Learning a Unified Sample Weighting Network for Object Detection*

Qi Cai†, Yingwei Pan‡, Yu Wang‡, Jingen Liu§, Ting Yao‡, and Tao Mei‡
† University of Science and Technology of China, Hefei, China
‡ JD AI Research, Beijing, China § JD AI Research, Mountain View, USA
{cqcaiqi, panyw.ustc, feather1014, jingenliu, tingyao.ustc}@gmail.com, tmei@jd.com

Abstract

Region sampling or weighting is significantly important to the success of modern region-based object detectors. Unlike some previous works, which only focus on "hard" samples when optimizing the objective function, we argue that sample weighting should be data-dependent and task-dependent. The importance of a sample for the objective function optimization is determined by its uncertainties to both object classification and bounding box regression tasks. To this end, we devise a general loss function to cover most region-based object detectors with various sampling strategies, and then based on it we propose a unified sample weighting network to predict a sample's task weights. Our framework is simple yet effective. It leverages the samples' uncertainty distributions on classification loss, regression loss, IoU, and probability score, to predict sample weights. Our approach has several advantages: (i). It jointly learns sample weights for both classification and regression tasks, which differentiates it from most previous work. (ii). It is a data-driven process, so it avoids some manual parameter tuning. (iii). It can be effortlessly plugged into most object detectors and achieves noticeable performance improvements without affecting their inference time. Our approach has been thoroughly evaluated with recent object detection frameworks and it can consistently boost the detection accuracy. Code has been made available at `https://github.com/caiqi/sample-weighting-network`.

1. Introduction

Modern region-based object detection is a multi-task learning problem, which consists of object classification and localization. It involves region sampling (sliding window or region proposal), region classification and regression, and non-maximum suppression. Leveraging region sampling, it converts object detection into a classification task, where a vast number of regions are classified and regressed. According to the way of region search, these detectors can be categorized into one-stage detectors [28, 30, 34, 45] and two-stage detectors [2, 15, 16, 17, 27, 36].

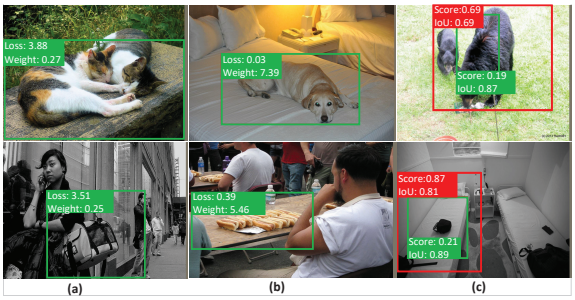

Figure 1: Samples from our training process. (a) The samples having large classification loss but small weight. (b) The samples having small classification loss but large weight. (c) The samples exhibiting inconsistency between classification score and IoU.

In general, the object detectors of the highest accuracy are based on the two-stage framework such as Faster R-CNN [36], which rapidly narrows down regions (most of them are from the background) during the region proposal stage. In contrast, the one-stage detectors, such as SSD [30] and YOLO [34], achieve faster detection speed but lower accuracy. It is because of the class imbalance problem (i.e., the imbalance between foreground and background regions), which is a classic challenge for object detection.

The two-stage detectors deal with class imbalance by a region-proposal mechanism followed by various efficient sample strategies, such as sampling with a fixed foreground-to-background ratio and hard example mining [13, 37, 40]. Although the similar hard example mining can be applied to one-stage detectors, it is inefficient due to a large number of easy negative examples [28]. Unlike the Online Hard Example Mining (OHEM) [37] which explicitly selects samples with high classification losses into the training loop, Focal-Loss [28] proposes a soft weighting strategy, which reshapes the classification loss to automatically down-weight the contributions of easy samples and thus focuses the training on hard samples. As a result, the manually tuned Focal-Loss can significantly improve the performance of one-stage detectors.

*This work was performed at JD AI Research.

The aforementioned "hard" samples generally refer to those with large classification loss. However, a "hard" sample is not necessarily important. As Figure 1 (a) (All samples are selected from our training process.) illustrates, the samples have high classification losses but small weights ("hard" but not important). Conversely, an "easy" sample can be significant if it captures the gist of the object class as shown in Figure 1 (b). In addition, the assumption that the bounding box regression is accurate when the classification score is high, does not always hold as examples shown in Figure 1 (c). There may be a misalignment between classification and regression sometimes [21]. Hence, an IoU-Net is proposed in [21] to predict a location confidence. Furthermore, there are ambiguities in bounding box annotations due to occlusion, inaccurate labeling, and ambiguous object boundary. In other words, the training data has uncertainties. Accordingly, [19] proposes a KL-Loss to learn bounding box regression and location uncertainties simultaneously. The samples with high uncertainty (high regression loss) are down-weighted during training.

Sample weighting is a very complicated and dynamic process. There are various uncertainties, which exist in individual samples when applying to a loss function of a multi-task problem. Inspired by [23], we argue that sample weighting should be *data-dependent* and *task-dependent*. On the one hand, unlike previous work, the importance of a sample should be determined by its intrinsic property compared to the ground truth and its response to the loss function. On the other hand, object detection is a multi-task problem. A sample's weights should balance among different tasks. If the detector trades its capacity for accurate classification and generates poor localization results, the mislocalized detection will harm average precision especially under high IoU criterion and vice versa.

