Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation

Golnaz Ghiasi[*1] Yin Cui[*1] Aravind Srinivas[*†1,2]
Rui Qian[†1,3] Tsung-Yi Lin[1] Ekin D. Cubuk[1] Quoc V. Le[1] Barret Zoph[1]

[1]Google Research, Brain Team [2] UC Berkeley [3] Cornell University

Abstract

*Building instance segmentation models that are data-efficient and can handle rare object categories is an important challenge in computer vision. Leveraging data augmentations is a promising direction towards addressing this challenge. Here, we perform a systematic study of the Copy-Paste augmentation (*e.g.*, [13, 12]) for instance segmentation where we randomly paste objects onto an image. Prior studies on Copy-Paste relied on modeling the surrounding visual context for pasting the objects. However, we find that the simple mechanism of pasting objects randomly is good enough and can provide solid gains on top of strong baselines. Furthermore, we show Copy-Paste is additive with semi-supervised methods that leverage extra data through pseudo labeling (*e.g.* *self-training). On COCO instance segmentation, we achieve 49.1 mask AP and 57.3 box AP, an improvement of +0.6 mask AP and +1.5 box AP over the previous state-of-the-art. We further demonstrate that Copy-Paste can lead to significant improvements on the LVIS benchmark. Our baseline model outperforms the LVIS 2020 Challenge winning entry by +3.6 mask AP on rare categories.* [1]

1. Introduction

Instance segmentation [22, 10] is an important task in computer vision with many real world applications. Instance segmentation models based on state-of-the-art convolutional networks [11, 57, 67] are often data-hungry. At the same time, annotating large datasets for instance segmentation [40, 21] is usually expensive and time-consuming. For example, 22 worker hours were spent per

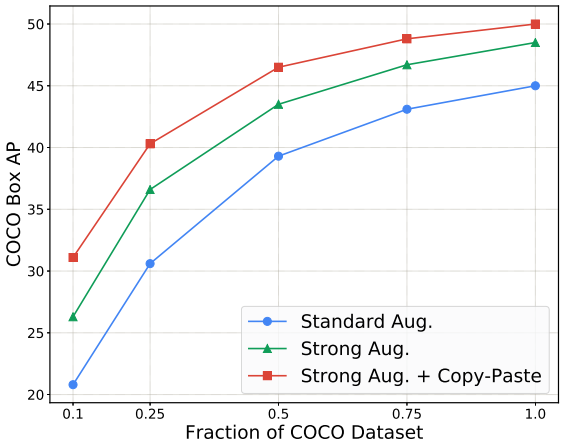


Figure 1. Data-efficiency on the COCO benchmark: Combining the Copy-Paste augmentation along with Strong Aug. (large scale jittering) allows us to train models that are up to 2× more data-efficient than Standard Aug. (standard scale jittering). The augmentations are highly effective and provide gains of +10 AP in the low data regime (10% of data) while still being effective in the high data regime with a gain of +5 AP. Results are for Mask R-CNN EfficientNet-B7 FPN trained on an image size of 640×640.

1000 instance masks for COCO [40]. It is therefore imperative to develop new methods to improve the data-efficiency of state-of-the-art instance segmentation models.

Here, we focus on data augmentation [50] as a simple way to significantly improve the data-efficiency of instance segmentation models. Although many augmentation methods such as scale jittering and random resizing have been widely used [26, 25, 20], they are more general-purpose in nature and have not been designed specifically for instance segmentation. An augmentation procedure that is more *object-aware*, both in terms of category and shape, is likely to be useful for instance segmentation. The Copy-Paste augmentation [13, 12, 15] is well suited for this need. By pasting diverse objects of various scales to new background images, Copy-Paste has the potential to create challenging and novel training data for free.

*Equal contribution. Correspondence to: golnazg@google.com.
†Work done during an internship at Google Research.
[1]Code and checkpoints for our models are available at https://github.com/tensorflow/tpu/tree/master/models/official/detection/projects/copy_paste

Figure 2. We use a simple copy and paste method to create new images for training instance segmentation models. We apply random scale jittering on two random training images and then randomly select a subset of instances from one image to paste onto the other image.

The key idea behind the Copy-Paste augmentation is to paste objects from one image to another image. This can lead to a *combinatorial* number of new training data, with multiple possibilities for: (1) choices of the pair of source image from which instances are copied, and the target image on which they are pasted; (2) choices of object instances to copy from the source image; (3) choices of where to paste the copied instances on the target image. The large variety of options when utilizing this data augmentation method allows for lots of exploration on how to use the technique most effectively. Prior work [12, 15] adopts methods for deciding where to paste the additional objects by modeling the surrounding visual context. In contrast, we find that a simple strategy of randomly picking objects and pasting them at random locations on the target image provides a significant boost on top of baselines across multiple settings. Specifically, it gives solid improvements across a wide range of settings with variability in backbone architecture, extent of scale jittering, training schedule and image size.

In combination with large scale jittering, we show that the Copy-Paste augmentation results in significant gains in the data-efficiency on COCO (Figure 1). In particular, we see a data-efficiency improvement of 2× over the commonly used standard scale jittering data augmentation. We also observe a gain of +10 Box AP on the low-data regime when using only 10% of the COCO training data.

