Scale-aware Automatic Augmentation for Object Detection

Yukang Chen[1*†], Yanwei Li[1†], Tao Kong[2], Lu Qi[1], Ruihang Chu[1*], Lei Li[2], Jiaya Jia[1,3]

[1]The Chinese University of Hong Kong [2]ByteDance AI Lab [3]SmartMore

Abstract

*We propose Scale-aware AutoAug to learn data augmentation policies for object detection. We define a new scale-aware search space, where both image- and box-level augmentations are designed for maintaining scale invariance. Upon this search space, we propose a new search metric, termed Pareto Scale Balance, to facilitate search with high efficiency. In experiments, Scale-aware AutoAug yields significant and consistent improvement on various object detectors (*e.g.*, RetinaNet, Faster R-CNN, Mask R-CNN, and FCOS), even compared with strong multi-scale training baselines. Our searched augmentation policies are transferable to other datasets and box-level tasks beyond object detection (*e.g.*, instance segmentation and keypoint estimation) to improve performance. The search cost is much less than previous automated augmentation approaches for object detection. It is notable that our searched policies have meaningful patterns, which intuitively provide valuable insight for human data augmentation design. Code and models are available at https://github.com/Jia-Research-Lab/SA-AutoAug.*

1. Introduction

Object detection, aiming to locate as well as classify various objects, is one of the core tasks in the computer vision. Due to the large scale variance of objects in real-world scenarios, it raises concerns on how to bring the scale adaptation to the network efficiently. Previous work handles this challenge mainly from two aspects, namely *network architecture* and *data augmentation*. To make the network scale invariant, in-network feature pyramids [28, 47, 23] and adaptive receptive fields [25] are usually employed. Another crucial technique to enable scale invariance is data augmentation, which is independent of specific architectures, and can be generalized among multiple tasks.

This paper focuses on data augmentation for object detection. Current data augmentation strategies can be grouped into color operations (*e.g.*, brightness, contrast, and whitening) and geometric operations (*e.g.*, re-scaling, flipping). Among them, geometric operations, such as multi-scale training, improve scale robustness [39, 19]. Several hand-crafted data augmentation strategies were developed to improve performance and robustness of the detector [41, 42]. Previous work [17, 15] also improves box-level augmentation by enriching foreground data. Though inspiring performance gain has achieved, these data augmentation strategies usually rely on heavy expert experience.

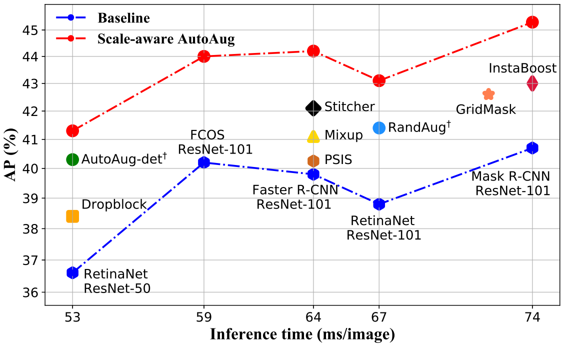


Figure 1: Comparison with object detection augmentation strategies on MS COCO dataset. Methods in the same vertical line are based upon the same detector. Scale-aware AutoAug outperforms both hand-crafted and learned strategies on various detectors.

Automatic data augmentation policies were widely explored in image classification [44, 50, 37, 35, 9]. Its potential for object detection, however, was not thoroughly released. One attempt to automatically learn data augmentation policies for object detectors is AutoAug-det [51][1], which performs color or geometric augmentation upon the context of boxes. It does not fully consider the scale issue from *image*- and *box-level*, which are found, however, essential in object detector design [41, 42, 17]. Moreover, the heavy computational search cost (*i.e.*, 400 TPU for 2 days) impedes it from vastly practical. Thus, scale-aware property and efficiency issue are essential to address for searching augmentation in box-level tasks.

In this paper, we propose a new way to automatically

[*] This work was done during an internship at ByteDance AI Lab. Tao Kong is responsible for correspondence. [†] Equal contribution.

[1] We refer it as AutoAug-det [51] to distinguish from AutoAugment [9].

learn scale-aware data augmentation strategies for object detection and relevant box-level tasks. We first introduce scale-awareness to the search space from two image- and box-levels. For image-level augmentations, zoom-in and -out operations are included with their probabilities and zooming ratios for search. For box-level augmentations, the augmenting areas are generalized with a new searchable parameter, i.e., area ratio. This makes box-level augmentations adaptive to object scales.

Based on our scale-aware search space, we further propose a new estimation metric to facilitate the search process with better efficiency. Previously, each candidate policy is estimated by the validation accuracy on a proxy task [9, 27], which lacks efficiency and accuracy to an extend. Our metric takes advantage of more specific statistics, that is, validation accuracy and accumulated loss over different scales, *to measure the scale balance*. We empirically show that it yields a clearly higher correlation coefficient with the actual accuracy than the previous proxy accuracy metric.

The proposed approach is distinguished from previous work from two aspects. First, different from hand-crafted policies, the proposed method utilizes automatic algorithms to search among a large variety of augmentation candidates. It is hard to be fully explored or achieved by human effort. Moreover, compared with previous learning-based methods, our approach fully explores the important scale issue in both image-level and box-level. With the proposed search space and evaluation metric, our method attains decent performance with much (*i.e.*, 40×) less search cost.

The overall approach, called *Scale-aware AutoAug*, can be easily instantiated for box-level tasks, which will be elaborated on in Sec. 3. To validate its effectiveness, we conduct extensive experiments on MS COCO and Pascal VOC dataset [30, 16] with several anchor-based and anchor-free object detectors, which are reported in Sec. 4.2.

In particular, with ResNet-50 backbone, the searched augmentation policies contribute non-trivial gains over the strong MS baseline of RetinaNet [29], Faster R-CNN [39], and FCOS [43], and achieve 41.3% AP, 41.8% AP, and 42.6% AP, respectively. We further experiment with more box-level tasks, like instance segmentation and keypoint detection. Without bells-and-whistles, our improved FCOS model attains 51.4% AP with the search augmentation policies. Besides, our searched policies present meaningful patterns, which provide intuitive insight for human knowledge.

