Multi-task Self-supervised Object Detection via Recycling of Bounding Box Annotations

Wonhee Lee Joonil Na Gunhee Kim
Seoul National University, Seoul, Korea
{wonhee, joonil}@vision.snu.ac.kr, gunhee@snu.ac.kr
http://vision.snu.ac.kr/projects/mtl-ssl-detection

Abstract

*In spite of recent enormous success of deep convolutional networks in object detection, they require a large amount of bounding box annotations, which are often time-consuming and error-prone to obtain. To make better use of given limited labels, we propose a novel object detection approach that takes advantage of both multi-task learning (MTL) and self-supervised learning (SSL). We propose a set of auxiliary tasks that help improve the accuracy of object detection. They create their own labels by recycling the bounding box labels (*i.e. *annotations of the main task) in an SSL manner, and are jointly trained with the object detection model in an MTL way. Our approach is integrable with any region proposal based detection models. We empirically validate that our approach effectively improves detection performance on various architectures and datasets. We test two state-of-the-art region proposal object detectors, including Faster R-CNN [39] and R-FCN [10], with three CNN backbones of ResNet-101 [22], Inception-ResNet-v2 [45], and MobileNet [23] on two benchmark datasets of PASCAL VOC [14] and COCO [30].*

1. Introduction

Recently, there has been significant progress in the field of object detection [39, 10, 21], leveraging deep convolutional networks that can learn hierarchical feature representation of input images. However, training a practical object detection model requires a large amount of bounding box annotations, which are often time-consuming and error-prone to obtain.

To mitigate the label shortage problem for deep neural networks, many studies have been conducted; in the context of object detection, multi-task learning and self-supervised learning may be two exemplary approaches for the problem. Multi-task learning (MTL) aims at jointly training multiple relevant tasks with less annotations to improve the performance of each task [33, 13, 46]. Its effectiveness has been well studied; for example, it leads the effect of regularization by providing inductive bias to each other [5]. It is also proven that as the number of tasks increases, the upper bound of the number of annotated data for better generalization decreases [3]. One of the most successful examples where MTL is helpful for object detection is Mask R-CNN [21], which enhances the performance of object detection by jointly performing an instance segmentation task. However, one of its practical limitations is that segmentation mask labels, which are more expensive than bounding box annotations, must be provided.

Self-supervised learning (SSL) aims at training the model from the annotations generated by itself with no additional human effort [9, 42]. In object detection literature, SSL has been applied to replace ImageNet pretraining [34, 53, 36, 25]. Its motivation is that creating a large-scale database like ImageNet is highly challenging and requires a lot of human effort, so it tries to pretrain the network from relevant tasks that do not require human-annotated data such as Jigsaw puzzles [34] or colorization [53]. However, the performance of most SSL algorithms is not as good as ImageNet pretraining, and thus it mostly fail to deliver practical benefit to object detection.

In this work, we propose a novel object detection approach that takes advantage of both multi-task learning and self-supervised learning. We start from a normal supervised object detection setting, where a region proposal based detector (*e.g.* Faster R-CNN [39] and R-FCN [10]) is given and bounding box annotations are available from the dataset. The key to our approach is to propose a set of auxiliary tasks that are relevant but not identical to object detection. They create their own labels by recycling the bounding box labels (*i.e.* annotations of the main task) in an SSL manner while regarding the bounding box as metadata. Then these auxiliary tasks are jointly trained with the object detection model in an MTL way. Our focus here is to improve the performance of the main task (object detection) rather than all main and auxiliary tasks.

Although auxiliary tasks are supposed to stimulate the main task to achieve better accuracy, it is hard to define appropriate and valid auxiliary tasks that actually help. In many cases, auxiliary task might be ineffective or even interfere with the main task. We empirically find three synergetic auxiliary tasks, including *multiple-object*, *closeness* and *foreground* labeling tasks. Thorough experimental results show that the proposed auxiliary tasks are substantially effective for accurate object detection.

The contributions of this work are outlined as follows.

1. To the best of our knowledge, this work is a first attempt to develop a multi-task self-supervised learning approach for two-stage object detection models. Our approach is orthogonal to any choice of region proposal based detection models.
2. We design a set of three auxiliary tasks that help improve the performance of object detection, while reusing bounding box annotations without any additional human effort. As far as we know, there has been no previous work to recycle bounding box annotations like ours in the self-supervised learning literature.
3. We demonstrate the accuracy improvement of our approach in multiple architectural combinations. We test two state-of-the-art region proposal detectors, including Faster R-CNN [39] and R-FCN [10], with three base CNN backbones of ResNet-101 [22], Inception-ResNet-v2 [45] and MobileNet [23] on two benchmark datasets of PASCAL VOC [14] and COCO [30].

2. Related Work

2.1. Multi-task Learning (MTL)

The MTL trains related tasks together to overcome the shortage of annotated data. It provides each task with inductive bias [5] to trigger regularization effect between one another, and decreases the upper-bound on the number of annotated data for better generalization as the number of tasks increases [3]. MTL has demonstrated its usefulness in a number of computer vision tasks, including depth estimation and scene parsing [49], synthetic imagery generation [40, 50], attributes prediction [1], immediacy prediction [8], person re-identification [43], and facial action unit detection [2].

Parameter Sharing. MTL methods can be classified into two groups according to how to share the parameters between different task models. In the hard parameter sharing, all task models share the exact same feature extractor and perform its own task through each branch head. Therefore, the main issue here is to design appropriate tasks and objective functions. Some examples in this category include TCDCN [55], HyperFace [38], Mask R-CNN [21], ResNetCrowd [31], LASSO architecture [13]. In the soft parameter sharing, each task has its own model with its own parameters. Hence, the methods in this category focus on how to design weight sharing methodology, such as what constraints and distance metrics are utilized between the parameters. Examples include cross-stitch network [33], DCNet [46], cross connection [15], partially shared structure [4], Sluice Networks [41] and NDDR-CNN [16].