Following the above idea, we propose a unified dynamic sample weighting network for object detection. It is a simple yet effective approach to learn sample-wise weights, which also balances between the tasks of classification and regression. Specifically, beyond the base detection network, we devise a sample weighting network to predict a sample's classification and regression weights. The network takes classification loss, regression loss, IoU and score as inputs. It serves as a function to transform a sample's current contextual feature into sample weight. Our sample weighting network has been thoroughly evaluated on MS COCO [29] and Pascal VOC [11] datasets with various one-stage and two-stage detectors. Significant performance gains up to 1.8% have been consistently achieved by ResNet-50 [18] as well as a strong ResNeXt-101-32x4d [43] backbone. The ablation studies and analysis further verify the effectiveness of our network and unveil its internal process.

In summary, we propose a general loss function for object detection, which covers most region-based object detectors and their sampling strategies, and based on it we devise a unified sample weighting network. Compared to previous sample weighting approaches [3, 16, 19, 28], our approach has the following advantages: (i). It jointly learns sample weights for both classification task and regression task. (ii). It is data-dependent, which enables to learn soft weights for each individual sample from the training data. (iii). It can be plugged into most object detectors effortlessly and achieves noticeable performance gains without affecting the inference time.

2. Related Work

Region-based object detection can be mainly categorized into two-stage and one-stage approaches. The two-stage approaches, e.g., R-CNN [16], Fast R-CNN [15] and Faster R-CNN [36], consist of region proposal stage and region classification stage. Various region proposal techniques have been devised, such as selective search [39] and Region Proposal Network [36]. In the second stage, regions are classified into object categories and bounding box regression is performed simultaneously. Significant improvements have been made by new designed backbones [7, 9, 27], architectures [2, 4, 8], and individual building blocks [10, 20, 21, 31, 41]. Inspired by domain adaptation for recognition [33, 44], another line of research [1, 6, 24] focuses on learning robust and domain-invariant detectors based on two-stage approaches. In contrast, one-stage approaches including SSD [30] and YOLO [34] remove the region proposal stage and directly predict object categories and bounding box offsets. This simplicity gains faster speed at the cost of degradation of accuracy.

Our sample weighting network (SWN) is devised to boost general region-based object detectors. It can be easily plugged into the aforementioned object detectors without adding much training cost. In fact, it does not affect the inference at all, which makes our approach very practical.

Region sampling or weighting strategy plays an important role in the training of object detection models. Random sampling along with a fixed foreground-background ratio is the most popular sampling strategy for early object detection [15, 36]. However, not every sample plays equal importance to optimization. Actually, the majority of negative samples are easy to be classified. As a result, various hard example mining strategies have been proposed, including hard negative examples mining [16, 30], Online Hard Example Mining (OHEM) [37], and IoU guided sampling [2, 28]. Instead of making hard selection, Focal-Loss [28] proposes to assign soft-weights to samples, such that it reshapes the classification loss to down-weight "easy" samples and focus training on "hard" ones. However, some recent works [3, 42] notice "easy" samples may be also important. Prime sampling [3] and IoU-balanced loss [42] have been advanced to make "easy" samples more impor-

tant for loss function optimization.

Beyond various sample weighting approaches, we devise a general loss function formulation which represents most region-based object detectors with their various sampling strategies. Based on this formulation, we design a unified sample weighting network to adaptively learn individual sample weights. Rather than manually crafted based on certain heuristics [3, 28], our sample weighting network is directly learned from the training data. In addition, unlike most existing methods [19, 28] designed for classification or regression, our approach is able to balance the weights between the classification and regression tasks.

Multi-task sample weighting has two typical directions of function design. One capitalizes on a monotonically increasing function w.r.t. the loss value, such as AdaBoost [14] and hard example mining [32]. The other designs monotonically decreasing function w.r.t. the loss value, especially when the training data is noisy. For example, Generalized Cross-Entropy [46] and SPL [26] propose to focus more on easy samples. Recently, some learning-based approaches are proposed to adaptively learn weighting schemes from data, which eases the difficulty of manually tuning the weighting function [12, 22, 35, 38]. In the regime of multi-task learning, [23] proposes using homoscedastic task uncertainty to balance the weighting between several tasks optimally where the tasks with higher uncertainties are down-weighted during training.

3. A Unified Sample Weighting Network

3.1. Review of Sampling Strategies

In this section, we briefly review the training objectives and sampling strategies for object detection. Recent research on object detection including one-stage and two-stage object detectors follows a similar region-based paradigm. Given a group of anchors $a_i \in \mathcal{A}$, i.e., prior boxes, which are regularly placed on an image to densely cover spatial positions, scales and aspect ratios, we can summarize the multi-task training objective as follows:

$$L = \frac{1}{N_1} \sum_{\{i:a_i \in \mathcal{A}^{cls}\}} L_i^{cls} + \frac{1}{N_2} \sum_{\{i:a_i \in \mathcal{A}^{reg}\}} L_i^{reg}, \quad (1)$$

where L_i^{cls} (L_i^{reg}) is the classification loss (regression loss), and $\mathcal{A}^{cls}$ ($\mathcal{A}^{reg}$) denotes the sampled anchors for classification (regression). N_1 and N_2 are the number of training samples and foreground samples. The relation $\mathcal{A}^{reg} \subset \mathcal{A}^{cls} \subset \mathcal{A}$ holds for most object detectors. Now, let s_i^{cls} and s_i^{reg} be sample a_i's weights for the classification and regression losses respectively, we formulate a generalized loss function for both two-stage and one-stage detectors with various sampling strategies, by converting Eq. 1 to:

$$L = \frac{1}{N_1} \sum_{\{i:a_i \in \mathcal{A}\}} s_i^{cls} L_i^{cls} + \frac{1}{N_2} \sum_{\{i:a_i \in \mathcal{A}\}} s_i^{reg} L_i^{reg}, \quad (2)$$

