# MultiSeg: Semantically Meaningful, Scale-Diverse Segmentations from Minimal User Input

Jun Hao Liew[1*] Scott Cohen[2] Brian Price[2] Long Mai[2] Sim-Heng Ong[1] Jiashi Feng[1]
[1] National University of Singapore [2] Adobe Research
liewjunhao@u.nus.edu {scohen,bprice,malong}@adobe.com {eleongsh,elefjia}@nus.edu.sg

## Abstract

*Existing deep learning-based interactive image segmentation approaches typically assume the target-of-interest is always a single object and fail to account for the potential diversity in user expectations, thus requiring excessive user input when it comes to segmenting an object part or a group of objects instead. Motivated by the observation that the object part, full object, and a collection of objects essentially differ in size, we propose a new concept called scale-diversity, which characterizes the spectrum of segmentations w.r.t. different scales. To address this, we present MultiSeg, a scale-diverse interactive image segmentation network that incorporates a set of two-dimensional scale priors into the model to generate a set of scale-varying proposals that conform to the user input. We explicitly encourage segmentation diversity during training by synthesizing diverse training samples for a given image. As a result, our method allows the user to quickly locate the closest segmentation target for further refinement if necessary. Despite its simplicity, experimental results demonstrate that our proposed model is capable of quickly producing diverse yet plausible segmentation outputs, reducing the user interaction required, especially in cases where many types of segmentations (object parts or groups) are expected.*

## 1. Introduction

Interactive image segmentation takes user input (*e.g.* clicks) to guide the segmentation process toward the desired output. Unlike semantic image segmentation which is fully automated, interactive image segmentation is a semi-automated process, allowing extra user input to be added for refinement until the segmentation performance is satisfactory. It has become a popular research topic in the past decades with a wide application domains such as data annotation, local image/video editing, image composition, and medical image analysis.

Existing deep learning-based interactive image segmentation methods [16, 18, 23, 26, 27, 36] have demonstrated significant improvement over the previous techniques using hand-crafted features, typically allowing users to extract accurate segmentation masks with just a few user inputs. Although these models excel in extracting single object given some user input, in practice, the target-of-interest may not always be a single object. It could be a local part within the object or a group of objects that this object belongs to. However, previous techniques do not account for such diversity in segmentation, thus demanding additional user input when the segmentation target is not a single object.

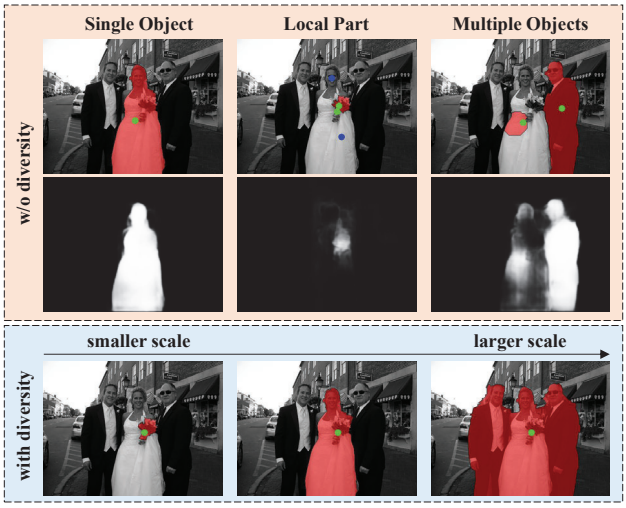



Figure 1: **(Top row)** Existing interactive image segmentation techniques typically do not account for diversity in segmentations. Despite their excellent performance on extracting single object, they usually demand extensive user input when segmenting a local part or a group of multiple objects. **(Bottom row)** Our porposed MultiSeg produces a set of scale-varying segmentations conforming to the given user input. Positive (foreground) and negative (background) clicks are represented by green and blue clicks, respectively.

Figure 1 illustrates how conventional segmentation methods ([36]) can struggle with handling varying user-desired segmentations. As shown in the first column, such a model is effective in extracting the single object (person), requiring only one foreground click in this case. However, when the segmentation target is a local part (the flower

*This work was mainly done during an internship at Adobe Research.

bouquet), a significant number of background clicks are required to deselect the person's body as shown in the second column. On the other hand, when it comes to segmenting a group of objects (third column), we notice a significant degradation in the segmentation quality for the first object (the bride) when an additional foreground click is added to the second object. Such methods are neither designed nor trained to produce diverse segmentation results.

Our key idea is to produce a set of diverse segmentations as recommendations to the user where each segmentation should conform to the user input. As the user provides more evidence, the model should quickly converge to one of them. In a recent work, Li *et al*. [22] have shown that training with a diversity loss to encourage the network to capture different segmentation modes can improve the segmentation performance. However, the diversity training framework in [22] is unconstrained in that there's no mechanism to encourage different branches to be either meaningful or different from one another. Most importantly, this approach still assumes a single true segmentation given the user clicks and therefore still cannot capture the multi-candidates nature of segmentation mentioned above. We argue that a good segmentation system needs to be able to suggest results that are both diverse and meaningful. Achieving this is a highly non-trivial task as the the number of plausible segmentations is in general unknown beforehand and it is unclear how to best define the spectrum of diversity for segmentation results.

