# AutoAssign: Differentiable Label Assignment for Dense Object Detection

Benjin Zhu, Jianfeng Wang, Zhengkai Jiang, Fuhang Zong, Songtao Liu, Zeming Li, Jian Sun
MEGVII Technology
{zhubenjin, wangjianfeng, jiangzhengkai, zongfuhang, liusongtao, lizeming, sunjian}@megvii.com

## Abstract

*Determining positive/negative samples for object detection is known as label assignment. Here we present an anchor-free detector named AutoAssign. It requires little human knowledge and achieves appearance-aware through a fully differentiable weighting mechanism. During training, to both satisfy the prior distribution of data and adapt to category characteristics, we present Center Weighting to adjust the category-specific prior distributions. To adapt to object appearances, Confidence Weighting is proposed to adjust the specific assign strategy of each instance. The two weighting modules are then combined to generate positive and negative weights to adjust each location's confidence. Extensive experiments on the MS COCO show that our method steadily surpasses other best sampling strategies by large margins with various backbones. Moreover, our best model achieves 52.1% AP, outperforming all existing one-stage detectors. Besides, experiments on other datasets,* e.g.*, PASCAL VOC, Objects365, and WiderFace, demonstrate the broad applicability of AutoAssign.*

## 1. Introduction

Current state-of-the-art CNN based object detectors perform a common paradigm of dense prediction. Both two-stage (the RPN [16] part) and one-stage detectors [10, 19, 25, 24] predict objects with various scales, aspect ratios, and classes over every CNN feature locations in a regular, dense sampling manner. This dense detection task raises an essential issue of sampling positives and negatives in the spatial locations, which we call *label assignment*. Moreover, as the modern CNN-based detectors commonly adopt multi-scale features (*e.g.*, FPN [9]) to alleviate scale variance, label assignment requires not only selecting locations among spatial feature maps (*spatial assignment*) but also choosing the level of features with appropriate scale (*scale assignment*).

As shown in Fig. 1, existing detectors mainly sample the positive and negative locations by human prior knowledge: (1) Anchor-based detectors like RetinaNet [10] preset several anchors of multiple scales and aspect ratios on each location and resort to the Intersection over Union (IoU) for sampling positives and negatives among spatial and scale-level feature maps. (2) Anchor-free detectors like FCOS [19] sample a fixed fraction of center area as positive spatial locations for each object, and select certain stages of FPN [9] by the pre-defined scale constraints. These detectors follow the *center prior* (objects are more likely to located around the center of their bounding box) in data distributions to design their assignment strategies, which are proved to be effective on benchmarks like Pascal VOC [2, 3] and MS COCO [11]. However, appearances of objects vary a lot across *categories* and *scenarios*. The above fixed center sampling strategy may pick locations outside objects (*e.g.*, bananas, umbrellas) as positives, thus cannot cover the diverse distributions of categories.

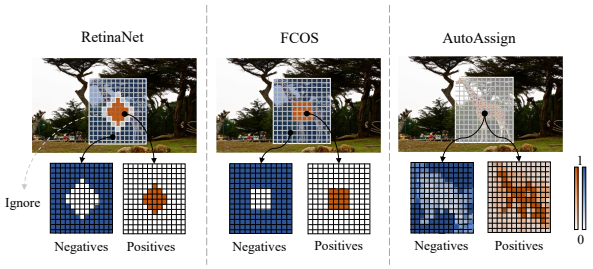


Figure 1. Illustration of different label assignment strategies. Compared to fixed label assignment strategies like RetinaNet and FCOS, AutoAssign do not rely on preset samples and can adapt to object appearance automatically. For better visualization, we stack locations across multiple scales to show the final results.

To deal with the diverse data distributions, a few recent works introduce some partially dynamic strategies in label assignment. GuidedAnchoring [20] and MetaAnchor [22] dynamically change the prior of anchor shapes before sampling, while other methods adaptively modify the sampling strategy for each object in the spatial dimension [25, 24, 8] or the scale dimension [27]. The success of these partially dynamic methods demonstrates great potential in making label assignment more adaptive. However, these strategies can only free part of the label assignment to be data-driven.

The other parts stay constrained by human designs, preventing label assignment to be further optimized.

Intuitively, sampling locations on objects is better than background because they are prone to generate higher quality proposals. Motivated by this, we present AutoAssign, which makes label assignment fully data-dependent and appearance-aware. By dropping the many human knowledge (*e.g.*, anchors, IoU thresholds, and top-$k$) and proposing a unified weighting mechanism across spatial and scales, we reach a fully differentiable strategy.

We adopt a similar paradigm of anchor-free detectors like FCOS [19] to predict one object proposal at each location directly. Given an object, we initially treat all the locations across FPN scales inside its bounding box as *both* positive and negative candidates for further optimization. To adapt to the data distribution of different categories, we propose a category-wise Center Weighting module to learn each category's distribution. To get adapted to each instance's appearance and scale, we propose a Confidence Weighting module to modify the positive and negative confidences of the locations in both spatial and scale dimensions. The two weighting modules are combined to generate positive and negative weight maps for all locations inside an object. According to Fig. 1, the assignment results can dynamically adapt to object appearances. The entire process of weighting is differentiable and can be conveniently optimized by back-propagation during training. All of the weighting modules are only used during loss calculation; thus, AutoAssign is inference cost-free. Moreover, the proposed method only requires the *center prior* knowledge, saving a lot of effort in hyper-parameters tuning, thus can accommodate other data distributions conveniently without any modification.

In summary, the contributions of this study are three-fold as follows:

1. An appearance-aware and fully differentiable weighting mechanism for label assignment is proposed. It enables spatial and scale assignment to be optimized in a unified manner.

2. Two weighting modules (*i.e.*, Center Weighting and Confidence Weighting) are proposed to adjust the category-specific prior distribution and the instance-specific sampling strategy in both spatial and scale dimensions.

3. AutoAssign achieves state-of-the-art performance on the challenging MS COCO [11] dataset. Competitive results on datasets from different distributions, such as PASCAL VOC [2, 3], Object365 [18] and WiderFace [21] demonstrate the effectiveness and broad applicability of AutoAssign.

