Ph.D. DISSERTATION

Improving Object Detection in Hard Conditions of Scale, Occlusion and Label

크기, 폐색 및 레이블에 대한 어려운 조건 하 물체 검출 개선

FEBRUARY 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Junhyug Noh

Improving Object Detection in Hard Conditions of Scale, Occlusion and Label

크기, 폐색 및 레이블에 대한 어려운 조건 하 물체 검출 개선

지도교수 김건희

이 논문을 공학박사학위논문으로 제출함

2019 년 12 월

서울대학교 대학원

컴퓨터공학부

노준혁

노준혁의 박사학위논문을 인준함

2019 년 12 월

위 원 장	장 병 탁	(인)
부위원장	김 건 희	(인)
위 원	유 승 주	(인)
위 원	전 병 곤	(인)
위 원	김 강 일	(인)

Abstract

Improving Object Detection in Hard Conditions of Scale, Occlusion and Label

Junhyug Noh
Department of Computer Science and Engineering
College of Engineering
Seoul National University

Object detection is one of the most essential and fundamental fields in computer vision. It is the foundation of not only the high-level vision tasks such as instance segmentation, object tracking, image captioning, scene understanding, and action recognition, but also the real-world applications such as video surveillance, self-driving car, robot vision, and augmented reality. Due to its important role in computer vision, object detection has been studied for decades, and drastically developed with the emergence of deep neural networks. Despite the recent rapid advancement, however, the performance of many detection models is limited under certain conditions. In this thesis, we examine three challenging conditions that hinder the robust application of object detection models and propose novel approaches to resolve the problems caused by the challenging conditions.

We first investigate how to improve the performance of detecting occluded objects and hard negatives in the domain of pedestrian detection. Occluded pedestrians are often recognized as background, whereas hard negative examples such as vertical objects are considered as pedestrians, which significantly degrades the detection performance. Since pedestrian detection often requires real-time processing, we propose a method that can alleviate two problems by improving a single-stage detection

model with the advantage in terms of speed. More specifically, we introduce an additional post-processing module that refines initial prediction results based on reliable classification of a person's body parts and grids of image.

We then study how to better detect small objects for general object classes. Although two-stage object detection models significantly outperform single-stage models in terms of accuracy, the performance of two-stage models on small objects is still much lower than human-level performance. It is mainly due to the lack of information in the features of a small region of interest. In this thesis, we propose a feature-level super-resolution method based on two-stage object detection models to improve the performance of detecting small objects. More specifically, by properly pairing input and target features for super-resolution, we stabilize the training process, and as a result, significantly improve the detection performance on small objects.

Lastly, we address the object detection problem under the setting of weak supervision. Particularly, weakly supervised object localization (WSOL) assumes there is only one object per image, and only provides class labels for training. For the absence of bounding box annotation, one dominant approach for WSOL has used class activation maps (CAM), which are generated through training for classification, and used to estimate the location of objects. However, since a classification model is trained to concentrate on the discriminative features, the localization results are often limited to small object region. To resolve this problem, we propose the methods that properly utilize the information in class activation maps.

Our proposed methods significantly improved the performance of base models on each benchmark dataset and achieved state-of-the-art performance in some settings. Based on the flexibility that is applicable to the various models, it is also expected to be applied to the more recent models, resulting in additional performance improvements.

Keywords: Object Detection, Computer Vision, Deep Learning, Weakly Supervised Object Localization, Pedestrian Detection

Student Number: 2015-30268

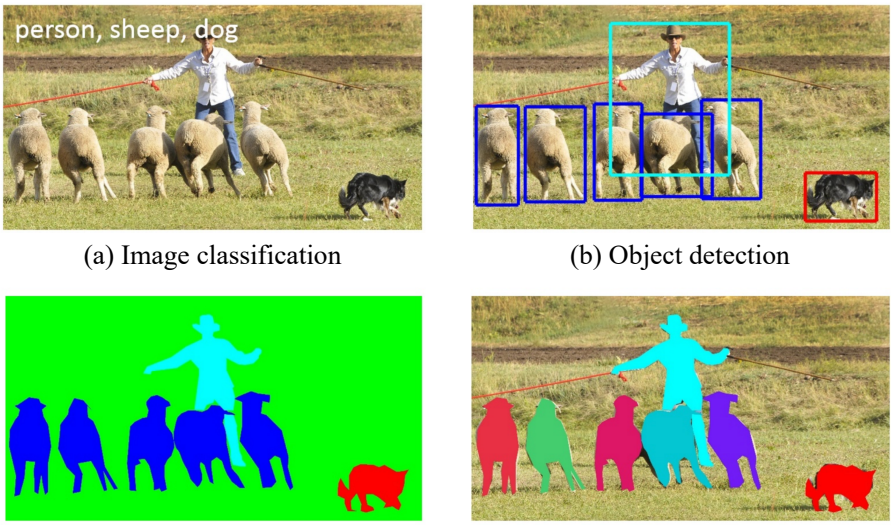
Contents

List of Figures

List of Tables

Chapter 1

Introduction



(a) Image classification (b) Object detection

(c) Semantic segmentation (d) Instance segmentation

Figure 1.1: Various types of object recognition [55]. Object detection is a task where a model learns to predict object classes as well as bounding boxes to locate objects as described in (b).

Object recognition is a technique to identify objects in images. Since there are various ways to *identify* objects, object recognition is often categorized into four different

tasks. A technique to classify the types of objects in an image (Figure 1.1a) and techniques to segment semantics or instances in pixel-wise (Figure 1.1c and 1.1d) are the examples of object recognition. Although the term, *object detection*, is often used to refer to object recognition, it actually refers to the technique to classify objects and locate them using bounding boxes as described in Figure 1.1b.[1]

Object detection has taken an important role as the foundation of various high-level vision tasks (*e.g.* instance segmentation, object tracking, image captioning, scene understanding and action recognition) as well as real world applications (*e.g.* video surveillance, self-driving car, robot vision and augmented reality). Due to the importance of its role, it has drawn high attention in computer vision community, and drastically developed in the recent years with the emergence of deep neural networks [33, 47]. Compared to the hand crafted features such as scale-invariance feature transform [57] and histogram of oriented gradient [15], these deep neural network algorithms automatically learn feature representations from data, which led major improvements in object detection.

Despite of its drastic development, however, some examples are still hard to be detected. These hard examples can be classified into hard positive and hard negative. In multi-class setting, positive and negative are defined based on a target class. If a class of an example is the same as the target class, the example is *positive*, otherwise, it is classified as *negative*. Suppose a simpler setting with a single-class. Under this setting, objects are positives whereas the background is negative. In practice, hard positives often cause false negatives, in other words, a model recognizes objects as the background. Similarly, hard negatives cause false positives which means a model recognizes the background as objects.

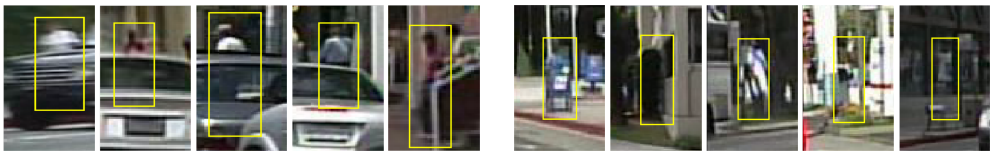
Then why are some examples *hard*? This is determined by a combination of a variety of factors, but we simplified and divided them into three main factors – (i) the capability of a detection model, (ii) the types of examples, and (iii) the amount

[1] All images in Figure 1.1 are credited to the slide: Abel Brown, *Introduction to Object Detection & Image Segmentation.* 2017.

of information provided during training. Depending on the capability of a detection model, the performance of detection varies. For instance, it is often harder to detect objects using hand crafted features than using deep neural networks, which implies the capability of a model takes an important role. Although the capacity of a model is high, however, it is still hard to detect some examples. One major reason would be the types of the examples. More specifically, objects that are occluded, small or blurred are harder to be detected than objects without any of them. Another major reason would be the amount of information provided for examples during training. For example, if data is partially labeled or there are only limited number of examples available for training, it can be extremely hard to detect examples which it would be easy to detect with enough number of training examples or full annotations.

Among the aforementioned three factors, the last two factors can be thought of as the conditions that determine the level of difficulties in detecting objects. In this thesis, we select some of the most challenging conditions of object detection task, and propose novel approaches to improve the detection performance under each condition.

Condition 1. Occluded Objects and Hard Negatives. We first discuss how to manage challenging conditions of pedestrian detection, one of the most important detection problems for various applications including autonomous driving and surveillance. Among many error sources of pedestrian detection, we are interested in two urgently critical issues: (i) *occlusion of target objects* (as false negative failure), and (ii) *confusion with hard negative examples* (as false positive failure). Occlusion is one of the most common and difficult problem in pedestrian detection, because real world scenes such as *street* are often crowded by many people and objects as shown in Figure 1.2a. Also, in the scenes of pedestrian detection, there are many hard negative examples like vertical structures, trees, and traffic lights as described in Figure 1.2b. Due to the characteristics of pedestrian detection which requires a real-time algorithm, single-stage models [67, 86, 68, 56, 25] are often preferred. In practice, because of the structural feature of single-stage models, the challenges originated from the aforementioned conditions are intensified.



(a) Occlusion as false negative failure (b) Hard negatives as false positive failure

Figure 1.2: Two critical issues of pedestrian detection problem. (i) occlusion of target objects as false negative failure cases, and (ii) confusion with hard negative examples as false positive failures.

Condition 2. Small Objects. One of the most common and well-studied problems of object detection is detecting small objects. Among two mainstream approaches for object detection: single-stage and two-stage, although two-stage object detection algorithms outperform the other approach in terms of accuracy, detection performance on small objects even using them is still far away from human-level performance. For instance, Huang *et al.* [41] show that mean Average Precision (mAP) scores of small objects are roughly 10 times lower than those of large objects. Figure 1.3 illustrates the features corresponding to the region of interest (RoI) that a two-stage model generates for a small object. For small proposals, the RoI pooling layer often extracts replicated feature vectors as inputs to a box predictor which eventually makes a prediction without enough detail inforamtion for small objects. Moreover, it is likely that the position of a RoI pooled feature and its actual position in the image are mismatched [40].

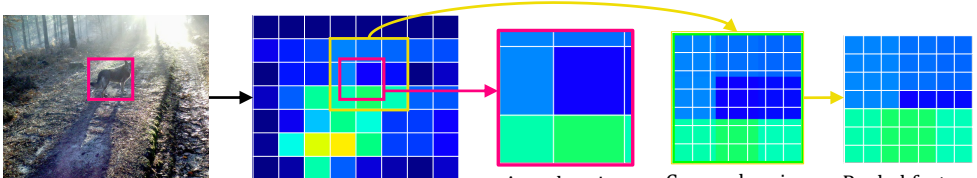


Figure 1.3: Features of a small object extracted by a two-stage model. The red region indicates the ground truth bounding box of the small object, *dog*, and yellow arrows illustrate RoI pooling operation [69] using the features in the red region.

Condition 3. Weakly Labeled Objects. The previous problems deal with difficulties in detecting objects under certain circumstances with full supervision from labels.

It is, however, particularly hard and expensive to annotate bounding boxes for objects due to their various locations, sizes and numbers. As a remedy to alleviate the issue, one that has drawn great attention in computer vision research is weakly supervised learning. In weak supervision setting, it only exploits the cheaper labels to solve both problems with and without labels. Particularly, weakly supervised object localization (WSOL) assumes that there is only one object per image and constraints only class labels are available for training. The conventional method to solve WSOL task is to estimate bounding boxes from class activation maps (CAM) generated from the last convolutional layer using the global average pooling (GAP) layer [97]. This approach, however, tends to predict smaller bounding boxes than the ground truth since only the discriminative regions are highly activated through classification training. For instance, according to the CAM in Figure 1.4, the classifier focuses on the head part of the *monkey* rather than the whole body, since the activations of the head is enough to correctly classify the image as *monkey*. Thus, the bounding box reduces to delineate the small high-activated region only.

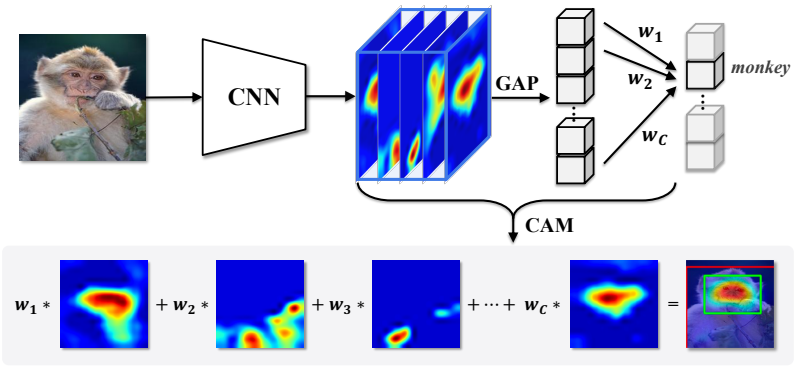


Figure 1.4: The limitation of the GAP-based CAM method. Since the whole object region is not required for classification, naive utilization of feature activations would result in small bounding box as the prediction of object location. For example, only the head part of the *monkey* is captured for localization because a classification model just needs to see the head part of the *monkey* to discriminate it with other classes.

1.1 Contributions

In this thesis, we introduce the approaches that improve existing methods under the aforementioned three difficult conditions. The contributions of our works as solutions for the conditions can be summarized as follows:

- **Solution 1. Part and Grid Classification Based Post-Refinement for Occluded Objects and Hard Negatives.** As solutions for two critical issues of pedestrian detection – (i) occlusion of objects, and (ii) confusion with hard negative examples, we propose a post-refinement method of initial output tensors of single-stage detection models. More specifically, we employ part confidence scores and grid classifier for occluded objects and hard negatives, respectively. Our proposed methods not only significantly improve the detection performance on pedestrian benchmarks, but also are trainable in an end-to-end manner with litter memory and time overhead. To the best of our knowledge, our proposed approach is the first approach that addresses both issues and is applicable to any single-stage detection models. We believe this robust applicability to any single-stage detection models is of a particular importance in the recent object detection research, because its progress is so fast that many new or updated models are published remarkably frequently.

  This work is published in:
  [62] **Junhyug Noh**, Soochan Lee, Beomsu Kim, Gunhee Kim. *Improving Occlusion and Hard Negative Handling for Single-Stage Pedestrian Detectors*. **CVPR 2018**.

- **Solution 2. Self-Supervised Feature Super-Resolution for Small Objects.** One approach to alleviate the aforementioned problem, the lack of information in the features of small RoIs, is to enhance the features using a super-resolution (SR) technique. We investigate how to improve a feature-level super-resolution technique, especially for small object detection, and discover its performance

can be significantly improved by (i) utilizing proper high-resolution target features as supervision signals for training of a SR model and (ii) matching the relative receptive fields of training pairs of input low-resolution features and target high-resolution features. We propose a novel feature-level super-resolution approach that not only correctly addresses these two desiderata but also is integrable with any proposal-based detectors with a feature pooling applied. In our experiments, our approach significantly improves the performance of Faster R-CNN [69] on three benchmarks. As a result, we achieve new state-of-the-art performance on a benchmark and highly competitive results on two other benchmarks.

This work is published in:
[61] **Junhyug Noh**, Wonho Bae, Wonhee Lee, Jinhwan Seo, Gunhee Kim. *Better to Follow, Follow to Be Better: Towards Precise Supervision of Feature Super-Resolution for Small Object Detection*. **ICCV 2019**.

- **Solution 3. Rectified Class Activation Mapping for Weakly Labeled Objects.** To resolve aforementioned problem of the CAM method described in Figure 1.4, recent studies [3, 84, 76, 10, 43, 94, 49, 85, 34, 11] have devised architectures to expand the activations of feature maps and obtain larger bounding boxes. Instead of endeavoring to expand activations by devising a new architecture, however, we propose a different approach – we focus on correctly utilizing most of the information that already exists in feature maps. We demonstrate the current CAM approaches for WSOL suffer from three fundamental issues: (i) the bias of GAP to assign a higher weight to a channel with a small activation area, (ii) negatively weighted activations inside the object regions and (iii) instability from the use of maximum CAM value as a thresholding reference. They collectively cause the problem that the localization prediction to be highly limited to the small region of an object. We propose three simple but robust techniques that alleviate the problems, including thresholded average pooling,

negative weight clamping and percentile as a thresholding standard. Our solutions are universally applicable to any CAM methods for WSOL, and improve their performance drastically. As a result, we achieve new state-of-the-art performance on the most widely used benchmarks for WSOL.

This work is published in:

[60] **Junhyug Noh***, Wonho Bae*, Gunhee Kim. *Rethinking Class Activation Maps for Weakly Supervised Object Localization*. preprint 2020. (*: equal contribution)

1.2 Thesis Organization

This thesis is organized as follows. In Chapter 2, we introduce previous studies that are related to the thesis. In Chapter 3–5, we introduce three different challenging conditions of object detection and present how to address each of them: detecting occluded objects and hard negatives for pedestrian detection (Chapter 3), detecting small objects for general object classes (Chapter 4), localizing objects only using weak supervision (Chapter 5). We conclude this thesis in Chapter 6.

Chapter 2

Related Work

This chapter provides the summery of previous studies that are highly related to this thesis, and briefly illustrates how our works have improved them. We first describes the most widely applied algorithms for object detection task (section 2.1). Then we introduce previous researches on object detection under challenging conditions in the order of (i) occluded objects and hard negatives (section 2.2), (ii) small objects (section 2.3), and (iii) objects localization without location annotations (section 2.4).

2.1 General Methods

Two-stage and single-stage methods are two dominant approaches in object detection task. We briefly introduce the principal algorithms of each approach.

Two-stage methods. Recent advances in object detection have been largely attributed to the successful application of convolutional neural networks (CNNs) to both region proposal and region classification. The R-CNN approach [27, 26, 69] has greatly improved the performance for a variety of object detection problems and is currently one of the best performing detection paradigms. This approach consists of the two stages of proposing regions and computing their confidences of object presence. Fast R-CNN [26] reduces the number of per-region operations by moving the

RoI extraction from the input image to the convolutional feature map. In addition, Faster R-CNN [69] introduces *Region Proposal Network* (RPN) that shares the convolutional layers with the overall detection network. As a slight different approach, R-FCN [14] generates a set of position-sensitive score maps through a fully convolutional network and performs per-region operations after RoI pooling from the score maps. Since per-region operations are small in number and do not have weights, it enables the network to operate much faster. In more recent years, object detection has been further advanced by Mask R-CNN [31] which segments instances simultaneously with detecting objects, and Cascade R-CNN [6, 7] which employ detectors for more than two stages.

Single-stage methods. YOLO [67] has brought another breakthrough in object detection research; it formulates the two stages of region proposal and classification into a single-stage regression problem to detect objects extremely fast. Since then, more advanced models based on this single-stage method are emerging, including SSD [56], DSSD [25], SqueezeDet+ [86], and YOLOv2 [68]. They all use a convolutional predictor to generate the final output tensor, and use anchors like the RPN to predict the offsets of boxes rather than coordinates. In addition, DSSD [25] generates a context feature map using the deconvolutional layer, which enables global information to be used to detect smaller objects. As with two-stage methods, single-stage methods have also drastically developed in the recent years. Especially, anchor-free methods such as CornerNet [45, 46], CenterNet [19], FoveaBox [44], FCOS [82], FSAF [98] have significantly outperformed the previous single-stage methods.

2.2 Methods for Occluded Objects and Hard Negatives

Methods for occluded objects. The part-based methods [58, 81] have been one of the most dominant approaches addressing the occlusion problem. Mathias *et al.* [58] propose the Franken-classifiers, consisting of a set of occlusion-specific classifiers using Integral Channel Features [16]. DeepParts [81] model constructs a set of data-driven part prototypes, trains a CNN classifier to detect each of them, and finally

explores their ensemble to improve the detection of occluded objects. Enzweiler *et al*. [20] leverage the features of intensity, depth and motion to build a part-based mixture-of-experts classification model. Ouyang *et al*. [64] propose a probabilistic framework that can predict well even with inaccurate scores of part detectors by modeling part visibility as latent variables. Later they [65] extend the probabilistic framework to represent the relations between the configurations estimated by single- and multi-pedestrian detectors. Tang *et al*. [79] develop a double-person detector and tracker that can detect multiple people that occlude one another, based on the DPM model [23].

One of the most relevant works to ours is the DeepParts [81] model, yet our approach has the following three contributions. First, our model can be plugged into any single-stage CNN architecture, whereas DeepParts is a stand-alone pedestrian detector. Second, our model is end-to-end learnable with any base networks, whereas DeepParts consists of multiple components that should be separately learned. For example, each semantic part of DeepParts has its own classification network, and the final score is obtained via additional linear SVM on the part detection scores. Finally, DeepParts uses 6 or 45 pre-defined semantic parts, whereas our approach does not require predefining semantic parts; instead, the best visibility patterns are directly learned from part confidence maps.

Methods for hard negative examples. False-positives due to hard negative examples account for a large portion of the errors in the pedestrian detection problem [92]. It is due to wrongly assigning a higher probability to a background region which looks like a person. However, for single-stage models, hard negative examples could be more harmful; the methods assume object candidates as anchors at every cell in a pre-fixed grid, and thus negative anchors are much more than positive anchors in their prediction. To resolve such a highly unbalanced distribution between positive and negative anchors, for example, SSD [56] and DSSD [25] select only three times of negative anchors (than positive ones) with the highest classification loss for training.

Recently, some state-of-the-art models [91, 18, 39] introduce additional post-refinement classifiers to reject hard negatives. For example, Zhang *et al*. [91] apply a boosted

forest classifier to the candidate boxes of pedestrians that are obtained by the RPN. Du *et al*. [18] exploit multiple neural networks in parallel for further refinement of pedestrian candidates obtained by the SSD. Compared to Du *et al*. [18] and Hu *et al*. [39], our approach has the following three contributions. First, our approach generates a set of grid confidence maps from multi-layer feature maps from which final detection scores are computed. This idea not only induces ensemble effect, but is also more robust against hard negatives that erroneously incur high detection confidence in a certain scale of a feature map. Second, we do not require pixel-level annotation for training, and use bounding box labels instead. Finally, our additional classifiers increase little inference time, and are also trainable with the overall networks in an end-to-end manner.

2.3 Methods for Small Objects

Methods using high-resolution images. One straightforward approach to effectively detect small objects is to generate high-resolution images as inputs to a detection model by increasing the resolution of the original input images. Hu *et al*. [38] and Singh *et al*. [75] use three different resolutions of input images to a shared CNN-based network for different sizes of objects. To detect small objects, they apply bilinear interpolation to obtain two times upsampled input images. Furthermore, to obtain higher quality upsampled images, Fookes *et al*. [24] use traditional super-resolution techniques to better recognize human faces. However, there are two potential problems of image-level super-resolution. First, super-resolution and detection models are often trained independently; the super-resolution model is trained to generate high-resolution images even for the parts that are not important for detection due to their independence. Second, the overall architecture can be too heavy as it takes enlarged super-resolved images as inputs, which may considerably increase inference time. Although Haris *et al*. [30] propose an end-to-end model that jointly trains super-resolution and detection models, it is still inefficient to perform super-resolution on large parts of images that are irrelevant to the detection task. Instead of super-

resolving the whole images, SOD-MTGAN [2] pools RoIs first and then train the super-resolution model using those pooled RoIs. Although their work resolves both problems by focusing only on RoIs, it still does not take the context information of RoIs into account.

Methods using high-resolution features. Due to the aforementioned distortion problem RoI pooling methods may create as described in Figure 1.3, some alternatives such as RoI Align [31] and Precise RoI pooling [42] have been proposed. Although these methods alleviate the limitations of RoI pooling method, the fundamental problem of small object features including lack of detail information still remains. To provide enough information, one notable approach for small object detection is Perceptual GAN [48] which which exploits super-resolved features of RoIs. Since it focuses on only the features of RoIs, it does not suffer from the two problems of image-level super-resolution. Moreover, since the features are extracted by the convolution with large receptive fields, the problem of SOD-MTGAN [2] is alleviated too. However, its super-resolution training can be unstable since it lacks direct supervision; there is no training pairs of low-resolution RoI features and their corresponding high-resolution features. Instead, it implicitly leverages the classification, localization and adversarial loss. For the image retrieval task, Tan *et al*. [78] add the feature-wise L_2 loss to train feature-level super-resolution model. They report that adding such stronger constraint helps the generative network produce better features with faster convergence. However, we observe that such direct supervision in [78] is not sufficient for object detection, since it may mislead the super-resolution process due to mismatch of the relative receptive fields between high and low-resolution features. In section 4.2, we elaborate this problem further.

Methods using context information. Many studies have empirically proved that the context information also helps detect small objects. As demonstrated in [56], the features from the top layers in CNNs are adequate to capture large objects but too coarse to detect small objects, while the features from the bottom layers contain too specific local information which is not useful for detecting large objects but useful for

small objects. Thus, many methods [4, 73, 52, 25, 88] employ additional layers to build context features from multiple layers. Another simple way to use context is to consider nearby regions too while RoI pooling. Hu *et al*. [38] extract surrounding regions along with RoIs to detect human faces since knowing the existence of human bodies in the nearby region is helpful. Relational information between objects has been also studied to enhance the detection model [36, 22, 12]. In [36], internal relational information among objects is used through an attention model. In [22], knowledge graphs from external sources are used. Also, [12] exploits a probabilistic model to capture contextual information of a scene and apply it to object recognition task. Lastly, several studies [9, 90, 96, 29] propose to use a mixture of convolution and atrous convolution layers to better segment small objects since atrous convolution layer covers larger receptive fields without losing resolution. Because of this trait, we also employ atrous convolution layers to match the relative receptive fields between high and low-resolution features. More detailed explanation is provided in section 4.2.

2.4 Methods for Weakly Labeled Objects

Due to the absence of location annotations, WSOL relies on the activations of feature maps to localize an object. To extract a bounding box of an object from feature maps, several approaches have been proposed. Oquab *et al*. [63] and Pinheiro *et al*. [66] propose to apply the max pooling and log-sum-exp layer respectively to the feature map from the last CNN layer to locate an object. Although the max pooling accurately tells where the most discriminative region of an object is, its localization is highly limited to that small region. Zhou *et al*. [97] use a GAP layer proposed in Lin *et al*. [51] as a replacement for the max pooling layer since the loss for average pooling benefits all the activated regions. To utilize a GAP layer, however, the last layer of a model has to be converted to a fully connected layer following a GAP layer, which is not the case for many of well-known classification CNNs [74, 32, 35, 77]. Because of this limitation, GradCAM [72] and GradCAM+ [8] propose gradient-based methods to obtain a CAM. Since a logit gradient with respect to the feature map activation is com-

putable in any architecture, the gradient-based CAM methods are applicable to any classification model with no modification of model structure. In spite of the robustness of these gradient-based methods, they are overwhelming in terms of the computation and memory cost without much improvement on the performance. Thus, the GAP-based CAM becomes a de facto standard approach to WSOL, including recent works such as [34, 95, 11, 89].

The major challenge of WSOL is to capture the whole region of an object rather than its most discriminative one in the image. Since the CNN backbone is trained for the objective of classification, the learned CAM is highly activated on the discriminative regions not on the whole regions of the object. As the localization is solely based on the CAM, expanding the activation beyond the discriminative region has been a major research topic for WSOL [3, 76, 10, 84, 43, 34, 94, 95, 11]. Bazzani *et al*. [3] perform object localization by finding the regions that leads to large drops of classification scores when they are masked out. Singh and Lee [76] propose the hide-and-seek (HaS) algorithm as an data augmentation technique for less discriminative parts. Choe *et al*. [10] improve the HaS by using the GoogLeNet Resize (GR) augmentation method. Wei *et al*. [84] propose an adversarial erasing method for the first time; it progressively erases the most discriminative parts detected by multiple classifiers and combines them for localization. Kim *et al*. [43] and SeeNet [34] also propose erasing methods but only in two phases. ACoL [94] design an end-to-end parallel adversarial architecture where two classifiers are learned to detect complementary regions via adversarial erasing feature maps. Choe *et al*. [11] propose an attention-based dropout layer (ADL) that randomly generates masks on feature maps, expecting the activations of feature maps to expand to less discriminative regions. As a slightly different approach, SPG [95] provides additional supervision to the earlier layers using the masks of highly discriminative regions from the latter layers. DANet [89] train intermediate layers using the different number of classes obtained from knowledge graphs of class hierarchy, expecting the expansion of activations to the common object regions.

Unlike previous methods, we aim at fully and correctly leveraging the information

obtained from classification networks. Thus, instead of endeavoring to obtain additional information from new model architecture, in this work, we focus on finding out which parts of the GAP-based CAM are fundamentally problematic and proposing simple and robust techniques to resolve them.

Chapter 3

Part and Grid Classification Based Post-Refinement for Occluded Objects and Hard Negatives

3.1 Overview

In this chapter, we aim our attention at *pedestrian detection* [17, 92, 93], which may be one of the most important detection problems for various applications, including autonomous driving and surveillance.

Among many error sources of pedestrian detection as Zhang *et al*. [92] systemically break down, we are interested in two critical issues: (i) *occlusion of target objects* (as false negative failure cases), and (ii) *confusion with hard negative examples* (as false positive failures). First, occlusion is one of key practical difficulties in pedestrian detection, because real world scenes like *street* are often crowded with many people and various objects; thus observation with occlusion is much more common than that without occlusion. Second, in the scenes for pedestrian detection, there are many hard negative examples like vertical structures, trees, and traffic lights, because of which, models detect a lot of false positives, and they amount to a large portion of overall errors.

