Interactive Full Image Segmentation by Considering All Regions Jointly

Eirikur Agustsson
Google Research
eirikur@google.com

Jasper R. R. Uijlings
Google Research
jrru@google.com

Vittorio Ferrari
Google Research
vittoferrari@google.com

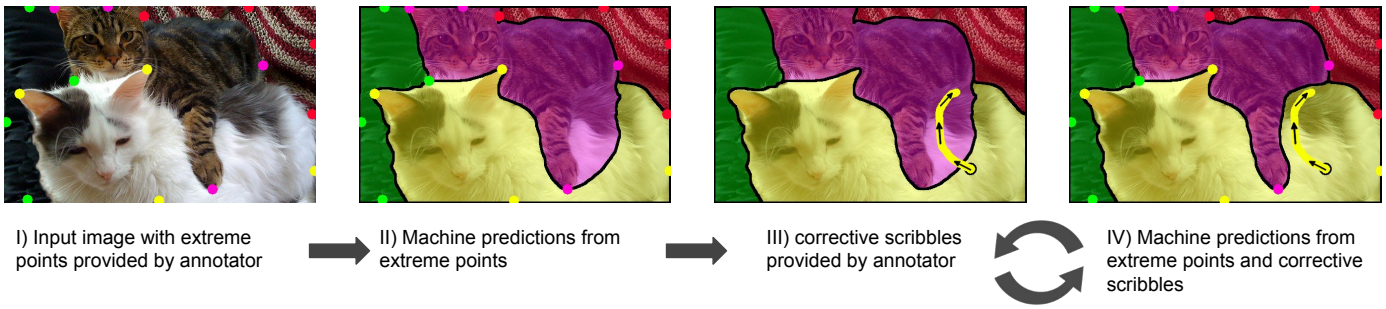

Figure 1. Illustration of our interactive full image segmentation workflow. First (I) the annotator marks extreme points. Then (II) our model (Sec. 3) uses them to generate a segmentation. This is presented to the annotator, after which we iterate: (III) the annotator makes corrections using scribbles (Sec. 4), and (IV) our model uses them to update the predicted segmentation (Sec. 3).

Abstract

We address interactive full image annotation, where the goal is to accurately segment all object and stuff regions in an image. We propose an interactive, scribble-based annotation framework which operates on the whole image to produce segmentations for all regions. This enables sharing scribble corrections across regions, and allows the annotator to focus on the largest errors made by the machine across the whole image. To realize this, we adapt Mask-RCNN [22] into a fast interactive segmentation framework and introduce an instance-aware loss measured at the pixel-level in the full image canvas, which lets predictions for nearby regions properly compete for space. Finally, we compare to interactive single object segmentation on the COCO panoptic dataset [11, 27, 34]. We demonstrate that our interactive full image segmentation approach leads to a 5% IoU gain, reaching 90% IoU at a budget of four extreme clicks and four corrective scribbles per region.

1. Introduction

We address the task of interactive full image segmentation, where the goal is to obtain accurate segmentations for all object and stuff regions in the image. Full image annotations are important for many applications such as self-driving cars [17, 19], navigation assistance for the blind [51], and automatic image captioning [25, 56]. However, creating such datasets requires large amounts of human labor. For example, annotating a single image took 1.5 hours for Cityscapes [17]. For COCO+stuff [11, 34], annotating one image took 19 minutes (80 seconds per object [34] plus 3 minutes for stuff regions [11]), which totals 39k hours for the 123k images. So there is a clear need for faster annotation tools.

This paper proposes an efficient interactive framework for full image segmentation (Fig. 1 and 2). Given an image, an annotator first marks extreme points [41] on all object and stuff regions. These provide a tight bounding box with four boundary points for each region, and can be efficiently collected (7s per region [41]). Next, the machine predicts an initial segmentation for the full image based on these extreme points. Afterwards we present the whole image with the predicted segmentation to the annotator and iterate between (A) the annotator providing scribbles on the errors of the current segmentation, and (B) the machine updating the predicted segmentation accordingly (Fig. 1).

Our approach of full image segmentation brings several advantages over modern interactive single object segmentation methods [24, 30, 31, 32, 36, 37, 60]: (I) It enables the annotator to focus on the largest errors in the whole image, rather than on the largest error of one given object. (II) It shares annotations across multiple object and stuff regions. In our approach, a single scribble correction specifies the extension of one region and the shrinkage of neighboring re-

gions (Sec. 3.2 and Fig. 3). In interactive single object segmentation, corrections are used for the given target object only. (III) Our approach lets regions compete for space in the common image canvas, ensuring that a pixel is assigned exactly one label (Fig. 3.1). In single object segmentation instead, pixels along boundary regions may be assigned to multiple objects, leading to contradictory labels, or to none, leading to holes. At the same time, since regions compete, corrections of one region influences nearby regions in our framework (e.g. Fig. 6). (IV) Instead of only annotating object instances, we also annotate stuff regions, capturing important classes such as `pavement` or `river`.

We realize interactive interactive full image segmentation by adapting Mask-RCNN [22] (Fig. 2). We start from extreme points [41], which define bounding boxes. Therefore we bypass the Region Proposal Network of Mask-RCNN and use these boxes directly to extract Region-of-Interest (RoI) features (Sec. 3.1). Afterwards, we incorporate extreme points and scribble annotations inside Mask-RCNN by concatenating them to the RoI-features. We encode annotations in a way that allows to share them across regions, enabling advantage (II) above (Sec. 3.2). Finally, while Mask-RCNN [22] predicts each mask separately, we project the mask predictions back on the pixels in the common image canvas 3.1. Then we define a new loss which is instance-aware yet lets predictions properly compete for space, enabling advantage (III) above (Sec. 3.3).