Motivated by the observation that different segmentations (object parts, full objects, and a group of multiple objects) essentially differ in size, we propose to impose a set of two dimensional scale priors to characterize the spectrum of segmentations such that each segmentation output is constrained to be of a certain scale while conforming to the user input. With this, we present **MultiSeg**, a scale-diverse interactive image segmentation network that produces a set of scale-varying proposals conditioned on the user input in a single forward pass. Specifically, given an (image, user input) pair, our model produces multiple outputs consistent with the user inputs, where each branch seeks a possible user-expected segmentation within a given horizontal and vertical scale. To train this model, we synthesize the diversity-aware training data containing simulated user clicks along with multiple possible ground-truths corresponding to each input click set.

Figure 1 (bottom row) shows an example result produced by our method. Note that the flower bouquet, single person, and the three people can be obtained with just a single foreground click. Extensive experimental analysis demonstrates that our model is capable of generating a set of diverse yet plausible segmentations consistent with the user input.

To further improve the user experience, we introduce an objectness classifier to select the most meaningful out of all the proposals. We then recommend the top proposals to the user. As a result, the user does not have to inspect each segmentation carefully and a quick glance is usually sufficient to check if a better segmentation exists among the proposals. Moreover, the MultiSeg can reuse the corresponding scale information to constrain the subsequent segmentation outputs when the user selected an alternative proposal.

The key contributions of our work are as follows:

- We propose to characterize the diversity in segmentations using a scale factor and we call this *scale-diversity*.
- We propose a new architecture that incorporates the scale diversity and generates a set of scale-varying proposals conditioned on the user input.
- We introduce a novel pipeline to synthesize diverse training samples to explicitly encourage the segmentation diversity.

## 2. Related Works

**Interactive Image Segmentation:** Early interactive image segmentation methods mainly exploit boundary properties for segmentation [17, 29] while more recent approaches are based on graphical models, such as graph cut [4, 21, 33], random walker [11] and geodesic approaches [2, 7, 31]. Various forms of priors have also been proposed to further improve the segmentation performance [10, 12, 35]. Nevertheless, these approaches rely on low-level features, such as color or texture, which often lead to poor segmentation quality especially in cases of complex background or varying illumination conditions.

More recently, deep-learning based interactive segmentation methods have shown remarkably improved accuracy compared to the traditional approaches. Xu *et al*. [36] transformed sparse user clicks into Euclidean distance maps and concatenated them with the input image to train an FCN in an end-to-end manner. Liew *et al*. [23] exploited the local regional context surrounding the user input together with a multi-scale global contextual prior for local refinement. [26] proposed an iterative training procedure to address the mismatch between training and testing. Hu *et al*. [16] proposed a fully convolutional two-stream fusion network that processes the input image and user clicks individually before fusing them such that the user input has a more direct impact on the segmentation output. Polygon-RNN [1, 5], on the other hand, formulated interactive image segmentation as a polygon prediction problem where a recurrent neural network is used to sequentially predict the vertices of the polygons outlining the object-to-segment. DEXTR [27] transformed extreme points (left-most, right-most, top and bottom pixels) into a Gaussian heatmap and concatenated with the image to perform segmentation. Le *et al*. [18] proposed an interactive boundary prediction network that takes

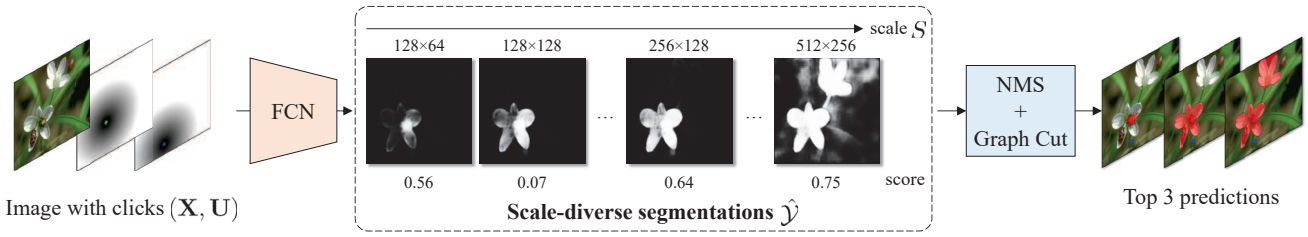



Figure 2: Overview of MultiSeg. Given an image and the distance-transformed user input, the FCN model outputs a set of scale-varying segmentation masks and corresponding object scores based on a set of predefined two-dimensional scales. Finally, we run NMS to keep at most the top $N$ predictions as recommendations to the user.

boundary clicks as inputs. Nonetheless, none of the aforementioned methods address the diversity in segmentations, especially when the number of user input is small.

**Diverse Predictions:** While there is a large body of work on making diverse predictions [3, 9, 13, 19, 20], we briefly discuss works that produce a diverse set of solutions in interactive image segmentation. Batra *et al*. [3] trained a single-output model and then find the top $M$ most probable solutions during inference using a greedy method which applies a penalty to each new solution if it is close to previously discovered solutions. However, there exists train-test disparity since the model is not aware of multiple outputs during training. Furthermore, the discovered solutions are not guaranteed to be semantically meaningful. More recently, Li *et al*. [22] formulated interactive image segmentation under a multiple choice learning setting where the loss for each training sample is backpropagated to the branch that gives the lowest loss during the forward pass. Nevertheless, the number of modes are data-dependent and the model will suffer from mode-collapse when one or more branches fail to receive any training signal. There is no definable meaning or organization behind the output of each branch and no way to direct the training to ensure that the full diversity of possible segmentations are learned. Unlike existing works where the diversity is either obtained via some constrained optimizations or learned in an unconstrained manner that may not cover the desired range of diversity, we provide a clear definition of diversity, *i.e.* scale, that allows both diverse and interpretable solutions. To our best knowledge, this is the first interactive image segmentation work that attempts to define the diversity in segmentations.

