Two-in-One Refinement for Interactive Segmentation

Soumajit Majumder[1]
majumder@cs.uni-bonn.de
Abhinav Rai[2]
e0403959@u.nus.edu
Ansh Khurana[3]
anshkhurana@iitb.ac.in
Angela Yao[2]
ayao@comp.nus.edu.sg

[1] Institute of Computer Science II
University of Bonn, Germany
[2] School of Computing
National University of Singapore
[3] Indian Institute of Technology Bombay
Mumbai, India

Abstract

Deep convolutional neural networks are now mainstream for click-based interactive image segmentation. Most frameworks refine false negatives and false positive regions via a succession of positive and negative clicks placed centrally in these regions. We propose a simple yet intuitive two-in-one refinement strategy placing clicks on the boundary of the object of interest. As boundary clicks are a strong cue for extracting the object of interest and we find that they are much more effective in correcting wrong segmentation masks. In addition, we propose a boundary-aware loss that encourages segmentation masks to respect instance boundaries. We place our new refinement scheme and loss formulation within a task-specialized segmentation framework and achieve state-of-the-art performance on the standard datasets - Berkeley, Pascal VOC 2012, DAVIS, and MS COCO. We exceed competing methods by $6.5\%, 9.4\%, 10.5\%$ and 2.5%, respectively.

1 Introduction

The goal of interactive image segmentation is to obtain an accurate pixel-level mask for an object with minimal user input. It is a fundamental processing stage for applications such as image editing [4] and medical imaging analysis [40]. More recently, with the increased popularity of deep learning, the demand for mask-level annotations for segmentation tasks has risen dramatically. Manual labelling of such data is highly laborious; a single image can take as long as 1.5 hours for Cityscapes [12]. Alternatively, one can leverage the advances of automated segmentation and use human-in-the-loop frameworks [5, 34, 37].

The holy grail of interactive instance segmentation is to achieve single-click segmentation. The user places one click on the object of interest, and the system returns an accurate mask. Despite significant progress, advanced deep learning-based frameworks [23, 28, 33] still require multiple clicks: one to indicate the object of interest, and others to refine the segmentation mask. In all interactive frameworks [20, 21, 27, 27, 31, 33, 41], refinement is done via a succession of positive and negative clicks. Intuitively, the positive clicks

Figure 1: **Two-in-One Refinement.** Our two-in-one refinement strategy places refinement clicks on the target boundary of error regions. The top row shows sample images with ground truth masks in green. The second row shows an initial segmentation from the *selection* stage overlaid in cyan, with yellow circles marking refinement click locations.

should add portions of missing foreground, while negative clicks remove falsely segmented parts of the background. Users place clicks centrally in regions that directly corresponds with false negatives and false positives. Distinguishing between the two forms of refinement, however, requires separate input encodings, so there are always at least two guidance maps [26, 27, 28, 41]. Additionally, a plateauing occurs, as the average per instance mIoU tends to stagnate beyond 5-6 clicks [27, 31, 33, 41].

In this paper, we do away with positive and negative clicks and propose a novel *two-in-one* refinement strategy. We ask users instead to place clicks on the instance *boundary* in the vicinity of errors (see Fig. 1). Boundary clicks have more utility, as they provide a much stronger cue than clicks placed centrally in regions of false positives and false negatives. Furthermore, as more and more boundary refinement clicks become available, the instance gets explicitly encircled with refinement clicks.

To accommodate our new refinement scheme, we use dedicated networks for object *selection* versus *refinement* (see Fig. 2). Most previous works have used a single network [21, 27, 31, 33, 41], treating the initial selection click simply as any other positive click. In our case, the initial selection click is placed centrally within the object [28], while refinement clicks are placed on object boundaries. As a result, task separation becomes necessary. Having two networks does add some computation overhead in training and inference. To mitigate the impact, we share feature representations across the networks (see Fig. 2). The final refinement network is a lightweight block using three successive convolutions (see Sec. 3.2). In return, separating the tasks allows more freedom for the networks to specialize and thereby achieves state-of-the-art results.

