Learning to Segment via Cut-and-Paste

Tal Remez, Jonathan Huang, and Matthew Brown

Google
{talremez,jonathanhuang,mtbr}@google.com

Abstract. This paper presents a weakly-supervised approach to object instance segmentation. Starting with known or predicted object bounding boxes, we learn object masks by playing a game of cut-and-paste in an adversarial learning setup. A mask generator takes a detection box and Faster R-CNN features, and constructs a segmentation mask that is used to cut-and-paste the object into a new image location. The discriminator tries to distinguish between real objects, and those cut and pasted via the generator, giving a learning signal that leads to improved object masks. We verify our method experimentally using Cityscapes, COCO, and aerial image datasets, learning to segment objects without ever having seen a mask in training. Our method exceeds the performance of existing weakly supervised methods, without requiring hand-tuned segment proposals, and reaches 90% of supervised performance.

Keywords: Instance segmentation, weakly-supervised, deep-learning.

1 Introduction

Instance segmentation has seen much progress in recent years, with methods such as Mask R-CNN [1] now able to generate realistic masks, by building on the success of convolutional object detectors [2,3]. Success has come at the cost of a significant labelling effort; the COCO segmentation dataset [4] required around 40 person-years of labelling time for its 80 object categories.

Modern object detection datasets have bounding boxes for up to 30k categories [5]. While still a considerable labelling effort, these bounding boxes can

(a) (b) (c) (d) (e)

Fig. 1. Learning to Segment by Cut and Paste. We iterate to learn accurate segmentation masks by trying to generate realistic images. A poor mask estimate (a) generates an unconvincing paste (b), while a good mask (c) results in a convincing one (d). Training a discriminator network to distinguish pasted from real images (e) creates a learning signal that encourages the generator to create better segmentations.

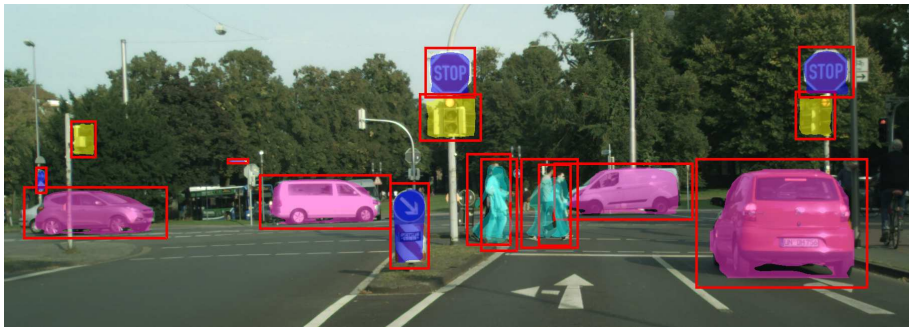

Fig. 2. Our method learns to segment objects without ever seeing ground truth masks and uses only bounding boxes as input.

be generated roughly 10 times faster than the per-pixel segmentation masks required for fully supervised instance segmentation training. Moreover, labelling boxes has fixed complexity, whereas pixel-level labelling takes longer for objects with complex boundaries. In the COCO dataset, for example, some complex object classes, such as 'bicycle', are at best approximately labelled (see Figure 15 in [4]). This motivates the question we address in this paper (Figure 2): can we learn instance segmentation directly from bounding box data, and without ground truth masks? We will propose a simple idea, which we call the "cut-and-paste prior", to solve this problem (see Figure 1).

We are not the first to address the problem of box-supervised instance or semantic segmentation. Dai et al. propose a box-supervised method that uses an unsupervised candidate mask generator to create regression targets for semantic segmentation [6]. The process is then iterated with the learned mask generator. Papandreou et al [7] propose a similar alternation, but using EM to calculate the pixel labels (E-step), and optimise the segmentation network parameters (M-step). "Simple Does It" by Khoreva et al. [8] also follow Dai et al. in proposing synthetic regression targets. They experiment with using detection boxes alone, as well as Grabcut [9] and MCG [10] proposal generators, and also extend their approach to instance segmentation (using just 1 iteration in this case). Deepcut [11] propose a modification to Grabcut using a CNN+CRF model. All of these approaches involve initialization and iteration between label estimation and generator training stages. Pathak et al. [12] take a different approach, specifying hand-tuned constraints on the output label space (e.g., 75% of pixels taking on the true label). Also related is [13], who grow object segmentation regions by sequentially discovering and erasing discriminative image regions, using classification rather than box supervision.