## 2. Related Work

**Fixed Label assignment** Classical object detectors sample positives and negatives with pre-defined strategies. The RPN in Faster R-CNN [16] preset anchors of different scales and aspect ratios at each location. Given an instance, assignments in both scale and spatial dimensions are guided by the anchor matching IoU. This anchor-based strategy quickly dominates modern detectors and extends to multi-scale outputs (*e.g.*, YOLO [14, 15], SSD [12], and RetinaNet [10]). Recently, attention has been geared toward anchor-free detectors. FCOS [19] and its precursors [6, 23, 13] drop the prior anchor settings and directly assign the spatial positions around bounding box center of each object as positives. In scale dimension, they pre-define scale ranges of different FPN [9] stages to assign instances of different sizes. Both the anchor-based and anchor-free strategies follow the *center prior* inherent in data distributions . However, all of these methods only depend on human knowledge to solve spatial and scale assignment separately and cannot adapt to instance appearances.

**Dynamic Label assignment** Recent detectors propose adaptive mechanisms to improve label assignment. GuidedAnchoring [20] leverages semantic features to guide the anchor settings and dynamically change the shape of anchors to fit various distributions of objects. MetaAnchor [22] randomly samples anchors of any shapes during training to cover different kinds of object boxes. Besides the modification of anchor prior, some works directly change the sampling for each object. FSAF [27] dynamically assigns each instance to the most suitable FPN feature level with minimal training loss. SAPD [26] re-weights the positive anchors and applies an extra meta-net to select the proper FPN stages. FreeAnchor [25] constructs a bag of top-$k$ anchor candidates based on IoU for every object and uses a Mean-Max function to weight among selected anchors, and NoisyAnchor [8] designs another weighting function to eliminate noisy anchors. ATSS [24] proposes an adaptive training sample selection mechanism by the dynamic IoU threshold according to the statistical characteristics of instances. Concurrent work PAA [7] adaptively separates anchors into positive and negative samples in a probabilistic manner. However, they still rely on hand-crafted anchors, thresholds, or other human knowledge for guiding the assignment, which could prevent label assignment from being further optimized.

## 3. Methodology

Before starting, we need to ask: *which part of label assignment is essential?* To answer the question, we present existing label assignment strategies from a more holistic perspective in Table 1. We organize the components of

| Method | Prior | Instance | | AP |
|---|---|---|---|---|
| | | scale | spatial | |
| RetinaNet [10] | anchor | size & IoU | IoU | 36.3 |
| FreeAnchor [25] | anchor | size & IoU | top-$k$ weighting, IoU | 38.7 |
| ATSS [24] | anchor | size & IoU | top-$k$, dynamic IoU | 39.3 |
| GuidedAnchoring [20] | dynamic anchor | size & IoU | IoU | 37.1 |
| FCOS* [19] | center | range | radius | 38.7 |
| FSAF [27] | anchor & center | loss | IoU & radius | 37.2 |
| **AutoAssign (Ours)** | **Center Weighting** | **Confidence Weighting** | | **40.5** |

Table 1. Comparison of label assignment between different typical detectors. Results in terms of $AP$ (%) are reported on the MS COCO 2017 *val* set, using ResNet-50 [5] as backbone. * denotes improved versions.

some representative methods as *prior*-related and *instance*-related. Clearly, apart from the heuristic-based methods like RetinaNet [10] and FCOS [19], all the existing dynamic strategies benefit from its dynamic parts. But they only make partial components of label assignment data-driven, and the other components still rely on hand-crafted rules. We can conclude that: (1) All of the existing detectors obey the *center prior*. Sampling locations near box centers is effective. (2) Both spatial and scale assignments need to be tackled. But existing methods all solve the scale and spatial assignments using two different strategies.

Motivated by these observations, our aim becomes making both the *prior*-related and *instance*-related components adapt to the category or instance characteristics. In this section, we will first give an overall picture of AutoAssign, then demonstrate how the *prior*- and *instance*-level tasks are solved.

### 3.1. Overview

As shown in Fig. 2, the upper gray box shows network architecture. We first follow the anchor-free manner like FCOS [19] to drop the pre-designed anchors and directly predict objects on each feature location. The network has three outputs: classification score, *Implicit-Objectness (ImpObj)* score (which will be described later), and localization offsets. During training (the bottom green box), we first convert all the network predictions into a joint confidence indicator. On top of this, we propose a weighting mechanism, which consists of a Center Weighting module and a Confidence Weighting module. The Center Weighting module is designed to both satisfy the inherent *center prior* property in data and adapt to each category's specific shape pattern. It starts from the standard *center prior* and then learns the distribution of each category from data. The Confidence Weighting module is for assigning the most appropriate locations of each instance based on its appearance and scale adaptively. For each location $i$ in a ground-truth (gt) box, The two modules are combined together to generate positive and negative weights. Finally, positive and negative classification loss will be calculated, and label assignment will be optimized jointly with the network.

From the label assignment perspective, given an object, AutoAssign can automatically find both its appropriate scales across FPN levels and spatial locations based on the network outputs. As a result, the task of label assignment is solved properly in a unified, appearance-aware, and differentiable manner.

### 3.2. Prior-level: Center Weighting

The prior distribution is a fundamental element for label assignment, especially in the early stage of training. In general, the distribution of objects is subject to the *center prior*. However, the objects from different categories, *e.g.*, giraffe, and human, may have distinct distributions. Keeping sampling center positions cannot capture the diverse distributions of different categories. Preferably, adaptive center distributions for different categories are more desired.

Starting from the *center prior*, we introduce a category-wise Gaussian-shape weighting function $G$ with learnable parameters. This Center Weighting module guarantees that locations closer to bounding box center have higher location weights than locations far from box center. Moreover, it can automatically adjust its shape according to data distributions of different categories. Here we define $G$ as:

$$G(\vec{d} \mid \vec{\mu}, \vec{\sigma}) = e^{\frac{-(\vec{d}-\vec{\mu})^2}{2\vec{\sigma}^2}}, \tag{1}$$

where $\vec{d}$ denotes the offsets of a certain position inside an object to its box center along $x$- and $y$-axis, which means it can be negative. $\vec{\mu}$ and $\vec{\sigma}$ are learnable parameters of shape $(K, 2)$. $K$ is the number of categories of a dataset. Each category has two parameters along spatial dimension. As $G$ contributes to the training loss, the parameters can be optimized by back-propagation. At the beginning, $\vec{\mu}$ is initialized to 0 and $\vec{\sigma}$ to 1. Intuitively, $\vec{\mu}$ controls center offset of each category from the box center. And $\vec{\sigma}$ measures each location's importance based on category characteristics. As