To the best of our knowledge, all deep interactive single object segmentation methods [24, 30, 31, 32, 36, 37, 60] are based on Fully Convolutional Networks (FCNs) [14, 35, 46]. We chose to start from Mask-RCNN [22] for efficiency. FCN-style interactive segmentation methods concatenate corrections to a crop of an RGB image, and pass that through a large neural net (e.g. ResNet-101 [23]). This requires a full inference pass for each region at each correction iteration. In our Mask-RCNN framework instead, the RGB image is first passed through the large backbone network. Afterwards, for each region only a pass over the final segmentation head is required (Fig. 2). This is much faster and more memory efficient (Sec. 6).

We perform thorough experiments in increasingly complex settings: (1) *Single object segmentation*: On the COCO dataset [34], our Mask-RCNN style architecture achieves similar performance to DEXTR [37] on single object segmentation from extreme points [41]. (2) *Full image segmentation*: We evaluate on the COCO panoptic challenge [11, 27, 34] the task of segmenting all object and stuff regions in an image, starting from extreme points. Our idea to share annotations across regions in combination with our pixel-wise loss yield a +3% IoU gain over an interactive single region segmentation baseline. (3) *Interactive full image segmentation*: On the COCO panoptic challenge, we demonstrate the combined effects of our three advantages (I)-(III) above: at a budget of four extreme clicks and four scribbles per region, we get a total +5% IoU gain over the interactive single region segmentation baseline.

2. Related Work

Semantic segmentation from weakly labeled data. Many works address semantic segmentation by training from weakly labeled data, such as image-level labels [28, 44, 58], point-clicks [5, 6, 13, 57], boxes [26, 37, 41] and scribbles [33, 59]. Boxes can be efficiently annotated using extreme points [41] which can also be used as an extra signal for generating segmentations [37, 41]. This is related as our method starts from extreme points for each region. However, the above methods operate from annotations collected before any machine processing. Our work instead is in the interactive scenario, where the annotator iteratively provide corrective annotations for the current machine segmentation.

Interactive object segmentation. Interactive object segmentation is a long standing research topic. Most classical approaches [3, 4, 8, 47, 18, 16, 21, 38] formulate object segmentation as energy minimization on a regular graph defined over pixels, with unary potential capturing low-level appearance properties and pairwise or higher-order potentials encouraging regular segmentation outputs.

Starting from Xu et al. [60], recent methods address interactive object segmentation with deep neural networks [24, 30, 31, 32, 36, 37, 60]. These works build on Fully Convolutional architectures such as FCNs [35] or Deeplab [14]. They input the RGB image plus two extra channels for object and non-object corrections, and output a binary mask.

In [15] they perform interactive object segmentation in video. They use Deeplab [14] to create a pixel-wise embedding space. Annotator corrections are used to create a nearest neighbor classifier on top of this embedding, enabling quick updates of the object predictions.

Finally, Polygon-RNN [1, 12] is an interesting alternative approach. Instead of predicting a mask, they uses a recurrent neural net to predict polygon vertices. Corrections made by the annotator are used by the machine to refine its vertex predictions.

Interactive full image segmentation. Recently, [2] proposed Fluid Annotation, which also addresses the task of full image annotation. Our work shares the spirit of focusing annotator effort on the biggest errors made by the machine across the whole image. However, [2] uses Mask-RCNN [22] to create a large pool of fixed segments and then provides an efficient interface for the annotator to rapidly *select which of these* should form the final segmentation. In contrast, in our work all segments are created from the initial extreme points and are all part of the final annotation. Our method then enables to *correct the shape* of segments

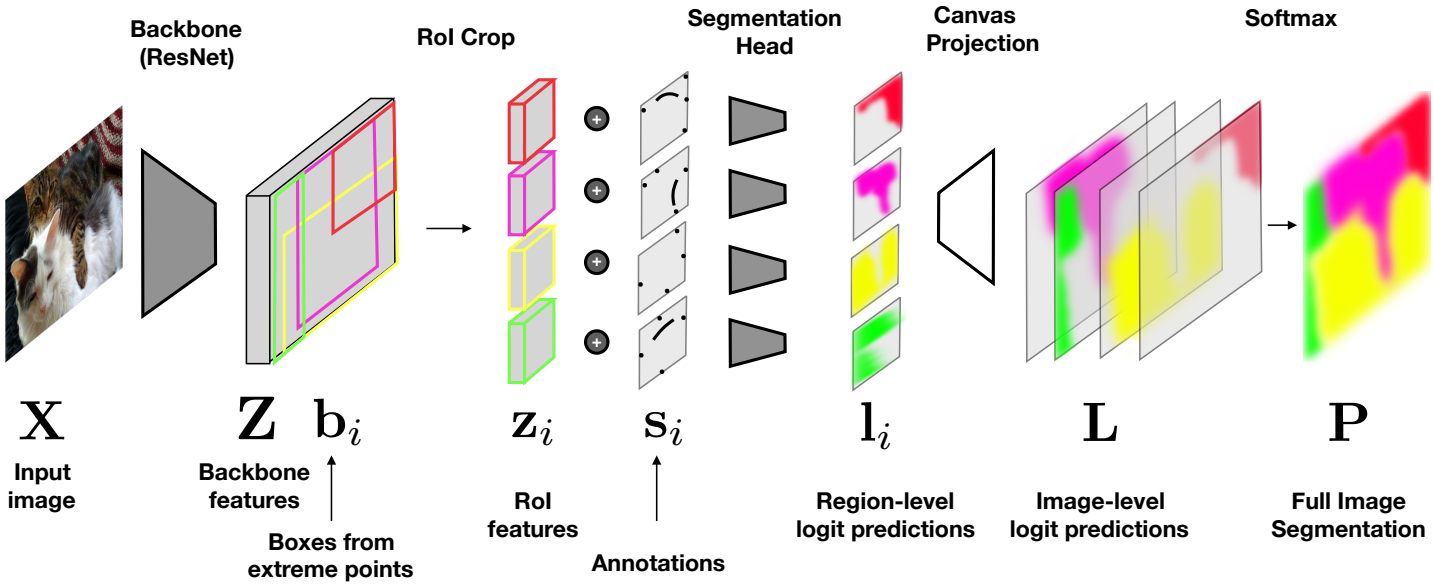

Figure 2: Our proposed region based model for interactive full image segmentation (see Sec. 3.1 for details). We start from Mask-RCNN [22], but use user provided boxes (from extreme points) instead of a box proposal networks for RoI cropping, and concatenate the RoI features with annotator provided corrective scribbles. Instead of predicting binary masks for each region, we project all region prediction into the common image canvas, where they compete for space. The network is trained end-to-end for a novel pixel-wise loss for the full image segmentation task (see Sec. 3.3).

to precisely match object boundaries.