For learning the selection network, we propose a new and highly effective boundary-aware cross-entropy loss. This loss encourages predicted segmentation masks to remain close to the instance boundaries, and through a *single* selection click, restricts the working image space. In contrast, previous works require two to four clicks. We find that isolating the region of interest for segmentation is particularly helpful for small objects, which previous works [23, 33] have already shown are the most difficult to segment.

The key contributions of our approach are summarized as follows:

- We propose a novel *two-in-one* refinement strategy for interactive segmentation where users refine segment masks by clicking on the instance boundary.
- We propose a boundary-aware cross-entropy loss which constrains the prediction and allows us to integrate a *crop* mechanism without the use of bounding boxes [5] or explicit confining clicks at extremal points [5, 34].
- A framework that separates selection and refinement tasks, with which we achieve relative mIoU gains of 10-15% over state-of-the-art that use even deeper backbones.

2 Related Works

Deep-learning approaches [28, 31, 33] based on fully convolutional network architectures [10, 11, 30] now excel at producing good quality segmentation masks even for challenging large-scale datasets [13, 29]. With user-provided cues such as bounding boxes [5], clicks [4, 31, 41], and scribbles [1], these approaches can generate instance segmentation masks with over 90% mean Intersection over Union (mIoU) w.r.t ground truth with less than four user clicks [23, 28, 33].

Of these, click-based interactive frameworks [20, 21, 26, 27, 31, 33, 41] have particularly gained popularity in the last five years. The initial work of [41] encoded user clicks as Euclidean distance maps; the maps are then concatenated with the RGB channels and fed as input to an FCN [30]. Based on the previous prediction errors, further clicks are added, usually on the center of the largest incorrect region [5, 28].

Across most of these approaches, user clicks are transformed guidance maps via Euclidean distance [20, 26, 28, 41] or Gaussians [4, 31, 34]. Alternatively, superpixels [4, 32, 33] and region-based object proposals [33] have also been used to generate guidance maps. Given an initial bounding box, Polygon-RNN [8] predicts the vertices of the polygon outlining the object. Other considerations for interactive segmentation frameworks include a two-stream network [20] and backpropagation refinement scheme [21].

Historically, many classical approaches for interactive segmentation [15, 22, 24, 25, 36] segmented objects by allowing users to interact with boundary pixels. Early variants used boundary tracing [14, 36] but as such approaches relied solely on low-level image features such as gradients [15], they do not work well on unconstrained images. More recently, the DEXTR framework [34] proposed using extremal boundary clicks to select and enclose the object. As DEXTR gives a minimum of four clicks at the outset, it requires significantly more user input than state-of-the-art methods [21, 23, 28, 33].

Recently, Lu *et al*. [24] proposed a deep learning-based framework for interactively finding object boundaries. Unlike [24] and DEXTR, we treat boundary clicks as a means of refining or correcting prediction errors. Click locations are not arbitrary [24] nor constrained to extremal points [34] but conditioned by prediction errors of the preceding selection network (see Fig. 2). Additionally, we do not make use of traditional boundary prediction losses [24, 44]. Finally, we report results on the more challenging Pascal VOC 2012 [13] and MS COCO [29] datasets.

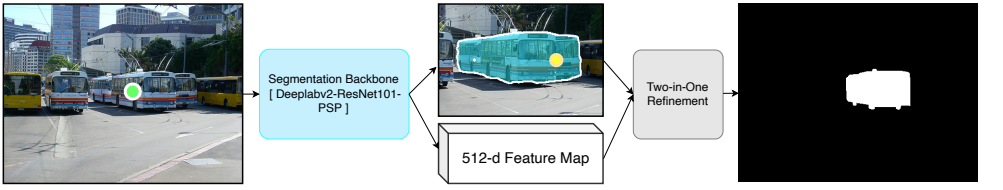


Figure 2: **Outline.** Given an initial selection click (shown in green) on the object of interest, our selection network generates a full-image segmentation, cropping co-ordinates and learned feature representations. Given the initial segmentation, the user refines with clicks on the boundary of the object in the cropped image and the segmentation mask is updated. Further refinement clicks can be added until a suitable segmentation mask is obtained.