We then show that the Copy-Paste augmentation strategy provides additional gains with self-training [44, 73] wherein we extract instances from ground-truth data and paste them onto unlabeled data annotated with pseudo-labels. Using an EfficientNet-B7 [56] backbone and NAS-FPN [17] architecture, we achieve 57.3 Box AP and 49.1 Mask AP on COCO `test-dev` without test-time augmentations. This result surpasses the previous state-of-the-art instance segmentation models such as SpineNet [11] (46.3 mask AP) and DetectoRS ResNeXt-101-64x4d with test time augmentation [43] (48.5 mask AP). The performance also surpasses state-of-the-art bounding box detection results of EfficientDet-D7x-1536 [57] (55.1 box AP) and YOLOv4-P7-1536 [61] (55.8 box AP) despite using a smaller image size of 1280 instead of 1536.

Finally, we show that the Copy-Paste augmentation results in better features for the two-stage training procedure typically used in the LVIS benchmark [21]. Using Copy-Paste we get improvements of 6.1 and 3.7 mask AP on the rare and common categories, respectively.

The Copy-Paste augmentation strategy is easy to plug into any instance segmentation codebase, can utilize unlabeled images effectively and does not create training or inference overheads. For example, our experiments with Mask-RCNN show that we can drop Copy-Paste into its training, and without any changes, the results can be easily improved, *e.g.*, by +1.0 AP for 48 epochs.

2. Related Work

Data Augmentations. Compared to the volume of work on backbone architectures [35, 51, 53, 27, 56] and detection/segmentation frameworks [19, 18, 47, 38, 26, 39], relatively less attention is paid to data augmentations [50] in the computer vision community. Data augmentations such as random crop [36, 35, 51, 53], color jittering [53], Auto/RandAugment [6, 7] have played a big role in achieving state-of-the-art results on image classification [27, 56], self-supervised learning [28, 24, 5] and semi-supervised learning [64] on the ImageNet [48] benchmark. These augmentations are more general purpose in nature and are mainly used for encoding *invariances to data transformations*, a principle well suited for image classification [48].

Mixing Image Augmentations. In contrast to augmentations that encode invariances to data transformations, there exists a class of augmentations that mix the information

contained in different images with appropriate changes to groundtruth labels. A classic example is the mixup data augmentation [66] method which creates new data points for free from convex combinations of the input pixels and the output labels. There have been adaptations of mixup such as CutMix [65] that pastes rectangular crops of an image instead of mixing all pixels. There have also been applications of mixup and CutMix to object detection [69]. The Mosaic data augmentation method employed in YOLO-v4 [1] is related to CutMix in the sense that one creates a new compound image that is a rectangular grid of multiple individual images along with their ground truths. While mixup, CutMix and Mosaic are useful in combining multiple images or their cropped versions to create new training data, they are still not *object-aware* and have not been designed specifically for the task of instance segmentation.

Copy-Paste Augmentation. A simple way to combine information from multiple images in an *object-aware* manner is to copy instances of objects from one image and paste them onto another image. Copy-Paste is akin to mixup and CutMix but only copying the exact pixels corresponding to an object as opposed to all pixels in the object's bounding box. One key difference in our work compared to Contextual Copy-Paste [12] and InstaBoost [15] is that we do not need to model surrounding visual context to place the copied object instances. A simple random placement strategy works well and yields solid improvements on strong baseline models. Instaboost [15] differs from prior work on Copy-Paste [12] by not pasting instances from other images but rather by jiterring instances that already exist on the image. Cut-Paste-and-Learn [13] proposes to extract object instances, blend and paste them on diverse backgrounds and train on the augmented images in addition to the original dataset. Our work uses the same method with some differences: (1) We do not use geometric transformations (*e.g.* rotation), and find Gaussian blurring of the pasted instances not beneficial; (2) We study Copy-Paste in the context of pasting objects contained in one image into another image already populated with instances where [13] studies Copy-Paste in the context of having a bank of object instances and background scenes to improve performance; (3) We study the efficacy of Copy-Paste in the semi-supervised learning setting by using it in conjunction with self-training. (4) We benchmark and thoroughly study Copy-Paste on the widely used COCO and LVIS datasets while Cut-Paste-and-Learn uses the GMU dataset [16]. A key contribution is that our paper shows the use of Copy-Paste in improving state-of-the-art instance segmentation models on COCO and LVIS.

Instance Segmentation. Instance segmentation [22, 23] is a challenging computer vision problem that attempts to both detect object instances and segment the pixels corresponding to each instance. Mask-RCNN [26] is a widely used framework with most state-of-the-art methods [67, 11, 43] adopting that approach. The COCO dataset is the widely used benchmark for measuring progress. We report state-of-the-art[2] results on the COCO benchmark surpassing SpineNet [11] by 2.8 AP and DetectoRS [43] by 0.6 AP.[3]

Copy and paste approach is also used for weakly supervised instance segmentation. Remez et al. [45] introduce an adversarial approach where it uses a generator network to predict the segmentation mask of an object within a given bounding box. Given the generated mask, the object is blended on another background and then a discriminator network is used to make sure the generated mask/image looks realistic. Different from this work, we use Copy-Paste as an augmentation method.