Several older works on interactive segmentation handle multiple labels in a single image [39, 40, 50, 53]. We present the first interactive deep learning framework which does this. Moreover, in contrast to those works, we explicitly demonstrate the benefits of interactive full image segmentation over interactive single object segmentation.

Other works on interactive annotation. In [48] they combine a segmentation network with a language module to allow a human to correct the segmentation by typing feedback in natural language, such as "there are no clouds visible in this image". The work of [42] annotates bounding boxes using only human verification, while [29] trained agents to determine whether it is more efficient to verify or draw a bounding box. The avant-garde work of [49] had a machine dispatching many labeling questions to annotators, including whether an object class is present, box verification, box drawing, and finding missing instances of a particular class in the image. In [54] they estimate the informativeness of having an image label, a box, or a segmentation for an image, which they use to guide an active learning scheme. Finally, several works tackle fine-grained classification through attributes interactive provided by annotators [9, 43, 7, 55].

3. Our interactive segmentation model

This section describes our model which we use to predict a segmentation from extreme points and scribble corrections (Fig. 1). We first discuss the model architecture (Sec. 3.1). We then describe how we feed annotations to the model (extreme points and scribble corrections, Sec. 3.2). Finally, we describe model training with our new loss function (Sec. 3.3).

3.1. Model architecture

Our model is based on Mask-RCNN [22]. In Mask-RCNN inference is done as follows: (1) An input image X is passed through a deep neural network backbone such as ResNet [23], producing a feature map $\mathbf{Z}$. (2) A specialized network module (RPN [45]) predicts box proposals based on $\mathbf{Z}$. (3) These box proposals are used to crop out Region-of-Interest (RoI) features $\mathbf{z}$ from $\mathbf{Z}$ with a RoI cropping layer (RoI-align [22]). (4) Then each RoI feature $\mathbf{z}$ is fed into three separate network modules which predict a class label, refined box coordinates, and a segmentation mask.

Fig. 2 illustrates how we adapt Mask-RCNN [22] for interactive full image segmentation. In particular, our network takes three types of inputs: (1) an image X of size $W \times H \times 3$; (2) N annotation maps $\mathbf{S}_1, \cdots, \mathbf{S}_N$ of size $W \times H$ (for extreme points and scribble corrections, Sec. 3.2); and (3) N boxes $\mathbf{b}_1, \cdots, \mathbf{b}_N$ determined by the extreme points provided by annotators. Here N is the number of regions that we want to segment, which is determined by the annotator, and which may vary per image.

As in Mask-RCNN, an image X is fed into our backbone architecture (ResNet [23]) to produce feature map Z of size $\frac{1}{r}W \times \frac{1}{r}H \times C$, where C is the number of feature channels and r is a reduction factor. Both C and r are determined by the choice of backbone architecture.

In contrast to Mask-RCNN, we already have boxes

$\mathbf{b}_1, \cdots, \mathbf{b}_N$, so we do not need a box proposal module. Instead, we use each box $\mathbf{b}_i$ directly to crop out an RoI feature $\mathbf{z}_i$ from feature map $\mathbf{Z}$. All features $\mathbf{z}_i$ have the same fixed size $w \times h \times C$ (i.e. w and h are only dependent on the RoI cropping layer). We concatenate to this the corresponding annotation map $\mathbf{s}_i$ which is described in Sec. 3.2, and obtain a feature map $\mathbf{v}_i$, which is of size $w \times h \times (C+2)$.

Using $\mathbf{v}_i$, our network predicts a logit map $\mathbf{l}_i$ of size $w' \times h'$ which represents the prediction of a single mask. While Mask-RCNN stops at such mask predictions and processes them with a sigmoid to obtain binary masks, we want to have predictions influence each other. Therefore we use the boxes $\mathbf{b}_1, \cdots, \mathbf{b}_N$ to re-project the logit predictions of all masks $\mathbf{l}_i$ back into the original image resolution which results in N prediction maps $\mathbf{L}_i$. We concatenate these prediction maps into a single tensor $\mathbf{L}$ of size $W \times H \times N$. For each pixel, we then obtain region probabilities $\mathbf{P}$ of dimension $W \times H \times N$ by applying a softmax to the logits,

$$(P_1^{(x,y)}, \cdots, P_N^{(x,y)}) = \text{softmax}(L_1^{(x,y)}, \cdots, L_N^{(x,y)}), \quad (1)$$

where $P_i^{(x,y)}$ denotes the probability that pixel (x, y) is assigned to region i. This makes multiple nearby regions compete for space in the common image canvas.

3.2. Incorporating annotations

Our model in Fig. 2 concatenates RoI features $\mathbf{z}$ with annotation map $\mathbf{s}$. We now describe how we create $\mathbf{s}$. First, for each region i we create a positive annotation map $\mathbf{S}_i$ which is of the same size $W \times H$ as the image. We choose the annotation map to be binary and we create it by pasting all extreme points and corrective scribbles for region i onto it. Extreme points are represented by a circle which is 6 pixels in diameter. Scribbles are 3 pixels wide.

For each region i, we collapse all annotations which do not belong to it into a single negative annotation map $\sum_{j \neq i} \mathbf{S}_j$. Then, we concatenate the positive and negative annotation maps into a two-channel annotation map F_i

$$F_i := \Big(\mathbf{S}_i, \sum_{j \neq i} \mathbf{S}_j\Big), \quad (2)$$

which is illustrated in Fig. 3. Finally, we apply RoI-align [22] to F_i using box $\mathbf{b}_i$ to obtain the desired cropped annotation map $\mathbf{s}_i$.

