Improving Occlusion and Hard Negative Handling for Single-Stage Pedestrian Detectors

Junhyug Noh Soochan Lee Beomsu Kim Gunhee Kim
Department of Computer Science and Engineering
Seoul National University, Seoul, Korea
{jh.noh, soochan.lee}@vision.snu.ac.kr, {123bskim, gunhee}@snu.ac.kr
http://vision.snu.ac.kr/projects/partgridnet

Abstract

*We propose methods of addressing two critical issues of pedestrian detection: (i) occlusion of target objects as false negative failure, and (ii) confusion with hard negative examples like vertical structures as false positive failure. Our solutions to these two problems are general and flexible enough to be applicable to any single-stage detection models. We implement our methods into four state-of-the-art single-stage models, including SqueezeDet+ [22], YOLOv2 [17], SSD [12], and DSSD [8]. We empirically validate that our approach indeed improves the performance of those four models on Caltech pedestrian [4] and CityPersons dataset [25]. Moreover, in some heavy occlusion settings, our approach achieves the best reported performance. Specifically, our two solutions are as follows. For better occlusion handling, we update the output tensors of single-stage models so that they include the prediction of part confidence scores, from which we compute a final occlusion-aware detection score. For reducing confusion with hard negative examples, we introduce average grid classifiers as post-refinement classifiers, trainable in an end-to-end fashion with little memory and time overhead (*e.g. *increase of 1–5 MB in memory and 1–2 ms in inference time).*

1. Introduction

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 approaches [10, 9, 18] have greatly improved the performance for a variety of object detection problems and are currently one of the best performing detection paradigms. These approaches consist of the two stages of proposing regions and computing their confidences of object presence. YOLO [16] has brought another breakthrough in object detection; 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 approach are emerging, including SqueezeDet+ [22], YOLOv2 [17], SSD [12], and DSSD [8]. They all use a convolutional predictor to generate the final output tensor, and use anchors like the region proposal network (RPN) to predict the offsets of boxes rather than coordinates. In addition, DSSD [8] generates a context feature map using the deconvolutional layer, which enables global information to be used to detect smaller objects.

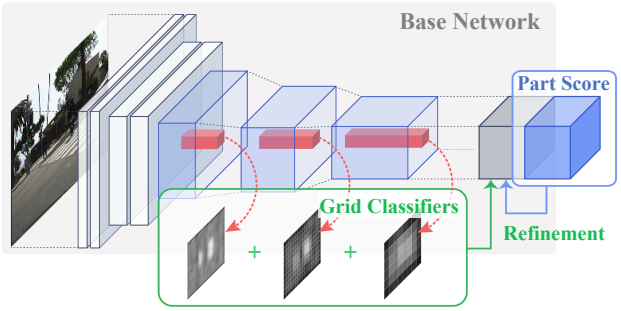


Figure 1: The overview of our refinement system.

While recent research progress has been significant on improving detection accuracy and speed, there are still big challenges. To be specific, we limit our discussion to *pedestrian detection* [4, 24, 25], 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.* [24] 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 peo-

ple 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.

Our objective is to propose the approaches that address these two problems of occlusion and hard negative examples. 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 recent state-of-the-art single-stage models, SqueezeDet+ [22], YOLOv2 [17], SSD [12], and DSSD [8]. We empirically validate that our approach indeed improves the performance of those four models on Caltech pedestrian [4] and CityPersons [25] dataset. As shown in Figure 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.

1.1. Related work

We briefly discuss previous research that tackles the two target problems: occlusion and hard negative examples.

Models for occlusion. The part-based methods [13, 21] have been one of the most dominant approaches addressing the occlusion problem. Mathias *et al*. [13] propose the Franken-classifiers, consisting of a set of occlusion-specific classifiers using Integral Channel Features [3]. DeepParts [21] 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*. [6] leverage the features of intensity, depth and motion to build a part-based mixture-of-experts classification model. Ouyang *et al*. [14] propose a probabilistic framework that can predict well even with inaccurate scores of part detectors by modeling part visibility as latent variables. Later they [15] extend the probabilistic framework to represent the relations between the configurations estimated by single- and multi-pedestrian detectors. Tang *et al*. [20] develop a double-person detector and tracker that can detect multiple people that occlude one another, based on the DPM model [7].