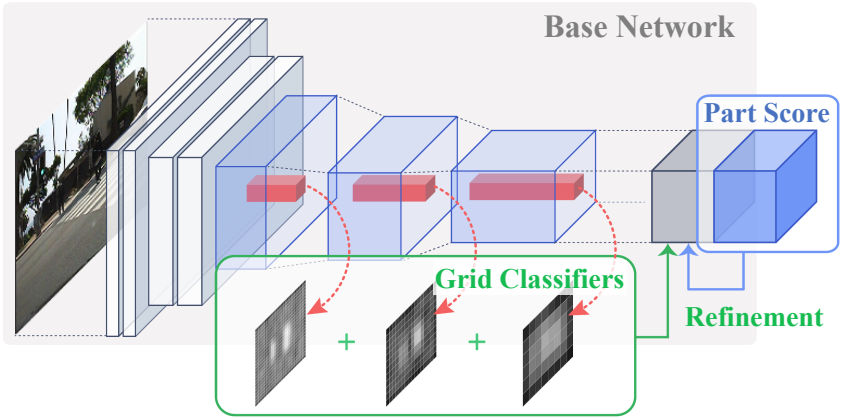


Figure 3.1: The overview of our poset-refinement system.

Our objective is to propose the approaches that address these two problems of occlusion and hard negative examples. Due to the characteristics of its applications which require real-time inference, single-stage detection algorithm is preferred for pedestrian detection. Therefore, our approaches are based on single-stage detection models. One of the key requirements is that the proposed methods should be general and flexible enough to be applicable to any single-stage detection models. We believe this requirement is of a particular importance in recent object detection research, because its progress is so fast that many new or updated models appear frequently. We integrate our approach with four single-stage models, SqueezeDet+ [86], YOLOv2 [68], SSD [56], and DSSD [25]. We empirically validate that our approach indeed improves the performance of those four models on Caltech pedestrian [17] and CityPersons [93] dataset. As shown in Figure 3.1, our approach involves two key ideas. For better occlusion handling, we propose to update the output tensors of single-stage models so that they include the information of part confidence scores, from which we obtain a final occlusion-aware detection score. For reducing the confusion with hard negative examples, we introduce average grid classifiers as post-refinement classifiers, trainable in an end-to-end manner without large time and memory overheads.

Our main contributions are two-fold:

(1) We propose an approach to address the two critical issues of pedestrian detec-

tion: (i) occlusion of objects, and (ii) confusion with hard negative examples. To the best of our knowledge, our approach is the first to be applicable to any single-stage detection models while addressing these two issues. As solutions, we propose to update output tensors of single-stage detection models to account for the information of part confidence scores, and introduce average grid classifiers for post-refinement, trainable in an end-to-end manner with little memory and time overhead (*e.g.* increase of 1–5 MB in memory and 1–2 ms in inference time).

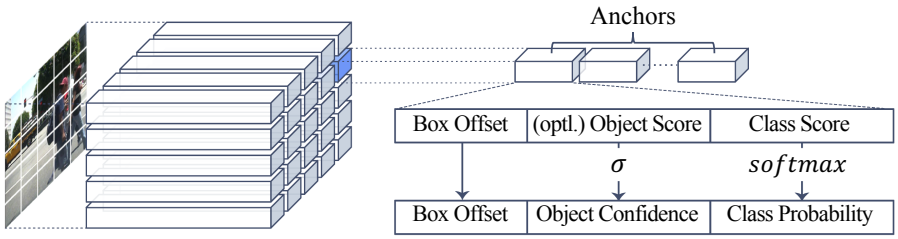
(2) We validate the flexibility and utility of our method on Caltech pedestrian [17] and CityPersons [93] dataset. First, we show that our approach is integrable with four single-stage models, SqueezeDet+ [86], YOLOv2 [68], SSD [56], and DSSD [25]. Second, we demonstrate that our approach indeed improves the performance of those four models for pedestrian detection. Moreover, in some heavy occlusion settings, our approach achieves the best reported performance on the datasets.

3.2 Our Approach

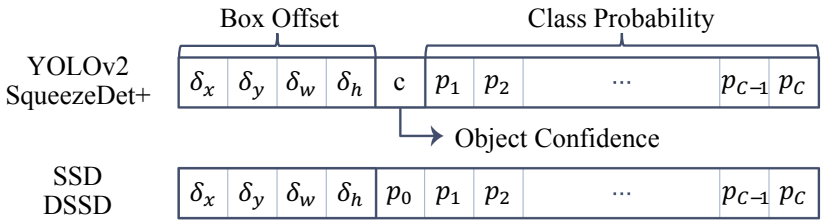
We first review the structure of the output tensors of four single-stage models, SqueezeDet+ [86], YOLOv2 [68], SSD [56], and DSSD [25], which are used as our base models (section 3.2.1). We then introduce our refinement methods for occluded objects (section 3.2.2) and hard negative examples (section 3.2.3) based on these base models.

3.2.1 A Unified View of Output Tensors

Most single-stage networks formulate the detection as a regression problem, and generate a tensor as prediction output [67, 68, 56, 25, 86]. As shown in Figure 3.2a, the width (W) and the height (H) of output tensors depend on the spatial grid of an input image, and the depth (K) depends on the number of anchors per grid. The prediction output per anchor is differently defined according to the model in Figure 3.2b. The box offset is defined by the position and scale between the ground truth $(x_{gt}, y_{gt}, w_{gt}, h_{gt})$ and its matched anchor $(x_i, y_j, w_k, h_k), i \in [1, W], j \in [1, H], k \in$



(a) Structure of the output tensor.



(b) Output formats of four methods per anchor.

Figure 3.2: A unified view of output tensors of four methods: YOLOv2, SqueezeDet+, SSD, and DSSD.

$[1, K]$. All the models use the scale parameters (δ_w, δ_h) to describe how different the scale is compared to that of an anchor:

$$\delta_{w,(ijk)} = \log\left(\frac{w_{gt}}{w_k}\right), \ \delta_{h,(ijk)} = \log\left(\frac{h_{gt}}{h_k}\right). \tag{3.1}$$

For the position parameters (δ_x, δ_y), YOLOv2 [67, 68] predicts the relative position of top-left corner in the grid with a bound of $[0, 1)$ in Eq.(3.2), whereas SqueezeDet+ [86], SSD [56], and DSSD [25] predict the relative position of center point to the anchor in Eq.(3.3).

$$\delta_{x,(ijk)} = \sigma\left(\frac{x_{gt} - x_i}{w_{grid}}\right), \delta_{y,(ijk)} = \sigma\left(\frac{y_{gt} - y_j}{h_{grid}}\right) \tag{3.2}$$

$$\delta_{x,(ijk)} = \frac{x_{gt} - x_i}{w_k}, \qquad \delta_{y,(ijk)} = \frac{y_{gt} - y_j}{h_k} \tag{3.3}$$

where σ is the sigmoid.

For the object likelihood, YOLOv2 and SqueezeDet+ define the confidence of object presence in Eq.(3.4), and follow the conditional probabilities of C object classes in Eq.(3.5). The final likelihood is obtained by multiplying the conditional probabilities

by the confidence.

$$c_{(ijk)} = P_{(ijk)}(\text{Object}) \times \text{IoU}^{gt}_{(ijk)}, \tag{3.4}$$

$$p_{m,(ijk)} = P_{(ijk)}(\text{Class} = m \mid \text{Object}), m \in [1, C]. \tag{3.5}$$

On the other hand, SSD and DSSD consider the background (*i.e.* absence of objects) as another class, and compute the likelihood of all $C + 1$ classes Eq.(3.6):

$$p_{m,(ijk)} = P_{(ijk)}(\text{Class} = m), \quad m \in [0, C]. \tag{3.6}$$

For pedestrian detection, there exists only one class of interest, *person* ($C = 1$); thus, a single value for object/class probability is necessary in the output per anchor for all models, and regard it as c.

Another difference between the models is which feature maps are used to generate output tensors. Table 3.1 shows the default shapes of output tensors for 640×480 input images. SSD and DSSD use multiple feature maps to regress output tensors. YOLOv2 has only one type of output, but it is created from concatenated feature maps, not from a single one.

Model	Shape of Output Tensors $\{W, H, K\}$
SqueezeDet+ [86]	$\{38, 28, 9\}$
YOLOv2 [68]	$\{20, 15, 9\}$
SSD [56]	$\{40, 30, 4\}$, $\{20, 15, 3\}$, $\{10, 8, 3\}$, $\{8, 6, 2\}$, $\{6, 4, 1\}$
DSSD [25]	$\{1, 1, 3\}$, $\{6, 4, 3\}$, $\{8, 6, 6\}$, $\{10, 8, 6\}$, $\{20, 15, 3\}$, $\{40, 30, 3\}$, $\{80, 60, 3\}$, $\{160, 120, 1\}$

Table 3.1: The shape of output tensors for a 640×480 image (W: width, H: height, K: number of anchors). In SSD and DSSD, output tensors come from multiple feature maps, and they are listed in a generation order.

3.2.2 Refinement for Occlusion Handling

Our key idea for occlusion handling is to divide the prediction confidence *by parts* rather than expressing it as a single value that existing single-stage networks do. While normal single-stage networks are likely to assign a low confidence to an occluded

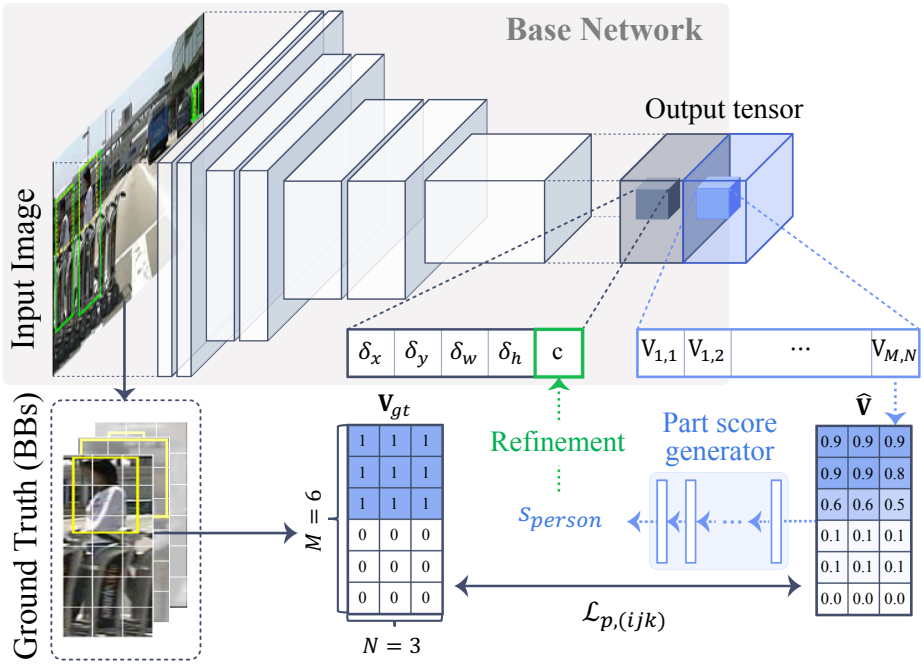


Figure 3.3: The overview of our occlusion handling method.

person due to the hidden parts, our model can leverage the confidences of visible *parts* of a body to correct the final detection confidence of a *person*.

We first introduce the concept of *part confidence map* denoted by $\mathbf{V}$, which is an $M \times N$ grid in the range of $[0, 1]$ (by applying a sigmoid function), as shown in Figure 3.3. The groundtruth for the part confidence map is generated as follows. We first identify a bounding box for a full-body person, and divide it as an $M \times N$ grid. For each cell $(m, n), m \in [1, M], n \in [1, N]$, we set $\mathbf{V}_{gt}(m, n) = 1$ if a pedestrian occupies more than τ_v times of area at the cell. In our experiments, we set $M = 6, N = 3, \tau_v = 0.4$.

Computing Occlusion-aware Detection Scores

For occlusion handling, we extend the output tensor to include the prediction of part confidence map $\widehat{\mathbf{V}}$ (See Figure 3.3). That is, the network predicts $\widehat{\mathbf{V}}$ as detection output, from which we compute a final occlusion-aware detection score of each anchor. We design two different methods: (i) a *max part score*, and (ii) a *soft part score*.

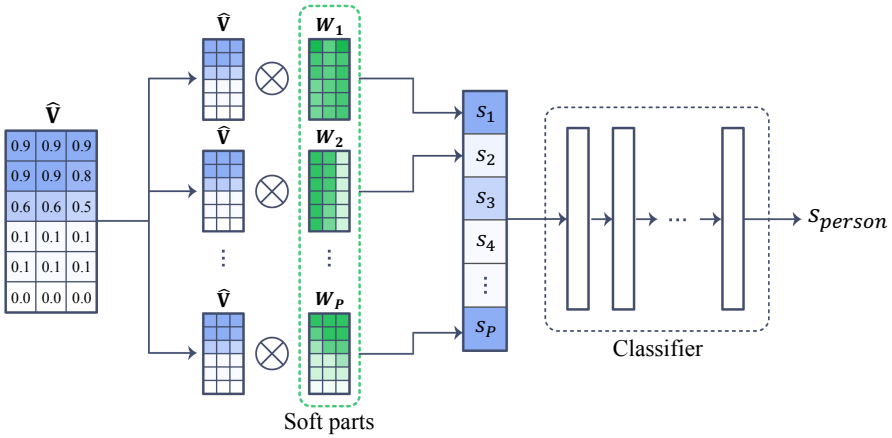


Figure 3.4: The generator module for the soft part score.

Max part score. One of the simplest ways to compute the final detection score is to apply the max pooling to a predicted part confidence map $\widehat{\mathbf{V}}$ (Figure 3.3). Its intuition is that if the score for a particular position is very high, it could be an occluded person whose confidence is high only at this position:

$$s_{\text{person}} = \max_{m,n} \widehat{\mathbf{V}}(m, n). \tag{3.7}$$

Soft part score. The approach of max part score has one limitation; it does not take into account the person occlusion patterns in real world. For example, in the Caltech pedestrian dataset [17], more than 97 % of occluded persons belong to only seven sets of occlusion patterns. As discussed in section 2.2, the DeepParts approach [81] thus defines a part pool containing representative semantic appearance of body parts, and decide the final score using a linear SVM with the score of those parts. However, DeepParts require an external classifier to compute a final detection score, and thus cannot be trained in an end-to-end manner.

Therefore, we propose an end-to-end learnable *soft part score* method, which is illustrated in Figure 3.4. We first define a P number of soft parts $\mathbf{W}_p \in \mathbb{R}^{M \times N}, p \in [1, P]$. We compute the interim part score s_p by element-wise dot product with a pre-

dicted part confidence map $\widehat{\mathbf{V}}$:

$$s_p = \sum_{m=1}^{M}\sum_{n=1}^{N}\left(\widehat{\mathbf{V}}(m,n)\cdot\mathbf{W}_p(m,n)\right) \tag{3.8}$$

Once computing $\mathbf{s} = [s_1, s_2, \cdots, s_P]$, the final score s_{person} is obtained via an MLP with one hidden and ReLU layer:

$$s_{\text{person}} = \sigma\left(\mathbf{w}_2^\top \max\left(\mathbf{0}, \mathbf{w}_1^\top \mathbf{s}\right)\right) \tag{3.9}$$

$\{\mathbf{W}_p\}_{p=1}^{P}$ and $\mathbf{w}_{1,2}$ are parameters to learn, and determined automatically from setting only the number of semantic parts P. The number of semantic parts depends on variability of occlusion patterns in the dataset, although it is fine to simply use a sufficiently large number, and we set $P = 45$. We test different configurations of MLPs, but the simple one in Eq.(3.9) performs the best.

Finally, we adjust the confidence per bounding box as the geometric mean of s_{person} and its initial confidence c.

$$c' = \sqrt{s_{\text{person}} c}. \tag{3.10}$$

Training Objective

Single-stage models used in this work have two types of losses: localization loss $\mathcal{L}_l$ and confidence loss $\mathcal{L}_c$. Since there is only one class in pedestrian detection, the classification loss is omitted. We use the losses proposed in the original paper of each model. On top of that, our occlusion handling introduces two additional losses: *part loss* and *score loss*. The part loss $\mathcal{L}_p$ is the ℓ_2 loss of the part confidence map for max/soft part scores:

$$\begin{aligned}\mathcal{L}_{p,(ijk)} = &\left(\lambda_p^+ \mathbb{I}_{(ijk)}^+ + \lambda_p^- \mathbb{I}_{(ijk)}^-\right) \\ &\times \sum_{m=1}^{M}\sum_{n=1}^{N}\left(\mathbf{V}_{(ijk)}(m,n) - \widehat{\mathbf{V}}_{(ijk)}(m,n)\right)^2\end{aligned} \tag{3.11}$$

where $\mathbb{I}^{+}_{(ijk)} = 1$ indicates that the (ijk)-th anchor is a positive example while $\mathbb{I}^{-}_{(ijk)} = 1$ indicates a negative example. The score loss $\mathcal{L}_s$ is identically defined as

$$\mathcal{L}_{s,(ijk)} = \left(\lambda_s^{+f}\mathbb{I}^{+f}_{(ijk)} + \lambda_s^{+o}\mathbb{I}^{+o}_{(ijk)}\right)\left(1 - \hat{s}_{p,(ijk)}\right)^2 + \lambda_s^{-}\mathbb{I}^{-}_{(ijk)}\hat{s}^2_{p,(ijk)} \tag{3.12}$$

where $\mathbb{I}^{+o}_{(ijk)} = 1$ indicates that the (ijk)-th anchor is positive but occluded, while $\mathbb{I}^{+f}_{(ijk)} = 1$ indicates a fully visible example. We divide the positive cases in these two ways in order to assign larger weights to occluded examples. Finally, the total loss is a weighted sum of all four losses:

$$\mathcal{L} = \sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{k=1}^{K}\left(\lambda_l\mathcal{L}_{l,(ijk)} + \lambda_c\mathcal{L}_{c,(ijk)} + \lambda_p\mathcal{L}_{p,(ijk)} + \lambda_s\mathcal{L}_{s,(ijk)}\right). \tag{3.13}$$

3.2.3 Refinement for Hard Negative Handling

Our key idea for reducing false-positives by hard negative examples is to introduce the *average grid classifiers*, which are not only universally applicable to any single-stage model, but also end-to-end trainable with little time overhead. Figure 3.5 presents the overview of our hard negative handling method. Given an image, each single-stage method internally generates a set of feature maps of various resolutions. We apply the convolutional classifiers to the intermediate feature maps to obtain a set of *grid confidence maps*, whose sizes are summarized in Table 3.2. We then resize all confidence maps to the resolution of the input image, and average them to obtain a single grid map of pixel-wise confidence. Finally, models adjust the confidence of each bounding box, using the pixel values of the grid map.

Grid confidence map. The *grid confidence map* of layer $l \in [1, L]$ is a $w_l \times h_l$ grid map denoted by $\mathbf{G}_l$ whose values are ranged in $[0, 1]$ (see Table 3.2). The groundtruth for $\mathbf{G}_{l,gt}$ is generated as follows. First, for layer l, the input image is divided as a $w_l \times h_l$ grid. At each cell (i, j), we calculate the area ratio of how much this cell is

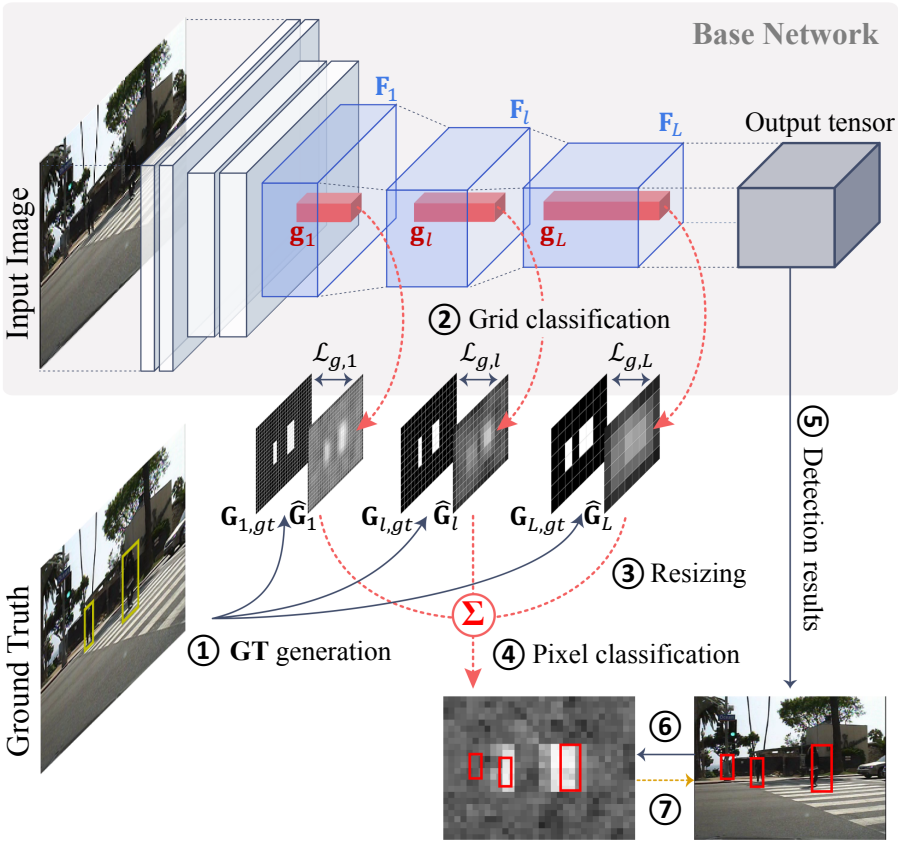


Figure 3.5: The overview of hard negative handling method.

Models	Shapes of grid confidence maps $\{w_l, h_l\}$
SqueezeDet+ [86]	$\{78, 58\}$, $\{38, 28\}$
YOLOv2 [68]	$\{80, 60\}$, $\{40, 30\}$, $\{20, 15\}$
SSD [56]	$\{40, 30\}$, $\{20, 15\}$
DSSD [25]	$\{40, 30\}$, $\{20, 15\}$, $\{40, 30\}$

Table 3.2: The dimensions of grid confidence maps for a 640×480 input image (w_l: width, h_l: height).

occupied by a groundtruth bounding box, which becomes the value of $\mathbf{G}_{l,gt}(i,j)$. We use $\mathbf{G}_{l,gt}(i,j)$ to learn the following grid classifiers that predict grid confidence maps.

In the forward pass, we can compute a feature map $\mathbf{F}_l \in \mathbb{R}^{w_l \times h_l \times c_l}$ at each layer $l \in [1, L]$. The grid classifier is implemented as an 1×1 convolutional filter $\mathbf{g}_l \in \mathbb{R}^{1 \times 1 \times c_l \times 1}$. Then the predicted grid confidence map $\widehat{\mathbf{G}}_l \in \mathbb{R}^{w_l \times h_l \times 1}$ is obtained by

convolution between the feature map $\mathbf{F}_l$ and the filter $\mathbf{g}_l$:

$$\widehat{\mathbf{G}}_l = \mathbf{F}_l * \mathbf{g}_l, \quad l \in [1, L]. \tag{3.14}$$

Once we compute a set of $\{\widehat{\mathbf{G}}_l\}_{l=1}^{L}$ for all L layers, we resize them to be the same with the input image using a bilinear interpolation: $\{\widehat{\mathbf{G}}'_l\}_{l=1}^{L}$. Finally we obtain a single averaged confidence map: $\widehat{\mathbf{G}} = \frac{1}{L}\sum_{l=1}^{L}\widehat{\mathbf{G}}'_l$, where $\widehat{\mathbf{G}} \in \mathbb{R}^{W \times H}$. Given an input image, suppose that its initial predicted bounding boxes are $\mathbf{bb}_k, k \in [1, B]$ where $\mathbf{bb}_k = \{x_k, y_k, w_k, h_k, c_k\}$. For each bounding box $\mathbf{bb}_k, k \in [1, B]$, we compute the averaged confidence score s_k:

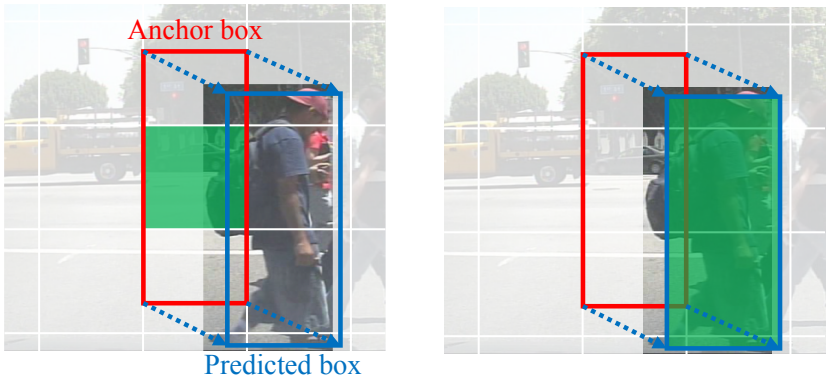
$$s_k = \frac{1}{w_k h_k} \sum_{i=x_k}^{x_k+w_k-1} \sum_{j=y_k}^{y_k+h_k-1} \widehat{\mathbf{G}}(i, j). \tag{3.15}$$

Finally, the adjusted confidence for each $\mathbf{bb}_k$ is computed as the geometric mean of s_k and its initial confidence c_k.

$$c'_k = \sqrt{s_k c_k}, \quad k \in [1, B]. \tag{3.16}$$

The intuition of why this method works is two-fold. First, except SSD [56] and DSSD [25] that use multiple feature maps, other single-stage models generate output tensors only from one (*e.g.* SqueezeDet+ [86]) or two feature maps (*e.g.* YOLOv2 [68]). However, relying on only one or two feature maps may be risky and error-prone especially to hard negative examples. Thus, our idea is to make a final detection decision based on the average of multi-resolution feature maps. Concatenating feature maps of several layers [4] or using skip connections [25, 73] can be alternatives, but our method is simpler and more intuitive.

Second, our grid classifiers complement one drawback of single-stage models: the mismatch of a predicted box and its feature representation. For better understanding, we present an example in Figure 3.6, in which an anchor is shown in red, a prediction box is in blue, and the feature region is shaded in green. The two-stage models using ROI pooling (*e.g.* [27, 26, 69]) use the features on the actual region of a predicted bounding box (Figure 3.6b), whereas the single-stage models use the features where



(a) In single-stage models (b) In region-based models

Figure 3.6: An intuition of why the single-stage models suffer from the mismatch of a predicted box with its feature representation. The anchor is shown in red, the predicted box is in blue, and the feature region is shaded in green.

the default anchor is located. (Figure 3.6a). Our grid classifiers alleviate this issue by allowing the model to use the features of the exact predicted region, which makes the detection output more reliable.

Training Objective

For the grid classifier, we add the grid loss $\mathcal{L}_g$ to the localization and confidence losses in section 3.2.2. The grid loss of each layer is defined as

$$\mathcal{L}_{g,l} = \sum_{i=1}^{w_l} \sum_{j=1}^{h_l} \left(\lambda_g^+ \mathbb{I}_{l,(ij)}^+ + \lambda_g^- \mathbb{I}_{l,(ij)}^- \right) \times \left(\mathbf{G}_l(i,j) - \widehat{\mathbf{G}}_l(i,j) \right)^2$$

where $\mathbb{I}_{l,(ij)}^+ = 1$ if $\mathbf{G}_l(i,j) > 0$ and $\mathbb{I}_{l,(ij)}^- = 1$ if $\mathbf{G}_l(i,j) = 0$. Finally, the total loss is

$$\mathcal{L} = \sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{k=1}^{K} \left(\lambda_l \mathcal{L}_{l,(ijk)} + \lambda_c \mathcal{L}_{c,(ijk)} \right) + \sum_{l=1}^{L} \lambda_{g,l} \mathcal{L}_{g,l}.$$

3.3 Experiment Settings

We focus on validating that the proposed approach for occlusion and hard negative handling indeed help improve accuracies of pedestrian detection. We use two

benchmark datasets: Caltech pedestrian [17] and CityPersons [93]. We apply our approach into four single-stage models: SqueezeDet+ [86], YOLOv2 [68], SSD [56], and DSSD [25]. For fair comparison, we implement all methods using TensorFlow [1]. For SqueezeDet+, we directly use the source code provided by the authors, while for all the other methods, we re-write the codes in TensorFlow.

3.3.1 Datasets

The Caltech dataset consists of about 250,000 frames taken from urban scenes. It is divided into 11 sets: *set00–set05* as training data, and *set06–set10* as test data. The label consists of three classes (*person*, *people*, and *person?*), and we only use *person* for training. We strictly following the experiment protocols of the dataset.

The CityPersons dataset contains 5,000 images in total and approximately 3,000 images are for training. Since the CityPersons dataset is derived from a subset of Cityscapes dataset [13] that has pixel-level instance labels, the visible area annotations can be generated automatically. It also includes full-body annotations at a fixed ratio 0.41 for four classes (*pedestrian*, *rider*, *sitting person* and *other person*), and we use the *pedestrian* class only.

