Cheaper Pre-training Lunch: An Efficient Paradigm for Object Detection

Dongzhan Zhou[1]*, Xinchi Zhou[1]*, Hongwen Zhang[2], Shuai Yi[3], and Wanli Ouyang[1]

[1] The University of Sydney, SenseTime Computer Vision Research Group, Australia
{d.zhou,xinchi.zhou1,wanli.ouyang}@sydney.edu.au
[2] Institute of Automation, Chinese Academy of Sciences & University of Chinese Academy of Sciences
hongwen.zhang@cripac.ia.ac.cn
[3] Sensetime Research
yishuai@sensetime.com

Abstract. In this paper, we propose a general and efficient pre-training paradigm, Jigsaw pre-training, for object detection. Jigsaw pre-training needs only the target detection dataset while taking only 1/4 computational resources compared to the widely adopted ImageNet pre-training. To build such an efficient paradigm, we reduce the potential redundancy by carefully extracting useful samples from the original images, assembling samples in a Jigsaw manner as input, and using an ERF-adaptive dense classification strategy for model pre-training. These designs include not only a new input pattern to improve the spatial utilization but also a novel learning objective to expand the effective receptive field of the pre-trained model. The efficiency and superiority of Jigsaw pre-training are validated by extensive experiments on the MS-COCO dataset, where the results indicate that the models using Jigsaw pre-training are able to achieve on-par or even better detection performances compared with the ImageNet pre-trained counterparts.

Keywords: Pre-training, Object Detection, Acceleration

1 Introduction

Pre-training on the classification dataset (*e.g.*, ImageNet [9]) is a common practice to achieve better network initialization for object detection. Under this paradigm, deep networks benefit from useful feature representations learned from large-scale data, which promotes the convergence of models during fine-tuning stage. Despite the benefits, the burdens caused by extra data should not be neglected.

Previous works [25,6,37] have proposed alternative solutions to directly train detection models from scratch with random initialization. However, there is always no free lunch. Training from scratch suffers from slower convergence,

* Co-first author.

namely, additional training iterations are needed to obtain competitive models. *Can we incorporate the merit of fast convergence via pre-training without paying for the extra data or expensive training cost?*

The answer is **Yes**. We find the cheaper lunch for pre-training. In this work, we propose a new pre-training paradigm, Jigsaw pre-training, which is based only on the detection dataset. Compared with ImageNet pre-training, Jigsaw pre-training takes only 1/4 computational resources without extra data while achieving on-par or even better performance on the target object detection task.

Jigsaw pre-training is built upon the observation that a large number of pixels seen by the model during naive training are invalid or less informative, *i.e.*, most pixels/neurons in background regions would not fire during the learning process. Those excessive background pixels inevitably lead to redundant computational costs. To tackle this issue, we carefully extract positive and negative samples from original images in the detection dataset for pre-training. Before being fed into the backbone network, these samples will be assembled in a Jigsaw manner in consideration of their aspect ratios to improve the spatial utilization. To further improve the pixel level utilization, we design an ERF-adaptive dense classification strategy to leverage the Effective Receptive Field (ERF) via assigning soft labels in the learning objective. Our Jigsaw pre-training largely takes every pixel seen by the model into account, which greatly reduces the redundancy and provides an efficient and general pre-training solution for object detection.

Our major contributions can be summarized as follows.

(1) We propose an efficient and general pre-training paradigm based only on detection dataset, which eliminates the burdens of additional data.

(2) We design rules of sample extraction, the Jigsaw assembly strategy, and the ERF-adaptive dense classification for efficient pre-training, which largely considers the network utilization and improves the learning efficiency and final performance.

(3) We validate the effectiveness of our Jigsaw pre-training on various detection frameworks and backbones and demonstrate the versatility of the proposed pre-training strategy. We hope this work would inspire more discussions about the pre-training of object detectors.

2 Related Work

Classification-based Pre-training for Object Detector. Various detection frameworks [4,14,23,7,16,36,30] follow the standard 'pre-training followed by fine-tuning' training procedure, where networks are first pre-trained on the large-scale dataset (*e.g.*, ImageNet [9]) and then fine-tuned on the target detection dataset. This pre-training paradigm is mainly classification-based and aims to learn strong or universal representations, which speed up the convergence of detection models. Many efforts have been devoted to push the boundary of transferability further through different learning modes such as supervised [8], weakly supervised [18,32], unsupervised [5] learning, or exploiting larger scale training data such as Instagram-17k [18] and JFT-300M [28]. Despite the improvements for transferability, the corresponding expensive training cost of large scale data should not be neglected. Our Jigsaw pre-training is entirely based on detection dataset which eliminates the burden of using external data. Meanwhile, the pre-training process is 4× faster than ImageNet-1k classification training.

Redundancy in Object Detector. Sample imbalance is a common source of redundancy for object detection, where many background pixels belonging to easy negative samples contribute no useful information for training. To alleviate this issue, several attempts have been made to improve the efficiency of detection training. OHEM [26] tries to solve the imbalance sampling by discarding easy negative samples. Focal loss [14] adopts a weighting factor to reduce loss weight for easy samples. Chen *et al.* propose a more reasonable method for sample evaluation in [1]. Libra R-CNN [20] proposes the IoU-balanced sampling to augment the hard cases. SNIPER [27] reduces the calculation burden of multi-scale training by only training on selected chips rather than the entire images. All these works mainly focus on the efficiency and performance within detection frameworks, but they still provide inspirations on sample selection in our work. By carefully selecting positive and negative samples for pre-training, the redundancy is significantly reduced, which eventually speeds up the classification pre-training process.

