Towards Accurate One-Stage Object Detection with AP-Loss

Kean Chen[1], Jianguo Li[2], Weiyao Lin[1*], John See[3], Ji Wang[4], Lingyu Duan[5], Zhibo Chen[4], Changwei He[4], Junni Zou[1]
[1]Shanghai Jiao Tong University, China, [2] Intel Labs, China,
[3] Multimedia University, Malaysia, [4] Tencent YouTu Lab, China, [5] Peking University, China

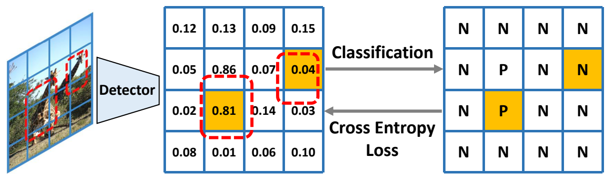
Abstract

One-stage object detectors are trained by optimizing classification-loss and localization-loss simultaneously, with the former suffering much from extreme foreground-background class imbalance issue due to the large number of anchors. This paper alleviates this issue by proposing a novel framework to replace the classification task in one-stage detectors with a ranking task, and adopting the Average-Precision loss (AP-loss) for the ranking problem. Due to its non-differentiability and non-convexity, the AP-loss cannot be optimized directly. For this purpose, we develop a novel optimization algorithm, which seamlessly combines the error-driven update scheme in perceptron learning and backpropagation algorithm in deep networks. We verify good convergence property of the proposed algorithm theoretically and empirically. Experimental results demonstrate notable performance improvement in state-of-the-art one-stage detectors based on AP-loss over different kinds of classification-losses on various benchmarks, without changing the network architectures.

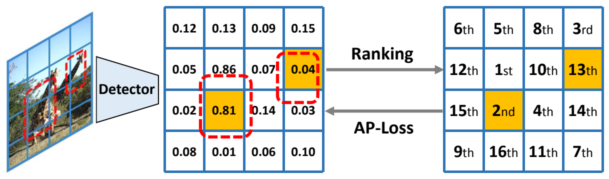
1. Introduction

Object detection needs to localize and recognize the objects simultaneously from the large backgrounds, which remains challenging due to the imbalance between foreground and background. Deep learning based detection solutions usually adopt a multi-task architecture, which handles classification task and localization task with different loss functions. The classification task aims to recognize the object in a given box, while the localization task aims to predict the precise bounding box of the object. Two-stage detectors [25, 7, 2, 14] first generates a limited number of object box proposals, so that the detection problem can be solved by adopting classification task on those proposals. However, the circumstance is different for one-stage detectors, which need to predict the object class directly from the densely pre-designed candidate boxes. The large number

*Corresponding Author, Email: wylin@sjtu.edu.cn



(a) Acc $= 0.88$



(b) AP $= 0.33$

Figure 1: Dashed red boxes are the ground truth object boxes. The orange filled boxes and other blank boxes are anchors with positive and negative ground truth labels, repectively. (a) shows that the detection performance is poor but the classification accuracy is still high due to large number of true negatives. (b) shows the ranking metric AP can better reflect the actual condition as it does not suffer from the large number of true negatives.

of boxes yield the imbalance between foreground and background which makes the optimization of classification task easily biased and thus impacts the detection performance. It is observed that the classification metric could be very high for a trivial solution which predicts negative label for almost all candidate boxes, while the detection performance is poor. Figure 1a illustrates one such example.

To tackle this issue in one-stage object detectors, some works introduce new classification losses such as balanced loss [23], Focal Loss [15], as well as tailored training method such as Online Hard Example Mining (OHEM) [18, 31]. These losses model each sample (anchor box) independently, and attempt to re-weight the foreground and background samples in classification losses to cater for the imbalance condition; this is done without considering the relationship among different samples. The designed balance weights are hand-crafted hyper-parameters, which do not generalize well across datasets. We argue that the gap between classification task and detection task hinder the performance of one-stage detectors. In this paper, instead of modifying the classification loss, we propose to replace

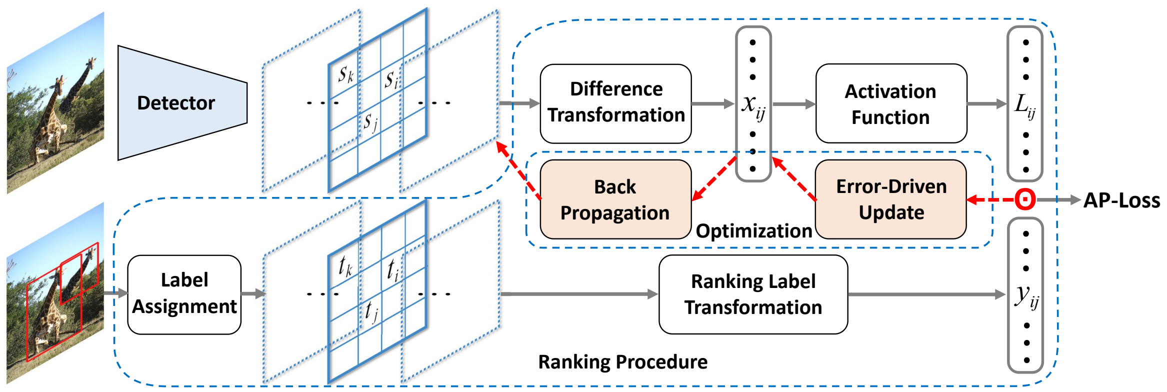


Figure 2: Overall framework of the proposed approach. We replace the classification-task in one-stage detectors with a ranking task, where the ranking procedure produces the primary terms of AP-loss and the corresponding label vector. The optimization algorithm is based on an error-driven learning scheme combined with backpropagation. The localization-task branch is not shown here due to no modification.

classification task with ranking task in one-stage detectors, where the associated ranking loss explicitly models sample relationship, and is invariant to the ratio of positive and negative samples. As shown in Figure 1b, we adopt Average Precision (AP) as our target loss which is inherently more consistent with the evaluation metric for object detection.

However, it is non-trivial to directly optimize the AP-loss due to the non-differentiability and non-decomposability, so that standard gradient descent methods are not amenable for this case. There are three aspects of studies for this issue. *First*, AP based loss is studied within structured SVM models [37, 19], which restricts in linear SVM model so that the performance is limited. *Second*, a structured hinge loss [20] is proposed to optimize the upper bound of AP-loss instead of the loss itself. *Third*, approximate gradient methods [34, 9] are proposed for optimizing the AP-loss, which are less efficient and easy to fall into local optimum even for the case of linear models due to the non-convexity and non-quasiconvexity of the AP-loss. Therefore, it is still an open problem for the optimization of the AP-loss.

In this paper, we address this challenge by replacing the classification task in one-stage detectors with a ranking task, so that we handle the class imbalance problem with a ranking based loss named AP-loss. Furthermore, we propose a novel error-driven learning algorithm to effectively optimize the non-differentiable AP based objective function. More specifically, some extra transformations are added to the score output of one-stage detector to obtain the AP-loss, which includes a linear transformation that transforms the scores to pairwise differences, and a non-linear and non-differentiable "activation function" that transform the pairwise differences to the primary terms of the AP-loss. Then the AP-loss can be obtained by the dot product between the primary terms and the label vector. It is worth noting that the difficulty for using gradient method on the AP-loss lies in passing gradients through the non-differentiable activation function. Inspired by the perceptron learning algorithm [26], we adopt an error-driven learning scheme to directly pass the update signal through the non-differentiable activation function. Different from gradient method, our learning scheme gives each variable an update signal proportional to the error it makes. Then, we adopt the backpropagation algorithm to transfer the update signal to the weights of neural network. We theoretically and experimentally prove that the proposed optimization algorithm does not suffer from the non-differentiability and non-convexity of the objective function. The main contributions of this paper are summarized as below:

- We propose a novel framework in one-stage object detectors which adopts the ranking loss to handle the class imbalance issue.
- We propose an error-driven learning algorithm that can efficiently optimize the non-differentiable and non-convex AP-based objective function with both theoretical and experimental verifications.
- We show notable performance improvement with the proposed method on state-of-the-art one-stage detectors over different kinds of classification-losses without changing the model architecture.

2. Related Work

One-stage detectors: In object detection, the one-stage approaches have relatively simpler architecture and higher efficiency than two-stage approaches. OverFeat [28] is one of the first CNN-based one-stage detectors. Thereafter, different designs of one-stage detectors are proposed, including SSD [18], YOLO [23], DSSD [6] and DSOD [30, 13]. These methods demonstrate good processing efficiency as one-stage detectors, but generally yield lower accuracy than two-stage detectors. Recently, RetinaNet [15] and RefineDet [38] narrow down the performance gap (especially on the challenging COCO benchmark [16]) between one-stage approaches and two-stage approaches with some innovative designs. As commonly known, the performance of one-stage detectors benefits much from densely designed anchors, which introduce extreme imbalance between foreground and background samples. To address this challenge, methods like OHEM [18, 31] and Focal Loss [15] have been proposed to reduce the loss weight for easy samples. However, there are two hurdles that are still open to discussion. Firstly, hand-crafted hyper-parameters for weight balance do not generalize well across datasets. Secondly, the relationship among sample anchors is far from well modeled.