Long-Tail Visual Recognition. Recently, the computer vision community has begun to focus on the long-tail nature of object categories present in natural images [59, 21], where many of the different object categories have very few labeled images. Modern approaches for addressing long-tail data when training deep networks can be mainly divided into two groups: data re-sampling [41, 21, 62] and loss re-weighting [30, 8, 3, 54, 37, 46]. Other more complicated learning methods (*e.g.*, meta-learning [63, 29, 32], causal inference [58], Bayesian methods [34], *etc.*) are also used to deal with long-tail data. Recent work [9, 3, 33, 71, 37] has pointed out the effectiveness of two-stage training strategies by separating the feature learning and the re-balancing stage, as end-to-end training with re-balancing strategies could be detrimental to feature learning. A more comprehensive summary of data imbalance in object detection can be found in Oksuz *et al*. [42]. Our work demonstrates simple Copy-Paste data augmentation yields significant gains in both single-stage and two-stage training on the LVIS benchmark, especially for rare object categories.

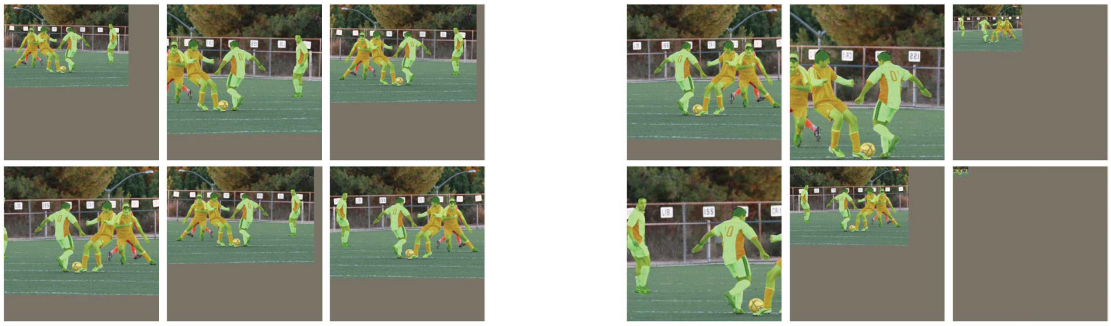
3. Method

Our approach for generating new data using Copy-Paste is very simple. We randomly select two images and apply random scale jittering and random horizontal flipping on each of them. Then we select a random subset of objects from one of the images and paste them onto the other image. Lastly, we adjust the ground-truth annotations accordingly: we remove fully occluded objects and update the masks and bounding boxes of partially occluded objects.

Unlike [15, 12], we do not model the surrounding context and, as a result, generated images can look very different from real images in terms of co-occurrences of objects or related scales of objects. For example, giraffes and

[2]Based on the entries in https://paperswithcode.com/sota/instance-segmentation-on-coco.

[3]We note that better mask / box AP on COCO have been reported in COCO competitions in 2019 - https://cocodataset.org/workshop/coco-mapillary-iccv-2019.html.



(a) **Standard Scale Jittering (SSJ)** (b) **Large Scale Jittering (LSJ)**

Figure 3. Notation and visualization of the two scale jittering augmentation methods used throughout the paper. Standard Scale Jittering (SSJ) resizes and crops an image with a resize range of 0.8 to 1.25 of the original image size. The resize range in Large Scale Jittering (LSJ) is from 0.1 to 2.0 of the original image size. If images are made smaller than their original size, then the images are padded with gray pixel values. Both scale jittering methods also use horizontal flips.

soccer players with very different scales can appear next to each other (see Figure 2).

Blending Pasted Objects. For composing new objects into an image, we compute the binary mask (α) of pasted objects using ground-truth annotations and compute the new image as $I_1 \times \alpha + I_2 \times (1 - \alpha)$ where I_1 is the pasted image and I_2 is the main image. To smooth out the edges of the pasted objects we apply a Gaussian filter to α similar to "blending" in [13]. But unlike [13], we also found that simply composing without any blending has similar performance.

Large Scale Jittering. We use two different types of augmentation methods in conjunction with Copy-Paste throughout the text: standard scale jittering (SSJ) and large scale jittering (LSJ). These methods randomly resize and crop images. See Figure 3 for a graphical illustration of the two methods. In our experiments we observe that the large scale jittering yields significant performance improvements over the standard scale jittering used in most prior works.

Self-training Copy-Paste. In addition to studying Copy-Paste on supervised data, we also experiment with it as a way of incorporating additional unlabeled images. Our self-training Copy-Paste procedure is as follows: (1) train a supervised model with Copy-Paste augmentation on labeled data, (2) generate pseudo labels on unlabeled data, (3) paste ground-truth instances into pseudo labeled and supervised labeled images and train a model on this new data.

4. Experiments

4.1. Experimental Settings

Architecture. We use Mask R-CNN [26] with EfficientNet [56] or ResNet [27] as the backbone architecture. We also employ feature pyramid networks [38] for multi-scale feature fusion. We use pyramid levels from P_2 to P_6, with an anchor size of 8×2^l and 3 anchors per pixel. Our strongest model uses Cascade R-CNN [2], EfficientNet-B7 as the backbone and NAS-FPN [17] as the feature pyramid with levels from P_3 to P_7. The anchor size is 4×2^l and we have 9 anchors per pixel. Our NAS-FPN model uses 5 repeats and we replace convolution layers with ResNet bottleneck blocks [27].

