare our method with other proposal suppression methods [23, 13, 24, 26, 27, 12] and report results in Table 2. Our method yields the best performance. Compared to the heuristic methods such as Box-voting [13] and Soft-NMS [24], we effectively utilize proposals' features for proposal suppression rather than exploiting only scalars of location and scores. Compared to learning-based methods, such as IoU-NMS [26] and Softer-NMS [27], our method pays more attention to the most important proposal within a cluster rather than treating them similarly important. Compared to SCE [12], our method is much simpler and can be easily integrated into existing frameworks for end-to-end training, which is however impossible for [12].

5 Ablation Studies

We perform extensive ablation studies to evaluate usefulness of each component in our method. We report results on both NMS stage and post-NMS stage, which reflects post-NMS stage's influence over previous one. Results on *val-2017* subset are reported in Table 3.

	NMS	*post-NMS*	AP	AP^{50}	AP^{75}	AP^S	AP^M	AP^L
baseline	✓		36.7	**58.4**	39.5	21.0	40.1	48.0
w/o DCL	✓		34.5	54.9	37.4	18.4	36.6	46.7
		✓	36.0	57.6	38.3	19.5	37.2	48.0
w/o CEF	✓		36.7	58.3	39.5	21.0	40.1	48.1
		✓	37.9	57.6	41.3	21.1	40.3	50.3
w/o IRC	✓		36.5	57.5	39.7	20.5	38.9	48.0
		✓	38.2	57.6	41.7	21.2	40.6	50.7
w/o IAC	✓		36.6	58.0	39.8	20.2	39.2	48.3
		✓	36.9	58.2	39.9	20.4	39.5	48.5
full model	✓		37.0	58.3	39.6	21.0	40.3	48.5
		✓	**38.6**	58.2	**41.9**	**21.6**	**41.6**	**51.4**
			+1.9	-0.2	**+2.4**	**+0.6**	**+1.5**	**+3.4**

Table 3: Ablation study results on *val-2017* subset. We compare our **full model** with Faster R-CNN ResNet-50-FPN **baseline**. **w/o X** means removing **X** from the full model and X could be **DCL** (decoupling layer), **CSF** (class encoding feature), **IRC** (inter-cluster learning) and **IAC** (intra-cluster learning). The last row is the overall improvement of full model over the baseline.

Decoupling Layer Removing the decoupling layer (**w/o DCL**) causes notable performance drop (about 2.5 mAP for both *NMS* and *post-NMS* settings). It is even worse than the baseline (2.2 mAP less). This demonstrates the necessity of separating post-NMS stage's feature from other modules, considering its unique objective.

Class Encoding Feature Removing the class encoding feature (**w/o CEF**) degrades the post-NMS stage into an class-agnostic refinement stage, which leads to 0.7 mAP performance decrease. This indicates the strong class prior introduced by class encoding feature is actually beneficial.

Inter-cluster Learning Removing inter-cluster learning (**w/o IRC**) also affects the performance by 0.4 mAP. This procedure integrates information within the cluster. We test different integration strategies, where we compute union, intersection and average of all proposals within a cluster. We also experiment with box-voting [13]. Results are reported in Table 4. Using the union box achieves the best performance, which suggests incorporating more context is useful for final performance.

Intra-cluster Learning Removing the intra-cluster learning (**w/o IAC**) degrades performance by 1.7 mAP. The gap is caused by the sparse supervision signal problem where introducing intra-cluster learning addresses it. We also analyze the impact of importance sampling strategy by exploring different sampling strategies: *normal* for independent classification and localization loss, w_{cls} *only* for localization-aware classification loss only, w_{loc} *only* for score-aware localization loss only, and *all* for both terms. Results are reported in Table 5. The *all* setting achieves the best performance. It is much better than the *normal* setting (with 1.1 mAP improvement), manifesting the importance of this design.

	inter	*avg*	*box-voting*	*union*
AP	37.9	38.2	38.2	**38.6**
AR	52.0	52.1	52.3	**53.1**

Table 4: Results of inter-cluster learning.

	normal	w_{cls} only	w_{loc} only	*all*
AP	37.5	38.1	38.4	**38.6**
AR	51.9	52.2	52.9	**53.1**

Table 5: Results of intra-cluster learning.

Inference Finally, we investigate the influence of information fusion at inference phase. We vary the weight terms and steps for re-scoring and re-localization, and report results in Table 6. The re-localization in post-NMS stage improves the box quality a lot because it makes the cluster more compact. Using weights of (0.8, 0.1, 0.1) yield the best performance.

w_o	w_c	w_i	Re-locate	AP	w_o	w_c	w_i	Re-locate	AP
1.0	0.0	0.0	×	37.0	1.0	0.0	0.0	✓	38.1
0.0	1.0	0.0	✓	33.4	0.0	0.0	1.0	✓	36.2
0.8	0.1	0.1	×	36.6	0.8	0.1	0.1	✓	**38.6**

Table 6: Results of different ways of information fusion at inference phase. Re-locate means using post-NMS stage's localization branch to refine the box.

6 Quantitative Analysis

We provide quantitative analysis in this section. More are in the supplementary file.