## 3. Method

We discuss our definition of diversity in Section 3.1, followed by details of the network architecture in Section 3.2, and the generation of diverse training data in Section 3.3.

### 3.1. Scale-Diversity

Given a set of user inputs, our goal is to generate a set of diverse and semantically meaningful segmentations that conform to the user input. However, without having a concrete definition of diversity, it remains unclear how to separate or distinguish the different segmentation outputs. A plausible choice is to represent the spectrum of segmentations based on some hierarchical partitioning. For example, a person can be partitioned into body parts and clothing. Therefore, placing a positive click on the shirt enables extraction of both the shirt and the full person. Nevertheless, such training data could be expensive to obtain. Moreover, modeling the hierarchical relationships is a non-trivial task. Instead, we propose a simple yet effective option, which is to characterize the variation of segmentations in terms of scale, and we call this *scale-diversity*. For instance, the flower bouquet, single person, and the three people in Figure 1 can be represented with different scale factors.

Inspired by the recent object detection pipeline [25, 32], we define the two-dimensional (horizontal and vertical) scale $S$ based on different combinations of aspect ratios and sizes. This two-dimensional parameterization provides a greater degree of freedom to represent various segmentations as compared to a one-dimensional representation such as area. Consider a color image $\mathbf{X} \in \mathbb{R}^{H\times W\times 3}$ and some user input $\mathbf{U} \in \mathbb{R}^{H\times W\times 2}$, we formulate the task of generating a diverse set of segmentations as learning a mapping function $f(;\theta, S)$ that is parameterized by $\theta$ and conditioned on a set of predefined scales $S$ (given in Section 4.1):

$$\hat{\mathcal{Y}} = f(\mathbf{X}, \mathbf{U}; \theta, S) \tag{1}$$

where $\hat{\mathcal{Y}} = \{\hat{\mathbf{Y}}_1, \hat{\mathbf{Y}}_2, ..., \hat{\mathbf{Y}}_M\} \in \mathbb{R}^{H\times W\times M}$ is the set of scale-diverse segmentation outputs where segmentation output $\mathbf{Y}_m$ corresponds to the $m$-th scale and $M$ is the number of predefined scales. In the next section, we will discuss the network architecture in detail.

### 3.2. Scale-Diverse Interactive Segmentation

As shown in Figure 2, our model takes as input the image to segment and a set of user-provided positive and negative clicks and generates $M$ scale-varying segmentation masks with the corresponding object scores indicating the presence of a plausible segmentation at each scale. Then, we

perform non-maximum suppression (NMS) and keep only the top $N$ (*e.g.* $N$=3) predictions as recommendations to the user. Following [23, 36], we apply graph cut [4] to the network prediction to obtain the final segmentation mask.

**Input representation** Following [22, 23, 36], we first transform the positive clicks $\mathbf{C}^+$ and negative clicks $\mathbf{C}^-$ to two truncated Euclidean distance maps $\mathbf{U} = (\mathbf{U}^+, \mathbf{U}^-)$ before concatenating with the input image to form a 5-channel input $(\mathbf{X}, \mathbf{U})$ to the network. See more details in [36].

**Network architecture** Here, we present the architecture of our segmentation network $f$. In this work, we employ the ResNet-101 [15] variant of DeepLabv3+ [6] as our backbone architecture.

In order to convert the DeepLabv3+ to our scale-diverse interactive segmentation network, we made the following three modifications: 1) the first convolution filter is modified to accept the additional two channels of user input $\mathbf{U}$; 2) the output layer is modified to have $M$ outputs/branches that correspond to $M$ different scales; 3) a global average pooling layer followed by a new fully connected layer with $M$ outputs are appended before the decoder to predict an object score for each of the $M$ segmentation masks.

Note that the proposed scale-diversity is generic. Any FCN-based interactive image segmentation network can be easily turned into a scale-diverse interactive segmentation counterpart with minor modifications as described above.

**Training** Next, we describe how to train our model to produce scale-diverse segmentation masks. Given a ground truth segmentation mask $\mathbf{Y}$, we first compute a tight bounding box enclosing the ground truth mask. Then, we generate a set of "anchors" centering at the same center as the ground truth bounding box but with different scales and look for the set of scales $S_\mathbf{Y}$ that overlap with the ground truth bounding box with intersection-over-union (IoU) larger than 0.5. In the cases where there is no scale with overlap larger than 0.5, we simply set the scale with the largest IoU as the ground truth scale. Then, we backpropagate the loss only through those branches:

$$\mathcal{L} = \frac{1}{|S_\mathbf{Y}|} \sum_{s \in S_\mathbf{Y}} \ell(f(\mathbf{X}, \mathbf{U}; \theta, \{s\}), \mathbf{Y}) \qquad (2)$$

where $\ell$ is the standard sigmoid cross-entropy loss.