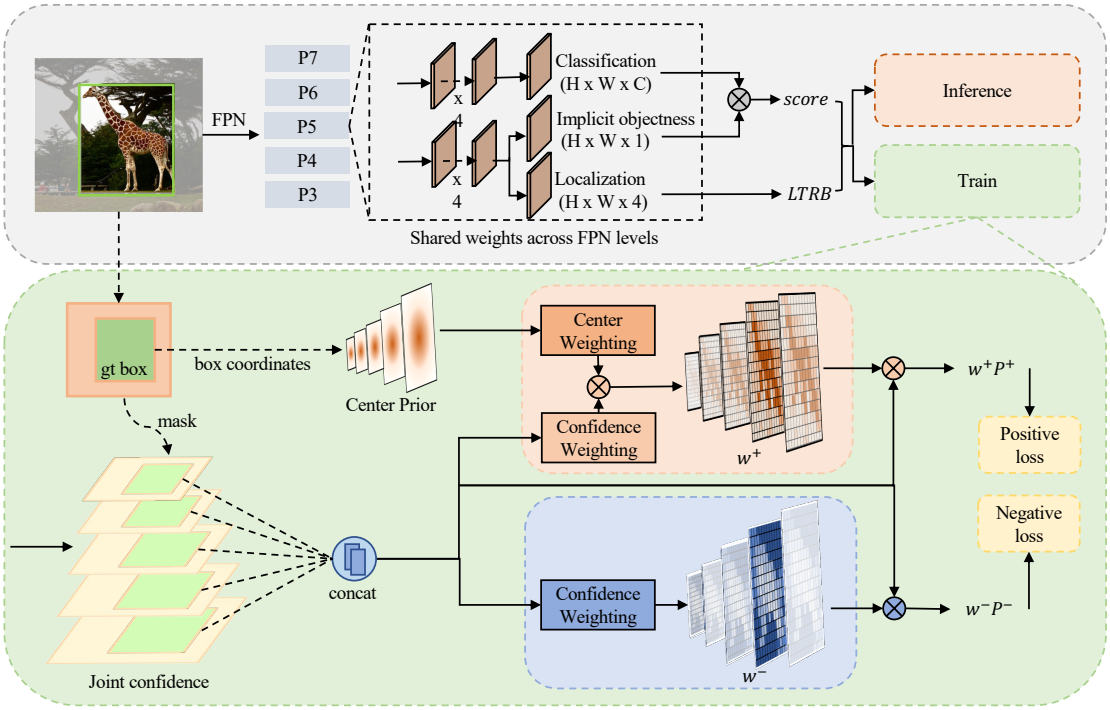


Figure 2. Illustration of AutoAssign. The upper block shows network architecture. The product of classification and *ImpObj* is used as final classification confidence. LTRB means the localization offsets are in *left-top-right-bottom* format. The bottom block presents the label assignment strategy. Given an object, its box coordinates are used for calculating the initial *center prior* and generating foreground masks to select inbox locations. The indexed locations will be flattened and concatenated together. For positive candidates, both Confidence Weighting and Center Weighting are used. For negative candidates, only Confidence Weighting is applied. As a result, positive and negative weight maps are generated. In this process, both spatial and scale assignments are finished jointly.

shown in Fig. 2, the bounding box will generate a location weight map as demonstrated in "Center Prior".

Given an object, we calculate the location weights using $G$ on every FPN stage individually, then stack the weighting results together for later usage. Furthermore, to mitigate the interference caused by the different scales of FPN, we normalize the distance $\vec{d}$ by its downscale ratio.

### 3.3. Instance-level: Confidence Weighting

As mentioned above, all locations inside a bounding box across FPN stages will be considered as both positive and negative sample candidates at the beginning. This operation will significantly increase the background locations in positive candidates and vice versa. This is quite different from all existing label assignment strategies, which only sample a subset of locations as positives before loss calculation. On the other hand, given a location inside a bounding box, to obtain a reasonable weight, all aspects, including classification and regression, need to be taken into account.

Motivated by these aspects, in Confidence Weighting, we propose a joint confidence indicator of both classification and localization to guide the weighting strategy in both spatial and scale dimensions.

**Classification confidence.** Generally speaking, selected positive samples of typical detectors imply that these locations have high confidence of containing instances. However, in our setting, the initial positives set tends to contain a considerable part of background locations, as an object can hardly fill its bounding box completely. Consequently, if a location is, in fact, background, all class predictions in the location should be unreasonable. So taking too many inferior background locations as positives will damage detection performance, which is also the case for the negatives set. To suppress noisy candidates (*i.e.*, backgrounds in positives set, foregrounds in negatives set) from the inferior locations, we introduce a novel *Implicit-Objectness (ImpObj)* branch, which is shown in Fig. 2. The form of *ImpObj* is just like the *center-ness* in FCOS, but here we meet another issue of lacking explicit supervisions. Considering the aim that we need to find and emphasize proper positives and filter out noise candidates dynamically, we optimize the *ImpObj* together with the classification branch. Specifically, we use the product of *ImpObj* and classification score as our rectified classification confidence. *ImpObj* thus shares supervision with the classification branch and does not require explicit labels.

**Joint confidence indicator.** For generating unbiased estimation of each location towards positives/negatives, we should include the localization confidence besides classification. The typical outputs of localization are box offsets, which are hard to measure the regression confidence directly. Considering the fact that Binary Cross-Entropy (BCE) loss is commonly adopted for classification task, we thus convert the localization loss $\mathcal{L}_i^{loc}$ into likelihood:

$$\mathcal{P}_i(loc) = e^{-\lambda \mathcal{L}_i^{loc}}, \tag{2}$$

for being combined with classification confidence conviently, in which $\lambda$ is a hyper-parameter to balance between classification and localization. GIoU loss [17] is used as $\mathcal{L}_i^{loc}$. Then we combine classification and regression likelihood together to get the joint confidence $\mathcal{P}_i$. For the positive candidates, we define positive confidence $\mathcal{P}_i^+ = \mathcal{P}_i(cls) \cdot \mathcal{P}_i(loc)$, where classification confidence $\mathcal{P}_i(cls)$ is the product of classification score and *ImpObj* score. For a location candidate in negatives set, considering the fact that only classification task will be performed on negative locations, thus the negative confidence $\mathcal{P}_i^- = \mathcal{P}_i(cls)$, which is the same as locations outside bounding boxes. Therefore, all background locations can be tackled uniformly.