3 Proposed Method

We follow the conventional paradigm of [9, 21, 27, 33, 41] which transforms user clicks into *'guidance'* maps of the same size as the input image. The guidance maps are appended as extra channels to the RGB image and given as input to an FCN. To generate the guidance maps, we use Gaussian transformations [9, 34] as it offers a favourable trade-off between simplicity and performance. We initialize an image-sized channel of zeros and place Gaussians with a small standard deviation of 10 pixels at each user click location.

During training, we pass the RGB image channels concatenated with the guidance map from the first click through the selection network. The selection network returns the initial segmentation mask and corresponding feature maps. Based on the initial masks, we crop the prediction and the feature maps. We then determine refinement clicks from the crops, transform them into a guidance map, and append the guidance maps with the cropped feature maps. Finally, we process with a lightweight refinement network to generate a refined segmentation mask. Further refinement clicks can be added iteratively to reach an acceptable segmentation quality. Fig. 2 illustrates this process.

3.1 Selection Network

We initialize our selection network with the weights from Deeplab-v2 [10] pre-trained on Pascal VOC 2012 [13] for semantic segmentation. Deeplab-v2 uses a ResNet-101 [18] as the feature extraction backbone and PSP module [45] for performing multi-scale feature aggregation. Extracted features from Deeplab-v2 has 512 channels. The final 1-channel prediction logits come from a 1×1 convolution. We modify Deeplab-v2 final layer to additionally return the 512-channel features, which we later reuse in the refinement network. We note that other backbones [11, 30] can be used for the selection network as well.

Typically, interactive frameworks are trained to minimize the standard class-balanced binary cross-entropy loss is given by,

$$\mathcal{L}_{\text{CB-CE}} = \sum_{p \in P} w_{y_p} \cdot \text{BCE}(y_p, \hat{y}_p). \tag{1}$$

Here P is the number of pixels in the image, BCE$(\cdot)$ is the standard cross-entropy loss between the label y_p and the prediction $\hat{y}_p$ at pixel location p [30]. w_{y_p} denotes the inverse normalized frequency of ground truth labels $y_p \in \{0,1\}$ of the mini-batch. We observe that

for Eqn. 1, all pixels, irrespective of their location, contribute equally to the loss function. This often leads to noisy and non-continuous segment predictions [2, 42] that require further post-processing [4, 41]. To circumvent this, several methods have explored learning from the distance transform [41, 43] via additional regression losses [2, 42].

To this end, we propose a boundary-aware cross entropy loss (BA-CE) which selectively increases the penalty for incorrect classification of background pixels lying within some distance (in pixels) of the target instance. Unlike [7], we opt for asymmetric penalty formulation. This focuses the network on learning instance-background transitions and prevents disjoint predictions far-off from the target instance. Formally, let $B\ (\in \mathcal{M})$ denote the set of boundary pixels for the ground truth mask $\mathcal{M}$. We define a per-pixel weighting factor $w_{(B,\,p)}$ as a function of the minimum Euclidean distance from the set of boundary pixels $p_b \in B$.

$$w_{(B,\,p)} = \begin{cases} 0 & d(B,\,p) > \tau \ \text{ or } \ p \in \mathcal{M} \\ \frac{d(B,\,p)}{k} & d(B,\,p) \leq \tau \ \text{ and } \ p \notin \mathcal{M} \end{cases} \tag{2}$$

where $d(B,\,p) = \min \|p - p_B\|_2^2 \ \ \forall\, p_B \in B$. τ denotes the width along the instance boundary (in pixels) where the penalty term is enforced. In our experiments, we fix $k = 255$ and $\tau = 40$. Our proposed BA-CE loss function is given as,

$$\mathcal{L}_{\text{BA-CE}} = \sum_{p \in P} w_{y_p} \cdot (1 + w_{(B,\,p)}) \cdot \text{BCE}(y_p, \hat{y}_p). \tag{3}$$

3.2 Refinement Network

Given the first selection click, the selection network returns 512-channel feature map $\mathcal{F} \in \mathcal{R}^{h \times w \times 512}$ and the predicted mask trimmed (see Fig. 2). Based on the trimmed predicted mask, the user annotates additional refinement clicks to undo the prediction errors. We encode these user-provided refinement clicks using Gaussians in a single channel $\mathcal{G} \in \mathcal{R}^{h \times w \times 1}$. We concatenate $\mathcal{F}$ and $\mathcal{G}$ along the feature-map dimension pass it through our lightweight refinement network.