In both datasets, a bounding box is assigned to the whole area of a person, which is the prediction target of our task. The visible area is additionally annotated for an occluded person, which allows us to make the ground truth of part confidence maps in section 3.2.2. We limit the ground truth that are labeled as *person* and its occluded fraction is less than 0.8, which amounts to 125,623 and 15,371 instances for Caltech and CityPersons dataset respectively. Table 3.3–3.4 show how *part confidence maps* are distributed according to its grid size ($M \times N$). For Caltech pedestrian dataset, more than 80% of examples are fully visible, and the examples where below parts are occluded are the most common cases among partially visible examples. In contrast, we can see that more occluded cases and patterns are found in CityPersons dataset.

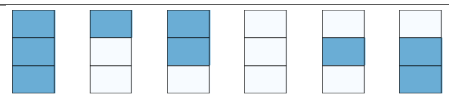
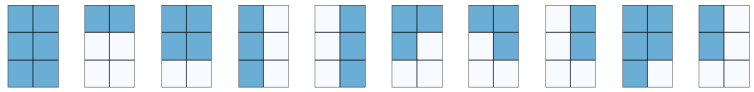
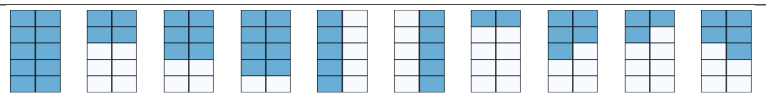
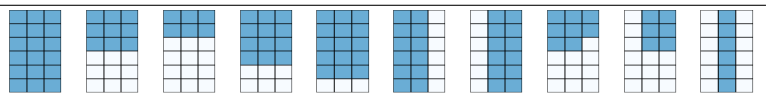
$M \times N$	Rank	1	2	3	4	5	6	7	8	9	10
1×3	Part ($n = 6$)										
	Freq.	105615	10537	8139	1007	170	155				
	Percent.	84.07	8.39	6.48	0.80	0.14	0.12				
	Cum.	84.07	92.46	98.94	99.74	99.88	100.0				
2×3	Part ($n = 28$)										
	Freq.	103935	9248	6572	938	891	692	633	467	366	351
	Percent.	82.47	7.36	5.23	0.75	0.71	0.55	0.50	0.37	0.29	0.28
	Cum.	82.47	90.10	95.33	96.08	96.78	97.34	97.84	98.21	98.50	98.78
2×5	Part ($n = 55$)										
	Freq.	102925	8367	4384	2805	789	766	597	378	372	367
	Percent.	81.93	6.66	3.49	2.23	0.63	0.61	0.48	0.30	0.30	0.29
	Cum.	81.93	88.59	92.08	94.31	94.94	95.55	96.03	96.33	96.62	96.92
3×6	Part ($n = 171$)										
	Freq.	101162	4128	3746	2369	1348	1256	1111	823	707	652
	Percent.	80.53	3.29	2.98	1.89	1.07	1.00	0.88	0.66	0.56	0.52
	Cum.	80.53	83.81	86.80	88.68	89.76	90.75	91.64	92.29	92.86	93.38

Table 3.3: Distribution of *part confidence maps* on Caltech *train* dataset. n is the total number of parts. The $blue$ areas represent visible parts.

Performance evaluation. The models are evaluated using the log-average miss rate, the official metric of both Caltech and CityPersons dataset. This is the average value of miss rates for 9 FPPI (false positives per image) rates evenly spaced in the log-space ranging from 10^{-2} to 10^{0}. Depending on occlusion levels and scales, there are different evaluation settings. The occlusion level is divided into *none*, *partial*, and *heavy*, meaning 0, $(0, 35]$, $(35, 80]$ percent fractions of occlusion, respectively. The scale is divided into *none*, *medium*, and *far*, corresponding to $[20, 30)$, $[30, 80)$, $[80, 480)$, respectively, based on the height in pixels.

3.3.2 Configuration Details

Training process. For each ground truth bounding box, we associate the following anchors as positive examples: (i) the anchor box with the highest IoU value, and (ii) the anchor boxes whose IoU values are over 0.5. We select anchor boxes whose IoU

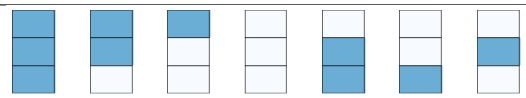
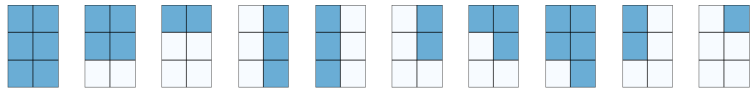
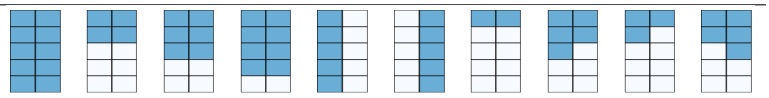
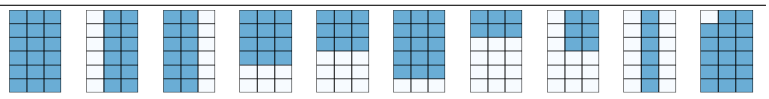
$M \times N$	Rank	1	2	3	4	5	6	7	8	9	10
1×3	Part ($n = 7$)										
	Freq.	11981	1823	1172	197	144	32	22			
	Percent.	77.95	11.86	7.62	1.28	0.94	0.21	0.14			
	Cum.	77.95	89.81	97.43	98.71	99.65	99.86	100.0			
2×3	Part ($n = 30$)										
	Freq.	10933	1382	895	748	366	204	175	95	93	90
	Percent.	71.13	8.99	5.82	4.87	2.38	1.33	1.14	0.62	0.61	0.59
	Cum.	71.13	80.12	85.94	90.81	93.19	94.52	95.65	96.27	96.88	97.46
2×5	Part ($n = 57$)										
	Freq.	10607	928	749	690	601	332	144	138	109	80
	Percent.	69.01	6.04	4.87	4.49	3.91	2.16	0.94	0.90	0.71	0.52
	Cum.	69.01	75.04	79.92	84.41	88.32	90.48	91.41	92.31	93.02	93.54
3×6	Part ($n = 163$)										
	Freq.	8865	1624	631	537	489	336	311	243	205	183
	Percent.	57.67	10.57	4.11	3.49	3.18	2.19	2.02	1.58	1.33	1.19
	Cum.	57.67	68.24	72.34	75.84	79.02	81.20	83.23	84.81	86.14	87.33

Table 3.4: Distribution of *part confidence maps* on CityPersons *train* dataset. n is the total number of parts. The $blue$ areas represent visible parts.

values are less than 0.4 as negative examples. We ignore anchors whose IoU Values are between 0.4 and 0.5 for calculating the loss. Because all base models define many anchor boxes as default, there are overwhelming many negative examples compared to positive examples. Therefore, instead of including them all negative examples in the loss calculation, we select only the negative examples that cause the highest loss for each loss type (confidence, part, score, and grid loss). We use the ratio of negative to positive examples as a hyperparameter. (Only for score loss L_s, we use the ratio to occluded examples.) We set 3 for SSD and 10 for the other models. To optimize the loss function, we use the Adam optimizer with ℓ_2 regularization. To improve the performance and ability of the model to generalize, we augment the dataset 5 times with shifting and flipping, and add noise to training images by changing brightness, saturation, and contrast at random.

Inference process. At test time, for a given image, we first compute the output

tensor via forward pass, and then obtain the final prediction results by applying the grid classifiers and non-maximum suppression (NMS). For fast inference, we apply these two steps to only top 256 predicted boxes with the highest confidences. We set the IoU threshold of NMS to 0.6 for Caltech and 0.5 for CityPersons dataset.

3.4 Experiment Results

We first evaluate our occlusion handling, hard negative handling, and joint learning of the two methods (section 3.4.1). Next, we provide ablation studies on individual methods (section 3.4.2), and analyze the size/time overhead of our approach (section 3.4.3). Lastly, we illustrate the qualitative results that visualize effects of occlusion and hard negative handling (section 3.4.4).

3.4.1 Quantitative Results

Evaluation of Occlusion Handling

The most widely used setting in Caltech dataset is called as *reasonable* setting, which only includes pedestrians whose sizes are greater than 50 pixels and occlusion levels are *none* or *partial*. However, one of our evaluation goals is occlusion robustness, thereby we test *all* setting, which includes all occlusion levels (*none*, *partial*, and *heavy*). We tune each model so that it performs the best for the *all* setting. That is, the model is trained to work well with the largest subset, so occasionally our performance for other small subsets can be not as good as the base networks.

Table 3.5 shows the breakdown performance of our occlusion handling method on Caltech dataset. Our methods of max/soft part scores lead significant performance improvement over all four base models. Overall, the error rates can be sorted in the following order: soft $<$ max $<$ base. The max part score is worse than the soft part score, but sometimes it is the best in the *heavy* setting. That is, the max part score is good at detecting severely occluded persons, because it attains a high detection score even if only a single cell of the part confidence map is high-valued.

Because the soft part score shows the best performance for the *all* setting, we use

Model	Reasonable	All	None	Partial	Heavy
SqueezeDet+ [86]	23.37	32.83	21.58	36.07	63.65
+ Max part	22.08	30.30	19.46	40.14	**56.60**
+ Soft part	**20.78**	**30.18**	**18.76**	**34.65**	59.87
YOLOv2 [68]	20.83	29.35	18.97	34.37	57.55
+ Max part	19.31	27.56	17.40	**31.69**	**53.90**
+ Soft part	**18.29**	**27.16**	**16.12**	31.94	57.02
SSD [56]	16.36	25.18	14.55	27.89	53.80
+ Max part	15.60	23.70	13.69	27.85	**50.02**
+ Soft part	**14.23**	**22.53**	**12.22**	**27.52**	50.46
DSSD [25]	13.25	20.53	11.23	**25.23**	44.13
+ Max part	12.72	20.23	10.72	25.80	44.81
+ Soft part	**10.97**	**18.58**	**8.88**	26.14	**44.11**

Table 3.5: Detailed breakdown performance of our occlusion handling methods on Caltech *test* dataset (Height $\geq$ 50). We report the log-average miss rate (lower is better).

it as our occlusion handling method for the rest of this section.

Table 3.7 shows additional results for SSD and DSSD models on the test subset of height ≥ 20. We choose SSD and DSSD as base models, because they are particularly robust to small objects among four base models, thanks to its adoption of multi-scale feature maps. We train and test SSD/DSSD-based models, including images with very small pedestrians (height ≥ 20), and observe that our occlusion handling consistently improve SSD and DSSD to detect very small and highly occluded pedestrians.

We also tested our occlusion handling methods on the CityPersons dataset and the results are summarized in Table 3.8. In every configuration in each model, our occlusion handling methods improve performance by a wide margin.

Evaluation of Hard Negative Handling

Table 3.6–3.8 show the detailed breakdown performance of our hard negative handling on the two datasets. The performance is always better than baseline when the grid classifiers are used only for training. However, if we use the adjusted confidence, SqueezeDet+ and YOLOv2 perform the best, but SSD and DSSD become worse than the baseline. As discussed in section 3.2.3, there are two cases where the grid classifiers are helpful: i) the refinement by the averaged results from multiple feature maps, and

Model	Reasonable	All	None	Partial	Heavy
SqueezeDet+ [86]	23.37	32.83	21.58	36.07	63.65
+ Part score	20.78	30.18	18.76	34.65	59.87
+ Grid classifiers	19.58	28.72	17.79	**29.68**	**56.53**
+ Joint learning	**18.99**	**28.29**	**16.83**	30.82	57.77
YOLOv2 [68]	20.83	29.35	18.97	34.37	57.55
+ Part score	18.29	27.16	16.12	31.94	57.02
+ Grid classifiers	**16.92**	27.65	**14.95**	27.44	63.57
+ Joint learning	17.56	**26.61**	16.59	**25.68**	**53.77**
SSD [56]	16.36	25.18	14.55	27.89	53.80
+ Part score	14.23	**22.53**	12.22	27.52	**50.46**
+ Grid classifiers*	**14.04**	23.79	**12.03**	**26.52**	55.10
+ Joint learning*	15.03	23.54	13.06	29.57	51.53
DSSD [25]	13.25	20.53	11.23	25.23	44.13
+ Part score	10.97	18.58	**8.88**	26.14	44.11
+ Grid classifiers*	**10.85**	**18.20**	9.00	24.28	**42.42**
+ Joint learning*	11.42	19.38	10.00	**21.11**	45.80

Table 3.6: Overall performance on Caltech *test* dataset (Height ≥ 50). * denotes that grid classifiers are used only for training.

Model	All	None	Partial	Heavy
SSD [56]	60.19	52.21	**67.96**	76.47
+ Part score	58.94	51.71	68.85	**74.37**
+ Grid classifiers*	59.66	51.60	68.93	76.04
+ Joint learning*	**58.88**	**51.52**	70.71	74.81
DSSD [25]	53.03	44.72	64.15	69.59
+ Part score	50.55	41.51	61.68	69.65
+ Grid classifiers*	**49.24**	**41.32**	**60.74**	**65.99**
+ Joint learning*	52.00	43.88	61.57	69.50

Table 3.7: Overall performance on Caltech *test* dataset (Height ≥ 20).

ii) mitigation of the mismatch between a predicted box and its feature representation. SSD and DSSD already uses rich information from several layers of both low- and high-resolution feature maps (*e.g.* five and eight layers respectively). And they have layers that care for the object scales; thus the feature representations of the groundtruth and its anchor are not significantly mismatched because of their similar scales.

Evaluation of Joint Learning

Table 3.6–3.8 also show the performance of joint application of the two methods for occlusion and hard negative handling. In this case, the adjusted confidence is com-

Model	Reasonable	All	None	Partial	Heavy
SqueezeDet+ [86]	28.42	43.90	20.48	28.64	62.61
+ Part score	26.33	41.90	19.38	**25.57**	60.01
+ Grid classifiers	26.69	41.92	19.26	26.32	61.56
+ Joint learning	**26.29**	**40.88**	**18.22**	26.22	**58.57**
YOLOv2 [68]	23.36	38.01	14.23	22.65	52.50
+ Part score	20.45	36.36	12.36	20.08	51.99
+ Grid classifiers	21.41	36.76	13.18	20.13	50.30
+ Joint learning	**19.19**	**34.09**	**10.77**	**18.69**	**50.18**
SSD [56]	22.54	35.61	16.91	21.95	50.66
+ Part score	19.01	33.95	13.18	**18.16**	51.48
+ Grid classifiers*	19.71	34.32	13.28	19.11	49.02
+ Joint learning*	**18.99**	**33.52**	**12.70**	19.33	**48.42**
DSSD [25]	19.70	34.37	15.75	18.90	51.88
+ Part score	18.25	33.16	13.79	17.65	49.47
+ Grid classifiers*	18.45	**31.67**	12.82	17.96	**46.60**
+ Joint learning*	**16.77**	31.71	**11.15**	**16.05**	48.52

Table 3.8: Overall performance on CityPersons *val* dataset (Height ≥ 50).

puted as the geometric mean of a part score (Eq.(3.7) or Eq.(3.9)), an averaged confidence score (Eq.(3.15)), and an initial confidence. For SSD and DSSD, we use the grid confidence map only for training, because this setting leads the best performance as discussed in section 3.4.1. As expected, the joint learning improve the performance of the models that are adjusted well by grid classifiers (*e.g*. SqueezeDet+ and YOLOv2). Especially, they achieve the best performances for standard subset of Caltech dataset (*i.e*. *all* for height ≥ 50).

Comparison with State-of-the-art Models

The goal of this work is to propose a lightweight approach that is applicable to single-stage detection models for improving their occlusion and hard negative handling. We do not argue that our approach is integrable with any detection models, but limited to single-stage models, which are often inferior to the two-stage models in performance, but are much faster and lighter. Therefore, we focus on improving the performance of base networks, instead of comparing with state-of-the-art methods. Our final detection accuracies depend on those of base models; thus if the base model is competitive, our method is as well.

Table 3.9 shows the performance comparison with state-of-the-art models on Caltech *test* dataset. Encouragingly, in some settings (*all*/*heavy* with Height $\geq$ 50 and $\geq$ 20), our approach with DSSD achieves the best as in Table 3.9.

Model	Height $\geq$ 50					Height $\geq$ 20			
	Reasonable	All	None	Partial	Heavy	All	None	Partial	Heavy
DeepParts [81]	11.89	22.79	10.64	19.93	60.42	64.78	58.43	70.39	81.81
MS-CNN [5]	9.95	21.53	8.15	19.24	59.94	60.95	53.67	67.16	79.51
RPN+BF [91]	9.58	24.01	7.68	24.23	69.91	64.66	56.38	72.55	87.48
F-DNN [18]	8.65	19.31	7.10	15.41	55.13	50.55	40.29	60.60	76.98
F-DNN+SS [18]	**8.18**	18.82	**6.74**	**15.11**	53.67	50.29	**40.21**	**60.08**	75.77
DSSD [25] + Ours	10.85	**18.20**	9.00	24.28	**42.42**	**49.24**	41.32	60.74	**65.99**

Table 3.9: Comparison with state-of-the-art models on Caltech *test* dataset (lower is better).

3.4.2 Ablation Experiments

We further provide the results of the experiment on occlusion and hard negative handling methods by differing settings for each method.

Occlusion Handling Methods

In this section, we present more experiment results about our occlusion handling methods, in which we measure the performance changes by varying the configuration of key components.

For the *max part score* method, we test four different settings of the grid size ($M \times N$) of the *part confidence map* to find the most proper size. Next, for the *soft part score* method, we measure the performance variation according to the grid size ($M \times N$) of the *part confidence map*, and the depth of additional layers for calculating the part score. The reason for considering the depth is to check that single layer is sufficient to interpret the *part confidence map*. We test four different combinations between 2×5 and 3×6 for the grid size, and single and double layers. Beyond the single layer setting we proposed as in Eq.(3.17), we also tested two fully connected layers to obtain the final detection score (Eq.(3.18)),

$$s_{\text{person}} = \sigma\left(\mathbf{W}_{s1,2}^{\top} \max\left(\mathbf{0}, \mathbf{W}_{s1,1}^{\top} \widehat{\mathbf{V}}\right)\right) \tag{3.17}$$

$$s_{\text{person}} = \sigma\left(\mathbf{W}_{s2,3}^{\top} \max\left(\mathbf{W}_{s2,2}^{\top} \max\left(\mathbf{0}, \mathbf{W}_{s2,1}^{\top} \widehat{\mathbf{V}}\right)\right)\right) \tag{3.18}$$

where σ is a sigmoid function, and parameters to learn include $\mathbf{W}_{s1,1}, \mathbf{W}_{s2,1} \in \mathbb{R}^{(M\times N)\times S}$, and $\mathbf{W}_{s1,2} \in \mathbb{R}^{S\times 1}$, $\mathbf{W}_{s2,2} \in \mathbb{R}^{S\times S'}$, $\mathbf{W}_{s2,3} \in \mathbb{R}^{S'\times 1}$. In our experiments, number of nodes per layer is fixed to $S = 6$ for single layer, and $(S, S') = (45, 64)$ for double layer setting.

Table 3.10–3.13 show the results of occlusion handling methods for each model. For all models, soft part score method shows the best performance in general. This generally means that the information in the semantic part is more meaningful than the information in the basic part (each grid of part confidence map). However, the max part score method shows better results on *heavy* subset. Since heavily occluded person is visible only for small area, its confidence is highly correlated to the score of basic part.

Method		Height ≥ 50				
Part score	Structure	Reasonable	All	None	Partial	Heavy
Baseline		23.37	32.83	21.58	36.07	63.65
Max	(1×3)	23.47	33.53	21.10	38.43	65.87
	(2×3)	22.07	32.40	19.51	38.37	65.83
	(2×5)	22.08	30.30	19.46	40.14	**56.60**
	(3×6)	22.92	32.15	21.01	33.65	62.35
Soft	(2×5) - 6	**20.30**	31.80	18.45	**33.28**	68.25
	(3×6) - 6	22.56	31.68	20.52	35.88	60.71
	(2×5) - 45 - 64	22.20	31.82	20.04	37.27	62.69
	(3×6) - 45 - 64	20.78	**30.18**	**18.76**	34.65	59.87

Table 3.10: Detailed breakdown performance of occlusion handling methods at *SqueezeDet+* on Caltech *test* dataset (lower is better).

Method		Height ≥ 50				
Part score	Structure	Reasonable	All	None	Partial	Heavy
Baseline		20.83	29.35	18.97	34.37	57.55
Max	(1×3)	18.33	28.02	16.58	**30.60**	60.93
	(2×3)	20.74	29.03	18.75	33.82	58.43
	(2×5)	19.31	27.56	17.40	31.69	**53.90**
	(3×6)	20.58	31.11	18.52	33.71	67.05
Soft	(2×5) - 6	18.91	28.17	16.90	31.26	60.07
	(3×6) - 6	**18.29**	**27.16**	**16.12**	31.94	57.02
	(2×5) - 45 - 64	18.77	28.51	16.83	30.61	61.78
	(3×6) - 45 - 64	18.98	27.93	16.51	33.85	57.42

Table 3.11: Detailed breakdown performance of occlusion handling methods at *YOLOv2* on Caltech *test* dataset (lower is better).

Method		Height ≥ 50				
Part score	Structure	Reasonable	All	None	Partial	Heavy
Baseline		16.36	25.18	14.55	27.89	53.80
Max	(1×3)	15.81	24.09	13.86	29.40	51.18
	(2×3)	16.19	23.87	14.86	30.03	**47.68**
	(2×5)	15.60	23.70	13.69	27.85	50.02
	(3×6)	16.16	24.84	14.27	**25.30**	53.19
Soft	(2×5) - 6	15.56	24.37	13.15	30.99	54.41
	(3×6) - 6	14.57	23.83	12.57	27.80	53.94
	(2×5) - 45 - 64	15.50	23.76	13.56	28.34	51.34
	(3×6) - 45 - 64	**14.23**	**22.53**	**12.22**	27.52	50.46

Table 3.12: Detailed breakdown performance of occlusion handling methods at *SSD* on Caltech *test* dataset (lower is better).

Method		Height ≥ 50				
Part score	Structure	Reasonable	All	None	Partial	Heavy
Baseline		13.25	20.53	11.23	25.23	44.13
Max	(1×3)	12.72	20.23	10.72	25.80	44.81
	(2×3)	13.04	22.44	10.82	29.42	53.66
	(2×5)	12.01	20.92	10.36	22.40	49.82
	(3×6)	13.01	20.52	11.36	23.83	44.44
Soft	(2×5) - 6	11.60	19.87	9.78	**22.12**	46.75
	(3×6) - 6	11.84	20.28	10.12	24.66	47.49
	(2×5) - 45 - 64	**10.97**	**18.58**	**8.88**	26.14	**44.11**
	(3×6) - 45 - 64	11.99	20.63	10.06	25.49	49.33

Table 3.13: Detailed breakdown performance of occlusion handling methods at *DSSD* on Caltech *test* dataset (lower is better).

Hard Negative Handling Methods

We perform ablation studies on grid classifiers by changing two configurations in the model. First, we change the size of convolutional filter that is used as a classifier (1×1 and 3×3). Second, we compare the performances of different uses of the grid confidence map as follows.

- *Baseline*: The results of base models.
- *Loss only*: The result of using the grid confidence map for training.
- *Adjustment*: The result of using the grid confidence map for training and refining the initial confidence.

The parameters of the models used by *loss only* and *adjustment* are the same. The only difference is whether to adjust the initial confidence using the predicted grid confidence map.

Table 3.14 shows the overall results of ablation experiments of hard negative handling methods. The performance of the *loss only* is always better than the *baseline*. However, in case of *adjustment*, the results are different depending on the base model. The *adjustment* performs the best in SqueezeDet+ and YOLOv2, but the worst in the SSD and DSSD (even worse than the baseline). In section 3.2.3, we mentioned about the two intuitions of why the grid classifiers help improve the performance: i) the refinement by the averaged results from multiple feature maps, and ii) resolving the mismatch between a predicted box and its feature representation in the base models. The SSD and DSSD have layers that care for the object scales; that is, the grid feature representations of the ground truth and its anchor are not significantly mismatched each other because of their similar scales. Therefore, the second effect is not much significant in SSD and DSSD.

Model	Height ≥ 50				
	Reasonable	All	None	Partial	Heavy
SqueezeDet+ [86]	23.47	32.88	21.69	34.05	62.96
+ 1×1 (*loss only*)	20.87	29.23	18.88	34.20	56.56
+ 1×1 (*adjustment*)	**19.58**	**28.72**	**17.79**	**29.68**	**56.53**
+ 3×3 (*loss only*)	21.36	31.56	19.83	30.44	65.51
+ 3×3 (*adjustment*)	20.61	30.48	18.96	30.21	62.42
YOLOv2 [68]	20.83	29.35	18.97	34.37	**57.55**
+ 1×1 (*loss only*)	18.66	28.79	16.74	31.12	62.27
+ 1×1 (*adjustment*)	**16.92**	**27.65**	**14.95**	**27.44**	63.57
+ 3×3 (*loss only*)	19.88	28.54	18.24	31.94	57.76
+ 3×3 (*adjustment*)	18.80	27.86	17.06	29.46	57.61
SSD [56]	16.36	25.18	14.55	27.89	53.80
+ 1×1 (*loss only*)	14.92	**23.68**	12.90	27.90	**52.93**
+ 1×1 (*adjustment*)	16.94	26.20	14.90	28.20	54.51
+ 3×3 (*loss only*)	**14.04**	23.79	**12.03**	**26.52**	55.10
+ 3×3 (*adjustment*)	16.51	27.17	14.39	26.93	57.51
DSSD [25]	13.25	20.53	11.23	25.23	44.13
+ 1×1 (*loss only*)	11.83	19.95	9.90	26.41	47.51
+ 1×1 (*adjustment*)	15.28	23.74	13.19	26.80	48.73
+ 3×3 (*loss only*)	**10.85**	**18.20**	**9.00**	24.28	42.42
+ 3×3 (*adjustment*)	14.40	21.69	12.83	**22.50**	**42.34**

Table 3.14: Detailed breakdown performance of hard negative handling methods on Caltech *test* dataset (lower is better).

3.4.3 Memory and Computation Time Analysis

We report model sizes and computation times of our methods in Table 3.15–3.16, which clearly show that the additional size and time overheads by our methods are very small. We test on a workstation with Intel Xeon Processor E5-2695 V4 CPU and NVIDIA Titan X Pascal GPU.

Model	Baseline	Additional methods		Total
		+ Part score	+ Grid cls.	
SqueezeDet+ [86]	27.59	1.99	0.04	29.62
YOLOv2 [68]	268.35	0.45	0.06	268.86
SSD [56]	93.06	4.65	0.06	97.77
DSSD [25]	345.07	2.07	0.09	347.23

Table 3.15: Comparison of model sizes (in MB).

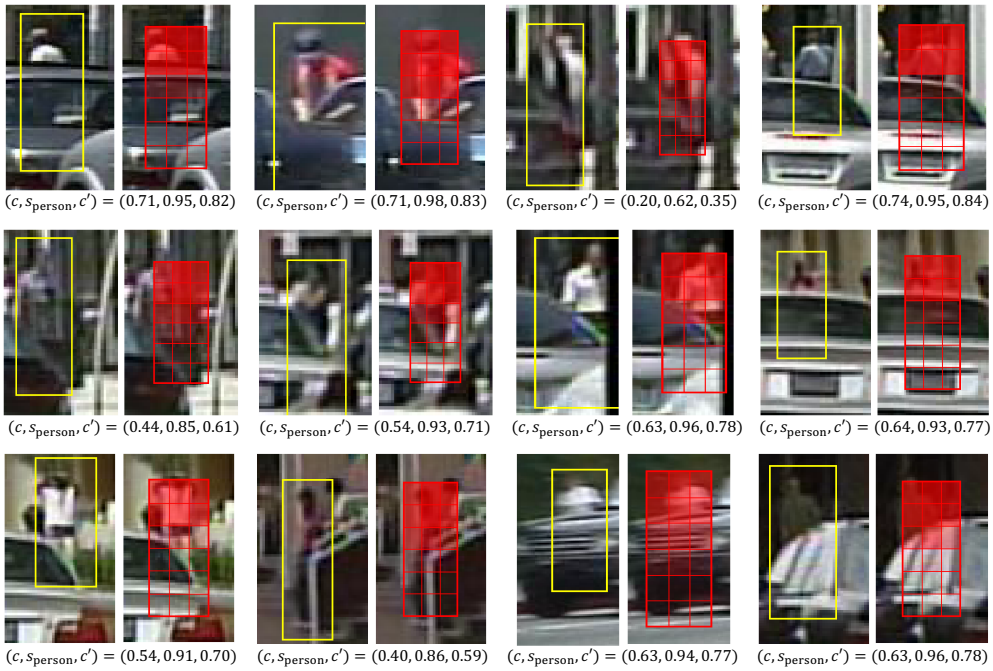
Model	Baseline	Additional methods		Total
		+ Part score	+ Grid cls.	
SqueezeDet+ [86]	23.02	0.89	0.54	24.45
YOLOv2 [68]	32.19	0.70	1.12	34.01
SSD [56]	32.50	1.08	1.18	34.76
DSSD [25]	84.36	0.97	1.55	86.88

Table 3.16: Comparison of inference time (in milliseconds).