Our approach is qualitatively different from all these prior approaches. We require no segment proposals, pre-trained boundary detectors, or other hand-tuned initialization/constraints. Neither do we require iteration towards prediction and label consistency. Instead, our priors will be encapsulated in the structure of our generator/discriminator networks, and in our "cut-and-paste" prior for object

segmentation. The cut-and-paste prior encapsulates the basic idea that objects can move independently of their background. More precisely, objects may be cut out from one portion of an image, and pasted into another, and still appear realistic (see Figure 1). With the help of a discriminator network to judge realism, we can use this process to provide a training signal for an instance segmentation network.

We build on the successful approach of Generative Adversarial Networks (GANs) [14], which have proved to be effective in modelling realistic images, e.g., hallucinating faces [15] and translating between image modalities [16]. However, rather than trying to generate images, we aim to generate segmentation masks. This allows us to use objective measures of performance (e.g., IoU against ground truth) for evaluation. Related to our approach is the work of Luc et al. [17], who also use an adversarial network to train a (semantic) segmentation algorithm. However, different to our approach, they use ground truth label maps as input to the discriminator. In our work we assume no such ground truth is available at training time. Also related is the work of Hu et al. [18] who use a partially supervised approach to generate object masks for a very large set of categories. They achieve this by joint learning using a set of fully supervised object classes and a larger set of box-only supervised classes, with a transfer function to map between box estimation and mask segmentation parameters. This seems to be a promising approach. However, in this work we focus on the unsupervised case, with only bounding boxes available for training.

Note that our approach of using cut-and-paste to form a loss function is *not* the same as training data augmentation via cut-and-paste, e.g., [19], which takes existing masks and creates more training data out of it. This and related methods [20,21] do however exploit the same idea that image compositing can be used to create realistic imagery for training. They also note that object placement, lighting etc. are important; we revisit this topic in Sections 2.1 and 4.3.

1.1 Contributions

The main contributions of this paper can be summarized as follows:

- We propose and formalize a new *cut-and-paste* adversarial training scheme for box-supervised instance segmentation, which captures an intuitive prior, that objects can move independently of their background.
- We discuss the problem of identifying *where* to paste new objects in an image. Even though objects are rarely truly independent of their background (e.g., cars do not typically appear in the middle of blue skies or on top of trees), we show that simple randomized heuristics for selecting pasting locations are surprisingly effective on real data.
- Finally we showcase the success and generality of our approach by demonstrating that our method effectively learns to segment objects on a variety of datasets (street scenes, everyday objects, aerial imagery), without ever having access to masks as supervision. We also show that our training method is stable and yields models that outperform existing weakly supervised methods, reaching 90% of supervised model performance.

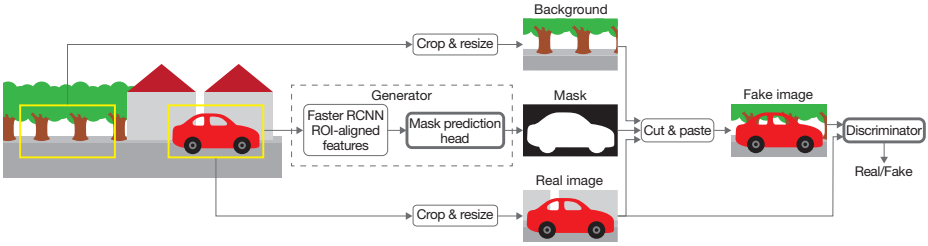

Fig. 3. Learning by cut-and-paste. A generator network receives a bounding box containing a car and predicts its mask. The discriminator alternately sees a cut+pasted car with a new background, or a real car image. Simultaneous training of generator and discriminator leads to improved object masks. Trainable blocks are outlined in **bold**.

2 An Adversarial Formulation of the Cut and Paste Loss

An overview of our learning approach is shown in Figure 3. We wish to train a model taking the form: $\mathcal{M} = G(X, \mathcal{B})$ that predicts an instance mask $\mathcal{M}$ given an image X and a bounding box $\mathcal{B}$ surrounding the instance of interest. For simplicity we will ignore classes and typically assume that instances are of the same class (e.g., 'person' or 'car'), training an independent model per class. Intuitively, we would like to assign a low loss to a predicted mask if copying the pixels from the mask $\mathcal{M}$ and pasting into a new part of the image X yields a *plausible* image patch and high loss otherwise (see Figure 1).