AP as a loss for Object Detection: Average Precision (AP) is widely used as the evaluation metric in many tasks such as object detection [5] and information retrieval [27]. However, AP is far from a good and common choice as an optimization goal in object detection due to its non-differentiability and non-convexity. Some methods have been proposed to optimize the AP-loss in object detection, such as AP-loss in the linear structured SVM model [37, 19], structured hinge loss as upper bound of the AP-loss [20], approximate gradient methods [34, 9], reinforcement learning to fine-tune a pre-trained object detector with AP based metric [22]. Although these methods give valuable results in optimizing the AP-loss, their performances are still limited due to the intrinsic limitations. In details, the proposed approach differs from them in 4 aspects. (1) Our approach can be used for any differentiable linear or non-linear models such as neural networks, while [37, 19] only work for linear SVM model. (2) Our approach directly optimizes the AP-loss, while [20] introduces notable loss gap after relaxation. (3) Our approach dose not approximate the gradient and dose not suffer from the non-convexity of objective function as in [34, 9]. (4) Our approach can train the detectors in an end-to-end way, while [22] cannot.

Perceptron Learning Algorithm: The core of our optimization algorithm is the "error-driven update" which is generalized from the perceptron learning algorithm [26], and helps overcome the difficulty of the non-differentiable objective functions. The perceptron is a simple artificial neuron using the Heaviside step function as the activation function. The learning algorithm was first invented by Frank Rosenblatt [26]. As the Heaviside step function in perceptron is non-differentiable, it is not amenable for gradient method. Instead of using a surrogate loss like cross-entropy, the perceptron learning algorithm employs an error-driven update scheme directly on the weights of neurons. This algorithm is guaranteed to converge in finite steps if the training data is linearly separable. Further works like [11, 1, 35] have studied and improved the stability and robustness of the perceptron learning algorithm.

3. Method

We aim to replace the classification task with AP-loss based ranking task in one-stage detectors such as RetinaNet [15]. Figure 2 shows the two key components of our approach, *i.e.*, the ranking procedure and the error-driven optimization algorithm. Below, we will first present how AP-loss is derived from traditional score output. Then, we will introduce the error-driven optimization algorithm. Finally, we also present the theoretical analyses of the proposed optimization algorithm and outline the training details. Note that all changes are made on the loss part of the classification branch without changing the backbone model and localization branch.

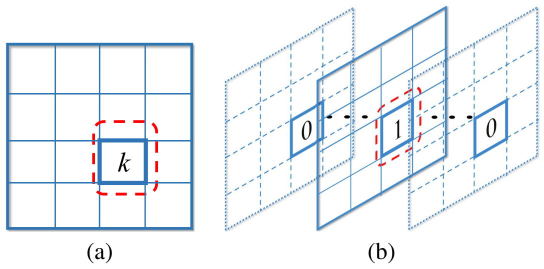


Figure 3: Comparison of label assignments. The dashed red box is the ground truth box with class k. (a) In traditional classification task of one-stage detectors, the anchor is assigned a foreground label k. (b) In our ranking task framework, the anchor replicates K times, and we assign the k-th anchor to label 1, others 0.

3.1. Ranking Task and AP-Loss

3.1.1 Ranking Task

In traditional one-stage detectors, given input image I, suppose the pre-defined boxes (also called anchors) set is B, each box $b_i \in B$ will be assigned a label $t_i \in \{-1, 0, 1, \ldots, K\}$ based on ground truth and the IoU strategy [7, 25], where label $1 \sim K$ means the object class ID, label "0" means background and label "-1" means ignored boxes. During training and testing phase, the detector outputs a score-vector $(s_i^0, \cdots, s_i^K)$ for each box b_i.

In our framework, instead of one box with $K+1$ dimensional score predictions, we replicate each box b_i for K times to obtain b_{ik} where $k = 1, \cdots, K$, and the k-th box is responsible for the k-th class. Each box b_{ik} will be assigned a label $t_{ik} \in \{-1, 0, 1\}$ through the same IoU strategy (label -1 for not counted into the ranking loss). Therefore, in the training and testing phase, the detector will predict only one scalar score s_{ik} for each box b_{ik}. Figure 3 illustrates our label formulation and the difference to traditional case.

The ranking task dictates that every positive boxes should be ranked higher than all the negative boxes w.r.t their scores. Note that AP of our ranking result is computed over the scores from all classes. This is slightly different from the evaluation metric meanAP for object detection systems, which computes AP for each class and obtains the average value. We compute AP this way because the score distribution should be unified for all classes while ranking each class separately cannot achieve this goal.

3.1.2 AP-Loss

For simplicity, we still use B to denote the anchor box set after replication, and b_i to denote the i-th anchor box without the replication subscript. Each box b_i thus corresponds to one scalar score s_i and one binary label t_i. Some transformations are required to formulate a ranking loss as illustrated in Figure 2. First, the *difference transformation* transfers the score s_i to the difference form

$$\forall i,j,\ \ x_{ij} = -(s(b_i;\boldsymbol{\theta}) - s(b_j;\boldsymbol{\theta})) = -(s_i - s_j) \tag{1}$$

where $s(b_i;\boldsymbol{\theta})$ is a CNN based score function with weights $\boldsymbol{\theta}$ for box b_i. The *ranking label transformation* transfers labels t_i to the corresponding pairwise ordering form

$$\forall i,j,\ \ y_{ij} = \mathbf{1}_{t_i=1,t_j=0} \tag{2}$$

where $\mathbf{1}$ is a indicator function which equals to 1 only if the subscript condition holds (*i.e.*, $t_i = 1, t_j = 0$), otherwise 0. Then, we define an vector-valued *activation function* $\boldsymbol{L}(\cdot)$ to produce the primary terms of the AP-loss as

$$L_{ij}(\boldsymbol{x}) = \frac{H(x_{ij})}{1 + \sum_{k\in\mathcal{P}\cup\mathcal{N},k\neq i} H(x_{ik})} = L_{ij} \tag{3}$$

where $H(\cdot)$ is the Heaviside step function:

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \tag{4}$$

A ranking is denoted as *proper ranking* when there are no two samples scored equally (*i.e.*, $\forall i \neq j,\ s_i \neq s_j$). Without loss of generality, we will treat all rankings as a proper ranking by breaking ties arbitrarily. Now, we can formulate the AP-loss $\mathcal{L}_{AP}$ as

$$\begin{aligned}\mathcal{L}_{AP} &= 1 - \text{AP} = 1 - \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \frac{rank^+(i)}{rank(i)} \\ &= 1 - \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \frac{1 + \sum_{j\in\mathcal{P},j\neq i} H(x_{ij})}{1 + \sum_{j\in\mathcal{P},j\neq i} H(x_{ij}) + \sum_{j\in\mathcal{N}} H(x_{ij})} \\ &= \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}} L_{ij} = \frac{1}{|\mathcal{P}|}\sum_{i,j} L_{ij}\cdot y_{ij} = \frac{1}{|\mathcal{P}|}\langle \boldsymbol{L}(\boldsymbol{x}), \boldsymbol{y}\rangle\end{aligned} \tag{5}$$

where $rank(i)$ and $rank^+(i)$ denote the ranking position of score s_i among all valid samples and positive samples respectively, $\mathcal{P} = \{i|t_i = 1\}$, $\mathcal{N} = \{i|t_i = 0\}$, $|\mathcal{P}|$ is the size of set $\mathcal{P}$, $\boldsymbol{L}$ and $\boldsymbol{y}$ are vector form for all L_{ij} and y_{ij} respectively, $\langle,\rangle$ means dot-product of two input vectors. Note that $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{L} \in \mathbb{R}^d$, where $d = (|\mathcal{P}| + |\mathcal{N}|)^2$.

Finally, the optimization problem can be written as:

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{AP}(\boldsymbol{\theta}) = 1 - \text{AP}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{P}|}\langle \boldsymbol{L}(\boldsymbol{x}(\boldsymbol{\theta})), \boldsymbol{y}\rangle \tag{6}$$

where $\boldsymbol{\theta}$ denotes the weights of detector model. As the activation function $\boldsymbol{L}(\cdot)$ is non-differentiable, a novel optimization/learning scheme is required instead of the standard gradient descent method.

Besides the AP metric, other ranking based metric can also be used to design the ranking loss for our framework. One example is the AUC-loss [12] which measures the area under ROC curve for ranking purpose, and has a slightly different "activation function" as

$$L'_{ij}(\boldsymbol{x}) = \frac{H(x_{ij})}{|\mathcal{N}|} \tag{7}$$

As AP is consistent with the evaluation metric of the object detection task, we argue that AP-loss is intuitively more suitable than AUC-loss for this task, and will provide empirical study in our experiments.