**Positive weights.** If a location has higher confidence towards positive samples, we prefer to take it as a foreground. Based on the joint confidence representation $\mathcal{P}_i^+$, we thus propose our confidence weighting function $C(\mathcal{P}_i^+)$ in an exponential form to emphasize the locations with high confidence containing objects as:

$$C(\mathcal{P}_i^+) = e^{\mathcal{P}_i^+ / \tau}, \tag{3}$$

where $\tau$ is a hyper-parameter to control the contributions of high and low confidence locations towards positive losses. Intuitively, given an object $i$, for all locations inside its bounding box, we should focus on the proper locations with more accurate predictions. However, at the start of the training process, the network parameters are randomly initialized, making its predicted confidences unreasonable. Thus guiding information from prior is also critical. For location $i \in S_n$, where $S_n$ denotes all locations inside the bounding box at all the scale levels of object $n$, we combine the category-specific prior $G(\vec{d}_i)$ from center weighting module and the confidence weighting module $C(\mathcal{P}_i^+)$ together to generate the positive weights $w_i^+$ as:

$$w_i^+ = \frac{C(\mathcal{P}_i^+) G(\vec{d}_i)}{\sum_{j \in S_n} C(\mathcal{P}_j^+) G(\vec{d}_j)}, \tag{4}$$

here for an object $n$, each $w_i^+$ is normalized by sum of location candidates in $S_n$ for the purpose of being used as valid weights.

**Negative weights.** As discussed above, a bounding box usually contains an amount of real-background locations, and we also need weighted negative losses to suppress these locations and eliminate false positives. Moreover, as the locations inside the boxes always tend to predict high confidence of positives, we prefer the localization confidence to generate the unbiased indicator of false positives. Paradoxically, the negative confidence $\mathcal{P}^-$ has no gradient for the regression task, which means the localization confidence $\mathcal{P}_i(loc)$ should not be optimized by negative loss. Hence we use IoUs between each position's predicted proposal and all objects to generate our negative weights $w_i^-$ as:

$$w_i^- = 1 - f(\text{iou}_i), \tag{5}$$

in which $f(\text{iou}_i) = 1/(1 - \text{iou}_i)$, $\text{iou}_i$ denotes max IoU between proposal of location $i \in S_n$ and all ground truth boxes. To be used as valid weights, we normalize $f(\text{iou}_i)$ into range $[0, 1]$ by its value range. This transformation sharpens the weight distributions and ensure that the location with highest IoU receives zero negative loss. For all locations outside bounding boxes, $w_i^-$ is set to 1 because they are backgrounds for sure.

### 3.4. Loss function

By generating positive and negative weight maps, we achieve the purpose of dynamically assigning more appropriate spatial locations and automatically selecting the proper FPN stages for each instance. As the weight maps contribute to the training loss, AutoAssign tackles the label assignment in a differentiable manner. The final loss function $\mathcal{L}$ of AutoAssign is defined as follows:

$$\mathcal{L} = -\sum_{n=1}^{N} \log\Big(\sum_{i \in S_n} w_i^+ \mathcal{P}_i^+\Big) - \sum_{k \in S} \log(1 - w_k^- \mathcal{P}_k^-), \tag{6}$$

$S$ denotes all the locations at all the scales on the output feature maps. To ensure at least one location matches object $n$, we use the weighted sum of all positive weights to get the final positive confidence. Thus for a location inside bounding boxes, both positive and negative loss will be calculated with different weights. The positive loss and negative loss are calculated independently. Thus the magnitude of positive and negative weights requires no extra operation. To handle the severe imbalance problem in negative samples, the Focal Loss [10] is applied to the negative loss in Eq. 6.

## 4. Experiments

Experiments are mainly evaluated on the MS COCO 2017 [11] benchmark, which contains around 118k images in the *train* set, 5k in the *val* set and 20k in the *test-dev* set. We report analysis and ablation studies on the *val* set and compare with other methods on the *test-dev* set.

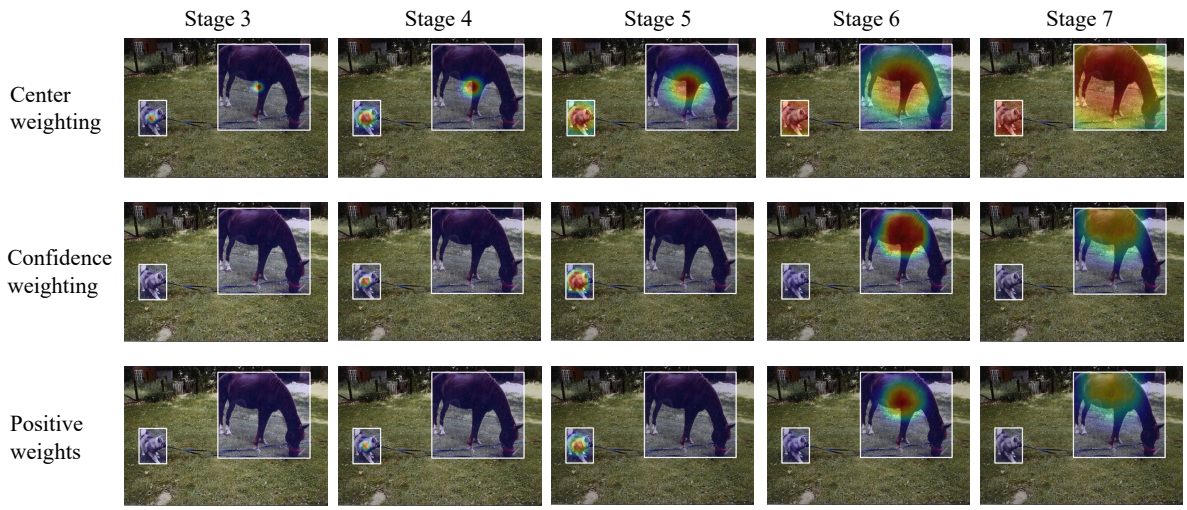


Figure 3. Visualization of center weighting, confidence weighting, and positive weights. From the 3rd row, objects of different shapes and sizes are assigned to its appropriate spatial locations and suitable scale stages automatically.