In order to measure the notion of "plausibility", we use a GAN, viewing the function G as a *generator*. Given a generated mask $\mathcal{M}$, we synthesize a new image patch F by compositing image $X_{\mathcal{B}}$ from bounding box $\mathcal{B}$ with a new background image $X_{\mathcal{B}'}$ from location $\mathcal{B}'$ (typically in the same image):

$$F = \mathcal{M}X_{\mathcal{B}} + (1 - \mathcal{M})X_{\mathcal{B}'}. \tag{1}$$

The fake image F is fed to a second model, the *discriminator*, whose job is to distinguish whether F is real or synthesized. We now simultaneously train the discriminator to distinguish reals from fakes and the generator to make the discriminator's error rate as high as possible. More formally, we maximize with respect to parameters of the discriminator D and minimize with respect to parameters of the generator G in the following loss function:

$$\mathcal{L}_{CPGAN} = \mathbb{E}\ \log D(X_{\mathcal{B}}) + \log(1 - D(F)). \tag{2}$$

We refer to this as the *cut-and-paste* loss, since it aims to align real images and their cut-and-pasted counterparts. Note that the fake image F is a function of the generator G via the mask $\mathcal{M} = G(X, \mathcal{B})$, as specified in Equation 1. The expectations are over $(X, \mathcal{B}) \sim p_{data}$ being the input set of images and bounding boxes, with $\mathcal{B}'$ drawn randomly as described in the Section 2.1 below.

(a) (b)

Fig. 4. Cut-and-paste locations. A few objects have been cut-and-pasted to new locations in the original image. In (a) they were pasted along the same scanline as their original position, making it harder to tell them apart (can you spot them?); In (b) they were pasted at random positions making it much easier.

Over training iterations, the hope is that the only way that the generator can successfully "fool" the discriminator is by generating correct masks. We now discuss several critical stepping stones to get such a model to train effectively.

2.1 Where to Paste

The choice of where to paste an object to generate a realistic looking result is clearly important for human observers, e.g., see Figure 4. It is also data dependent. For example, buildings may appear at any (x, y) location in our aerial imagery with equal probability (Figure 11), but realistic pedestrian placement and scale is highly constrained in street scenes. Whilst sophisticated pasting strategies might be devised, we find that good results can still be obtained using simple ones. In this work we experiment with two main pasting strategies: 1) Uniform pasting: paste anywhere into the same image, taking care not to overlap the same object class, 2) Depth sensitive pasting: we take care to preserve the correct scale when pasting using knowledge of the scene geometry. We discuss this further in the experiments reported in Section 4.3 on the Cityscapes, COCO, and aerial imagery datasets.

2.2 Avoiding Degenerate Solutions

Our learning objective is based on realism in the pasted result; this strategy usually leads to good solutions, but there are a few degenerate cases. For example, realistic images can be generated by choosing all of the pixels or none of the pixels in the bounding box (though in the latter case this doesn't contain the object). Also, some objects are modular and part of the object can be pasted and still give a realistic image. We examine each of these cases in turn.

The first case (generator marks all pixels as foreground) can be mitigated by giving the discriminator a larger viewport than the region into which the generator pastes. Giving the discriminator a small band of context around the pasted object (typically 10% of box width) allows for easy identification of this failure mode, as the background will change abruptly at the bounding box borders.[1]

[1] Note that this strategy will fail in cases where source and destination backgrounds are identical, e.g., pasting an airplane from one blue sky to another, but these cases are rare for most classes.

If the generator decides to label none of the pixels as belonging to the object, the resulting fake image will look realistic, but will not contain the object of interest. This case should be automatically solved in our framework, since the discriminator will expect to see the object present. However, we found that adding an explicit classification loss significantly aids stability and improves performance in some cases. To this end we add an additional classification network D_{CLS} which explicitly encourages the model to ensure that the object of interest is really present (see Figure 10(b)). One way to think about this new loss is that our generator is now trying to fool two discriminators: one that had been trained on a previous classification task (and is frozen), and another that is training and evolving with the generator. This gives an additional classification loss term for the generator:

$$\mathcal{L}_{CLS} = \mathbb{E}\ \log(1 - D_{CLS}(F)). \tag{3}$$

A final failure mode can occur if the generator chooses to paste some subpart of an object that may still be realistic in isolation, e.g., part of a building or other modular structure. This is to some extent addressed by the classification loss $\mathcal{L}_{CLS}$, which favours complete objects being pasted.

2.3 Overall Loss Function

Our overall loss function is the sum of the cut+paste and classification losses:

$$\mathcal{L} = \mathcal{L}_{CPGAN} + w_{cls}\mathcal{L}_{CLS}. \tag{4}$$

In practice, we use a LSGAN [22] formulation, which converts min/max optimization of GAN loss terms of the form $\mathcal{L} = \mathbb{E}\ \log(1-D(X)) + \log(1-D(G(X)))$ into separate optimizations for the discriminator and generator:

$$\min_D\ \mathbb{E}\ (D(G(X))^2 + (D(X) - 1)^2,\ \ \min_G\ \mathbb{E}\ (D(G(X) - 1)^2. \tag{5}$$

3 Architecture

There are three modules in our model: (1) the generator, which predicts a mask, (2) the cut-and-paste module, which produces a "fake patch" given the predicted mask, and (3), the discriminator, which distinguishes between real and fake patches, see Figure 3. In the following, we describe the architecture for each of these modules that we have used in our experiments.