For cropping the initial prediction from the selection network to the target instance, we make use of the RoIAlign layer [19] to obtain $\mathcal{F}_t$ from $\mathcal{F}$. RoIAlign [19] takes as input the region of interest (ROI) coordinates and the feature map and produces a fixed-size representation regardless of ROI size. Instead of RoIAlign [19], naive cropping is also an alternative. We obtain the cropping coordinates by first applying a sigmoid operator on the initial prediction logits and then selecting pixels with values higher than a specific threshold. We pass these coordinates to the RoIAlign layer and obtain features $\mathcal{F}_t$ for the cropped representation. We then concatenate $\mathcal{F}_t$ with $\mathcal{G}_{\mathcal{F}_t}$ and pass it through the refinement network. $\mathcal{G}_{\mathcal{F}_t}$ denotes the user click encoding on the cropped initial prediction. During training, we observed that naively processing raw features $\mathcal{F}_t$ through the refinement network is detrimental. Thus, we perform a channel-wise normalization of $\mathcal{F}_t$ before concatenating it with $\mathcal{G}_{\mathcal{F}_t}$.

Our refinement network consists of 3 convolutional blocks. We denote the i-th block as $C_i(m,n,k)$, which consists of a $k \times k \times m \times n$ convolution followed by batch normalization and ReLU. Here, k refers to kernel size, m the number of input feature channels and n the number of output feature channels. The architecture of our refinement layer is given by,

$$\Big[C_1(513,512,3) \rightarrow C_2(512,256,3) \rightarrow C_3(256,256,3)\Big] \tag{4}$$

The final prediction logits are obtained via 1×1 convolution which squashes the 256-channel into a single channel output. Threshold selection and instance recall are discussed in more detail in Sec. 4.2. We note that our selection network is not trained to minimize the bounding box regression loss [19] as per standard instance segmentation.

3.3 Click Sampling

Like previous works [4, 9, 20, 28, 31, 41], we simulate clicks for training and evaluation.

Selection click sampling. The initial *selection* click is typically assigned on the center of the target object [28]. For convexly shaped instances, the center of mass computed using the ground truth serves as a prior for click initialization. In concave shapes, we shift the initialization to an interior point furthest from the instance boundary. To mimic humans in placing clicks and make our training robust, we further add random displacements of up to 25 pixels to the sampled click location. Visualizations of selection click placements are shown in Fig. 3.

Refinement click sampling. Refinement clicks are sampled according to segmentation outputs from the selection network. Given the prediction and ground truth masks, false positive and false negative regions are computed. The erroneous regions are then clustered based on connectivity [31] while discarding regions with pixels fewer than $m\%$ of the ground truth. We then select a minimum of k_1 clicks and a maximum of k_2 from the remaining error regions. In our experiments, we vary m between 1-5 while k_1 and k_2 fixed at 2 and 4 respectively *i.e.*, the network sees 2-4 refinement clicks to refine each instance.

For sampling refinement clicks corresponding to the false negative regions, a one-sided (directed) Hausdorff distance is used. Let $\mathcal{B}_{FN}$ and π_{FN} denote the set of ground truth boundary pixels and the predicted boundary pixels outlining the false negative region respectively. For the false positive clusters, we opt for the min$-$min formulation. Likewise, let $\mathcal{B}_{FP}$ and π_{FN} denote the set of ground truth boundary pixels and the predicted boundary pixels outlining the corresponding false positive region respectively. The sampled click locations $c_{b,FN} \in \mathcal{B}_{FN}$ and $c_{b,FP} \in \mathcal{B}_{FP}$ corresponding to false negative and false positive regions are given respectively by,

$$c_{b,FN} = \arg\max_{x\in\mathcal{B}_{FN}} \min_{y\in\pi_{FN}} \|x-y\|_2 \qquad \text{and} \qquad c_{b,FP} = \arg\min_{x\in\mathcal{B}_{FP}} \min_{y\in\pi_{FP}} \|x-y\|_2. \tag{5}$$

During inference, it is unlikely that users clicks will align exactly with the object boundary. During training, we randomly perturb the click location up to 10 pixels to compensate.