2. Related Work

Data augmentation has been widely utilized for network optimization and proven to be beneficial in vision tasks [11, 40, 39, 32, 33, 36]. Traditional approaches could be roughly divided into color operations (*e.g.,* brightness, contrast, and whitening) and geometric operations (*e.g.,* scaling, flipping, translation, and shearing), which require hyper-parameter tuning and are usually task-specific [31]. Some commonly used strategies on image classification include random cropping, image mirroring, color shifting/whitening [24], Cutout [12], and Mixup [49].

Scale-wise augmentations also play a vital role for the optimization of object detectors [46, 5]. For example, SNIPER [42] generates image crops around ground truth instances with multi-scale training. YOLO-v4 [2] and Stitcher [8] introduce mosaic inputs that contain rescaled sub-images. For box-level augmentation, Dwibedi et al. [15] improve detection performance with the cut-and-paste strategy. And the visual context surrounding objects are modeled in [14]. Furthermore, InstaBoost [17] augments training images using annotated instance masks with a location probability map. However, these hand-crafted designs still highly rely on expert efforts.

Inspired by recent advancements in neural architecture search (NAS) [52, 53, 38, 7], researchers try to learn augmentation policies from data automatically. An example is AutoAugment [9], which searches data augmentations for image classification and achieves promising results. PBA [22] uses population-based search method for better efficiency. Fast AutoAugment [27] applies Bayesian optimization to learn data augmentation policies. RandAug [10] removes the search process at the price of manually tailoring the search space to a very limited volume. AutoAugDet [51] extends AutoAugment [9] to object detection by taking box-level augmentations into consideration.

3. Scale-aware AutoAug

In this section, we first briefly review the auto augmentation pipeline. Then, the *scale-aware search space* and *estimation metric* will be respectively elaborated in Sec. 3.2 and Sec. 3.3. We finally show the search framework in Sec. 3.4.

3.1. Review of AutoAug

Auto augmentation methods [9, 51, 22, 27, 26] commonly formulate the process of finding the best augmentation policy as a search problem. To this end, three main components are needed, namely the *search space*, *search algorithm*, and *estimation metric*. Search space may vary according to tasks. For example, the search space [9, 22, 27] is developed to image classification, while it is not the specified case for box-level tasks. As for search algorithms, reinforcement learning [52] and evolutionary algorithm [38] are usually utilized to explore the search space in iterations. During this procedure, each child model, which is optimized with the searched policy p, is evaluated on a designed metric to estimate its effectiveness. This metric serves as feedback for the search algorithm.

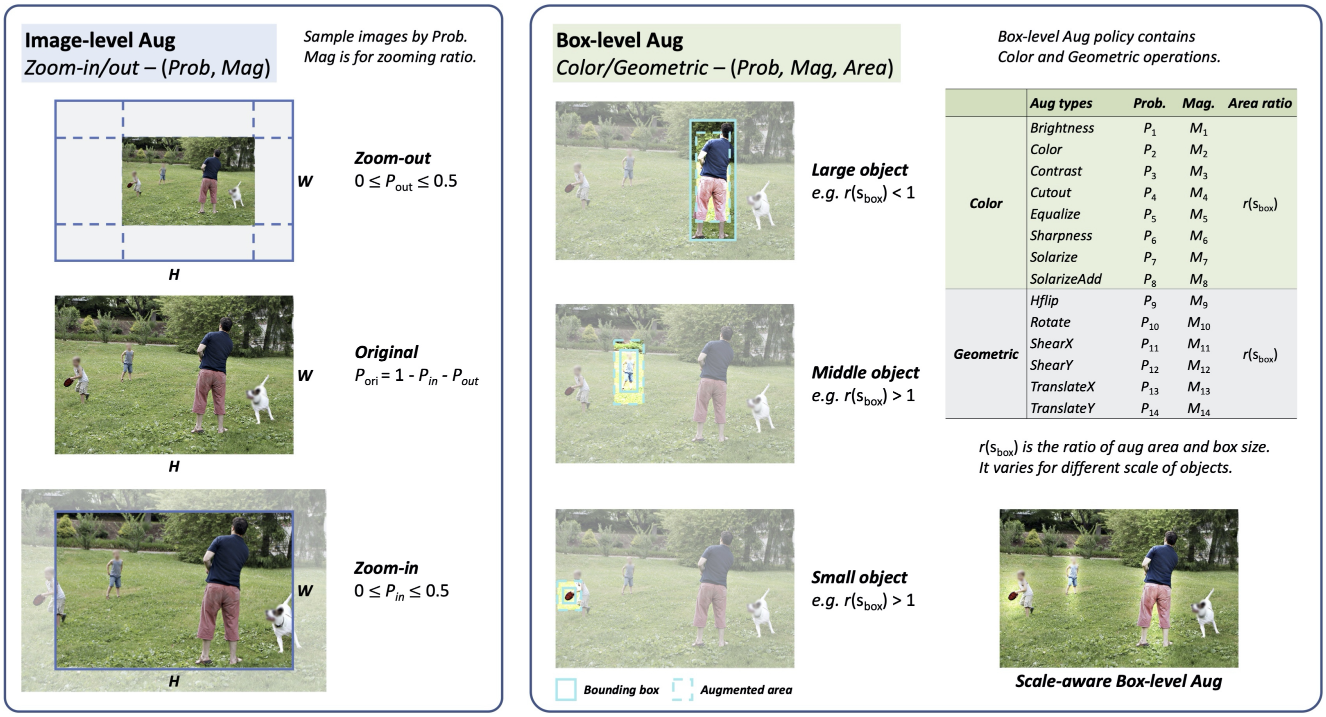


Image-level Aug
Zoom-in/out – (Prob, Mag)
Sample images by Prob.
Mag is for zooming ratio.
W
H
Zoom-out
$0 \leq P_{out} \leq 0.5$
W
H
Original
$P_{ori} = 1 - P_{in} - P_{out}$
W
H
Zoom-in
$0 \leq P_{in} \leq 0.5$
Box-level Aug
Color/Geometric – (Prob, Mag, Area)
Box-level Aug policy contains
Color and Geometric operations.
Large object
e.g. $r(s_{box}) < 1$
Middle object
e.g. $r(s_{box}) > 1$
Small object
e.g. $r(s_{box}) > 1$
$r(s_{box})$ is the ratio of aug area and box size.
It varies for different scale of objects.
Bounding box
Augmented area
Scale-aware Box-level Aug