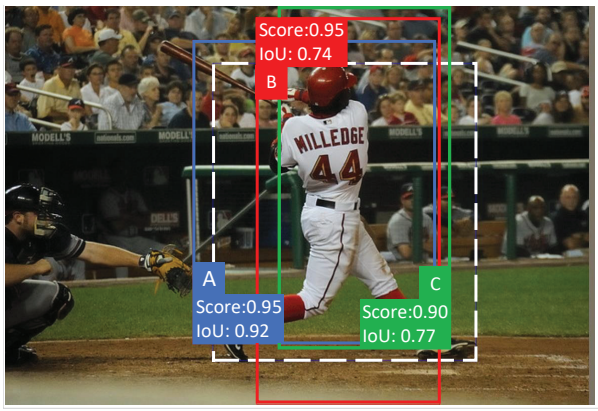

Figure 2: Training samples of Faster R-CNN after the first epoch. The dashed white box denotes the ground truth. A, B, C are three positive samples with different predicted scores and IoUs.

where $s_i^{cls} = I[a_i \in \mathcal{A}^{cls}]$ and $s_i^{reg} = I[a_i \in \mathcal{A}^{reg}]$. $I[\cdot]$ is the indicator function which outputs one when condition satisfied, otherwise zero. As a result, we can employ $S^{cls} = \{s_i^{cls}\}$ and $S^{reg} = \{s_i^{reg}\}$ to represent various existing sample strategies. Here, we reinterpret region sampling as a special case of sample weighting, which allows for soft sampling. In the following paragraph, we will briefly explain most popular sampling or weighting approaches under our general loss formulation.

3.2. Problems in Existing Sampling Approaches

RPN, Random Sampling and OHEM Region Proposal Network (RPN) classifies each sample into class-agnostic foreground or background class. Taking RPN as a data-driven sampling strategy, the classification weight for a_i is defined as: $s_i^{cls} = I[p(a_i) > \rho] * I[a_i \in \mathcal{A}_{NMS}]$ where ρ is the threshold to filter out samples with low foreground scores, and $\mathcal{A}_{NMS}$ is the anchor set after applying Non-Maximum-Suppression (NMS). Random Sampling uniformly selects n_p samples from $\mathcal{A}^P$ (positive) and n_n samples from $\mathcal{A}^N$ (negative), where n_p and n_n represent the required number of positive and negative samples, respectively. The classification weights for selected samples are assigned to be 1, while the rest to be 0. Instead of randomly sampling with equal probability, OHEM first ranks positive and negative samples separately in a monotonically decreasing order based on their loss values. Then the classification weights of top-n_p positive and top-n_n negative samples are assigned to be 1, and the rest to be 0. For all sampling, their samples' regression weights can be defined as $s_i^{reg} = I[s_i^{cls} = 1] * I[a_i \in \mathcal{A}^P]$.

Focal-Loss and KL-Loss Focal-Loss reshapes the loss function to down-weight easy samples and focus the training on hard ones. It can be regarded as assigning soft classification weight to each sample: $s_i^{cls} = (1 - p(a_i))^\gamma$ where $\gamma > 0$. And the regression loss are computed on all positive samples, $s_i^{reg} = I[a_i \in \mathcal{A}^P]$. KL-Loss re-weights the

regression loss depending on the estimated uncertainty σ_i^2: $s_i^{reg} = 1/\sigma_i^2$. The classification weights are the same as that of Random Sampling and OHEM.

Given a set of anchors $\mathcal{A} = \mathcal{A}^P \cup \mathcal{A}^N$, the goal of sample weighting is to find a weighting assignments S^{cls} and S^{reg} for better detection performance. Now, let us have a close inspection of two important components, i.e., NMS and mAP, to understand their particular roles in sample weighting. In general, the NMS filters cluttered bounding boxes by removing the boxes having relatively low scores. Taking the three boxes A, B, C in Figure 2 for example, C is suppressed during the inference due to its relatively lower score compared with A and B. In contrast, when OHEM is applied, C will be selected for training due to its higher loss (lower score). Putting too much attention to "hard" examples like "C" may not be always helpful, because during the inference we also pursue a good ranking. Focal-Loss also faces a similar problem as it assigns the same classification weight to box A and B. But, given that the IoU of A with regard to the ground truth is higher than that of B, aiming at improving the score of A would potentially be more beneficial. This is because the mAP is computed at various IoU thresholds, which favors more precisely localized detection results. KL-Loss, on the other hand, assigns different sample weights for regression loss based on bounding box uncertainty, while it ignores re-weighting classification loss.

Given these drawbacks of existing methods, we propose to learn sample weights jointly for both classification and regression from a data-driven perspective. Briefly speaking, previous methods concentrate on re-weighting classification (e.g., OHEM & Focal-Loss) or regression loss (e.g., KL-Loss). But our approach jointly re-weights classification and regression loss. In addition, being different from mining "hard" examples in OHEM & Focal-Loss approaches, which have higher classification loss, our approach focuses on important samples, which could be "easy" ones as well.

3.3. Joint Learning for Sample Weighting

Inspired by a recent work on uncertainty prediction for multi-task learning [23], we reformulate the sample weighting problem in a probabilistic format, and measure the sample importance via the reflection of uncertainties. We demonstrate that our proposed method enables the sample weighting procedure to be flexible and learnable via deep learning. Note that our work differentiates from [23], because our probabilistic modeling addresses not only the sample wise weighting, but also the balance between classification and localization tasks. Yet, the work [23] only considers the multi-task setting where all training samples share the same weights.