### 4.1. Implementation Details

We use ResNet-50 [5] with FPN [9] as backbone for all experiments if not specifically pointed out. We initialize the backbone with weights pre-trained on ImageNet [1]. Following common practice, all models are trained for $1\times$ schedule named in [4], i.e., 90k iterations with an initial learning rate of 0.01, which is then divided by 10 at 60k and 80k iterations, with the weight decay of 0.0001 and the momentum of 0.9. Random horizontal flipping is used in data augmentation. For all ablations, we use an image scale of 800 pixels for training and testing, unless otherwise specified. We set $\tau = 1/3$ in Eq. 3, and $\lambda = 5.0$ in $\mathcal{P}(loc)$. Focal Loss with $\alpha = 0.25$ and $\gamma = 2.0$ is applied for negative classification. NMS with IoU threshold 0.6 is applied to merge the results.

### 4.2. Ablation Studies

**Baseline.** None of existing label assignment strategies can be used as baseline of our AutoAssign, because we only rely on the *center prior*, and do not require any other human knowledge like anchors, IoU thresholds, and top-$k$, which is indispensable for many other detectors. As a result, we build AutoAssign from a very simple and clean start point in Table 2. The 17.7 mAP baseline can be seen as removing $w^+$ and $w^-$ from Eq. 6. Other detectors, like RetinaNet, can also be implemented by adding modules to this simple baseline.

**Overall weighting mechanism.** To demonstrate the effectiveness of the two weighting modules, we construct the positives weights $w_i^+$ using only Center Weighting or Confidence Weighting separately in Table 2, while keeping the negatives weighting unchanged. Center Weighting brings relatively significant performance gain, suggesting that the prior distribution is critical for guiding the training. Besides, confidence weighting further improves the accuracy as it dynamically changes the strategy for each object in both spatial and scale dimensions according to object appearances. More design choices of the two modules can be found in Supplementary Materials.

| Center | Conf | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| | | 17.7 | 30.9 | 18.1 | 15.7 | 24.2 | 23.3 |
| | ✓ | 21.5 | 35.8 | 22.6 | 16.6 | 28.9 | 36.0 |
| ✓ | | 37.7 | 57.4 | 40.6 | 20.3 | 41.4 | 52.0 |
| ✓ | ✓ | **40.5** | **59.8** | **43.9** | **23.1** | **44.7** | **52.9** |

Table 2. Effectiveness of Center Weighting and Confidence Weighting. "Center" means center weighting, and "Conf" indicates confidence weighting.

To better understand how spatial and scale assignment is solved through the weighting mechanism, we visualize the positive weight maps separately in each FPN stage from a well-trained detector. From Fig. 3, the Center Weighting is applied to all FPN stages to achieve a coarse weighting based on the category-specific *center prior*. Then the Confidence Weighting generates weights according to object appearances. The two modules perform spatial and scale assignments of each instance jointly.

**Center Weighting.** To analyze the design of the center weighting, we compare different prior distributions in Table 3. We denote the Gaussian-shape function $G$ without learnable parameters as "fixed", while "shared" means all categories share one group of learnable $\vec{\mu}$ and $\vec{\sigma}$. Compared

motorcycle parking meter bear surfboard hotdog

Figure 4. Visualization of learned center weighting weights of different categories. All of the objects are visualized on the same scale. Center weighting results of motorcycle and surfboard show the prior distribution become ellipses (controlled by $\sigma$) to accommodate the shape characteristics of these categories. Center offsets (controlled by $\mu$) are actually larger than 10 pixels in the raw image, which means it could shift one or more grids on output feature maps. Images are best viewed in color.

to the fixed prior, "shared" prior slightly drops the AP by 0.1%, while our category-wise prior increases the AP by 0.2% on MS COCO. As MS COCO contains 80 categories with a huge amount of data, its object distribution generally falls into a normal distribution. Thus the total improvement of category-wise prior is not significant. But when we look at some classes with unique distributions, *e.g.*, surfboard, and hotdog, the improvements are notable.

| Center | $AP$ | moto | prk-mtr | bear | surfboard | hotdog |
|---|---|---|---|---|---|---|
| none | 21.5 | 15.2 | 14.9 | 66.3 | 7.9 | 11.5 |
| fixed | 40.3 | 42.2 | 41.9 | 71.9 | 32.4 | 33.5 |
| shared | 40.2 | 41.8 | 40.7 | 69.2 | 33.1 | 32.7 |
| **category** | **40.5** | **42.9** | **43.3** | **73.6** | **34.8** | **35.8** |

Table 3. Results of different center weighting choices over the whole categories of MS COCO and the subset. "moto" means motorcycle, "prk-mtr" means parking meter.

This can also be evidenced by the visualization of the learned priors for each category in Fig. 4. We mark white points as the center of bounding boxes and red points as the center of learned priors. We can see that in the categories of parking meter and hotdog, the learned centers $\vec{\mu}$ shift down as these categories tend to have more essential clues at the bottom half. Moreover, the category-specific $\vec{\sigma}$ is also changed for each category. For the categories of motorcycle and surfboard, the prior becomes ellipses to accommodate the shape characteristics of these categories.

**Confidence Weighting** We evaluate the effectiveness of classification confidence $\mathcal{P}(cls)$, localization confidence $\mathcal{P}(loc)$, and *ImpObj* separately in Table 4. In the first two rows, we respectively use classification confidence $\mathcal{P}(cls)$ and localization confidence $\mathcal{P}(loc)$ alone in the confidence weighting. The combination of the two confidences (AutoAssign) achieves higher performance, indicating that the joint confidence indicator is the preferable choice when evaluating a location's quality.