	Aug types	Prob.	Mag.	Area ratio
Color	Brightness	P_1	M_1	$r(s_{box})$
	Color	P_2	M_2	
	Contrast	P_3	M_3	
	Cutout	P_4	M_4	
	Equalize	P_5	M_5	
	Sharpness	P_6	M_6	
	Solarize	P_7	M_7	
	SolarizeAdd	P_8	M_8	
Geometric	Hflip	P_9	M_9	$r(s_{box})$
	Rotate	P_{10}	M_{10}	
	ShearX	P_{11}	M_{11}	
	ShearY	P_{12}	M_{12}	
	TranslateX	P_{13}	M_{13}	
	TranslateY	P_{14}	M_{14}	

Figure 2: Scale-aware search space. It contains image-level and box-level augmentation. Image-level augmentation includes zoom-in and zoom-out functions with probabilities and magnitudes for search. In box-level, we introduce scale-aware area ratios, which make operations adaptive to objects in different scales. Augmented images are further generalized with the Gaussian map.

3.2. Scale-aware Search Space

The designed scale-aware search space contains both *image-level* and *box-level* augmentations. The image-level augmentations include zoom-in and zoom-out functions on the whole image. As for box-level augmentations, color and geometric operations are searched for objects in images.

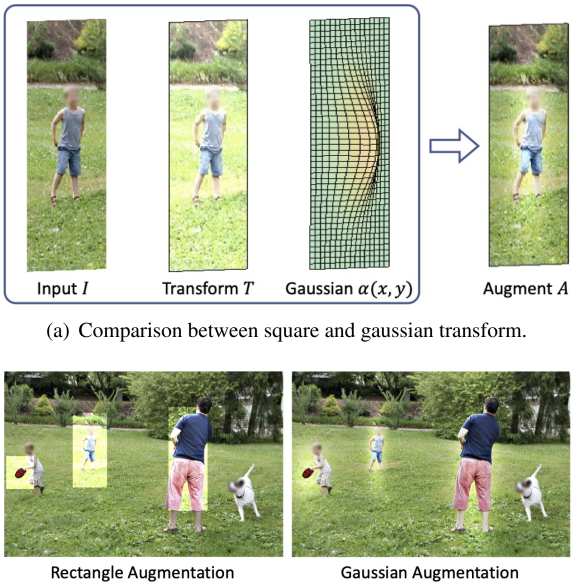
Image-level augmentations. To handle scale variations, object detectors are commonly trained with image pyramids. However, these scale settings highly rely on hand-crafted selection. In our search space, we alleviate this burden by searchable zoom-in and zoom-out functions. As illustrated in the left part of Fig. 2, zoom-in and zoom-out functions are specified by probabilities P and magnitudes M. Specifically, the probabilities P_{in} and P_{out} are searched in the range from 0 to 0.5. With this range, the existence of original scale could be guaranteed with the probability

$$P_{\text{ori}} = 1 - P_{\text{out}} - P_{\text{in}}. \tag{1}$$

The magnitude M represents the zooming ratio for each function. For the zoom-out function, we search a zooming ratio from 0.5 to 1.0. For the zoom-in function, we search a zooming ratio from 1.0 to 1.5. For example, if a zooming ratio of 1.5 is selected, it means that the input images might be increased by $1.5\times$. In traditional multi-scale training, large-scale images would introduce an additional computational burden. To avoid this issue, we reserve the original shape in the zoom-in function with random cropping.

After the search procedure, input images are randomly sampled from zoom-in, zoom-out, and original scale images with the searched P and M in each training iteration. In other words, we sample from 3 resolutions, a larger one, a small one and the original with the searched probabilities, *i.e.*, $\{P_{in}, P_{out}, P_{ori}\}$. To our best knowledge, no previous work considers automatic scale-aware transformation search for object detection. Experiments validate the superiority over traditional multi-scale training in Tab. 2.

Box-level augmentations. The box-level augmentations are designed to conduct transformation for each object box. Different from [51], the proposed approach further smooths the augmentations and relaxes it to contain learnable factors, *i.e.,* area ratio. In particular, previous box-level augmentation [51] works exactly in the whole bounding box annotations without attenuation, which generate an obvious boundary gap between the augmented and original region. The sudden appearance change could reduce the difficulty for networks to locate the augmented objects, which brings the gap between training and inference. To solve this issue, we extend the original rectangle augmentation to a



(a) Comparison between square and gaussian transform.

(b) Gaussian-based transform process.

Figure 3: An example of Gaussian-based box-level augmentation. It removes the original hard boundary and the augmented areas are adjustable to the Gaussian variance.

gaussian-based manner. A visualization example of a box-level *color brightness* operation is given in Fig. 3(a). We blend the original and the transformed pixels with a spatial-wise Gaussian map $\alpha(x, y)$ by

$$A = \alpha(x, y) \cdot I + (1 - \alpha(x, y)) \cdot T, \tag{2}$$

where I, T, and A denotes the input, transformation function, and augmented region, respectively. This process is depicted in Fig. 3(b). Actually, the designed gaussian-based process softens the boundary gap in a more natural manner.

The second issue in previous operations is the lack of considering receptive fields and object scales. A common belief is that neural networks largely rely on the context information to recognize objects [4, 34]. Experimentally, we find that it may not be correct for objects in all scales, while the effect varies with objects scales. This is demonstrated with widely applied two-stage and one-stage detectors, namely the Faster R-CNN [39] and RetinaNet [29]. As presented in Tab. 1, if we test it on the COCO validation set with all context (background) pixels removed, its accuracy on small objects, AP$_s$, dramatically declines from 25.2% to 18.0%. In contrast, AP$_l$ increases from 53.0% to 56.1%. It is consistent with that in RetinaNet [29]. This motivates us that augmentations merely inside/outside object boxes may not deal with objects in all scales appropriately. To this end, we introduce a searchable parameter, *area ratio*, which makes the aug area adaptive to object sizes.