Object Detector Trained from Scratch. Many works [19,29,25,10,12,6,37] have proposed another possible training paradigm which is to train the detector from scratch. For instance, DSOD [25] is motivated by designing a pre-training free detector, but limited to the structure they designed. CornerNet [10] and DetNet [12] present the results of their models trained from scratch. These efforts indicate that pre-training might be unnecessary when adequate data is available. Furthermore, doubts on ImageNet pre-training are also raised recently. He *et al.* [6] and Zhu *et al.* [37] suggest that ImageNet pre-training might be a historical workaround. However, although these solutions get rid of the burdens for large-scale external data, the random initialized detection models suffer from the problem of low convergence speed, which comes at the cost of extending training iterations by 4-5 times to obtain competitive models. Inspired by these works, we move steps forward to exploit an efficient pre-training paradigm for pre-training on detection data, which takes the advantages of both fast convergence and no extra data at the same time.

3 Methodology

The pipeline of using the proposed Jigsaw pre-training scheme is shown in Fig. 1. Given a detection dataset $\mathcal{D}$, positive and negative samples will be extracted from the images of $\mathcal{D}$ and saved as classification dataset beforehand (Sec. 3.1). These samples will be assembled in a Jigsaw manner (Sec. 3.2) and fed into the detector backbone for pre-training, where an ERF-adaptive loss is used as the loss function (Sec. 3.3). After pre-training, the object detector will be fine-tuned on $\mathcal{D}$ under the detection task. Note that our pre-training scheme is flexible and

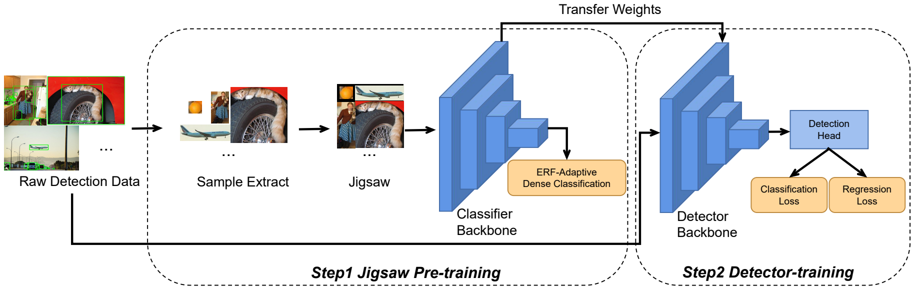


Fig. 1. Pipeline of the proposed Jigsaw pre-training scheme. Firstly, we will extract positive and negative samples from detection data to build classification dataset. The pre-training process is conducted under Jigsaw assembly manner and ERF-adaptive loss. Finally, the backbones will be fine-tuned on target detection task.

can be applied to object detectors with diverse detection head and backbone architectures.

3.1 Sample Selection

As demonstrated in previous works [26,20], balanced sample selection is critical during the training of object detectors. For efficient pre-training, we carefully select regions extracted from original images as positive and negative samples, which will be further assembled and fed into the detector backbone. The positive samples are regions that should be classified as one of the C foreground categories in the detection dataset, while the negative samples are background regions. To effectively select diverse and important samples, we set up several rules for the sample extraction.

Positive samples. To generate positive samples, we extract regions from the original images according to the ground-truth bounding boxes. Considering that contextual information is beneficial to learn better feature representations [3] [35], we randomly enlarge the bounding boxes so that more context is involved in the positive samples. Specifically, the upper-left and bottom-right corner points of the ground-truth bounding boxes randomly move outwards so that the width/height could be up to 2× of the original width/height. Enlarged bounding boxes beyond original images will be truncated at the edges. Fig. 2 depicts the generation of positive samples. Experimental results indicate that adding context to the positive samples during pre-training will bring 1.6% mAP gain for Faster R-CNN [24] with ResNet50 backbone.

Negative samples. To make the pre-trained models more adaptive to detection scenario, we also add some negative samples generated from background regions to the classification dataset. Firstly, we randomly generate some candidate proposals. After that, we require that all negative samples meet the requirement $IoU\left(pos, neg\right) = 0$, where IoU indicates Intersection-over-Union. In this way,

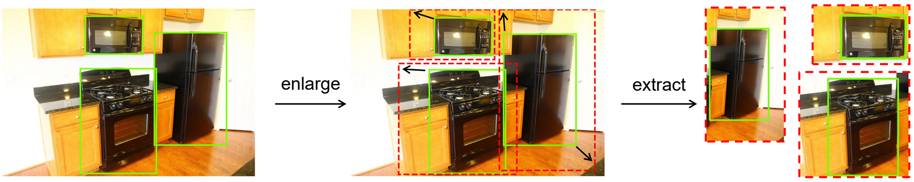


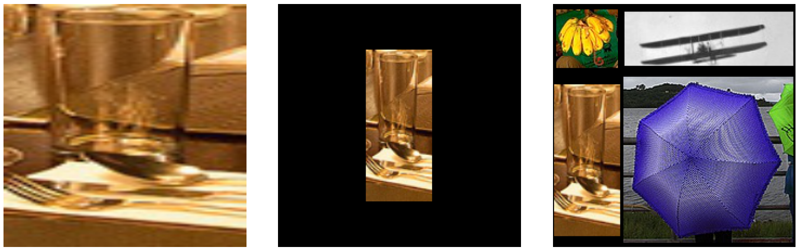
Fig. 2. The process for generating positive samples. First, we will use ground-truth bounding boxes to locate target regions. Then the regions are randomly enlarged to incorporate context information. Finally the enlarged regions are extracted from the original images as positive samples for pre-training. The green solid lines stand for GT-bounding boxes and red dash lines for enlarged boxes.

negative samples are exclusive with those positive ones. In our pre-training experiments, the ratio of the number of positive samples to negative ones is 10 : 1.