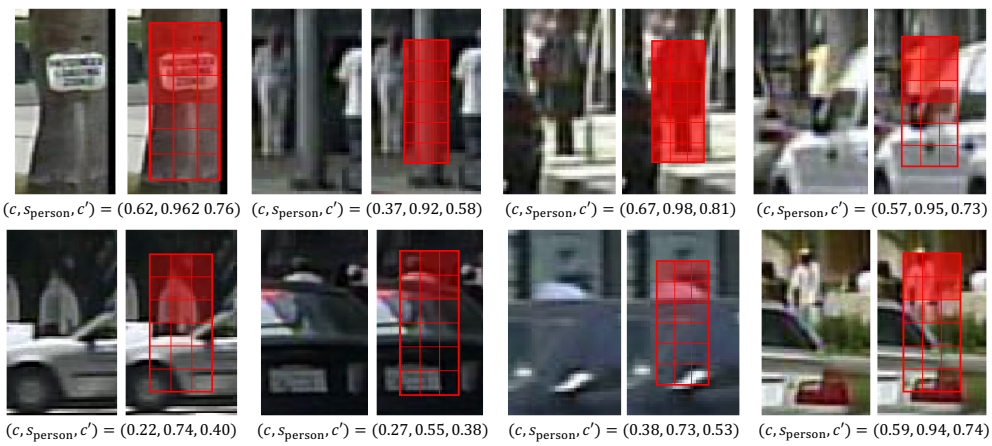
3.4.4 Qualitative Results

Figure 3.7 shows examples of success and failure cases of our occlusion handling. In the success cases of Figure 3.7a, the initial confidences for the person are relatively low, but they are correctly adjusted thanks to the high part scores. Many success examples are the cases whose visible areas are upper parts. The models can easily detect those examples, mainly because much occlusion in the training set is such cases. as discussed in section 3.3.1. The failure cases (Figure 3.7b) include hard negative and mislabelled examples.

Figure 3.8 shows the examples of applying the grid classifiers to the SqueezeDet+. As in Table 3.2, SqueezeDet+ uses two feature maps (*i.e.* $L = 2$), from which we generate the grid confidence map. In Figure 3.8b, our grid classifiers effectively suppress the confidence scores of false positives. We also find that there are some cases where our method increases the confidence scores of hard positives as in 3.8a. We can also check the robustness of using multiple layers. If one of the layers predicts its confidence map incorrectly as the initial confidence, the other layer can refine its value.

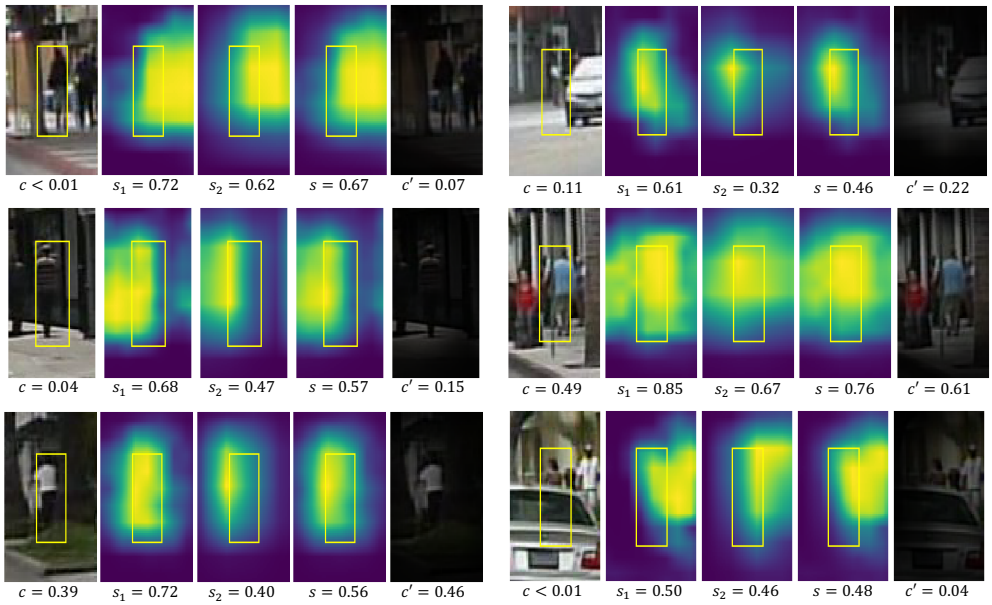


(a) Success cases

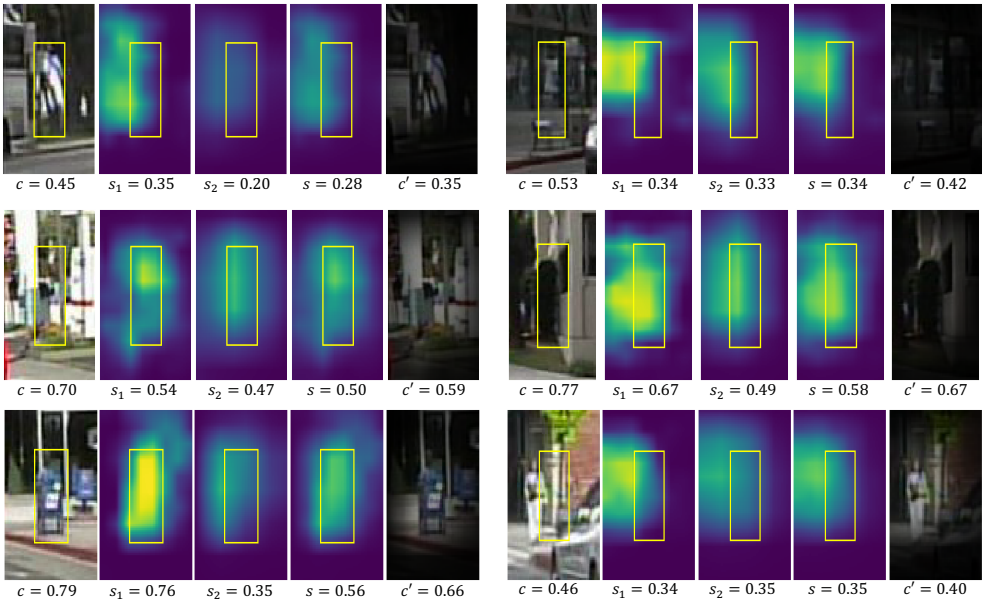


(b) Failure cases

Figure 3.7: Examples of occlusion handling. For better visualization, we crop detection regions from images.



(a) Positive examples



(b) Negative examples

Figure 3.8: Examples of adjustment by grid classifiers. For better visualization, we crop detection regions from images.

3.5 Conclusion

We addressed the two critical issues of pedestrian detection: occlusion and confusion with hard negative examples. Our approach is general and flexible enough to be applicable to any single-stage detectors. We implemented our occlusion and hard negative handling methods into four single-stage models, including SqueezeDet+ [86], YOLOv2 [68], SSD [56], and DSSD [25]. We demonstrated that our approach indeed improved the performance of four base models for pedestrian detection on Caltech [17] and CityPersons [93] datasets. One future work may be to apply our methods to other general object detection problems. Since our approach can be universally integrated with any general-purpose detectors, there is no fundamental limitation to extend our approach into other domains.

Chapter 4

Self-Supervised Feature Super-Resolution for Small Objects

4.1 Overview

In this chapter, we focus on how to improve the performance of detecting small objects in the proposal-based detection framework such as Faster R-CNN [69].

The proposal-based detectors fundamentally suffer from the issue that the region proposals for small objects are too small to identify. For instance, Huang *et al.* [41] show that mean Average Precision (mAP) scores of small objects are roughly 10 times lower than those of large objects. For small proposals, the region of interest (RoI) pooling layer often extracts replicated feature vectors as inputs to a box predictor, which eventually makes a prediction without enough detail information for small objects. Moreover, it is likely that the position of a RoI pooled feature and its actual position in the image are mismatched [40]. Such distortion of RoI pooling can be partly alleviated by some advanced pooling techniques such as RoI align [31] and PrRoI pooling [42]. However, they do not provide additional information a box predictor can use to better detect small objects.

To enrich the information in small proposals, some previous studies exploit image

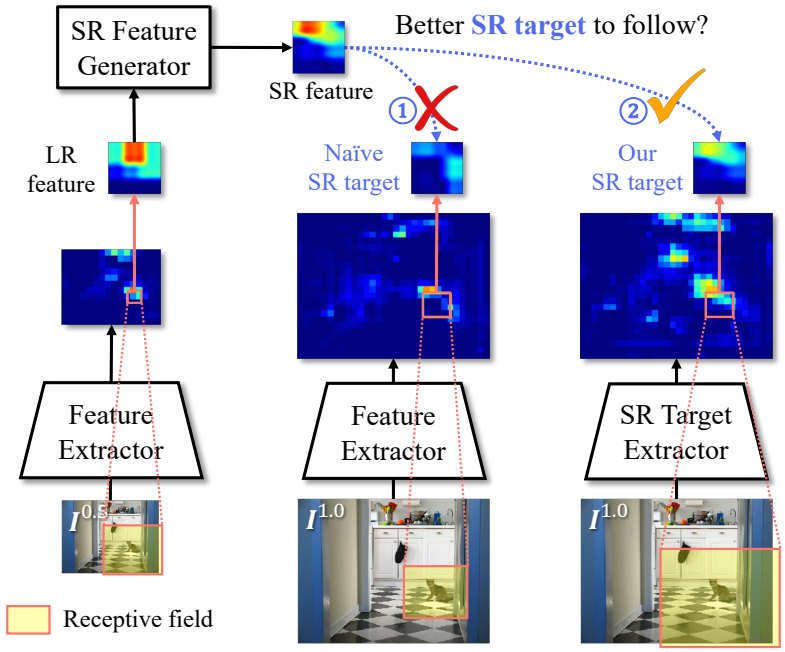


Figure 4.1: For feature-level super-resolution (SR), it is crucial to have direct supervision from high-resolution target features. However, if we extract them from the same feature extractor as low-resolution (LR) features, the relative receptive fields of two features are mismatched (①), which can significantly misguide the SR feature generator. We introduce SR target extractor that provides proper high-resolution features while keeping the relative receptive fields the same (②).

super-resolution [24, 71, 30]. Due to the serious inefficiency of super-resolving the whole image, Bai *et al.* [2] propose to super-resolve image pixels of the small proposals to be similar to those of large proposals. However, its RoI super-resolution cannot take the context information into account since it focuses only on the RoIs. This drawback can be partly resolved by the feature-level super-resolution which utilizes the context information as the features of proposals are extracted with large receptive fields of consecutive convolution operations. Particularly, Perceptual GAN [48] exploits Generative Adversarial Networks (GAN) [28] to super-resolve the features of proposals and improves the detection accuracy on small objects.

However, existing feature-level super-resolution models for small object detection have one significant limitation: *lack of direct supervision*. That is, their super-

resolution models are trained without explicit target features, which results in training instability and restricted quality of super-resolution features. For the image retrieval task, Tan *et al*. [78] show that the feature-wise content loss between the pairs of low-resolution and its high-resolution features leads to better super-resolution features with faster convergence.

Not only that it is important for better training to construct proper high-resolution features as targets, our analysis also reveals that it is critical to match the relative receptive fields between the pairs, especially for small RoIs (Figure 4.1). That is, in the image retrieval task of [78] where only features of overall images are considered, the relative receptive fields are not much different between the pairs of high and low-resolution features. On the other hand, the difference is extremely large for small RoIs that are common in the object detection tasks, and it leads to poor quality of super-resolution of small proposals.

With this context, the contributions of this work are three-fold:

(1) We thoroughly inspect existing feature-level super-resolution methods for small object detection and discover the performance is significantly improved by (i) utilizing high-resolution target features as supervision signals and (ii) matching the relative receptive fields of input and target features.

(2) We propose a novel feature-level super-resolution approach that is orthogonally applicable on top of any proposal-based detectors with feature pooling. It fully takes advantage of direct supervision of the high-resolution target features that are created by our new target extractor, which exploits atrous convolution with requiring no additional parameters as it shares parameters with CNN backbone of the base detector. Moreover, we propose an iterative refining generator as a novel way to super-resolve features.

(3) Our approach significantly improves the performance of Faster R-CNN for small object detection on three benchmark datasets of Tsinghua-Tencent 100K [99], PASCAL VOC [21] and MS COCO [55] with various CNN backbones such as ResNet-50, ResNet-101 [32] and MobileNet [35]. The improvement for small objects is remarkably large, and encouragingly, those for medium and large objects are nontrivial

too. As a result, we achieve new state-of-the-art performance on Tsinghua-Tencent 100K and highly competitive results on both PASCAL VOC and MS COCO.

4.2 Mismatch of Relative Receptive Fields

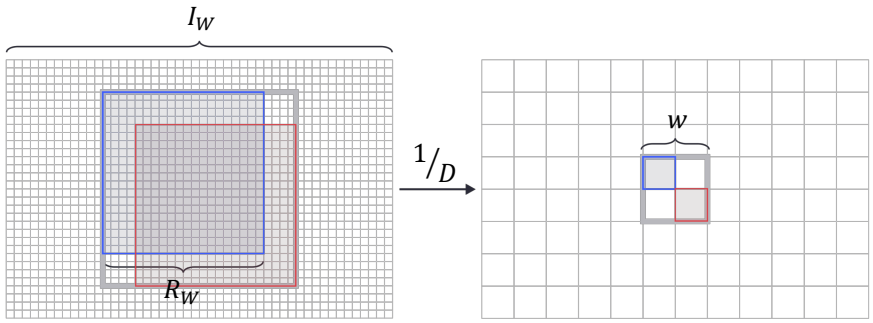


Figure 4.2: Suppose an input image with width of I_W and its corresponding feature map resized at ratio of $1/D$. An RoI with width of w (grey box) on the feature map has the receptive field surrounded by the grey box on the image. Meanwhile, a single feature cell on the feature map (*i.e.* blue box) has the receptive field with width of R_W on the image. The receptive fields of nearby feature cells are highly overlapped on the image space as described with shared colors.

In this section, we discuss why matching relative receptive fields is important to obtain adequate pairs of low-resolution input features and high-resolution target features. Based on this discussion, in the following section, we propose our novel super-resolution target extractor.

One straightforward way to obtain the pairs is to take a large RoI from the original image and its smaller version from the downsampled image [78]. Unfortunately, the features of these pairs do not exactly match up in terms of relative receptive fields. In order to clearly see why such discrepancy occurs, we present an intuitive example in Figure 4.2 with notations. Considering only one horizontal axis for easiness of discussion, the absolute receptive field (ARF) for the feature of an RoI with width of w is

$$ARF(w) = R_W + (w - 1) \times D. \tag{4.1}$$

The relative receptive field (RRF), defined as ARF relative to the size of an image I_W, is

$$\begin{aligned} RRF(w, I_W) &= ARF(w)/I_W \\ &= \left(R_W + (w-1) \times D\right)/I_W. \end{aligned} \tag{4.2}$$

Let us discuss how RRF differs as the input image resizes. In $\times 0.5$ downsampled input image, the width of the image is $I_W/2$ and that of the RoI on the feature map is $w/2$. We define the discrepancy in RRF (DRRF) of the RoIs between the original and downsampled images as

$$\begin{aligned} DRRF_{1/2}(w, I_W) &= \frac{RRF(w/2, I_W/2)}{RRF(w, I_W)} \\ &= \frac{\left(R_W + (w/2-1) \times D\right)/(I_W/2)}{\left(R_W + (w-1) \times D\right)/I_W} \\ &= \frac{2R_W + wD - 2D}{R_W + wD - D} \\ &= \frac{2R_W + 2wD - 2D}{R_W + wD - D} - \frac{2wD - 2D - wD + 2D}{R_W + wD - D} \\ &= 2 - \frac{wD}{R_W + wD - D} \\ &= 2 - \frac{w}{\left(\frac{R_W}{D} - 1\right) + w} \\ &= 2 - \frac{w}{c + w} \end{aligned} \tag{4.3}$$

where $c = \frac{R_W}{D} - 1$. As R_W and D are the constants determined by a backbone structure, c is also a constant.

According to Eq.(4.3), as w approaches to 0, DRRF converges to 2, while it goes to 1 as w increases. That is, for a small RoI, the relative receptive field (RRF) of the same RoI can be as $\times 2$ as different between the original and downsampled images. On the other hand, the RRFs become similar if the size of a proposal is sufficiently large. For example, for an RoI with $w = 4$ from the input image with $I_W = 1600$, if we use Faster R-CNN with ResNet-50 backbone where $R_W = 291$ and $D = 16$, then $DRRF_{1/2}(4, 1600)$ is close to 1.8. That is, the RRF of the RoI from the downsampled

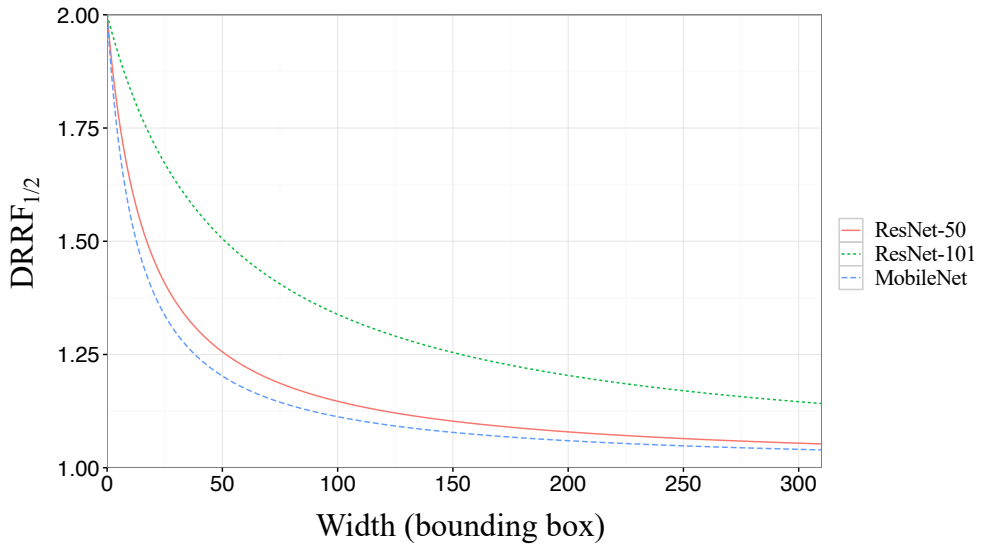


Figure 4.3: As the size of the bounding box decreases, DRRF, as defined in equation 4.3, increases. It implies that if the size of a proposal is large, the discrepancy in RRF is not significant. However, it can be significantly large when the size of a proposal is small.

image is around 1.8 times larger than that from the original image. Figure 4.3 shows how $DRRF_{1/2}$ changes as the size of a bounding box w increases for three different backbones. The plots are different by backbones since R_W's are different as 291, 835 and 219 for ResNet-50, ResNet-101 and MobileNet, respectively, although D's are the same for 16.[1] Tan *et al*. [78] deal with the image retrieval task where the entire image features are super-resolved, so the discrepancy in RRF is not significant. On the contrary, for the super-resolution of small RoIs for detection as in our work, the discrepancy in RRF is critically large and it can seriously misguide the super-resolution model.

4.3 Our Approach

We propose a novel method that enhances the quality of feature super-resolution for small object detection, based on two key ideas: (i) direct supervision for the super-

[1] We calculate ARF referring to TensorFlow library.

resolution generator and (ii) the receptive field matching via atrous convolution. We introduce four additional components on top of the base detector model: SR feature generator and discriminator, SR target extractor and small predictor. As a GAN-based model, the SR feature generator produces high-resolution features under the guidance of the SR feature discriminator using the features from the SR target extractor as targets. Additionally, the small predictor is a replica of the predictor in the base detector, which we call as the large predictor. The large predictor computes the confidence of classification and localization for large proposals as done in normal detectors, whereas the small predictor carries out the same task for small proposals that are enhanced first by the SR feature generator. We set the thresholds for the small proposals as (32×32) for Tsinghua-Tencent and (96×96) for VOC and COCO datasets. Figure 4.4 shows the overall architecture of our model. We explain the model based on Faster R-CNN [69], although our approach is integrable with any proposal-based detector

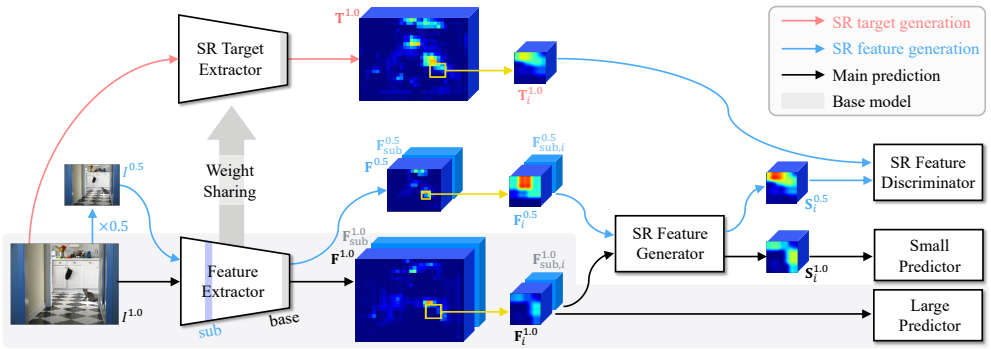


Figure 4.4: Overall model architecture. Four new components are proposed on top of the base detector model: SR target extractor (section 4.3.1), SR feature generator and discriminator (section 4.3.2), and small predictor. As a GAN-based model, the SR feature generator learns to create high-resolution features under the guidance of the SR feature discriminator using the features from the SR target extractor as targets (section 4.3.3). At inference (specified as *main prediction* arrows), a large proposal is directly passed to the large predictor for classification and localization, while a small proposal is first super-resolved by the SR feature generator and then passed to the small predictor (section 4.3.4).

with feature pooling.[2]

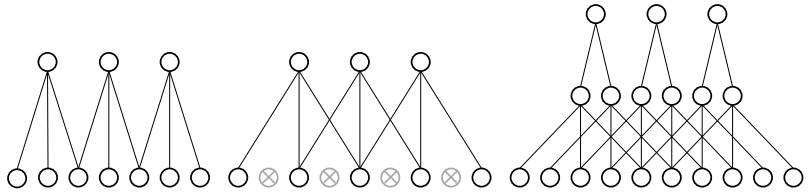
4.3.1 Super-Resolution Target Extractor

We denote the original input image by $I^{1.0}$ and its $\times 0.5$ downsampled image by $I^{0.5}$. We use $\mathbf{F}_i^{1.0}$ to denote the feature for the i-th RoI from the original image. In section 4.2, we reveal that it is not a good idea to use $\mathbf{F}_i^{1.0}$ as a super-resolution target for $\mathbf{F}_i^{0.5}$. Instead, we need to extract proper high-resolution target feature denoted by $\mathbf{T}_i^{1.0}$ that has similar RRF with low-resolution feature $\mathbf{F}_i^{0.5}$. To this end, we introduce an additional CNN feature extractor named *super-resolution target extractor* to generate $\mathbf{T}_i^{1.0}$ as in Figure 4.4. We let the SR target extractor share the same parameters with the CNN backbone (*i.e*. the normal feature extractor in the base detector), because they should not produce different features by channel for the same input.

One important requirement for the SR target extractor is to adequately address RRF at every layer where the receptive fields are expanded. In regular CNNs, the receptive fields are expanded whenever applying convolution or pooling layers whose filter sizes are greater than 1. Thus, our SR target extractor should be designed to cover the same expanded receptive fields whenever either of those layers are used in the CNN backbone. For parameter-free pooling layers, it can be easily achieved by increasing the filter size. However, for convolution layers, increasing the filter size is not valid as it makes the parameters different from those of the CNN backbone. Therefore, we employ atrous (dilated) convolution layer [9], which involves the same number of parameters as a regular convolution layer while its receptive fields are controlled by a dilation rate. We apply atrous convolution layers with dilation rate of 2 at every convolution layer with the filter size greater than 1 on the CNN backbone.

One additional treatment is for the stride. As shown in Figure 4.5(a), if the stride of convolution layer in the CNN backbone is not 1 (*e.g*. 2), it is not valid to simply use the same stride size for atrous convolution because it skips every other pixel as shown in Figure 4.5(b). This problem can be solved by applying atrous convolution

[2]Most two-stage proposal-based detectors use *feature pooling*, while a few models exploit *score pooling* such as RFCN [14].



(a) CONV (b) Atrous CONV (c) Atrous CONV + POOL

Figure 4.5: Connections between input and output nodes. (a) One convolution layer with filter size of 3 and stride of 2. (b) One atrous convolution layer with filter size of 3, stride of 2 and rate of 2. (c) The same atrous convolution layer as (b) with stride of 1, followed by one pooling layer with filter size of 2 and stride of 2.

with stride of 1 and then max pooling with 2 as in Figure 4.5(c).

In comparison with the DRRF from the existing feature extractor, let's take a closer look at the DRRF when SR target extractor is used. We denote it as $DRRF^{SR}_{1/2}$ and it is computed as Eq.(4.4). For the SR target extractor, the size of the receptive field corresponding to a single feature cell is approximately two times larger than that of the backbone feature extractor. Thus, R_W in Eq.(4.2) is replaced by $2R_W$ to compute RRF of the SR target extractor which is denoted as RRF^{SR}.

$$\begin{aligned} DRRF^{SR}_{1/2}(w, I_W) &= \frac{RRF(w/2, I_W/2)}{RRF^{SR}(w, I_W)} \\ &\approx \frac{\left(R_W + (w/2-1)\times D\right)/(I_W/2)}{\left(2R_W + (w-1)\times D\right)/I_W} \\ &= \frac{2R_W + wD - 2D}{2R_W + wD - D} \\ &= 1 - \frac{1}{\left(\frac{2R_W}{D} - 1\right) + w} \\ &= 1 - \frac{1}{c' + w} \end{aligned} \tag{4.4}$$

For ResNet-50 with $w = 1$, $DRRF^{SR}_{1/2} = 0.97$, which implies the RRFs are almost identical whereas $DRRF_{1/2} = 1.95$.

In summary, the SR target extractor consists of atrous convolution and pooling layers arranged to keep the same RRF as the CNN backbone while sharing the same parameters. The feature $\mathbf{T}^{1.0}_i$ from the SR target extractor is a better target to train the super-resolution model than $\mathbf{F}^{1.0}_i$ from the CNN backbone. Furthermore, $\mathbf{T}^{1.0}_i$ covers

larger receptive fields than $\mathbf{F}_i^{1.0}$; they contain more context information that may be useful to detect small objects.

4.3.2 Super-Resolution Feature Generator and Discriminator

Our feature-level super-resolution model is based on Generative Adversarial Networks (GAN) [28]. Its ultimate goal is to transform the pooled features $\mathbf{F}_i^{1.0}$ of small proposals to super-resolved features $\mathbf{S}_i^{1.0}$. In order to make a pair of low-resolution and high-resolution target features, we first downsample the original image at $\times 0.5$, obtain $\mathbf{F}_i^{0.5}$ for i-th proposal and pair it with $\mathbf{T}_i^{1.0}$ generated from the SR target extractor. That is, the super-resolution feature generator in Figure 4.6 is learned to iteratively refine $\mathbf{F}_i^{0.5}$ into the super-resolution features $\mathbf{S}_i^{0.5}$ so that $\mathbf{S}_i^{0.5}$ is as similar to $\mathbf{T}_i^{1.0}$ as possible.

For this objective, we design the feature-wise content ℓ_2 loss as

$$\mathbb{I}_{cont,i} = \begin{cases} \mathbf{1}, & \text{if } area_i^{1.0} > l_{cont} \wedge area_i^{0.5} \leq u_{cont} \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{4.5}$$

$$\mathcal{L}_{cont} = \sum_{i=1}^{N} \mathbb{I}_{cont,i} \|\mathbf{T}_i^{1.0} - \mathbf{S}_i^{0.5}\|_2^2. \tag{4.6}$$

where $area_i^{1.0}$ and $area_i^{0.5}$ indicate the area of i-th RoI on the original input image $I^{1.0}$ and downsampled image $I^{0.5}$, respectively, and l_{cont} and u_{cont} denote the lower and upper bounds of $area_i^{1.0}$ and $area_i^{0.5}$ regarding $\mathcal{L}_{cont}$.

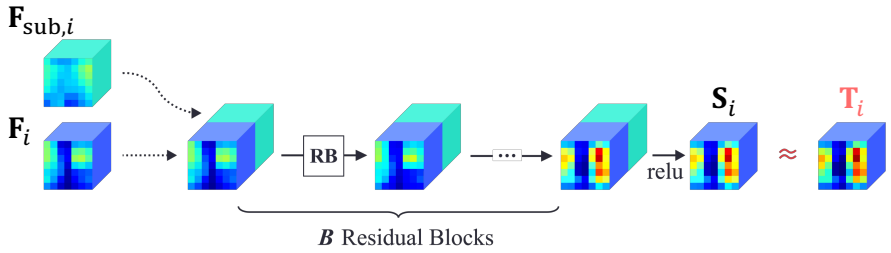


Figure 4.6: The *super-resolution feature generator*. It transforms the low-resolution input feature $\mathbf{F}_i$ into a super-resolution feature $\mathbf{S}_i$, with additional input $\mathbf{F}_{\text{sub},i}$. It iteratively refines the features via B residual blocks, each of which is the element-wise sum of the input feature and residual with two CONV layers as filters. At the end, only $\mathbf{F}_i$ part is sliced to be $\mathbf{S}_i$.

