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

Golnaz Ghiasi∗, Yin Cui∗, Aravind Srinivas†, Rui Qian†, Tsung-Yi Lin1, Ekin D. Cubuk1, Quoc V. Le1, Barret Zoph1

1Google 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 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.
1Code and checkpoints for our models are available at https://github.com/tensorflow/tpu/tree/master/models/official/detection/projects/copy_paste
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 $2 \times$ 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 YOLOv4 [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 jittering 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 results on the COCO benchmark surpassing SpineNet [11] by 2.8 AP and DetectoRS [43] by 0.6 AP.

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

---


3We 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.
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 ($\alpha$) 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 $\alpha$ 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 $8 \times 2^l$ 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 $4 \times 2^l$ 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 [63] 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-
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 [56, 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 as 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.

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.

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 [56, 69] data augmentation does not help when it is used with large scale jittering.
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>FLOPs</th>
<th>Box AP</th>
<th>Mask AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res-50 FPN (1024) w/ Copy-Paste</td>
<td>431 B</td>
<td>47.2</td>
<td>41.8</td>
</tr>
<tr>
<td>Res-101 FPN (1024) w/ Copy-Paste</td>
<td>509 B</td>
<td>48.4</td>
<td>42.8</td>
</tr>
<tr>
<td>Res-101 FPN (1280) w/ Copy-Paste</td>
<td>693 B</td>
<td>49.1</td>
<td>43.1</td>
</tr>
<tr>
<td>Eff-B7 FPN (640) w/ Copy-Paste</td>
<td>286 B</td>
<td>48.5</td>
<td>42.7</td>
</tr>
<tr>
<td>Eff-B7 FPN (1024) w/ Copy-Paste</td>
<td>447 B</td>
<td>50.8</td>
<td>44.7</td>
</tr>
<tr>
<td>Eff-B7 FPN (1280) w/ Copy-Paste</td>
<td>595 B</td>
<td>51.1</td>
<td>44.8</td>
</tr>
</tbody>
</table>

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.

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.

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
shows the results for rigor-

<table>
<thead>
<tr>
<th>Model</th>
<th>FLOPs</th>
<th># Params</th>
<th>APval</th>
<th>AP_test-dec</th>
<th>Mask APval</th>
<th>Mask AP_test-dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpineNet-190 (1536) [11]</td>
<td>2076B</td>
<td>176M</td>
<td>52.2</td>
<td>52.5</td>
<td>46.1</td>
<td>46.3</td>
</tr>
<tr>
<td>DetectoRS ResNeXt-101-64x4d [43]</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>55.7</td>
<td>—</td>
<td>48.5</td>
</tr>
<tr>
<td>SpineNet-190 (1280) [11]</td>
<td>1885B</td>
<td>164M</td>
<td>52.6</td>
<td>52.8</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SpineNet-190 (1280) w/ self-training [72]</td>
<td>1885B</td>
<td>164M</td>
<td>54.2</td>
<td>54.3</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>EfficientDet-D7x (1536) [57]</td>
<td>410B</td>
<td>77M</td>
<td>54.4</td>
<td>55.1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>YOLOv4-P7 (1536) [61]</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>55.8</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Cascade Eff-B7 NAS-FPN (1280) w/ Copy-Paste</td>
<td>1440B</td>
<td>185M</td>
<td>(1.4) 55.9</td>
<td>(1.2) 56.0</td>
<td>(0.4) 47.2</td>
<td>(0.5) 47.4</td>
</tr>
<tr>
<td>Cascade Eff-B7 NAS-FPN (1280) w/ self-training Copy-Paste</td>
<td>1440B</td>
<td>185M</td>
<td>(2.5) 57.0</td>
<td>(2.5) 57.3</td>
<td>(2.1) 48.9</td>
<td>(2.2) 49.1</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>AP50</th>
<th>AP*</th>
</tr>
</thead>
<tbody>
<tr>
<td>RefineDet512+ [68]</td>
<td>83.8</td>
<td>—</td>
</tr>
<tr>
<td>SNIPER [52]</td>
<td>86.9</td>
<td>—</td>
</tr>
<tr>
<td>Cascade Eff-B7 NAS-FPN w/ Copy-Paste pre-training</td>
<td>(1.4) 89.3</td>
<td>(1.5) 76.5</td>
</tr>
</tbody>
</table>

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.

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 rebalancing losses to help handle the class imbalance.

### 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.

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-
We observe that Copy-Paste augmentation outperforms RFS on AP, AP$_c$ and AP$_r$, 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$_c$.

**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 $\beta = 0.999$. During the second stage fine-tuning, we train the model with a $\times 3$ 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$_c$ and +2.6 on AP$_r$). 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.

5We 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$^5$ 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) 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.
References


[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.