Training Parameters. All models are trained using synchronous batch normalization [31, 20] using a batch size of 256 and weight decay of 4e-5. We use a learning rate of 0.32 and a step learning rate decay [25]. At the beginning of training the learning rate is linearly increased over the first 1000 steps from 0.0032 to 0.32. We decay the learning rate at 0.9, 0.95 and 0.975 fractions of the total number of training steps. We initialize the backbone of our largest model from an ImageNet checkpoint pre-trained with self-training [64] to speed up the training. All other results are from models with random initialization unless otherwise stated. Also, we use large scale jittering augmentation for training the models unless otherwise stated. For all different augmentations and dataset sizes in our experiments we allow each model to train until it converges (*i.e.*, the validation set performance no longer improves). For example, training a model from scratch with large scale jittering and Copy-Paste augmentation requires 576 epochs while training with only standard scale jittering takes 96 epochs. For the self-training experiments we double the batch size to 512 while we keep all the other hyper-parameters the same with the exception of our largest model where we retain the batch size of 256 due to memory constraints.

Dataset. We use the COCO dataset [40] which has 118k training images. For self-training experiments, we use the unlabeled COCO dataset (120k images) and the Objects365 dataset [49] (610k images) as unlabeled images. For transfer learning experiments, we pre-train our models on the COCO dataset and then fine-tune on the Pascal VOC dataset [14]. For semantic segmentation, we train our mod-

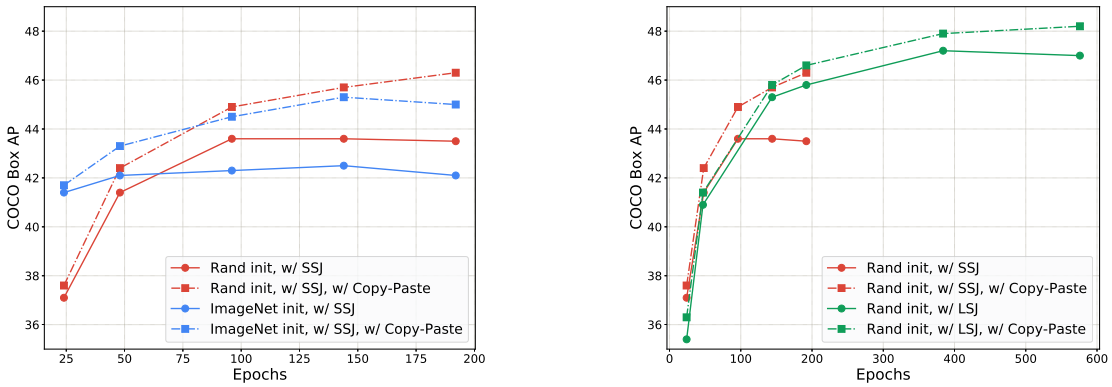


Figure 4. Copy-Paste provides gains that are robust to training configurations. We train Mask R-CNN (ResNet-50 FPN) on 1024×1024 image size for varying numbers of epochs. **Left Figure:** Copy-Paste with and without initializing the backbone by ImageNet pre-training. **Right Figure:** Copy-Paste with standard and large scale jittering. Across all of the configurations training with Copy-Paste is helpful.

els on the `train` set (1.5k images) of the PASCAL VOC 2012 segmentation dataset. For detection, we train on the `trainval` set of PASCAL VOC 2007 and PASCAL VOC 2012. We also benchmark Copy-Paste on LVIS v1.0 (100k training images) and report results on LVIS v1.0 `val` (20k images). LVIS has 1203 classes to simulate the long-tail distribution of classes in natural images.

4.2. Copy-Paste is robust to training configurations

In this section we show that Copy-Paste is a strong data augmentation method that is robust across a variety of training iterations, models and training hyperparameters.

Robustness to backbone initialization. Common practice for training Mask R-CNN is to initialize the backbone with an ImageNet pre-trained checkpoint. However He *et al*. [25] and Zoph *et al*. [73] show that a model trained from random initialization has similar or better performance with longer training. Training models from ImageNet pre-training with strong data-augmentation (*i.e*. RandAugment [7]) was shown to hurt the performance by up to 1 AP on COCO. Figure 4 (left) demonstrates that Copy-Paste is additive in both setups and we get the best result using Copy-Paste augmentation and random initialization.

Robustness to training schedules. A typical training schedule for Mask R-CNN in the literature is only 24 (2×) or 36 epochs (3×) [25, 26, 15]. However, recent work with state-of-the-art results show that longer training is helpful in training object detection models on COCO [73, 57, 11]. Figure 4 shows that we get gains from Copy-Paste for the typical training schedule of 2× or 3× and as we increase training epochs the gain increases. This shows that Copy-Paste is a very practical data augmentation since we do not need a longer training schedule to see the benefit.

Copy-Paste is additive to large scale jittering augmentation. Random scale jittering is a powerful data augmentation that has been used widely in training computer vision models. The standard range of scale jittering in the literature is 0.8 to 1.25 [39, 25, 6, 15]. However, augmenting data with larger scale jittering with a range of 0.1 to 2.0 [57, 11] and longer training significantly improves performance (see Figure 4, right plot). Figure 5 demonstrates that Copy-Paste is additive to both standard and large scale jittering augmentation and we get a higher boost on top of standard scale jittering. On the other hand, as it is shown in Figure 5, mixup [66, 69] data augmentation does not help when it is used with large scale jittering.