3.4 Implementation Details

Similar to [23, 27, 31, 33, 41], we simulate user click behavior and use the 1464 training images from Pascal VOC 2012 [13] plus the additional instance annotations from SBD provided by [17]. We further augment with random scaling, flipping, and rotation operations. Unlike [28, 31], we do not use training instances from MS COCO [29].

For training both the selection and refinement network, we minimize the BA-CE loss (Eqn. 3) using stochastic gradient descent with Nesterov (0.9) and apply a fixed learning rate of 10^{-8} across all epochs with weight decay of 0.0005. Training of the selection network typically converges within 80 epochs. For training the refinement network, we pick training

instances with $0.2-0.6$ mIoU based on predictions by the selection network. Training requires $10-15$ epochs; to make the network more responsive to refinement clicks, we increase the pixel contribution to the loss by a scalar factor of 2 within a 40×40 neighbourhood of the sampled refinement click location.

4 Experiments

4.1 Datasets & Evaluation

We evaluate our proposed approach on five publicly available datasets. **Grabcut** [39] and **Berkeley** [35] are have 49 and 96 images respectively, and feature mostly a single prominent foreground object. **Pascal VOC 2012** [13] has 3427 instances across 1449 validation images spread across 20 object categories. **MS COCO** has 5000 validation (val2017) images over 80 categories, 20 of which overlap with Pascal. Like [23, 27, 33, 41], we pick 10 images randomly per category for evaluation and tabulate the 20 seen Pascal versus 60 other unseen categories separately. The **DAVIS** dataset [38] consists of 50 high-resolution videos; following [21, 26], we randomly sample 10% of the frames to get 345 images.

In existing literature, non-interactive semantic segmentation [10, 11, 30] frameworks are evaluated with mean intersection over union (mIoU) [13] by comparing the ground truth mask with the segmentation predictions of the network. Unlike semantic segmentation, in interactive segmentation, user interactions aimed to improve the current network prediction are available. Typically, such interactions come in the form of clicks and should increase the mIoU of the predicted mask by fixing errors in foreground or background areas.

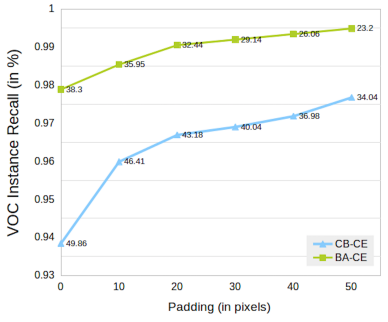
We evaluate our framework with the two standard metrics [4, 31, 41] (a) **clicks@mIoU%** which is the average number of clicks needed to reach fixed mIoU on each instance and (b) **mIoU%@clicks**, the mean intersection over union given k user-assigned selection and/or refinement clicks per instance. In keeping with existing approaches [4, 27, 28, 31, 33, 41], we threshold the number of clicks per instance to 20. The minimum mIoU threshold is set at 90% for the GrabCut and Berkeley dataset. For the more challenging DAVIS, Pascal and MS COCO datasets, the threshold is 85% mIoU [4, 20, 27, 33, 41].

4.2 Selection Network

We train two selection networks to minimize the standard class-balanced BCE (Eqn. 1) and our proposed variant (Eqn. 3). We observe an outright gain of 2.4% over the Pascal VOC 2012 *val* set [13] and significant mIoU gains over recent state-of-the-art methods [31] across the three datasets for the network trained with the BA-CE loss (Eqn. 3). The most significant

	iFCN[41]	ITIS[31]	CAG[33]	*with* CB-CE	*with* BA-CE
Grabcut [39]	62.9	82.1	83.2	84.0	**84.8**
Berkeley [35]	61.3	-	-	82.9	**85.3**
VOC12 [13]	53.6	71.0	74.0	78.2	**80.6**

Table 1: **Selection network.** Average mIoU for the target for competing methods and our framework using a standard class-balanced cross-entropy loss (CB-CE) and our boundary-aware cross-entropy (BA-CE) given only one (selection) click. Best results in **bold**.