Generator: Our generator is similar to that of Mask R-CNN [1]. A ResNet-50 backbone is used to extract ROI-aligned features and a mask prediction head is applied to these features. Our mask prediction head is described in Table 1, and is comprised of a series of convolutions, bilinear upsampling operations, and a Sigmoid nonlinearity resulting in a 28×28 mask output. We find that using corner-aligned bilinear upsampling generally provides better results than transposed convolutions and nearest neighbour upsampling layers.

Generator

Output size	Layer
7×7×2048	Input, ROI-aligned features
7×7×256	Conv, 1×1 × 256, stride 1
7×7×256	Conv, 3×3×256, stride 1
14×14×256	Bilinear upsampling
14×14×256	Conv, 3×3×256, stride 1
28×28×256	Bilinear upsampling
28×28×256	Conv, 3×3×256, stride 1
28×28×1	Conv, 3×3×1, stride 1
28×28×1	Sigmoid

Discriminator

Output size	Layer
34×34×3	Input image patch
32×32×64	Conv, 3×3×64, stride 1, valid
15×15×128	Conv, 3×3×128, stride 2, valid
7×7×256	Conv, 3×3×256, stride 2, valid
3×3×512	Conv, 3×3×512, stride 2, valid
4608	Flatten
2	Fully connected
2	Softmax

Table 1. Generator and discriminator architectures. Our generator takes ROI-aligned features from a Faster R-CNN detector and applies a mask prediction head similar to that used in Mask R-CNN [1]. Our discriminator is applied directly on 34×34 image patches. After each convolution we use ReLU nonlinearities for the generator and Leaky ReLUs (with $\alpha = 0.2$) for the discriminator.

Cut-and-Paste Module: We implement the cut-and-paste operation using standard alpha compositing (Equation 1). The inferred mask is typically at a lower resolution than the foreground and background images, so we downsample to the mask resolution before compositing. Note that careful sampling in this step is critical, as convolutional networks can easily detect any aliasing or blurring artifacts, which are easy indicators that an image is fake. As explained in Section 2.2, we allow the discriminator a larger viewport than the original mask size, therefore our 28×28 masks are padded with 3 pixels of zeros on each side.

Discriminator: Our discriminator receives an $N \times N$ image patch as input, and predicts whether the given patch is real or fake. Our discriminator architecture is presented in Table 1, and is comprised of a series of valid convolutions (convolutions without padding) followed by a fully connected layer and a Softmax.

Training Procedure: Our models are implemented in TensorFlow [23] and are trained using a batch size of 4 instances for the generator and 8 instances for the discriminator (4 real and 4 fake). We use the Adam optimizer [24] with learning rate of $5 \cdot 10^{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We train for 1 million iterations, alternating optimization equally between generator and discriminator. Our supervised model is trained similarly but with a cross-entropy loss on the ground truth masks. The backbone generating the features for our generator was pretrained on the COCO detection challenge data and is held frozen through training. The rest of the generator and discriminator layers are initialized using random Xavier initialization [25]. CityScapes and COCO training data are augmented by adding random horizontal flips.

4 Experiments

In this section we present the results of experiments using street scenes (Cityscapes), common objects (COCO) and aerial image datasets. Overall results (Tables 2 and 3) indicate that our models are competitive or better than other weakly supervised baselines. We also investigate some of the strengths and failure modes of our approach, including analysing dataset specific performance, effect of pasting strategies, settings for loss hyperparameters, and the effect of data scaling.

Method	Car	Person	Traffic-light	Traffic-sign
Box	0.62	0.49	0.76	0.79
GrabCut [9]	0.62	0.50	0.64	0.65
Simple Does It [8]	**0.68**	0.53	0.60	0.51
Cut&Paste (Ours)	0.67	**0.54**	**0.77**	**0.79**
FullySupervised	0.80	0.61	0.79	0.81

Table 2. mIOU performance on Cityscapes

4.1 Evaluation Methodology and Baselines.

We compare our proposed approach (which we will refer to in below tables as **Cut&Paste**) to a few baseline methods, all of which take as input (1) an image and (2) a bounding box surrounding the instance to be segmented, and output a segmentation mask. The simplest baseline strategy (which we call **Box**) is to simply declare all pixels within the given ground truth bounding box to be the foreground/object. Since bounding boxes are tight around the objects in the datasets that we use, this is often a reasonable guess, assuming that no additional information is available. Another baseline is the **GrabCut** [9] algorithm. We use 5 iterations of the OpenCV implementation, guiding with a central foreground rectangle 40% of the box size if the initial iterations return a zero-mask.