The object detection objective can be decomposed into regression and classification tasks. Given the i^{th} sample, we start by modeling the regression task as a Gaussian likelihood, with the predicted location offsets as mean and a standard deviation σ_i^{reg}:

$$p(gt_i|a_i^*) = \mathcal{N}(a_i^*, {\sigma_i^{reg}}^2), \tag{3}$$

where vector gt_i represents the ground truth bounding box coordinates, and a_i^* is the estimated bounding box coordinates. In order to optimize the regression network, we maximize the log probability of likelihood:

$$\log p(gt_i|a_i^*) \propto -\frac{1}{{\sigma_i^{reg}}^2}||gt_i - a_i^*||_2^2 - \log \sigma_i^{reg}, \tag{4}$$

By defining $L_i^{reg} = ||gt_i - a_i^*||_2^2$, multiplying Eq. 4 with -1 and ignoring the constant, we obtain the regression loss:

$$L_i^{reg*} = \frac{1}{{\sigma_i^{reg}}^2} L_i^{reg} + \lambda_2 \log \sigma_i^{reg}, \tag{5}$$

where λ_2 is a constant value absorbing the global loss scale in detection objective. By writing $1/{\sigma_i^{reg}}^2$ as s_i^{reg}, Eq. 5 can be roughly viewed as a weighted regression loss with a regularization term preventing the loss from reaching trivial solutions. As the deviation increases, the weight on L_i^{reg} decreases. Intuitively, such weighting strategy places more weights on confident samples and penalizes more on mistakes made by these samples during training. For classification, the likelihood is formulated as a softmax function:

$$p(y_i|a_i^*) = softmax(y_i, \frac{1}{t_i} p(a_i^*)), \tag{6}$$

where the temperature t_i controls the flatness of the distribution. $p(a_i^*)$ and y_i are the unnormed predicted logits and ground truth label of a_i^*, respectively. The distribution of $p(y_i|a_i^*)$ is in fact a Boltzmann distribution. To make its form consistent with that of the regression task, we define $t_i = 1/{\sigma_i^{cls}}^2$. Let $L_i^{cls} = -\log softmax(y_i, p(a_i^*))$, the classification loss is approximated by:

$$L_i^{cls*} = \frac{1}{{\sigma_i^{cls}}^2} L_i^{cls} + \lambda_1 \log \sigma_i^{cls}, \tag{7}$$

Combining weighted classification loss Eq. 7 and weighted regression loss Eq. 5 yields the overall loss:

$$\begin{aligned} L_i =& L_i^{cls*} + L_i^{reg*} \\ =& \frac{1}{{\sigma_i^{cls}}^2} L_i^{cls} + \frac{1}{{\sigma_i^{reg}}^2} L_i^{reg} + \lambda_1 \log \sigma_i^{cls} + \lambda_2 \log \sigma_i^{reg}, \end{aligned} \tag{8}$$

Note that directly predicting ${\sigma_i^{\cdot}}^2$ brings implementation difficulties since ${\sigma_i^{\cdot}}^2$ is expected to be positive and putting ${\sigma_i^{\cdot}}^2$ in the denominator position has the potential danger of division by zeros. Following [23], we instead predict $m_i^{\cdot} := log(\sigma_i^{\cdot})$, which makes the optimization more numerically stable and allows for unconstrained prediction output. Eventually, the overall loss function becomes:

$$\begin{aligned} L_i =& exp(-2 * m_i^{cls}) L_i^{cls} + \lambda_1 m_i^{cls} \\ &+ exp(-2 * m_i^{reg}) L_i^{reg} + \lambda_2 m_i^{reg}, \end{aligned} \tag{9}$$

Theoretic analysis. There exist two opposite sample weighting strategies for object detector training. On the one

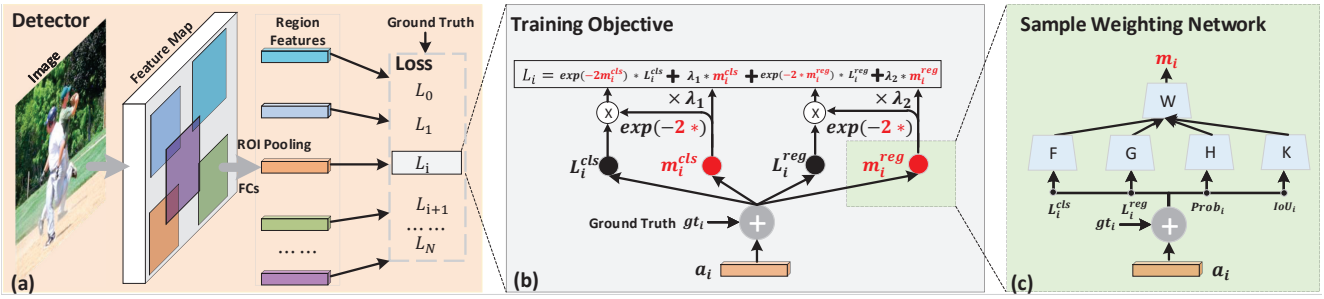

Figure 3: The framework for our Sample Weighting Network (SWN). (a) The general framework for a two-stage detector (it can be replaced with one-stage detector). In the forward pass, each sample is compared with its ground truth. The classification and regression losses are computed. In the backward pass, the loss of all samples are averaged to optimize the model parameters. (b) The break down of loss function which supervises the base detection network and SWN. The gradient can be backpropagated to the detection network and sample weighting network. (c) depicts the SWN design. It absorbs L_i^{cls}, L_i^{reg},$Prob_i$,IoU_i as input and generates weights for each sample.

hand, some prefer "hard" samples, which can effectively accelerate the training procedure via a more significant magnitude of loss and gradient. On the other hand, some believe that "easy" examples need more attention when ranking is more important for evaluation metric and the class imbalance problem is superficial. However, it is usually not realistic to manually judge how hard or noisy a training sample is. Therefore, involving *sample level* variance as in Eq. 5 introduces more flexibility, as it allows adapting the sample weights automatically based on the effectiveness of each sample feature.