2.2. Self-supervised Learning (SSL)

While it takes a lot of human effort to create high-quality annotations for supervised learning, SSL [9, 42] creates labels by models themselves without additional human effort. In computer vision research, different types of information have been adopted as a signal for SSL, including colorization [53, 54], inpainting [37], spatial patches [12, 34] or temporal clues [29, 44, 47], text [6, 19, 26], sound [35], optical flow [36] and tracking [27, 48].

Transfer Learning. One of the primary uses of SSL is in transfer learning. In the field of object detection, many self-supervision tasks have been applied to replace ImageNet pretraining. Noroozi *et al*. [34] propose a pretext task to solve Jigsaw puzzles, and transfer the networks learned for Jigsaw puzzles to solve object classification and detection. Pathak *et al*. [36] pretrain networks via motion-based grouping cues on videos. Jenni and Favaro [25] introduce a pretext task to differentiate between real and artifact-containing images, and train the model in an adversarial manner for transferring to object detection. Zhang *et al*. [53] train a network for the colorization task and finetune it for the detection task. There have been many other attempts to pretrain the network from relevant tasks in self-supervised or unsupervised manner, but most of them still have not reached the performance of ImageNet pretraining. Our work is distinguishable from this line of work in that we do not try to replace the ImageNet pretraining but introduce a set of complementary auxiliary tasks that are trainable with no additional annotations and improve the performance of object detection.

Annotation Reuse. Gong *et al*. [20] and Zhan *et al*. [51] show that reusing labels of one task is not only helpful to create new tasks and their labels but also capable of improving the performance of the main task through pretraining. They use pixel-wise segmentation masks as the annotation to be reused in the context of human parsing and semantic segmentation, respectively. On the other hand, our work focuses on recycling bounding box labels for object detection, which has not been discussed yet.

3. Approach

We design a multi-task self-supervised learning model for object detection. We assume that annotations are available only for the main task (*i.e*. bounding box labels for object detection). We introduce a set of three auxiliary tasks

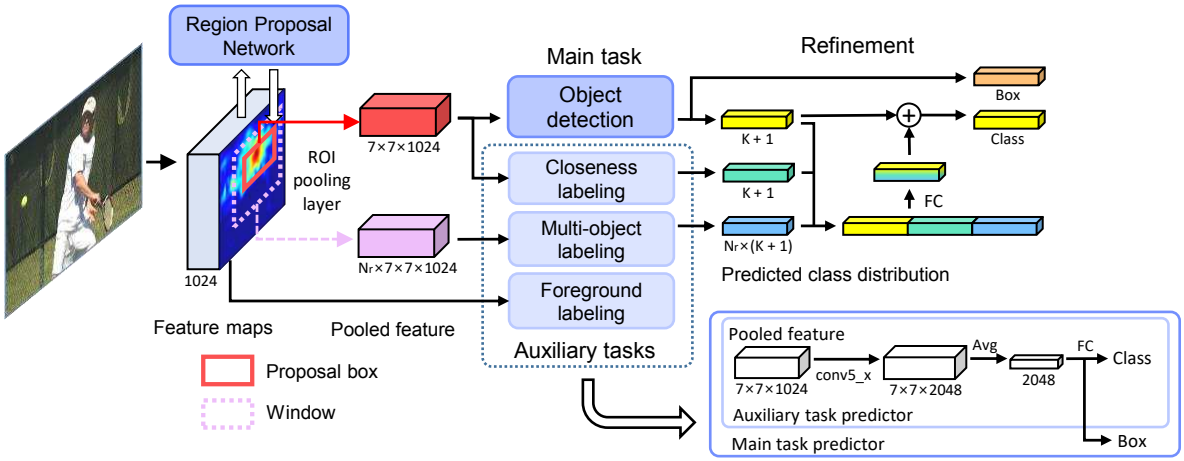

Figure 1: Overall architecture of our multi-task self-supervised approach. It shows how the object detector (*i.e.* main task model) such as Faster R-CNN [39] makes a prediction for a given proposal box (red) with assistance of three auxiliary tasks (section 3.1) at inference. The auxiliary task models (shown in the bottom right) are almost identical to the main task predictor except no box regressor (section 3.2). The refinement of detection prediction (shown in right) is also collectively done by cooperation of the main and auxiliary task models (section 3.3). K is the number of categories.

that are jointly trained with the main task in a multi-task learning manner. The auxiliary task models are free to recycle bounding-box annotations for building their own ground truth (GT) labels in a self-supervised way. Here our goal is to improve the performance of the main task (object detection), instead of the averaged performance of all main and auxiliary tasks. We discuss the details of the auxiliary tasks and their models in section 3.1–3.2.

The proposed auxiliary tasks are beneficial in both feature extraction and prediction. First, the three auxiliary tasks enhance the quality of the shared features through cooperative feature learning. Second, the outputs of the auxiliary tasks provide contextual information to refine the object detection prediction, especially classification accuracy of region proposals. We refer to this process as *refinement*, which will be discussed in section 3.3.

Figure 1 shows the overall architecture of our model, where three auxiliary task models are integrated with a region proposal-based object detector (*e.g.* Faster R-CNN). It shows how a single RoI is processed at inference time. We will present the training of the whole model in section 3.4.

3.1. Auxiliary tasks

We below describe our three auxiliary tasks, including *multiple-object*, *closeness* and *foreground* labeling tasks.

Multi-object labeling. The annotation for object detection normally consists of two types of information: i) the coordinates of the smallest bounding box that encloses the target object and ii) one-hot encoding for a single corresponding class. The first auxiliary task named as *multi-object labeling* relaxes these two labeling conditions. It randomly samples a bounding box window in the image and assigns a *soft* label to it rather than a hard one-hot encoding, to be interpreted as a probability of several classes in a single window. The key benefit of this auxiliary task is to populate many positive boxes even though their quality may not be as good as that of GTs. Nonetheless, it can alleviate one important issue of the general object detection pipeline where positive boxes are too few compared to negative boxes per image. This has similar intention to *mixup* [52], which combines pairs of images and their labels linearly.