3.2. Optimization Algorithm

3.2.1 Error-Driven Update

Recalling the perceptron learning algorithm, the update for input variable is "error-driven", which means the update is directly derived from the difference between desired output and current output. We adopt this idea and further generalize it to accommodate the case of activation function with vector-valued input and output. Suppose x_{ij} is the input and L_{ij} is the current output, the update for x_{ij} is thus

$$\Delta x_{ij} = L^*_{ij} - L_{ij} \tag{8}$$

where L^*_{ij} is the desired output. Note that the AP-loss achieves its minimum possible value 0 when each term $L_{ij}\cdot y_{ij} = 0$. There are two cases. If $y_{ij} = 1$, we should set the desired output $L^*_{ij} = 0$. If $y_{ij} = 0$, we do not care the update and set it to 0, since it does not contribute to the AP-loss. Consequently, the update can be simplified as

$$\Delta x_{ij} = -L_{ij}\cdot y_{ij} \tag{9}$$

3.2.2 Backpropagation

We now have the desired vector-form update $\Delta\boldsymbol{x}$, and then will find an update for model weights $\Delta\boldsymbol{\theta}$ which will produce most appropriate movement for $\boldsymbol{x}$. We use dot-product to measure the similarity of successive movements, and regularize the change of weights (*i.e.* $\Delta\boldsymbol{\theta}$) with L_2-norm based penalty term. The optimization problem can be written as:

$$\arg\min_{\Delta\boldsymbol{\theta}}\{-\langle\Delta\boldsymbol{x}, \boldsymbol{x}(\boldsymbol{\theta}^{(n)} + \Delta\boldsymbol{\theta}) - \boldsymbol{x}(\boldsymbol{\theta}^{(n)})\rangle + \lambda\|\Delta\boldsymbol{\theta}\|_2^2\} \tag{10}$$

where $\boldsymbol{\theta}^{(n)}$ denotes the model weights at the n-th step. With that, the first-order expansion of $\boldsymbol{x}(\boldsymbol{\theta})$ is given by:

$$\boldsymbol{x}(\boldsymbol{\theta}) = \boldsymbol{x}(\boldsymbol{\theta}^{(n)}) + \frac{\partial\boldsymbol{x}(\boldsymbol{\theta}^{(n)})}{\partial\boldsymbol{\theta}}\cdot(\boldsymbol{\theta} - \boldsymbol{\theta}^{(n)}) + o(\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(n)}\|) \tag{11}$$

where $\partial\boldsymbol{x}(\boldsymbol{\theta}^{(n)})/\partial\boldsymbol{\theta}$ is the Jacobian matrix of vector-valued function $\boldsymbol{x}(\boldsymbol{\theta})$ at $\boldsymbol{\theta}^{(n)}$. Ignoring the high-order infinitesimal, we obtain the step-wise minimization process:

$$\boldsymbol{\theta}^{(n+1)} - \boldsymbol{\theta}^{(n)} = \arg\min_{\Delta\boldsymbol{\theta}}\{-\langle\Delta\boldsymbol{x}, \frac{\partial\boldsymbol{x}(\boldsymbol{\theta}^{(n)})}{\partial\boldsymbol{\theta}}\Delta\boldsymbol{\theta}\rangle + \lambda\|\Delta\boldsymbol{\theta}\|_2^2\} \tag{12}$$

The optimal solution can be obtained by finding the stationary point. Then, the form of optimal $\Delta\boldsymbol{\theta}$ is consistent with the chain rule of derivative, which means, it can be directly implemented by setting the gradient of x_{ij} to $-\Delta x_{ij}$ (c.f. Equation 9) and proceeding with backpropagation. Hence the gradient for score s_i can be obtained by backward propagating the gradient through the difference transformation:

$$\begin{aligned} g_i = -\sum_{j,k}\Delta x_{jk}\cdot\frac{\partial x_{jk}}{\partial s_i} &= \sum_j \Delta x_{ij} - \sum_j \Delta x_{ji} \\ &= \sum_j L_{ji}\cdot y_{ji} - \sum_j L_{ij}\cdot y_{ij} \end{aligned} \tag{13}$$

3.3. Analyses

Convergence: To better understand the characteristics of the AP-loss, we first provide a theoretical analysis on the convergence of the optimization algorithm, which is generalized from the convergence property of the original perceptron learning algorithm.

Proposition 1 *The AP-loss optimizing algorithm is guaranteed to converge in finite steps if below conditions hold:*
(1) the learning model is linear;
(2) the training data is linearly separable.

The proof of this proposition is provided in Appendix-1 of supplementary. Although convergence is somewhat weak due to the need of strong conditions, it is **non-trivial** since the AP-loss function is not convex or quasiconvex even for the case of linear model and linearly separable data, so that gradient descent based algorithm may still fail to converge on a smoothed AP-loss function even under such strong conditions. One such example is presented in Appendix-2 of supplementary. It means that, under such conditions, our algorithm still optimizes better than the approximate gradient descent algorithm for AP-loss. Furthermore, with some mild modifications, even though the training data is not separable, the accumulated AP-loss can also be bounded proportionally by the best performance of the learning model. More details are presented in Appendix-3 of supplementary.
Consistency: Besides convergence, We observed that the proposed optimization algorithm is inherently consistent with widely used classification-loss functions.

Observation 1 *When the activation function $L(\cdot)$ takes the form of softmax function and loss-augmented step function, our optimization algorithm can be expressed as the gradient descent algorithm on cross-entropy loss and hinge loss respectively.*

The detailed analysis of this observation is presented in Appendix-4 of supplementary. We argue that the observed consistency is on the basis of the "error-driven" property. As is known, the gradients of those widely used loss functions are proportional to their prediction errors, where the

Algorithm 1 Minibatch training for Interpolated AP

```
Input: All scores $\{s_i\}$ and corresponding labels $\{t_i\}$ in a minibatch
Output: Gradient of input $\{g_i\}$
 1: $\forall i,\ \ g_i \leftarrow 0$
 2: MaxPrec $\leftarrow 0$
 3: $\mathcal{P} \leftarrow \{i \mid t_i = 1\},\ \ \mathcal{N} \leftarrow \{i \mid t_i = 0\}$
 4: $O \leftarrow argsort(\{s_i \mid i \in \mathcal{P}\})$    ▷ Indexes of scores sorted in ascending order
 5: for $i \in O$ do
 6:     Compute $x_{ij} = s_j - s_i$ for all $j \in \mathcal{P}\cup\mathcal{N}$ and $L_{ij}$ for all $j \in \mathcal{N}$    ▷ According to Equation 3 and Equation 14
 7:     Prec $\leftarrow 1 - \sum_{j\in\mathcal{N}} L_{ij}$
 8:     if Prec $\geq$ MaxPrec then
 9:         MaxPrec $\leftarrow$ Prec
10:     else    ▷ Interpolation
11:         $\forall j \in \mathcal{N},\ \ L_{ij} \leftarrow L_{ij}\cdot(1-\text{MaxPrec})/(1-\text{Prec})$
12:     end if
13:     $g_i \leftarrow -\sum_{j\in\mathcal{N}} L_{ij}$    ▷ According to Equation 13
14:     $\forall j \in \mathcal{N},\ \ g_j \leftarrow g_j + L_{ij}$    ▷ According to Equation 13
15: end for
16: $\forall i,\ \ g_i \leftarrow g_i/|\mathcal{P}|$    ▷ Normalization
```

prediction here refers to the output of activation function. In other words, their activation functions have a nice property: the vector field of prediction errors is conservative, allowing it being the gradient of some surrogate loss function. However, our activation function does not have this property, which makes our optimization not able to express as gradient descent with any surrogate loss function.

3.4. Details of Training Algorithm

Minibatch Training The minibatch training strategy is widely used in deep learning frameworks [8, 18, 15] as it accounts for more stability than the case with batch size equal to 1. The mini-batch training helps our optimization algorithm quite a lot for escaping the so-called "score-shift" scenario. The AP-loss can be computed both from a batch of images and from a single image with multiple anchor boxes. Consider an extreme case: our detector can predict perfect ranking in both image I_1 and image I_2, but the lowest score in image I_1 is even greater than the highest score in image I_2. There are "score-shift" between two images so that the detection performance is poor when computing AP-loss per-image. Aggregating scores over images in a mini-batch can avoid such problem, so that the minibatch training is crucial for good convergence and good performance.
Piecewise Step function During early stage of training, scores s_i are very close to each other (*i.e.* almost all inputs to Heaviside step function $H(x)$ are near zero), so that a small change of input will cause a big output difference, which destabilizes the updating process. To tackle this issue, we replace $H(x)$ with a piecewise step function:

$$f(x) = \begin{cases} 0\,, & x < -\delta \\ \dfrac{x}{2\delta} + 0.5\,, & -\delta \leq x \leq \delta \\ 1\,, & \delta < x \end{cases} \tag{14}$$

Batch Size	AP	AP$_{50}$	AP$_{75}$
1	52.4	80.2	56.7
2	53.0	81.7	57.8
4	52.8	82.2	58.0
8	**53.1**	**82.3**	**58.1**

(a) Varying batch size

δ	AP	AP$_{50}$	AP$_{75}$
0.25	50.2	80.7	53.6
0.5	51.3	81.6	55.4
1	**53.1**	82.3	**58.1**
2	52.8	**82.6**	57.2

(b) Varying δ for piecewise step function

Interpolated	AP	AP$_{50}$	AP$_{75}$
No	52.6	82.2	57.1
Yes	**53.1**	**82.3**	**58.1**

(c) Interpolated *vs.* not interpolated

Table 1: Ablation experiments. Models are tested on VOC2007 `test` set.