(a) VOC Instance Recall in % *vs* padding

(b) Prediction results with CB-CE (Row 1) *vs* BA-CE (Row 2)

Figure 3: **CB-CE *vs* BA-CE.** Comparison of the class balanced cross-entropy (CB-CE) with our boundary aware cross-entropy (BA-CE) loss. Green points denote initial selection clicks.

aspect of the BA-CE loss function is its ability to constrain the network prediction; this is a desirable and necessary property which allows us to crop to the target instance. With a single click, we achieve 80.6% mIoU averaged over Pascal VOC 2012 [13].

For further evaluation, we apply the sigmoid operation on the 1-channel logits predicted by both variants of the selection networks and crop the image region to pixel values > 0.001. To avoid tight cropping, which can lead to undesired boundary artifacts, we relax the cropped region via padding. The instance recall rate for all the validation instances in the Pascal [13] dataset *vs* padding in pixels is shown in Fig. 3(a). The BA-CE network successfully recalls 99.6% instances, *i.e.* the cropped feature map retains the entire object 3414 times out of 3427; for the other 13 instances, we assume a penalty of 20 clicks.

We additionally plot the area reduction (in %-age of the image area) after cropping given a fixed padding (shown in Fig. 3(a)). We observe that the network trained to minimize the CB-CE loss (Eqn. 1) offers more area reduction at the cost of a lower recall. However, this network is undesirable, as we intend to segment all annotated instances in the Pascal VOC 2012 *val* set [13]. Qualitative results are shown in Fig. 3(b). We observe that BA-CE encourages spatially coherent segmentation masks.

4.3 Comparison to the state-of-the-art

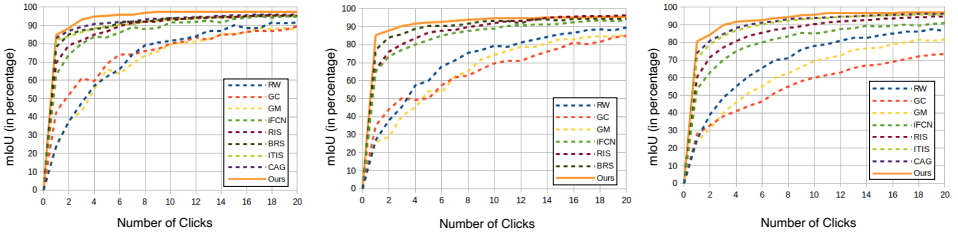
We comparing the **clicks@mIoU%** performance of our method against competing methods [21, 27, 28, 32, 41] in see Table 2. As an ablation study to verify the effectiveness of task separation in the interactive workflow, we also test a variant where the refinement network is trained to accept iterative positive and negative clicks, as per [27, 28, 31, 32, 33, 41] (Ours-*PNR*). In Ours-*PNR*, based on an initial mask from the selection network, users iteratively provide positive and negative clicks; the clicks are then encoded as Gaussians in two separate channels. The rest of the pipeline is unchanged.

This variant outperforms several competing methods, suggesting that existing frameworks can also benefit by task splitting across dedicated networks. Our full variant using the two-in-one refinement (Ours-*BRC*) achieves state-of-the-art on almost all the benchmarks. Our boundary-based refinement offers a 10% improvement in comparison to Ours-*PNR*.