Figure 2 shows some examples of windows for multi-object labeling. We first sample N_t number of windows (*e.g.* $N_t = 64$ in our experiments) by randomly picking their top-left corners and width/height at the minimum size of 32. We constrain that the windows should have nonzero intersection with any GT boxes in the image. We then obtain a soft label $\boldsymbol{l}^m$ for each window, following Algorithm 1. The label $\boldsymbol{l}^m$ acts as a GT annotation for the multi-object labeling task. Simply, we assign a class probability to a window W, according to its area portion with GT classes. The length of $\boldsymbol{l}^m$ is $K+1$, where K is the number of classes and $\boldsymbol{l}^m[0]$ denotes the *no object* probability that is proportional to the background area in W.

Closeness labeling. The distribution of objects in an image is not random. For example, in the PASCAL VOC images, there is likely to be chairs near a dining table and cars near a bus. In the skip-gram model of natural language processing [32], the model learns the meaning of a specific word from those of surrounding words in a sentence. Likewise, if an auxiliary task enforces the model to learn to predict both the class and its surrounding classes using the fea-

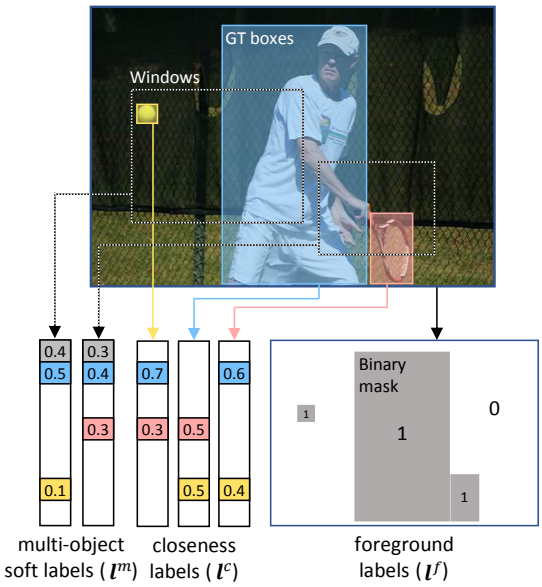

Figure 2: An example of how to generate labels of auxiliary tasks via recycling of GT bounding boxes. The multi-object soft label assigns the area portions occupied by each class's GT boxes within a window. The closeness label scores the distances from the center of the GT box to those of other GT boxes. The foreground label is a binary mask between foreground and background.

tures of a region proposal, the learned features are likely to encode the context information of the image region. This may enable the model to predict the class of the box better. We coin this auxiliary task as *closeness labeling*. Figure 2 and Algorithm 2 show how the closeness auxiliary task recycles the GT boxes to obtain its own labels. Note that the closeness label $\boldsymbol{l}^c$ is only defined for GT boxes, whereas the previous multi-object label $\boldsymbol{l}^m$ is for a randomly sampled window. Therefore, the closeness auxiliary task predicts possible objects around the box, while the multi-object labeling task predicts the possible objects within the window. The closeness label $\boldsymbol{l}^c$ for each GT box b assigns a higher value to the object whose GT box is closer to b. $\boldsymbol{l}^c[0]$ is 1 if there is no GT box nearby.

Foreground labeling. The final auxiliary task, named as *foreground labeling*, aims at predicting the foreground and background regions in the entire image. This task can aid the feature learning to be more accurate for the coordinate regression of region proposals. As shown in Figure 2, it is simple to generate a label $\boldsymbol{l}^f$ for this task; we simply assign 1 to the GT box regions and 0 to the other regions.

3.2. Auxiliary Task Models

As shown in Figure 2, the three auxiliary tasks eventually predict class probability labels (*i.e.* $\boldsymbol{l}^m, \boldsymbol{l}^c, \boldsymbol{l}^f$), even though

Algorithm 1: OBTAINING A MULTI-OBJECT LABEL

Input: Image I, GT boxes $\{B_i\}_{i=1}^K$, Window W
Output: The multi-object soft label $\boldsymbol{l}^m$ for W
$\boldsymbol{l}^m \leftarrow$ An array of length $(K+1)$
$\boldsymbol{l}^m[0] \leftarrow \sqrt{area(W) - area((\cup_{i \in K}\{B_i\}))}$
for $i \leftarrow 1$ ***to*** K **do**
  $\boldsymbol{l}^m[i] \leftarrow \sqrt{area(W \cap \{B_i\})}$
return $\boldsymbol{l}^m / sum(\boldsymbol{l}^m)$

Algorithm 2: OBTAINING CLOSENESS LABEL

Input: Image I, GT boxes $\{B_i\}_{i=1}^K$, A GT box b
Output: The closeness soft label $\boldsymbol{l}^c$ for b
$\boldsymbol{l}^c \leftarrow$ An array of zeros with a length $(K+1)$
$D \leftarrow$ The diagonal distance of I
if $\{B\} - b = \emptyset$ **then**
  $\boldsymbol{l}^c[0] \leftarrow 1$
else
  for $i \leftarrow 1$ ***to*** K **do**
    $\boldsymbol{l}^c[i] \leftarrow D - \min_{b' \in \{B_i\}} \|center(b) - center(b')\|_2$
return $\boldsymbol{l}^c / sum(\boldsymbol{l}^c)$

the purpose of each task is different one another. Hence, we design the models for auxiliary tasks to have the same architecture with the head of the main task model. For instance, the lower part of Figure 1 shows the auxiliary task model for Faster R-CNN [39] with ResNet-101 [22]. Only difference between the main and auxiliary predictor is the existence of box regression, which is unnecessary for auxiliary tasks.