3.2 Jigsaw Assembly

There are different ways to assemble samples and feed them into the backbone for pre-training. Two straightforward assembling methods are warping (method 1) or padding (method 2) a sample to a pre-defined input size, *e.g.*, 224 × 224. However, forcing all samples to be warped to the same size may destroy the texture information and distort the original shapes, while padding would introduce many uninformative padded pixels and hence bring additional costs in both training time and computational resources. These two straightforward methods are either harmful or wasteful for the pre-training process. For more efficient pre-training, we propose to assemble samples in a Jigsaw manner in consideration of the scale and aspect ratio of objects. Specifically, four samples will be stitched into a new image and then taken as input for pre-training.

As depicted in Fig. 3, compared to warping and padding, our Jigsaw assembly can not only preserve original texture information but also eliminate the uninformative padded pixels.



(a) Warping (b) Padding (c) Jigsaw

Fig. 3. Different methods to adjust sample to pre-defined input size. (a) Warping distorts the original shape or texture. (b) Padding introduces many uninformative pixels. (c) Jigsaw preserves original information while improving space utilization.

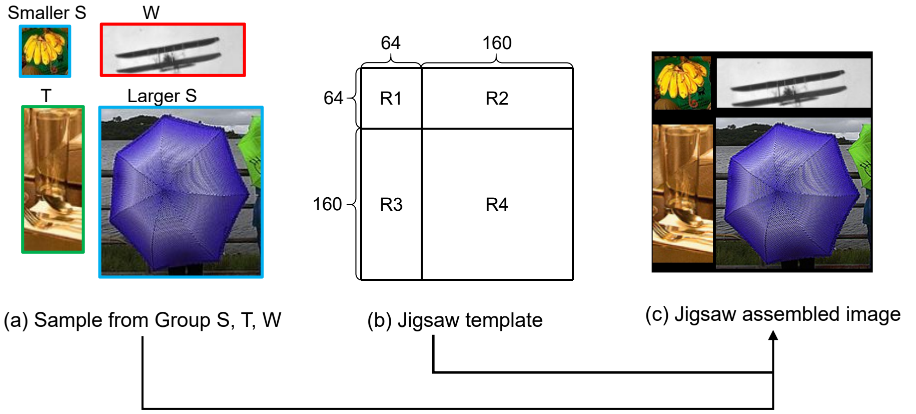
Objects vary in aspect ratio. Jigsaw assembly takes this property into consideration so that samples could be stitched together more naturally according to their aspect ratios. To this end, samples will be first divided into three Groups according to their aspect ratios, *i.e.*, Group S (square), T (tall), and W (wide). Samples in Group S should have the aspect ratios between 0.5 and 1.5, while samples in Group T and W should respectively have aspect ratio smaller than 0.5 and larger than 1.5. For simplicity, samples from Group S, T, and W are referred to as S-samples, T-samples, and W-samples, respectively.

As shown in Fig. 4, for every Jigsaw assembled image, 2 S-samples, 1 T-sample, and 1 W-sample will be selected randomly from above three groups and stitched into four regions accordingly. Specifically, the S-sample with smaller bounding box area is at the top-left region, while the larger S-sample is at the bottom-right region. The T-sample and W-sample will be respectively assigned to bottom-left and top-right regions.

Pad & Crop vs Warp vs Resize. If the sizes of samples do not match the pre-defined sizes of regions in the new image, we apply padding or random cropping to the samples, which is conditioned on whether their sizes are smaller or larger than the pre-defined ones. Note that we do not warp or resize the samples, as the experimental results in Table 1 show that such operations will degrade the final performance. Visualization examples of the three operations can be found in the supplementary.

3.3 ERF-adaptive Dense Classification

During pre-training, the Jigsaw assembled images will be fed into the backbone network to obtain the feature maps $\mathbf{X} \in \mathbb{R}^{C\times\alpha H\times\alpha W}$ before the final average pooling. Here we omit the number of samples in $\mathbf{X}$ for simplicity. Compared to the conventional classification pre-training, Jigsaw pre-training should have different learning strategy since there are four samples stitched in one assembled



(a) Sample from Group S, T, W

(b) Jigsaw template

(c) Jigsaw assembled image

Fig. 4. Pipeline for Jigsaw assembled image generation. We first randomly select 2 S-samples, 1 T-sample and 1 W-sample, respectively as shown in (a), then assemble the samples according to the template (b) and get an assembled image (c). The numbers on Jigsaw template denote the height/width of each region.

Method	AP	AP_{50}	AP_{75}
Warp	34.6	54.7	36.8
Resize	34.7	54.7	37.1
Pad&Crop	**35.2**	**55.7**	**37.6**

Table 1. Comparison of Warp, Resize and Pad & Crop for scale adjustment during the pre-training process. The network structure is ResNet50. The three pre-trained models are used for the subsequent detection training of Faster R-CNN[24] and results are evaluated on COCO val2017. The results show that pad & crop is more helpful for obtaining better pre-trained models. For the results of 'Warp', we change both the size and aspect ratio while for those of 'Resize', we only change the size.

image. In the following, we discuss two alternative strategies and then introduce our proposed ERF-adaptive Dense Classification.

Global classification. As shown in Fig. 4, an image contains four objects in our Jigsaw assembled image. As an intuitive strategy, we can assign the whole image a single global label, which is the weighted sum of the labels of the four objects according to their region areas. This strategy could be reminiscent of the CutMix [34], where certain region of the original image will be replaced by a patch from another image and the corresponding label will also be mixed proportionally with the label of the new patch. The visualization of global classification will be provided in the supplementary material.