**Prediction of object score** Since not all the scales necessarily correspond to some meaningful segmentations, we also train our MultiSeg to predict an object score that indicates whether a particular scale contains object(s). Specifically, we append a global average pooling layer, followed by a fully connected layer at the output of the encoder to yield $M$ confidence scores for the $M$ scale-diverse segmentations. Since the distribution of positive/negative samples is usually imbalanced due to the fact there is only a small set of scales that contain object(s), we train the score prediction branch with a class-balancing sigmoid cross entropy loss.

### 3.3. Generating Diverse Training Samples

In the previous section, we described how each ground truth object is assigned to the corresponding branch for training. However, this does not guarantee diversity in the segmentation outputs. Instead, we propose to obtain the diversity in segmentations by synthesizing diverse training data with a new click sampling strategy to explicitly encourage the model to learn to generate diverse predictions given the same set of user input. Note that any existing segmentation dataset with instance-level annotations can be used without the need to collect new training samples.

For each object, we first extract all the neighboring objects and build a hierarchical list of segmentations based on different combinations of objects[1]. We then randomly sample $K$ samples from the list of segmentations for training. Specifically, given $K$ ground truth masks $\mathcal{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_K\}$, we use the following loss to encourage the model to learn the diversity in segmentations:

$$\mathcal{L} = \frac{1}{|\mathcal{Y}|} \sum_{\mathbf{Y} \in \mathcal{Y}} \frac{1}{|S_\mathbf{Y}|} \sum_{s \in S_\mathbf{Y}} \ell(f(\mathbf{X}, \mathbf{U}; \theta, \{s\}), \mathbf{Y}) \qquad (3)$$

When only a single ground truth is given, this loss reduces to Eq.(2). Note that we combine the neighboring objects in a class-agnostic manner, *i.e.* we ignore the object classes when forming a diverse set of ground truths. This allows us to cover common coocurring objects (*e.g.* a person in a chair). Using depth ordering to combine objects with similar depth and taking object-object interactions into account might be useful to decide if two neighboring objects should be segmented altogether but this is beyond the scope of this work. This will be further investigated in the future.

**Sampling of Clicks** We follow the clicks sampling protocol proposed by [36] to generate a set of positive and negative clicks. However, this strategy is not aware of the presence of multiple possible segmentations when sampling clicks for training. To explicitly encourage the model to learn to produce multiple equally plausible segmentations given the same set of user clicks, we also include another click sampling strategy by sampling positive and negative clicks only on the common foreground and background among the $K$ ground truth masks. After sampling

[1]The hierarchical list of segmentations refer to the different combinations of objects that include the sampled object. For example, given an object instance $a$ and its neighboring instances $b, c$, the hierarchical list of segmentations will be $\{a\}$, $\{a, b\}$, $\{a, c\}$ and $\{a, b, c\}$.

the clicks, we also add in all other segmentation masks that are consistent with those clicks for training.

### 3.4. Inference

During testing, given some user clicks, $M$ diverse segmentations can be obtained in just a single forward pass. However, it is impractical for the user to inspect each and every segmentation mask each time when a new click is added since this takes significantly longer time than adding a new click alone. This is further complicated by the fact that not all the segmentations produced are necessarily semantically meaningful. Moreover, some segmentation branches may not have a semantically meaningful object(s) at that scale, and the branch will likely to produce the object(s) that is closest to that scale. Thus, nearby scale branches may produce similar segmentation outputs.

To overcome this, during testing, we perform non-maximum suppression (NMS) to remove redundant proposals and keep *at most* the top $N$ segmentations (we set $N$ to 3 in this work) as recommendations to the user. In this case, a quick glance is usually sufficient to quickly locate the closest segmentation. An example is shown at Figure 3.

Given an initial positive click, we take the segmentation mask with the highest predicted object score as our default output, which will be overlaid on the main canvas for further processing. When an alternative proposal is selected instead, the scale prior obtained from the corresponding branch will be reused by taking the segmentation output from that branch as the default output in the next round. Otherwise, we always take the segmentation mask from the branch that yields the highest object score in the first round as our default output.

Interestingly, we noticed that inspection of the segmentation list is usually required only in the first few rounds when the target of segmentation is still ambiguous. As the interactive segmentation process moves on, the model eventually narrows down to one of the solutions given more clicks and the user can focus on adding new clicks for refinement.

## 4. Experiments

As the goal of this paper is to produce diverse segmentations for improving interactive segmentation in scale diverse tasks, we evaluate the quality of our diverse segmentations as well as perform a user study showing the practical effectiveness of our approach in more real world situations where the target selection varies in scale and number of objects. However, we first evaluate our method on traditional single object segmentations. While our goal is not to produce single object segmentation with the minimal amount of clicks, but rather to better handle scale-diverse segmentation tasks, it is useful to see how our method compares to prior works that have fine-tuned to this more limited task in order to verify that we have not lost this ability.