Such architecture sharing is advantageous in several aspects. First, it makes our multi-task approach easily integrable with object detection models, because the implementation of auxiliary tasks is straightforward. Second, it makes it easy to initialize weights for auxiliary models by simply duplicating those of the pretrained detector. Such replicate initialization empirically leads to better performance than training the heads of auxiliary tasks from scratch.

3.3. Detection Refinement

The auxiliary tasks are beneficial in both feature extraction and prediction. In the region proposal stage, the auxiliary tasks are jointly trained with the main task to learn shared features that help object detection prior to RoI pooling. In the prediction stage, the outputs of auxiliary tasks can directly refine the detection prediction, especially classification of region proposals. In this section, we discuss the detection refinement in the second stage.

The multi-object labeling model can predict soft class memberships for a given proposal and boxes surrounding

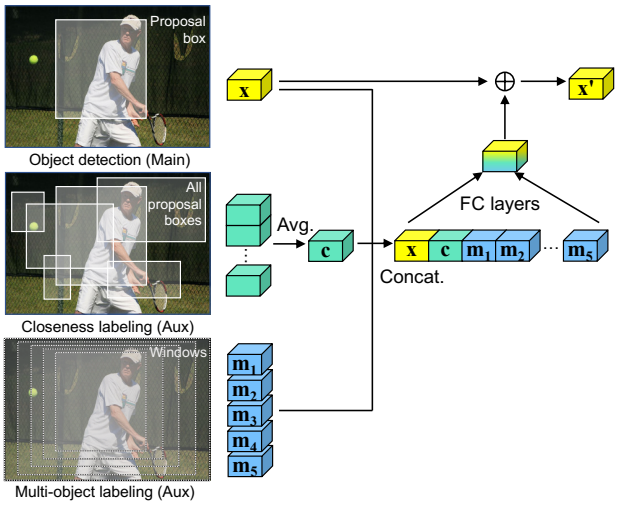

Figure 3: Detection refinement. The main detection head computes $\mathbf{x}$ as a classification result for a proposal box. It is updated into $\mathbf{x}'$ using a single FC layer on the prediction outputs of the two auxiliary models, $\mathbf{c}$ and $\mathbf{m}_1, \cdots, \mathbf{m}_r$.

it. The closeness labeling model can predict the possible co-occurrence of nearby objects even if they do not actually exist. The key idea of our detection refinement is to let the main task head (*i.e.* object detector) take advantage of the predictions of the two auxiliary tasks, because they can provide useful contextual information for the detector to make better decision for classification. That is, for a given proposal box that the object detector needs to predict, the multi-object model provides soft label prediction for the local and global context around the box, while the closeness model delivers predicted proximity to the surrounding objects. We here do not use the foreground labeling task's output because it has no additional information beyond the two auxiliary tasks.

Figure 3 shows the process of refinement. In a normal object detector (*e.g.* Faster R-CNN [39]), the detection head computes a classification result $\mathbf{x} \in \mathbb{R}^{K\times 1}$ for a given proposal box, and passes it through a softmax layer to generate a class probability $\mathbf{y}$. Our refinement updates $\mathbf{x}$ into $\mathbf{x}'$ using the auxiliary models' outputs as follows.

First, in order to leverage the prediction by the learned multi-object labeling model, we create N_r number of windows (*e.g.* $N_r = 5$ in our experiments) surrounding the proposal with various sizes. We set the sizes by dividing the space between the whole image and the proposal box in $N_r - 1$ uniform intervals. We then obtain multi-object labels of N_r windows, denoted by $\mathbf{m}_1, \cdots, \mathbf{m}_{N_r}$. Second, we obtain the closeness labels for all proposals in the image, and average them into a single vector denoted $\mathbf{c}$. It is used as a context summary of the image, which is empirically better than using individual outputs for each proposal box. Finally, we obtain

$$\begin{aligned}\mathbf{x}' &= \text{refine}(\mathbf{x}|\mathbf{c}, \mathbf{m}_1, \ldots, \mathbf{m}_{N_r}) \\ &= \mathbf{W}_r[\mathbf{x}, \mathbf{c}, \mathbf{m}_1, \ldots, \mathbf{m}_{N_r}] + \mathbf{x},\end{aligned} \tag{1}$$

where $\mathbf{W}_r$ is a projection matrix. In summary, we concatenate $\mathbf{x}$, $\mathbf{c}$ and $\mathbf{m}_1, \cdots, \mathbf{m}_r$, and feed it into a fully-connected layer with a residual connection. The presented refinement model is designed after thorough validation; for example, we tried multiple FC layers instead of Eq.(1), but they were not as good as the single-layer version.

3.4. Training

Loss functions. We define the loss for each auxiliary task as a cross-entropy loss, since they basically perform prediction of class labels:

$$\mathcal{L}_* = -\frac{1}{N_*}\sum_{j=1}^{N_*} {y_j^*}^T \log\big(\text{softmax}(a_j^*)\big). \tag{2}$$

For the loss of multi-object labeling $\mathcal{L}_m$, we set $N_* = N_t$, $y_j^* = y_j^m$ and $a_j^* = a_j^m$ where N_t is the number of windows, y_j^m is the GT soft label vector for the j-th window and $\text{softmax}(a_j^m)$ is its predicted class probability by the auxiliary head in section 3.2. For the loss of closeness labeling task $\mathcal{L}_c$, we use N_p as the number of positive proposal boxes matched with GT boxes, y_j^c and $\text{softmax}(a_j^c)$ are GT and predicted soft label vectors for the surrounding area of the j-th box. Finally, for the foreground labeling task $\mathcal{L}_f$, we use N_f as the number of pixels on the foreground mask, y_j^f and $\text{softmax}(a_j^f)$ as the GT and predicted foreground label of the j-th pixel.

The overall auxiliary loss is the weighted sum of all task losses:

$$\mathcal{L}_{aux} = \lambda_m \mathcal{L}_m + \lambda_c \mathcal{L}_c + \lambda_f \mathcal{L}_f. \tag{3}$$

As the loss for the refinement, we use the same crossover entropy loss as in the main task for classification. We also apply *stop_gradient* operation, which ensures that the refinement loss does not affect the predictor of each task and the feature extractor. That is, since the main task and each auxiliary task have their own losses, the refinement loss updates only the weights of the refinement layers.