In the next two rows, we evaluate the contribution of *ImpObj*. "explicit-obj" means that we explicitly supervise the *objectness* branch with consistent labels (*i.e.*, 1 for foregrounds and 0 for backgrounds) for all the locations inside the boxes. We find that simply using hard labels for the *objectness* has no help to performance, while our *ImpObj* can significantly boost the performance by $\sim 1\%$ AP. Moreover, the performance of objects at all sizes can obtain obvious performance gains. We think the contribution of *ImpObj* comes from its effect on both filtering out the noise candidates and achieving better separation from the background. Visualizations can be found in Supplementary Materials.

| Confidence | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|
| $\mathcal{P}(cls)$-only | 38.7 | **59.9** | 41.6 | 22.9 | 42.0 | 49.5 |
| $\mathcal{P}(loc)$-only | 39.7 | 58.4 | 43.1 | 22.4 | 43.6 | 51.6 |
| no-obj | 39.4 | 58.7 | 42.5 | 22.4 | 43.5 | 50.7 |
| explicit-obj | 39.5 | 58.8 | 42.3 | 21.6 | 43.4 | 52.2 |
| AutoAssign | **40.5** | 59.8 | **43.9** | **23.1** | **44.7** | **52.9** |

Table 4. Comparison of different choices for confidence weighting. $\mathcal{P}(cls)$-only means only use $\mathcal{P}(cls)$ for confidence weighting. "no-obj" means do not use *ImpObj* for $\mathcal{P}(cls)$. "explicit-obj" means give the object-ness branch individual supervision, rather than sharing with classification.

### 4.3. Comparison with State-of-the-art

We compare AutoAssign with other detectors on MS COCO *test-dev* set. We adopt $2\times$ schedule following the previous works [19, 25, 24]. Results are shown in Table 5. Under the same training setting, AutoAssign can consistently outperform other counterparts. For example, AutoAssign with ResNet-101 backbone achieves 44.5% AP, and our best model achieves 52.1% AP, which outperforms all existing one-stage detectors.

### 4.4. Generalization

Another benefit of using little human knowledge it that huge effort on hyper-parameters tuning when transfer to other datasets. To demonstrate the generalization ability,

| Method | Iteration | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| **ResNet-101** | | | | | | | |
| RetinaNet [10] | 135k | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| FCOS [10] | 180k | 41.5 | 60.7 | 45.0 | 24.4 | 44.8 | 51.6 |
| FreeAnchor [25] | 180k | 43.1 | 62.2 | 46.4 | 24.5 | 46.1 | 54.8 |
| SAPD [26] | 180k | 43.5 | 63.6 | 46.5 | 24.9 | 46.8 | 54.6 |
| ATSS [24] | 180k | 43.6 | 62.1 | 47.4 | **26.1** | 47.0 | 53.6 |
| AutoAssign (Ours) | 180k | **44.5** | **64.3** | **48.4** | 25.9 | **47.4** | **55.0** |
| **ResNeXt-64x4d-101** | | | | | | | |
| FCOS* [19] | 180k | 44.7 | 64.1 | 48.4 | 27.6 | 47.5 | 55.6 |
| FreeAnchor [25] | 180k | 44.9 | 64.3 | 48.5 | 26.8 | 48.3 | 55.9 |
| SAPD [26] | 180k | 45.4 | 65.6 | 48.9 | 27.3 | 48.7 | **56.8** |
| ATSS [24] | 180k | 45.6 | 64.6 | 49.7 | **28.5** | 48.9 | 55.6 |
| AutoAssign (Ours) | 180k | **46.5** | **66.5** | **50.7** | 28.3 | **49.7** | 56.6 |
| **ResNeXt-64x4d-101-DCN** | | | | | | | |
| SAPD [26] | 180k | 47.4 | 67.4 | 51.1 | 28.1 | 50.3 | 61.5 |
| ATSS [24] | 180k | 47.7 | 66.5 | 51.9 | 29.7 | 50.8 | 59.4 |
| AutoAssign (Ours) | 180k | 48.3 | 67.4 | 52.7 | 29.2 | 51.0 | 60.3 |
| AutoAssign (Ours)† | 180k | 49.5 | 68.7 | 54.0 | 29.9 | 52.6 | 62.0 |
| AutoAssign (Ours)†‡ | 180k | **52.1** | **69.6** | **58.0** | **33.9** | **54.0** | **64.0** |

Table 5. Performance comparison with state-of-the-art one-stage detectors on MS COCO 2017 *test-dev* set. All results listed adopt multi-scale training. † indicates multi-scale training with wider range $[480, 960]$ used in [25]. ‡ indicates multi-scale testing. * indicates improved versions.

| Method | PASCAL VOC | | | Objects365 | | | WiderFace | | |
|---|---|---|---|---|---|---|---|---|---|
| | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP$ | $AP_{50}$ | $AP_{75}$ |
| RetinaNet [10] | 55.4 | 81.0 | 60.1 | 18.4 | 28.4 | 19.6 | 46.7 | 83.7 | 47.1 |
| FCOS* [19] | 55.4 | 80.5 | 61.1 | 20.3 | 29.9 | 21.9 | 48.1 | 87.1 | 48.4 |
| FreeAnchor [25] | 56.8 | 81.1 | 62.1 | 21.4 | 31.5 | 22.8 | 46.3 | 81.6 | 47.5 |
| ATSS [24] | 56.6 | 80.7 | 62.6 | 20.7 | 30.0 | 22.4 | 48.9 | 87.1 | 49.7 |
| AutoAssign (Ours) | **57.9** | **81.6** | **64.1** | **21.6** | **31.7** | **23.2** | **49.5** | **88.2** | **49.9** |

Table 6. Performance comparison with typical detectors on PASCAL VOC, Objects365 and WiderFace. * indicates improved versions.

we evaluate AutoAssign and several other detectors on different data distributions, including general object detection (PASCAL VOC [2, 3], Objects365 [18]) and face detection (WiderFace [21]). In these experiments, we keep all the hyper-parameters unchanged and only adjust the training settings following the common paradigm of each dataset.

Results are shown in Table 6. We find that the performance of other methods with fixed or partly fixed assigning strategies are unstable on different datasets. Although they may achieve excellent performance on certain datasets, their accuracies on the other dataset may be worse. This proves that the label assignment strategy of these methods has low robustness, thus needs to be adjusted cautiously. In contrast, AutoAssign can automatically adapt to different data distributions and achieve superior performance without any adjustment.

## 5. Conclusions

In this paper, we propose a differentiable label assignment strategy named AutoAssign. It tackles label assignment in a fully data-driven manner by automatically determine the positives/negatives in both spatial and scale dimensions. It achieves consistent improvement to all the existing sampling strategies by $\sim$1% AP with various backbones on MS COCO. Besides, extensive experiments on other datasets demonstrate that AutoAssign can conveniently transfer to other datasets and tasks without additional modification.

This new label assignment strategy also comes with new challenges. For example, the current weighting mechanism is not simple enough and can be further simplified, which will be left for future work.