Figure 3: User interface for the user study

### 4.1. Implementation Details

We trained our MultiSeg network on the PASCAL VOC dataset [8] augmented with extra labels from SBD [14] (10,582 images) following the common practice. The generation of diverse training data and simulation of user clicks have been discussed in Section 3.3. All images are resized to $512 \times 512$ during training. We adopt random horizontal flipping as the only data augmentation. The network is initialized from the DeepLabv3+ model pre-trained on ImageNet [34], MS COCO [24], and PASCAL VOC dataset [8]. For the new layers and the two extra channels in the first convolutional layer, we randomly initialize them from a Gaussian distribution with standard deviation of 0.01. The learning rate is set to $1 \times 10^{-8}$, with momentum of 0.9 and weight decay of $5 \times 10^{-4}$. We trained our model using stochastic gradient descent with a batch size of 5 images on a single NVIDIA Pascal Titan X GPU for 20 epochs. All our experiments are conducted on PyTorch framework.

For all the experiments, we use 3 sizes of 64, 128, 256 and 3 aspect ratios of 1:1, 1:2, and 2:1. On top of that, we also include 3 additional scales of $16 \times 16$, $32 \times 32$, and $512 \times 512$ to cover the extremely small and large objects, resulting in $M$=12. Although our model needs to predict $M$=12 diverse predictions, an average single forward pass that produces all $M$ segmentations only takes less than 100 ms using a Pascal Titan X GPU, making it suitable for practical interactive segmentation application.

### 4.2. Segmentation of Single Object

We first evaluate the performance of our MultiSeg method in segmenting single object by comparing it with the state-of-the-art interactive image segmentation methods on three public benchmarks with instance-level annotations, including PASCAL VOC validation set [8], GrabCut [33] and Berkeley dataset [28].

The standard practice for evaluating the performance of a single-output interactive image segmentation system is as follows: given an initial positive click at the center of the segmentation target, the model outputs an initial prediction.

| Segmentation Models | PASCAL (85% IoU) | GrabCut (90% IoU) | Berkeley (90% IoU) |
|---|---|---|---|
| DIOS [36] | 6.88 | 6.04 | 8.65 |
| RIS-Net [23] | 5.12 | 5.00 | 6.03 |
| ITIS [26] | 3.80 | 5.60 | - |
| DEXTR [27] | 4.00 | 4.00 | - |
| LDN [22] | - | 4.79 | - |
| FCTSFN [16] | 4.58 | 3.76 | 6.49 |
| DIOS (ours) | **3.51** | **1.96** | 4.31 |
| MultiSeg (ours) | 3.88 | 2.30 | **4.00** |
| #clicks | 3.56 | 2.22 | 3.87 |
| #select | 0.32 | 0.08 | 0.13 |

Table 1: Comparison with the state-of-the-art interactive segmentation methods. The values are the average number of clicks to achieve a specific IoU on a given dataset.

Subsequent clicks are iteratively added to the center of the largest wrongly labeled region and this step is repeated until the maximum number of clicks (fixed as 20) is reached. The IoU at each step is recorded. The average number of clicks required to achieve a certain IoU on a particular dataset is reported. The clicks number is thresholded to 20 if the target IoU cannot be achieved within 20 clicks.

However, since our model produces multiple segmentations, we also have to consider the amount of interactions needed when selecting one of the proposals (#select). As described in Section 3.4, given the first positive click, our MultiSeg produces *at most* $N$ segmentations after running NMS. We take the segmentation mask with the highest object score ($\hat{\mathbf{Y}}_{\text{default}}$) as our default segmentation output. Meanwhile, we also find the best segmentation mask *w.r.t.* the ground truth ($\hat{\mathbf{Y}}_{\text{best}}$) and compute its overlap with the default output. If the IoU of the two masks is smaller than $T_1$ (implying $\hat{\mathbf{Y}}_{\text{default}}$ and $\hat{\mathbf{Y}}_{\text{best}}$ are very different) and the relative improvement in IoU is larger than $T_2$, we will increase #select by one and the segmentation output from that branch will serve as the default output for the next round. Subsequent clicks are added in the same way as before. We set $T_1$ and $T_2$ to 0.5 and 0.05, respectively. The total number of interactions (the sum of #clicks and #select) is reported.

The quantitative results are summarized in Table 1. We first notice that our MultiSeg model, despite not being trained specifically to segment single objects, performs comparably to or even outperforms the other state-of-the-art methods that were trained to do so. One could argue that earlier methods use older backbone architectures, and thus our improvement could be based solely on that (though some use similar advanced architectures: DEXTR [27] utilizes ResNet-101 with a PSP head while ITIS [26] uses an Xception-based DeepLabv3+). To further investigate this, we retrained DIOS using our exact same backbone architecture, and we do see a large jump in its performance. We achieve very similar results to the improved DIOS without having the strong bias toward segmenting single object. This demonstrates that our MultiSeg retained the capability to segment single objects while it was learning the ability to generate diverse segmentations. Interestingly, we also notice that MultiSeg outperforms all other methods on the Berkeley dataset as it deals better with samples comprising of multiple objects in this dataset.

| Models | #Clicks | #Select | #Total |
|---|---|---|---|
| DIOS | 9.13 | - | 9.13 |
| MultiSeg-RDL | 8.14 | 3.87 | 12.01 |
| MultiSeg | 7.65 | 0.96 | **8.61** |

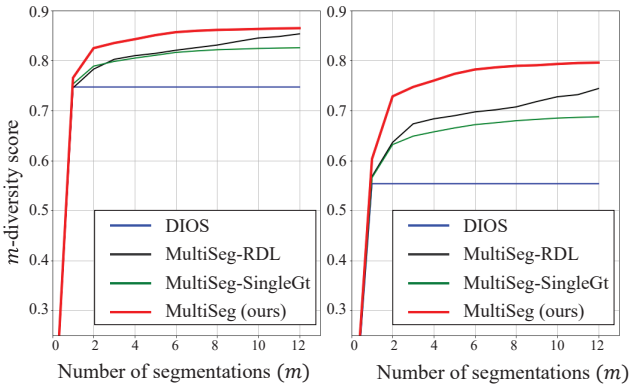
Table 2: Quantitative evaluation on Fashionista dataset.