The way we construct F_i enables the sharing of all annotation information across multiple object and stuff regions in the image. The negative annotations for one region are formed by collecting the positive annotations of all other regions. In contrast, in single object segmentation works [3, 8, 18, 16, 21, 24, 30, 31, 32, 38, 36, 37, 47, 60] both positive and negative annotations are made only on the target object and they are never shared, so they only have an effect on that one object.

3.3. Training

Training data. As training data, we have ground-truth masks for all objects and stuff regions in all images. We represent the (non-overlapping) N ground truth masks of an image $\mathbf{X}$ with region indices. This results in a map $\mathbf{Y}$ of dimension $W \times H$, which assigns each pixel $X^{(x,y)}$ to a region $Y^{(x,y)} \in \{1, ...N\}$.

Pixel-wise loss. Standard Mask-RCNN is trained with Binary Cross Entropy (BCE) losses for each mask prediction separately. This means that there is no direct interaction between adjacent masks, and they might even overlap. Instead, we propose a novel instance-aware loss which lets predictions compete for space in the original image canvas.

In particular, as described in Sec. 3.1 we project all region-specific logits into a single image-level logit tensor $\mathbf{L}$, which is softmaxed into a region assignment probabilities $\mathbf{P}$ of size $W \times H \times N$.

As described above, the ground-truth segmentation is represented by $\mathbf{Y}$ with values in $\{1, \cdots, N\}$, which specifies for each pixel its region index. Since we simulate the extreme points from the ground-truth masks, there is a direct correspondence between the region assignment probabilities $\mathbf{P}_1, \cdots, \mathbf{P}_N$ and $\mathbf{Y}$. Thus, we can train our network end-to-end for the Categorical Cross Entropy (CCE) loss for the region assignments:

$$\mathcal{L}_{\text{pixelwise}} = \sum_{(x,y)} -\log P_{Y^{(x,y)}}^{(x,y)} \quad (3)$$

We note that while the CCE loss is commonly used in fully convolutional networks for semantic segmentation [14, 35, 46], we instead use it in an architecture based on Mask-RCNN [22]. Furthermore, usually the loss is defined over a fixed number of classes [14, 35, 46], whereas we define it over the number of regions N. This number of regions may vary per image.

The loss in (3) is computed over the pixels in the full resolution common image canvas. Consequently, larger regions have a greater impact on the loss. However, in our experiments we measure Intersection-over-Union (IoU) between ground-truth masks and predictions, which considers all regions equally independent of their size. Therefore we weigh the terms in (3) as follows. For each pixel we find the smallest box $\mathbf{b}_i$ which contains it, and reweigh the loss for that pixel by the inverse of the size of $\mathbf{b}_i$. This causes each region to contribute to the loss approximately equally.

Our loss shares similarities with [10]. They used Fast-RCNN [20] with selective search regions [52] and generate a class prediction vector for each region. Then they project this vector back into the image canvas using its corresponding region, while resolving conflicts using a max operator. In our work instead, we project a full logit map back into the image (Fig. 2). Furthermore, while in [10] the number

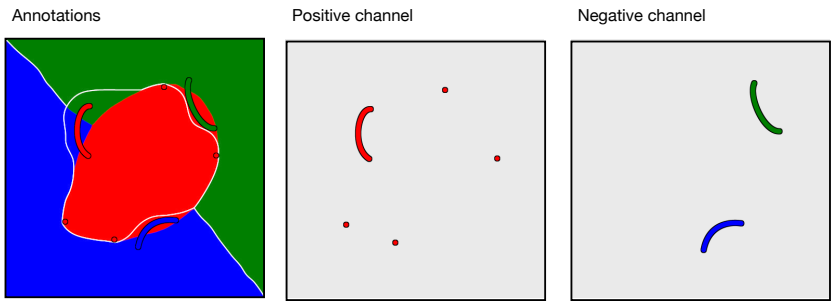

Figure 3: We illustrate how we combine all annotations near a region (red) into two annotation maps specific for that region. The colored regions denote the current predicted segmentation and the white boundaries depict true object boundaries. For the red region, the extreme points and the single positive scribble are combined into a single positive binary channel. All scribbles from other nearby regions are collected into a single negative binary channel.

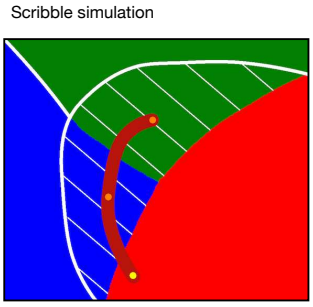

Figure 4: To simulate a corrective scribble, we first sample an initial control point to indicate which region we want to expand (yellow), followed by two control points (orange) sampled uniformly from the error region.

of logit channels is equal to the number of classes C, in our work it depends on the number of regions N, which may vary per image.

3.4. Implementation details

The original implementation of Mask-RCNN [22] creates for each RoI feature mask predictions for all classes that it is trained on. At inference time, it uses the predicted class to select the corresponding predicted mask. Since we build on Mask-RCNN, we also do this in our framework for convenience of the implementation. During training we use the class labels to train class-specific mask prediction logits. During inference, for each region i we use the class label predicted by Mask-RCNN to select which mask logits which we use as $\mathbf{l}_i$. Hence during inference time, we have implicit class labels. However, class labels are never exposed to the annotator and are considered to be irrelevant for this paper.

4. Annotations and their simulation

Our annotations consists of both extreme points and scribble corrections. We chose scribble corrections [3, 8, 47] instead of click corrections [24, 30, 31, 32, 36, 37, 60] as they are a more natural choice in our scenario. As we consider multiple regions in an image, any annotation first needs to indicate which region should be extended. With scribbles one can start inside the region to be extended, followed by a path which specifies how to extend the region.

In all our experiments we simulate annotations, following previous interactive segmentation works [1, 12, 24, 30, 31, 32, 36, 37, 60].