Taking derivatives of Eq. 5 with respect to the variance σ_i^{reg}, equating to zero and solving (assuming $\lambda_2 = 1$) , the optimal variance value satisfies $\sigma_i^{reg,*2} = L_i^{reg}$. Plugging this value back into Eq. 5 and ignoring constants, the overall regression objective reduces to $\log L_i^{reg}$. This function is a concave non-decreasing function that heavily favors $L_i^{reg} = ||gt_i - a_i^*||_2^2 \rightarrow 0$, while it applies only soft penalization for *large* L_i^{reg} values. This makes the algorithm robust to outliers and noisy samples having large gradients that potentially degrade the performance. This also prevents the algorithm focusing too much on hard samples where L_i^{reg} is drastically large. In this way, the regression function Eq. 5 favors a selection of samples having large IoUs as this encourages a faster speed that drives the loss towards minus infinity. This, in turn, creates an incentive for the feature learning procedure to *weigh more* on these samples, while samples having relatively smaller IoUs still maintain a modest gradient during the training.

Note that we have different weights $(\exp(-2*m_i^{cls})$ and $(\exp(-2*m_i^{reg})$ tailored for each sample. This is critical for our algorithm as it allows to adjust the multi-task balance weight at a sample level. In the next section, we describe how the loss function effectively drives the network to learn useful sample weights via our network design.

3.4. Unified Sample Weighting Network Design

Figure 3 shows the framework of our Sample Weighting Network (SWN). As we can see, the SWN is a sub-network of the detector supervised by detection objective, which takes some input features to predict weights for each sample. Our network is very simple, which consists of two levels of Multiple Layer Perception (MLP) networks as shown in Figure 3 (c). Instead of directly using the sample's visual feature, which actually misses the information from the corresponding ground truth, we design four discriminative features from the detector itself. It leverages the interaction between the estimation and the ground truth i.e., the IoU and classification score, because both classification and regression losses inherently reflect the prediction uncertainty to some extent.

More specifically, it adopts the following four features: the classification loss L_i^{cls}, the regression loss L_i^{reg}, IoU_i and $Prob_i$, respectively, as an input. For negative samples, the IoU_i and $Prob_i$ are set to 0. Next, we introduce four functions F, G, H and K to transform the inputs into dense features for a more comprehensive representation. These functions are all implemented by the MLP neural networks, which are able to map each one dimension value into a higher dimensional feature. We encapsulate those features into a sample-level feature d_i:

$$d_i = concat(F(L_i^{cls}); G(L_i^{reg}); H(IoU_i); K(Prob_i)), \tag{10}$$

In the upcoming step, the adaptive sample weight m_i^{cls} for classification loss and m_i^{reg} for regression loss are learned from the sample feature d_i, as follows:

$$m_i^{cls} = W_{cls}(d_i) \text{ and } m_i^{reg} = W_{reg}(d_i), \tag{11}$$

where W_{cls} and W_{reg} represent two separate MLP networks for classification and regression weight prediction.

As shown in Figure 3, our SWN has no assumption on the basic object detectors, which means it can work with most region-based object detectors, including Faster R-CNN, RetinaNet, and Mask R-CNN. To demonstrate the generalization of our method, we make minimal modifications to the original frameworks. Faster R-CNN consists of region proposal network (RPN) and Fast R-CNN network. We leave the RPN unchanged and plug the sample weighting network into the Fast R-CNN branch. For each sample,

we firstly compute L_i^{cls}, L_i^{reg}, IoU_i, and $Prob_i$ as the inputs to the SWN. The predicted weights $\exp(-2 * m_i^{cls})$ and $\exp(-2 * m_i^{reg})$ are then inserted into Eq. 9 and the gradient is backpropagated to the base detection network and sample weighting network. For RetinaNet, we follow a similar process to generate the classification and regression weights for each sample. As Mask R-CNN has an additional mask branch, we include another branch into sample weighting network to generate adaptive weights for mask loss, where the classification, bounding box regression and mask prediction are jointly estimated. In order to match the additional mask weights, we also add the mask loss as an input to the sample weighting network.

In our experiments, we find that the predicted classification weights are not stable since the uncertainties among negative samples and positive samples are much more diverse than that of regression. Consequently, we average the classification weights of positive samples and negative samples separately in each batch, which can be viewed as a smooth version of weight prediction for classification loss.

4. Experiments

We conducted thorough experiments on the challenging MS COCO [29] and Pascal VOC [11] datasets and evaluated our method with both one-stage and two-stage detectors.