### 4.3. Segmentation of Local Parts

Here, we examine the generalization capability of the models when segmenting smaller parts within the objects. We use the Fashionista [37] which contains 685 images with 18 categories for testing. For each image, we randomly sample one of the ground truths for evaluation.

In addition to the DeepLabv3+ based DIOS which represents single-output model, we also compare our model with another baseline by replacing the scale-diversity setting with the ranked diversity loss introduced in [22]. Specifically, it was trained with a diversity loss that backpropagates the loss of each ground truth mask through the segmentation branch with the smallest loss during the forward pass and a ranking loss that imposes an ordering on the generated outputs. We call this model MultiSeg-RDL. Note that since all methods use the same backbone architecture, thus allowing us to directly compare single object segmentation, the diversity loss approach and our scale-diversity approach. The evaluation scheme remains the same as before. It is noted that for the MultiSeg-RDL baseline, we always take the prediction from the first segmentation branch as the default proposal regardless if an alternative branch was selected in the previous round since the model was trained to rank its solutions. The results are summarized in Table 2.

Not surprisingly, since the "human" class has been seen in the training, the default segmentation usually covers the full person. Thus, our MultiSeg requires about 1 #select in order to segment the smaller local parts. MultiSeg-RDL trained with ranked diversity loss requires a significant number of #select since it does not reuse the diversity information when an alternative branch is selected. On the other hand, as expected, the single-solution model, DIOS performs poorly when segmenting object parts possibly because the mismatch between training and testing (the model was trained to segment object instances but is asked to segment subparts) introduces a strong bias towards selecting the complete object. Thus, a large number of clicks are needed to "deselect" the body parts. On the other hand, our




(a) (b)

Figure 4: $m$-diversity score on the (a) complete VOC dataset and (b) images with multiple ground truths only.

MutliSeg trained with scale-diversity constrains its proposal to adhere to a predefined scale, thus alleviating the bias despite it was not trained to segment object parts before.

### 4.4. Evaluating Diverse Segmentations

Since the main idea of this paper is to produce a set of diverse segmentations for recommendation to the user, here we evaluate the diverse predictions of our model. Inspired by the k-best oracle evaluation scheme used in most diverse predictions literature [13, 20], we propose a new evaluation metric called $m$-diversity score to evaluate the diverse segmentations when multiple right answers exist. Specifically, given $K$ possible ground truths $\mathcal{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_K\}$ and $M$ ranked diverse predictions $\hat{\mathcal{Y}} = \{\hat{\mathbf{Y}}_1, \hat{\mathbf{Y}}_2, ..., \hat{\mathbf{Y}}_M\}$ from the model, the $m$-diversity score is defined as follows:

$$m\text{-diversity score} = \frac{1}{K} \sum_{\mathbf{Y} \in \mathcal{Y}} \max_{\hat{\mathbf{Y}} \in \hat{\mathcal{Y}}} \text{IoU}(\mathbf{Y}, \hat{\mathbf{Y}}) \quad (4)$$

For each ground truth answer, we find the closest prediction from the $M$ outputs and compute its accuracy in term of IoU. The $m$-diversity score is simply defined as the average accuracy across all the $K$ possible ground truths. Sweeping $M$ (by taking the top $M$ predictions after NMS) allows us to plot a non-decreasing curve of average accuracy against the number of predictions allowed as shown in Figure 4.

We evaluate the segmentation diversity of our model on the PASCAL VOC validation set [8]. For each image, we randomly sample an object and find its neighbor(s) to form a sequence of possible ground truths based on different combinations of objects. Then, we randomly sample a positive and a negative click on the common foreground and background, respectively for evaluation. Figure 4 shows the average $m$-diversity score on the complete VOC validation set. Among all the sampled instances, two-thirds do not have any neighboring object. Thus, we also report the average $m$-diversity score on those samples with neighbors in Figure 4(b).

We first notice that the diversity score of DIOS does not vary with $m$ since it produces a single solution only. We also compared with another baseline which was trained without the simulated training data described in Section 3.3 (MultiSeg-SingleGt). As expected, this baseline performs worse than our full model, suggesting that the explicit synthesis of diverse training samples is beneficial for the generation of diverse solutions. On the other hand, MultiSeg-RDL also performs worse than our full MultiSeg model, indicating that explicitly defining output branches by scale is useful. In comparing Figure 4(a) and 4(b), we see that the gap from MultiSeg to other baseline methods is larger when restricting the evaluation to test with multiple possible ground truth segmentations consistent with the user input.