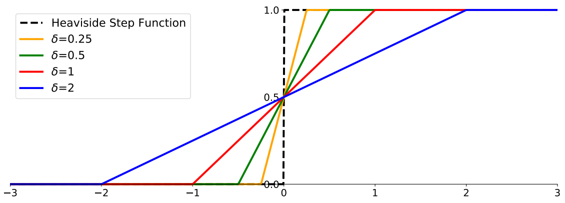


Figure 4: Heaviside step function and piecewise step function. (Best viewed in color)

The piecewise step functions with different δ are shown in Figure 4. When δ approaches $+0$, the piecewise step function approaches the original step function. Note that $f(\cdot)$ is only different from $H(\cdot)$ near zero point. We argue that the precise form of the piecewise step function is not crucial. Other monotonic and symmetric smooth functions that only differs from $H(\cdot)$ near zero point could be equally effective. The choice of δ relates closely to the weight decay hyper-parameter in CNN optimization. Intuitively, parameter δ controls the width of decision boundary between positive and negative samples. Smaller δ enforces a narrower decision boundary, which causes the weights to shrink correspondingly (similar effect to that caused by the weight decay). Further details are presented in the experiments.

Interpolated AP The interpolated AP [27] is widely adopted by many object detection benchmarks like PASCAL VOC [5] and MS COCO [16]. The common justification for interpolating the precision-recall curve [5] is "to reduce the impact of 'wiggles' in the precision-recall curve, caused by small variations in the ranking of examples". Under the same consideration, we adopt the interpolated AP instead of the original version. Specifically, the interpolation is applied on L_{ij} to make the precision at the k-th smallest positive sample monotonically increasing with k where the precision is $(1 - \sum_{j\in\mathcal{N}} L_{ij})$ in which i is the index of the k-th smallest positive sample. It is worth noting that the interpolated AP is a smooth approximation of the actual AP so that it is a practical choice to help to stabilize the gradient and to reduce the impact of 'wiggles' in the update signals. The details of the interpolated AP based algorithm is summarized in Algorithm 1.

4. Experiments

4.1. Experimental Settings

We evaluate the proposed method on the state-of-the-art one-stage detector RetinaNet [15]. The implementation details are the same as in [15] unless explicitly stated. Our experiments are performed on two benchmark datasets: PASCAL VOC [5] and MS COCO [16]. The PASCAL VOC dataset has 20 classes, with VOC2007 containing 9,963 images for train/val/test and VOC2012 containing 11,530 for train/val. The MS COCO dataset has 80 classes, containing 123,287 images for train/val. We implement our codes with the MXNET framework, and conduct experiments on a workstation with two NVidia TitanX GPUs.

PASCAL VOC: When evaluated on the VOC2007 `test` set, models are trained on the VOC2007 and VOC2012 `trainval` sets. When evaluated on the VOC2012 `test` set, models are trained on the VOC2007 and VOC2012 `trainval` sets plus the VOC2007 `test` set. Similar to the evaluation metrics used in the MS COCO benchmark, we also report the AP averaged over multiple IoU thresholds of $0.50 : 0.05 : 0.95$. We set $\delta = 1$ in Equation 14. We use ResNet [8] as the backbone model which is pre-trained on the ImageNet-1k classification dataset [4]. At each level of FPN [14], the anchors have 2 sub-octave scales ($2^{k/2}$, for $k \leq 1$) and 3 aspect ratios [0.5, 1, 2]. We fix the batch normalization layers to be frozen in training phase. We adopt the minibatch training on 2 GPUs with 8 images per GPU. All evaluated models are trained for 160 epochs with an initial learning rate of 0.001 which is then divided by 10 at 110 epochs and again at 140 epochs. Weight decay of 0.0001 and momentum of 0.9 are used. We adopt the same data augmentation strategies as [18], while do not use any data augmentation during testing phase. In training phase, the input image is fixed to 512×512, while in testing phase, we maintain the original aspect ratio and resize the image to ensure the shorter side with 600 pixels. We apply the non-maximum suppression with IoU of 0.5 for each class.

MS COCO: All models are trained on the widely used `trainval35k` set (80k train images and 35k subset of val images), and tested on `minival` set (5k subset of val images) or `test-dev` set. We train the networks for 100 epochs with an initial learning rate of 0.001 which is then divided by 10 at 60 epochs and again at 80 epochs. Other details are similar to that for PASCAL VOC.

4.2. Ablation Study

We first investigate the impact of our design settings of the proposed framework. We fix the ResNet-50 as backbone and conduct several controlled experiments on PAS-

Training Loss	PASCAL VOC			COCO		
	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
CE-Loss + OHEM	49.1	81.5	51.5	30.8	50.9	32.6
Focal Loss	51.3	80.9	55.3	33.9	55.0	35.7
AUC-Loss	49.3	79.7	51.8	25.5	44.9	26.0
AP-Loss	**53.1**	**82.3**	**58.1**	**35.0**	**57.2**	**36.6**

Table 2: Comparison through different training losses. Models are tested on VOC2007 `test` and COCO `minival` sets. The metric AP is averaged over multiple IoU thresholds of $0.50 : 0.05 : 0.95$.

CAL VOC2007 `test` set (and COCO `minival` if stated) for this ablation study.

4.2.1 Comparison on Different Parameter Settings

Here we study the impact of the practical modifications introduced in Section 3.4. All results are shown in Table 1.

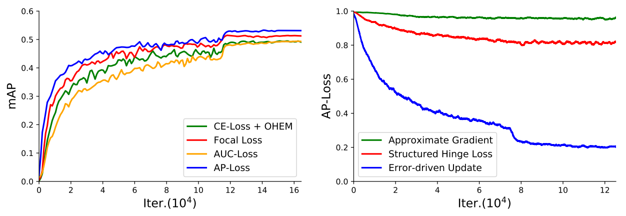
Minibatch Training: First, we study the mini-batch training, and report detector results at different batch-size in Table 1a. It shows that larger batch-size (*i.e.* 8) outperforms all the other smaller batch-size. This verifies our previous hypothesis that large minibatch training helps to eliminate the "score-shift" from different images, and thus stabilizes the AP-loss through robust gradient calculation. Hence, batch-size $= 8$ is used in our further studies.

Piecewise Step Function: Second, we study the piecewise step function, and report detector performance on the piecewise step function with different δ in Table 1b. As mentioned before, we argue that the choice of δ is trivial and is dependent on other network hyper-parameters such as weight decay. Smaller δ makes the function sharper, which yields unstable training at initial phase. Larger δ makes the function deviate from the properties of the original AP-loss, which also worsens the performance. $\delta = 1$ is a good choice we used in our further studies.

Interpolated AP: Third, we study the impact of interpolated AP in our optimization algorithm, and list the results in Table 1c. Marginal benefits are observed for interpolated AP over standard AP, so we use interpolated AP in all the following studies.

4.2.2 Comparison on Different Losses

We evaluate with different losses on RetinaNet [15]. Results are shown in Table 2. We compare traditional classification based losses like focal loss [15] and cross entropy loss (CE-loss) with OHEM [18] to the ranking based losses like AUC-loss and AP-loss. Although focal loss is significantly better than CE-loss with OHEM on COCO dataset, it is interesting that focal-loss does not perform better than CE-loss at AP_{50} on PASCAL VOC. This is likely because the hyper-parameters of focal loss are designed to suit the imbalance condition on COCO dataset which is not suitable for PASCAL VOC, so that focal loss cannot generalize well



(a) (b)

Figure 5: (a) Detection accuracy (mAP) on VOC2007 `test` set. (b) Convergence curves of different AP-loss optimizations on VOC2007 `trainval` set. (Best viewed in color)

to PASCAL VOC without tuning its hyper-parameters. The proposed AP-loss performs much better than all the other losses on both two datasets, which demonstrates its effectiveness and stronger generalization ability on handling the imbalance issue. It is worth noting that AUC-loss performs much worse than AP-loss, which may be due to the fact that AUC has equal penalty for each misordered pair while AP imposes greater penalty for the misordering at higher positions in the predicted ranking. It is obvious that object detection evaluation concerns more on objects with higher confidence, which is why AP provides a better loss measure. Furthermore, an assessment of the detection performance at different training iterations, as shown in Figure 5a, outlines the superiority of the AP-loss for snapshot time points.