One of the most relevant works to ours is the DeepParts [21] 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 pre-defining semantic parts; instead, the best visibility patterns are directly learned from part confidence maps.

Models for hard negative examples. False-positives due to hard negative examples account for a large portion of the errors in the pedestrian detection problem [24]. 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 [12] and DSSD [8] select only three times of negative anchors (than positive ones) with the highest classification loss for training.

Recently, some state-of-the-art models [23, 5, 11] introduce additional post-refinement classifiers to reject hard negatives. For example, Zhang *et al*. [23] apply a boosted forest classifier to the candidate boxes of pedestrians that are obtained by the RPN. Du *et al*. [5] exploit multiple neural networks in parallel for further refinement of pedestrian candidates obtained by the SSD. Compared to Du *et al*. [5] and Hu *et al*. [11], 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.

1.2. Contributions

Our main contributions are two-fold.

(1) We propose an approach to address the two critical issues of pedestrian detection: (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 [4] and CityPersons [25] dataset. First, we show that our approach is integrable with four state-of-the-art single-stage models, SqueezeDet+ [22], YOLOv2 [17], SSD [12], and DSSD [8]. 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.

2. A Unified View of Output Tensors

Most single-stage networks formulate the detection as a regression problem, and generate a tensor as prediction output [16, 17, 12, 8, 22]. As shown in Figure 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 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 $(x_i, y_j, w_k, h_k), i \in [1, W], j \in [1, H], k \in [1, K]$. All the models use the scale parameters (δ_w, δ_h) to describe how different the scale is compared to that of an anchor:

$$\delta_{w,(ijk)} = \log\left(\frac{w_{gt}}{w_k}\right), \ \delta_{h,(ijk)} = \log\left(\frac{h_{gt}}{h_k}\right). \quad (1)$$

For the position parameters (δ_x, δ_y), YOLOv2 [16, 17] predicts the relative position of top-left corner in the grid with a bound of $[0, 1)$ in Eq.(2), whereas SqueezeDet+ [22], SSD [12], and DSSD [8] predict the relative position of center point to the anchor in Eq.(3).

$$\delta_{x,(ijk)} = \sigma\left(\frac{x_{gt} - x_i}{w_{grid}}\right), \delta_{y,(ijk)} = \sigma\left(\frac{y_{gt} - y_j}{h_{grid}}\right) \quad (2)$$

$$\delta_{x,(ijk)} = \frac{x_{gt} - x_i}{w_k}, \qquad \delta_{y,(ijk)} = \frac{y_{gt} - y_j}{h_k} \quad (3)$$

where σ is the sigmoid.

For the object likelihood, YOLOv2 and SqueezeDet+ define the confidence of object presence in Eq.(4), and follow the conditional probabilities of C object classes in Eq.(5). The final likelihood is obtained by multiplying the conditional probabilities by the confidence.

$$c_{(ijk)} = P_{(ijk)}(\text{Object}) \times \text{IOU}^{gt}_{(ijk)}, \quad (4)$$

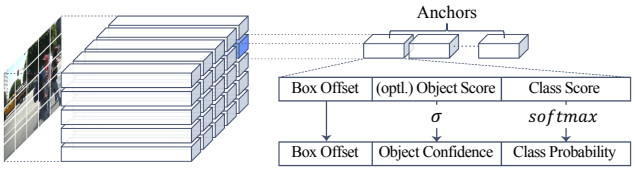
$$p_{m,(ijk)} = P_{(ijk)}(\text{Class} = m \mid \text{Object}), m \in [1, C]. \quad (5)$$

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.(6):

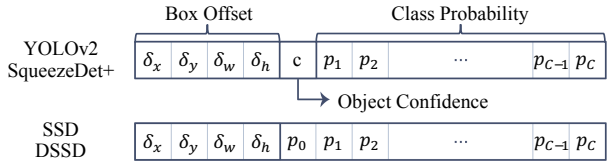
$$p_{m,(ijk)} = P_{(ijk)}(\text{Class} = m), \quad m \in [0, C]. \quad (6)$$

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 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.



(a) Structure of the output tensor.