Block-wise classification. Another intuitive strategy would perform individually for each block/region, that is, the average pooling is independently applied to the four blocks of feature maps $\mathbf{X}$ corresponding to four samples, followed by individual classification according to the label of each sample. However, these two

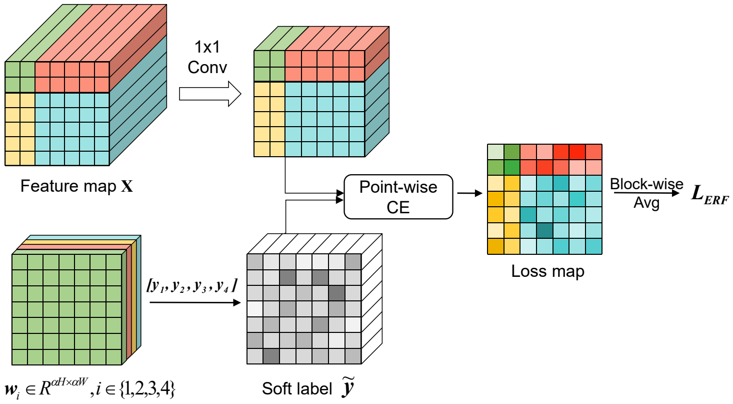


Fig. 5. Process of our Dense Classification Strategy. We use different colors to distinguish different regions, *e.g.*, green for R_1, and the brightness difference in soft label and loss map represents different values. The feature map $\mathbf{X}$ is convolved by a 1×1 kernel to reduce the number of channels to C (category number). Given the weight w_i of label $\mathbf{y_i}$, we obtain the soft label tensor for each point. Then the cross-entropy loss is densely imposed on the feature map and we will get the loss value at each point (denoted as loss map). After that, block-wise average is exerted on the loss map to generate average losses for each region. The final ERF-adaptive loss is the mean of four region losses. Best viewed in color.

intuitive strategies confine the learning of each block to the corresponding sample. As can be seen in Fig. 6(a) and 6(b), the Effective Receptive Field (ERF) [17] of the top-left region in $\mathbf{X}$ mainly concentrates on the area of the corresponding smaller S-sample. The confined receptive field may empircally degrade the performance of deep models, as illustrated in [15,11,22]. The visualization of block-wise classification will be provided in the supplementary material.

Our strategy. To largely take every seen pixel into account, we propose an ERF-adaptive Dense Classification strategy to perform classification for each position at $\mathbf{X}$, where its soft labels are computed based on the corresponding effective receptive field. The process is depicted in Fig. 5.

Specifically, for the four regions in the Jigsaw template as shown in Fig. 4(b), we denote $\mathbf{y}_i$ as the original label for the region R_i, $i = 1, 2, 3, 4$.

At the position (j, k) of feature map $\mathbf{X}$ ($j = 1, \ldots, \alpha H, k = 1, \ldots, \alpha W$), the soft label $\tilde{\mathbf{y}}^{j,k}$ is the weighted sum of four labels:

$$\tilde{\mathbf{y}}^{j,k} = \sum_{i=1}^{4} w_i^{j,k} \mathbf{y}_i, \tag{1}$$

where the weight $w_i^{j,k}$ is dependent on its ERF. At the position (j, k) of feature map $\mathbf{X}$ ($j = 1, \ldots, \alpha H, k = 1, \ldots, \alpha W$), we obtain the corresponding ERF $\mathbf{G}^{j,k} \in \mathbb{R}^{H \times W}$ on the input space. Then, the weight $w_i^{j,k}$ for the label $\mathbf{y}_i$ at position (j, k) should be proportional to the ratio of the summed activation within the region i to the whole summed activation. Moreover, if the position (j,k) is in region i, we empirically set a threshold τ for $w_i^{j,k}$ to make sure that $\mathbf{y}_i$ is dominant at region i. Hence, for the position (j,k) at region r, we have the weight $w_i^{j,k}$ of the label $\mathbf{y}_i$ as follows:

$$w_i^{j,k} = \begin{cases} \max(\tau, \frac{\sum_{h=1,w=1}^{H,W} \mathbf{G}_{h,w}^{j,k} \cdot \mathbf{M}_{h,w}^{i}}{\sum_{h=1,w=1}^{H,W} \mathbf{G}_{h,w}^{j,k}}), \text{if } i = r, \\ (1 - w_r^{j,k}) \frac{\sum_{h=1,w=1}^{H,W} \mathbf{G}_{h,w}^{j,k} \cdot \mathbf{M}_{h,w}^{i}}{\sum_{h=1,w=1}^{H,W} \mathbf{G}_{h,w}^{j,k} \cdot (1 - \mathbf{M}_{h,w}^{r})}, \text{if } i \neq r, \end{cases} \tag{2}$$

where $\mathbf{M}^i \in \{0,1\}^{H \times W}$ denotes the binary mask selecting the region i in the ERF.

Denote $\mathbf{x}^{j,k} \in \mathbb{R}^C$ as the features at the position (j,k) of $\mathbf{X}$ $(j = 1, \ldots, \alpha H, k = 1, \ldots, \alpha W)$. After obtaining the weights $\{w_i^{j,k}\}_{i=1}^4$, we perform dense classification upon the feature $\mathbf{x}^{j,k}$, where its soft label $\tilde{\mathbf{y}}^{j,k}$ is defined in Eq. (1). In our implementation, the final fully connected layer is replaced by a 1×1 convolution layer and the cross-entropy loss is imposed on the category prediction at every position. To make a balance among different regions, the final ERF-adaptive loss is the block-wise average of the loss map, as the last step in Fig. 5. We also need to clarify that the weights of soft label Eq. (1) are updated at every 5k iterations instead of at each iteration. Thus, even if dense classification is adopted, its effect on training time is negligible.

The effective receptive field of the top-left region for the different pre-training strategies is visualized in Fig. 6. Our strategy in Fig. 6(c) has the largest ERF among the above three strategies.