Note that only the RoIs that satisfy the condition of the indicator are used to compute $\mathcal{L}_{cont}$. If $area_i^{1.0}$ is too small, $\mathbf{T}_i^{1.0}$ is not a desirable target for $\mathbf{S}_i^{0.5}$ to follow due to the low-resolution of $\mathbf{T}_i^{1.0}$. On the other hand, if $area_i^{0.5}$ is too large, $\mathbf{F}_i^{0.5}$ is detail enough to not need further enhancement through super-resolution.

As inputs to the generator for training, we use both the features from the former layer $\mathbf{F}_{\text{sub},i}^{0.5}$ (**sub** layer) and the latter layer $\mathbf{F}_i^{0.5}$ (**base** layer). Since $\mathbf{F}_i^{0.5}$ only contains coarse and low-frequency information for a small RoI, we supplement its fine and high-frequency information $\mathbf{F}_{\text{sub},i}^{0.5}$ from the former layer.

For the SR feature discriminator, we use a multi-layer perceptron (MLP) with three layers. The discriminator is trained to be able to distinguish between $\mathbf{T}_i^{1.0}$ and $\mathbf{S}_i^{0.5}$, while the generator is trained to transform $\mathbf{F}_i^{0.5}$ into $\mathbf{S}_i^{0.5}$ indistinguishable from $\mathbf{T}_i^{1.0}$. To this end, adversarial losses ($\mathcal{L}_{gen}$, $\mathcal{L}_{dis}$) are defined as follows.

$$\mathbb{I}_{adv,i}^{+} = \begin{cases} \mathbf{1}, & \text{if } area_i^{1.0} > t_{adv} \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{4.7}$$

$$\mathbb{I}_{adv,i}^{-} = \begin{cases} \mathbf{1}, & \text{if } area_i^{0.5} \leq t_{adv} \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{4.8}$$

$$\mathcal{L}_{gen} = -\sum_{i=1}^{N} \mathbb{I}_{adv,i}^{-} \log D(\mathbf{S}_i^{0.5}) \tag{4.9}$$

$$\mathcal{L}_{dis} = -\sum_{i=1}^{N} \left(\mathbb{I}_{adv,i}^{+} \log D(\mathbf{T}_i^{1.0}) + \mathbb{I}_{adv,i}^{-} \log \left(1 - D(\mathbf{S}_i^{0.5})\right) \right) \tag{4.10}$$

where t_{adv} denotes a threshold for both $area_i^{1.0}$ and $area_i^{0.5}$ regarding $\mathcal{L}_{gen}$ and $\mathcal{L}_{dis}$. For instance, only high-resolution features corresponding to the large enough RoIs whose area ($area_i^{1.0}$) is larger than t_{adv} are involved in computing $\mathcal{L}_{dis}$. On the other hand, super-resolution features corresponding to the small enough RoIs whose area ($area_i^{0.5}$) is smaller than t_{adv} are used to compute both $\mathcal{L}_{gen}$ and $\mathcal{L}_{dis}$.

So far, we have discussed how the generator refines the low-resolution feature $\mathbf{F}_i^{0.5}$ to be similar to the target feature $\mathbf{T}_i^{1.0}$. However, our ultimate goal is to better detect small objects; thus, we need to train the generator to super-resolve features in

a way that they indeed help detect small objects well. To this end, we further train the generator as follows. After the generator produces the super-resolved features $\mathbf{S}_i^{1.0}$ from $\mathbf{F}_i^{1.0}$, we input it to the small box predictor. Then, we compute the classification loss ($\mathcal{L}_{cls}$) and localization loss ($\mathcal{L}_{loc}$) of the box predictor as in [69], and flow the gradient signals to the generator for fine-tuning. For $\mathcal{L}_{cls}$ and $\mathcal{L}_{loc}$, t_{main} is used to determine whether i-th RoI is treated as small or large. If $area_i^{1.0}$ is larger than t_{main}, $\mathbf{F}_i^{1.0}$ is passed into the large predictor. Otherwise, $\mathbf{F}_i^{1.0}$ is first super-resolved into $\mathbf{S}_i^{1.0}$ through the super-resolution feature generator and then passed to the small predictor. The values used for thresholds on different datasets are provided in Table 4.1.

Dataset	l_{cont}	u_{cont}	t_{adv}	t_{main}
Tsinghua-Tencent 100K [99]	16×16	32×32	32×32	32×32
PASCAL VOC [21], MS COCO [55]	16×16	128×128	96×96	96×96

Table 4.1: The lower/upper bounds (l_{cont}, u_{cont}) and thresholds (t_{adv}, t_{main}) used to filter out the invalid features of proposals for different losses on different datasets.

4.3.3 Training

We first train the base detector model, which consists of the feature extractor, region proposal network (RPN) and the large predictor. Then, the generator and discriminator are alternatively trained using the features ($\mathbf{F}_i^{1.0}$, $\mathbf{F}_i^{0.5}$ and $\mathbf{T}_i^{1.0}$) while freezing the feature extractors and RPN. The generator is trained under the guidance of the weighted sum of generator, content, classification and localization losses while the discriminator is trained only from the discriminator loss. Along with the GAN structure, the small predictor is simultaneously trained using the super-resolved features $\mathbf{S}_i^{1.0}$ from the classification and localization losses. Notice that we initialize the SR target extractor and small predictor using the weights of the feature extractor and the large predictor of the base detector, respectively.

Once both generator and discriminator converge, we further fine-tune the small and large predictors while freezing all the others. Fine-tuning is useful for the small predictor because it is trained only on super-resolved features which may not be per-

fectly identical to the target features. It also helps further boost up the performance by focusing solely on classification and localization losses. The large predictor is fine-tuned only with large proposals since the features of the small proposals are no longer passed into it.

4.3.4 Inference

Once training is done, the inference is much simpler. We only use the SR feature generator and the small predictor on top of the base model, which corresponds to the *main prediction* part in Figure 4.4. Given an input image $I^{1.0}$, we obtain the features from the CNN backbone $\mathbf{F}^{1.0}$. If the feature proposal is large, the large predictor takes it to make prediction on its class and location. On the other hand, if the feature proposal is small, it is super-resolved first using the SR feature generator and passed into the small predictor.

4.4 Experiment Settings

We evaluate the performance of our approach on Faster R-CNN [69] as the base network with various backbones (ResNet-50, ResNet-101 [32], and MobileNet [35]) on three benchmark datasets of Tsinghua-Tencent 100K [99], PASCAL VOC [21] and MS COCO [55].

4.4.1 Datasets

Tsinghua-Tencent 100K. Tsinghua-Tencent 100K [99] is a large benchmark about traffic signs with severe illuminance changes caused by weathers and complex backgrounds. It provides a traffic sign dataset in real world where the sizes of target objects are very small compared to the image size (2048×2048). The dataset has 6K train images and 3K test images. It divides the data in terms of size in the same way as MS COCO [55], which is categorized as small ($area \leq 32 \times 32$), medium ($32 \times 32 < area \leq 96 \times 96$) and large ($area > 96 \times 96$) objects. The portions of small, median and large objects are $(42, 50, 8)\%$, respectively. Due to such dominant

presence of small objects, Tsinghua-Tencent 100K is one of the best benchmarks to verify the performance of small object detection.

Following the protocol of [99], we evaluate for 45 classes that include more than 100 instances among 182 classes. While only recall and accuracy in terms of sizes are reported in [99], we additionally report F1 scores since they can balance the two metrics. The detection is counted as correct if IoU with the groundtruth is greater than or equal to 0.5.

PASCAL VOC & MS COCO. We also evaluate our model on PASCAL VOC [21] and MS COCO [55], although the ratio of small objects in these benchmarks are much less than Tsinghua-Tencent 100K. PASCAL VOC consists of 20 object categories with 5K *trainval* and 5K *test* images in 2007 and 11K *trainval* images in 2012. We use 2007 *trainval* + 2012 *trainval* for training and 2007 *test* set for test. MS COCO 2017 consists of 80 object categories with 115K *train*, 5K *val* and 20K *test-dev* images. We use the *train* set for training, and the *val* and *test-dev* set for test.

For PASCAL VOC, we use the mAP@.5 metric, which is the averaged AP over all classes when the matching IoU threshold with the groundtruth is greater than or equal to 0.5. For MS COCO, we use the mAP@.5:.95, which is the averaged mAP over different matching IoU thresholds from 0.5 to 0.95. We also divide the results on PASCAL VOC into three different categories according to object sizes; small (AP-S), medium (AP-M) and large (AP-L), as with MS COCO. We set the threshold to 96×96 for small proposals since the object sizes are much larger than those of Tsinghua-Tencent 100K.

4.4.2 Configuration Details

We generally follow the configurations used in Faster R-CNN [32, 69] for the base model. More specifically, for a **sub** and **base** layer in Figure 4.4, we use *conv1* block and *conv4* block for ResNet, and *conv4dw* and *conv11* for MobileNet. For the super-resolution part, the output channels of the first convolution layer before $\mathbf{F}_{\text{sub}}$ for ResNet and MobileNet are set to 512 and 256, respectively. Also, we set the number of

residual blocks (B) in the super-resolution feature generator to 3. In terms of training, we use stochastic gradient descent with momentum of 0.9, and train the generator twice for every training of the discriminator. Lastly, we implement all of our algorithms using TensorFlow [1, 41] and employ the implementation of third-party for RoI pooling[3] as well as RoI align [87].

4.5 Experiment Results

We evaluate our proposed feature-level super-resolution method on three benchmark datasets (section 4.5.1) and conduct ablation studies on different methods and architectures (section 4.5.2). In the end, we provide the qualitative results that demonstrate both effectiveness and weakness of our proposed method (section 4.5.3).

4.5.1 Quantitative Results

Tsinghua-Tencent 100K

We compare the performance of our model to the base models with three backbones as previously specified. We set the threshold for the size of small proposals to 32×32; only the proposals whose area is less than the threshold are treated as inputs to the super-resolution model.

Table 4.2 summarizes the performance on the Tsinghua-Tencent 100K *test* dataset. We resize the input images from 2048 to 1600 to make learning and inference faster as in [48]. The performance improvement by our approach is significant in the order of small (75.2→84.3 in F1 scores with ResNet-101), medium (92.2→94.6) and large objects (92.2→93.2). The large improvement on small objects are consistent for different CNN backbones such as 63.4→71.0 with MobileNet and 74.9→82.2 with ResNet-50.

One remark is that although we only super-resolve the small proposals, we obtain the performance gain for medium and large objects as well. It is partially because the large predictor is fine-tuned without considering small proposals, which is helpful to focus its modeling power on the medium and large objects. Another reason for

[3] https://github.com/endernewton/tensorflow

Model	Small			Medium			Large			Overall		
	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1
MobileNet	56.1	72.9	63.4	85.1	**84.3**	84.7	90.9	**83.6**	**87.1**	74.7	80.7	77.5
+ Ours	**62.7**	**81.7**	**71.0**	**87.6**	84.0	**85.7**	**91.5**	82.1	86.5	**78.5**	**83.1**	**80.7**
ResNet-50	68.8	81.9	74.9	90.8	93.1	91.9	91.6	92.3	91.9	82.5	89.2	85.7
+ Ours	**78.2**	**86.5**	**82.2**	**94.7**	**93.8**	**94.3**	**93.6**	**93.0**	**93.3**	**88.4**	**91.1**	**89.7**
ResNet-101	69.8	81.5	75.2	90.9	93.5	92.2	92.4	92.0	92.2	83.1	**89.2**	86.0
+ Ours	**86.6**	**82.1**	**84.3**	**95.5**	**93.7**	**94.6**	**93.7**	**92.7**	**93.2**	**91.9**	89.1	**90.5**

Table 4.2: Overall performance on Tsinghua-Tencent 100K *test* dataset. Our proposed model achieves consistent improvement over the base models regardless of the backbone structures.

improvement in the medium subset is that some proposals that eventually fall in the medium subsets are predicted using the small predictor, due to the offsets added to the proposals in the final step. Given the fact that about 14% of the total objects are in between 32×32 and 40×40, it may be a valid reason that explains the performance gain for the medium subset.

Comparison with the state-of-the-art methods. Table 4.3 shows that our proposed model achieves new state-of-the-art performance on Tsinghua-Tencent 100K dataset. In these experiments, we train our model using ResNet-101 as a backbone on the images with their original size. Throughout all the subsets, ours outperform all the previous state-of-the-art models especially in terms of F1 scores.

Model	Small			Medium			Large			Overall		
	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1
Zhu *et al*. [99]	87.0	82.0	84.4	94.0	91.0	92.5	88.0	91.0	89.5	–	–	–
Perceptual GAN [48]	89.0	84.0	86.4	96.0	91.0	93.4	89.0	91.0	89.9	–	–	–
Liang *et al*. [50]	**93.0**	84.0	88.3	97.0	**95.0**	95.9	92.0	**96.0**	93.9	–	–	–
SOS-CNN [59]	–	–	–	–	–	–	–	–	–	93.0	90.0	91.5
FRCNN (ResNet-101)	80.3	81.6	80.9	94.5	94.8	94.7	94.3	92.6	93.5	89.1	89.7	89.4
+ Ours	92.6	**84.9**	**88.6**	**97.5**	94.5	**96.0**	**97.5**	93.3	**95.4**	**95.7**	**90.6**	**93.1**

Table 4.3: Performance comparison with the state-of-the-art models on Tsinghua-Tencent 100K *test* dataset.

PASCAL VOC and MS COCO

Table 4.4–4.6 compare the performance of our model to the baselines on VOC 2007 *test*, COCO 2017 *val* and COCO 2017 *test-dev*, respectively. We observe the similar trend as in Tsinghua-Tencent 100K that the detection enhancement is more

Model	mAP	AP-S	AP-M	AP-L
MobileNet	73.2	5.1	39.3	75.9
+ Ours	**77.0**	**10.1**	**47.2**	**76.9**
ResNet-50	77.1	6.8	42.9	81.1
+ Ours	**79.1**	**10.5**	**47.9**	**81.4**
ResNet-101	78.8	5.9	46.2	82.3
+ Ours	**80.6**	**11.1**	**48.9**	**82.7**

Table 4.4: Detailed performance on VOC 2007 *test* set. S, M and L denote the subset of small ($area \leq 32 \times 32$), medium ($32 \times 32 < area \leq 96 \times 96$) and large ($area > 96 \times 96$) objects, respectively.

Model	AP-.5:.95	AP-.5	AP-.75	AP-S	AP-M	AP-L	AR-1	AR-10	AR-100	AR-S	AR-M	AR-L
MobileNet	19.4	38.7	17.1	3.5	16.6	30.6	20.4	33.5	35.8	11.7	34.1	51.8
+ Ours	**21.6**	**40.7**	**20.6**	**7.1**	**20.9**	**30.7**	**22.4**	**37.3**	**39.7**	**18.7**	**40.1**	**52.5**
ResNet-50	29.5	51.6	29.8	6.4	26.0	**45.3**	26.7	42.0	44.5	18.2	42.7	**61.8**
+ Ours	**31.0**	**53.7**	**32.0**	**10.0**	**28.6**	45.1	**27.7**	**44.3**	**46.8**	**23.7**	**46.6**	61.5
ResNet-101	31.9	54.5	32.6	7.6	27.9	48.9	28.4	43.8	46.5	19.7	44.4	63.8
+ Ours	**34.0**	**56.6**	**35.7**	**11.6**	**31.5**	**49.0**	**29.5**	**46.7**	**49.3**	**26.5**	**49.2**	**63.9**

Table 4.5: Detailed performance on COCO 2017 *val* set. AP and AR denote the average precision and average recall. Also, S, M and L denote the subset of small ($area \leq 32 \times 32$), medium ($32 \times 32 < area \leq 96 \times 96$) and large ($area > 96 \times 96$) objects, respectively. AR-$\{1, 10, 100\}$ means the average recall given $\{1, 10, 100\}$ detections per image.

Model	AP-.5:.95	AP-.5	AP-.75	AP-S	AP-M	AP-L	AR-1	AR-10	AR-100	AR-S	AR-M	AR-L
MobileNet	19.3	38.7	16.9	5.4	20.6	**29.2**	20.3	32.3	34.0	12.2	36.5	**53.2**
+ Ours	**21.9**	**41.0**	**21.0**	**10.9**	**23.8**	29.0	**22.4**	**36.4**	**38.1**	**19.3**	**41.3**	53.1
ResNet-50	29.5	52.0	29.8	10.2	31.5	**44.7**	27.0	41.7	43.6	20.1	46.8	64.7
+ Ours	**31.2**	**54.2**	**32.4**	**14.3**	**32.4**	**44.7**	**28.2**	**44.2**	**46.0**	**25.0**	**49.3**	**64.8**
ResNet-101	32.0	54.7	32.8	11.3	34.3	**48.1**	28.5	43.6	45.7	21.5	48.9	**67.3**
+ Ours	**34.2**	**57.2**	**36.1**	**16.2**	**35.7**	**48.1**	**29.9**	**46.7**	**48.8**	**28.1**	**51.8**	67.2

Table 4.6: Detailed performance on COCO 2017 *test-dev* set. The notations are consistent with Table 4.5.

significant in the order of small, medium and large objects.

4.5.2 Ablation Experiments

We further provide the results of the experiment on different super-resolution methods as well as varied architectures of the proposed model.

Comparison of Super-Resolution Methods

In this section, we perform an ablation study to analyze different super-resolution methods both quantitatively and qualitatively. We use ResNet-50 as the CNN backbone. We compare our super-resolution approach with two inferior variants; (1) SR without supervision: the model without the content loss ($\mathcal{L}_{cont}$) and (2) SR with naïve supervision: the model trained using the target features from the base feature extractor instead of our SR target extractor.

Model	Small	Medium	Large	Overall
Base model	74.9	91.9	91.9	85.7
+ SR (w.o. supervision)	76.8	93.6	**93.3**	87.5
+ SR (Naïve supervision)	74.4	91.8	92.3	85.3
+ SR (Ours)	**82.2**	**94.3**	**93.3**	**89.7**

Table 4.7: Comparison of F1 scores between super-resolution methods with ResNet-50 on Tsinghua-Tencent 100K.

Table 4.7 compares F1 scores of different super-resolution models on Tsinghua-Tencent 100K. The other two SR variants obtain only limited performance gains compared to the base model. On the other hand, our SR model achieves significant performance gains, especially for the small subsets. One remark here is SR without supervision performs better than SR with naïve supervision, which implies the improper supervision due to the mismatch of RRF can degrade the performance.

Figure 4.7 qualitatively visualizes the superiority of our model compared to the other super-resolution methods: SR without supervision and SR with naïve supervision. SR without supervision does not improve the features much compared to the low-resolution features and the similar pattern appears for SR with naïve supervision although there are the target features (Naïve). On the other hand, the low-resolution input features (LR) are reasonably well super-resolved into their target features (Ours) through our super-resolution model.

To compare the target features from the existing feature extractor and our proposed SR feature extractor, we further provide the feature maps as shown in Figure 4.8. We

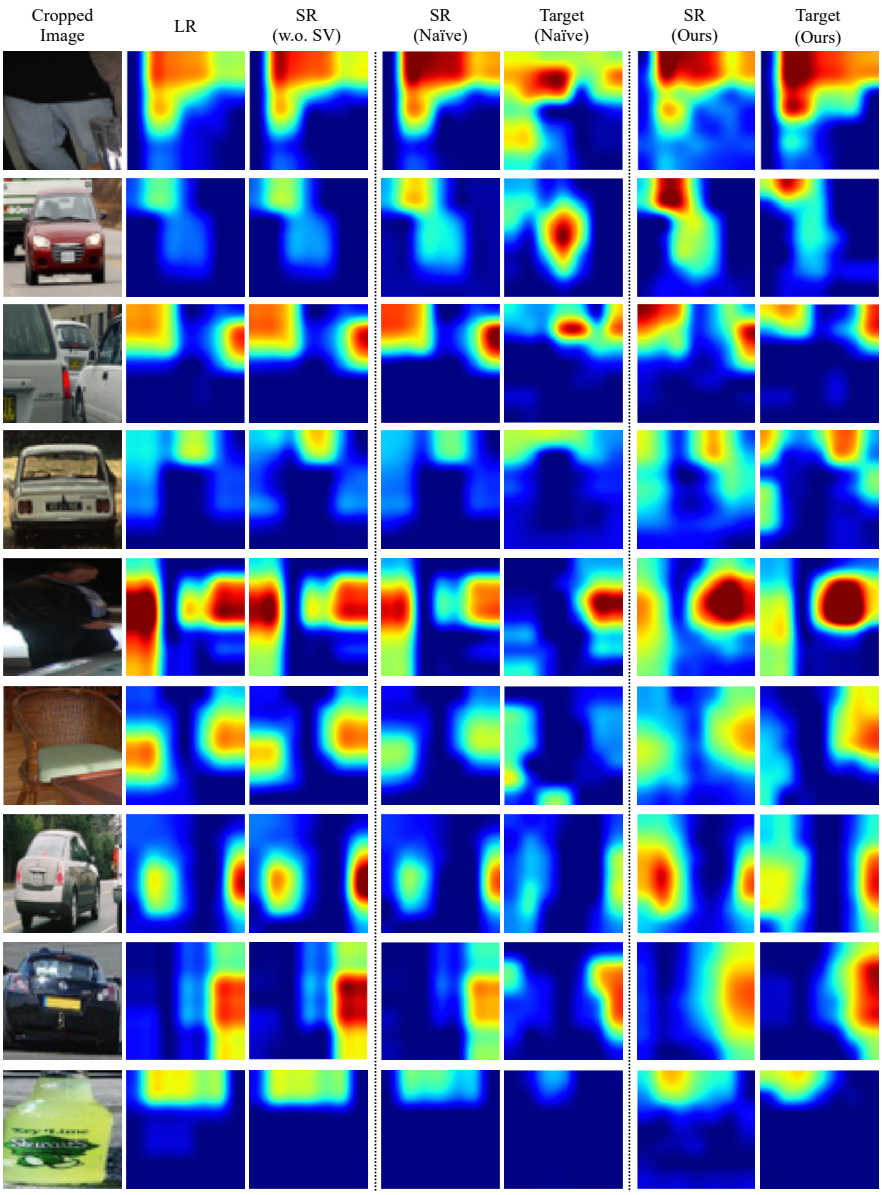


Figure 4.7: The qualitative results for how RoI features differ by different super-resolution methods on PASCAL VOC dataset. The low-resolution features (LR) extracted from the cropped images are super-resolved to be SR (w.o. SV), SR (Naïve) and SR (Ours) using SR without supervision, with naïve supervision and ours, respectively. SR without supervision does not make much improvement compared to the input feature. Such tendency remains unchanged for the SR with naïve supervision method. On the other hand, ours look very close to its target feature.

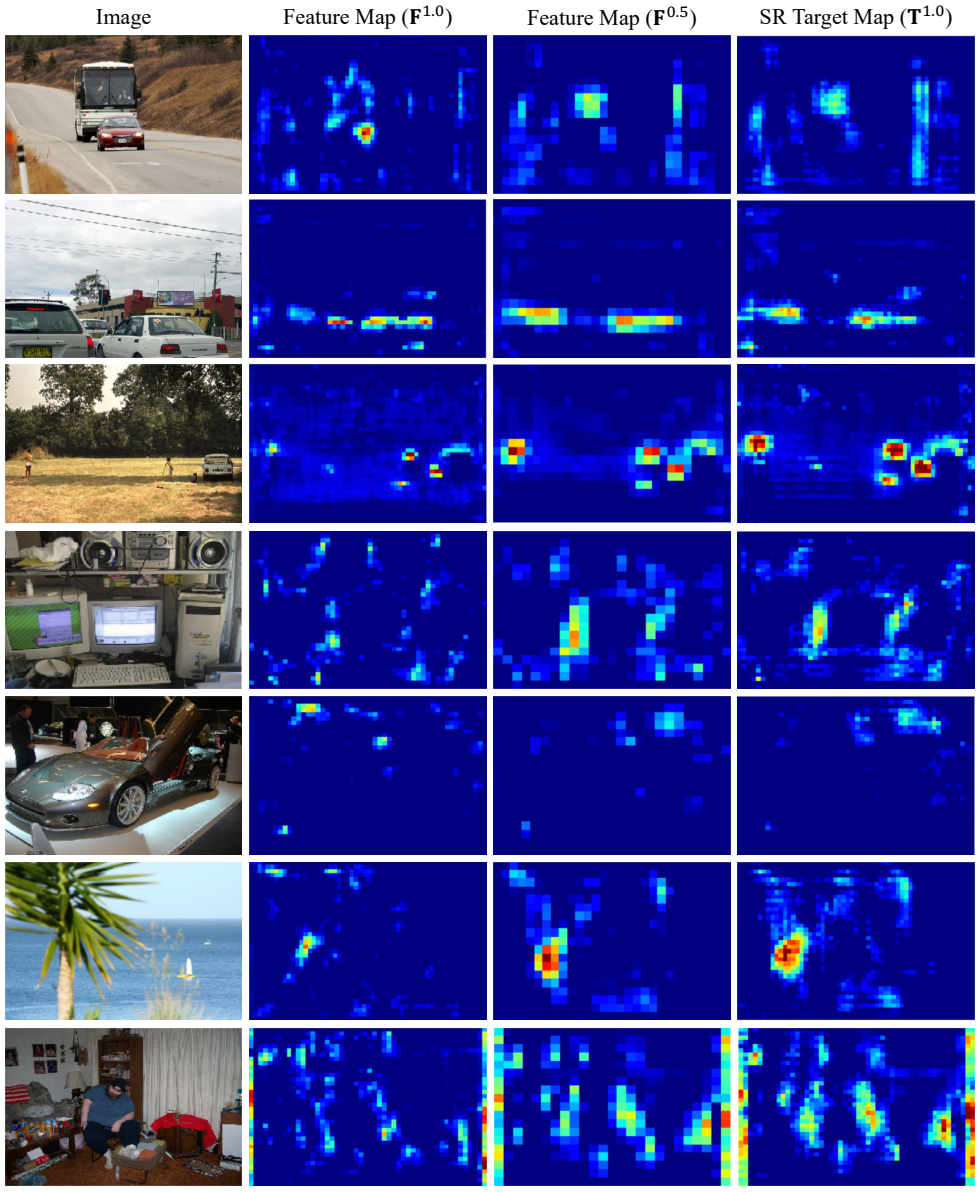


Figure 4.8: Comparison on the feature maps from different feature extractors. Both high-resolution ($F^{1.0}$) and low-resolution ($F^{0.5}$) feature maps are extracted from the existing feature extractor using high and low-resolution images, respectively, whereas the high-resolution target feature maps ($T^{1.0}$) are extracted from our proposed SR feature extractor.

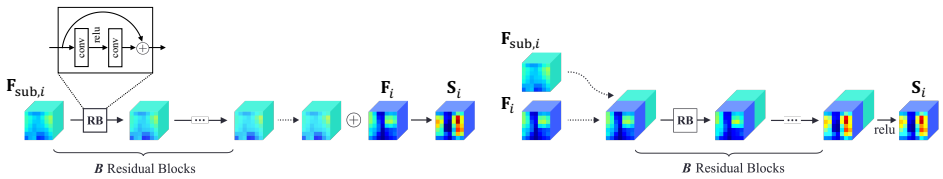
can easily tell the feature maps from the existing feature extractor $\mathbf{F}^{1.0}$ are significantly different from the low-resolution feature maps $\mathbf{F}^{0.5}$. However, the feature maps from

the SR feature extractor $\mathbf{T}^{1.0}$ are fairly close to the low-resolution feature maps $\mathbf{F}^{0.5}$, which demonstrates the validity of using our proposed SR feature extractor.

Experiments of Model Architecture

In this section, we present more experiment results for our model architecture. More specifically, we measure the contribution of each key component on the performance of our model. The following results are based on Tsinghua-Tencent 100K [99] with ResNet-50 [32] as a backbone unless otherwise stated.

The first experiment is on the structure of the super-resolution target generator. Figure 4.9 shows two different structures of the generator. Figure 4.9a is the structure of the generator proposed in Perceptual GAN [48] whereas Figure 4.9b describes the generator used in our model. In Figure 4.9a, the feature ($\mathbf{F}_{\text{sub},i}$) extracted from the **sub** layer is enhanced through $B(=6)$ residual blocks and combined with the feature ($\mathbf{F}_i$) from the **base** layer at the end. Each residual block consists of two 3×3 convolution layers followed by batch normalization. On the other hand, our proposed generator consists of $B(=3)$ residual blocks which take the concatenated feature as an input. After iteratively refined through the residual blocks, the first half of the feature is sliced out. Table 4.8 shows the meaningful increases in metrics on both small and medium objects using our generator compared with the generator proposed in Perceptual GAN.



(a) Perceptual GAN [48] (b) Ours

Figure 4.9: The structures of Perceptual GAN and our super-resolution feature generator.

Generator type	Small			Medium			Large			Overall		
	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1
Perceptual GAN	76.1	86.3	80.9	92.1	**94.3**	93.2	93.4	**93.0**	93.2	87.3	90.4	88.8
Ours	**78.2**	**86.5**	**82.2**	**94.7**	93.8	**94.3**	**93.6**	**93.0**	**93.3**	**88.4**	**91.1**	**89.7**

Table 4.8: Comparison on the different architectures of the super-resolution feature generator with ResNet-50 on Tsinghua-Tencent 100K.