## References


[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 6

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 1, 2, 8

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 1, 2, 8

[4] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *The IEEE International Conference on Computer Vision*, 2019. 6

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3, 6

[6] Lichao Huang, Yi Yang, Yafeng Deng, and Yinan Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015. 2

[7] Kang Kim and Hee Seok Lee. Probabilistic anchor assignment with iou prediction for object detection. *arXiv preprint arXiv:2007.08103*, 2020. 2

[8] Hengduo Li, Zuxuan Wu, Chen Zhu, Caiming Xiong, Richard Socher, and Larry S Davis. Learning from noisy anchors for one-stage object detection. *arXiv preprint arXiv:1912.05086*, 2019. 1, 2

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 6

[10] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *The IEEE International Conference on Computer Vision*, 2017. 1, 2, 3, 5, 8, 11

[11] 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 *The European Conference on Computer Vision*, 2014. 1, 2, 5

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *The European Conference on Computer Vision*, 2016. 2

[13] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2

[14] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[15] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. 1, 2

[17] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 5

[18] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *The IEEE International Conference on Computer Vision*, 2019. 2, 8

[19] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *The IEEE International Conference on Computer Vision*, 2019. 1, 2, 3, 7, 8, 11

[20] Jiaqi Wang, Kai Chen, Shuo Yang, Chen Change Loy, and Dahua Lin. Region proposal by guided anchoring. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 3

[21] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2, 8

[22] Tong Yang, Xiangyu Zhang, Zeming Li, Wenqiang Zhang, and Jian Sun. Metaanchor: Learning to detect objects with customized anchors. In *Advances in Neural Information Processing Systems*, 2018. 1, 2

[23] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. In *The ACM International Conference on Multimedia*, 2016. 2

[24] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. *arXiv preprint arXiv:1912.02424*, 2019. 1, 2, 3, 7, 8, 11

[25] Xiaosong Zhang, Fang Wan, Chang Liu, Rongrong Ji, and Qixiang Ye. Freeanchor: Learning to match anchors for visual object detection. In *Advances in Neural Information Processing Systems*, 2019. 1, 2, 3, 7, 8, 11

[26] Chenchen Zhu, Fangyi Chen, Zhiqiang Shen, and Marios Savvides. Soft anchor-point object detection. *arXiv preprint arXiv:1911.12448*, 2019. 2, 8

[27] Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 3

# A. Design Choices of AutoAssign

## A.1. Baseline

To better understand the 17.7 AP Baseline in Table 3 of our AutoAssign, we demonstrate how other methods can be implemented by adding corresponding elements onto it. For example, RetinaNet can be implemented by adding 9 *anchors* to each FPN stage(so there are 9×5 anchors in total), and corresponding positive and negative IoU threshold. FCOS_imprv can be implemented by adding center sampling (CS) radius, artificial FPN scale assignment rules and *center-ness* branch. Here we demonstrate how to add modules one by one to get the final FCOS_imprv in Table 7.

| Method | $AP$ |
|---|---|
| Baseline | 17.4 |
| Baseline + CS | 31.6 |
| Baseline + scale assignment | 34.0 |
| Baseline + CS + scale assignment | 37.7 |
| Baseline + CS + scale assignment + *center-ness* | 38.8 |

Table 7. Build FCOS_imprv on top of our Baseline step by step. The last row corresponds to official FCOS_imprv. CS denots "center sampling".

## A.2. Initialization of Center Weighting

We fix the learnable parameters $\mu$ and $\sigma$ to analyze if the learnable mechanism works and is useful for Center Weighting. We firstly set standard Gaussian parameters and find it works well. Then we change them to quite different parameters and find the performance goes down, which means LR are sensitive to these parameters. After that we release $\mu$ or $\sigma$ respectively, and we find that the performance recovers gradually, which means the learnable mechanism is working. Finally we initialize $\mu$ or $\sigma$ by the aforementioned values but make them learnable, which shows the learnable mechanism is slightly superior and robust to initialization.

| Hyper-parameters | $AP$ | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| fix $\mu = 0, \sigma = 1$ | 40.3 | 59.7 | 43.6 |
| fix $\mu = 1, \sigma = 2$ | 39.9 | 58.9 | 43.2 |
| fix $\mu = 0$ | 40.2 | 59.4 | 43.5 |
| fix $\mu = 1$ | 40.2 | 59.4 | 43.5 |
| fix $\sigma = 1$ | 40.3 | 59.4 | 43.6 |
| fix $\sigma = 2$ | 40.1 | 59.2 | 43.2 |
| init $\mu = 0, \sigma = 1$ (AutoAssign) | **40.5** | **59.8** | **43.9** |
| init $\mu = 1, \sigma = 2$ | 40.4 | 59.6 | 43.6 |

Table 8. The results of AutoAssign when changing $\mu$ and $\sigma$. Bold fonts indicate the best performance

| method | $f(x)$ | $AP$ | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|
| $\exp(x/\tau)$ | $\tau = 1$ | 39.8 | 59.1 | 42.9 |
| $\exp(x/\tau)$ | $\tau = 1/3$ | **40.5** | **59.8** | **43.9** |
| $\exp(x/\tau)$ | $\tau = 1/5$ | 40.0 | 59.4 | 43.5 |

Table 9. Different $\tau$ in the Confidence Weighting functions in AutoAssign. Bold fonts indicate the best performance.

## A.3. $\tau$ in Confidence Weighting

Here we present the choices of Confidence Weighting functions in Table. 9. We can find that neither too flat or steep slopes cann achieve the best performance. We guess that this hyper-parameter is similar to the $\gamma$ in the Focal Loss.

## A.4. Implicit-Objectness.

We propose *Implicit-Objectness* to help us to classify foreground or background. As shown in Table 10, *ImpObj* can boost AP on all detectors, but more significant on AutoAssign. The gain of *ImpObj* in AutoAssign is $\sim$ 1.0 AP, much more than other detectors, shows that there is better interaction between the dynamic weighting mechanism and *ImpObj*. We believe that the manual designed label assignment might conflict with the prediction of *ImpObj* in some cases. For an instance, the center sampling assumes the areas near the center have the highest possibilities to locate on an object, which is unlikely in some cases like a bounding box of a giraffe, but *ImpObj* will possibly tell us the center areas are background. On the other hand, applying $ImpObj$ on a small set of pre-selected positive samples might cause the effectiveness of it cannot be fully explored.