Finally, the overall loss $\mathcal{L}_{total}$ is the sum of the object detection loss $\mathcal{L}_{main}$ of the base detector, the auxiliary loss $\mathcal{L}_{aux}$ and the refinement loss $\mathcal{L}_{ref}$. We set $\lambda_m = \lambda_f = \lambda_r = 1$ and $\lambda_c = 0.3$ in our experiments.

$$\mathcal{L}_{total} = \mathcal{L}_{main} + \mathcal{L}_{aux} + \lambda_r \mathcal{L}_{ref}. \tag{4}$$

Training. We initialize the backbone CNN by ImageNet pretraining [11, 18]. We then simultaneously train the whole network using the GTs of both main and auxiliary tasks.

We freeze *conv1* and *conv2_x* layers in ResNet-101 [22] for fast convergence. For MobileNet [23] and Inception-ResNet-v2 [45], we do not freeze any layer.

Implementation. We resize images so that their minimum size is 600. We use the TensorFlow object detection API [24] for training. The *crop_and_resize* [7] method is employed instead of ROI pooling operation [17]. We use the momentum optimizer with a rate of 0.9 and a weight decay of 0.0001. We use only random horizontal flips for data augmentation.

4. Experiments

Our approach is applicable to any two-stage object detection models with region proposal. To show the generality of our approach, we evaluate with various architectures and datasets (section 4.1–4.2). We carry out ablation experiments about the effect of multi-task learning and refinement (section 4.3) and show some qualitative results (section 4.4). We present more results in the supplementary file.

4.1. Experimental Settings

Datasets. We test various configurations of datasets, following previous literature on object detection. We use three training settings: VOC07 *trainval*, VOC07+12 *trainval*, and COCO17 *train*, and four test settings: VOC07 *test*, VOC12 *test*, COCO17 *val*, COCO17 *test-dev*. More exact training/test splits are described in each table.

Object detection architecture. Our model is integrable with any two-stage object detection model. We choose Faster R-CNN [39] as one of the state-of-the-art detectors, and R-FCN [10] as another fully convolutional region proposal-based detection model.

Backbone CNNs. ResNet-101 [22] is one of the most popular backbones on object detection. MobileNet [23] is a lightweight efficient architecture for mobile and embedded applications. Inception-ResNet-v2 [45] is another state-of-the-art network that has a bigger size than ResNet-101 and attains better results in our experiments.

Evaluation metrics. We report the standard metrics for each dataset: the mean Average Precision (mAP) for VOC [14], and mAP over IoU from 0.5 to 0.95 (mAP@[.5:.95]) for COCO [30].

4.2. Detection Results

Table 1 shows the detection improvement of our approach over the baseline on the two datasets. We use Faster R-CNN with ResNet-101 as the baseline. More specifically, we present the detailed performance for VOC07+12 *trainval* and VOC12 *test* in Table 2 and for COCO17 *train* and COCO17 *test-dev* in Table 3. That is, we present the detailed detection accuracies over all 20 object classes of VOC in Table 2, and multiple mAP over IoU values, performance variation according to object sizes and additional average recall scores in Table 3. All the results assure that our auxiliary tasks consistently enhance the detection performance in various dataset splits. Table 1 shows that mAP values increase on average by about 2.0%p in VOC and about 1.9%p in COCO. Encouragingly, our approach leads to better performance in all 20 categories of VOC as in Table 2 and does too in all metrics of precision and recall regardless of object sizes in COCO as in Table 3.

Dataset	VOC			COCO	
Training	07	07+12		17 *train*	
Test	07	07	12	17 *val*	17 *test-dev*
Baseline	77.0	81.7	75.3	32.7	32.8
+ Task1	**78.9**	**83.8**	77.4	34.1	34.2
+ Task2	77.3	83.0	76.0	33.3	33.5
+ Task3	77.0	82.0	75.1	32.9	32.8
+ Task1,2	78.5	83.7	77.3	34.5	34.6
+ Task1,2,3	78.7	83.7	**77.5**	**34.6**	**34.7**

Table 1: Detection accuracies (mAP (%)) on VOC and COCO. Baseline is Faster R-CNN [39] with ResNet-101 [22]. Task1,2,3 indicate multi-object, closeness and foreground labeling auxiliary task, respectively.

Table 4 summarizes the performance variation according to backbone networks. Fixing Faster R-CNN as the detection architecture, we test MobileNet and Inception-ResNet-v2, in addition to ResNet-101 in Table 1. The performance gains by our method are more significant in the order of MobileNet, ResNet-101 and Inception-ResNet-v2. Given that it is the reverse order of the backbone's detection accuracy, the benefit of our approach could be larger when the network capability is inferior. Importantly, no matter what backbone CNN is used, our approach consistently improves the detection accuracy with substantial margins.

Table 5 reports the results for another object detection architecture, R-FCN, which turns out to be slightly worse than Faster R-CNN in Table 1. Still, we can see the same pattern that our approach nontrivially increases the detection accuracy in all experiments of PASCAL VOC.