Relationships among Different Strategies. The above three strategies perform classification at different scale levels, where the proposed ERF-adaptive classification is the most fine-grained one while the global classification is the coarsest one. Compared with the other two alternative strategies, the proposed one has different soft labels for each position at $\mathbf{X}$. The ERF-adaptive dense classification would be equivalent to the block-wise classification with threshold τ set to 1. The block-wise classification would be also equivalent to the global classification if the region losses are re-weighted in a CutMix manner. Under different label assignment strategies, the pre-trained model has different pixel level utilization and hence behaves differently. As reported in Table 2, the performance of the model pre-trained under the proposed strategy is best among three strategies.

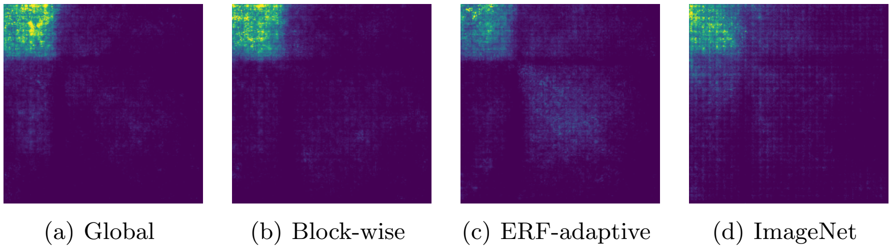


Fig. 6. Visualization of Effective Receptive Field of the top-left region for different pre-trained models. (a) Global represents conducting global average pooling and set global label as weighted sum of labels from four regions. (b) Block-wise refers to the intuitive strategy which performs classification individually on each region. (c) ERF-adaptive refers to adopting ERF-adaptive dense classification. (d) ImageNet stands for the ImageNet pretrained model officially provided by PyTorch [21].

Strategy	AP	AP_{50}	AP_{75}
Global Cls.	34.3	54.5	36.3
Block-wise Cls.	35.2	55.7	37.6
ERF-adaptive Dense Cls.	**36.3**	**56.5**	**38.9**

Table 2. Comparison of different classification strategies. The backbone CNN is ResNet50 and detection framework is Faster R-CNN[24]. All results are evaluated on COCO val2017. The results show that the ERF-adaptive dense classification strategy clearly outperforms the other two strategies.

4 Experiments

4.1 Implementation details

This section introduces the implementation details of the classification pre-training and detector training.

Pre-training Settings. Unless otherwise specified, the models are pre-trained for 64k iterations on 8 Tesla V100 GPUs with the total batch size of 512. Note that the batch size 512 is for Jigsaw assembled images, so the total number of individual samples in each batch is 2048 (an assembled image consists of 4 samples). Warm-up is used during the first 1250 iterations, where the learning rate starts from 0.2 and then linearly increases to 0.8. Afterwards, the learning rate decreases to 0.0 following a cosine scheduler. The weight decay is 1e−4. We update weights $w_i^{j,k}$ of soft labels in Eq. 2 for every 5k iterations. The threshold τ in Eq. 2 for ERF-adaptive classification is set to 0.7.

Pre-training Augmentation. The samples will first be resized according to a resize ratio chosen randomly from $[0.8, 1.5]$. Both the height and width will be adjusted by the same ratio so that its aspect ratio keeps unchanged. Random horizontal flip with probability 0.5 is also applied on each sample before being assembled into the new image. During Jigsaw assembly, random cropping or zero padding is used to adjust the samples to the pre-defined sizes. After the stitching, the channels of assembled image are normalized with mean $[0.485, 0.456, 0.406]$ and std $[0.229, 0.224, 0.225]$.
Training Details of Detectors. For fair comparisons, we adopt the same training settings on detection for both ImageNet pre-trained models and Jigsaw pre-trained models. We train our models on MS-COCO train2017 split. If not specified, all models are trained for 13 epochs on 8 Tesla V100 GPUs with total batch size 16. We use SGD as the optimizer with momentum 0.9 and weight decay 0.0001. The initial learning rate is 0.02 and decreases by factor 0.1 at epoch 9 and 12. The batch normalization layers are frozen during training. The images are resized to 1333 × 800 and randomly flipped with probability 0.5 for augmentation.

4.2 Main Results

We conduct the Jigsaw pre-training process based on samples extracted from MS-COCO train2017 split, and fine-tune the detection models on the same dataset. The backbone is ResNet-50. Note that the Jigsaw pre-training process only consumes **1/4** computation resources compared with ImageNet pre-training. As reported in Table 3, the results show that the models using our Jigsaw pre-training strategy are able to achieve on-par or even better performances compared with the ImageNet pre-training counterparts for various detection frameworks. For original Faster R-CNN [24], the AP increases from 34.8% to 36.3% (+1.5%), for Faster R-CNN with FPN [13], AP increases from 36.2% to 36.5% (+0.3%), for Mask R-CNN with FPN [7], AP increases from 37.2% to 37.4%(+0.2%).

We notice that the improvement is most significant in the original Faster R-CNN structure (denoted as C4 in Table 3). We suspect the possible reason is that, compared with FPN structure, the backbone accounts for a larger proportion in C4. In other words, for detection models with FPN structure, the lateral connections and entire structures at the second stage will be randomly initialized without being transferred from pre-trained model. But for the original Faster R-CNN, the main part of the second stage is still transferred from the pre-trained network. We speculate that the improvement of detection models with FPN may be consistent with that of C4 if the FPN structure is incorporated into pre-training, and we will leave this exploration for future work.