4.1. Datasets and Evaluation Metrics

MS COCO [29] contains 80 common object categories in everyday scenes. Following the common practice, we used the *train2017* split for training. It has 115k images and 860k annotated objects. We tested our approach as well as other compared methods on COCO *test-dev* subset. Since the labels of *test-dev* are not publicly available, we submitted all results to the evaluation server for evaluation. Yet all ablation experiments are evaluated on the *val2017* subset which contains 5k images. Pascal VOC [11] covers 20 common categories in everyday life. We merged the *VOC07 trainval* and *VOC12 trainval* split for training and evaluated on *VOC07 test* split. Our evaluation metric is the standard COCO-style mean Average Precision (mAP) under different IoU thresholds, ranging from 0.5 to 0.95 with an interval of 0.05. It reflects detection performance under various criteria and favors high precisely localized detection results.

4.2. Implementation Details

We implemented our methods based on the publicly available mmdetection toolbox[5]. In our experiments, all models were trained end-to-end with 4 Tesla P40 GPUs (each GPU holds 4 images) for 12 epochs, which is commonly referred as 1x training schedule. The base detection networks excluding the SWN is trained with stochastic gradient descent (SGD). The initial learning rate was set to 0.02 and decreased by 0.1 after epoch 8 and 11. For the sample weighting network, we adopted Adam [25] with 0.001 learning rate and followed the same learning rate decay schedule as base detection network. The weight decay of 0.0001 was used for both optimizers. Other hyperparameters closely follow the settings in mmdetection unless otherwise specified. We initialized the weights of FC layers in the SWN with Gaussian distribution. The standard deviation and mean were set to 0.0001 and 0, and thus the predicted weights are nearly uniform across samples at the beginning of training. We also enforced the predicted weights to fall into the range of $[-2, 2]$ by clipping the values out of bounds, which stabilizes the training in practice. Faster R-CNN, Mask R-CNN and RetinaNet are chosen as the representative two-stage and one-stage detectors. Two classical networks, ResNet-50 and ResNext-101-32x4d are adopted as backbones and FPN is used by default. *Please note that our method is fairly general and thus not limited to the aforementioned detectors and backbones.* In fact, it is applicable to any two-stage and one-stage detectors and is transparent to the choice of backbone networks.

4.3. Results

As discussed, our sample weighting network (SWN) can be applied to any region-based object detector. To verify the effectiveness of our method for performance boosting, we evaluated it thoroughly on Faster R-CNN, Mask R-CNN and RetinaNet (one of the latest one-stage detectors performing better than SSD) with two backbones ResNet-50 and ResNeXt-101-32x4d. Table 1 shows the results on COCO *test-dev* in terms of Average Precision (AP). Thanks to the proposed SWN, all detectors have achieved consistent performance gains up to 1.8%. Especially, the boost to RetinaNet is very impressive because it already has a strong sample weighting strategy. All improvements indicate that our SWN is complementary to the detectors' internal sample weighting strategies. In addition, from column AP_S, AP_M and AP_L (AP results for small, medium and large objects respectively) , we notice that our weighting strategy works better for "large" objects. Furthermore, we can infer from the results that the AP boosts are larger at higher IoU.

It is worth mentioning that SWN only affects the detector training with minimal extra cost. As an example, adding SWN to "Faster R-CNN + ResNet-50" detector only increased the training time from 1.009s to 1.024s per iteration and parameters from 418.1M to 418.4M. More importantly, since the inference is exactly the same, our approach does not add any additional cost to the test, which makes our sampling strategy more practical.

We also conducted similar evaluations on the PASCAL VOC 2007 dataset. The experimental reports are summarized in Table 2. In terms of AP, our approach further demonstrates its effectiveness on performance improvements. According to the gains on both popular benchmark

Table 1: Results of different detectors on COCO *test-dev*.

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage detectors							
Faster R-CNN	ResNet-50	36.7	58.8	39.6	21.6	39.8	44.9
Faster R-CNN	ResNeXt-101	40.3	62.7	44.0	24.4	43.7	49.8
Mask R-CNN	ResNet-50	37.5	59.4	40.7	22.1	40.6	46.2
Mask R-CNN	ResNeXt-101	41.4	63.4	45.2	24.5	44.9	51.8
Faster R-CNN w/ SWN	ResNet-50	**38.5**$_{\uparrow 1.8}$	58.7	42.1	22.0	41.3	48.2
Faster R-CNN w/ SWN	ResNeXt-101	**41.4**$_{\uparrow 1.1}$	61.9	45.3	24.1	44.7	52.0
Mask R-CNN w/ SWN	ResNet-50	**39.0**$_{\uparrow 1.5}$	58.9	42.7	21.9	42.1	49.2
Mask R-CNN w/ SWN	ResNeXt-101	**42.5**$_{\uparrow 1.1}$	64.1	46.6	24.8	46.0	53.5
Single-stage detectors							
RetinaNet	ResNet-50	35.9	56.0	38.3	19.8	38.9	45.0
RetinaNet	ResNeXt-101	39.0	59.7	41.9	22.3	42.5	48.9
RetinaNet w/ SWN	ResNet-50	**37.2**$_{\uparrow 1.3}$	55.8	39.8	20.6	40.1	46.2
RetinaNet w/ SWN	ResNeXt-101	**40.8**$_{\uparrow 1.8}$	60.1	43.8	23.2	44.0	51.1

Table 2: Results of different detectors on VOC2007 *test*.

Method	Backbone	AP
Two-stage detectors		
Faster R-CNN	ResNet-50	51.0
Faster R-CNN	ResNeXt-101	54.2
Faster R-CNN w/ SWN	ResNet-50	**52.5**$_{\uparrow 1.5}$
Faster R-CNN w/ SWN	ResNeXt-101	**56.0**$_{\uparrow 1.8}$
Single-stage detectors		
RetinaNet	ResNet-50	52.0
RetinaNet	ResNeXt-101	55.3
RetinaNet w/ SWN	ResNet-50	**53.4**$_{\uparrow 1.4}$
RetinaNet w/ SWN	ResNeXt-101	**56.8**$_{\uparrow 1.5}$

datasets, we can believe our SWN can consistently boost the performance of region-based object detectors.