Table 1: Analysis on the context for scales. On well-trained ResNet-101 detectors, AP$_s$ drops and AP$_l$ increases consistently if contexts are removed in validation images.

	with context	AP	AP$_s$	AP$_m$	AP$_l$
Faster R-CNN	✓	41.4	25.2	44.8	53.0
	✗	40.5	18.0	45.7	56.1
	Δ	-0.9	**-7.2**	+0.9	**+3.1**
RetinaNet	✓	40.3	23.3	44.0	53.3
	✗	39.8	16.7	44.4	57.7
	Δ	-0.5	**-6.6**	+0.4	**+4.4**

Here, we generalize the Gaussian map with the parameter of area ratio. Given an image with size $H \times W$ and bounding box annotations, the box (x_c, y_c, h, w) could be represented with the central point (x_c, y_c) and the height/width h/w. We formulate the Gaussian map by

$$\alpha(x, y) = \exp\left(-\left(\frac{(x - x_c)^2}{2\sigma_x^2} + \frac{(y - y_c)^2}{2\sigma_y^2}\right)\right). \tag{3}$$

Then, we define the augmentation area V as the integration of the Gaussian map, where

$$V = \int_0^H \int_0^W \alpha(x, y)\, \mathrm{d}x\mathrm{d}y. \tag{4}$$

The *area ratio* for the box-level augmentation is denoted as r. Here, $r(s_{\text{box}}) = V/s_{\text{box}}$ is searchable for different scales, which determines the spatially augmentation area for each object. Thus, the standard deviation factors, σ_x and σ_y, could be calculated as in Eq. (5). We provide the detailed calculation process in the *supplementary materials*.

$$\sigma_x = h\sqrt{\frac{W/H}{2\pi}r}, \quad \sigma_y = w\sqrt{\frac{H/W}{2\pi}r}. \tag{5}$$

Search space summary. Our search space contains both image-level and box-level augmentations. For the image-level augmentation, we search for the parameters of zoom-in and zoom-out operations. To keep consistent with the convention [51], our box-level operations have 5 sub-policies, where sub-policy consists of a *color* operation as well as a *geometric* operation. Each operation contains two parameters, namely the *probability* and *magnitude*. The probability is sampled from a set of 6 discrete values, from 0 to 1.0 with 0.2 as the interval. The magnitude represents the strength factor for each operation with custom range values. We map the magnitude range to a standardized set of 6 discrete values, from 0 to 10 with 2 as the interval. For box-level operations, there are 3 area ratios to search for small, middle and large scales. Each area ratio is independently searched in a discrete set of 10 values. We list the details of these operations in the *supplementary materials*. In summary, the total search space provides $(6^2)^2 \times (((6 \times 6^2) \times (8 \times 6^2))^5 \times 10^3) = 1.2^{30}$ candidate policies, which is twice large as [51].

3.3. Scale-aware Estimation Metric

Autoaugment methods commonly employ validation accuracies on a proxy task (a small subset with training images) as the search metric. However, such a manner is found to be inaccurate and computationally expensive [10], which will be further demonstrated in Fig. 4. In contrast, all operations in our search space, both image-level and box-level, have explicit relationships with each scale. Thanks for the convenience, a scale-aware metric can be further proposed to capture more specific statistics of different scales. Specifically, the evaluation metric is established based on an observation that *balanced optimization over different scales would be beneficial to training*. Thus, a scale-aware metric can be formulated with the record of the accumulated loss and accuracy on different scales during fine-tuning.

Given a plain model trained without data augmentation, we record its validation accuracy AP and accuracies on each scale AP_i with $i \in S$. For each candidate policy p, a child model is further fine-tuned upon it. We record the accumulated loss L_i^p, the validation accuracies on each scale AP_i^p, and the overall AP to formulate the objective function as

$$\min_p f(\{L^p_{i\in S}\}, \{AP^p_{i\in S}\}). \tag{6}$$

Balanced optimization over different scales is essential to the performance and robustness of object detectors. An intuitive way to measure the balance is the standard deviation $\sigma(\{L_i^p | i \in S\})$ of losses on various scales. However, we find it sometimes delves into a sub-optimal where other scales are sacrificed to achieve the optimization balance.

Here, we adopt the principle of Pareto Optimality [1] to overcome this obstacle. In particular, we introduce a concept, named Pareto Scale Balance, to describe our objective: *the optimization over scales can not be better without hurting the accuracy of any other scale*. To this end, we introduce a penalization factor Φ to punish the scales $\widehat{S}$ where accuracy drops after fine-tuning with the policy p. Therefore, the metric function can be upgraded to

$$f(\{L^p_{i\in S}\}, \{AP^p_{i\in S}\}) = \sigma(\{L^p_{i\in S}\}) \cdot \Phi(\{AP^p_{i\in \widehat{S}}\}), \tag{7}$$

where $\Phi(\{AP^p_{i\in \widehat{S}}\}) = \prod_{i\in \widehat{S}} \frac{AP_i}{AP_i^p}$ and $\frac{AP_i}{AP_i^p}$ is the scale-wise ratio of the original and the fine-tuned accuracy.

Compared with previous proxy-accuracy metrics, ours is superior in *computational efficiency* and *estimation accuracy*. Towards efficient computation, child models are fine-tuned upon the given plain model, instead of training from scratch. We record the changes that resulted from the augmented fine-tuning to compute our metric. For accurate estimation, more specific statistics, that is, AP and loss over various scales, is reasonable to receive a higher coefficient with the actual performance. Experimentally, we carefully compare the proposed search metric with the original accuracy metric to verify the effectiveness in Sec. 4.2.