Our experiments show the following trends of results about our approach. First, it is highly promising that our approach is constantly effective, regardless of base detectors, backbone CNNs and datasets. These results could sufficiently validate the generality of our MTL SSL approach. Second, the auxiliary tasks are individually more helpful for object detection in the order of task 1 (multi-object), 2 (closeness), and 3 (foreground). The task 1 is the most useful because it can generate many windows as needed, compared to the other tasks that create only a fixed handful number of labels; the labels of task 2 per image are bounded by the number of GT boxes and the label of task 3 is always

Method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Baseline	75.3	86.2	83.0	78.0	62.8	59.9	78.0	81.2	90.7	56.4	79.5	56.1	88.2	83.3	83.8	84.9	53.9	81.9	66.7	83.5	68.9
+ Task1	77.4	**88.0**	84.3	79.9	**63.6**	60.7	79.8	**82.6**	**93.2**	**58.0**	**84.5**	**59.4**	**91.5**	86.3	**86.5**	**86.0**	56.7	84.8	67.3	84.8	71.0
+ Task2	76.0	86.1	**84.4**	77.8	63.2	58.9	78.5	81.8	91.2	57.3	81.5	57.7	89.1	84.9	84.7	85.7	54.2	81.7	67.5	83.6	70.0
+ Task3	75.1	86.2	82.3	76.8	61.6	59.5	78.5	81.4	90.3	56.1	79.3	57.4	88.4	83.9	83.3	85.2	54.1	80.8	65.2	82.9	68.6
+ Task1,2	77.3	87.7	84.3	79.6	62.9	59.9	**80.1**	82.5	92.8	57.6	83.5	58.5	91.3	86.8	85.9	85.4	57.8	**85.1**	**70.1**	84.9	70.2
+ Task1,2,3	**77.5**	87.6	**84.4**	**80.4**	63.4	**61.2**	79.1	**82.6**	92.6	57.7	84.3	59.3	91.4	**87.1**	86.0	**86.0**	**57.9**	84.1	68.7	**85.7**	**70.3**

Table 2: Detailed performance on VOC 2012 *test*. The mAP values over 20 object classes of PASCAL VOC are also reported.

Method	Average Precision						Average Recall					
	IoU=.50:.95	IoU=.50	IoU=.75	small	medium	large	max=1	max=10	max=100	small	medium	large
Baseline	32.8	52.7	34.7	13.3	36.1	47.1	29.5	46.3	48.7	24.1	53.3	69.3
+Task1	34.2	55.2	36.1	14.1	37.7	49.7	29.7	46.6	49.1	23.9	**53.9**	**70.6**
+Task2	33.5	54.1	35.4	14.0	36.7	48.0	29.5	46.3	48.7	24.2	53.4	69.2
+Task3	32.8	52.6	34.6	13.2	35.9	46.8	29.5	46.3	48.6	24.1	53.0	69.3
+Task1,2	34.6	55.6	**36.6**	14.4	**38.1**	**50.0**	29.7	46.6	**49.2**	24.1	**53.9**	70.2
+Task1,2,3	**34.7**	**55.8**	**36.6**	**14.5**	**38.1**	**50.0**	**29.9**	**46.7**	**49.2**	**24.5**	**53.9**	70.3

Table 3: Detailed performance on COCO 2017 *test-dev*. The mAP metrics over multiple IoU values are reported. The results are also separately shown for 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. The average recall values are measured given $\{1, 10, 100\}$ detections at maximum per image.

Backbone	MobileNet [23]			Inception-ResNet-v2 [45]		
Training	07	07+12		07	07+12	
Test	07	07	12	07	07	12
Baseline	61.2	68.6	62.0	80.7	84.3	78.2
+ Task1	63.4	**71.3**	**64.5**	81.7	85.9	**80.5**
+ Task2	62.5	69.3	62.6	81.0	84.8	79.0
+ Task3	61.3	68.8	61.7	80.6	84.2	78.3
+ Task1,2	**63.9**	70.9	**64.5**	**81.8**	**86.1**	80.1
+ Task1,2,3	63.8	70.8	64.4	**81.8**	86.0	80.0

Table 4: Detection accuracies (mAP) with various backbone networks on VOC. Baseline is Faster R-CNN [39].

Training	07	07+12	
Test	07	07	12
Baseline	73.4	78.6	72.1
+ Task1	74.3	80.1	74.0
+ Task2	73.5	78.7	72.2
+ Task3	73.3	78.4	71.9
+ Task1,2	**75.0**	80.4	**74.2**
+ Task1,2,3	74.7	**80.6**	73.9

Table 5: Detection accuracies (mAP) on VOC. Baseline is R-FCN [10] with ResNet-101 [22] backbone.

Training	07	07+12	
Test	07	07	12
Baseline	77.0	81.7	75.3
+ MTL	78.0 (+1.0)	83.0 (+1.3)	76.7 (+1.4)
+ Refinement	78.3 (+1.3)	82.7 (+1.0)	76.4 (+1.1)
+ Both	**78.7 (+1.7)**	**83.7 (+2.0)**	**77.5 (+2.2)**

Table 6: Ablation results of multi-task learning and refinement on VOC. Baseline is Faster R-CNN with ResNet-101.

one per image. The task 3 is the worst among the auxiliary tasks since it is the simplest and may deliver the least information. Third, when using all three tasks jointly, the results are the best or closest to the best.

4.3. Ablation Experiments on Refinement

We perform an ablation study about the effect of refinement. In normal MTL, the outputs of auxiliary tasks do not directly refine the results of the main task. On the other hand, our auxiliary tasks can improve the classification of the main task, because they provide contextual information about the surroundings of RoIs. Table 6 shows how much detection accuracies are improved by the refinement. The mAP values increase by an average of 0.7, thanks to the effect of refinement when compared to using MTL only.

To further investigate the effect of refinement alone, we apply the *stop_gradient* to prevent the losses of auxiliary tasks from affecting the learning of the shared features. Its result is shown in the row (+Refinement) of Table 6. The mAP increases by 1.2 on average compared to the baseline, although the best performance is achieved with the feature learning together. These results assure that both feature learning by MTL and inference refinement are profitable.