Copy-Paste works across backbone architectures and image sizes. Finally, we demonstrate Copy-Paste helps models with standard backbone architecture of ResNet [27] as well the more recent architecture of EfficientNet [56]. We train models with these backbones on the image size of 640×640, 1024×1024 or 1280×1280. Table 1 shows that we get significant improvements over the strong baselines trained with large scale jittering for all the models. Across 6 models with different backbones and images sizes Copy-Paste gives on average a 1.3 box AP and 0.8 mask AP improvement on top of large scale jittering.

4.3. Copy-Paste helps data-efficiency

In this section, we show Copy-Paste is helpful across a variety of dataset sizes and helps data efficiency. Figure 5 reveals that Copy-Paste augmentation is always helpful across all fractions of COCO. Copy-Paste is most helpful in the low data regime (10% of COCO) yielding a 6.9 box AP improvement on top of SSJ and a 4.8 box AP improvement on top of LSJ. On the other hand, mixup is only helpful in a low data regime. Copy-Paste also greatly helps with data-efficiency: a model trained on 75% of COCO with Copy-Paste and LSJ has a similar AP to a model trained on 100% of COCO with LSJ.

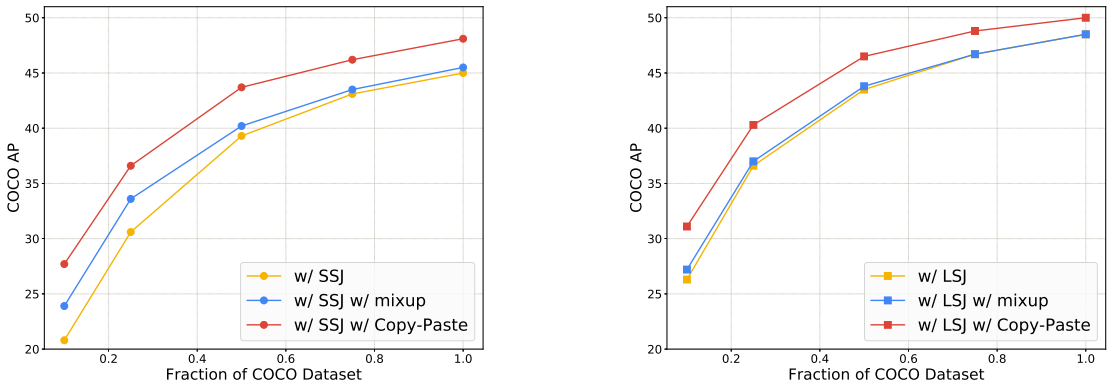


Figure 5. Copy-Paste is additive to large scale jittering augmentation. Improvement from mixup and Copy-Paste data augmentation on top of standard scale jittering **(Left Figure)** and large scale jittering **(Right Figure)**. All results are from training Mask R-CNN EfficientNetB7-FPN on the image size of 640×640.

Model	FLOPs	Box AP	Mask AP
Res-50 FPN (1024)	431 B	47.2	41.8
w/ Copy-Paste	431 B	(+1.0) 48.2	(+0.6) 42.4
Res-101 FPN (1024)	509 B	48.4	42.8
w/ Copy-Paste	509 B	(+1.4) 49.8	(+0.8) 43.6
Res-101 FPN (1280)	693 B	49.1	43.1
w/ Copy-Paste	693 B	(+1.2) 50.3	(+1.1) 44.2
Eff-B7 FPN (640)	286 B	48.5	42.7
w/ Copy-Paste	286 B	(+1.5) 50.0	(+1.0) 43.7
Eff-B7 FPN (1024)	447 B	50.8	44.7
w/ Copy-Paste	447 B	(+1.1) 51.9	(+0.5) 45.2
Eff-B7 FPN (1280)	595 B	51.1	44.8
w/ Copy-Paste	595 B	(+1.5) 52.6	(+1.1) 45.9

Table 1. Copy-paste works well across a variety of different model architectures, model sizes and image resolutions. See table 13 in the Appendix for benchmark results on different object sizes.

Setup	Box AP	Mask AP
Eff-B7 FPN (640)	48.5	42.7
w/ self-training	(+1.5) 50.0	(+1.3) 44.0
w/ Copy-Paste	(+1.5) 50.0	(+1.0) 43.7
w/ self-training Copy-Paste	**(+2.9) 51.4**	**(+2.3) 45.0**

Table 2. Copy-Paste and self-training are additive for utilizing extra unlabeled data. We get significant improvement of 2.9 box AP and 2.3 mask AP by combining self-training and Copy-Paste.

4.4. Copy-Paste and self-training are additive

In this section, we demonstrate that a standard self-training method similar to [64, 73] and Copy-Paste can be combined together to leverage unlabeled data. Copy-Paste and self-training individually have similar gains of 1.5 box AP over the baseline with 48.5 Box AP (see Table 2).

To combine self-training and Copy-Paste we first use a supervised teacher model trained with Copy-Paste to generate pseudo labels on unlabeled data. Next we take ground truth objects from COCO and paste them into pseudo labeled images and COCO images. Finally, we train the student model on all these images. With this setup we achieve 51.4 box AP, an improvement of 2.9 AP over the baseline.