4.3 Ablation study

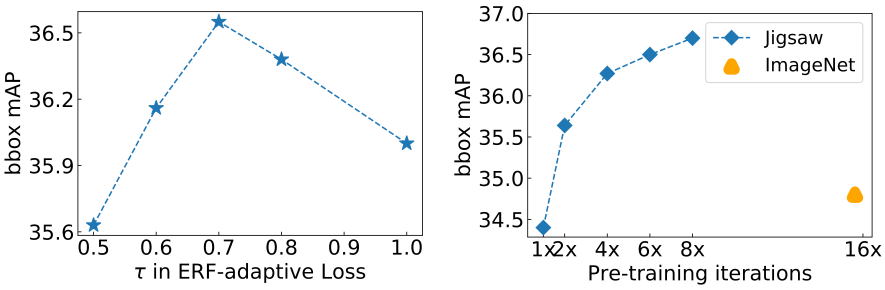
Threshold for ERF-adaptive Dense Classification. When ERF-adaptive Dense Classification is used, there is a threshold τ in Eq. (1) to make sure that the original label y_i is dominant at its corresponding region i. We explore

Method	Cost	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
C4[24] + ImageNet	6.80	34.8	55.5	36.8	18.3	38.7	48.4
C4[24] + Jigsaw	1.73	36.3	56.5	38.9	18.9	40.8	49.7
Δ	-5.07	+1.5	+1.0	+2.1	+0.6	+2.1	+1.3
FPN[13] + ImageNet	6.80	36.2	58.0	39.2	21.2	39.9	45.6
FPN[13] + Jigsaw	1.73	36.5	58.3	39.2	22.2	40.4	45.8
Δ	-5.07	+0.3	+0.3	0.0	+1.0	+0.5	+0.2
Mask[7] + ImageNet	6.80	37.3	59.0	40.3	21.9	40.6	46.2
Mask[7] + Jigsaw	1.73	37.5	58.9	40.6	22.8	41.2	46.9
Δ	-5.07	+0.2	-0.1	+0.3	+0.9	+0.6	+0.7

Table 3. Results on different detection frameworks with backbone ResNet-50. The cost refers to pre-training cost and the unit is GPU days. The AP results are evaluated on COCO val2017. 'C4' denotes original Faster R-CNN without FPN [24], 'FPN' denotes Faster R-CNN with FPN [13], 'Mask' denotes Mask R-CNN with FPN [7]. '+ ImageNet' means the backbone is pre-trained on ImageNet dataset. '+ Jigsaw' denotes that the backbone is pre-trained with our Jigsaw strategy. Δ measures the difference in absolute AP or cost between adopting Jigsaw and ImageNet pre-trained backbones, respectively.

the effects of this threshold and the results are depicted in Fig. 7(a). Although using mixed labels is beneficial, relatively low proportion of original label (*e.g.*, 0.5) may still hinder the pre-training. As the threshold becomes higher, the loss is gradually approaching the use of single hard label for each point, which may suffer from relatively confined receptive field, as analyzed in Section 3.3. Therefore, it is important to choose proper threshold and we find 0.7 is an ideal choice. Fig. 7(a) also shows that setting the threshold in [0.6 0.8] will not cause much variation in mAP. Therefore, the experimental results are not so sensitive to this hyper-parameter.

Iterations for pre-training. We also investigate the influences of changing the pre-training iterations and visualize the results in Fig. 7(b). Naturally, increasing training iterations will provide better pre-trained models, which leads to better detection performance. But we also observe that the gains from longer iterations are not so significant after 64k iterations (4× in Fig. 7(b)). Considering the trade-off between performance and computation, we choose to train 64k iterations during pre-training, which consumes only 1/4 computation resources but achieve 1.5% higher mAP compared with ImageNet pre-training.



(a) (b)

Fig. 7. (a) Trend of bbox mAP with different τ settings in ERF-adaptive loss. (b) bbox mAP for different pre-training iterations, the values on x-axis stand for multiple of 16k iterations. Our pre-training approach uses 4× as the default setting, which requires 1/4 the number of iterations but achieves 1.5% higher mAP when compared with ImageNet counterpart. The results are evaluated on COCO val2017.

Different backbone structures. We also implement our pre-training strategy on different backbone structures to evaluate the versatility. The results in Table 4 show that Jigsaw pre-training strategy does not rely on specific network structures but will consistently keep on-par performance or obtain improvements.

Method	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Resnet101 + ImageNet	38.3	58.9	41.1	20.0	42.8	53.0
Resnet101 + Jigsaw	39.2	59.6	42.0	20.6	43.3	54.8
Δ	+0.9	+0.7	+0.9	+0.6	+0.5	+1.8
X101-32x4d + ImageNet	40.2	61.2	43.2	21.2	44.6	55.7
X101-32x4d + Jigsaw	40.2	61.0	43.0	21.3	44.5	55.7
Δ	0.0	-0.2	-0.2	+0.1	-0.1	0.0

Table 4. Results for different backbone structures evaluated on COCO val2017. The detection framework is original Faster R-CNN. ImageNet means the backbone is trained on ImageNet dataset. Jigsaw denotes that the backbone is pre-trained with Jigsaw strategy. X101-32x4d refers to ResNeXt101-32x4d [33].

4.4 Compatibility to other designs

We also examine the compatibility of Jigsaw pre-training strategy with commonly used designs in object detection, including longer training iterations (2x schedule), deformable convolution [2], multi-scale augmentation, etc. The results in Fig. 8 indicate that Jigsaw pre-training can still achieve comparable or even higher performance even on various enhanced baselines.

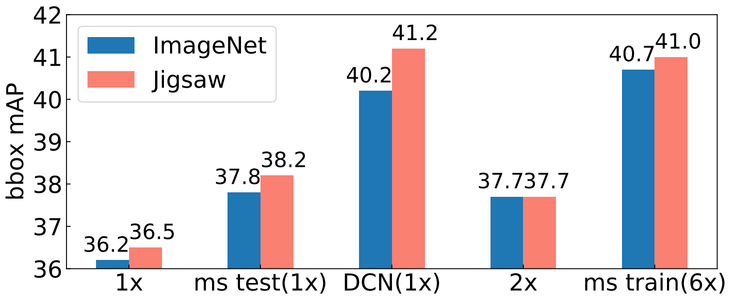


Fig. 8. Comparison between **ImageNet** pre-training and **Jigsaw** pre-training of various strategies on Faster R-CNN FPN framework with ResNet-50 backbone, the results are evaluated on COCO val2017. Strategies include: (1) 1×: serving as original strategy that train for 1× schedule (13 epochs), (2) ms test: adding multi-scale augmentation during test stage, (3) DCN: replace the 3×3 convolution layers of stage 2-4 in backbone with 3×3 deformable convolution layer [2], (4) 2×: extending the training time to 2× schedule, (5) ms train: implementing multi-scale augmentation during train and test stage and extending training epochs to 6× schedule.