Simulating extreme points. To simulate the extreme points that the annotator provides at the beginning, we use the code provided by [37].

Simulating scribble corrections. To simulate scribble corrections during the interactive segmentation process, we first need to select an error region. Error regions are defined as a connected group of pixels of a ground-truth region which has been wrongly assigned to a different region (Fig. 4). We assess the importance of an error region by measuring how much segmentation quality (IoU) would improve if it was completely corrected. We use this to create annotator corrections on the most important error regions (the exact way depends on the particular experiment, details in Sec. 5).

To correct an error, we need a scribble that starts inside the ground-truth region and extends into the error region (Fig. 1). We simulate such scribbles with a three-step process, illustrated in Fig. 4: (1) first we randomly sample the first point on the border of the error region that touches the ground-truth region (yellow point in Fig. 4; (2) then we sample two more points uniformly inside the error region (yellow points in Fig. 4). (3) we construct a scribble as a smooth trajectory through these three points (using a bezier curve). We repeat this process ten times, and keep the longest scribble that is exclusively inside the ground-truth region (while all simulated points are within the ground-truth, the curve could cover parts outside the ground-truth).

5. Results

We use Mask-RCNN as basic segmentation framework instead of Fully Convolutional architectures [14, 35, 46] commonly used in single object segmentation works [24, 30, 31, 32, 36, 37, 60]. We first demonstrate in Sec. 5.1 that this is a valid choice by comparing to DEXTR [37] in the non-interactive setting where we generate masks starting from extreme points [41]. In Sec 5.2 we move to the full image segmentation task and demonstrate improvements resulting from sharing extreme points across regions and from our new pixel-wise loss. Finally, in Sec. 5.3 we show results on interactive full image segmentation, where we also share scribble corrections across regions, and allow the annotator to freely allocate scribbles to regions while considering the whole image.

Method	IoU
DEXTR [37]	82.1
DEXTR (released model)	81.9
Our `single region` model	81.6

Table 1: Performance on COCO (objects only). The accuracy of our `single region` model is comparable to DEXTR [37].

	X-points not shared	X-points shared
Mask-wise loss	75.8	76.0
Pixel-wise loss	78.4	79.1

Table 2: Performance on the COCO Panoptic validation set when predicting masks from extreme points (X-points). We vary the loss and whether extreme points are shared across regions. The top-left entry corresponds to our `single region` model, the bottom-right entry corresponds to our `full image` model.

5.1. Single object segmentation

DEXTR. In DEXTR [37] they predict object masks from four extreme points [41]. DEXTR is based on Deeplab-v2 [14], using a ResNet-101 [23] backbone architecture and a Pyramid Scene Parsing network [61] as prediction head. As input they crop a bounding box out of the RGB image based on the extreme points provided by the annotator. The locations of the extreme points are Gaussian blurred and fed as a heatmap to the network, concatenated to the cropped RGB input. The DEXTR segmentation model obtained state-of-the-art results on this task [37].

Details of our model. We compare DEXTR to a single object segmentation variant of our model (`single region` model). It uses the original Mask-RCNN loss, computed individually per mask, and does not share annotations across regions. For fair comparison to DEXTR, here we also use a ResNet-101 [23] backbone, which due to memory constraints limits the resolution of our RoI features to 14×14 pixels and our predicted mask to 33×33. Moreover, we use their released code to generate simulated extreme point annotations. In contrast to subsequent experiments, here we also use the same Gaussian blurred heatmaps to input annotations to our model as used in [37].

Dataset. We follow the experimental setup of [37] on the COCO dataset [34], which has 80 object classes. Models are trained on the 2014 training set and evaluated on the 2017 validation set (previously referred to as 2014 `minival`). We measure performance in terms of Intersection-over-Union averaged over all instances.

Results. Tab. 1 reports the original results from DEXTR [37], our reproduction using their publicly released model, and results of our `single region` model. Their publicly released model and our model deliver very similar results (81.9 and 81.6 IoU). This demonstrate that Mask-RCNN-style models are competitive to commonly used FCN-style models for this task.

5.2. Full image segmentation

Experimental setup. Given extreme points for each object and stuff region, we now predict a full image segmentation. We demonstrate the benefits of using our pixel-wise loss (Sec. 3.3) and sharing extreme points across regions (i.e. extreme points for one region are used as negative information for nearby regions, Sec. 3.2).

Details of our model. In preliminary experiments we found that RoI features of 14×14 pixels resolution were limiting accuracy when feeding in annotations into the segmentation head. Therefore we increased both the RoI features and the predicted mask to 41×41 pixels and switched to ResNet-50 [23] due to memory constraints. Importantly, in all experiments from now on our model uses the two-channel annotation maps described in Sec. 3.2.

Dataset. We perform our experiments on the COCO panoptic challenge dataset [11, 27, 34], which has 80 object classes and 53 stuff classes. Since the final goal is to efficiently annotate data, we train on only 12.5% of the 2017 training set (15k images). We evaluate on the 2017 validation set and measure IoU averaged over all object and stuff regions in all images.

Results. As Tab. 2 shows, our `single region` model yields 75.8 IoU. It uses a mask-wise loss and does not share extreme points across regions. When only sharing extreme points, we get a small gain of +0.2 IoU. In contrast, when only switching to our pixel-wise loss, results improve by +2.6 IoU. Sharing extreme points is more beneficial in combination with our new loss, yielding an additional improvement of +0.7 IoU. Overall this model with both improvements achieves 79.1 IoU, +3.3 higher than the `single region` model. We call it our `full image` model.

5.3. Interactive full image segmentation

We now move to our final system for interactive full image segmentation. We start from the segmentations from extreme points made by our `single region` and `full image` models from Sec. 5.2. Then we iterate between: (A) adding scribble corrections by the annotator, and (B) updating the machine segmentations accordingly.