Next, we also visualize the proposals generated by our MultiSeg, MultiSeg-SingleGt and MultiSeg-RDL in Figure 5. We observe that MultiSeg-RDL yields visually similar results across all the proposals. As it had no mechanism to encourage each branch to produce a unique output, it seemingly never learned the desired diversity. Similarly, MultiSeg-SingleGt trained without the diverse training samples, also produces proposals with limited diversity. On the other hand, our MultiSeg enforces each segmentation branch to extract the most likely region within the corresponding predefined scale. Thus, a diverse set of scale-varying outputs can be obtained with just a single positive click. More qualitative results are shown in Figure 6. Given only a positive and negative clicks, our MultiSeg can generate a wide variety of segmentations. More interestingly, our model can segment object parts (arm, hat and wheels) despite it was not trained with parts annotations before.

### 4.5. User Study

We also conducted a user study to justify the effectiveness of the presented approach with the real human input. We collected 50 images from PASCAL VOC [8], COCO [24], Berkeley [28], Fashionista [37] and DAVIS [30] to compile a benchmark that consists of single object instance, parts, and multiple objects. It should be noted that the same image could be associated with more than one annotation, simulating the practical scenario where anything in the same image could be segmented.

A snapshot of our user interface is shown in Figure 3. The user can choose to either add a positive or negative click using the left and right mouse click, respectively on the main canvas or select one of the proposals suggested on the right panel. The target segmentation (the ground truth that we want our participants to segment) is shown on a separate floating window. The chosen segmentation mask will be overlaid on the main canvas for display.

We recruited five participants where each participant is requested to perform 30 interactive segmentation tasks. Each sample is tested with three different models (DIOS,

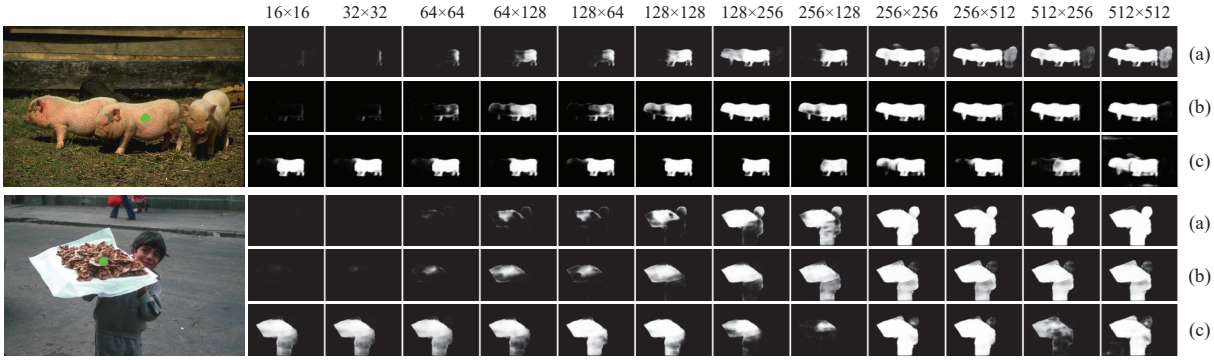
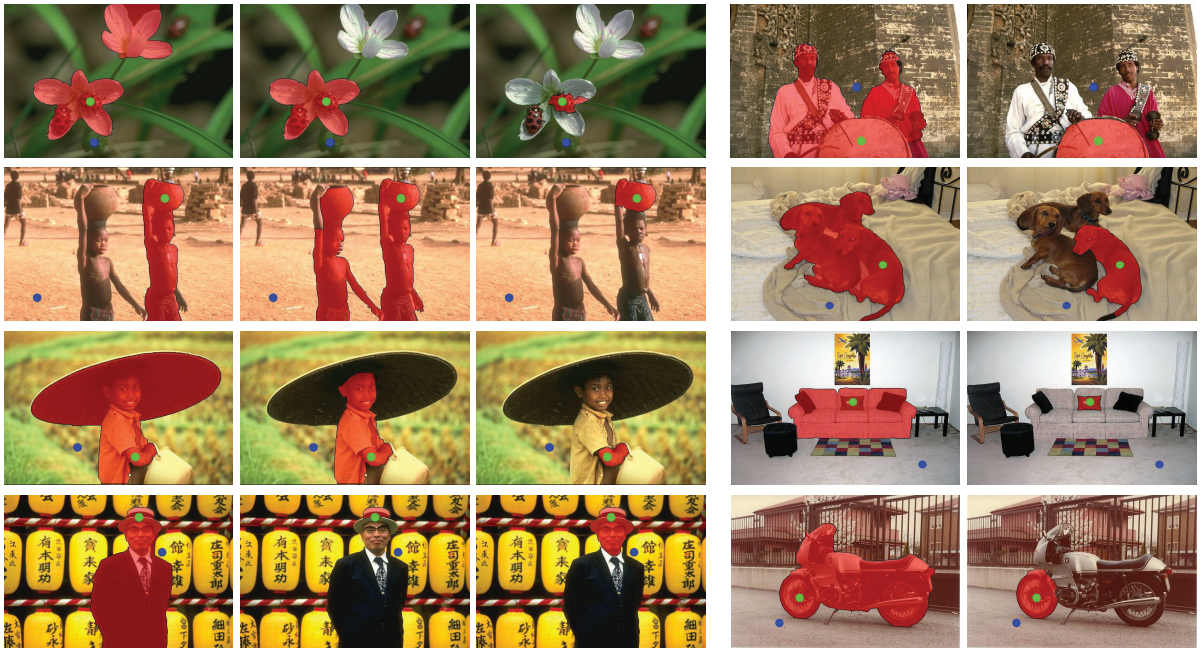



Figure 5: Diverse predictions from (a) MultiSeg, (b) MultiSeg-SingleGt and (c) MultiSeg-RDL given a single positive click only. The input image with the click is shown on the left.