4.2.3 Comparison on Different Optimization Methods

We also compare our optimization method with the approximate gradient method [34, 9] and structured hinge loss method [20]. Both [34, 9] approximate the AP-loss with a smooth expectation and envelope function, respectively. Following their guidance, we replace the step function in AP-loss with a sigmoid function to constrain the gradient to neither zero nor undefined, while still keep the shape similar to the original function. Same as [9], we adopt the log space objective function, *i.e.* $log(\mathrm{AP} + \epsilon)$, to allow the model to quickly escape from the initial state. We train the detector on VOC2007 `trainval` set and turn off the bounding box regression task. The convergence curves shown in Figure 5b reveal some essential observations. It can be seen that AP-loss optimized by approximate gradient method does not even converge, likely because its non-convexity and non-quasiconvexity fail on a direct gradient descent method. Meanwhile, AP-loss optimized by the structured hinge loss method [20] converges slowly and stabilizes near 0.8, which is significantly worse than the asymptotic limit of AP-loss optimized by our error-driven update scheme. We believe that this method does not optimize the AP-loss directly but rather an upper bound of it, which is controlled by a discriminant function [20]. In ranking task, this discriminant function is hand-picked and has an AUC-like form, which may cause variability in optimization.

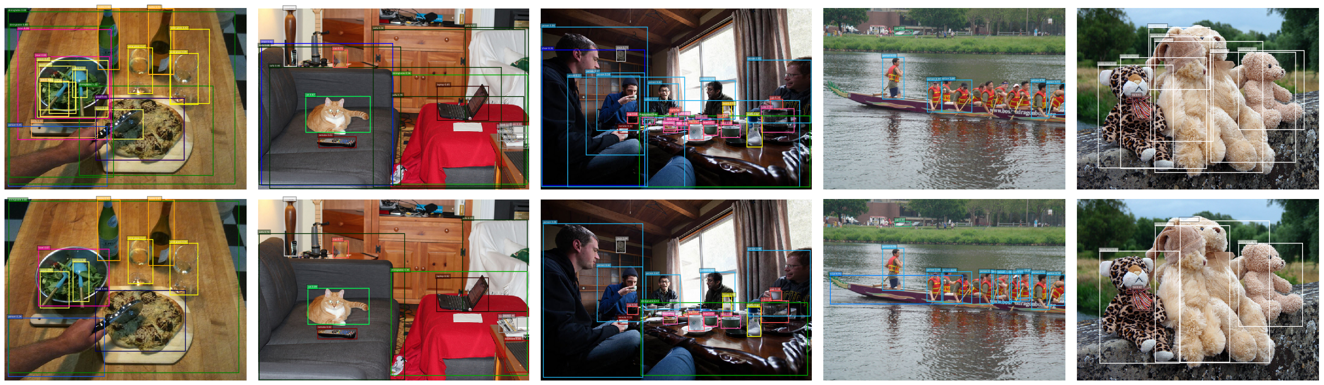


Figure 6: Some detection examples. Top: Baseline results by RetinaNet with focal loss. Bottom: Our results with AP-loss.

Method	Backbone	Multi-Scale	VOC07	VOC12	COCO					
			AP$_{50}$	AP$_{50}$	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
YOLOv2 [24]	DarkNet-19	✗	78.6	73.4	21.6	44.0	19.2	5.0	22.4	35.5
DSOD300 [30]	DS/64-192-48-1	✗	77.7	76.3	29.3	47.3	30.6	9.4	31.5	47.0
SSD512 [18]	VGG-16	✗	79.8	78.5	28.8	48.5	30.3	-	-	-
SSD513 [6]	ResNet-101	✗	80.6	79.4	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [6]	ResNet-101	✗	81.5	80.0	33.2	53.3	35.2	13.0	35.4	51.1
DES512 [39]	VGG-16	✗	81.7	80.3	32.8	53.2	34.6	13.9	36.0	47.6
RFBNet512 [17]	VGG-16	✗	82.2	-	33.8	54.2	35.9	16.2	37.1	47.4
PFPNet-R512 [10]	VGG-16	✗	82.3	80.3	35.2	57.6	37.9	**18.7**	38.6	45.9
RefineDet512 [38]	VGG-16	✗	81.8	80.1	33.0	54.5	35.5	16.3	36.3	44.3
RefineDet512 [38]	ResNet-101	✗	-	-	36.4	57.5	39.5	16.6	39.9	51.4
RetinaNet500 [15]	ResNet-101	✗	-	-	34.4	53.1	36.8	14.7	38.5	49.1
RetinaNet500+AP-Loss (ours)	ResNet-101	✗	**83.9**	**83.1**	**37.4**	**58.6**	**40.5**	17.3	**40.8**	**51.9**
PFPNet-R512 [10]	VGG-16	✓	84.1	83.7	39.4	61.5	42.6	25.3	42.3	48.8
RefineDet512 [38]	VGG-16	✓	83.8	83.5	37.6	58.7	40.8	22.7	40.3	48.3
RefineDet512 [38]	ResNet-101	✓	-	-	41.8	62.9	45.7	**25.6**	**45.1**	**54.1**
RetinaNet500+AP-Loss (ours)	ResNet-101	✓	**84.9**	**84.5**	**42.1**	**63.5**	**46.4**	**25.6**	45.0	53.9

Table 3: Detection results on VOC2007 `test`, VOC 2012 `test` and COCO `test-dev` sets.

4.3. Benchmark Results

With the settings selected in ablation study, we conduct experiments to compare the proposed detector to state-of-the-art one-stage detectors on three widely used benchmark, *i.e.* VOC2007 `test`, VOC2012 `test` and COCO `test-dev` sets. We use ResNet-101 as backbone networks instead of ResNet-50 in ablation study. We use an image scale of 500 pixels for testing. Table 3 lists the benchmark results comparing to recent state-of-the-art one-stage detectors such as SSD [18], YOLOv2 [24], DSSD [6], DSOD [30], DES [39], RetinaNet [15], RefineDet [38], PFPNet [10], RFBNet [17]. Compared to the baseline model RetinaNet500 [15], our detector achieves a 3.0% improvement (37.4% *vs.* 34.4%) on COCO dataset. Figure 6 illustrates some detection results by the RetinaNet with focal loss and our AP-loss. Besides, our detector outperforms all the other methods for both single-scale and multi-scale tests in all the three benchmarks. We should emphasize that this verifies the great effectiveness of our AP-loss since our detector achieves such a great performance gain just by replacing the focal-loss with our AP-loss in RetinaNet without whistle and bells, without using advanced techniques like deformable convolution [3], SNIP [33], group normalization [36], etc. The performance could be further improved with these kinds of techniques and other possible tricks. Our detector has the same detection speed (*i.e.*, $\sim$11 fps on one NVidia TitanX GPU) as RetinaNet500 [15] since it does not change the network architecture for inference.

5. Conclusion

In this paper, we address the class imbalance issue in one-stage object detectors by replacing the classification sub-task with a ranking sub-task, and proposing to solve the ranking task with AP-Loss. Due to non-differentiability and non-convexity of the AP-loss, we propose a novel algorithm to optimize it based on error-driven update scheme from perceptron learning. We provide a grounded theoretical analysis of the proposed optimization algorithm. Experimental results show that our approach can significantly improve the state-of-the-art one-stage detectors.

Acknowledgements. This paper is supported in part by: National Natural Science Foundation of China (61471235), Shanghai 'The Belt and Road' Young Scholar Exchange Grant (17510740100), CREST Malaysia (No. T03C1-17), and the PKU-NTU Joint Research Institute (JRI) sponsored by a donation from the Ng Teng Fong Charitable Foundation. We gratefully acknowledge the support from Tencent YouTu Lab.

References

[1] JK Anlauf and M Biehl. The adatron: an adaptive perceptron algorithm. *EPL (Europhysics Letters)*, 10(7):687, 1989.

[2] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, pages 379–387, 2016.

[3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, et al. Deformable convolutional networks. In *ICCV*, pages 764–773, 2017.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[5] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015.

[6] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[7] Ross Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[9] Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. In *ACCV*, pages 198–213, 2016.

[10] Seung-Wook Kim, Hyong-Keun Kook, Jee-Young Sun, Mun-Cheon Kang, and Sung-Jea Ko. Parallel feature pyramid network for object detection. In *ECCV*, pages 234–250, 2018.

[11] Werner Krauth and Marc Mézard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A: Mathematical and General*, 20(11):L745, 1987.

[12] Jianguo Li and Y Zhang. Learning surf cascade for fast and accurate object detection. In *CVPR*, 2013.

[13] Yuxi Li, Jiuwei Li, Weiyao Lin, and Jianguo Li. Tiny-DSOD: Lightweight object detection for resource-restricted usages. In *BMVC*, 2018.

[14] Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 3, 2017.

[15] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Trans on PAMI*, 2018.

[16] 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*, pages 740–755, 2014.

[17] Songtao Liu, Di Huang, and andYunhong Wang. Receptive field block net for accurate and fast object detection. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37, 2016.

[19] Pritish Mohapatra, CV Jawahar, and M Pawan Kumar. Efficient optimization for average precision svm. In *NIPS*, pages 2312–2320, 2014.

[20] Pritish Mohapatra, Michal Rolinek, C.V. Jawahar, Vladimir Kolmogorov, and M. Pawan Kumar. Efficient optimization for rank-based loss functions. In *CVPR*, 2018.

[21] A Novikoff. On convergence proofs for perceptrons. *Proc.sympos.math.theory of Automata*, pages 615–622, 1963.

[22] Yongming Rao, Dahua Lin, Jiwen Lu, and Jie Zhou. Learning globally optimized object detector via policy gradient. In *CVPR*, pages 6190–6198, 2018.