Dataset and training. As before, we experiment on the COCO panoptic challenge dataset and report results on the 2017 validation set. Since during iterations our models input scribble corrections in addition to extreme points, we train two new interactive models: `single region scribble` and `full image scribble`. These models have the same architecture as their counterparts in Sec. 5.2 (which input only extreme points), but are trained differently. To create training data for one of these inter-

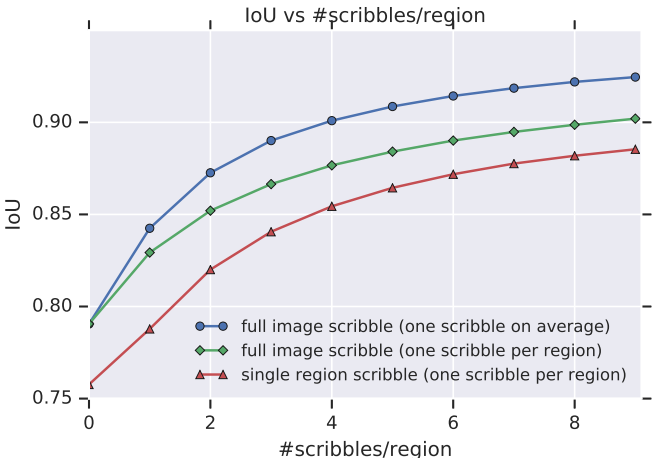

Figure 5: Results on the COCO Panoptic validation set for the interactive full image segmentation task. We measure average IoU vs the number of scribbles per region. We compare our `full image scribble` model under two scribble allocation strategies to the `single region scribble` baseline.

active models, we apply its counterpart to another 12.5% of the 2017 training set. We generate simulated corrective scribbles as described in Sec. 4 and train each model on the combined extreme points and scribbles annotations (Sec. 3.2). We keep these models fixed throughout all iterations of interactive segmentation. Note how, in addition to sharing extreme points as in Sec. 5.2, the `full image scribble` model also shares scribble corrections across regions.

Allocation of scribble corrections. When using our `single region scribble` model, in every iteration we allocate exactly one scribble to each region. Instead, when using our `full image scribble` model we also consider an alternative interesting strategy: one scribble per region *on average*, but the annotator can freely allocate these scribbles to the regions in the image. This enables the annotator to focus efforts on the biggest errors across the whole image, typically resulting in some regions receiving multiple scribbles and some receiving none.

Results. Fig. 5 shows annotation quality (IoU) vs cost (number of scribbles per region). The two starting points at zero scribbles are the same as the top-left and bottom-right entries of Tab. 2 since they are made using the same non-interactive models (from extreme points only).

We first compare `single region scribble` to `full image scribble` while using the same allocation strategy: exactly one scribble per region. Fig. 5 shows that for both models accuracy rapidly improves with more scribble corrections. However, `full image scribble` always offers a better trade-off between annotation effort and segmentation quality, e.g. to reach 85% IoU it takes 4 scribbles per region for the `single region scribble` model but only 2 scribbles for our `full image scribble` model. Similarly, to reach 88% IoU it takes 7 scribbles vs 4 scribbles. This confirms that the benefits of sharing annotations across regions and of our pixel-wise loss persist also in the interactive setting.

We now compare the two scribble allocation strategies on the `full image scribble` model. As Fig. 5 shows, using the strategy of freely allocating scribbles to regions (one scribble on average) brings further efficiency gains. `full image scribble` reaches a very high 90% IoU at just 4 scribbles per region on average. Reaching this IoU instead requires allocating exactly 8 scribbles per region with the other strategy. This demonstrate the benefits of focusing annotation effort on the largest errors across the whole image.

Overall, at a budget of four extreme clicks and four scribbles per region, we get a total 5% IoU gain over `single region scribble` (90% vs 85%). This gain is brought by the combined effects of our contributions: sharing annotations across regions, the pixel-wise loss which lets regions compete on the common image canvas, and the free scribble allocation strategy.

Fig. 6 shows various examples for how annotation progresses over iterations. Notice how in the first example, the corrective scribble on the left bear induces a negative scribble for the rock, which in turn improves the segmentation of the right bear. This demonstrates the benefit of sharing scribble annotations and competition between regions.

6. Discussion

Mask-RCNN vs FCNs. Our work builds on Mask-RCNN [22] rather than FCN-based models [14, 35, 46] because it is faster and requires less memory. To see this, we can reinterpret Fig. 2 as an FCN-based model: ignore the backbone network, replace the backbone features $\mathbf{Z}$ by the RGB image, and make the segmentation head a full FCN.

At inference time, we need to do a forward pass through the segmentation head for every region for every correction. When using Mask-RCNN, the heavy ResNet [23] backbone network is applied only once for the whole image, and then only a small 4-layer segmentation head is applied to each region. For the FCN-style alternative instead, nothing can be precomputed and the segmentation head itself is the heavy ResNet. Hence our framework is much faster during interactive annotation.

During training, typically all intermediate network activations are stored in memory. Crucially, for each region distinct activations are generated in the segmentation head. For FCN-style models this is a heavy ResNet and requires lots of memory. This is why DEXTR [37] reports a maximal batch size of 5 regions. Therefore, it would be difficult to train with our pixel-wise loss in an FCN-style model, as that requires processing all regions in each image simulta-