For the next ablation study, we compare the performance by varying the **sub** layer from *conv1* to *conv3*. As stated in section 4.3.2, we extract sub-features from the earlier layer to secure the fine and high-frequency information. The results provided in Table 4.9 align with our assumption that the features from the earlier layer contain more detailed information so that they help to identify small objects better.

Layer name	Small			Medium			Large			Overall		
	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1
conv1	**78.2**	**86.5**	**82.2**	**94.7**	**93.8**	**94.3**	**93.6**	93.0	**93.3**	**88.4**	**91.1**	**89.7**
conv2	76.7	86.0	81.1	93.2	**93.8**	93.5	93.5	**93.1**	**93.3**	87.6	90.2	88.9
conv3	77.2	75.2	76.2	92.6	92.7	92.7	93.4	91.2	92.3	86.9	85.8	86.3

Table 4.9: Comparison on the super-resolution feature generators using different sub layers with ResNet-50 on Tsinghua-Tencent 100K.

Lastly, we compare the model architectures with single unified predictor and two separated predictors: small and large predictors. In fact, we previously designed our model to have one shared predictor, but we changed our model to have two separate predictors because the super-resolved features cannot perfectly imitate the high-resolution target features. According to Table 4.10, adding a small predictor (Separated) gives slight improvement over the model with only the large predictor (Unified). If one considers time/memory complexity more important, only one shared predictor can be employed.

Predictor	Small			Medium			Large			Overall		
	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1
Unified	77.7	86.4	81.8	94.6	**94.0**	**94.3**	93.3	92.9	93.1	88.0	**91.1**	89.5
Separated	**78.2**	**86.5**	**82.2**	**94.7**	93.8	**94.3**	**93.6**	**93.0**	**93.3**	**88.4**	**91.1**	**89.7**

Table 4.10: Comparison on the number of predictors used with ResNet-50 on Tsinghua-Tencent 100K.

4.5.3 Qualitative Results

We have examined the prediction examples to see if our approach has particular weaknesses. Figure 4.10 shows some of the failure cases of our model. For each row, we present the test result of the base model on the left, our model in the middle and groundtruth on the right. As illustrated in Figure 4.10, the most common failure cases of our model are due to false positives. For the objects that the base model detects as the background (false negatives), our model recognizes them as objects but for wrong classes (false positive). For instance, in the first row, our model recognizes the *p130* sign as *pm30* which the base model recognizes as the background. In fact, false positives tend to lower the detection metrics such as F1 score and mAP than false negatives. However, given the fact that the performance of our model is significantly higher than the base model, we can infer our model makes correct predictions (true positives or true negatives) more than the base model in general.

Figure 4.11–4.13 demonstrate the superiority of our model with some selected examples from Tsinghua-Tencent 100K, PASCAL VOC and COCO datasets. For each pair, we show the test results of the base model on the left and our model on the right. Compared to the base model, our model often detects small objects better with higher confidence.

4.6 Conclusion

We proposed a novel feature-level super-resolution approach to improve small object detection for the proposal-based detection framework. Our method is applicable on top of any proposal-based detectors with feature pooling. The experiments on Tsinghua-Tencent 100K, PASCAL VOC and MS COCO benchmarks validated our super-resolution approach was indeed effective to detect small objects. In particular, our work proved that it is important to provide direct supervision using proper high-resolution target features that share the same relative receptive field with the low-resolution input features.

Figure 4.10: Failure cases on Tsinghua-Tencent 100K (row1 and 2), PASCAL VOC 2007 (row3 and 4) and MS COCO 2017 (row5 and 6) datasets. For Tsinghua-Tencent 100K, green, red and blue rectangles represent true positives, false positives and false negatives, respectively. For each row, we show the test result of the base model (left), our model (middle) and groundtruth (right). The background is cropped out of some images for better visualization.

Figure 4.11: Detection examples on Tsinghua-Tencent 100K *test* dataset. Green, red and blue rectangles represent true positives, false positives and false negatives, respectively. Each pair indicates the results from the base model (left) and our model (right)

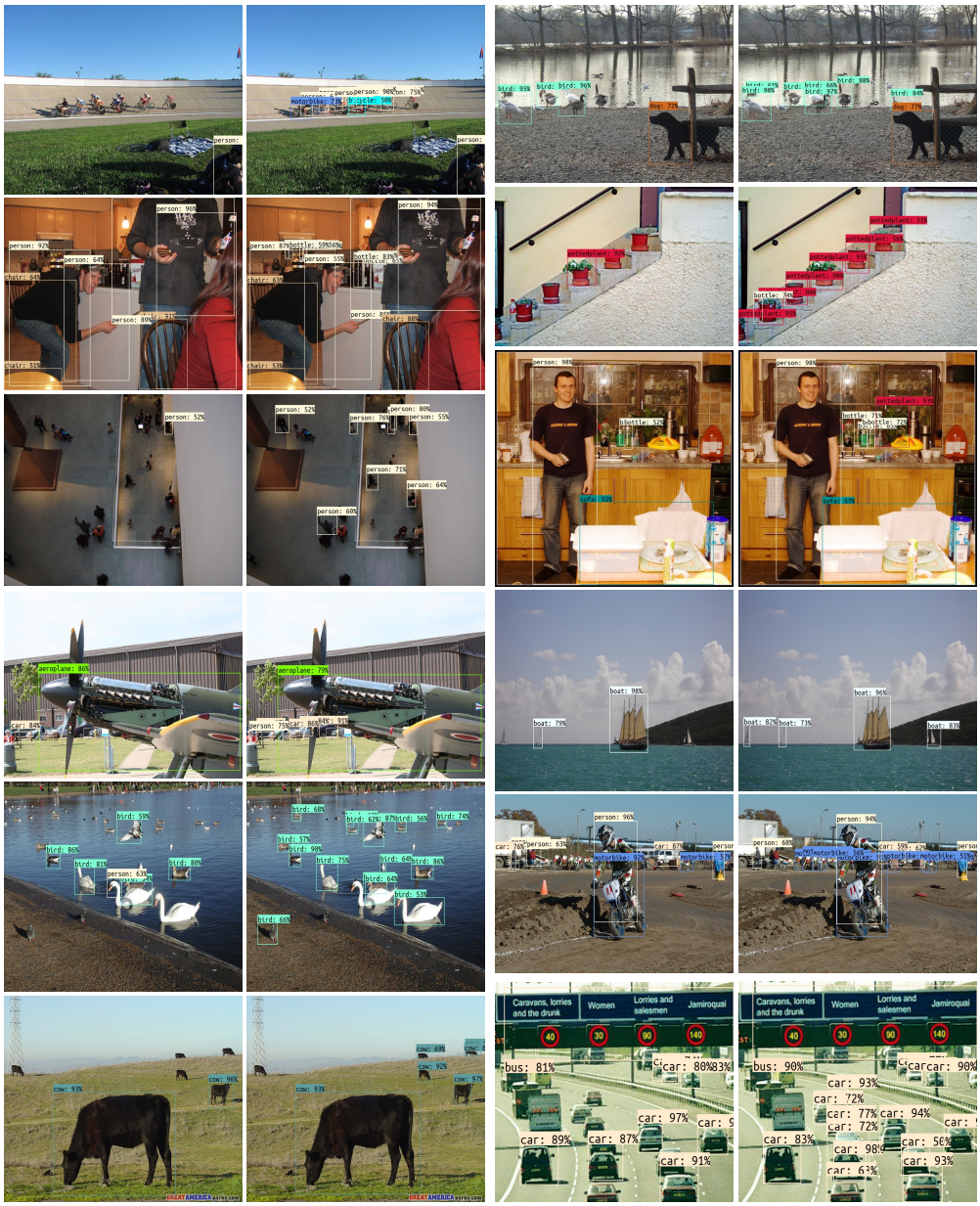


Figure 4.12: Detection examples on PASCAL VOC 2007 *test* dataset. For each pair, we show the test results of the base model (left) and our model (right). The background is cropped out of some images for better visualization.

Figure 4.13: Detection examples on MS COCO 2017 *val* dataset. For each pair, we show the test results of the base model (left) and our model (right). The background is cropped out of some images for better visualization.

Chapter 5

Rectified Class Activation Mapping for Weakly Labeled Objects

5.1 Overview

Many object detection algorithms such as Faster R-CNN [69], YOLO [67], SSD [56], R-FCN [14] and their variants [31, 68, 54] have successfully solved challenging benchmarks of object detection [21, 55]. However, due to the necessity of heavy manual labor for bounding box annotations, weakly supervised object localization (WSOL) have drawn great attention in the computer vision research [97, 76, 10, 95, 94, 11, 89]. Contrast to fully-supervised object detection, WSOL solely relies on image-level labels to localize an object in an image. Thus, previous studies on WSOL utilize the activations of feature maps from the last convolutional layer to generate class activation maps (CAM) from which bounding boxes are estimated.

Since the global average pooling (GAP) based CAM method [97] was introduced, most of previous studies have followed its convention to first generate CAMs and extract object locations out of them. However, this approach suffers from severe underestimation of an object region since the discriminative region activated through classification training is often much smaller than the object's actual region. For instance,

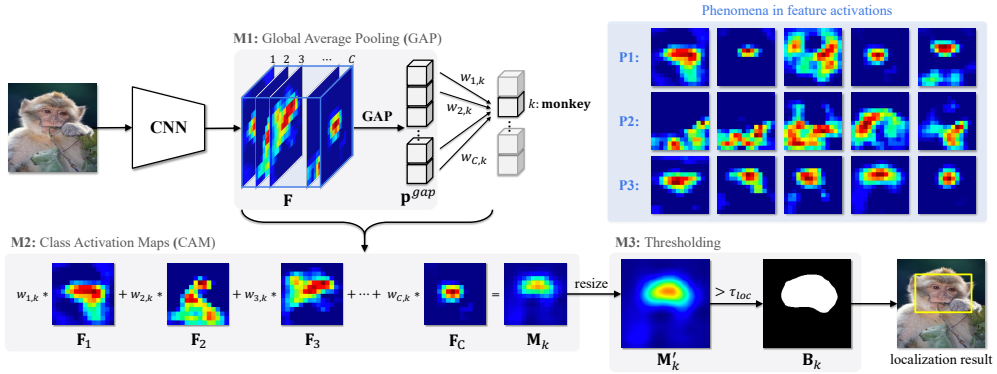


Figure 5.1: The overview of the GAP-based CAM localization [97]. We investigate three important phenomena of the feature maps (**F**). **P1.** The areas of the activated regions largely differ by channels. **P2.** The activated regions corresponding to the negative weights ($w_c < 0$) often cover large parts of the target object (*e.g. monkey*). **P3.** The most activated regions of each channel significantly overlap at small region. The GAP-based CAM model which consists of three modules (**M1**–**M3**) in gray boxes does not take these phenomena into account, which results in the localization being limited to small discriminative regions of an object. We elaborate a problem of each module followed by a proposed solution in section 5.2.2–5.2.4.

according to the CAM ($\mathbf{M}'_k$) in Figure 5.1, the classifier focuses on the head part of the *monkey* rather than the whole body, since the activations of the head is enough to correctly classify the image as *monkey*. Thus, the bounding box reduces to delineate the small high-activated region only. To resolve this problem, recent studies have devised architectures to obtain larger bounding boxes; for example, it erases the most discriminative region and trains a classifier only using the regions left, expecting the expansion of activation to the next most discriminative regions [3, 84, 76, 10, 43, 94, 49, 85, 34, 11]. These methods have significantly improved the performance of WSOL as well as other relevant tasks such as semantic segmentation.

In this work, we propose a different approach; instead of endeavoring to expand activations by devising a new architecture, we focus on correctly utilizing most of the information that already exists in feature maps. The major contribution of this work is two-fold. First, we reveal three fundamental issues in the way that the GAP-based CAM uses the feature map activations, which cause localization limited to small region

of an object. Second, we propose three simple but robust techniques that alleviate the problems. As a result, our solution is easily applicable to any WSOL algorithms using the GAP-based CAM, and in our experiments, we achieve the new state-of-the-art performance on both CUB-200-2011 and ImageNet-1K datasets.

We outline the three issues and our solutions as follows.

1. Usually, the areas of the activated regions largely differ by channels. (Figure 5.1.P1). However, a GAP operation is biased to assign a higher weight to a channel with small activation area, which results in the small region being highly weighted in the CAM. Therefore, our solution is to replace a GAP layer with a *thresholded average pooling* (TAP) layer, which only takes into account the activations greater than a given threshold for average pooling.

2. Ideally, the activated regions in the feature maps with negative weights are supposed to be *no-object regions* (*e.g*. background); however, they often correspond to less important object regions (*e.g*. monkey's body) such as (Figure 5.1.P2). As a result, less important object regions are suppressed in the CAM by the features with negative weights. Therefore, our solution is simply clamping the negative weight to zero so that all the relevant activations securely contribute to localization.

3. The most activated regions largely overlap across different channels (Figure 5.1.P3). Since the CAM sums all the weighted channels and the bounding box is determined using the maximum value of the CAM as reference, small overlapped regions with too high activation values become overdominant to the localization. Therefore, we simply replace the maximum value with a percentile as a thresholding reference, which helps stably and robustly extract proper bounding boxes.

5.2 Our Approach

We first review how the GAP-based CAM [97] localizes an object in WSOL (section 5.2.1). We then elaborate its three fundamental problems that cause the localization to be limited to small discriminative regions, followed by our solutions to alleviate the problems (section 5.2.2–5.2.4).

5.2.1 GAP-based CAM Localization

In the CNN that is trained for image classification, a class activation map (CAM) is the weighted average of the feature maps from the last convolutional (CONV) layer with the weights from the fully connected (FC) layer. Let the feature map be $\mathbf{F} \in \mathbb{R}_{\geq 0}^{H \times W \times C}$ where $\mathbb{R}_{\geq 0}$ is a non-negative real number. $\mathbf{F}_c \in \mathbb{R}_{\geq 0}^{H \times W}$ denotes c-th channel of $\mathbf{F}$ where $c = 1, \ldots, C$. As described in Figure 5.1, to generate a CAM, $\mathbf{F}$ is passed into a global average pooling (GAP) layer that averages each $\mathbf{F}_c$ spatially and outputs a pooled feature vector, $\mathbf{p}^{\text{gap}} \in \mathbb{R}_{\geq 0}^{C}$ as follows,

$$p_c^{\text{gap}} = \frac{1}{H \times W} \sum_{(h,w)} \mathbf{F}_c(h, w), \tag{5.1}$$

where p_c^{gap} denotes a scalar of $\mathbf{p}^{\text{gap}}$ at c-th channel, and $\mathbf{F}_c(h, w)$ is an activation of $\mathbf{F}_c$ at spatial position (h, w).

The pooled feature is then transformed into K-dim logits through the FC layer where K is the number of classes. We denote the weights of the FC layer as $\mathbf{W} \in \mathbb{R}^{C \times K}$. Hence, the CAM for a class k denoted as $\mathbf{M}_k$ is computed as

$$\mathbf{M}_k = \sum_{c=1}^{C} w_{c,k} \cdot \mathbf{F}_c, \tag{5.2}$$

where $\mathbf{M}_k \in \mathbb{R}^{H \times W}$ and $w_{c,k}$ is an (c, k) element of $\mathbf{W}$.

For localization, $\mathbf{M}'_k$ is first generated by resizing $\mathbf{M}_k$ to the original image size. With a localization threshold

$$\tau_{loc} = \theta_{loc} \cdot \max \mathbf{M}'_k, \tag{5.3}$$

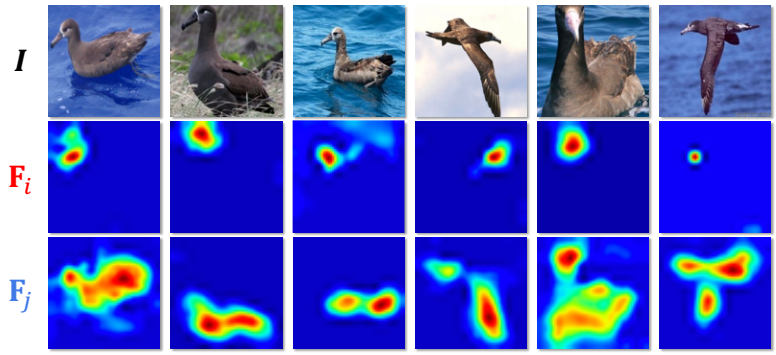
a binary mask $\mathbf{B}_k$ identifies the regions where the activations of $\mathbf{M}'_k$ is greater than τ_{loc}: $\mathbf{B}_k = \mathbb{1}(\mathbf{M}'_k > \tau_{loc})$. Finally, the localization is predicted as the bonding box that circumscribes the contour of the regions with the largest positive area of $\mathbf{B}_k$.

5.2.2 Thresholded Average Pooling

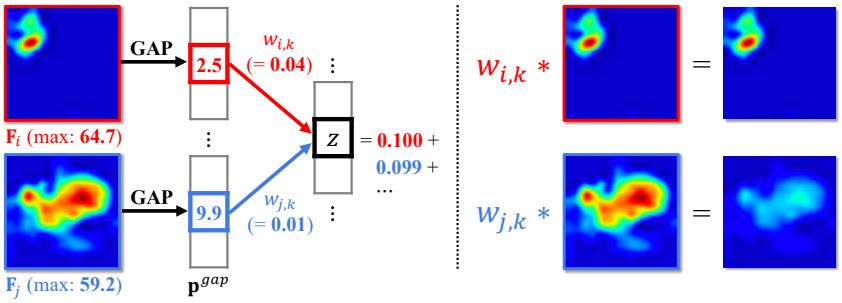
Problem. In WSOL, a GAP layer is employed to compute a weight of each channel to generate a CAM. However, the GAP layer tends to produce distorted weights for localization. More specifically, as in Eq.(5.1), the GAP layer sums all the activations and divides by $H \times W$ without considering the actual activated area per channel. The difference in the activated area per channel is, however, not negligible. As an example in Fig 5.2, suppose i-th feature in (a) captures the head of a *bird* whereas j-th feature captures its body. While the area activated in $\mathbf{F}_i$ is much smaller than that in $\mathbf{F}_j$, the GAP layer divides both of them by $H \times W$, and thus the pooled feature p_i^{gap} of $\mathbf{F}_i$ is also much smaller than p_j^{gap}. However, it does not mean the importance of $\mathbf{F}_i$ for classification is less than $\mathbf{F}_j$. For the ground truth class (k: *bird*), to compensate this difference, the FC weight $w_{i,k}$ corresponding to $\mathbf{F}_i$ is trained to be higher than $w_{j,k}$. As a result, when generating $\mathbf{M}_k$ in Eq.(5.2), small activated regions of $\mathbf{F}_i$ are highly overstated due to a large value of $w_{i,k}$, which causes localization to be limited to small region as localization depends on the maximum value of a CAM.

A batch normalization (BN) layer [53] can partly alleviate this issue through normalization. However, a BN layer may distort the activated area of a channel by forcing the distribution of activations in each channel to be similar. For example, when a channel captures a small region like ears of a *monkey*, the BN layer may expand its originally activated area, and as a result, localization can be overestimated if the feature is at the edge of the object. On the other hand, the following proposed solution alleviates the aforementioned problem without distorting the originally activated area.

Solution. To alleviate the problem of GAP, we propose the *thresholded average pooling* (TAP) layer. By replacing a GAP layer with a TAP layer, the pooled feature at c-th channel (Eq.(5.1)) is redefined as



(a) Input images and their two features (*head* and *body*)



(b) Classification phase (c) Localization phase

Figure 5.2: An example illustrating a problem of using the GAP layer. (a) $\mathbf{F}_i$ and $\mathbf{F}_j$ are the features capturing the head and body part of a *bird*, respectively. (b) When the features from the first image in (a) are passed to the GAP layer, although their max values are similar, the pooled features, p_i^{gap} and p_j^{gap}, are significantly different $(2.5, 9.9)$. Although the contributions of two features to the logit (z) are nearly the same $(0.1, 0.099)$, the FC weights, $w_{i,k}$ and $w_{j,k}$, are trained to be highly different to compensate the difference introduced by the GAP layer. (c) In localization phase, we can verify the weighted feature with small activation region, $w_{i,k} \cdot \mathbf{F}_i$, is highly overstated.

$$\mathbf{p}_c^{\text{tap}} = \frac{\sum_{(h,w)} \mathbb{1}(\mathbf{F}_c(h,w) > \tau_{tap}) \mathbf{F}_c(h,w)}{\sum_{(h,w)} \mathbb{1}(\mathbf{F}_c(h,w) > \tau_{tap})}, \tag{5.4}$$

where $\tau_{tap} = \theta_{tap} \cdot \max \mathbf{F}_c$ denotes a threshold value where $\theta_{tap} \in [0, 1)$ is a hyper-parameter. The TAP layer can be regarded as a generalized pooling layers in between global max pooling (GMP) and global average pooling (GAP). Although GAP is more effective for WSOL since the loss expands the activation to broader regions, GMP also has an important trait for WSOL that it can precisely focus on the important activa-

tions of each channel for pooling. The TAP layer inherits the benefits of both GMP and GAP; by including broad spatial areas, it can have the loss propagate to the feature map activations. Also, by excluding inactive or less active regions, the pooled channel value $\mathbf{p}_c^{tap}$ can better represent the core unique activations of the channel.

5.2.3 Negative Weight Clamping

Problem. When CNNs are trained for classification, a large number of the weights from the FC layer are negative. The features with negative weights help a model discriminate between different classes by decreasing the prediction probability of a target class. Existing CAM methods include the features with negative weights, and its underlying assumption is that they are mostly activated in *no-object* region like background. In contrast to this expectation, our analysis reveals many features with negative

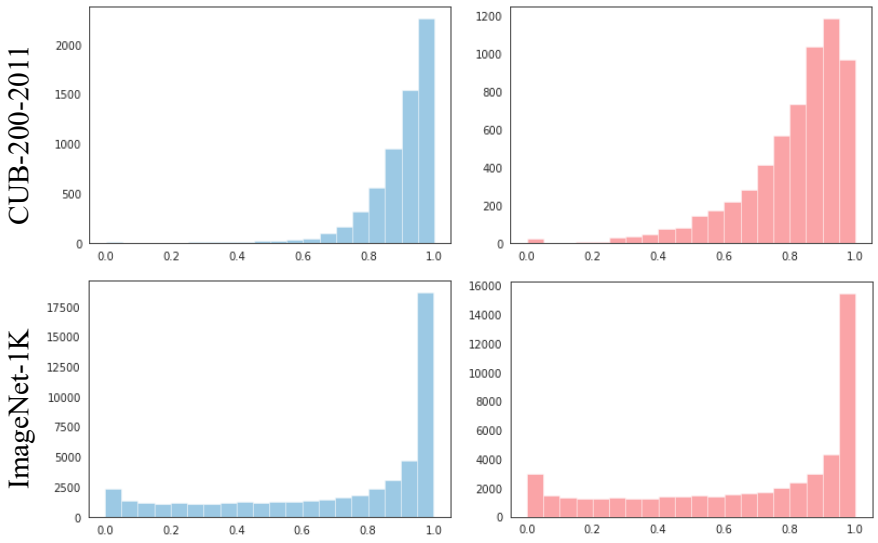


Figure 5.3: Intersection over Area (IoA) between the ground truth boxes and the CAMs generated from positive (a) and negative (b) weighted features. It indicates how much the features with the corresponding weights are activated in the object region. Surprisingly, a majority of the features with negative weights (b) are activated inside the objects regardless of dataset, which is even comparable to those with positive weights (a). This tendency is stronger in CUB-200-2011 than ImageNet-1K since the images of ImageNet-1K often contain multiple objects.

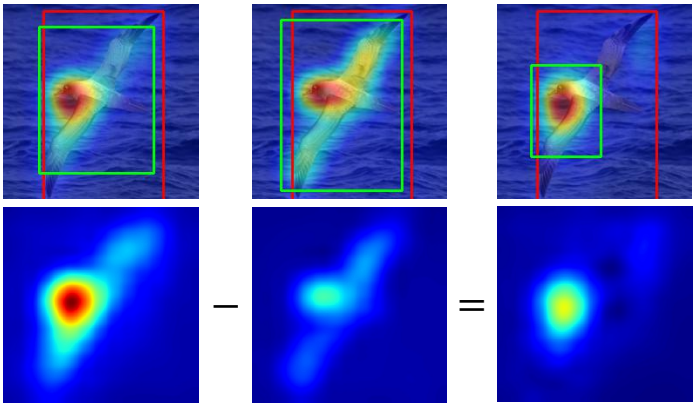


Figure 5.4: An example illustrating a problem of naively using negative weighted feature maps. We show the bounding boxes (green: predicted and red: GT) on the image (top) and CAMs (bottom) obtained using feature maps with (a) positive weights only, (b) negative weights only and (c) both . Negative weights depress the activations of less discriminative object areas like wings.

weights are concentrated within the object region as shown in Figure 5.3. Especially, their activations are high in the less discriminative regions compared to the features with positive weights.

We conjecture this phenomenon is closely related to the setting of WSOL: only one object is in an image. Suppose an image with a single object (*e.g*. *dog*). The features corresponding to negative weights mostly capture the characteristics of different classes (*i.e*. *cat*) inside the region of *dog*. It is because they are similar to *dog* class not the background where there is no object. In contrast to the single object cases, for an image with multiple objects (*e.g*. *dog* and *cat*), when the target class is *dog*, the *cat* region is likely to be highly activated by some features with negative weights since they help discriminate between *dog* and *cat*.

Figure 5.4 illustrates an example of this problem. The left, middle and right heat maps in the second row are respectively produced using (a) only positive weights, (b) negative weights and (c) both from $\mathbf{W}$. We take absolute values of the negative weights to produce the heat map in (b). We observe that the features corresponding to the negative weights are significantly activated in the region inside the *bird*. The

weighted features with negative weights largely abate the activations of the object region, and as a result, the localization is limited to the head part of the *bird*.

Solution. To mitigate this problem, we simply clamp negative weights to zero to generate a CAM. Hence, Eq.(5.2) is redefined as

$$\mathbf{M}_k = \sum_{c=1}^{C} \mathbb{1}(w_{c,k} > 0) \cdot w_{c,k} \cdot \mathbf{F}_c. \tag{5.5}$$

By doing this, we can secure the activations that are depreciated in the object regions. We will further discuss alternative approaches to handle negative weights in section 5.4.2.

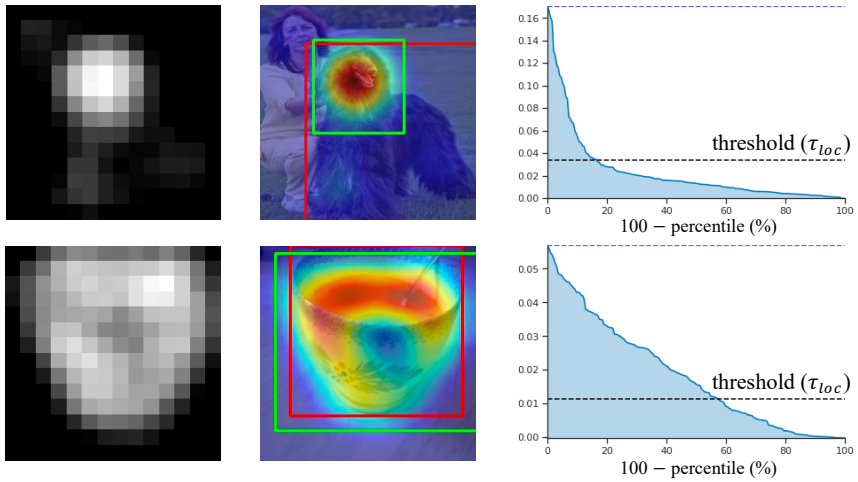
5.2.4 Percentile as a Thresholding Standard

Problem. Another issue of the CAM method is that many channels have high activations at small overlapping regions. Figure 5.5 compares two examples of problematic (top) and successful (bottom) localization. Figure 5.5(a) depicts the number of channels whose activations are greater than $\tau_{0.8} = 0.8\times$(the max of weighted features) at each position. In the top row of Figure 5.5(c), when the activation distribution follows Zipf's law, the maximum value (dotted line in blue) is not a robust metric as a thresholding standard for localization, since the localization threshold τ_{loc} (dotted line in black) captures only small region of the object when high activations overlap. Contrarily, high activations are distributed throughout the object in the bottom successful case.

Solution. Instead of using the maximum, the percentile may be one of the simplest but the most robust metrics that are not sensitive to outliers and exponential distributions of activations. Hence, the Eq. (5.3) for the localization threshold τ_{loc} is redefined as

$$\tau_{loc} = \theta_{loc} \cdot \mathrm{per}_i(\mathbf{M}'_k), \tag{5.6}$$

where per_i is an i-th percentile. Although any value in $[0, 1]$ is available for θ_{loc}, for percentile i, due to the small object cases where even 70-th percentile of a CAM is close to zero, we constraint the possible values for i to $[70, 100]$.



(a) Num of channels (activation $> \tau_{0.8}$)

(b) Result with CAM

(c) CAM values (descending order)

Figure 5.5: An example illustrating the problem of the overlapping high activation (top) compared to a successful case (bottom). In the problematic case (top), when high activations (activation $> \tau_{0.8}$) are concentrated in the small discriminative region, the localization threshold τ_{loc} in Eq.(5.3) becomes too high due to high maximum value of the CAM, which results in localization being limited to small region.

5.3 Experiment Settings

In this section, we describe the benchmark datasets and evaluation metrics we use to validate the effectiveness of our proposed methods (section 5.3.1). Furthermore, we present configuration details employed to conduct experiments (section 5.3.2).