(b) Output formats of four methods per anchor.

Figure 2: A unified view of output tensors of four methods: YOLOv2, SqueezeDet+, SSD, and DSSD.

Model	Shape of Output Tensors $\{W, H, K\}$
SqueezeDet+ [22]	$\{38, 28, 9\}$
YOLOv2 [17]	$\{20, 15, 9\}$
SSD [12]	$\{40, 30, 4\}$, $\{20, 15, 3\}$, $\{10, 8, 3\}$, $\{8, 6, 2\}$, $\{6, 4, 1\}$
DSSD [8]	$\{1, 1, 3\}$, $\{6, 4, 3\}$, $\{8, 6, 6\}$, $\{10, 8, 6\}$, $\{20, 15, 3\}$, $\{40, 30, 3\}$, $\{80, 60, 3\}$, $\{160, 120, 1\}$

Table 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. 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 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*.

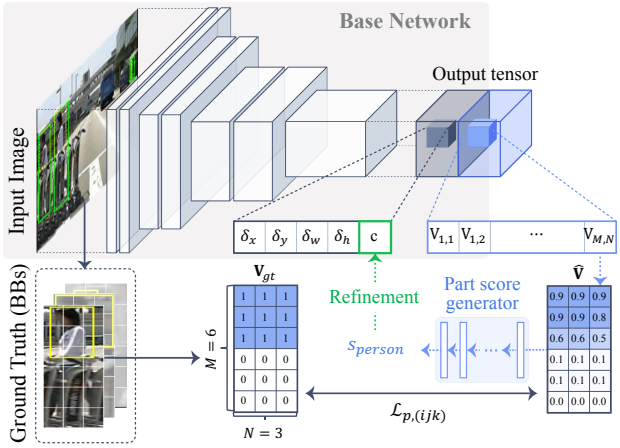


Figure 3: The overview of our occlusion handling method.

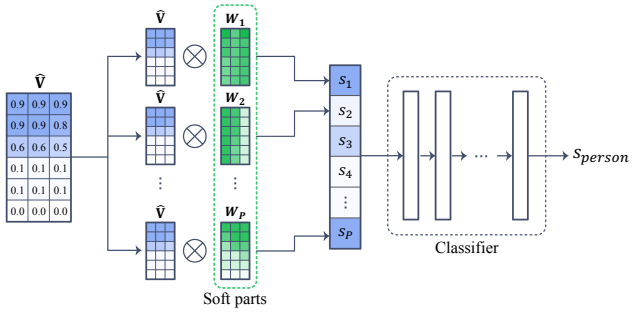


Figure 4: The generator module for the soft part score.

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. 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 $M = 6, N = 3, \tau_v = 0.4$.

3.1. 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). 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*.

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). 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:

$$s_{\text{person}} = \max_{m,n} \widehat{\mathbf{V}}(m, n). \tag{7}$$

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 [4], more than 97 % of occluded persons belong to only seven sets of occlusion patterns. As discussed in section 1.1, the DeepParts approach [21] 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 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 predicted part confidence map $\widehat{\mathbf{V}}$:

$$s_p = \sum_{m=1}^{M} \sum_{n=1}^{N} \left(\widehat{\mathbf{V}}(m, n) \cdot \mathbf{W}_p(m, n) \right) \tag{8}$$

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:

$$s_{\text{person}} = \sigma \left(\mathbf{w}_2^\top \max \left(\mathbf{0}, \mathbf{w}_1^\top \mathbf{s} \right) \right) \tag{9}$$

$\{\mathbf{W}_p\}_{p=1}^{P}$ 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.(9) performs the best.