Figure 4 demonstrates some qualitative performance comparisons between RetinaNet and RetinaNet+SWN on COCO dataset. Following a common threshold of 0.5 used for visualizing detected objects, we only illustrate a detection when its score is higher than the threshold. As we can see, some so-called "easy" objects such as a child , a coach, a hot dog and so on, which are missed by RetinaNet, have been successfully detected by the boosted RetinaNet with SWN. We conjecture that original RetinaNet may concentrate too much on "hard" samples. As a result, the "easy" samples get less attention and make less contributions to the model training. The scores for these "easy" examples have been depressed, which results in the missing detections. The purpose of Figure 4 is not to show the "bad" of RetinaNet in score calibration, because the "easy" ones can be detected anyway when decreasing the threshold. Figure 4 actually illustrates that unlike RetinaNet, SWN doesn't weigh less on "easy" examples.

There is another line of research, which aims to improve bounding box regression. In other words, they attempt to optimize the regression loss by learning with IoU as the supervision or its combination with NMS. Based on the Faster R-CNN + ResNet-50 + FPN framework, we make a comparison on COCO *val2017* as shown in Table 3. The performance comparison shows that both our SWN and its extension SWN+Soft-NMS outperform the IoU-Net and IoU-Net+NMS. It further confirms the advantages of learning sample weights for both classification and regression.

Table 3: Performance comparisons with IoU-based approaches.

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline	36.4	58.4	39.1	21.6	40.1	46.6
IoU-Net [21]	37.0	58.3	-	-	-	-
IoU-Net+NMS [21]	37.6	56.2	-	-	-	-
SWN	38.2	58.1	41.6	21.3	41.7	50.2
SWN + Soft-NMS	**39.2**	58.6	43.3	22.3	42.6	51.1

Table 4: Effectiveness of each component.

CLS	REG	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
		36.4	58.4	39.1	21.6	40.1	46.6
✓		36.7$_{\uparrow 0.3}$	58.7	39.5	21.2	40.2	47.9
	✓	37.0$_{\uparrow 0.6}$	56.6	40.1	21.2	40.4	47.9
✓	✓	**38.2**$_{\uparrow 1.8}$	58.1	41.6	21.3	41.7	50.2

4.4. Ablation Study and Analysis

For a better understanding to our SWN, we further conducted a series of ablation studies on COCO *val2017* using Faster R-CNN + ResNet-50 as our baseline.

The first group of experiments we did is to verify how well our approach works for each individual task, i.e., object classification (CLS) and regression (REG). Table 4 shows the detailed results. If a component is selected, it means our weighting strategy has been applied to it. The results clearly demonstrate that when the sample weighting is applied to only one task, the performance boost is trivial. Nonetheless, jointly applying it to both tasks can achieve a significant performance improvement of 1.8%. This observation is consistent with the goal of our SWN design.

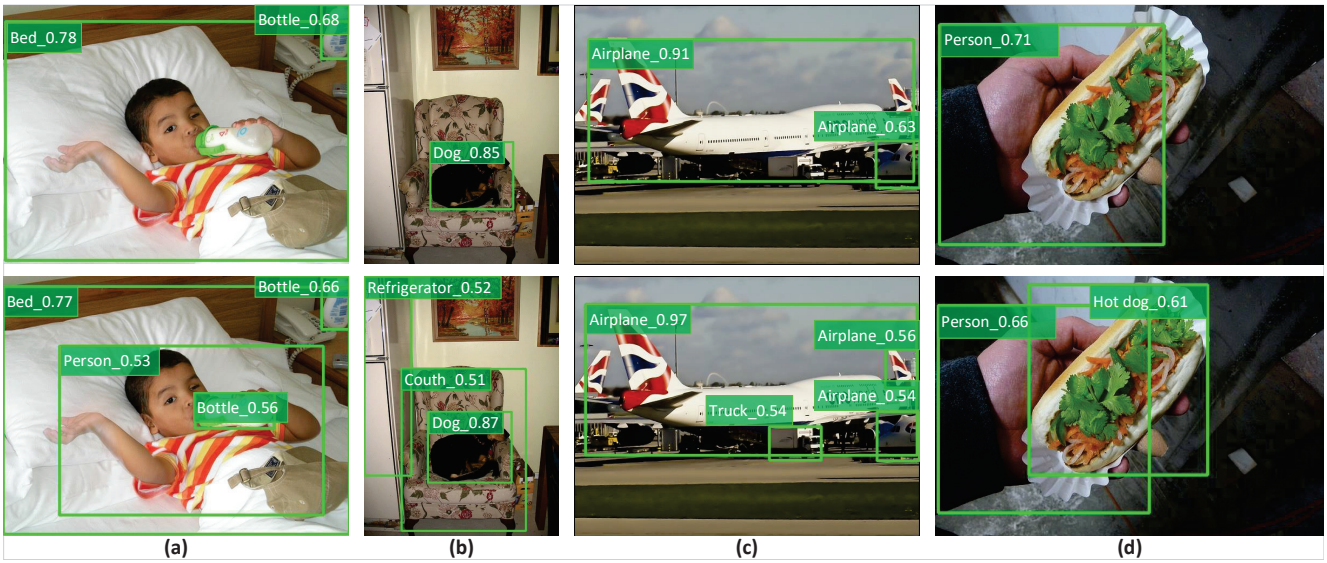

Figure 4: Examples of detection results of RetinaNet (first row) and RetinaNet w/SWN (second row). RetinaNet missed detecting some "easy" objects such as a child, a coach, a hot dog, and etc., which have been successfully detected by the boosted RetinaNet with SWN.