[23] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016.

[24] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.

[26] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[27] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[28] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[29] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[30] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Dsod: Learning deeply supervised object detectors from scratch. In *ICCV*, 2017.

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

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[33] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection–snip. In *CVPR*, 2018.

[34] Yang Song, Alexander Schwing, Raquel Urtasun, et al. Training deep neural networks via direct loss minimization. In *ICML*, pages 2169–2177, 2016.

[35] A Wendemuth. Learning the unlearnable. *Journal of Physics A: Mathematical and General*, 28(18):5423, 1995.

[36] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

[37] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *SIGIR*, pages 271–278. ACM, 2007.

[38] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *CVPR*, 2018.

[39] Zhishuai Zhang, Siyuan Qiao, Cihang Xie, Wei Shen, Bo Wang, and Alan L. Yuille. Single-shot object detection with enriched semantics. In *CVPR*, 2018.

A1. Convergence

We provide proof for the proposition mentioned in Section 3.3.1 of the paper. The proof is generalized from the original convergence proof [21] for perceptron learning algorithm.

Proposition 2 *The AP-loss optimizing algorithm is guaranteed to converge in finite steps if below conditions hold:*
(1) the learning model is linear;
(2) the training data is linearly separable.

Proof. Let $\boldsymbol{\theta}$ denote the weights of the linear model. Let $\boldsymbol{f}_k^{(n)}$ denote the feature vector of k-th box in n-th training sample. Hence the score of k-th box is $s_k^{(n)} = \langle \boldsymbol{f}_k^{(n)}, \boldsymbol{\theta} \rangle$. Define $x_{ij}^{(n)} = -(s_i^{(n)} - s_j^{(n)})$. Note that the training data is separable, which means there are $\epsilon > 0$ and $\boldsymbol{\theta}^*$ that satisfy:

$$\forall n,\ \forall i \in \mathcal{P}^{(n)},\ \forall j \in \mathcal{N}^{(n)},\ \langle \boldsymbol{f}_i^{(n)}, \boldsymbol{\theta}^* \rangle \geq \langle \boldsymbol{f}_j^{(n)}, \boldsymbol{\theta}^* \rangle + \epsilon \quad (15)$$

In the t-th step, a training sample which makes an error (if there is no such training sample, the model is already optimal and algorithm will stop) is randomly chosen. Then the update of $\boldsymbol{\theta}$ is:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}} L_{ij}(\boldsymbol{x}) \cdot (\boldsymbol{f}_i - \boldsymbol{f}_j) \quad (16)$$

where

$$L_{ij}(\boldsymbol{x}) = \frac{H(x_{ij})}{1 + \sum_{k\neq i} H(x_{ik})} \quad (17)$$

Here, since the discussion centers on the current training sample, we omit the superscript hereon.

From (16), we have

$$\begin{aligned}
\langle \boldsymbol{\theta}^{(t+1)}, \boldsymbol{\theta}^* \rangle &= \langle \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^* \rangle + \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}} L_{ij} \langle (\boldsymbol{f}_i - \boldsymbol{f}_j), \boldsymbol{\theta}^* \rangle \\
&\geq \langle \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^* \rangle + \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}} L_{ij}\epsilon \\
&\geq \langle \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^* \rangle + \max_{i\in\mathcal{P}, j\in\mathcal{N}} \{L_{ij}\}\epsilon \\
&\geq \langle \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^* \rangle + \frac{1}{|\mathcal{P}| + |\mathcal{N}|}\epsilon
\end{aligned} \quad (18)$$

Hence we have

$$\langle \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^* \rangle \geq \frac{1}{|\mathcal{P}| + |\mathcal{N}|}\epsilon \cdot t \quad (19)$$

Then

$$\|\boldsymbol{\theta}^{(t)}\| \geq \frac{\langle \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^* \rangle}{\|\boldsymbol{\theta}^*\|} \geq \frac{1}{(|\mathcal{P}| + |\mathcal{N}|)\cdot\|\boldsymbol{\theta}^*\|}\epsilon \cdot t \geq c \cdot t \quad (20)$$

Here, c is a positive constant.

From (16), we also have

$$\begin{aligned}
&\|\boldsymbol{\theta}^{(t+1)}\|^2 \\
=&\|\boldsymbol{\theta}^{(t)}\|^2 + \|\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}} L_{ij}(\boldsymbol{f}_i - \boldsymbol{f}_j)\|^2 \\
&\qquad + 2\langle \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}} L_{ij}(\boldsymbol{f}_i - \boldsymbol{f}_j), \boldsymbol{\theta}^{(t)} \rangle \\
=&\|\boldsymbol{\theta}^{(t)}\|^2 + \|\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}} L_{ij}(\boldsymbol{f}_i - \boldsymbol{f}_j)\|^2 + 2\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}} L_{ij}x_{ji} \\
\leq&\|\boldsymbol{\theta}^{(t)}\|^2 + \|\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}} L_{ij}(\boldsymbol{f}_i - \boldsymbol{f}_j)\|^2 \\
\leq&\|\boldsymbol{\theta}^{(t)}\|^2 + |\mathcal{P}| \cdot |\mathcal{N}| \cdot \max_{i\in\mathcal{P}, j\in\mathcal{N}} \{\|\boldsymbol{f}_i - \boldsymbol{f}_j\|^2\} \\
\leq&\|\boldsymbol{\theta}^{(t)}\|^2 + C
\end{aligned} \quad (21)$$

Here, C is a positive constant. Hence we arrive at:

$$\|\boldsymbol{\theta}^{(t)}\|^2 \leq C \cdot t \quad (22)$$

Then, combining (20) and (22), we have

$$c^2 \cdot t^2 \leq \|\boldsymbol{\theta}^{(t)}\|^2 \leq C \cdot t \quad (23)$$

which means

$$t \leq \frac{C}{c^2} \quad (24)$$

It shows that the algorithm will stop at most after C/c^2 steps, which means that the training model will achieve the optimal solution at most after C/c^2 steps.

A2. An Example of Gradient Descent Failing on Smoothed AP-loss

We approximate the step function in AP-loss by sigmoid function to make it amenable to gradient descent. Specifically, the smoothed AP-loss function is given by:

$$F = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}} \frac{S(x_{ij})}{1 + \sum_{k\neq i} S(x_{ik})} \quad (25)$$

where

$$S(x) = \frac{e^x}{1 + e^x} \quad (26)$$

Consider a linear model $s = f_1\theta_1 + f_2\theta_2$ and three training samples $(0, 0), (1, 0), (-3, 1)$ (the first one is negative sample, others are positive samples). Then we have

$$\begin{aligned}
s^{(1)} &= 0 \cdot \theta_1 + 0 \cdot \theta_2 \\
s^{(2)} &= 1 \cdot \theta_1 + 0 \cdot \theta_2 \\
s^{(3)} &= -3 \cdot \theta_1 + 1 \cdot \theta_2
\end{aligned} \quad (27)$$

Note that the training data is separable since we have $s^{(2)} > s^{(1)}$ and $s^{(3)} > s^{(1)}$ when $0 < \theta_1 < \frac{1}{3} \cdot \theta_2$.

Under this setting, the smoothed AP-loss become

$$F(\theta_1,\theta_2) = \frac{1}{2}(\frac{S(-\theta_1)}{1+S(-\theta_1)+S(\theta_2-4\theta_1)} + \frac{S(3\theta_1-\theta_2)}{1+S(4\theta_1-\theta_2)+S(3\theta_1-\theta_2)}) \tag{28}$$

If θ_1 is sufficiently large and $\theta_1 > \theta_2 > 0$, then the partial derivatives satisfy the following condition:

$$\frac{\partial F}{\partial \theta_1} < \frac{\partial F}{\partial \theta_2} < 0 \tag{29}$$

which means θ_1 and θ_2 will keep increasing with the inequality $\theta_1 > \theta_2$ according to the gradient descent algorithm. Hence the objective function F will approach $1/6$ here. However, the objective function F approaches the global minimum value 0 if and only if $\theta_1 \to +\infty$ and $\theta_2 - 3\theta_1 \to +\infty$. This shows that the gradient descent fails to converge to global minimum in this case.

A3. Inseparable Case

In this section, we will provide analysis for our algorithm with inseparable training data. We demonstrate that the bound of accumulated AP-loss depends on the best performance of learning model. The analysis is based on online learning bounds [29].