5.3.1 Datasets

We evaluate our proposed approach on two standard benchmarks for WSOL: CUB-200-2011 [83] and ImageNet-1K [70]. CUB-200-2011 [83] is a dataset of 200 bird species. The numbers of images in training and test sets are 6,033 and 5,755, respectively. The bounding box annotations are provided for both training and test images. ImageNet-1K [70] is a dataset with 1,000 different categories; the numbers of images in training and validation sets are about 1.3 million and 50,000, respectively. The bounding box annotations are provided only for the validation images, on which we evaluate the performance of models.

Performance evaluation. We report the performance of models using three metrics: *Top-1 Cls*, *GT Loc* and *Top-1 Loc*. *Top-1 Cls* is the top-1 accuracy of classification, and *GT Loc* measures the localization accuracy with known ground truth class. For *Top-1 Loc*, the prediction is counted as correct if the predictions on both classification and localization (*i.e*. IoU ≥ 0.5) are correct. We use *Top-1 Loc* as a main metric since *Top-1 Loc* consider the performance of both classification and localization.

5.3.2 Configuration Details

To validate the robustness of our methods, we employ four different CNN backbones: VGG16 [74], ResNet50-SE [32, 37], MobileNetV1 [35] and GoogleNet [77]. For VGG16, we replace the last pooling layer and two following FC layers with a GAP layer as done in [97]. We add SE blocks [37] on top of ResNet50 to build ResNet50-SE following ADL [11]. For GoogleNet, we replace the last inception block with two CONV layers based on SPG [95]. For the threshold τ_{tap} of TAP layer defined in Eq.(5.4), we set $\theta_{tap} = 0.1$ for VGG16 and MobileNetV1 and $\theta_{tap} = 0.0$ for ResNet50-SE and GoogleNet through hyperparameter tuning on the validation set drawn from CUB-200-2011 training set. θ_{loc} and i-th percentile used to determine the localization threshold τ_{loc} are set to 0.35 and 90, respectively, by observing some qualitative results on training data of CUB-200-2011 as done in HaS [76]. Due to the small object cases as mentioned in section 5.2.4, we choose 90-th percentile as a standard to determine the threshold for localization .

5.4 Experiment Results

We first provide quantitative results with (i) different components of our proposed methods applied, (ii) varied backbone structures, and (iii) comparison of state-of-the-art models of WSOL (section 5.4.1). Then we further demonstrate the effectiveness of our proposed methods through ablation studies on different components (section 5.4.2). Finally, we present qualitative results that show the effectiveness and weakness of our proposed methods (section 5.4.3).

5.4.1 Quantitative Results

Our approach consistently improves the performance with various CNN backbones and CAM methods; especially we achieve the new state-of-the-art performance on both datasets.

Results with Different Components

We demonstrate the effectiveness of each proposed solution on CUB-200-2011 and ImageNet-1K. Table 5.1 shows the performance variations of the GAP-based CAM [97] with ResNet50-SE. In Table 5.1, three leftmost columns denote whether each of our solutions is applied to the *baseline*, which refers to the GAP-based CAM [97]. We verify the TAP layer improves the performance of both classification (CUB: $75.58 \rightarrow 76.79$, ImageNet: $71.62 \rightarrow 73.60$) and localization (CUB: $45.60 \rightarrow 46.84$, ImageNet: $46.73 \rightarrow 47.73$). The weight clamping method as well as 90-th percentile standard also constantly improve the performance of localization regardless of datasets (CUB: $45.60 \rightarrow 51.62, 53.40$, ImageNet: $46.73 \rightarrow 47.48, 47.49$). With all the components applied, the localization accuracies further improve on both datasets.

In addition to the quantitative results with different components on ResNet50-SE [32, 37] as provided in Table 5.1, we further provide the results on the other backbone structures: VGG16 [74], MobileNetV1 [35] and GoogleNet [77]. Table 5.2–5.4 provide the performance of VGG16, MobileNetV1 and GoogleNet with different combination of our proposed methods applied. Regardless of different backbones, the per-

Method	TAP	NWC	PaS	CUB-200-2011			ImageNet-1K		
				Top-1 Cls	GT Loc	Top-1 Loc	Top-1 Cls	GT Loc	Top-1 Loc
Baseline				75.58	60.17	45.60	71.62	61.83	46.73
+ Ours	✓			**76.79**	60.20	46.84	**73.60**	61.53	47.73
		✓		75.58	69.54	51.62	71.62	63.05	47.48
			✓	75.58	67.05	53.40	71.62	61.33	47.49
	✓	✓		**76.79**	69.76	53.18	**73.60**	60.68	47.29
	✓		✓	**76.79**	67.05	53.40	**73.60**	61.33	47.49
		✓	✓	75.58	72.01	54.26	71.62	**64.28**	**48.60**
	✓	✓	✓	**76.79**	**75.53**	**58.39**	**73.60**	62.02	47.99

Table 5.1: Performance variations of the GAP-based CAM [97] with ResNet50-SE according to different uses of our solutions. TAP, NWC and PaS refer to thresholded average pooling, negative weight clamping and percentile as a thresholding standard.

formance in *Top-1 Loc* significantly improves on both CUB-200-2011 and ImageNet-1K: VGG16 (CUB: $37.05 \rightarrow 61.30$, ImageNet: $41.62 \rightarrow 44.69$), MobileNetV1 (CUB: $44.46 \rightarrow 57.63$, ImageNet: $42.21 \rightarrow 46.44$) and GoogleNet (CUB: $46.86 \rightarrow 51.05$, ImageNet $46.98 \rightarrow 47.70$).

Method	TAP	NWC	PaS	CUB-200-2011			ImageNet-1K		
				Top-1 Cls	GT Loc	Top-1 Loc	Top-1 Cls	GT Loc	Top-1 Loc
Baseline				69.95	53.68	37.05	64.56	59.81	41.62
+ Ours	✓			**74.91**	64.10	48.53	**67.28**	61.25	44.58
		✓		69.95	64.30	44.15	64.56	60.46	42.05
			✓	69.95	65.90	48.45	64.56	61.96	43.52
	✓	✓		**74.91**	73.58	54.41	**67.28**	61.45	44.52
	✓		✓	**74.91**	72.87	56.64	**67.28**	61.21	44.60
		✓	✓	69.95	76.42	54.30	64.56	**62.53**	43.83
	✓	✓	✓	**74.91**	**80.72**	**61.30**	**67.28**	61.69	**44.69**

Table 5.2: Performance variations of the GAP-based CAM [97] with VGG16 according to different uses of our solutions.

Method	TAP	NWC	PaS	CUB-200-2011			ImageNet-1K		
				Top-1 Cls	GT Loc	Top-1 Loc	Top-1 Cls	GT Loc	Top-1 Loc
Baseline				72.09	58.92	44.46	66.52	58.85	42.21
+ Ours	✓			**75.82**	67.76	52.97	**68.09**	60.01	44.16
		✓		72.09	60.58	45.43	66.52	58.60	41.89
			✓	72.09	59.75	44.94	66.52	60.05	43.06
	✓	✓		**75.82**	**74.44**	**58.04**	**68.09**	59.08	43.36
	✓		✓	**75.82**	67.03	52.11	**68.09**	61.85	45.54
		✓	✓	72.09	62.89	46.95	66.52	60.50	43.24
	✓	✓	✓	**75.82**	74.28	57.63	**68.09**	**63.72**	**46.44**

Table 5.3: Performance variations of the GAP-based CAM [97] with MobileNetV1 according to different uses of our solutions.

Method	TAP	NWC	PaS	CUB-200-2011			ImageNet-1K		
				Top-1 Cls	GT Loc	Top-1 Loc	Top-1 Cls	GT Loc	Top-1 Loc
Baseline				74.35	61.67	46.86	70.50	62.32	46.98
+ Ours	✓			**75.04**	62.17	49.00	**71.09**	62.17	47.24
		✓		74.35	64.69	49.14	70.50	62.39	47.11
			✓	74.35	60.10	45.75	70.50	62.63	47.30
	✓	✓		**75.04**	**65.14**	50.66	**71.09**	62.04	47.12
	✓		✓	**75.04**	61.51	48.53	**71.09**	62.46	47.45
		✓	✓	74.35	64.48	48.62	70.50	**63.04**	47.57
	✓	✓	✓	**75.04**	65.10	**51.05**	**71.09**	62.76	**47.70**

Table 5.4: Performance variations of the GAP-based CAM [97] with GoogleNet according to different uses of our solutions.

From the experiment results with MobileNetV1 in Table 5.3, we can see that applying all of our proposed methods does not necessarily give the highest performance: the performance of TAP + NWC is slightly higher than TAP + NWC + PaS for *Top-1 Loc* by 0.41. It is because some of the aforementioned three problems are not as clear on some backbones as on the other backbones. For example, the overall activations of the features from MobileNetV1 are much smaller than those from the other backbones. Because of the small activations, the problem of the overlap of high activation is less severe on MobileNetV1 than the other backbones. Therefore, a combination of two solutions can be better than the combination of all the solutions due to the characteristics of different backbones.

Another thing to be noticed from the experiments is that the improvement of localization (*GT-known Loc* and *Top-1 Loc*) on ImageNet-1K is not as significant as on CUB-200-2011. We conjecture the reasons are two-fold and both are highly related to the characteristics of ImageNet-1K dataset where a number of images contain multiple objects. First, negative weight clamping is not as effective on ImageNet-1K as on CUB-200-2011 since many images in ImageNet-1K contain multiple objects. As stated in section 5.2.3, when multiple objects exist in an image, the features with negative weights tend to be activated in the background region with objects of different classes. Hence, negative weight clamping often makes a CAM to be activated on the larger region containing multiple objects rather than the object region of interest. We further demonstrate this phenomenon with examples in section 5.4.3. Moreover, along with the aforementioned characteristics of ImageNet-1K, due to the variety of classes and images of ImageNet-1K dataset compared to CUB-200-2011, the activations of feature maps are relatively more distributed. In other words, the discriminative regions of CUB-200-2011 are relatively smaller than those of ImageNet-1K since many images in CUB-200-2011 share the common features *i.e* feathers, wings. Since our proposed methods focus on expanding localization from the discriminative to less-discriminative region, for some cases, the application of the proposed methods results in localization to be too large.

Results with Different Backbones

To validate the robustness of our solutions, we evaluate models with different backbones. Table 5.5 summarizes the results on CUB-200-2011 and ImageNet-1K. Although the performance is sometimes slightly higher with two components applied, we employ all three components for **+ Ours**. Our approach improves the *baseline* with significant margins (CUB: 13.60, ImageNet: 2.32 on average). The results are compatible or even better than the state-of-the-art methods on both CUB-200-2011 and ImageNet-1K as shown in Table 5.6.

Backbone	Method	CUB-200-2011			ImageNet-1K		
		Top-1 Cls	GT Loc	Top-1 Loc	Top-1 Cls	GT Loc	Top-1 Loc
VGG16	Baseline	69.95	53.68	37.05	64.56	59.81	41.62
	+ Ours	**74.91**	**80.72**	**61.30**	**67.28**	**61.69**	**44.69**
ResNet50-SE	Baseline	75.58	60.17	45.60	71.62	61.83	46.73
	+ Ours	**76.79**	**75.53**	**58.39**	**73.60**	**62.02**	**47.99**
MobileNetV1	Baseline	72.09	58.92	44.46	66.52	58.85	42.21
	+ Ours	**75.82**	**74.28**	**57.63**	**68.09**	**63.72**	**46.44**
GoogleNet	Baseline	74.35	61.67	46.86	70.50	62.32	46.98
	+ Ours	**75.04**	**65.10**	**51.05**	**71.09**	**62.76**	**47.70**

Table 5.5: Performance of our proposed methods applied to the GAP-based CAM (Baseline) with various backbone structures.

Comparison with the State-of-the-Arts

Since our proposed methods are applicable to any WSOL algorithms using the GAP-based CAM, we validate their compatibility with another base model. First, we select ADL [11] since it is currently the best performing model for WSOL.

Table 5.6 compares our proposed methods on top of ADL with the state-of-the-art models: ACoL [94], HaS [76], SPG [95] and DANet [89]. We validate the proposed approaches further improve ADL on both CUB-200-2011 and ImageNet-1K, and significantly outperform all the state-of-the-arts on CUB-200-2011 and obtain the comparable results on ImageNet-1K. To the best of our knowledge, the performance of **Baseline + Ours** with VGG16 in Table 5.5 is the new state-of-the-art performance on CUB-200-2011. Also, **ADL + Ours** with GoogleNet achieves the new state-of-the-art results on ImageNet-1K.

Backbone	Method	CUB-200-2011			ImageNet-1K		
		Top-1 Cls	GT Loc	Top-1 Loc	Top-1 Cls	GT Loc	Top-1 Loc
VGG16	ACoL*	71.90	–	45.92	67.50	–	**45.83**
	SPG*	**75.50**	–	48.93	–	–	–
	DANet*	75.40	–	52.52	–	–	–
	ADL	69.05	73.96	53.40	65.66	60.62	43.04
	ADL + Ours	75.01	**76.30**	**58.96**	**68.67**	**60.73**	44.62
ResNet50-SE	ADL	**76.53**	71.99	57.40[1]	75.06	61.04	48.23
	ADL + Ours	75.03	**77.58**	**59.53**	**75.82**	**62.20**	**49.42**
MobileNetV1	HaS-32*	66.64	–	44.67	**67.48**	–	41.87
	ADL	71.90	62.55	47.69	67.02	59.21	42.89
	ADL + Ours	**73.51**	**78.60**	**59.41**	67.15	**61.69**	**44.78**
GoogleNet	Has-32*	–	–	–	70.70	60.29	45.21
	ACoL*	–	–	–	–	–	46.72
	SPG*	–	–	46.64	–	–	48.60
	DANet*	71.20	–	49.45	72.50	–	47.53
	ADL	73.37	66.81	51.29	74.38	60.84	47.72
	ADL + Ours	**73.65**	**69.95**	**53.04**	**74.25**	**64.44**	**50.56**

Table 5.6: Comparison of our proposed methods applied to ADL with other state-of-the-art algorithms. The methods with * indicate that the performances are directly referred from the original paper. – indicates no accuracy reported in the paper.

Dataset	Backbone	Method	Top-1 Cls	GT Loc	Top-1 Loc
CUB-200-2011	VGG16	ACoL	**71.37**	62.00	45.96
		+ Ours	71.18	**67.40**	**50.52**
		HaS-32	66.10	71.57	49.46
		+ Ours	**70.12**	**78.58**	**57.37**
	MobileNetV1	HaS-32	65.98	67.31	46.70
		+ Ours	**71.16**	**75.04**	**55.56**
	GoogleNet	HaS-32	**75.35**	61.08	47.36
		+ Ours	74.25	**67.03**	**50.64**
ImageNet-1K	VGG16	HaS-32	62.28	61.23	41.64
		+ Ours	**66.21**	**61.48**	**43.91**
	MobileNetV1	HaS-32	65.45	60.12	42.73
		+ Ours	**65.60**	**62.76**	**44.69**
	GoogleNet	HaS-32	**68.92**	60.55	44.64
		+ Ours	67.86	**62.36**	**45.36**

Table 5.7: The performance of ACoL and HaS-32 with and without our proposed methods applied on CUB-200-2011 and ImageNet-1K.

To demonstrate the robustness of our proposed methods, we further provide the performance of them on top of other state-of-the-art algorithms: ACoL [94] and HaS [76]. We provide the experiment results on some combinations of dataset, backbone and method which are reported in the previous papers as well as reproducible using pub-

[1]The reported number in ADL paper is 62.29 which is obtained by tweaking the number of classes from 200 to 1000. Since we think it is not very intuitive, the number of classes is left as 200.

licly available source code. Table 5.7 shows the performance of ACoL and HaS-32 with and without our proposed methods applied. From the experiment, we validate the proposed methods significantly improve *Top-1 Loc* of ACoL (CUB: $45.96 \rightarrow 50.52$). They also drastically improve *Top-1 Loc* of HaS-32 regardless of dataset (CUB: 6.68, ImageNet: 1.65 on average).

5.4.2 Ablation Experiments

We provide the results of ablation experiments on each component proposed which further demonstrate the effectiveness of our proposed methods.

The advantage of a TAP layer. Table 5.8 demonstrates the advantage of the TAP layer by varying the application of TAP and BN layer in ResNet50-SE and MobileNetV1. To remarkably remove the normalization effect of BN layer, we remove the BN layers in the last two blocks and the last block for ResNet50-SE and MobileNetV1, respectively, when it is specified BN layer is are not applied in Table 5.8. When BN layers are not applied, *Top-1 Loc* noticeably drops. But even without BN layers, by replacing a GAP layer with a TAP layer, the better performance can be achieved since the TAP layer explicitly alleviates the bias introduced by a GAP layer. Furthermore, although a model only with a TAP layer is worse than that only with BN layers in terms of classification, its *Top-1 Loc* which considers both classification and localization performance is higher, which implies a TAP layer is significantly more beneficial for localization compared to a BN layer.

TAP	BN	ResNet50-SE		MobileNetV1	
		Top-1 Cls	Top-1 Loc	Top-1 Cls	Top-1 Loc
		73.28	44.82	63.79	37.68
✓		73.45	45.74	70.26	50.93
	✓	75.58	45.60	72.09	44.46
✓	✓	**76.79**	**46.84**	**75.82**	**52.97**

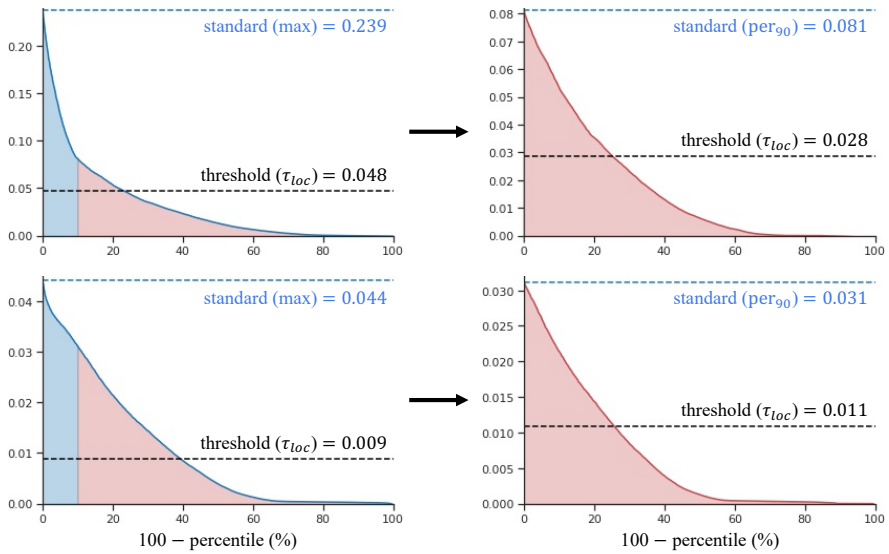
Table 5.8: The advantage of the TAP layer with and without the batch normalization (BN) layers in ResNet50-SE and MobileNetV1.

An alternative way to handle negative weights. Instead of clamping negative

weights to zero, one may suggest to map negative weights to any positive values using functions such as absolute value function. It is, however, counter-intuitive as the negative weights imply the corresponding features negatively contribute to predicting a target class. If the absolute value of a negative weight is larger than a positive weight, the channel with the negative weight can be more weighted than that with the positive weight. As a piece of empirical evidence, when using the absolute value, the performance of localization drops from $41.62 \rightarrow 41.59$ compared to 42.05 for negative weight clamping with VGG16 on ImageNet-1K.

Robustness of the percentile as a thresholding standard. Figure 5.6(a) show two different distributions of CAM values where the shades in red represent the values less than 90-th percentile. By zooming in the shades, we obtain the distributions in (b). The dotted lines in blue and black represent the maximum value and threshold value for localization of each distribution, respectively. Note that the threshold values of (a) and (b) are different since different θ_{loc}'s are applied. The distribution at the top of Figure 5.6(a) follows the Zipf's law whereas the values of the bottom one linearly decreases. When CAM values follow Zipf's law, due to the exponentially high maximum value, localization is often limited to small region. By replacing the maximum standard with the percentile standard, this problem can be largely alleviated. The shape of two distributions in (b) look very similar to each other. Therefore, by removing the pattern of Zipf's law using the percentile standard, we can robustly obtain a proper threshold value for localization.

Although the percentile standard is much more robust than the maximum standard, however, the object size needs to be considered to choose the right value of i in i-th percentile as stated in section 5.2.4. According to our analysis on the training images of CUB-200-2011, 11% and 28% of the ground truth boxes are smaller than 20% and 30% of the image size, respectively. It implies if i in per_i is too small, i-th percentile can be very close to zero. Thus, i should be adaptively chosen depending on dataset. In our case, since less than 0.5% of the total images contain the ground truth boxes where their sizes are less than 10% of the image size, we employ 90-th percentile as a



(a) CAM values (descending order)

(b) CAM values less 90-th percentile (descending order)

Figure 5.6: Distributions of CAM values sorted in descending order. (a) shows two different distributions of CAM values where the shades in red represent the values less than 90-th percentile. The distributions in (b) are obtained by zooming in the shades in (a). Although the distribution at the top in (a) follows Zipf's law and the bottom does not, both distributions in (b) linearly decrease and their shapes are very similar to each other. Thus, by employing the percentile standard as a replacement of the maximum standard, a proper threshold for localization can be robustly obtained.

standard for localization threshold.

5.4.3 Qualitative Results

As qualitative results, we provide CAM results on different proposed methods and backbone structures employed. We also show some near miss cases to illustrate under which condition, our proposed methods may not work.

Results by Different Proposed Method

In this section, we provide some qualitative results by different proposed method. The following results empirically demonstrate the problems raised in section 5.2.2–5.2.4 and effectiveness of our proposed solutions.

Thresholded average pooling. As stated in the section 5.2.2, a TAP layer de-

creases the bias introduced by different size of the activated area per channel. Thus, the corresponding weight of each channel to be properly trained, and as a result, it prevents small discriminative region from being overstated. Figure 5.7 demonstrates the effectiveness of a TAP layer compared to a GAP layer. Given an image (1st column), a model with a GAP layer generates CAMs in 2nd column whereas a model with a TAP layer generates CAMs in 3rd column, which produce the bounding boxes as shown in 4th and 5th columns, respectively. We can clearly see that when applying a TAP layer, the activations of a CAM are distributed throughout the object region, which is often not the case for a GAP layer.

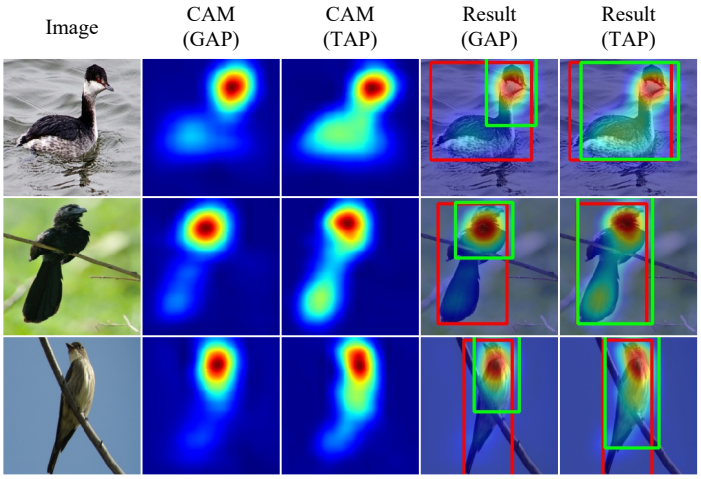


Figure 5.7: Qualitative results comparing between GAP and TAP layer. The boxes in red and green represent the ground truths and predictions of localization, respectively.

Negative weight clamping. Figure 5.8 demonstrates the effect of negative weight clamping. Given an image (1st column), we provide localization results overlaid with a CAM generated using positive (5th column), negative (6th column) and both (7th column) weights of $\mathbf{W}$. Also, 2–4th columns show only the CAMs corresponding to 5th, 6th and 7th columns, respectively. The figure evidently illustrates the problem of including the features corresponding to the negative weights as stated in section 5.2.3. The CAM generated only using the features with negative weights largely abate the activations of the object region. Using negative weight clamping, we prevent it from

abating the activations in less-discriminative region inside the object.

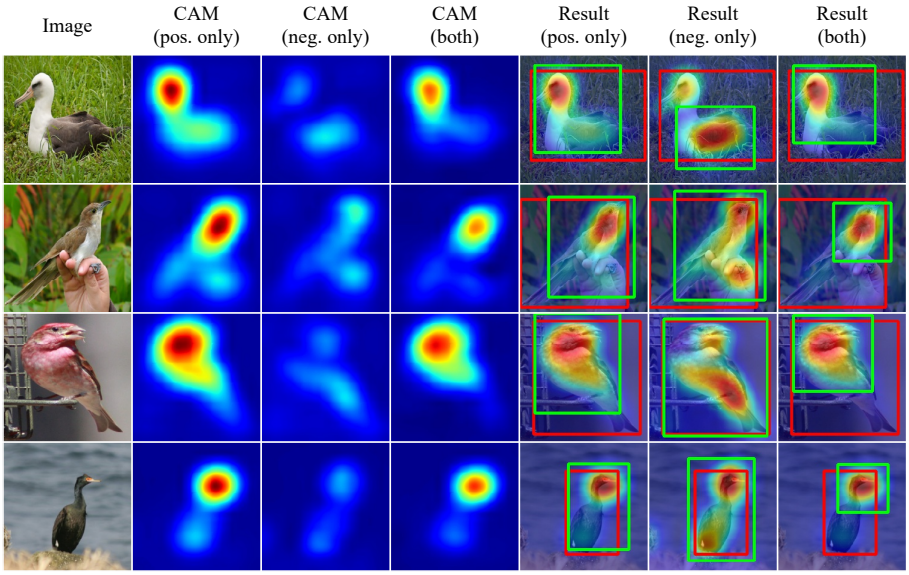


Figure 5.8: Qualitative results comparing between the CAM results with and without negative weight clamping applied. Positive only (2nd and 5th columns) and both (4th and 7th columns) correspond to the CAM and localization results with and without negative weight clamping applied, respectively. The boxes in red and green represent the ground truths and predictions of localization, respectively.

Percentile as a thresholding standard Lastly, Figure 5.9 illustrates the robustness of percentile (PaS) compared to the maximum as standard (MaS) for localization threshold. Note that replacing a standard to percentile does not change the activations of a CAM. Although there are many cases where the maximum standard properly estimates bounding boxes as shown in the first half of the columns, it often extracts too small bounding boxes as provided in the second half of the columns. On the other hand, the percentile standard more robustly estimates the location of an object. As shown in the figure, the variance of size of the bounding boxes extracted using the maximum standard is much higher than those extracted using the percentile standard depending on the distribution of the activations.

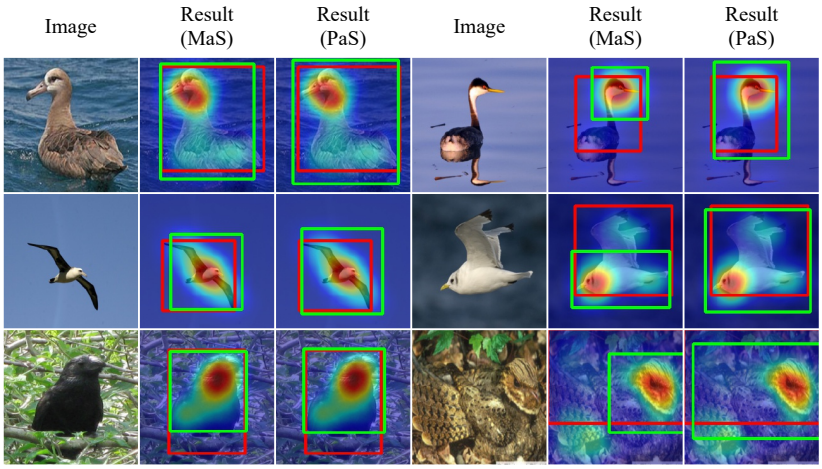


Figure 5.9: Qualitative results comparing between the maximum and percentile as a standard for localization threshold. The first half of columns show the cases where both standard properly estimate the bounding boxes whereas the second half of columns show the cases where only percentile properly estimates the bounding boxes. The boxes in red and green represent the ground truths and predictions of localization, respectively.

Results on Different Backbones

We present the qualitative results for our proposed methods with VGG16 [74], ResNet50-SE [32, 37], MobileNetV1 [35] and GoogleNet [77] compared to the baselines: CAM and ADL, where CAM refers to the GAP-based CAM. In Figure 5.10–5.11, we verify that in general, our proposed methods help a model to utilize more activations in object region, which results in the expansion of the bounding boxes compared to the ones from CAM and ADL on both CUB-200-2011 and ImageNet-1K. One thing to note from the first three examples of MobileNetV1 is that the proposed methods do not always expand the activations of a CAM. They rather adjust the activations to fit in the object regions by resolving the aforementioned three problems of the existing CAM method.