Setup	Pasting into	Box AP	Mask AP
self-training	-	50.0	44.0
+Copy-Paste	COCO	(+0.4) 50.4	44.0
+Copy-Paste	Pseudo data	(+0.8) 50.8	(+0.5) 44.5
+Copy-Paste	COCO & Pseudo data	(+1.4) 51.4	(+1.0) 45.0

Table 3. Pasting ground-truth COCO objects into both COCO and pseudo labeled data gives higher gain in comparison to doing either on its own.

Data to Paste on. In our self-training setup, half of the batch is from supervised COCO data (120k images) and the other half is from pseudo labeled data (110k images from unlabeled COCO and 610k from Objects365). Table 3 presents results when we paste COCO instances on different portions of the training images. Pasting into pseudo labeled data yields larger improvements compared to pasting into COCO. Since the number of images in the pseudo labeled set is larger, using images with more variety as background helps Copy-Paste. We get the maximum gain over self-training (+1.4 box AP) when we paste COCO instances on both COCO and pseudo labeled images.

Data to Copy from. We also explore an alternative way to use Copy-Paste to incorporate extra data by pasting pseudo labeled objects from an unlabeled dataset directly into the COCO labeled dataset. Unfortunately, this setup shows no additional AP improvements.

4.5. Copy-Paste improves COCO state-of-the-art

Next we study if Copy-Paste can improve state-of-the-art instance segmentation methods on COCO. Table 4 shows the results of applying Copy-Paste on top of a strong 54.8 box AP COCO model. This table is meant to serve as

Model	FLOPs	# Params	AP_{val}	$AP_{test\text{-}dev}$	Mask AP_{val}	Mask $AP_{test\text{-}dev}$
SpineNet-190 (1536) [11]	2076B	176M	52.2	52.5	46.1	46.3
DetectoRS ResNeXt-101-64x4d [43]	—	—	—	55.7†	—	48.5 †
SpineNet-190 (1280) [11]	1885B	164M	52.6	52.8	—	—
SpineNet-190 (1280) w/ self-training [72]	1885B	164M	54.2	54.3	—	—
EfficientDet-D7x (1536) [57]	410B	77M	54.4	55.1	—	—
YOLOv4-P7 (1536) [61]	—	—	—	55.8†	—	—
Cascade Eff-B7 NAS-FPN (1280)	1440B	185M	54.5	54.8	46.8	46.9
w/ Copy-Paste	1440B	185M	(+1.4) **55.9**	(+1.2) **56.0**	(+0.4) **47.2**	(+0.5) **47.4**
w/ self-training Copy-Paste	1440B	185M	(+2.5) **57.0**	(+2.5) **57.3**	(+2.1) **48.9**	(+2.2) **49.1**

Table 4. Comparison with the state-of-the-art models on COCO object detection and instance segmentation. Parentheses next to the model name denote the input image size. † indicates results with test time augmentation.

Model	AP50	AP
RefineDet512+ [68]	83.8	-
SNIPER [52]	86.9	-
Cascade Eff-B7 NAS-FPN	88.6	75.0
w/ Copy-Paste pre-training	(+0.7) **89.3**	(+1.5) **76.5**

Table 5. PASCAL VOC 2007 detection result on `test` set. We present results of our EfficientNet-B7 NAS-FPN model pre-trained with and without Copy-Paste on COCO.

Model	mIOU
DeepLabv3+ † [4]	84.6
ExFuse † [70]	85.8
Eff-B7 [73]	85.2
Eff-L2 [73]	88.7
Eff-B7 NAS-FPN	83.9
w/ Copy-Paste pre-training	(+2.7) **86.6**

Table 6. PASCAL VOC 2012 semantic segmentation results on `val` set. We present results of our EfficientNet-B7 NAS-FPN model pre-trained with and without Copy-Paste on COCO. † indicates multi-scale/flip ensembling inference.

a reference for state-of-the-art performance.[4] For rigorous comparisons, we note that models need to be evaluated with the same codebase, training data, and training settings such as learning rate schedule, weight decay, data pre-processing and augmentations, controlling for parameters and FLOPs, architectural regularization [60], training and inference speeds, *etc*. The goal of the table is to show the benefits of the Copy-Paste augmentation and its additive gains with self-training. Our baseline model is a Cascade Mask-RCNN with EfficientNet-B7 backbone and NAS-FPN. We observe an improvement of +1.2 box AP and +0.5 mask AP using Copy-Paste. When combined with self-training using unlabeled COCO and unlabeled Objects365 [49] for pseudo-labeling, we see a further improvement of 2.5 box AP and 2.2 mask AP, resulting in a model with a strong performance of **57.3** box AP and **49.1** mask AP on COCO `test-dev` without test-time augmentations and model ensembling.

4.6. Copy-Paste produces better representations for PASCAL detection and segmentation

Previously we have demonstrated the improved performance that the simple Copy-Paste augmentation provides on instance segmentation. In this section we study the transfer learning performance of the pre-trained instance segmentation models that were trained with Copy-Paste on COCO. Here we perform transfer learning experiments on the PASCAL VOC 2007 dataset. Table 5 shows how the learned Copy-Paste models transfer compared to baseline models on PASCAL detection. Table 6 shows the transfer learning results on PASCAL semantic segmentation as well. On both PASCAL detection and PASCAL semantic segmentation we find our models trained with Copy-Paste transfer better for fine-tuning than the baseline models.