Algorithm 1: Search Framework

```
Input  : Plain model $m$, Initialized Population $\mathbf{P}$,
         Training set $\mathbb{T}$, Val set $\mathbb{V}$, Iterations #I.
$f^*, p^*$ ← ($\infty, \varnothing$)
for $k \in (1\ to\ \#\mathrm{I})$ do
    for $p \in \mathbf{P}$ do
        $m_p$, $\{L^p_{i\in S}\}$ ← finetune($m$,$\mathbb{T}$,$\theta_p$)
        $\{AP^p_{i\in S}\}$ ← evaluate($m_p$, $\mathbb{V}$)
        $f_p$ ← $f(\{L^p_{i\in S}\}, \{AP^p_{i\in S}\})$ -- Eq.(7)
        if $f_p < f^*$ then
            $f^*, p^*$ ← ($f_p, p$)
    $\mathcal{P}$ ← select-topk($\mathbf{P}$)
    $\mathbf{P}$ ← mutate-crossover($\mathcal{P}$)
Output: The best augmentation policy $p^*$
```

Table 2: Improvement details on RetinaNet ResNet-50.

	AP	AP$_{50}$	AP$_{75}$	AP$_s$	AP$_m$	AP$_l$
MS Baseline	38.2	57.3	40.5	23.0	41.6	50.3
Ours image-level	40.1	59.8	43.3	24.0	44.1	53.1
+ box-level	40.6	60.4	43.6	24.1	44.4	53.5
+ scale-aware area	**41.3**	**61.0**	**44.1**	**25.2**	**44.5**	**54.6**

Table 3: Comparison with AutoAug-det on RetinaNet ResNet-50.

	search cost	AP	AP$_s$	AP$_m$	AP$_l$
AutoAug-det [51]	800 TPU-days	36.7→39.0	-	-	-
AutoAug-det† [51][2]	800 TPU-days	38.2→40.3	23.6	43.9	53.8
Ours	20 GPU-days	38.2→**41.3**	25.2	44.5	54.6

Table 4: Search on RetinaNet ResNet-50 with different metrics.

	AP	AP$_{50}$	AP$_{75}$	AP$_s$	AP$_m$	AP$_l$
proxy accuracy in [51]	40.0	59.7	42.5	23.9	44.1	52.6
scale loss std σ	40.7	60.5	43.5	24.1	44.5	53.5
our metric - Eq. (7)	**41.3**	**61.0**	**44.1**	**25.2**	44.5	**54.6**

3.4. Search Framework

Given the above search space and search metric, we describe the search framework in this section. In this work, the evolutionary algorithm, *e.g.,* tournament selection algorithm [38], is adopted as the search controller. Specifically, a population of $|P|$ policies are sampled from the search space in each iteration. After evaluating the sampled policies, we select the top k policies as *parent* for the next generation. Then, child policies are produced by mutation and crossover among the parent policies. This process is repeated for iterations until convergence.

To evaluate augmentation policies, we first train a plain model with no data augmentation. Then, we fine-tune it upon each augmentation policy for n iterations and record the accumulated loss during optimization. We also record its accuracy *before* and *after* fine-tuning. With these statistics, the search metric for each policy could be obtained. This search framework is illustrated in Alg. 1.

[2] † means our implementation with the same baseline settings to ours.

Table 5: Improvements across detection frameworks.

Models	*policy*	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
RetinaNet:							
ResNet-50[3]	Baseline	36.6	55.7	39.1	20.8	40.2	49.4
	MS Baseline	38.2	57.3	40.5	23.0	41.6	50.3
	Ours	**41.3**	61.0	44.1	25.2	44.5	54.6
ResNet-101	Baseline	38.8	59.1	42.3	21.8	42.7	50.2
	MS Baseline	40.3	59.8	42.9	23.2	44.0	53.2
	Ours	**43.1**	62.8	46.0	26.2	46.8	56.7
Faster R-CNN:							
ResNet-50	Baseline	37.6	57.8	41.0	22.2	39.9	48.4
	MS Baseline	39.1	60.8	42.6	24.1	42.3	50.3
	Ours	**41.8**	63.3	45.7	26.2	44.7	54.1
ResNet-101	Baseline	39.8	61.3	43.5	23.1	43.2	52.3
	MS Baseline	41.4	60.4	44.8	25.0	45.5	53.1
	Ours	**44.2**	65.6	48.6	29.4	47.9	56.7
FCOS:							
ResNet-50	MS Baseline	40.8	59.6	43.9	26.2	44.9	51.9
	Ours	**42.6**	61.2	46.0	28.2	46.4	54.3
ResNet-101	MS Baseline	41.8	60.3	45.3	25.6	47.7	56.1
	Ours	**44.0**	62.7	47.3	28.2	47.8	56.1

Table 6: Improvements across tasks on Mask R-CNN.

Models	*policy*	$AP^{m/k}$	$AP^{m/k}_{50}$	$AP^{m/k}_{75}$	AP^{b}	AP^{b}_{50}	AP^{b}_{75}
Instance Segmentation:							
ResNet-50	MS Baseline	36.4	58.8	38.7	40.4	61.9	44.0
	Ours	**38.1**	60.9	40.8	**42.8**	64.4	46.9
ResNet-101	MS Baseline	37.9	60.4	40.4	42.3	63.8	46.6
	Ours	**40.0**	63.2	42.9	**45.3**	66.4	49.8
Keypoint Estimation:							
ResNet-50	MS Baseline	64.1	85.9	69.7	53.5	82.7	58.4
	Ours	**65.7**	86.6	71.7	**55.5**	84.2	60.9
ResNet-101	MS Baseline	65.1	86.5	71.2	54.8	83.2	60.0
	Ours	**66.4**	87.5	72.7	**56.5**	84.6	62.1

4. Experiments

4.1. Implementation Details

Policy search. In the search phase, we adopt RetinaNet [29] on ResNet-50 [21] backbone. We split the detection dataset into a training set for child model training, a validation set for evaluation during search, and the test set `val2017` for final evaluation. The validation set contains 5k images randomly sampled from the `train2017` in MS COCO [30] and the remains are for child model training. Each child model is fine-tuned for 1k iterations on the plain model, which is just an arbitrary partially trained baseline model. In the evolutionary search, the evolution process is repeated for 10 iterations. The evolution population size is 50 and the top 10 models are selected as subsequent parents.