For each cluster, we record the following variables: KeyIoU, MaxIoU, and AvgIoU. KeyIoU is IoU of its most confident proposal with its matched ground truth. MaxIoU and AvgIoU is the maximal and average IoU among all proposals in the cluster respectively. We compute these variables in both before post-NMS stage and after post-NMS stages, denoted by suffix '_Pre' and '_Post'. A histogram of these variables is drawn in Figure 2. The horizon axis marks IoU intervals ranging from 0 to 1 with step 0.05. The vertical axis labels the total number of X within the interval, where X represents KeyIoU, MaxIoU, and AvgIoU.

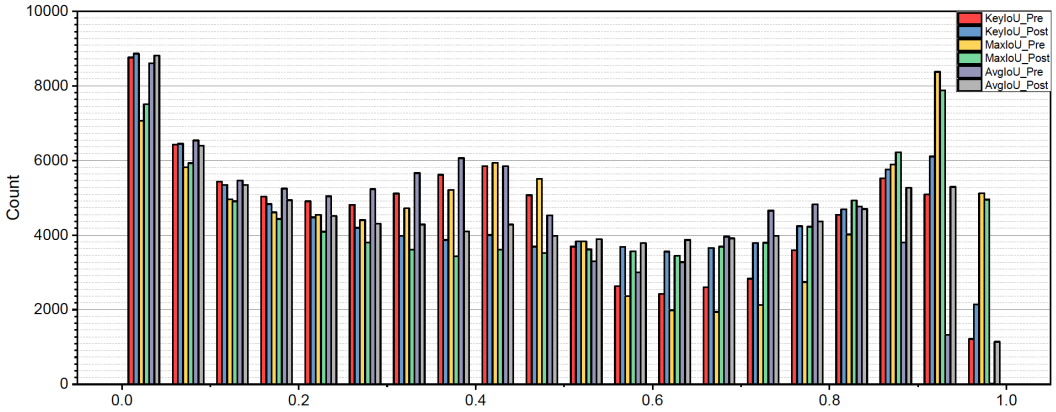

Figure 2: Histogram of IoU statistics. Horizon axis denotes different IoU intervals ranging from 0 to 1 with step 0.05. Verical axis labels the counting of variables within the interval. The legend in the top right corner denotes differnt variables.

The figure shows that KeyIoU_Post are consistently larger than KeyIoU_Pre in the IoU intervals of 0.5 to 1 and consistently smaller than KeyIoU_Pre from intervals of 0 to 0.5. This means the post-NMS stage can effectively improve the box quality and reduce the number of low-quality proposals. The MaxIoU term also gains similar improvement where MaxIoU_Post are usually larger than MaxIoU_Pre.

Note that AvgIoU_Post are significantly larger than AvgIoU_Pre on high IoU intervals (from 0.8 to 0.9). This is because the post-NMS stage works on units of clusters, making information more compact than that on individual proposals.

7 Conclusion

We have presented a novel *post-NMS training strategy* to directly perform optimization upon result of NMS, which manages to integrate NMS into training course. It yields a number of new advantages compared with previous solutions. Instead of choosing the most appropriate bounding boxes, we unify information within clusters and refine the key results with our special design including decoupling layer and class-specific features with inter- and intra-cluster learning. Extensive experiments and ablation studies were conducted and the consistent improvement manifests the effectiveness of our approach over different state-of-the-art backbone frameworks. New high is achieved on all of them. Other information fusion strategies will be explored in our future work.

References

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[2] Ross B. Girshick. Fast R-CNN. In *ICCV*, 2015.

[3] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[4] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: delving into high quality object detection. In *CVPR*, 2018.

[5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

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

[9] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *arXiv*, 2019.

[10] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv*, 2019.

[11] Jan Hendrik Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *CVPR*, 2017.

[12] Lu Qi, Shu Liu, Jianping Shi, and Jiaya Jia. Sequential context encoding for duplicate removal. In *NeurIPS*, 2018.

[13] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *CVPR*, 2015.

[14] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010.

[15] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[16] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.

[17] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *CVPR*, 2017.

[18] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. *CVPR*, 2018.

[19] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019.

[20] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *arXiv*, 2019.

[21] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[22] Francisco Massa and Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. `https://github.com/facebookresearch/maskrcnn-benchmark`, 2018. Accessed: [Insert date here].

[23] Alexander Neubeck and Luc J. Van Gool. Efficient non-maximum suppression. In *ICPR*, 2006.

[24] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-nms - improving object detection with one line of code. In *CVPR*, 2017.

[25] Shu Liu, Cewu Lu, and Jiaya Jia. Box aggregation for proposal decimation: Last mile of object detection. In *CVPR*, 2015.

[26] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, 2018.

[27] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *CVPR*, 2019.

[28] François Fleuret and Donald Geman. Coarse-to-fine face detection. *International Journal of Computer Vision*, 2001.

[29] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.

[30] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra R-CNN: towards balanced learning for object detection. *arXiv: 1904.02701*, 2019.

[31] Yuhang Cao, Kai Chen, Chen Change Loy, and Dahua Lin. Prime sample attention in object detection. *arXiv:1904.04821*, 2019.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[33] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.