4.7. Copy-Paste provides strong gains on LVIS

We benchmark Copy-Paste on the LVIS dataset to see how it performs on a dataset with a long-tail distribution of 1203 classes. There are two different training paradigms typically used for LVIS: (1) single-stage where a detector is trained directly on the LVIS dataset, (2) two-stage where the model from the first stage is fine-tuned with class re-balancing losses to help handle the class imbalance.

Copy-Paste improves single-stage LVIS training. The single-stage training paradigm is quite similar to our Copy-Paste setup on COCO. In addition to the standard training setup, certain methods are used to handle the class imbalance problem on LVIS. One common method is Repeat Factor Sampling (RFS) from [21], with $t = 0.001$. This method aims at helping the large class imbalance problem on LVIS by over-sampling images that contain less frequent object categories. For single-stage training on LVIS, we follow the same training parameters on COCO to train our models for 180k steps using a 256 batch size. As suggested by [21], we increase the number of detections per image to 300 and reduce the score threshold to 0. Table 8 shows the results of applying Copy-Paste to a strong single-stage LVIS base-

[4] https://paperswithcode.com/sota/object-detection-on-coco

	Mask AP	Mask AP_r	Mask AP_c	Mask AP_f	Box AP
cRT (ResNeXt-101-32×8d) [33]	27.2	19.6	26.0	31.9	—
LVIS Challenge 2020 Winner† [55]	38.8	28.5	39.5	42.7	41.1
ResNet-50 FPN (1024)	30.3	22.2	29.5	34.7	31.5
w/ Copy-Paste	(+2.0) 32.3	(+4.3) 26.5	(+2.3) 31.8	(+0.6) 35.3	(+2.8) 34.3
ResNet-101 FPN (1024)	31.9	24.7	30.5	36.3	33.3
w/ Copy-Paste	(+2.1) 34.0	(+2.7) 27.4	(+3.4) 33.9	(+0.9) 37.2	(+3.1) 36.4
EfficientNet-B7 FPN (1024)	33.7	26.4	33.1	37.6	35.5
w/ Copy-Paste	(+2.3) 36.0	(+3.3) 29.7	(+2.7) 35.8	(+1.3) 38.9	(+3.7) 39.2
EfficientNet-B7 NAS-FPN (1280)	34.7	26.0	33.4	39.8	37.2
w/ Copy-Paste	(+3.4) 38.1	(+6.1) 32.1	(+3.7) 37.1	(+2.1) 41.9	(+4.4) 41.6

Table 7. Comparison with the state-of-the-art models on LVIS v1.0 object detection and instance segmentation. Parentheses next to our models denote the input image size. † We report the 2020 winning entry's result without test-time augmentation.

Setup (single-stage)	AP	AP_r	AP_c	AP_f
Eff-B7 FPN (640)	27.7	9.7	28.1	35.1
w/ RFS	28.2	15.4	27.8	34.3
w/ Copy-Paste	29.3	12.8	**30.1**	**35.7**
w/ RFS w/ Copy-Paste	**30.1**	**18.4**	30.0	35.4

Table 8. Single-stage training results (mask AP) on LVIS.

Setup (two-stage)	AP	AP_r	AP_c	AP_f
Eff-B7 FPN (640)	31.3	25.0	30.6	34.9
w/ RFS	30.1	21.8	29.7	34.1
w/ Copy-Paste	**33.0**	**27.3**	**33.2**	**35.7**
w/ RFS w/ Copy-Paste	32.0	26.3	31.8	34.7

Table 9. Two-stage training results (mask AP) on LVIS.

line of EfficientNet-B7 FPN with 640×640 input size. We observe that Copy-Paste augmentation outperforms RFS on AP, AP_c and AP_f, but under-performs on AP_r (the AP for rare classes). The best overall result comes from combining RFS and Copy-Paste augmentation, achieving a boost of +2.4 AP and +8.7 AP_r.

Copy-Paste improves two-stage LVIS training. Two-stage training is widely adopted to address data imbalance and obtain good performance on LVIS [37, 46, 55]. We aim to study the efficacy of Copy-Paste in this two-stage setup. Our two-stage training is as follows: first we train the object detector with standard training techniques (*i.e.*, same as our single-stage training) and then we fine-tune the model trained in the first stage using the Class-Balanced Loss [8]. The weight for a class is calculated by $(1-\beta)/(1-\beta^n)$, where n is the number of instances of the class and $\beta = 0.999$.[5] During the second stage fine-tuning, we train the model with 3× schedule and only update the final classification layer in Mask R-CNN using the classification loss only. From mask AP results in Table 9, we can see models trained with Copy-Paste learn better features for low-shot classes (+2.3 on AP_r and +2.6 on AP_c). Interestingly, we find RFS, which is quite helpful and additive with Copy-Paste in single-stage training, hurts the performance in two-stage training. A possible explanation for this finding is that features learned with RFS are worse than those learned with the original LVIS dataset. We leave a more detailed investigation of the tradeoffs between RFS and data augmentations in two stage training for future work.

[5] We scale class weights by dividing the mean and then clip their values to $[0.01, 5]$, as suggested by [37].

Comparison with the state-of-the-art. Furthermore, we compare our two-stage models with state-of-the-art methods for LVIS[6] in Table 7. Surprisingly, our smallest model, ResNet-50 FPN, outperforms a strong baseline cRT [33] with ResNeXt-101-32×8d backbone.