Final policy evaluation. Models are trained with the searched augmentation policy in the typical pre-training and fine-tuning schedule on MS COCO dataset. The training images are resized such that the shorter size is 800 pixels. Faster R-CNN and RetinaNet models are trained for 540k iterations to fully show its potential, while others are trained for 270k iterations. Multi-scale training baselines are enhanced by randomly selecting a scale between 640 to 800 during training. We train models on 8 GPUs with a total 16 images per batch. The learning rate is initialized as 0.02. We set weight decay as 0.0001 and momentum as 0.9.

4.2. Verification

In this section, we systematically evaluate our proposed Scale-aware AutoAug. We first present the improvements from the search policy on the target task and then show its transferability to other tasks and datasets. After that, we analyze the proposed search metric in detail.

Improvements analysis. The top block in Tab. 5 shows the improvements from our searched augmentation policy on RetinaNet. On ResNet-50 backbone, our searched augmentation policy enhances the competitive multi-scale training baseline to 41.3% AP by 3.1%. On ResNet-101, it achieves a 2.8% gains to 43.1% AP. We also perform experiments upon the large scale jittering [13] in the *supplementary materials*. These improvements come from training data augmentations and introduce no additional cost to inference.

For a better understanding of the improvements, we show the component-wise improvements in Tab. 2. The image-level augmentations boost the performance by 1.9% AP from 38.2% to 40.1%. Upon this, the non-scale-aware box-level augmentations improve the performance to 40.6%. If it is further upgraded to be scale-aware, the performance gets an additional 0.7% enhancement to 41.3%. In contrast, in AutoAug-det [51], the box-level augmentations yield only 0.4% improvements. The improvements mostly come from small and large objects, which verifies the effectiveness of scale-aware box-level augmentations.

In addition, we compare with the previous state-of-the-art auto augmentation method in Tab. 3. On RetinaNet [29] with ResNet-50 [21] backbone, AutoAug-det [51] improves the baseline from 36.7% to 39.0% by 2.3% AP. For better comparison, we implement the searched policy in AutoAug-det [51] on our baseline. It is trained on the exact same settings, except for the data augmentation policy. It improves the baseline from 38.2% to 40.3% by 2.1% AP. It is inferior to our +3.1% improvement. For small objects, our searched policy gets a more balanced performance (*i.e.*, +1.6 AP_s) thanks to the scale-aware search space and metric. In terms of search cost, the data augmentation policy in AutoAug-det [51] costs 800 TPU-days (400 TPUs on 2 days) for search, while our search costs only 8 GPUs (Tesla-V100) on 2.5 days. It is a 40× computational saving, without considering the machine type difference.

Transferability. Although our data augmentation policy is searched in object detection on RetinaNet, we make comprehensive experiments to show its effectiveness to work on other object detectors, datasets and relevant tasks.

[3] FPN [28] is used as a default setting, unless -C4 is denoted.

Table 7: Improvements on PASCAL VOC with Faster R-CNN on ResNet-50 backbone.

	mAP	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
baseline	78.6	80.9	80.8	79.3	72.3	67.2	87.4	88.5	88.6	62.6	86.0	71.2	88.0	88.9	80.6	79.9	52.6	78.7	74.0	86.2	78.3
+ ours	**81.6**	88.7	88.2	80.1	74.1	73.6	88.3	89.1	88.9	68.1	87.2	73.8	88.4	88.9	87.5	87.1	56.2	79.0	79.7	87.2	78.6

Table 8: Comparison with state-of-the-art data augmentation methods for object detection.

Method	Detector	Backbone	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Hand-crafted:								
Dropblock [18]	RetinaNet	ResNet-50	38.4	56.4	41.2	-	-	-
Mix-up [49]	Faster R-CNN	ResNet-101	41.1	-	-	-	-	-
PSIS* [45]	Faster R-CNN	ResNet-101	40.2	61.1	44.2	22.3	45.7	51.6
Stitcher [8]	Faster R-CNN	ResNet-101	42.1	-	-	26.9	45.5	54.1
GridMask [6]	Faster R-CNN	ResNeXt-101	42.6	65.0	46.5	-	-	-
InstaBoost* [17]	Mask R-CNN	ResNet-101	43.0	64.3	47.2	24.8	45.9	54.6
SNIP (MS test)* [41]	Faster R-CNN	ResNet-101-DCN-C4	44.4	66.2	49.9	27.3	47.4	56.9
SNIPER (MS test)* [42]	Faster R-CNN	ResNet-101-DCN-C4	46.1	67.0	51.6	29.6	48.9	58.1
Automatic:								
AutoAug-det [51]	RetinaNet	ResNet-50	39.0	-	-	-	-	-
AutoAug-det [51]	RetinaNet	ResNet-101	40.4	-	-	-	-	-
AutoAug-det† [51]	RetinaNet	ResNet-50	40.3	60.0	43.0	23.6	43.9	53.8
AutoAug-det† [51]	RetinaNet	ResNet-101	41.8	61.5	44.8	24.4	45.9	55.9
RandAug [10]	RetinaNet	ResNet-101	40.1	-	-	-	-	-
RandAug† [10]	RetinaNet	ResNet-101	41.4	61.4	44.5	25.0	45.4	54.2
Ours:								
Scale-aware AutoAug	RetinaNet	ResNet-50	41.3	61.0	44.1	25.2	44.5	54.6
Scale-aware AutoAug	RetinaNet	ResNet-101	43.1	62.8	46.0	26.2	46.8	56.7
Scale-aware AutoAug	Faster R-CNN	ResNet-101	44.2	65.6	48.6	29.4	47.9	56.7
Scale-aware AutoAug (MS test)	Faster R-CNN	ResNet-101-DCN-C4	47.0	68.6	52.1	32.3	49.3	60.4
Scale-aware AutoAug	FCOS	ResNet-101	44.0	62.7	47.3	28.2	47.8	56.1
Scale-aware AutoAug	FCOS‡	ResNeXt-32x8d-101-DCN	48.5	67.2	52.8	31.5	51.9	63.0
Scale-aware AutoAug (1200 size)	FCOS‡	ResNeXt-32x8d-101-DCN	49.6	68.5	54.1	35.7	52.5	62.4
Scale-aware AutoAug (MS test)	FCOS‡	ResNeXt-32x8d-101-DCN	**51.4**	69.6	57.0	37.4	54.2	65.1

In object detection, we verify our policy on mainframe anchor-based one-stage, two-stage, and anchor-free detectors. In addition to the previous RetinaNet experiments, we show our results on Faster R-CNN and FCOS in Tab. 5. The improvements on Faster R-CNN are remarkable, *i.e.*, +2.7% and +2.8% on ResNet-50 and ResNet-101, respectively. On the anchor-free detector FCOS, it achieves 44.0% AP on ResNet-101 with similar improvements.