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

$$c' = \sqrt{s_{\text{person}} c}. \tag{10}$$

3.2. The Training Objective

Single-stage models used in this paper 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:

$$\mathcal{L}_{p,(ijk)} = \left(\lambda_p^+ \mathbb{I}_{(ijk)}^+ + \lambda_p^- \mathbb{I}_{(ijk)}^- \right) \times \sum_{m=1}^{M} \sum_{n=1}^{N} \left(\mathbf{V}_{(ijk)}(m, n) - \widehat{\mathbf{V}}_{(ijk)}(m, n) \right)^2 \tag{11}$$

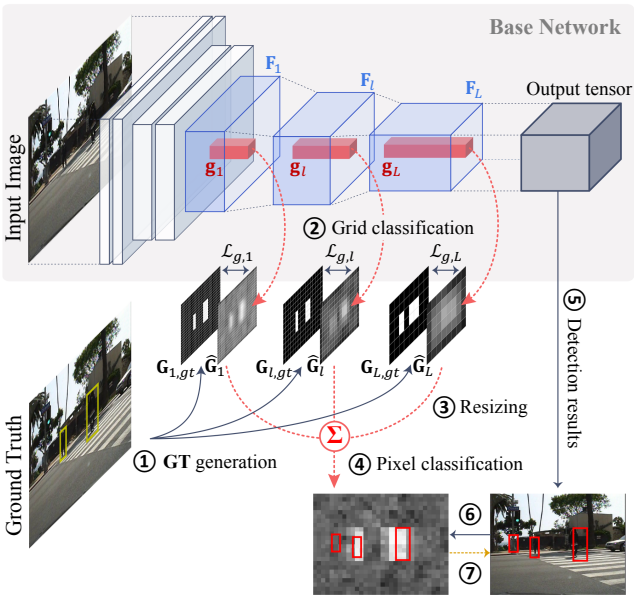


Figure 5: The overview of hard negative handling method.

Models	Shapes of grid confidence maps $\{w_l, h_l\}$
SqueezeDet+ [22]	$\{78, 58\}$, $\{38, 28\}$
YOLOv2 [17]	$\{80, 60\}$, $\{40, 30\}$, $\{20, 15\}$
SSD [12]	$\{40, 30\}$, $\{20, 15\}$
DSSD [8]	$\{40, 30\}$, $\{20, 15\}$, $\{40, 30\}$

Table 2: The dimensions of grid confidence maps for a 640×480 input image (w_l: width, h_l: height).

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

$$\mathcal{L}_{s,(ijk)} = \left(\lambda_s^{+f}\mathbb{I}^{+f}_{(ijk)} + \lambda_s^{+o}\mathbb{I}^{+o}_{(ijk)}\right)\left(1 - \hat{s}_{p,(ijk)}\right)^2 + \lambda_s^{-}\mathbb{I}^{-}_{(ijk)}\hat{s}^2_{p,(ijk)} \quad (12)$$

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:

$$\mathcal{L} = \sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{k=1}^{K}\left(\lambda_l\mathcal{L}_{l,(ijk)} + \lambda_c\mathcal{L}_{c,(ijk)} + \lambda_p\mathcal{L}_{p,(ijk)} + \lambda_s\mathcal{L}_{s,(ijk)}\right). \quad (13)$$

4. 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 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 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 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 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 $\mathbf{g}_l$:

$$\widehat{\mathbf{G}}_l = \mathbf{F}_l * \mathbf{g}_l, \quad l \in [1, L]. \quad (14)$$

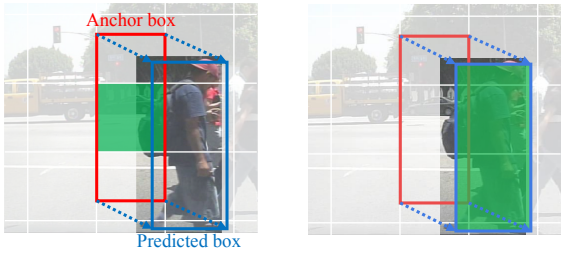
Once we compute a set of $\{\widehat{\mathbf{G}}_l\}_{l=1}^{L}$ for all L layers, we resize them to be the same with the input image using a bilinear interpolation: $\{\widehat{\mathbf{G}}'_l\}_{l=1}^{L}$. Finally we obtain a single averaged confidence map: $\widehat{\mathbf{G}} = \frac{1}{L}\sum_{l=1}^{L}\widehat{\mathbf{G}}'_l$, 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:

$$s_k = \frac{1}{w_k h_k}\sum_{i=x_k}^{x_k+w_k-1}\sum_{j=y_k}^{y_k+h_k-1}\widehat{\mathbf{G}}(i, j). \quad (15)$$