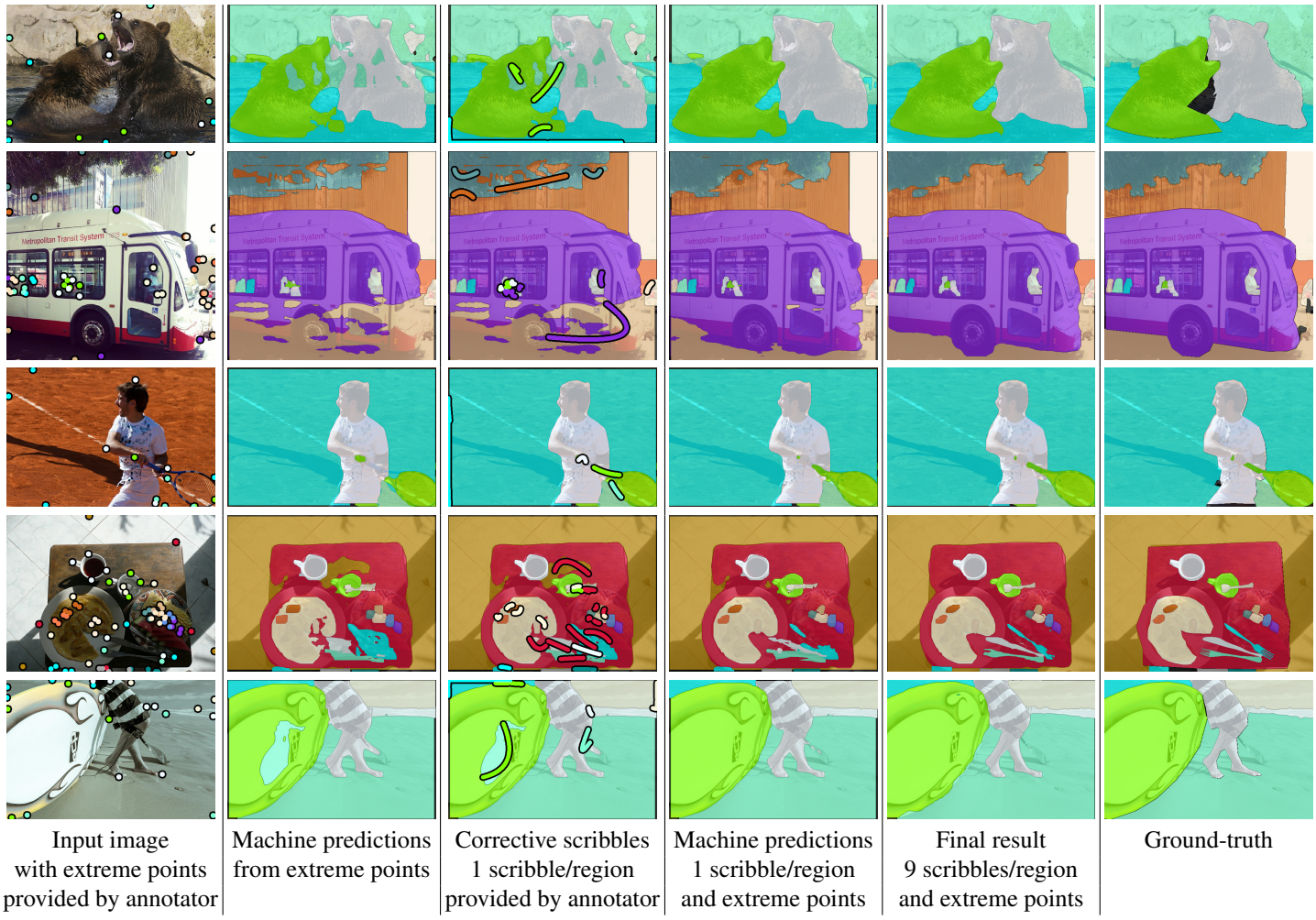

Figure 6: We show example results obtained by our system using the `full image scribble` model with a free allocation strategy. The first two columns show the input image with extreme points and predictions. Column 3 shows the first annotation step with one scribble correction per region on average, and column 4 shows the updated predictions. The last two columns compare the final result after 9 steps (using 9 scribbles per region on average) with the COCO ground-truth segmentation.

neously (15 regions per image on average).

In fact our Mask-RCNN based architecture (Fig. 2) and its reinterpretation as an FCN-based model span a continuum. Its design space can be explored by varying the size of the backbone and the segmentation head, as well as their input and output resolution. We leave such exploration of the trade-off between memory requirements, inference speed, and model accuracy for future work.

Scribble and point simulations. Like other interactive segmentation works [1, 12, 24, 30, 31, 32, 36, 37, 60], we simulate annotations. It remains to be studied how to best select the simulation parameters so that the models generalize well to real human annotators. The optimal parameters will likely depend on various factors, such as the desired annotation quality and the accuracy of the provided corrections.

7. Conclusions

We proposed an interactive annotation framework which operates on the whole image to produce segmentations for all object and stuff regions. Our key contributions derive from considering the full image at once: sharing annotations across regions, focusing annotator effort on the biggest errors across the whole image, and a pixel-wise loss for Mask-RCNN that lets regions compete on the common image canvas. We have shown through experiments on the COCO panoptic challenge dataset [11, 27, 34] that all the elements we propose improve the trade-off between annotation cost and quality, leading to a very high IoU of 90% using just four extreme points and four corrective scribbles per region (compared to 85% for the baseline).

References

[1] D. Acuna, H. Ling, A. Kar, and S. Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*, 2018. 2, 5, 8

[2] M. Andriluka, J. R. R. Uijlings, and V. Ferrari. Fluid annotation: A human-machine collaboration interface for full image annotation. In *ACM Multimedia*, 2018. 2

[3] X. Bai and G. Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *IJCV*, 2009. 2, 4, 5

[4] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. Interactively co-segmentating topically related images with intelligent scribble guidance. *IJCV*, 2011. 2

[5] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. What's the point: Semantic segmentation with point supervision. 2016. 2

[6] S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database. In *CVPR*, 2015. 2

[7] A. Biswas and D. Parikh. Simultaneous active learning of classifiers & attributes via relative feedback. In *CVPR*, 2013. 3

[8] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, 2001. 2, 4, 5

[9] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *ECCV*, 2010. 3

[10] H. Caesar, J. Uijlings, and V. Ferrari. Region-based semantic segmentation with end-to-end training. In *ECCV*, 2016. 4

[11] H. Caesar, J. Uijlings, and V. Ferrari. COCO-stuff: Thing and stuff classes in context. In *CVPR*, 2018. 1, 2, 6, 8

[12] L. Castrejón, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-rnn. In *CVPR*, 2017. 2, 5, 8