We also evaluate the performance of the recent **Simple Does It** approach by Khoreva et al., [8] by running their publicly available pretrained instance segmentation model $DeepLab_{BOX}$, which was trained on PASCAL VOC [26] and COCO.

In addition to these baselines, we also train a fully supervised version of our model (called **FullySupervised**), which uses the same architecture as our generator, but is trained using cross entropy loss against ground truth masks. This gives us an idea of the best performance we should expect from our weakly supervised methods.

For methods outputting low-resolution masks (this includes **Cut&Paste**, **FullySupervised**, and **Simple Does It**), we resize their masks using bicubic interpolation back to the original image resolution prior to evaluation.

In contrast to typical generative models of images based on GANs, we can evaluate our method based on objective measures. We present results in this section in terms of the mean intersection-over-union ($mIoU$) measure, a commonly used metric for segmentation. Since our bounding boxes are assumed to be given, we refrain from presenting average precision/recall based measures such as those used by the COCO dataset since they depend on the detected boxes.

4.2 CityScapes

The CityScapes dataset [27] consists of densely annotated imagery of street scenes from cameras mounted on a car. Images are usually wide enough that it

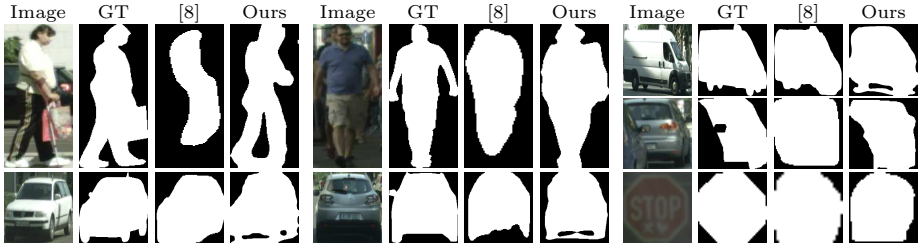

Fig. 5. Cityscapes mask comparison. From left to right: the original image, the ground truth mask (GT), the mask predicted by Simple Does It [8], and our mask.

Fig. 6. Cityscapes examples. Masks produced by our method.

is easy to find plausible pasting positions for fake objects, so one would expect our method to perform well on this kind of data.

To prepare the CityScapes data for training our models, we separate the official training set into a training set and a development set (the latter containing the sequences of Aachen, Bremen and Bochum), using the official validation set as a test set for all methods. We extract instance segmentation masks from the fine-grained annotated "left-camera" images for four classes: "car", "person", "traffic-light" and "traffic-sign"[2], removing any cars or people smaller than 100 pixels along either axis, and any traffic lights or signs smaller than 25 pixels. Ground truth instance segmentations are used for evaluation, and for training the supervised version of our model. For the box-supervised version, we use a combination of ground truth bounding boxes from the $2,975$ annotated images, and additional bounding boxes generated by running a Faster R-CNN object detector[3] on the $89,240$ unannotated images in the `leftImg8bit_sequence` set.

Our results, shown in Table 2, demonstrate that across all four classes, we are consistently better than the **Box** and **GrabCut** baselines. Note that **Box**

[2] "Traffic-light" and "Traffic-sign" instance segmentation masks are not provided with the dataset, but semantic segmentation masks *are* provided; thus to extract masks, we consider each connected component of these classes as a separate instance.

[3] The detector is pretrained on the COCO dataset, the model can be found in TensorFlow Object Detection API model zoo: `https://github.com/tensorflow/models/blob/master/research/object_detection/`

performs suprisingly well on some of these classes, notably signs and traffic lights, for which the ground truth bounding box is typically already a good fit. We also outperform the **Simple Does It** approach and are within 90% of our fully supervised baseline on all but the "Car" class. Figure 5 shows a qualitative comparison between masks generated by our method and those by **Simple Does It**. Typically the masks from both methods are comparable in quality, except in the case of people where our **Cut&Paste** method performs noticeably better, especially in fine details such as arms and legs. Figure 6 presents more examples of our masks. These results used the $\mathcal{L}_{CPGAN}$ loss, with zero weight to the classification loss term ($w_{cls} = 0$). See Section 4.6 for discussion and results of loss term weightings. All methods were evaluated on images at 600×1200 resolution.