4.4. Qualitative Results

Figure 4 shows some qualitative examples of detection improvement of our approach on VOC and COCO. In each set, we show the result of the baseline (upper) and our ap-

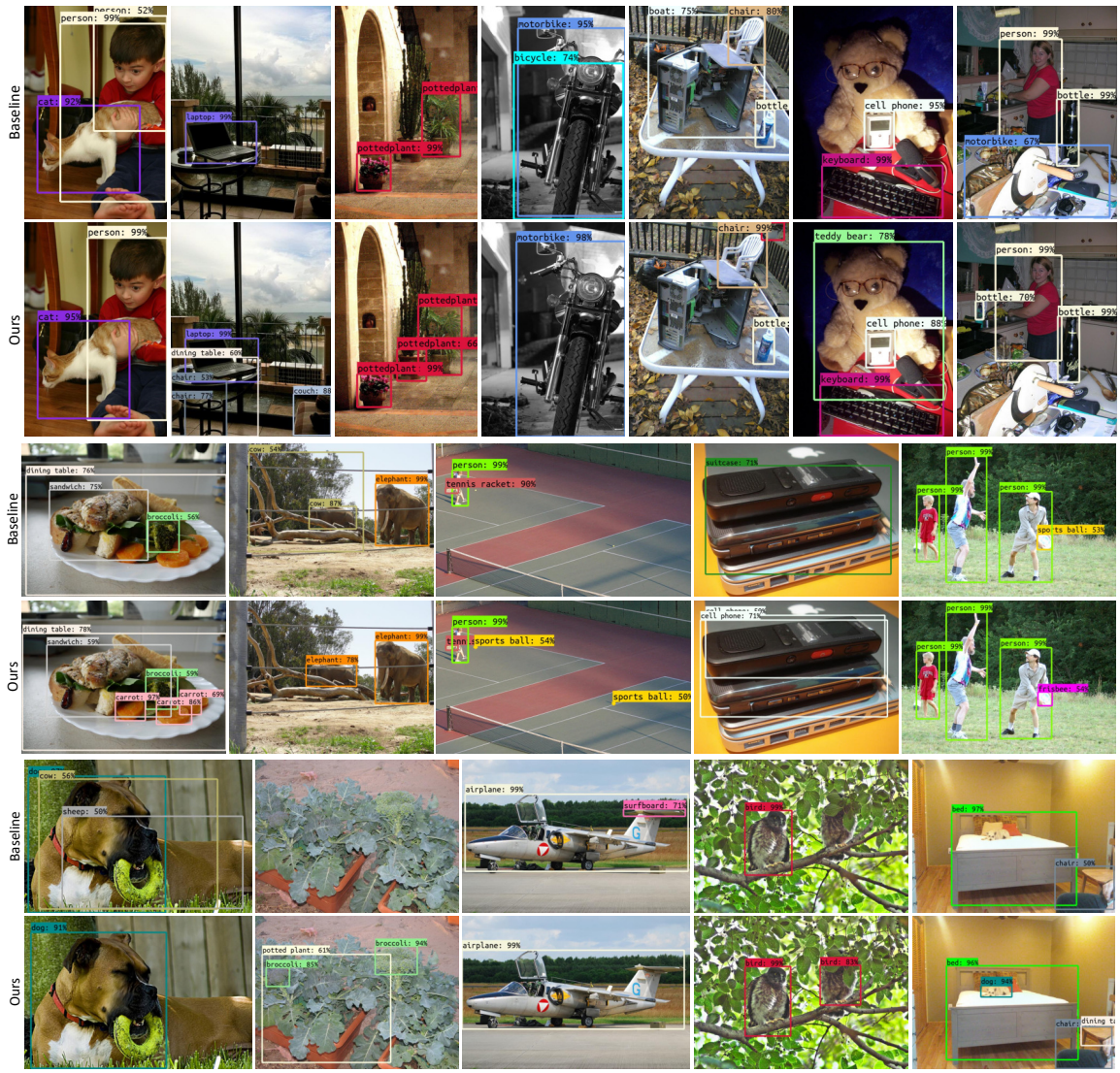

Figure 4: Comparison of detection between baseline (upper) and our approach (lower). Our approach improves the baseline's detection by correcting several false negatives and false positives such as background, similar object and redundant detection.

proach (lower). Our method is often able to correct several false negatives and false positives such as background, similar objects and redundant detection.

5. Conclusion

We proposed a novel multi-task self-supervised learning approach for object detection, where three auxiliary tasks were designed to improve the performance of object detection. They created their own labels by recycling the bounding box labels, and were jointly trained with the object detection model. Our experiments validate that our approach improves detection accuracies with various architectures and backbones. Our approach was helpful for detection regardless of the dataset size, as it achieved consistent improvement from small (VOC07, 25K objects) to large (COCO, 850K objects) datasets in our experiments.

There are several possible directions beyond this work. First, we could design auxiliary tasks that help box regression while this work dealt with only classification-enhancing tasks. Second, auxiliary tasks can be extended to recycle other labels such as segmentation masks for detection improvement. Lastly, we may verify our method in an extremely large dataset like OpenImages [28].

Acknowledgements. 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] A. H. Abdulnabi, G. Wang, J. Lu, and K. Jia. Multi-Task CNN Model for Attribute Prediction. *Multimedia*, 17(11):1949–1959, 2015. 2

[2] T. Almaev, B. Martinez, and M. Valstar. Learning to Transfer: Transferring Latent Task Structures and Its Application to Person-Specific Facial Action Unit Detection. In *ICCV*, 2015. 2

[3] J. Baxter. A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling. *Machine learning*, 28(1):7–39, 1997. 1, 2

[4] J. Cao, Y. Li, and Z. Zhang. Partially Shared Multi-Task Convolutional Neural Network with Local Constraint for Face Attribute Learning. In *CVPR*, 2018. 2

[5] R. Caruana. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *ICML*, 1993. 1, 2

[6] X. Chen and A. Gupta. Webly Supervised Learning of Convolutional Networks. In *ICCV*, 2015. 2

[7] X. Chen and A. Gupta. An Implementation of Faster RCNN with Study for Region Sampling. *arXiv:1702.02138*, 2017. 6

[8] X. Chu, W. Ouyang, W. Yang, and X. Wang. Multi-Task Recurrent Neural Network for Immediacy Prediction. In *ICCV*, 2015. 2