We significantly raise the bar on the Berkeley, Pascal VOC 2012 *val* set, and DAVIS datasets by 6.5%, 9.4% and 10.5% respectively. Additionally, we outperform current state-

Method	Grabcut @90%	Berkeley @90%	VOC-2012 @85%	DAVIS @85%	COCO-20 @85%	COCO-60 @85%
iFCN [41]$_{\text{CVPR'16}}$	6.04	8.65	6.88	-	8.31	7.82
RIS [27]$_{\text{ICCV'17}}$	5.00	6.03	5.12	-	5.98	6.44
ITIS [31]$_{\text{BMVC'18}}$	5.60	-	3.80	-	-	-
CAG [33]$_{\text{CVPR'19}}$	3.58	5.60	3.62	-	5.40	6.10
LIS [32]$_{\text{GCPR'19}}$	3.46	5.18	3.70	-	5.15	5.70
BRS [21]$_{\text{CVPR'19}}$	3.60	5.08	-	5.58	-	-
LFC [23]$_{\text{arXiv'19}}$	3.07	4.94	3.18	5.57	9.14	9.87
MSG [28]$_{\text{ICCV'19}}$	1.96	4.00	3.51	-	-	-
Ours-*PNR*	2.62	4.08	3.27	5.11	5.26	6.31
Ours-*BCR*	2.30	**3.74**	**2.88**	**4.98**	**5.02**	**5.65**

Table 2: Average number of clicks required to achieve a fixed mIoU across state-of-the-art deep-learning -based interactive frameworks. Best results in **bold**.



(a) GrabCut (b) Berkeley (c) Pascal VOC 2012

Figure 4: **mIoU%@clicks** across 3 datasets for our approach *vs* 8 competing methods.

of-the-art [32] on MS COCO [29] *seen* and *unseen* by around 2.5% and 1% respectively. On Grabcut [39], we are comparable with current state-of-the-art MSG [28].

In Fig. 4, we plot our average mIoU *vs* the number of clicks against competing methods RW [16], GC [6], GM [3], iFCN [41], RIS [27], ITIS [31], CAG [33]. Task-separation give us a head-start w.r.t all existing approaches for the first click placement. Overall, our approach shows better performance in most cases.

4.4 User study

We validate the use of boundary clicks for refinement with a user study, using 50 images from Pascal VOC *val*-set with mIoU between 60%-75% as predicted by the selection network. We overlay the prediction of the selection network and also presented the ground truth mask as a separate image. Our 5 participants are tasked with providing refinement clicks on the boundary pixels corresponding to the largest error regions. On average, participants correctly identified the largest erroneous region with a mean accuracy of 88%. Errors may arise when multiple blobs have similar sizes; for example, in Fig. 2, the erroneous region on the left and right boundary of the bus is almost similar in size. Click placements were typically fast, with a mean $\pm$ standard deviation of 3.38 ± 0.8 seconds. Click placements were quite accurate and fell within 2-5 pixels of the instance boundary. For comparison, for each instance, DEXTR [34] acquired 4 extremal clicks within 7 seconds.

5 Conclusion

In this paper, we have proposed a new two-in-one refinement strategy for interactive object segmentation. Instead of conventional positive and negative clicks to fix false negative and false positive segmentations, we ask users to click on object boundaries. To accompany this new refinement scheme, we propose a new boundary-aware loss formulation and a task-specialized framework where dedicated networks are applied to select and refine the object. The refinement network is lightweight and reuses feature maps from the selection network, so additional computational overhead minimal. We find that boundary clicks coupled with our boundary-aware loss provide a strong cue for interactive segmentation and raises state-of-the-art on several benchmark datasets.

Acknowledgment. This work was supported in part by National Research Foundation Singapore under its NRF Fellowship Programme [NRF-NRFFAI1-2019-0001] and NUS Startup Grant R-252-000-A40-133.

References

[1] E. Agustsson, J. Uijlings, and V. Ferrari. Interactive full image segmentation by considering all regions jointly. In *CVPR*, 2019.

[2] N. Audebert, A. Boulch, B. Le Saux, and S. Lefèvre. Distance transform regression for spatially-aware deep semantic segmentation. *CVIU*, 189(102809), 2019.

[3] X. Bai and G. Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *IJCV*, 82(2):113–132, 2009.

[4] A. Benard and M. Gygli. Interactive video object segmentation in the wild. *arXiv preprint:1801.00269*, 2017.

[5] R. Benenson, S. Popov, and V. Ferrari. Large-scale interactive object segmentation with human annotators. In *CVPR*, 2019.

[6] Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *ICCV*, 2001.

[7] F. Calivá, C. Iriondo, A. Martinez, S. Majumdar, and V. Pedoia. Distance map loss penalty term for semantic segmentation. *arXiv preprint arXiv:1908.03679*, 2019.