A3.1. Preliminary

To handle the inseparable case, a mild modification on the proposed algorithm is needed, *i.e.* in the error-driven update scheme, L_{ij} is modified to

$$\widetilde{L}_{ij} = \frac{\widetilde{H}(x_{ij})}{1+\sum_{k\in\mathcal{P}\cup\mathcal{N},k\neq i} H(x_{ik})} \tag{30}$$

where $\widetilde{H}(\cdot)$ is defined in Section 3.4.2 (Piecewise Step Function) of the paper. The purpose is to introduce a non-zero decision margin for the pairwise score x_{ij} which makes the algorithm more robust in the inseparable case. In contrast to the case in Section 3.4.2, here we only change $H(\cdot)$ to $\widetilde{H}(\cdot)$ in the numerator for the convenience of theoretical anaysis. However, such algorithm still suffers from the discontinuity of $H(\cdot)$ in the denominator. Hence the strategy in Section 3.4.2 is also practical consideration, necessary for good performance. Then, consider the AP-loss:

$$\mathcal{L}_{AP}(\boldsymbol{x}) = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \frac{\sum_{j\in\mathcal{N}} H(x_{ij})}{1+\sum_{j\in\mathcal{P}\cup\mathcal{N},j\neq i} H(x_{ij})} \tag{31}$$

and define a surrogate loss function:

$$l(\boldsymbol{x},\hat{\boldsymbol{x}}) = \frac{1}{|P|}\sum_{i\in\mathcal{P}} \frac{\sum_{j\in\mathcal{N}} Q(x_{ij})}{1+\sum_{j\in\mathcal{P}\cup\mathcal{N},j\neq i} H(\hat{x}_{ij})} \tag{32}$$

where $Q(x) = \int_{-\infty}^{x} \widetilde{H}(v)dv$. Note that the AP-loss is upper bounded by the surrogate loss:

$$l(\boldsymbol{x},\boldsymbol{x}) \geq \frac{\delta}{4}\mathcal{L}_{AP}(\boldsymbol{x}) \tag{33}$$

The learning model can be written as $\boldsymbol{x} = \boldsymbol{X}_d(\boldsymbol{\theta})$, where $d \in \mathcal{D}$ denotes the training data for one iteration and D is the whole training set. Then, the modified error-driven algorithm is equivalent to gradient descent on surrogate loss $l(\boldsymbol{X}_{d^{(t)}}(\boldsymbol{\theta}), \boldsymbol{X}_{d^{(t)}}(\boldsymbol{\theta}^{(t)}))$ at each step t. We further suppose below conditions are satisfied:
(1) For all $\hat{\boldsymbol{\theta}}$ and $d \in \mathcal{D}$, $l(\boldsymbol{X}_d(\boldsymbol{\theta}), \boldsymbol{X}_d(\hat{\boldsymbol{\theta}}))$ is convex *w.r.t* $\boldsymbol{\theta}$.
(2) For all $d \in \mathcal{D}$, $\|\partial \boldsymbol{X}_d(\boldsymbol{\theta})/\partial\boldsymbol{\theta}\|$ is upper bounded by a constant R. Here $\|\cdot\|$ is the matrix norm induced by the 2-norm for vectors.
Remark 1. Note that these two conditions are satisfied if the learning model is linear.

A3.2. Bound of Accumulated Loss

By the convexity, we have:

$$l^{(t)}(\boldsymbol{\theta}) \leq l^{(t)}(\boldsymbol{u}) + \langle \boldsymbol{\theta}-\boldsymbol{u}, \frac{\partial l^{(t)}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\rangle \tag{34}$$

where we use $l^{(t)}(\boldsymbol{\theta})$ to denote $l(\boldsymbol{X}_{d^{(t)}}(\boldsymbol{\theta}), \boldsymbol{X}_{d^{(t)}}(\boldsymbol{\theta}^{(t)}))$ and $\boldsymbol{u}$ can be any vector of model weights. Then, let $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$ and compute the sum over $t = 1 \sim T$, we have:

$$\begin{aligned}\sum_{t=1}^{T} l^{(t)}(\boldsymbol{\theta}^{(t)}) - \sum_{t=1}^{T} l^{(t)}(\boldsymbol{u}) &\leq \sum_{t=1}^{T}\langle \boldsymbol{\theta}^{(t)}-\boldsymbol{u}, \frac{\partial l^{(t)}(\boldsymbol{\theta}^{(t)})}{\partial \boldsymbol{\theta}}\rangle \\ &= \sum_{t=1}^{T}\langle \boldsymbol{\theta}^{(t)}-\boldsymbol{u}, \frac{1}{\eta}(\boldsymbol{\theta}^{(t)}-\boldsymbol{\theta}^{(t+1)})\rangle \\ &\leq \frac{1}{2\eta}\|\boldsymbol{u}-\boldsymbol{\theta}^{(1)}\|^2 + \frac{1}{2\eta}\sum_{t=1}^{T}\|\boldsymbol{\theta}^{(t)}-\boldsymbol{\theta}^{(t+1)}\|^2 \\ &= \frac{1}{2\eta}\|\boldsymbol{u}-\boldsymbol{\theta}^{(1)}\|^2 + \frac{\eta}{2}\sum_{t=1}^{T}\|\frac{\partial l^{(t)}(\boldsymbol{\theta}^{(t)})}{\partial \boldsymbol{\theta}}\|^2\end{aligned} \tag{35}$$

where η is the step size of gradient descent. Note that

$$\frac{\partial l^{(t)}(\boldsymbol{\theta}^{(t)})}{\partial \boldsymbol{\theta}} = \frac{\partial \boldsymbol{X}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \cdot \frac{\partial l(\boldsymbol{x},\boldsymbol{x}^{(t)})}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\boldsymbol{X}(\boldsymbol{\theta}^{(t)})} \tag{36}$$

and

$$\begin{aligned}\|\frac{\partial l(\boldsymbol{x},\boldsymbol{x}^{(t)})}{\partial \boldsymbol{x}}\|^2 &= \frac{1}{|\mathcal{P}|^2}\sum_{i\in\mathcal{P}} \frac{\sum_{j\in\mathcal{N}} \widetilde{H}^2(x_{ij})}{(1+\sum_{j\in\mathcal{P}\cup\mathcal{N},j\neq i} H(x_{ij}^{(t)}))^2} \\ &\leq \frac{1}{|\mathcal{P}|^2}\sum_{i\in\mathcal{P}} \frac{\frac{1}{\delta}\sum_{j\in\mathcal{N}} Q(x_{ij})}{1+\sum_{j\in\mathcal{P}\cup\mathcal{N},j\neq i} H(x_{ij}^{(t)})} \leq \frac{1}{\delta} l(\boldsymbol{x},\boldsymbol{x}^{(t)})\end{aligned} \tag{37}$$

Note that both $\mathcal{P}_d$ and $\mathcal{N}_d$ depend on d. However, we omit the subscript d here since the discussion only centers on the current training sample $d^{(t)}$.

Hence we have:

$$\sum_{t=1}^{T} l^{(t)}(\boldsymbol{\theta}^{(t)}) - \sum_{t=1}^{T} l^{(t)}(\boldsymbol{u}) \leq \frac{1}{2\eta}\|\boldsymbol{u}-\boldsymbol{\theta}^{(1)}\|^2 + \frac{\eta R^2}{2\delta}\sum_{t=1}^{T} l^{(t)}(\boldsymbol{\theta}^{(t)}). \tag{38}$$

Let $\eta = \delta/R^2$, rearrange and get the expression:

$$\frac{1}{2}\sum_{t=1}^{T} l^{(t)}(\boldsymbol{\theta}^{(t)}) \leq \frac{R^2}{2\delta}\|\boldsymbol{u}-\boldsymbol{\theta}^{(1)}\|^2 + \sum_{t=1}^{T} l^{(t)}(\boldsymbol{u}) \tag{39}$$

This entails the bound of surrogate loss l:

$$\sum_{t=1}^{T} l^{(t)}(\boldsymbol{\theta}^{(t)}) \leq 2\sum_{t=1}^{T} l^{(t)}(\boldsymbol{u}) + \frac{R^2}{\delta}\|\boldsymbol{u}-\boldsymbol{\theta}^{(1)}\|^2 \tag{40}$$

which implies the bound of AP-loss $\mathcal{L}_{AP}$:

$$\sum_{t=1}^{T} \mathcal{L}_{AP}(\boldsymbol{X}(\boldsymbol{\theta}^{(t)})) \leq \frac{8}{\delta}\sum_{t=1}^{T} l^{(t)}(\boldsymbol{u}) + \frac{4R^2}{\delta^2}\|\boldsymbol{u}-\boldsymbol{\theta}^{(1)}\|^2 \tag{41}$$

As a special case, if there exists a $\boldsymbol{u}$ such that $l^{(t)}(\boldsymbol{u}) = 0$ for all t, then the accumulated AP-loss is bounded by a constant, which implies that convergence can be achieved with finite steps (similar to that of the separable case). Otherwise, with sufficiently large T, the average AP-loss mainly depends on $\frac{1}{T}\frac{8}{\delta}\sum_{t=1}^{T} l^{(t)}(\boldsymbol{u})$. This implies that the bound is meaningful if there still exists a sufficiently good solution $\boldsymbol{u}$ in such inseparable case.