Figure 7 shows "fake images" created by cut-and-pasting objects using our generated masks. Generally, the task of generating a realistic composite is well aligned with accurate object segmentation, but there are examples where this is not the case. One such example is the shadows beneath cars, which are important to include in order to synthesize realistic images, but not actually part of the object.

4.3 Effect of Pasting Strategy

The CityScapes dataset contains object instances at a wide variety of scales corresponding to the wide range of scene depth. For realistic results, it is important to paste objects at the appropriate scale (see Figure 4). A simple heuristic to achieve this is to paste the object along the same horizontal scanline. We experiment with this approach, shifting with a mean translation of $2 \times W$ and standard deviation W (disallowing overlaps), where W is the bounding box width. This strategy leads to a 4% absolute increase in per-pixel mask prediction accuracy (from 68% to 72%), when compared to uniformly pasting objects along both the horizontal and vertical axes. As a sanity check, we also tried pasting Cityscape images into random COCO images for training. This reduced the accuracy to 60% on average and the training process was less stable.

4.4 Sampling Issues for the Discriminator Network

Convolutional networks are highly sensitive to low-level image statistics, and unintended subtle cues may allow them to "cheat", rather than solving the intended problem. An example is described in [28], where a convnet used chromatic aberration cues to judge image position. We find a similar effect with sampling artifacts in our approach. In particular, we find that pasting with a mask at lower resolution than the source/destination images leads to a significant drop in performance. In our final implementation we perform compositing at the resolution of the mask. If we instead attempt to composite at $2\times$ this resolution, we observe that the performance decreases from 71% to 66% in terms of per-pixel mask accuracy. We hypothesize that the discriminator picks up on the additional blurring incurred by the lower resolution mask in real vs fake images in this case. This suggests that careful image processing is important when dealing with adversarial networks.

Fig. 7. Examples of Cityscapes images and masks generated by our method. The top row shows the original image, and the middle row is the fake generated by compositing onto a random background with the inferred mask (bottom row).

Method	Person	Chair	Car	Cup	Bottle	Book	Bowl	Handbag	Potted plant	Umbrella	**All**
Box	0.53	0.54	0.64	0.75	0.67	0.58	0.70	0.52	0.58	0.51	0.57
GrabCut [9]	0.57	0.54	0.59	0.70	0.62	0.58	0.69	0.53	0.57	**0.63**	0.61
Simple Does It [8]	**0.60**	**0.56**	0.62	0.72	0.67	0.55	0.72	0.54	0.62	0.61	0.62
Cut&Paste (Ours)	**0.60**	**0.56**	**0.66**	**0.78**	**0.74**	**0.61**	**0.77**	**0.58**	**0.65**	0.61	**0.64**
FullySupervised	0.70	0.63	0.75	0.83	0.79	0.67	0.81	0.63	0.70	0.67	0.70

Table 3. mIoU performance on the 10 most common COCO categories. The final column shows average performance across all 80 categories.

4.5 COCO

The COCO dataset [4] contains a much wider variety of scene content and geometry than our CityScapes and aerial imagery experiments, and the objects typically occupy a much larger fraction of the image. Whilst these appear to be more difficult conditions for our cut+paste approach, we find that our method still works well.

Since our method requires an object to be pastable within the same image at a new position, we remove objects that are more than 30% of image width as well as very small objects (less than 14 pixels). This results in removing 36% of the total number of objects, approximately half of which are too small and half too large. For all instances, we define the ground truth bounding box as the tightest axis-aligned box that encloses the instance mask. We set aside 15% of the official training set as a development set.

Table 3 presents the results for the 10 most common COCO classes, and summary results for all 80 classes. These models were trained using $w_{cls} = 0$. Our method exceeds the performance of **GrabCut** in all cases, and **Simple Does It** [8] in 70% of all COCO classes. We perform particularly well in comparison to [2] on "baseball bat" (0.43 vs 0.32 mIoU) and "skis" (0.27 vs 0.23 mIoU). These objects occupy a small fraction of the bounding box, which is problematic for [8], but fine for our method. We perform less well on "kite" (0.51 vs 0.56 mIoU) and "airplane" (0.48 vs 0.55). This is perhaps due to the uniform backgrounds that are common for these classes, which will reduce the training signal we see from the cut-and-paste operation (the boundary is arbitrary when pasting with

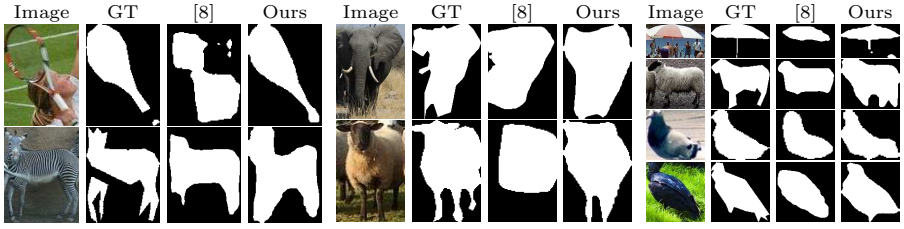

Fig. 8. COCO examples. From left to right: the original image, the ground truth mask (GT), the mask predicted by Simple Does It [8], and our mask.