Table 5: Performance comparisons by varying λ.

λ	0.1	0.3	0.5	0.7	1.0
AP	29.3	37.4	**38.2**	37.9	37.2

There are two regularization hyperparameters (i.e., λ_1 and λ_2) in our loss function. In this set of experiments, we assigned various values to these parameters to check how sensitive our approach is to different regularization magnitudes. In our implementation, two parameters always share the same value. Table 5 illustrates the comparisons. It shows the results are relatively stable when λ lies in the range of 0.3 and 0.7, and achieves best performance at 0.5.

To understand the learning process, we draw the distribution of classification loss over samples at different IoUs as shown in Figure 5. We picked up the data from two training epochs to derive the distributions for both Baseline and SWN. The x-axis represents samples at a certain IoU with ground truth. Samples with higher IoUs shall have less uncertainties and thus higher weights to be considered by the loss optimization. There are two observations from the distributions. First, during the optimization process, the classification loss will draw more attentions to "easy" samples (i.e., the ones with high IoU values). Second, our approach generally put more weights to samples with high IoU values when computing the loss. All observations are consistent with our previous analysis of SWN.

5. Conclusion

We have demonstrated that the problem of sample weighting for region-based object detection is both data-dependent and task-dependent. The importance of a sample

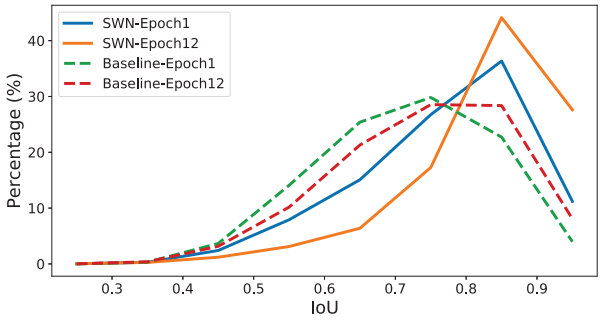

Figure 5: Classification loss distribution of positive samples with different IoUs. Higher IoUs mean easier samples. Y-axis denotes the percentage of weighted loss. For example, percentage=20% at IoU=0.85 with SWN-Epoch12 means the the losses of samples whose IoUs fall between 0.8 and 0.9 take up 42% of total loss.

to detection optimization is also determined by its uncertainties shown in two correlated classification and regression losses. We derive a general principled loss function which can automatically learn sample-wise task weights from the training data. It is implemented with a simple yet effective neural network, which can be easily plugged into most region-based detectors without additional cost to inference. The proposed approach has been thoroughly tested on different datasets, and consistent performance gains up to 1.8% have been observed. Some qualitative results clearly illustrate that our approach can detect some "easy" objects which are missed by other detectors. In future work, we will work on a complete explanation of this phenomenon. In addition, we can continue to improve our approach such that it can deal with "hard" and "easy" samples more smartly at different optimization phrases.

References

[1] Qi Cai, Yingwei Pan, Chong-Wah Ngo, Xinmei Tian, Lingyu Duan, and Ting Yao. Exploring object relation in mean teacher for cross-domain detection. In *CVPR*, 2019.

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.

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

[4] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *CVPR*, 2019.

[5] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[6] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild. In *CVPR*, 2018.

[7] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Chunhong Pan, and Jian Sun. Detnas: Neural architecture search on object detection. In *NeurIPS*, 2019.

[8] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NeurIPS*, 2016.

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

[10] Jiajun Deng, Yingwei Pan, Ting Yao, Wengang Zhou, Houqiang Li, and Tao Mei. Relation distillation networks for video object detection. In *ICCV*, 2019.

[11] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[12] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. *ICLR*, 2018.

[13] Pedro F Felzenszwalb, Ross B Girshick, and David McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010.

[14] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *JCSS*, 1997.

[15] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.

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

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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

[20] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *CVPR*, 2018.

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

[22] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *ICML*, 2018.

[23] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

[24] Mehran Khodabandeh, Arash Vahdat, Mani Ranjbar, and William G Macready. A robust learning approach to domain adaptive object detection. In *ICCV*, 2019.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[26] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NeurIPS*, 2010.

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

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

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

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

[31] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and Junjie Yan. Grid r-cnn. In *CVPR*, 2019.

[32] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[33] Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. Transferrable prototypical networks for unsupervised domain adaptation. In *CVPR*, 2019.

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

[35] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *ICML*, 2018.

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

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

[38] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. *NeurIPS*, 2019.

[39] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 2013.

[40] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[41] Jiaqi Wang, Kai Chen, Shuo Yang, Chen Change Loy, and Dahua Lin. Region proposal by guided anchoring. In *CVPR*, 2019.

[42] Shengkai Wu and Xiaoping Li. Iou-balanced loss functions for single-stage object detection. *arXiv preprint arXiv:1908.05641*, 2019.

[43] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[44] Ting Yao, Yingwei Pan, Chong-Wah Ngo, Houqiang Li, and Tao Mei. Semi-supervised domain adaptation with subspace learning for visual recognition. In *CVPR*, 2015.

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

[46] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 2018.