It is worth noting that the most obvious improvement has been achieved when replacing some convolution layers to deformable convolution layers. We suspect that this improvement may come from the relief of domain shift between pre-training dataset and detection dataset. Therefore, our approach has the potential of further boosting the performance gains from new designs on backbones.

4.5 Comparison with training from scratch method

We also compare our Jigsaw pre-training strategy with the training detection from scratch method (denoted as scratch for simplicity). Scratch and our strategy share an advantage that the entire training process is only based on detection dataset without introducing any external data. However, adopting pre-training process will speed up the convergence of detection models, which helps the models achieve better performance under common training iterations, such as 1× or 2× schedules. The results are presented in Table 5. To make a fair comparison, we keep total training costs similar for the two methods, that is, the total costs in our method include both pre-training and detection training consumptions. We follow the experimental settings for scratch in [6] where all batch norm layers in the network are replaced by group norm [31]. The batch norm layers are frozen at detection stage when transferring from our pre-trained backbones. The results indicate that even with group normalization, which is proven to improve the performance of detection models, training from scratch still shows suboptimal performance compared with our pre-training strategy.

Method	Total Cost (GPU days)	AP	AP_{50}	AP_{75}
Scratch	6.0	28.6	46.5	30.1
Jigsaw + 1×	5.8	36.3	56.5	38.9
Scratch	9.5	28.9	47.1	30.5
Jigsaw + 2×	9.2	37.5	57.6	40.7

Table 5. Comparison of scratch and Jigsaw pre-training under similar computation costs. The detection framework is Faster R-CNN with backbone ResNet-50, and the AP results are evaluated on COCO val2017. 1× refer to training detection models for widely adopted 1× schedule, while 2× to extend the iterations to twice.

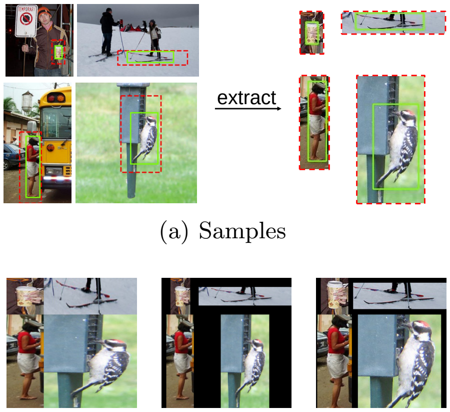
5 Conclusion

In this work, we present a choice to obtain cheaper lunch on pre-training for object detection, which is able to reduce the consumption of pre-training to 1/4 compared with the original ImageNet pre-training, while achieving on-par or even higher performance. We define a novel pre-training paradigm based only on detection dataset, which eliminates the burdens of extra training data while retaining the advantage of fast convergence. Under the consideration that many pixels in detection images only provide negligible information, such as easy background, we carefully extract positive and negative regions from detection images to build our pre-training dataset, which effectively raises the proportion of useful pixels. Moreover, based on the observation that the extracted regions vary in both scales and aspect ratios, we propose a sample stitching method, Jigsaw Assembly strategy, which further reduces the amount of meaningless pixels and improves space utilization. We also design ERF-adaptive dense classification strategy, which boosts the performance of pre-trained models via expanding the receptive field. We expect this work would help researchers reduce the trial-and-error cost, inspire more future research on pre-training process, and facilitate new backbone CNN architecture design/search tailored for object detection.

Appendix

A Visualization examples of adjusting sample size

This part provides visual examples of three operations to adjust sample size. The examples are depicted in Fig. A1. From the examples, we can see that resize would result in too many pixels being uninformative and warping will distort the image. Crop is able to retain the shape and preserve more information, which makes it a better choice for size adjustment. Results in Table 1 also indicate that warp and resize lead to suboptimal performances compared with crop.



(a) Samples

(b) Warp (c) Resize (d) Crop

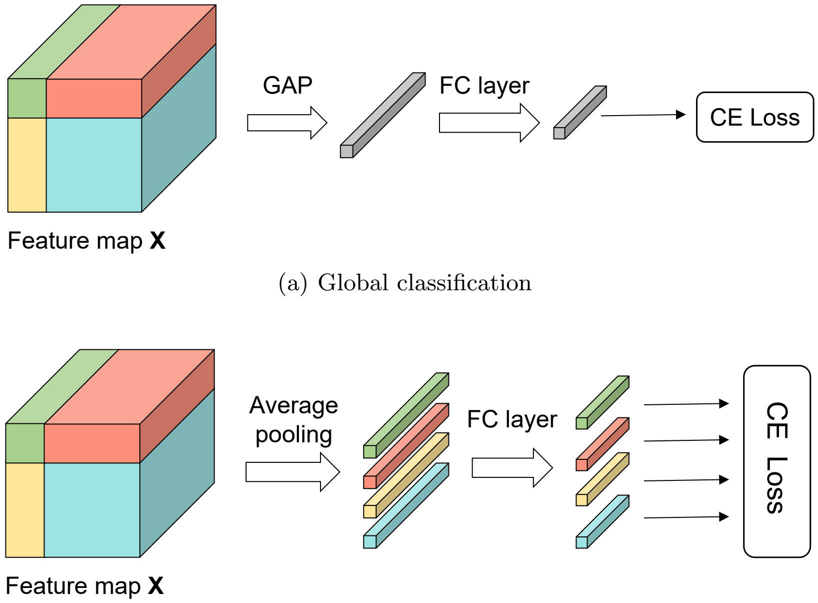
Fig. A1. Visualization of three operations to adjust sample scale. (a) Samples are extracted from the original detection images. Similar to Fig. 2 in the main text, the boxes with green solid lines refer to Ground-Truth bounding boxes, those with red dash lines to the samples, which are randomly enlarged to incorporate more context information. (b) In 'Warp', we change both the aspect ratio and scale of samples. (c) For 'Resize', the aspect ratio is not changed and only the size is changed, then padding is applied to samples whose sizes are smaller than the pre-defined ones. (d) In 'crop', we apply padding or random cropping to samples, conditioned on whether their sizes are smaller or larger than the pre-defined sizes.