[8] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a Polygon-RNN. In *CVPR*, 2017.

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

[10] L-C Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40(4):834–848, 2018.

[11] L-C Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[12] 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. In *CVPR*, 2016.

[13] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

[14] A. Falcão, J. Udupa, S. Samarasekera, S. Sharma, B. Hirsch, and R. Lotufo. User-steered image segmentation paradigms: Live wire and live lane. *Graphical Models and Image Processing*, 60:233–260, 1998.

[15] M. Gleicher. Image snapping. In *SIGGRAPH*, 1995.

[16] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann. Random walks for interactive organ segmentation in two and three dimensions: Implementation and validation. In *MICCAI*, 2005.

[17] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017.

[20] Y. Hu, A. Soltoggio, R. Lock, and S. Carter. A fully convolutional two-stream fusion network for interactive image segmentation. *Neural Networks*, 109:31–42, 2019.

[21] W-D Jang and C-S Kim. Interactive image segmentation via backpropagating refinement scheme. In *CVPR*, 2019.

[22] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4): 321–331, 1988.

[23] T. Kontogianni, M. Gygli, J. Uijlings, and V. Ferrari. Continuous adaptation for interactive object segmentation by learning from corrections. *arXiv preprint arXiv:1911.12709*, 2019.

[24] H. Le, L. Mai, B. Price, S. Cohen, H. Jin, and F. Liu. Interactive boundary prediction for object selection. In *ECCV*, 2018.

[25] Y. Li, J. Sun, C-K Tang, and H-Y Shum. Lazy snapping. *ACM Transactions on Graphics (ToG)*, 23(3):303–308, 2004.

[26] Z. Li, Q. Chen, and V. Koltun. Interactive image segmentation with latent diversity. In *CVPR*, 2018.

[27] J.H. Liew, Y. Wei, W. Xiong, S-H Ong, and J. Feng. Regional interactive image segmentation networks. In *ICCV*, 2017.

[28] J.H. Liew, S. Cohen, B. Price, L. Mai, S. Ong, and J. Feng. Multiseg: Semantically meaningful, scale-diverse segmentations from minimal user input. In *ICCV*, 2019.

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

[30] J. Long, E. Shelhamer, and T Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[31] S. Mahadevan, P. Voigtlaender, and B. Leibe. Iteratively trained interactive segmentation. In *BMVC*, 2018.

[32] S. Majumder and A. Yao. Localized interactive instance segmentation. In *GCPR*, 2019.

[33] S. Majumder and A. Yao. Content-aware multi-level guidance for interactive instance segmentation. In *CVPR*, 2019.

[34] K-K Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018.

[35] K. McGuinness and N. O'connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434–444, 2010.

[36] E. Mortensen and W. Barrett. Intelligent scissors for image composition. In *SIGGRAPH*, 1995.

[37] A. Mykhaylo, J. Uijlings, and V. Ferrari. Fluid annotation: A human-machine collaboration interface for full image annotation. In *MM*, 2018.

[38] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[39] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

[40] G. Wang, W. Li, M. Zuluaga, R. Pratt, P. Patel, M. Aertsen, T. Doel, A. David, J. Deprest, S. Ourselin, and T. Vercauteren. Interactive medical image segmentation using deep learning with image-specific fine-tuning. *IEEE Trans. Medical Imaging*, 37(7): 1562–1573, 2018.

[41] N. Xu, B. Price, S. Cohen, J. Yang, and T. Huang. Deep interactive object selection. In *CVPR*, 2016.

[42] Y. Xue, H. Tang, Z. Qiao, G. Gong, Y. Yin, Z. Qian, C. Huang, W. Fan, and X. Huang. Shape-aware organ segmentation by predicting signed distance maps. *arXiv preprint arXiv:1912.03849*, 2019.

[43] Q-Z Ye. The signed Euclidean distance transform and its applications. In *ICPR*, 1988.

[44] Z. Yu, C. Feng, M-Y Liu, and S. Ramalingam. Casenet: Deep category-aware semantic edge detection. In *CVPR*, 2017.

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