Segmentation 1 Segmentation 2 Segmentation 3 Segmentation 1 Segmentation 2

Figure 6: Given only one positive click (green) and one negative click (blue), the top 2 or 3 segmentation results of our MultiSeg after NMS and graph cut optimization are presented.

MultiSeg-RDL and MultiSeg) and the order of the models is randomized, such that the participants were not aware of the segmentation model driving the interface. We record #clicks, #select and the actual time needed until either 20 clicks are placed or 85% of IoU is reached. The results are shown in Table 3. Our MultiSeg uses the least amount of interactions and time to achieve 85% IoU.

## 5. Conclusion

In this work, we presented MultiSeg, a scale-diverse interactive image segmentation network that incorporates a set of two-dimensional scale priors into the model for generating a set of scale-varying segmentations that are consistent with the user input. To enable this, we also introduced a novel diverse training data generation pipeline to explicitly encourage the model to learn the diversity. Extensive experimental results have demonstrated that our proposed model is capable of producing multiple diverse and semantically meaningful segmentations, being useful for building a more efficient interactive segmentation system.

| Models | #Clicks | #Select | #Total | Time (s) |
|---|---|---|---|---|
| DIOS | 4.62 | - | 4.62 | 8.84 |
| MultiSeg-RDL | 4.24 | 0.14 | 4.38 | 9.47 |
| MultiSeg | 3.30 | 0.24 | **3.54** | **7.55** |

Table 3: User study.

**Acknowledgements** Jiashi Feng was partially supported by NUS IDS R-263-000-C67-646, ECRA R-263-000-C87-133 and MOE Tier-II R-263-000-D17-112. This work is supported in part by gifts from Adobe.

## References


[1] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with Polygon-RNN++. In *CVPR*, 2018.

[2] Xue Bai and Guillermo Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *ICCV*, 2007.

[3] Dhruv Batra, Payman Yadollahpour, Abner Guzman-Rivera, and Gregory Shakhnarovich. Diverse m-best solutions in markov random fields. In *ECCV*, 2012.

[4] Yuri Y Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *ICCV*, 2001.

[5] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. In *CVPR*, 2017.

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018.

[7] Antonio Criminisi, Toby Sharp, and Andrew Blake. GeoS: Geodesic image segmentation. In *ECCV*, 2008.

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

[9] Michael Firman, Neill D. F. Campbell, Lourdes Agapito, and Gabriel J. Brostow. Diversenet: When one right answer is not enough. In *CVPR*, 2018.

[10] Daniel Freedman and Tao Zhang. Interactive graph cut based segmentation with shape priors. In *CVPR*, 2005.

[11] Leo Grady. Random walks for image segmentation. *TPAMI*, 2006.

[12] Varun Gulshan, Carsten Rother, Antonio Criminisi, Andrew Blake, and Andrew Zisserman. Geodesic star convexity for interactive image segmentation. In *CVPR*, 2010.

[13] Abner Guzmán-rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In *NeurIPS*, 2012.

[14] Bharath Hariharan, Pablo Arbelez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

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

[16] Yang Hu, Andrea Soltoggio, Russell Lock, and Steve Carter. A fully convolutional two-stream fusion network for interactive image segmentation. *Neural Networks*, 2019.

[17] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *IJCV*, 1988.

[18] Hoang Le, Long Mai, Brian Price, Scott Cohen, Hailin Jin, and Feng Liu. Interactive boundary prediction for object selection. In *ECCV*, 2018.

[19] Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, David Crandall, and Dhruv Batra. Why M heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.

[20] Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. In *NeurIPS*, 2016.

[21] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. In *ACM ToG*, 2004.

[22] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Interactive image segmentation with latent diversity. In *CVPR*, 2018.

[23] Jun Hao Liew, Yunchao Wei, Wei Xiong, Sim-Heng Ong, and Jiashi Feng. Regional interactive image segmentation networks. In *ICCV*, 2017.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

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

[26] Sabarinath Mahadevan, Paul Voigtlaender, and Bastian Leibe. Iteratively trained interactive segmentation. In *BMVC*, 2018.

[27] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018.

[28] Kevin McGuinness and Noel E O'Connor. Toward automated evaluation of interactive segmentation. *Computer Vision and Image Understanding*, 2011.

[29] Eric N Mortensen and William A Barrett. Intelligent scissors for image composition. In *International conference on computer graphics and interactive techniques*, 1995.

[30] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[31] Brian L Price, Bryan Morse, and Scott Cohen. Geodesic graph cut for interactive image segmentation. In *CVPR*, 2010.

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

[33] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM ToG*, 2004.

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

[35] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Graph cut based image segmentation with connectivity priors. In *CVPR*, 2008.

[36] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. Deep interactive object selection. In *CVPR*, 2016.

[37] Kota Yamaguchi, M Hadi Kiapour, Luis E Ortiz, and Tamara L Berg. Parsing clothing in fashion photographs. In *CVPR*, 2012. Dataset: https://github.com/lemondan/HumanParsing-Dataset.