Our augmentation policy is feasible in any box-level tasks. We validate its performance using Mask R-CNN [20] on instance segmentation and keypoint estimation. Similar improvements are consistently present in Tab. 6. For instance segmentation, our Mask R-CNN model achieves 40.0% mask AP on ResNet-101 backbone. In addition, we also transfer our augmentation policy to PASCAL VOC dataset. We train a Faster R-CNN model on ResNet-50 for 48k iterations and divide the learning rate at 36k iterations. It improves the baseline by 3% mAP as in Tab. 7.

Table 9: Searched augmentation policy.

Image-level	(Zoom-in, 0.2, 4)	(Zoom-out, 0.4, 10)
Box-level	*Color operations*	*Geometric operations*
Sub-policy 1.	(Color, 0.4, 2)	(TranslateX, 0.4, 4)
Sub-policy 2.	(Brightness, 0.2, 4)	(Rotate, 0.4, 2)
Sub-policy 3.	(Sharpness, 0.4, 2)	(ShearX, 0.2, 6)
Sub-policy 4.	(SolarizeAdd, 0.2, 2)	(Hflip, 0.3, 0)
Sub-policy 5.	Original	(TranslateY, 0.2, 8)
Area ratio	Small - 6 Middle - 2 Large - 0.4	

Search metric analysis. We compare our search metric with the proxy accuracy metric. For the proxy accuracy metric in [51], each model is trained on a subset training set, 5k images. For each search metric, we train 50 models with policies randomly sampled in the search space. Each model is trained for 90k iterations and evaluated on `val2017` to obtain the actual accuracy. Meanwhile, the proxy accuracy metric and our std-based metric are computed for each model. We illustrate the Pearson coefficients in Fig. 4. Our std-based metric is horizontally flipped in [0, 1] for better illustration. It shows that our metric has a clearly higher coefficient to actual accuracies than the proxy accuracy metric. In addition, we use different metrics for search in Tab. 4.

[‡] For FCOS ResNeXt-32x8d-101-DCN models, it is an improved version with ATSS [48] for performance boosting. * results on `test-dev`. Our comparisons on `test-dev` are in the *supplementary materials*.

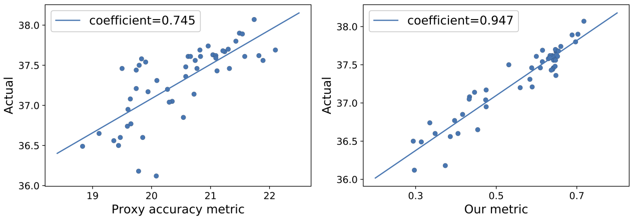


Figure 4: Coefficients between actual accuracy and metrics. Our metric presents a higher coefficient than the proxy accuracy [51].

The search metric in Eq.(7) is slightly better than the purely std metric, thanks to the penalty factor.

4.3. Comparison

We compare our final models on the MS COCO dataset with other data augmentation methods in object detection. The training settings are consistent with the implementation details mentioned before. As shown in Tab. 8, our augmentation method on Faster R-CNN with ResNet-101 backbone achieves 44.2% AP, without any testing techniques. It is better than the augmentation methods with the same backbone, including InstaBoost [17] on Mask R-CNN (43.0% AP). To compare with the state-of-the-art hand-crafted augmentations, SNIP [41] and SNIPER [42] on Faster R-CNN, we use the exactly the same settings, which includes multi-scale testing on [480, 800, 1400] sizes, valid ranges, and Soft-NMS [3]. No flipping or other enhancements are used. Our model on the same backbone achieves 47.0% AP. It is better than the 46.1% SNIPER. We also compare with automatic augmentation methods, AutoAug-det [51] and RandAug [10]. For a fair comparison, we train them with the same training settings to our methods on various backbones, denoted as †. They are inferior to ours.

In addition to these common comparisons, we conduct experiments on large-scale models to push the envelope of Scale-aware AutoAug. The baseline is the improved version of FCOS [48] on ResNeXt-32x8d-101-DCN backbone with multi-scale training. It has 47.5% AP in the standard single scale testing. Without any bells and whistles, Scale-aware AutoAug enhances this strong baseline to 48.5% AP by + 1.0% increase. It is further improved to 49.6% AP on larger training images with 1200 size. When it is equipped with multi-scale testing, it is promoted to 51.4% AP.

4.4. Discussion

Understanding the searched policy. Tab. 9 illustrates our learned augmentation policy in details. We present each individual augmentation in the format of {type, probability, magnitude}. Probabilities are in the range of [0, 1.0]. The magnitude ranges for augmentations are listed in the *supplementary materials*. We measure it with 0 to 10 during search. This searched policy presents meaningful patterns.

Table 10: Scale variation issue on a clean Faster R-CNN.

	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
ResNet-50-C4	34.7	55.7	37.1	18.2	38.8	48.3
with MS train	34.8	55.6	37.3	18.9	39.2	47.6
with FPN	36.7	58.4	39.6	21.1	39.8	48.1
with Ours	**36.8**	58.0	39.5	21.0	41.2	49.1

- Zoom-out has higher probability and magnitude than zoom-int. This matches the fact that object detectors usually have unsatisfied performance in small objects, while zoom-out benefits detecting small objects.
- The area ratio decreases dramatically from small scale to large scale. Note that the area ratio is searched independently in various scales from a set of discrete numbers. This phenomenon shows that augmentation involving the context (area ratio larger than 1.0) would be beneficial to small and middle object recognition.
- In box-level augmentations, geometric operations generally have higher probability and magnitude than color operations. It intuitively reveals that geometric operations, *e.g.*, rotation, translation, shearing, might have more effect than color ones in object detection.