B Visualization of global and block-wise classification strategies

This section provides visualization on the process of global and block-wise classification strategies.

Fig. B1(a) shows the process of global classification, where global average pooling is exerted on the feature map and we assign the entire image a single global label. The global label is the weighted sum of labels of the four regions according to their region areas.

Fig. B1(b) depicts the process of block-wise classification. Different from global classification, the average pooling is independently exerted on the four regions of feature map **X** corresponding to samples. Then we will apply classification operation on each region individually according to its label.



(a) Global classification

(b) Block-wise classification

Fig. B1. Visualization on the process of global classification (a) and block-wise classification (b). We use different colors to distinguish regions corresponding to the four samples. Best viewed in color.

References

1. Chen, K., Li, J., Lin, W., See, J., Wang, J., Duan, L., Chen, Z., He, C., Zou, J.: Towards accurate one-stage object detection with ap-loss. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5119–5127 (2019)
2. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 764–773 (2017)
3. Divvala, S.K., Hoiem, D., Hays, J.H., Efros, A.A., Hebert, M.: An empirical study of context in object detection. In: 2009 IEEE Conference on computer vision and Pattern Recognition. pp. 1271–1278. IEEE (2009)
4. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
5. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. arXiv preprint arXiv:1911.05722 (2019)
6. He, K., Girshick, R., Dollár, P.: Rethinking imagenet pre-training. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4918–4927 (2019)
7. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
8. Kornblith, S., Shlens, J., Le, Q.V.: Do better imagenet models transfer better? In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2661–2671 (2019)
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
10. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 734–750 (2018)
11. Li, Y., Chen, Y., Wang, N., Zhang, Z.: Scale-aware trident networks for object detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6054–6063 (2019)
12. Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., Sun, J.: Detnet: A backbone network for object detection. arXiv preprint arXiv:1804.06215 (2018)
13. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
14. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
15. Liu, S., Huang, D., et al.: Receptive field block net for accurate and fast object detection. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 385–400 (2018)
16. Lu, X., Li, B., Yue, Y., Li, Q., Yan, J.: Grid r-cnn. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7363–7372 (2019)
17. Luo, W., Li, Y., Urtasun, R., Zemel, R.: Understanding the effective receptive field in deep convolutional neural networks. In: Advances in neural information processing systems. pp. 4898–4906 (2016)
18. Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., van der Maaten, L.: Exploring the limits of weakly supervised pretraining.

In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 181–196 (2018)

19. Matan, O., Burges, C.J., LeCun, Y., Denker, J.S.: Multi-digit recognition using a space displacement neural network. In: Advances in neural information processing systems. pp. 488–495 (1992)
20. Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., Lin, D.: Libra r-cnn: Towards balanced learning for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 821–830 (2019)
21. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. pp. 8024–8035 (2019)
22. Peng, J., Sun, M., ZHANG, Z.X., Tan, T., Yan, J.: Efficient neural architecture transformation search in channel-level for object detection. In: Advances in Neural Information Processing Systems. pp. 14290–14299 (2019)
23. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
24. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
25. Shen, Z., Liu, Z., Li, J., Jiang, Y.G., Chen, Y., Xue, X.: Dsod: Learning deeply supervised object detectors from scratch. In: Proceedings of the IEEE international conference on computer vision. pp. 1919–1927 (2017)
26. Shrivastava, A., Gupta, A., Girshick, R.: Training region-based object detectors with online hard example mining. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 761–769 (2016)
27. Singh, B., Najibi, M., Davis, L.S.: Sniper: Efficient multi-scale training. In: Advances in neural information processing systems. pp. 9310–9320 (2018)
28. Sun, C., Shrivastava, A., Singh, S., Gupta, A.: Revisiting unreasonable effectiveness of data in deep learning era. In: Proceedings of the IEEE international conference on computer vision. pp. 843–852 (2017)
29. Szegedy, C., Toshev, A., Erhan, D.: Deep neural networks for object detection. In: Advances in neural information processing systems. pp. 2553–2561 (2013)
30. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection. arXiv preprint arXiv:1911.09070 (2019)
31. Wu, Y., He, K.: Group normalization. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 3–19 (2018)
32. Xie, Q., Hovy, E., Luong, M.T., Le, Q.V.: Self-training with noisy student improves imagenet classification. arXiv preprint arXiv:1911.04252 (2019)
33. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1492–1500 (2017)
34. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
35. Zheng, W.S., Gong, S., Xiang, T.: Quantifying and transferring contextual information in object detection. IEEE transactions on pattern analysis and machine intelligence **34**(4), 762–777 (2011)

36. Zhu, C., He, Y., Savvides, M.: Feature selective anchor-free module for single-shot object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 840–849 (2019)
37. Zhu, R., Zhang, S., Wang, X., Wen, L., Shi, H., Bo, L., Mei, T.: Scratchdet: Training single-shot object detectors from scratch. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2268–2277 (2019)