[9] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski. Self-supervised Monocular Road Detection in Desert Terrain. In *RSS*, 2006. 1, 2

[10] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In *NIPS*, 2016. 1, 2, 6, 7

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 5

[12] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised Visual Representation Learning by Context Prediction. In *ICCV*, 2015. 2

[13] C. Doersch and A. Zisserman. Multi-task Self-Supervised Visual Learning. In *ICCV*, 2017. 1, 2

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 1, 2, 6

[15] S. Fukuda, R. Yoshihashi, R. Kawakami, S. You, M. Iida, and T. Naemura. Cross-connected Networks for Multi-task Learning of Detection and Segmentation. *arXiv:1805.05569*, 2018. 2

[16] Y. Gao, Q. She, J. Ma, M. Zhao, W. Liu, and A. L. Yuille. NDDR-CNN: Layer-wise Feature Fusing in Multi-Task CNN by Neural Discriminative Dimensionality Reduction. *arXiv:1801.08297*, 2018. 2

[17] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 6

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *CVPR*, 2014. 5

[19] L. Gomez, Y. Patel, M. Rusinol, D. Karatzas, and C. V. Jawahar. Self-Supervised Learning of Visual Features Through Embedding Images Into Text Topic Spaces. In *CVPR*, 2017. 2

[20] K. Gong, X. Liang, X. Shen, and L. Lin. Look into Person: Self-supervised Structure-sensitive Learning and A New Benchmark for Human Parsing. In *CVPR*, 2017. 2

[21] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 1, 2, 4, 6, 7

[23] 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. 1, 2, 6, 7

[24] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/Accuracy Tradeoffs for Modern Convolutional Object Detectors. In *CVPR*, 2017. 6

[25] S. Jenni and P. Favaro. Self-Supervised Feature Learning by Learning to Spot Artifacts. In *CVPR*, 2018. 1, 2

[26] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning Visual Features from Large Weakly Supervised Data. In *ECCV*, 2016. 2

[27] K. Kumar Singh, F. Xiao, and Y. Jae Lee. Track and Transfer: Watching Videos to Simulate Strong Human Supervision for Weakly-Supervised Object Detection. In *CVPR*, 2016. 2

[28] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. In *arXiv:1811.00982*, 2018. 8

[29] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang. Unsupervised Representation Learning by Sorting Sequences. In *ICCV*, 2017. 2

[30] 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. 1, 2, 6

[31] M. Marsden, K. McGuinness, S. Little, and N. E. O'Connor. ResnetCrowd: A Residual Deep Learning Architecture for Crowd Counting, Violent Behaviour Detection and Crowd Density Level Classification. In *AVSS*, 2017. 2

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *ICLR Workshop*, 2013. 3

[33] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-Stitch Networks for Multi-Task Learning. In *CVPR*, 2016. 1, 2

[34] M. Noroozi and P. Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In *ECCV*, 2016. 1, 2

[35] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient Sound Provides Supervision for Visual Learning. In *ECCV*, 2016. 2

[36] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning Features by Watching Objects Move. In *CVPR*, 2017. 1, 2

[37] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context Encoders: Feature Learning by Inpainting. In *CVPR*, 2016. 2

[38] R. Ranjan, V. M. Patel, and R. Chellappa. HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition. *TPAMI*, 2017. 2

[39] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*, 2015. 1, 2, 3, 4, 5, 6, 7

[40] Z. Ren and Y. Jae Lee. Cross-Domain Self-Supervised Multi-Task Feature Learning Using Synthetic Imagery. In *CVPR*, 2018. 2

[41] S. Ruder, J. Bingel, I. Augenstein, and A. Sogaard. Learning What to Share between Loosely Related Tasks. *arXiv:1705.08142*, 2017. 2

[42] D. Stavens and S. Thrun. A Self-supervised Terrain Roughness Estimator for Offroad Autonomous Driving. In *UAI*, 2006. 1, 2

[43] C. Su, F. Yang, S. Zhang, Q. Tian, L. S. Davis, and W. Gao. Multi-Task Learning With Low Rank Attribute Embedding for Person Re-Identification. In *ICCV*, 2015. 2

[44] O. Sumer, T. Dencker, and B. Ommer. Self-supervised Learning of Pose Embeddings from Spatiotemporal Relations in Videos. In *ICCV*, 2017. 2

[45] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017. 1, 2, 6, 7

[46] L. Trottier, P. Giguère, and B. Chaib-draa. Multi-Task Learning by Deep Collaboration and Application in Facial Landmark Detection. *arXiv:1711.00111*, 2017. 1, 2

[47] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An Uncertain Future: Forecasting from Static Images Using Variational Autoencoders. In *ECCV*, 2016. 2

[48] X. Wang, K. He, and A. Gupta. Transitive Invariance for Self-supervised Visual Representation Learning. In *ICCV*, 2017. 2

[49] D. Xu, W. Ouyang, X. Wang, and N. Sebe. PAD-Net: Multi-Tasks Guided Prediction-and-Distillation Network for Simultaneous Depth Estimation and Scene Parsing. In *CVPR*, 2018. 2

[50] J. Yim, H. Jung, B. Yoo, C. Choi, D. Park, and J. Kim. Rotating Your Face Using Multi-Task Deep Neural Network. In *CVPR*, 2015. 2

[51] X. Zhan, Z. Liu, P. Luo, X. Tang, and C. C. Loy. Mix-and-Match Tuning for Self-Supervised Semantic Segmentation. In *AAAI*, 2018. 2

[52] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond Empirical Risk Minimization. In *ICLR*, 2018. 3

[53] R. Zhang, P. Isola, and A. A. Efros. Colorful Image Colorization. In *ECCV*, 2016. 1, 2

[54] R. Zhang, P. Isola, and A. A. Efros. Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction. In *CVPR*, 2017. 2

[55] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial Landmark Detection by Deep Multi-task Learning. In *ECCV*, 2014. 2