EfficientNet-B7 NAS-FPN model (without Cascade [7]) trained with Copy-Paste achieves comparable performance to LVIS challenge 2020 winner on overall Mask AP and Box AP without test-time augmentation. Also, it obtains 32.1 mask AP_r for rare categories, outperforming the LVIS Challenge 2020 winning entry by +3.6 mask AP_r.

5. Conclusion

Data augmentation is at the heart of many vision systems. In this paper, we rigorously studied the Copy-Paste data augmentation method, and found that it is very effective and robust. Copy-Paste performs well across multiple experimental settings and provides significant improvements on top of strong baselines, both on the COCO and LVIS instance segmentation benchmarks.

The Copy-Paste augmentation strategy is simple, easy to plug into any instance segmentation codebase, and does not increase the training cost or inference time. We also showed that Copy-Paste is useful for incorporating extra unlabeled images during training and is additive on top of successful self-training techniques. We hope that the convincing empirical evidence of its benefits make Copy-Paste augmentation a standard augmentation procedure when training instance segmentation models.

[6] https://www.lvisdataset.org/challenge_2020

[7] We find using Cascade in our experiments improves AP_f but hurts AP_r.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.

[3] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, 2019.

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. In *CVPR*, 2019.

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

[8] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019.

[9] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR*, 2018.

[10] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016.

[11] 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*, 2020.

[12] Nikita Dvornik, Julien Mairal, and Cordelia Schmid. Modeling visual context is key to augmenting object detection datasets. In *ECCV*, 2018.

[13] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *ICCV*, 2017.

[14] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[15] Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In *ICCV*, 2019.

[16] Georgios Georgakis, Md Alimoor Reza, Arsalan Mousavian, Phi-Hung Le, and Jana Košecká. Multiview rgb-d dataset for object instance detection. In *3DV*, 2016.

[17] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019.

[18] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

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

[21] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019.

[22] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *ECCV*, 2014.

[23] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[24] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[25] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *ICCV*, 2019.

[26] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[28] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

[29] Xinting Hu, Yi Jiang, Kaihua Tang, Jingyuan Chen, Chunyan Miao, and Hanwang Zhang. Learning to segment the tail. In *CVPR*, 2020.

[30] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *CVPR*, 2016.

[31] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[32] Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong. Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective. In *CVPR*, 2020.

[33] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *ICLR*, 2020.

[34] Salman Khan, Munawar Hayat, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Striking the right balance with uncertainty. In *CVPR*, 2019.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[36] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[37] Yu Li, Tao Wang, Bingyi Kang, Sheng Tang, Chunfeng Wang, Jintao Li, and Jiashi Feng. Overcoming classifier imbalance for long-tail object detection with balanced group softmax. In *CVPR*, 2020.

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

[39] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[40] 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*, 2014.

[41] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

[42] Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Akbas. Imbalance problems in object detection: A review. *TPAMI*, 2020.

[43] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *arXiv preprint arXiv:2006.02334*, 2020.

[44] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. In *CVPR*, 2018.

[45] Tal Remez, Jonathan Huang, and Matthew Brown. Learning to segment via cut-and-paste. In *ECCV*, 2018.

[46] Jiawei Ren, Cunjun Yu, Zhongang Cai, and Haiyu Zhao. Balanced activation for long-tailed visual recognition. In *LVIS Challenge Workshop at ECCV*, 2020.

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[48] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[49] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *ICCV*, 2019.

[50] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 2019.

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[52] Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. In *NeurIPS*, 2018.

[53] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[54] Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. Equalization loss for long-tailed object recognition. In *CVPR*, 2020.

[55] Jingru Tan, Gang Zhang, Hanming Deng, Changbao Wang, Lewei Lu, Quanquan Li, and Jifeng Dai. 1st place solution of lvis challenge 2020: A good box is not a guarantee of a good mask. *arXiv preprint arXiv:2009.01559*, 2020.

[56] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[57] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, 2020.

[58] Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. Long-tailed classification by keeping the good and removing the bad momentum causal effect. *NeurIPS*, 2020.

[59] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018.

[60] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013.

[61] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. *arXiv preprint arXiv:2011.08036*, 2020.

[62] Tao Wang, Yu Li, Bingyi Kang, Junnan Li, Junhao Liew, Sheng Tang, Steven Hoi, and Jiashi Feng. The devil is in classification: A simple framework for long-tail object detection and instance segmentation. In *ECCV*, 2020.

[63] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Meta-learning to detect rare objects. In *ICCV*, 2019.

[64] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *CVPR*, 2020.

[65] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

[66] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[67] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander Smola. Resnest: Split-attention networks. In *arXiv preprint arXiv:2004.08955*, 2020.

[68] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *CVPR*, 2018.

[69] Zhi Zhang, Tong He, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of freebies for training object detection neural networks. *arXiv preprint arXiv:1902.04103*, 2019.

[70] Zhenli Zhang, Xiangyu Zhang, Chao Peng, Xiangyang Xue, and Jian Sun. Exfuse: Enhancing feature fusion for semantic segmentation. In *ECCV*, 2018.

[71] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *CVPR*, 2020.

[72] 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.

[73] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D. Cubuk, and Quoc V. Le. Rethinking pre-training and self-training. In *NeurIPS*, 2020.