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

$$c'_k = \sqrt{s_k c_k}, \quad k \in [1, B]. \quad (16)$$

The intuition of why this method works is two-fold. First, except SSD [12] and DSSD [8] that use multiple feature maps, other single-stage models generate output tensors only from one (*e.g.* SqueezeDet+ [22]) or two feature maps (*e.g.* YOLOv2 [17]). 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



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

Figure 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.

detection decision based on the average of multi-resolution feature maps. Concatenating feature maps of several layers [1] or using skip connections [8, 19] 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 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.* [10, 9, 18]) use the features on the actual region of a predicted bounding box (Figure 6b), whereas the single-stage models use the features where the default anchor is located. (Figure 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.

4.1. The Training Objective

For the grid classifier, we add the grid loss $\mathcal{L}_g$ to the localization and confidence losses in Section 3.2. 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

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

5. Experiments

We focus on validating that the proposed approach for occlusion and hard negative handing indeed help improve accuracies of pedestrian detection. We use two benchmark datasets: Caltech pedestrian [4] and CityPersons [25]. We apply our approach into four state-of-the-art single-stage models: SqueezeDet+ [22], YOLOv2 [17], SSD [12], and DSSD [8]. For fair comparison, we implement all methods using TensorFlow. 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.

We evaluate our occlusion handling method (section 5.2), hard negative handling method (section 5.3), and joint learning of the two methods (section 5.4). We also analyze the size/time overhead of our approach (section 5.5). We present detailed experimental setup and additional results in the supplementary file.

5.1. Experimental Setting

Dataset. 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 experimental 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 [2] 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. For training, we augment the dataset 5 times with shifting and flipping, and add noise to training images by changing brightness, saturation, and contrast at random.

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.

5.2. Evaluation on 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

Model	Reasonable	All	None	Partial	Heavy
SqueezeDet+ [22]	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 [17]	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 [12]	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 [8]	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: 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).

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.

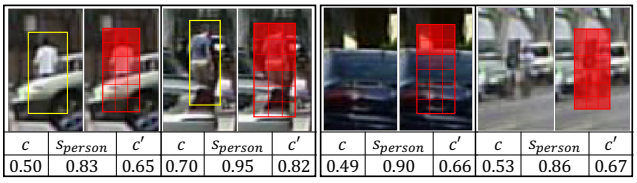
Table 3–5 show the breakdown performance of our occlusion handling on Caltech and CityPersons dataset. Our methods of max/soft part scores lead significant performance improvement over all four base models. Overall, the error rates can be sorted in the following order: soft $<$ max $<$ base. The max 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.

Table 4 shows additional results for SSD and DSSD models on the test subset of height $\geq$ 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 $\geq$ 20), and observe that our occlusion handling consistently improve SSD and DSSD to detect very small and highly occluded pedestrians.

Figure 7 shows examples of success and failure cases of our occlusion handling. In the success cases of Figure 7a, the initial confidences for the person are relatively low (*e.g.* $c = 0.50$ and 0.70), but they are correctly adjusted thanks to the high part scores (*e.g.* $c' = 0.65$ and 0.82). Figure 7b shows some negative cases, in which some hard negatives such as vertical objects confusingly look like pedestrians, and thus their confidences increase in a wrong direction.

5.3. Evaluation of Hard Negative Handling

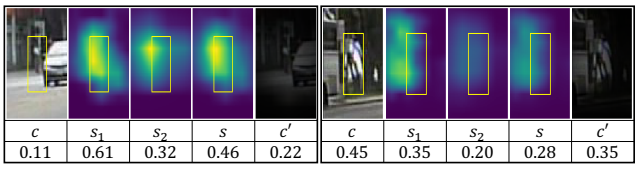
Table 4–5 show the detailed breakdown performance of our hard negative handling on the two datasets. The performance is always better than baseline when the grid classifiers are used only for training. However, if we use the adjusted confidence, SqueezeDet+ and YOLOv2 perform the best, but SSD and DSSD become worse than the baseline. As discussed in section 4, there are two cases where the grid classifiers are helpful: i) the refinement by the averaged results from multiple feature maps, and 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.