Fig. 9. COCO examples. Examples of the masks produced by our method.

identical backgrounds). See Figures 8 and 9 for examples of masks produced by our method and by **Simple Does It**.

4.6 Aerial Imagery

To demonstrate the effectiveness of our method in a different setting, we experiment with building segmentation using a proprietary dataset of aerial images consisting of 1000×1000 image tiles with annotated building masks. From this dataset, we select a subset of images each of which contain no more than 15 houses (in order to allow space in the same image for pasting), yielding a dataset with 1 million instances. We also similarly generate a validation set containing 2000 instances. The large size of this dataset also allows us to test performance gains as a function dataset size.

For these experiments, we trained a Faster R-CNN Inception Resnet v2 (atrous) house detector using the TensorFlow Object Detection API [3] to be used as a backbone for feature extraction. Since our aerial images are taken at a single scale and orthorectified, we paste objects into images at locations selected uniformly at random in both x and y directions, rejecting pasting locations that overlap with other bounding boxes in the image.

Effect of Dataset Scale. Figure 10(a) shows the effect of data size on the average performance of our models. Increasing data size helps the training process,

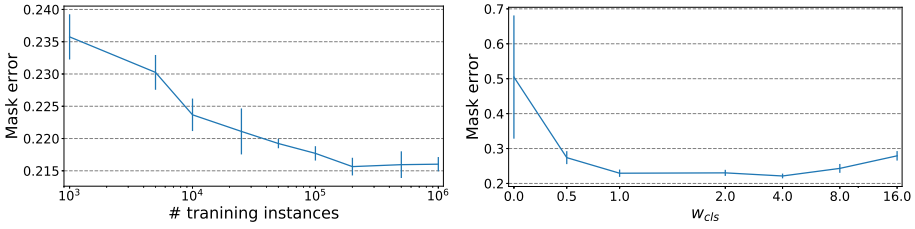

Fig. 10. Effect of classification loss and dataset scale on mask accuracy. (a) demonstrates the effect of scale of the training set on mask accuracy for aerial imagery. (b) demonstrates the effect of classification loss weight w_{cls} on the accuracy of the predicted masks; curves show mean and standard deviation over 4 runs for (a) and 5 models for (b).

Fig. 11. Aerial imagery examples. Examples of masks produced by our method.

increasing the number of training instances from 5K to 1M reduces the mask prediction error by about 10%.

Effect of Loss Weightings. Figure 10(b) shows the effect of the classification loss weight w_{cls} on the overall performance of the model. With no classification loss ($w_{cls} = 0$) the performance is poor and the model is unstable, as indicated by the error bars. With increasing classification loss, performance improves and the error bars become tighter showing the training process is much more stable. The optimal weight in this case is in the range of $w_{cls} \in [1, 4]$. When conducting a similar experiment for the Cityscapes dataset we found that the classification weight increases stability but does not improve performance overall. This may be due to the high incidence of occlusion in our aerial image data, e.g., a subsection of a mask often resembles a building occluded by trees. Figure 11 shows a few examples of typical aerial images and the segmentation masks our method produces when trained using $w_{cls} = 1$.

4.7 Failure Cases

A few failure cases of our method are presented in Figure 12. For the Giraffe and Kite examples, the mask is overestimated, but the non-unique backgrounds lead to convincing fake images. Note that the shadow of the Giraffe is copied in the first case, another common failure mode. Other examples include missing or

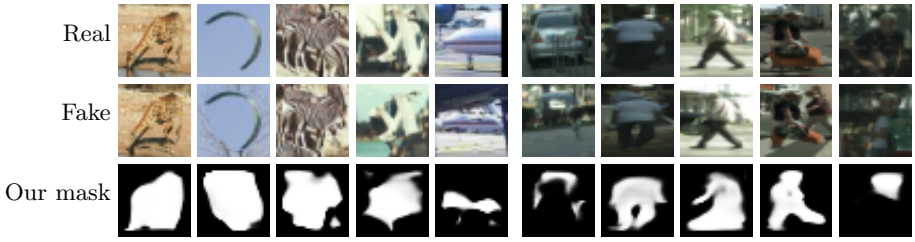

Fig. 12. Failures. COCO images are on the left and Cityscapes are on the right.

added parts, such as the missing head and extra "skirt" connecting the legs for the people towards the right of the figure. Next to these is an interesting example with a person carrying a suitcase. The generator decides it would be more realistic to change this to legs in this example, presumably because suitcases are a rare occurrence in the training set. Note that the "fake" images are often still realistic in many of these cases.