A3.3. Offline Setting

With the offline setting ($d^{(t)} = d$ for all t), a bound with simpler form can be revealed. For simplicity, we will omit the subscript d of $\boldsymbol{X}_d(\boldsymbol{u}), \mathcal{P}_d, \mathcal{N}_d$ and define $A_i(\boldsymbol{u}) = \sum_{j\in\mathcal{N}} Q(X_{ij}(\boldsymbol{u}))$, $Z(\boldsymbol{u}) = \max_{i\in\mathcal{P}}\{A_i(\boldsymbol{u})\}$. Then,

$$\begin{aligned} l^{(t)}(\boldsymbol{u}) &= \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \frac{\sum_{j\in\mathcal{N}} Q(X_{ij}(\boldsymbol{u}))}{1+\sum_{j\in\mathcal{P}\cup\mathcal{N},j\neq i} H(X_{ij}(\boldsymbol{\theta}^{(t)}))} \\ &= \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \frac{A_i(\boldsymbol{u})}{1+\sum_{j\in\mathcal{P}\cup\mathcal{N},j\neq i} H(X_{ij}(\boldsymbol{\theta}^{(t)}))} \\ &\leq \frac{Z(\boldsymbol{u})}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|}\frac{1}{i} \leq \frac{\ln|\mathcal{P}|+1}{|\mathcal{P}|}Z(\boldsymbol{u}) \end{aligned} \tag{42}$$

The second last inequality is based on the fact that $(1 + \sum_{j\in\mathcal{P}\cup\mathcal{N},j\neq i} H(X_{ij}(\boldsymbol{\theta}^{(t)})))$ are picked from $1 \sim (|\mathcal{P}| + |\mathcal{N}|)$ without replacement (assume no ties; if ties exist, this inequality still holds). Combining the results from Equation 42 and Equation 41, we have:

$$\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_{AP}(\boldsymbol{X}(\boldsymbol{\theta}^{(t)})) \leq \frac{\ln|\mathcal{P}|+1}{|\mathcal{P}|}\cdot\frac{8}{\delta}Z(\boldsymbol{u}) + \frac{1}{T}\frac{4R^2\|\boldsymbol{u}-\boldsymbol{\theta}^{(1)}\|^2}{\delta^2} \tag{43}$$

Next,

$$\begin{aligned} l^{(t)}(\boldsymbol{u}) &= \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \frac{A_i(\boldsymbol{u})}{1+\sum_{j\in\mathcal{P}\cup\mathcal{N},j\neq i} H(X_{ij}(\boldsymbol{\theta}^{(t)}))} \\ &= \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \frac{1+\sum_{j\in\mathcal{P},j\neq i} H(X_{ij}(\boldsymbol{\theta}^{(t)}))}{1+\sum_{j\in\mathcal{P}\cup\mathcal{N},j\neq i} H(X_{ij}(\boldsymbol{\theta}^{(t)}))} \\ &\qquad \cdot \frac{A_i(\boldsymbol{u})}{1+\sum_{j\in\mathcal{P},j\neq i} H(X_{ij}(\boldsymbol{\theta}^{(t)}))} \\ &\leq \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \frac{1+\sum_{j\in\mathcal{P},j\neq i} H(X_{ij}(\boldsymbol{\theta}^{(t)}))}{1+\sum_{j\in\mathcal{P}\cup\mathcal{N},j\neq i} H(X_{ij}(\boldsymbol{\theta}^{(t)}))}\cdot Z(\boldsymbol{u}) \\ &= (1-\mathcal{L}_{AP}(\boldsymbol{X}(\boldsymbol{\theta}^{(t)})))\cdot Z(\boldsymbol{u}) \end{aligned} \tag{44}$$

Combining the results from Equation 44 and Equation 41, we have:

$$\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_{AP}(\boldsymbol{X}(\boldsymbol{\theta}^{(t)})) \leq \frac{\frac{8}{\delta}Z(\boldsymbol{u})}{1+\frac{8}{\delta}Z(\boldsymbol{u})} + \frac{1}{T}\frac{4R^2\|\boldsymbol{u}-\boldsymbol{\theta}^{(1)}\|^2}{\delta^2} \tag{45}$$

If $Z(\boldsymbol{u})$ is small, the bound in Equation 43 is active, otherwise the bound in Equation 45 is active. Consequently, we have:

$$\overline{\mathcal{L}_{AP}} \leq \min\{\frac{\ln|\mathcal{P}|+1}{|\mathcal{P}|}\frac{8}{\delta}Z(\boldsymbol{u}), \frac{\frac{8}{\delta}Z(\boldsymbol{u})}{1+\frac{8}{\delta}Z(\boldsymbol{u})}\} + \epsilon \tag{46}$$

where $\overline{\mathcal{L}_{AP}}$ denotes the average AP-loss, $\epsilon \to 0$ as T increases.

A4. Consistency

Observation 2 *When the activation function $L(\cdot)$ takes the form of softmax function and loss-augmented step function, our optimization algorithm can be expressed as the gradient descent algorithm on cross-entropy loss and hinge loss respectively.*

Cross Entropy Loss: Consider the multi-class classification task. The outputs of neural network are $(x_1, \ldots, x_K)$ where K is the number of classes, and the ground truth label is $y \in \{1, \ldots, K\}$. Using softmax as the activation function, we have:

$$\begin{aligned} &(L_1, \ldots, L_K) \\ &= softmax(\boldsymbol{x}) = (\frac{e^{x_1}}{\sum_i e^{x_i}}, \ldots, \frac{e^{x_K}}{\sum_i e^{x_i}}) \end{aligned} \tag{47}$$

The cross entropy loss is:

$$\mathcal{L}_{ce} = -\sum_i \mathbf{1}_{y=i}\log(L_i) \tag{48}$$

Hence the gradient of x_i is

$$g_i = L_i - \mathbf{1}_{y=i} \tag{49}$$

Note that g_i is "error-driven" with the desired output $\mathbf{1}_{y=i}$ and current output L_i. This form is consistent with our error-driven update scheme (c.f. Section 3.2.1 of the paper).

Training Loss	PASCAL VOC		
	AP	AP_{50}	AP_{75}
CE-Loss + OHEM	43.6	76.0	44.7
Focal Loss	39.3	69.9	38.0
AUC-Loss	33.8	63.7	31.5
AP-Loss	**45.2**	**77.3**	**47.3**

Table 4: Comparison through different training losses. Models are tested on VOC2007 `test` set. The metric AP is averaged over multiple IoU thresholds of $0.50 : 0.05 : 0.95$.

Hinge Loss: Consider the binary classification task. The output of neural network is x, and the ground truth label is $y \in \{1, 2\}$. Define $(x_1, x_2) = (-x, x)$. Using loss-augmented step function as the activation function, we have:

$$(L_1, L_2) = (H(x_1 - 1), H(x_2 - 1)) \tag{50}$$

where $H(\cdot)$ is the Heaviside step function. The hinge loss is:

$$\mathcal{L}_{hinge} = \mathbf{1}_{y=1} \max\{1 - x_1, 0\} + \mathbf{1}_{y=2} \max\{1 - x_2, 0\} \tag{51}$$

Hence the gradient of x_i is

$$g_i = \mathbf{1}_{y=i} \cdot (L_i - 1) \tag{52}$$

There are two cases. If $y = i$, the gradient g_i is "error-driven" with the desired output 1 and current output L_i. If $y \neq i$, the gradient g_i equals zero, since x_i does not contribute to the loss. This form is consistent with our error-driven update scheme (c.f. Section 3.2.1 of the paper).

A5. Additional Experiments on SSD

A5.1. Experimental Settings

We also evaluate the proposed AP-loss on another one-stage detector SSD [18]. The models are trained on VOC2007 and VOC2012 `trainval` sets, and tested on VOC2007 `test` set. We use VGG-16 [32] as the backbone model which is pre-trained on the ImageNet-1k classification dataset [4]. We use `conv4_3`, `conv7`, `conv8_2`, `conv9_2`, `conv10_2`, `conv11_2`, `conv12_2` to predict both location and their corresponding confidences. An additional convolution layer is added after `conv4_3` to scale the feature. The associated anchors are the same as that designed in [18]. In testing phase, the input image is fixed to 512×512. For focal loss with SSD, we observe that the hyper-parameters $\gamma = 1, \alpha = 0.25$ lead to a much better performance than the original settings in [15] which are $\gamma = 2, \alpha = 0.25$. Hence we evaluate the focal loss with new γ and α in our experiments on SSD. Other details are similar to that in Section 4.1 of the paper.

A5.2. Results

The results are shown in Table 4 and Figure 7. Note that the AP-loss outperforms all the other losses at both the final state and various snapshot time points. Together with the results on RetinaNet [15] in Section 4.2.2 of the paper, we observe the robustness of the proposed AP-loss, which performs much better than the other competing losses on different datasets (*i.e.* PASCAL VOC [5], MS COCO [16]) and different detectors (*i.e.* RetinaNet [15], SSD [18]). This demonstrates the effectiveness and strong generalization ability of our proposed approach.

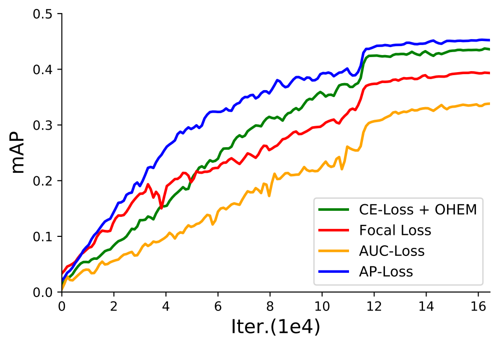


Figure 7: Detection accuracy (mAP) on VOC2007 `test` set.(Best viewed in color)