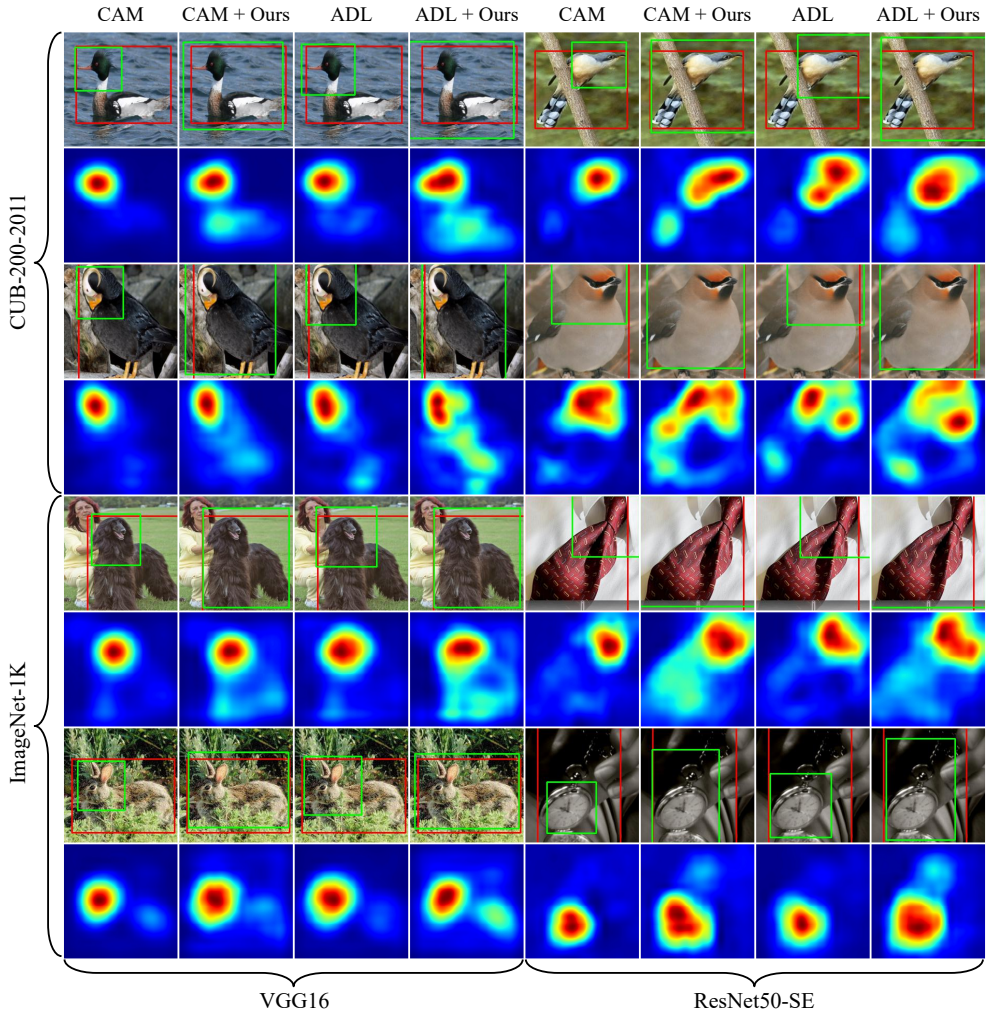


Figure 5.10: Qualitative results with VGG16 and ResNet50-SE on CUB-200-2011 and ImageNet-1K datasets. The boxes in red and green represent the ground truths and predictions of localization, respectively.

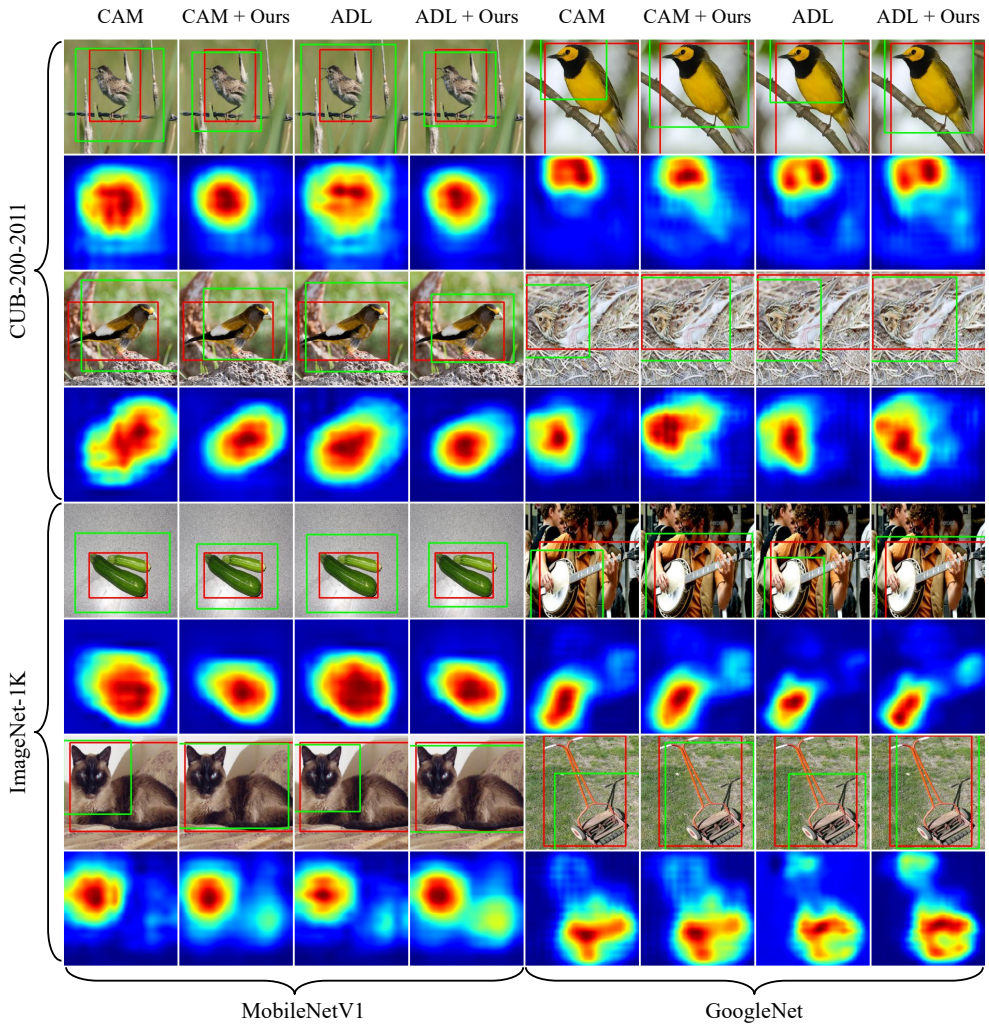


Figure 5.11: Qualitative results with MobileNetV1 and GoogleNet on CUB-200-2011 and ImageNet-1K datasets. The boxes in red and green represent the ground truths and predictions of localization, respectively.

Near Miss Cases

As stated in section 5.4.1, in this section, we provide some examples where the features with negative weights are activated in the background region with multiple objects of different classes when there are multiple objects in the given image. The images in Figure 5.12 which contain multiple objects are all from ImageNet-1K dataset. Given the original images (1st column), we provide localization results (5th–

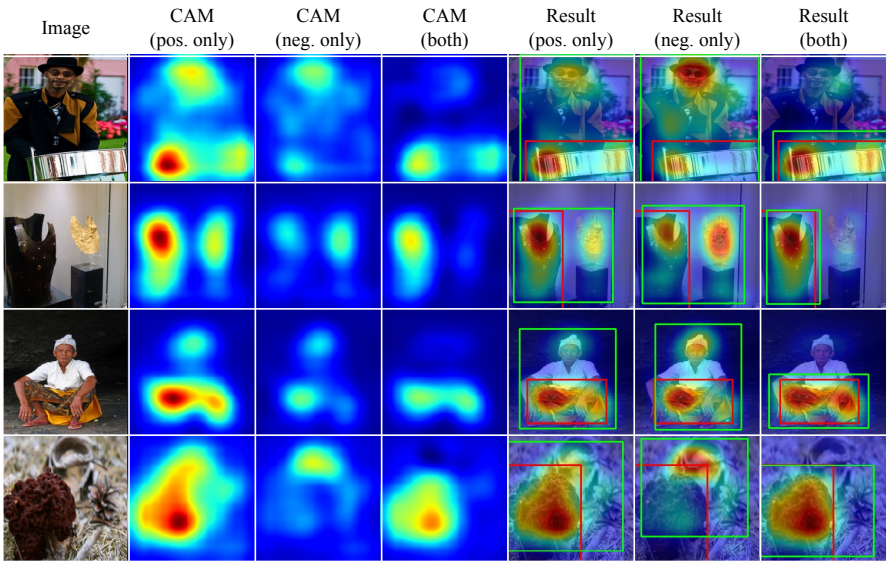


Figure 5.12: Qualitative results illustrating the activations of the features with negative weights for the multiple object cases. 3rd and 6th columns demonstrate when there are multiple objects in an image, the features corresponding to the negative weights tend to be activated in the object which is not a target class for classification. The boxes in red and green represent the ground truths and predictions of localization, respectively.

7th columns) and only CAMs (2nd–4th columns). The CAMs are generated using the FC weights of either positive only, negative only or both as specified at the top of Figure 5.12. The images in the 3rd and 6th columns demonstrate when there are multiple objects in the image, the features corresponding to the negative weights tend to be more activated in the object which is not a target class for classification. As a result, after negative weight clamping, the final CAM captures broader regions than the regions of the target object.

5.5 Conclusion

The GAP-based CAM, the most widely used CAM method for WSOL, have three major flaws which cause the localization to be limited to the small discriminative region. Instead of endeavoring to obtain additional information as done in the most of the previous studies on WSOL, in this paper, we proposed three simple but robust methods to properly and more efficiently utilize the information obtained from a clas-

sification model. We validated our proposed approach largely mitigate the problems, and as a result, achieved the new state-of-the-art performance on both CUB-200-2011 and ImageNet-1K. As a result, it improves the performance of the GAP-based CAM method to be compatible with the cutting-edge algorithms and achieves the state-of-the-art performance applied to one of the cutting-edge algorithms.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we have studied several object detection problems and related issues. We summarize the main contributions of this thesis in the following.

In Chapter 3, we addressed the two critical issues of pedestrian detection: occlusion and confusion with hard negative examples. Our approach is general and flexible enough to be applicable to any single-stage detectors. We implemented our occlusion and hard negative handling methods into four state-of-the-art single-stage models. The experiment results on various pedestrian detection benchmarks demonstrated that our approach indeed improved the performance of single-stage detectors for pedestrian detection.

In Chapter 4, we proposed a novel feature-level super-resolution approach to improve small object detection for the proposal-based detection framework. Our method is applicable on top of any proposal-based detectors with feature pooling. The experiments on various object detection benchmarks validated our super-resolution approach was indeed effective to detect small objects. In particular, our work proved that it is important to provide direct supervision using proper high-resolution target features that

share the same relative receptive field with the low-resolution input features.

In Chapter 5, we proposed surpassing localization methods for weakly supervised learning. The GAP-based CAM, the most widely used CAM method for WSOL, have three major flaws which cause the localization to be limited to the small discriminative region. Instead of endeavoring to obtain additional information as done in the most of the previous studies on WSOL, in this thesis, we proposed three simple but robust methods to properly and more efficiently utilize the information obtained from a classification model. We validated the proposed approach largely mitigate the problems, and as a result, achieved the new state-of-the-art performance on multiple benchmarks for WSOL.

6.2 Future Works

As future works, we may consider improving our proposed methods in this thesis in the following ways:

- **Chapter 3. Part and Grid Classification Based Post-Refinement for Occluded Objects and Hard Negatives.** For the method of handling occluded objects, it has a limitation that the annotations on visible parts must be additionally given. The simplest way to deal with occlusion without such information is to consider deleting a portion of the groundtruth object arbitrarily. However, the shape is very different from the real image that the covering area is simply replaced with 0 (or any specific value). As a way of solving this problem, we can think of a generation model that can create natural obstacles when given an area to be covered or even select an area to be hidden. Moreover, we may expand the target objects from pedestrians to general objects. For our proposed method, we fixed the grid of a part confidence map since the aspect ratio of pedestrians does not vary much. On the other hand, for multi-class setting, the aspect ratios of different classes substantially vary. It is, however, extremely inefficient to find the optimal grid size for every class using hyperparameter tuning. One potential

solution for this would be using AutoML [80] which helps to find the optimal hyperparameters.

- **Chapter 4. Self-Supervised Feature Super-Resolution for Small Objects.** For feature super-resolution, we fixed the downsampling ratio to 2. It is, however, intuitively not optimal because small objects may appear in different ratios in the test set. Hence, we may consider to adaptively select the ratio depending on the characteristics of RoIs. Furthermore, although the computation associated with the SR feature generator is the only overhead a model had for inference time, due to the large number of parameters for residual blocks, the size of overhead was fairly large. To alleviate this problem, we may be able to employ a more efficient super-resolution generator model.

- **Chapter 5. Rectified Class Activation Mapping for Weakly Labeled Objects.** To handle weakly labeled objects, based on the phenomenon that the activation areas corresponding to the negative weight occur mainly in the object, instead of removing the corresponding areas from the localization map, we used a negative weight clamping method which simply ignores them. However, if there is a feature that catches the background, or if an image with multiple objects appears, the method will behave incorrectly. If we figure out what the meaning of the weights is, in other words, how they relate to an object a model is trying to localize, we may use this information to better localize an object. The current CAM-based method, however, determines everything using only weights for a given class, in which case there is a limit to knowing the relationship. By using other externally information such as the weights of other classes, it is expected to be able to find the relationship between the weights and an object, which we left as a future work.

All the methods proposed in this thesis are based on the most widely used models for each task. As we stated, they are all applicable to any other models which use the same mechanism as our base models but can be further applied to the broader

model categories with little modification. For instance, our proposed methods to handle occluded objects and hard negatives are originally applied to a single-stage detector but can be modified to be applied to two-stage detectors. Similarly, the method for small object detection can be further improved to be applied to sing-stage detectors. Eventually, it would be ideal if we can propose a robust and integrated method that can resolve all the problems we stated in this thesis using one single model.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[2] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem. SOD-MTGAN: Small Object Detection via Multi-Task Generative Adversarial Network. In *ECCV*, 2018.

[3] L. Bazzani, A. Bergamo, D. Anguelov, and L. Torresani. Self-Taught Object Localization With Deep Networks. In *WACV*, 2016.

[4] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick. Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks. In *CVPR*, 2016.

[5] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection. In *ECCV*, 2016.

[6] Z. Cai and N. Vasconcelos. Cascade R-CNN: Delving into High Quality Object Detection. In *CVPR*, 2018.

[7] Z. Cai and N. Vasconcelos. Cascade R-CNN: High Quality Object Detection and Instance Segmentation. *IEEE TPAMI*, 2019.

[8] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian. Grad-Cam++: Improved Visual Explanations for Deep Convolutional Networks. In *WACV*, 2018.

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *arXiv:1412.7062*, 2014.

[10] J. Choe, J. H. Park, and H. Shim. Improved Techniques for Weakly-Supervised Object Localization. *Arxiv:1802.07888*, 2018.

[11] J. Choe and H. Shim. Attention-Based Dropout Layer for Weakly Supervised Object Localization. In *CVPR*, 2019.

[12] M. J. Choi, A. Torralba, and A. S. Willsky. A Tree-based Context Model for Object Recognition. *IEEE TPAMI*, 34(2):240–252, 2011.

[13] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *CVPR*, 2016.

[14] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In *NIPS*, 2016.

[15] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. *CVPR*, pages 886–893, 2005.

[16] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral Channel Features. In *BMVC*, 2009.

[17] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian Detection: An Evaluation of the State of the Art. *IEEE TPAMI*, 34(4):743–761, 2012.

[18] X. Du, M. El-Khamy, J. Lee, and L. Davis. Fused DNN: A Deep Neural Network Fusion Approach to Fast and Robust Pedestrian Detection. In *WACV*, 2017.

[19] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. CenterNet: Keypoint Triplets for Object Detection. In *ICCV*, 2019.

[20] M. Enzweiler, A. Eigenstetter, B. Schiele, and D. M. Gavrila. Multi-cue Pedestrian Classification with Partial Occlusion Handling. In *CVPR*, 2010.

[21] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010.

[22] Y. Fang, K. Kuan, J. Lin, C. Tan, and V. Chandrasekhar. Object Detection Meets Knowledge Graphs. In *IJCAI*, 2017.

[23] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-based Models. *IEEE TPAMI*, 32(9):1627–1645, 2010.

[24] C. Fookes, F. Lin, V. Chandran, and S. Sridharan. Evaluation of Image Resolution and Super-Resolution on Face Recognition Performance. *Journal of Visual Communication and Image Representation*, 23(1):75–93, 2012.

[25] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD: Deconvolutional Single Shot Detector. *arXiv:1701.06659*, 2017.

[26] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[27] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *CVPR*, 2014.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *NIPS*, 2014.

[29] R. Hamaguchi, A. Fujita, K. Nemoto, T. Imaizumi, and S. Hikosaka. Effective Use of Dilated Convolutions for Segmenting Small Object Instances in Remote Sensing Imagery. In *WACV*, 2018.

[30] M. Haris, G. Shakhnarovich, and N. Ukita. Task-Driven Super Resolution: Object Detection in Low-resolution Images. *arXiv:1803.11316*, 2018.

[31] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017.

[32] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[33] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *science*, 313(5786):504–507, 2006.

[34] Q. Hou, P. Jiang, Y. Wei, and M.-M. Cheng. Self-Erasing Network for Integral Object Attention. In *NeurIPS*, 2018.

[35] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861*, 2017.

[36] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation Networks for Object Detection. In *CVPR*, 2018.

[37] J. Hu, L. Shen, and G. Sun. Squeeze-and-Excitation Networks. In *CVPR*, 2018.

[38] P. Hu and D. Ramanan. Finding Tiny Faces. In *CVPR*, 2017.

[39] Q. Hu, P. Wang, C. Shen, A. van den Hengel, and F. Porikli. Pushing the Limits of Deep CNNs for Pedestrian Detection. *IEEE TCSVT*, 2017.

[40] X. Hu, X. Xu, Y. Xiao, H. Chen, S. He, J. Qin, and P.-A. Heng. SINet: A Scale-insensitive Convolutional Neural Network for Fast Vehicle Detection. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[41] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/Accuracy Trade-offs for Modern Convolutional Object Detectors. In *CVPR*, 2017.

[42] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. Acquisition of Localization Confidence for Accurate Object Detection. In *ECCV*, 2018.

[43] D. Kim, D. Cho, D. Yoo, and I. So Kweon. Two-Phase Learning for Weakly Supervised Object Localization. In *ICCV*, 2017.

[44] T. Kong, F. Sun, H. Liu, Y. Jiang, and J. Shi. FoveaBox: Beyond Anchor-based Object Detector. *arXiv:1904.03797*, 2019.

[45] H. Law and J. Deng. CornerNet: Detecting Objects as Paired Keypoints. In *ECCV*, 2018.

[46] H. Law, Y. Teng, O. Russakovsky, and J. Deng. CornerNet-Lite: Efficient Keypoint Based Object Detection. *arXiv:1904.08900*, 2019.

[47] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *nature*, 521(7553):436–444, 2015.

[48] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan. Perceptual Generative Adversarial Networks for Small Object Detection. In *CVPR*, 2017.

[49] K. Li, Z. Wu, K.-C. Peng, J. Ernst, and Y. Fu. Tell Me Where to Look: Guided Attention Inference Network. In *CVPR*, 2018.

[50] Z. Liang, J. Shao, D. Zhang, and L. Gao. Small Object Detection Using Deep Feature Pyramid Networks. In *Pacific Rim Conference on Multimedia*, 2018.

[51] M. Lin, Q. Chen, and S. Yan. Network in Network. In *ICLR*, 2014.

[52] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, 2017.

[53] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 2015.

[54] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. In *ICCV*, 2017.

[55] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014.

[56] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. In *ECCV*, 2016.

[57] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[58] M. Mathias, R. Benenson, R. Timofte, and L. Van Gool. Handling Occlusions with Franken-Classifiers. In *ICCV*, 2013.

[59] Z. Meng, X. Fan, X. Chen, M. Chen, and Y. Tong. Detecting Small Signs from Large Images. In *IEEE International Conference on Information Reuse and Integration (IRI)*, 2017.

[60] J. Noh, W. Bae, and G. Kim. Rethinking Class Activation Maps for Weakly Supervised Object Localization. *preprint*, 2020.

[61] J. Noh, W. Bae, W. Lee, J. Seo, and G. Kim. Better to Follow, Follow to Be Better: Towards Precise Supervision of Feature Super-Resolution for Small Object Detection. In *ICCV*, 2019.

[62] J. Noh, S. Lee, B. Kim, and G. Kim. Improving Occlusion and Hard Negative Handling for Single-Stage Pedestrian Detectors. In *CVPR*, 2018.

[63] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is Object Localization for Free? – Weakly-Supervised Learning With Convolutional Neural Networks. In *CVPR*, 2015.

[64] W. Ouyang and X. Wang. A Discriminative Deep Model for Pedestrian Detection with Occlusion Handling. In *CVPR*, 2012.

[65] W. Ouyang and X. Wang. Single-pedestrian Detection Aided by Multi-pedestrian Detection. In *CVPR*, 2013.

[66] P. O. Pinheiro and R. Collobert. From Image-Level to Pixel-Level Labeling With Convolutional Networks. In *CVPR*, 2015.

[67] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *CVPR*, 2016.

[68] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *CVPR*, 2017.

[69] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*, 2015.

[70] O. Russakovsky, J. Deng, H. SU, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

[71] M. S. Sajjadi, B. Schölkopf, and M. Hirsch. EnhanceNet: Single Image Super-Resolution through Automated Texture Synthesis. In *ICCV*, 2017.

[72] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-Cam: Visual Explanations From Deep Networks via Gradient-Based Localization. In *ICCV*, 2017.

[73] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond Skip Connections: Top-Down Modulation for Object Detection. *arXiv:1612.06851*, 2016.

[74] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556*, 2014.

[75] B. Singh and L. S. Davis. An Analysis of Scale Invariance in Object Detection - SNIP. In *CVPR*, 2018.

[76] K. K. Singh and Y. J. Lee. Hide-And-Seek: Forcing a Network to Be Meticulous for Weakly-Supervised Object and Action Localization. In *ICCV*, 2017.

[77] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *CVPR*, 2015.

[78] W. Tan, B. Yan, and B. Bare. Feature Super-Resolution: Make Machine See More Clearly. In *CVPR*, 2018.

[79] S. Tang, M. Andriluka, and B. Schiele. Detection and Tracking of Occluded People. *IJCV*, 110(1):58–69, 2014.

[80] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *SIGKDD*, 2013.

[81] Y. Tian, P. Luo, X. Wang, and X. Tang. Deep Learning Strong Parts for Pedestrian Detection. In *ICCV*, 2015.

[82] Z. Tian, C. Shen, H. Chen, and T. He. FCOS: Fully Convolutional One-Stage Object Detection. In *ICCV*, 2019.

[83] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report Cns-Tr-2011-001, California Institute of Technology, 2011.

[84] Y. Wei, J. Feng, X. Liang, M.-M. Cheng, Y. Zhao, and S. Yan. Object Region Mining With Adversarial Erasing: A Simple Classification to Semantic Segmentation Approach. In *CVPR*, 2017.

[85] Y. Wei, Z. Shen, B. Cheng, H. Shi, J. Xiong, J. Feng, and T. Huang. TS2C: Tight Box Mining with Surrounding Segmentation Context for Weakly Supervised Object Detection. In *ECCV*, 2018.

[86] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. *arXiv:1612.01051*, 2016.

[87] Y. Wu et al. Tensorpack. `https://github.com/tensorpack/`, 2016.

[88] W. Xiang, D.-Q. Zhang, H. Yu, and V. Athitsos. Context-aware Single-Shot Detector. In *WACV*, 2018.

[89] H. Xue, F. Wan, J. Jiao, X. Ji, and Y. Qixiang. DANet:Divergent Activation for Weakly supervised Object Localization. In *ICCV*, 2019.

[90] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. *arXiv:1511.07122*, 2015.

[91] L. Zhang, L. Lin, X. Liang, and K. He. Is Faster R-CNN Doing Well for Pedestrian Detection? In *ECCV*, 2016.

[92] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele. How Far Are We from Solving Pedestrian Detection? In *CVPR*, 2016.

[93] S. Zhang, R. Benenson, and B. Schiele. CityPersons: A Diverse Dataset for Pedestrian Detection. In *CVPR*, 2017.

[94] X. Zhang, Y. Wei, J. Feng, Y. Yang, and T. S. Huang. Adversarial Complementary Learning for Weakly Supervised Object Localization. In *CVPR*, 2018.

[95] X. Zhang, Y. Wei, G. Kang, Y. Yang, and T. Huang. Self-Produced Guidance for Weakly-Supervised Object Localization. In *ECCV*, 2018.

[96] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network. In *CVPR*, 2017.

[97] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. In *CVPR*, 2016.

[98] C. Zhu, Y. He, and M. Savvides. eature Selective Anchor-Free Module for Single-Shot Object Detection. In *ICCV*, 2019.

[99] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu. Traffic-Sign Detection and Classification in the Wild. In *CVPR*, 2016.

요약

물체 검출은 인스턴스 분할(instance segmentation), 물체 추적(object tracking), 이미지 캡션(image captioning), 장면 이해(scene understanding), 행동 인식(action recognition) 등 고차원의 컴퓨터 비전 태스크(task)뿐만 아니라, 비디오 감시(video surveillance), 자율주행차(self-driving car), 로봇 비전(robot vision), 증강현실(augmented reality) 등 실제 어플리케이션(application)에도 다양하게 적용되는 중요한 분야이다. 이러한 중요성에 의해 본 분야는 수십 년이라는 오랜 기간 동안 연구되어 왔으며, 특히 딥러닝의 등장과 함께 크게 발전하였다. 하지만 이런 발전에도 불구하고, 물체 검출을 하는 데 있어 어려움을 겪게 되는 조건들이 여전히 존재한다. 본 논문에서는 일반적으로 잘 알려진 세 가지 어려운 조건들에 대해, 기존의 대표적인 검출 모형들이 더 잘 대응할 수 있도록 개선하는 것을 목표로 한다.

먼저, 보행자 검출 문제에서 발생하는 폐색(occlusion)과 어려운 비물체(hard negative)에 대한 문제를 다룬다. 가려진 보행자의 경우에는 배경으로, 수직 물체와 같은 비물체는 반대로 보행자로 인식되는 경우가 많아 전체적인 성능 저하의 큰 원인이 된다. 보행자 검출 문제는 실시간 처리를 필요로 하는 경우가 많기 때문에, 본 논문에서는 속도 측면에서 장점이 있는 일단계(single-stage) 검출 모형을 개선하여 두 가지 문제를 완화할 수 있는 방법론을 제안한다. 제안된 방법론은 사람의 부위(part) 및 이미지의 격자(grid)에 대한 신뢰도 높은 분류 결과를 바탕으로, 기존 예측 결과를 보정하는 후처리 방식으로 이루어진다.

다음으로는, 일반적인 물체 검출 문제에서 발생하는 작은 물체에 대한 문제를 다룬다. 특히, 정확도 측면에서 장점이 있는 이단계(two-stage) 검출 모형에서조차 작은 물체 검출에 대한 정확도는 크게 떨어지는 편이다. 그 이유는 작은 영역에 해당하는 피쳐(feature)의 정보가 매우 부족하기 때문이다. 본 논문에서는 이단계 검출 모형을 기반으로, 피쳐 수준 초해상도(super-resolution) 방법론을 도입하여 이 문제를 해결하는 것을 소개한다. 특히, 초해상도를 위한 목표(target) 피쳐를 설정해줌으로써 학습을 안정화하고, 성능을 더욱 향상시킨다.

마지막으로, 학습환경에서 분류 레이블(classification label)만이 주어지는 약지도(weakly supervised) 학습환경에서 발생하는 문제를 다룬다. 약지도 물체 검출(weakly supervised object localization)은 일반적인 물체 검출에 대해, 이미지별 하나의 물체가 주어지고, 그 물체의 클래스(class) 정보만 학습에 활용 가능하다는 제약이 추가된 문제이다. 이에 대한 대표적인 검출 방법론은 피쳐의 활성(activation) 값들을 활용하는 CAM(class activation mapping)을 들 수 있다. 하지만 해당 방법론을 사용하는 경우, 예측 영역이 실제 물체의 영역에 비해 굉장히 좁게 잡히는 문제가 있는데, 이는 분류에 필요한 정보를 가진 피쳐들의 영역이 좁고, 검출 과정에서 이들의 비중을 높게 처리하기 때문에 발생한다. 본 논문에서는 이를 개선하여, 다양한 피쳐들의 정보를 검출에 최적화된 방식으로 활용하여 물체의 영역을 정확히 예측할 수 있는 방법론을 제안한다.

제안한 방법론들은 각 문제의 대표적인 벤치마크(benchmark) 데이터셋들에 대해 기존의 검출 모형들의 성능을 크게 향상시켰으며, 일부 환경에서는 최고 수준(state-of-the-art)의 성능을 달성하였다. 또한 다양한 모형들에 적용 가능한 유연성(flexibility)을 바탕으로, 추후 발전된 모형들에도 적용하여 추가적인 성능향상을 가져올 수 있을 것으로 기대된다.

주요어: 물체 검출, 컴퓨터 비전, 딥러닝, 약지도 물체 위치 추정, 보행자 검출
학번: 2015-30268