5 Conclusions

We have presented a new approach to instance segmentation that uses a simple property of objects, namely that they are "cut-and-pastable", coupled with a generative adversarial network to learn object masks. Our method exceeds the performance of existing box-supervised methods on the CityScapes and COCO datasets, with no mask ground truth and without the need for pre-trained segment or boundary detectors.

We have shown that intelligent object placement in the paste step can significantly improve mask estimation. This suggests an interesting direction for future work, where the compositing step is also data-dependent. For example, object placement, colour and illumination could depend on the destination image. Related work shows this works well for data augmentation [19,20,21].

More generally, and as in work such as [29], we could envisage a range of settings where vision+graphics imitate photography, with adversarial losses to jointly optimise image understanding and rendering stages. Such methods could open up the possibility of performing detailed visual perception with reduced dependence on large-scale supervised datasets.

References

1. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: IEEE International Conference on Computer Vision (ICCV). (2017) 2980–2988
2. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems. (2015) 91–99
3. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: IEEE CVPR. (2017)
4. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: European Conference on Computer Vision, Springer (2014) 740–755
5. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., et al.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision **123**(1) (2017) 32–73
6. Dai, J., He, K., Sun, J.: Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1635–1643
7. Papandreou, G., Chen, L.C., Murphy, K.P., Yuille, A.L.: Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1742–1750
8. Khoreva, A., Benenson, R., Hosang, J., Hein, M., Schiele, B.: Simple does it: Weakly supervised instance and semantic segmentation. In: IEEE International Conference on Computer Vision and Pattern Recognition. (2017) 876–885
9. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM transactions on graphics (TOG). Volume 23., ACM (2004) 309–314
10. Pont-Tuset, J., Arbelaez, P., Barron, J.T., Marques, F., Malik, J.: Multiscale combinatorial grouping for image segmentation and object proposal generation. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(1) (2017) 128–140
11. Rajchl, M., Lee, M.C., Oktay, O., Kamnitsas, K., Passerat-Palmbach, J., Bai, W., Damodaram, M., Rutherford, M.A., Hajnal, J.V., Kainz, B., et al.: Deepcut: Object segmentation from bounding box annotations using convolutional neural networks. IEEE Transactions on Medical Imaging **36**(2) (2017) 674–683
12. Pathak, D., Krahenbuhl, P., Darrell, T.: Constrained convolutional neural networks for weakly supervised segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1796–1804
13. Wei, Y., Feng, J., Liang, X., Cheng, M.M., Zhao, Y., Yan, S.: Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In: IEEE CVPR. (2017)
14. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems. (2014) 2672–2680
15. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: International Conference on Learning Representations (ICLR). (2018)

16. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: International Conference on Computer Vision and Pattern Recognition (CVPR). (2017) 1125–1134
17. Luc, P., Couprie, C., Chintala, S., Verbeek, J.: Semantic Segmentation using Adversarial Networks. In: NIPS Workshop on Adversarial Training, Barcelona, Spain (December 2016)
18. Hu, R., Dollár, P., He, K., Darrell, T., Girshick, R.: Learning to segment every thing. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2018)
19. Dwibedi, D., Misra, I., Hebert, M.: Cut, paste and learn: Surprisingly easy synthesis for instance detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2017) 1301–1310
20. Georgakis, G., Mousavian, A., Berg, A.C., Kosecka, J.: Synthesizing training data for object detection in indoor scenes. In: Robotics Science and Systems (RSS). (2017)
21. Abu Alhaija, H., Mustikovela, S.K., Mescheder, L., Geiger, A., Rother, C.: Augmented reality meets computer vision: Efficient data generation for urban driving scenes. International Journal of Computer Vision (Mar 2018)
22. Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Smolley, S.P.: Least squares generative adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV), IEEE (2017) 2813–2821
23. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: OSDI. Volume 16. (2016) 265–283
24. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR). (2015)
25. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. (2010) 249–256
26. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html
27. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The Cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
28. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1422–1430
29. Tung, H.Y.F., Harley, A.W., Seto, W., Fragkiadaki, K.: Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In: The IEEE International Conference on Computer Vision (ICCV). Volume 2. (2017)