# B. Visualization

To better understand the behavior of *ImpObj*, we compare the final classification score used for NMS in Fig. 5. From visualization and evaluation results, improvements come from both recall and precision. The proposed *implicit-objectness* can filter out the noise and achieve better separation from background. Then we investigate the learning process of confidence weighting in Fig. 6. We only visualize two representative FPN stages for simplicity. In the beginning, confidence weighting is weak for all objects because the probabilities among all locations are similarly low. With the training progress progressing, confidences become more salient and gradually converge to their appropriate FPN stages for objects of different sizes, demonstrating the effectiveness of our learnable strategy.

| Method | *ImpObj* | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| RetinaNet [10] | | 35.9 | 55.9 | 38.2 | 19.8 | 39.5 | 47.9 |
| | ✓ | 36.1 | 56.3 | 38.8 | 20.0 | 39.9 | 48.2 |
| FCOS_imprv [19] | | 38.8 | 57.6 | 42.2 | 22.7 | 42.9 | 50.0 |
| | ✓ | 39.0 | 58.2 | 42.3 | 22.8 | 42.8 | 50.6 |
| FreeAnchor [25] | | 38.4 | 57.1 | 41.2 | 21.2 | 41.6 | 51.5 |
| | ✓ | 38.5 | 57.6 | 41.5 | 20.9 | 41.9 | **53.1** |
| ATSS [24] | | 39.3 | 57.8 | 42.6 | 23.0 | 43.0 | 50.7 |
| | ✓ | 39.7 | 58.4 | 42.9 | 23.0 | 43.6 | 51.6 |
| AutoAssign | | 39.4 | 58.7 | 42.5 | 22.4 | 43.5 | 50.7 |
| | ✓ | **40.5** | **59.8** | **43.9** | **23.1** | **44.7** | 52.9 |

Table 10. The performances of the application of *ImpObj* on different methods. Bold fonts indicate the best performance

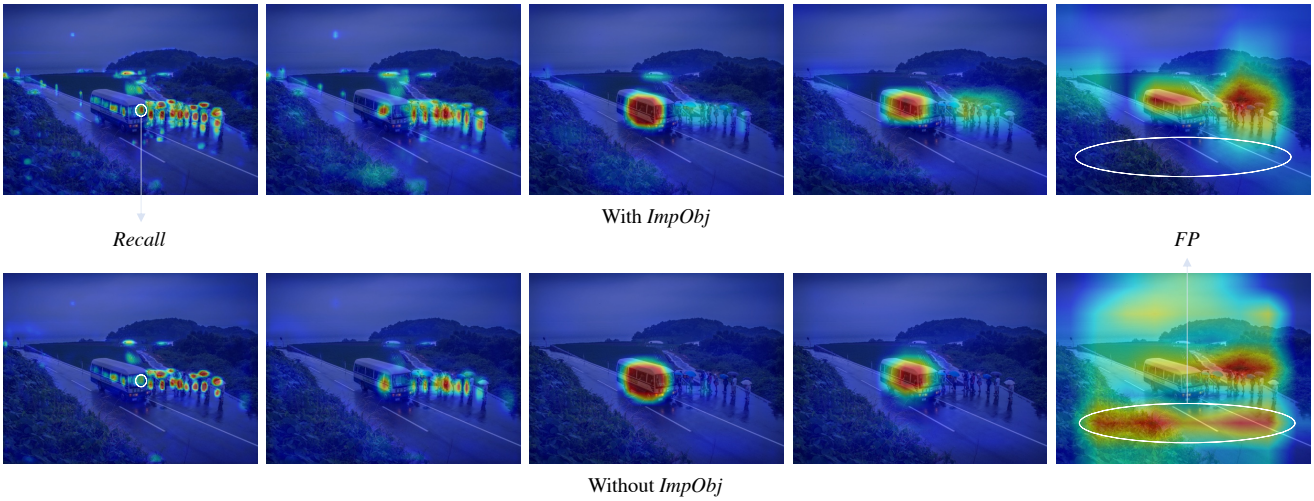


Figure 5. Confidence visualization of detectors trained with/without $ImpObj$. With the help of $ImpObj$, more objects can be detected, and false positives from background locations can be surpressed, thus recall and precision can be improved

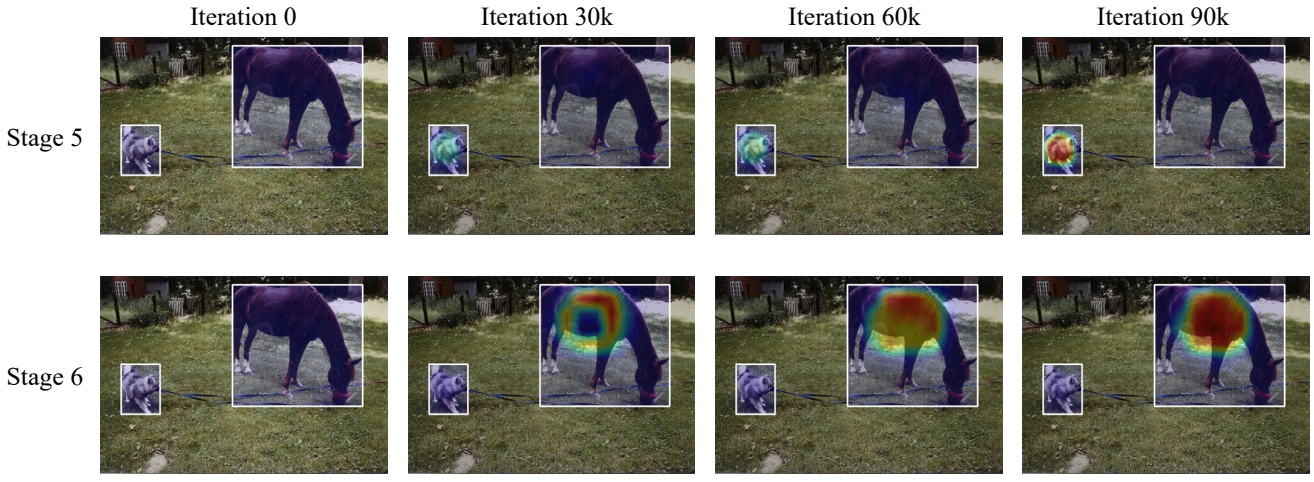


Figure 6. Illustration of confidence weighting evolution from iteration 0 to 90k. We select the most activated stages for different objects