[13] D.-J. Chen, J.-T. Chien, H.-T. Chen, and L.-W. Chang. Tap and shoot segmentation. In *AAAI*, 2018. 2

[14] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. on PAMI*, 2017. 2, 4, 5, 6, 7

[15] Y. Chen, J. Pont-Tuset, A. Montes, and L. Van Gool. Blazingly fast video object segmentation with pixel-wise metric learning. In *CVPR*, 2018. 2

[16] M.-M. Cheng, V. A. Prisacariu, S. Zheng, P. H. S. Torr, and C. Rother. Densecut: Densely connected crfs for realtime grabcut. *Computer Graphics Forum*, 2015. 2, 4

[17] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *CVPR*, 2016. 1

[18] A. Criminisi, T. Sharp, C. Rother, and P. Perez. Geodesic image and video editing. In *ACM Transactions on Graphics*, 2010. 2, 4

[19] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, 2013. 1

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 4

[21] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *CVPR*, 2010. 2, 4

[22] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2, 3, 4, 5, 7

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3, 6, 7

[24] Y. Hu, A. Soltoggio, R. Lock, and S. Carter. A fully convolutional two-stream fusion network for interactive image segmentation. *Neural Networks*, 2019. 1, 2, 4, 5, 8

[25] J. Johnson, A. Karpathy, and L. Fei-Fei. DenseCap: Fully convolutional localization networks for dense captioning. In *CVPR*, 2016. 1

[26] A. Khoreva, R. Benenson, J. Hosang, M. Hein, and B. Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *CVPR*, 2017. 2

[27] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. In *ArXiv*, 2018. 1, 2, 6, 8

[28] A. Kolesnikov and C. Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *ECCV*, 2016. 2

[29] K. Konyushkova, J. Uijlings, C. Lampert, and V. Ferrari. Learning intelligent dialogs for bounding box annotation. In *CVPR*, 2018. 3

[30] H. Le, L. Mai, B. Price, S. Cohen, H. Jin, and F. Liu. Interactive boundary prediction for object selection. In *ECCV*, 2018. 1, 2, 4, 5, 8

[31] Z. Li, Q. Chen, and V. Koltun. Interactive image segmentation with latent diversity. In *CVPR*, 2018. 1, 2, 4, 5, 8

[32] J. Liew, Y. Wei, W. Xiong, S.-H. Ong, and J. Feng. Regional interactive image segmentation networks. In *ICCV*, 2017. 1, 2, 4, 5, 8

[33] D. Lin, J. Dai, J. Jia, K. He, and J. Sun. ScribbleSup: Scribble-supervised convolutional networks for semantic segmentation. In *CVPR*, 2016. 2

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1, 2, 6, 8

[35] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2, 4, 5, 7

[36] S. Mahadevan, P. Voigtlaender, and B. Leibe. Iteratively trained interactive segmentation. In *BMVC*, 2018. 1, 2, 4, 5, 8

[37] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018. 1, 2, 4, 5, 6, 7, 8

[38] N. S. Nagaraja, F. R. Schmidt, and T. Brox. Video segmentation with just a few strokes. In *ICCV*, 2015. 2, 4

[39] C. Nieuwenhuis and D. Cremers. Spatially varying color distributions for interactive multilabel segmentation. *IEEE Trans. on PAMI*, 2013. 3

[40] C. Nieuwenhuis, S. Hawe, M. Kleinsteuber, and D. Cremers. Co-sparse textural similarity for interactive segmentation. In *ECCV*, 2014. 3

[41] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari. Extreme clicking for efficient object annotation. In *ICCV*, 2017. 1, 2, 5, 6

[42] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari. We don't need no bounding-boxes: Training object class detectors using only human verification. In *CVPR*, 2016. 3

[43] A. Parkash and D. Parikh. Attributes for classifier feedback. In *ECCV*, 2012. 3

[44] D. Pathak, P. Krähenbuhl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *ICCV*, 2015. 2

[45] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 3

[46] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2, 4, 5, 7

[47] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004. 2, 4, 5

[48] C. Rupprecht, I. Laina, N. Navab, G. D. Hager, and F. Tombari. Guide me: Interacting with deep networks. In *CVPR*, 2018. 3

[49] O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. In *CVPR*, 2015. 3

[50] J. Santner, T. Pock, and H. Bischof. Interactive multi-label segmentation. In *ACCV*, 2010. 3

[51] M. Serrão, S. Shahrabadi, M. Moreno, J. José, J. I. Rodrigues, J. M. Rodrigues, and J. H. du Buf. Computer vision and GIS for the navigation of blind persons in buildings. *Universal Access in the Information Society*, 2015. 1

[52] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 2013. 4

[53] V. Vezhnevets and V. Konushin. GrowCut - interactive multi-label N-D image segmentation by cellular automata. In *GraphiCon*, 2005. 3

[54] S. Vijayanarasimhan and K. Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *CVPR*, 2009. 3

[55] C. Wah, G. Van Horn, S. Branson, S. Maji, P. Perona, and S. Belongie. Similarity comparisons for interactive fine-grained categorization. In *CVPR*, 2014. 3

[56] G. Wang, P. Luo, L. Lin, and X. Wang. Learning object interactions and descriptions for semantic image segmentation. In *CVPR*, 2017. 1

[57] T. Wang, B. Han, and J. Collomosse. Touchcut: Fast image and video segmentation using single-touch interaction. *CVIU*, 2014. 2

[58] Y. Wei, H. Xiao, H. Shi, Z. Jie, J. Feng, and T. S. Huang. Revisiting dilated convolution: A simple approach for weakly- and semi-supervised semantic segmentation. In *CVPR*, 2018. 2

[59] J. Xu, A. G. Schwing, and R. Urtasun. Learning to segment under various forms of weak supervision. In *CVPR*, 2015. 2

[60] N. Xu, B. Price, S. Cohen, J. Yang, and T. Huang. Deep interactive object selection. In *CVPR*, 2016. 1, 2, 4, 5, 8

[61] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017. 6