The above patterns accord with our intuition and could provide valuable insights to human knowledge.

Image/Feature pyramids v.s. Scale-aware AutoAug. Feature pyramid network [28] is proposed for solving the scale variance issue in Faster R-CNN and has been widely used in this area. Here we show that our Scale-aware AutoAug could be a substitute for FPN on Faster R-CNN detector as in Tab. 10. Multi-scale training is commonly known to be scale-invariant. However, on a clean baseline of Faster R-CNN [39] without FPN in 90k training iterations, it provides almost no benefits. In contrast, our augmentation policy improves the baseline to the performance that requires training with FPN. Note that our augmentation policy is cost-free and requires no network modification.

5. Conclusion

In this work, we present Scale-aware AutoAug for object detection. It aims at the common scale variation issue with our search space and search metric. Scale-aware AutoAug spends 20 GPU-days searching augmentation policies, 40 × saving compared to previous work. Our method shows significant improvements over several strong baselines. Although the augmentation policy is searched in object detection on the COCO dataset, it is transferable to other tasks and dataset. Thus, it provides a practical solution for research and applications of augmentations in object detection. Finally, the searched augmentation policy have meaningful patterns, which might, in return, provide valuable insights for the hand-crafted data augmentation design.

References

[1] John Black, Nigar Hashimzade, and Gareth Myles. *A dictionary of economics*. Oxford university press, 2012. 5

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020. 2

[3] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-nms - improving object detection with one line of code. In *ICCV*, pages 5562–5570, 2017. 8

[4] Ali Borji and Seyed Mehdi Iranmanesh. Empirical upper bound in object detection and more. *CoRR*, 2019. 4

[5] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *CoRR*, 2019. 2

[6] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Gridmask data augmentation. *CoRR*, abs/2001.04086, 2020. 7

[7] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS*, pages 6638–6648, 2019. 2

[8] Yukang Chen, Peizhen Zhang, Zeming Li, Yanwei Li, Xiangyu Zhang, Gaofeng Meng, Shiming Xiang, Jian Sun, and Jiaya Jia. Stitcher: Feedback-driven data provider for object detection. *CoRR*, abs/2004.12432, 2020. 2, 7

[9] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, pages 113–123, 2019. 1, 2

[10] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*, 2020. 2, 5, 7, 8

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 2

[12] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. 2

[13] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V Le, and Xiaodan Song. SpineNet: Learning scale-permuted backbone for recognition and localization. In *CVPR*, pages 11592–11601, 2020. 6

[14] Nikita Dvornik, Julien Mairal, and Cordelia Schmid. Modeling visual context is key to augmenting object detection datasets. In *ECCV*, pages 364–380, 2018. 2

[15] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *ICCV*, pages 1301–1310, 2017. 1, 2

[16] Mark Everingham, S. M. Ali Eslami, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 2

[17] Haoshu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yonglu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In *ICCV*, pages 682–691, 2019. 1, 2, 7, 8

[18] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Dropblock: A regularization method for convolutional networks. In *NeurIPS*, pages 10750–10760, 2018. 7

[19] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron, 2018. 1

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 7

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 6

[22] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *ICML*, pages 2731–2741, 2019. 2

[23] Tao Kong, Anbang Yao, Yurong Chen, and Fuchun Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *CVPR*, pages 845–853, 2016. 1

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012. 2

[25] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *ICCV*, 2019. 1

[26] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy M. Hospedales, Neil Martin Robertson, and Yongxing Yang. DADA: differentiable automatic data augmentation. In *ECCV*, 2020. 2

[27] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In *NeurIPS*, pages 6662–6672, 2019. 2

[28] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1, 6, 8

[29] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2999–3007, 2017. 2, 4, 6

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755, 2014. 2, 6

[31] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *IJCV*, pages 261–318, 2020. 2

[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37, 2016. 2

[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 2

[34] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard S. Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *NeurIPS*, pages 4898–4906, 2016. 4

[35] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017. 1

[36] Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with KINS dataset. In *CVPR*, pages 3014–3023, 2019. 2

[37] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *NeurIPS*, pages 3236–3246, 2017. 1

[38] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, pages 4780–4789, 2019. 2, 5

[39] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149, 2017. 1, 2, 4, 8

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014. 2

[41] Bharat Singh and Larry S. Davis. An analysis of scale invariance in object detection SNIP. In *CVPR*, pages 3578–3587, 2018. 1, 7, 8

[42] Bharat Singh, Mahyar Najibi, and Larry S. Davis. SNIPER: efficient multi-scale training. In *NeurIPS*, pages 9333–9343, 2018. 1, 2, 7, 8

[43] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, pages 9627–9636, 2019. 2

[44] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models. In *NeurIPS*, pages 2797–2806, 2017. 1

[45] Hao Wang, Qilong Wang, Fan Yang, Weiqi Zhang, and Wangmeng Zuo. Data augmentation for object detection via progressive and selective instance-switching. *CoRR*, abs/1906.00358, 2019. 7

[46] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 2

[47] Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *ICCV*, 2019. 1

[48] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, pages 9756–9765, 2020. 7, 8

[49] Zhi Zhang, Tong He, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of freebies for training object detection neural networks. *CoRR*, abs/1902.04103, 2019. 2, 7

[50] Xinyue Zhu, Yifan Liu, Zengchang Qin, and Jiahong Li. Data augmentation in emotion classification using generative adversarial networks. *arXiv preprint arXiv:1711.00648*, 2017. 1

[51] Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V. Le. Learning data augmentation strategies for object detection. In *ECCV*, 2020. 1, 2, 3, 4, 5, 6, 7, 8

[52] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. 2

[53] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018. 2