(a) Success cases (b) Failure cases

Figure 7: Examples of occlusion handling. For better visualization, we crop detection regions from images.

Figure 8 shows the examples of applying the grid classifiers to the SqueezeDet+ model. As in Table 2, SqueezeDet+ uses two feature maps (*i.e.* $L = 2$), from which we generate the grid confidence map. In Figure 8b, our grid classifiers effectively suppress the confidence scores of false positives. The score of initially misclassified vertical structure is reduced from 0.45 to 0.35. We also find that there are some cases where our method increases the confidence scores of hard positives as in 8a. The score of a faint pedestrian is doubled through the refinement of grid classifiers.



(a) Positive example (b) Negative example

Figure 8: Examples of grid classifiers in SqueezeDet+. For better visualization, we crop detection regions from images.

5.4. Evaluation of Joint Learning

Table 4–5 also show the performance of joint application of the two methods for occlusion and hard negative handling. In this case, the adjusted confidence is computed as the geometric mean of a part score (Eq.(7) or Eq.(9)), an averaged confidence score (Eq.(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

Model	Height ≥ 50					Height ≥ 20			
	Reasonable	All	None	Partial	Heavy	All	None	Partial	Heavy
SqueezeDet+ [22]	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 [17]	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 [12]	16.36	25.18	14.55	27.89	53.80	60.19	52.21	**67.96**	76.47
+ Part score	14.23	**22.53**	12.22	27.52	**50.46**	58.94	51.71	68.85	**74.37**
+ Grid classifiers*	**14.04**	23.79	**12.03**	**26.52**	55.10	59.66	51.60	68.93	76.04
+ Joint learning*	15.03	23.54	13.06	29.57	51.53	**58.88**	**51.52**	70.71	74.81
DSSD [8]	13.25	20.53	11.23	25.23	44.13	53.03	44.72	64.15	69.59
+ Part score	10.97	18.58	**8.88**	26.14	44.11	50.55	41.51	61.68	69.65
+ Grid classifiers*	**10.85**	**18.20**	9.00	24.28	**42.42**	**49.24**	**41.32**	**60.74**	**65.99**
+ Joint learning*	11.42	19.38	10.00	**21.11**	45.80	52.00	43.88	61.57	69.50

Table 4: Overall performance on Caltech *test* dataset (lower is better). * denotes that grid classifiers are used only for training.

Model	Reasonable	All	None	Partial	Heavy
SqueezeDet+ [22]	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 [17]	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 [12]	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 [8]	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 5: Overall performance on CityPersons *val* dataset (Height ≥ 50). * denotes that grid classifiers are used only for training.

Model	Baseline	Additional methods		Total
		+ Part score	+ Grid cls.	
SqueezeDet+ [22]	27.59	1.99	0.04	29.62
YOLOv2 [17]	268.35	0.45	0.06	268.86
SSD [12]	93.06	4.65	0.06	97.77
DSSD [8]	345.07	2.07	0.09	347.23

Table 6: Comparison of model sizes (in MB).

Model	Baseline	Additional methods		Total
		+ Part score	+ Grid cls.	
SqueezeDet+ [22]	23.02	0.89	0.54	24.45
YOLOv2 [17]	32.19	0.70	1.12	34.01
SSD [12]	32.50	1.08	1.18	34.76
DSSD [8]	84.36	0.97	1.55	86.88

Table 7: Comparison of inference time (in milliseconds).

as discussed in section 5.3. 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).

5.5. Memory and Computation Time Analysis

We report model sizes and computation times of our methods in Table 6–7, 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.

6. 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 state-of-the-art single-stage models, including SqueezeDet+ [22], YOLOv2 [17], SSD [12], and DSSD [8]. We demonstrated that our approach indeed improved the performance of four base models for pedestrian detection on Caltech [4] and CityPersons [25] 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.

Acknowledgements. We thank Yunseok Jang and Juyong Kim for helpful discussions. This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-TC1603-01. Gunhee Kim is the corresponding author.

References

[1] 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.

[2] 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.

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

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

[5] 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 *IEEE WACV*. IEEE, 2017.

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

[7] 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.

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

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

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

[11] 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.

[12] 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.

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

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

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

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

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

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

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

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

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

[22] 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.

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

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

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