Iteratively Trained Interactive Segmentation

Sabarinath Mahadevan
mahadevan@vision.rwth-aachen.de
Paul Voigtlaender
voigtlaender@vision.rwth-aachen.de
Bastian Leibe
leibe@vision.rwth-aachen.de

Computer Vision Group
Visual Computing Institute
RWTH Aachen University
Germany

Abstract

Deep learning requires large amounts of training data to be effective. For the task of object segmentation, manually labelling data is very expensive, and hence interactive methods are needed. Following recent approaches, we develop an interactive object segmentation system which uses user input in the form of clicks as the input to a convolutional network. While previous methods use heuristic click sampling strategies to emulate user clicks during training, we propose a new iterative training strategy. During training, we iteratively add clicks based on the errors of the currently predicted segmentation. We show that our iterative training strategy together with additional improvements to the network architecture results in improved results over the state-of-the-art.

1 Introduction

Recently, deep learning has revolutionized computer vision and has led to greatly improved results across different tasks. However, to achieve optimal performance, deep learning requires large amounts of annotated training data. For some tasks like image classification, manual labels can be obtained with low effort and hence huge amounts of data are available (*e.g.*, ImageNet [10]). For the task of image segmentation, however, the effort to obtain annotations by drawing pixel accurate masks or polygons by hand is much larger. Thus, segmentation datasets (*e.g.*, [11, 19]) are much smaller and cover only a restricted set of categories. Hence, interactive segmentation methods that greatly reduce the amount of manual effort are required to scale up the amount of data. Additionally, interactive segmentation methods can be used for other applications like interactive image or video editing.

Interactive segmentation methods can be based on different kinds of user inputs, such as scribbles or clicks which correct mistakes in the segmentation. In this work, we focus on interactive segmentation of objects using clicks as user inputs [28]. Positive and negative clicks are used by the annotator to add pixels to or to remove pixels from the object of interest, respectively.

Following Xu *et al*. [28], we train a convolutional network which takes an image and some user clicks as input and produces a segmentation mask (see Fig. 1 for an overview of the proposed method). Since obtaining actual user clicks for training the network would require significant effort, recent methods [17, 28] use emulated click patterns. Xu *et al*. [28] use a combination of three different heuristic click sampling strategies to sample a set of

clicks for each input image during training. At test time, they add clicks one by one and sample the clicks based on the errors of the currently predicted mask to imitate a user who always corrects the largest current error. The strategies applied during training and testing are very different and the sampling strategy for training is independent of the errors made by the network. We propose to solve this mismatch between training and testing by aligning the strategies between the training and testing domain, and demonstrate significantly improved results. We further show that the improvements do not merely result from "overfitting" to the evaluation criterion by demonstrating that the results of our method are robust against variations in the click sampling strategy applied at test time.

Additionally, we compare different design choices for representing click and mask inputs to the network. Adopting the state-of-the-art DeepLabV3+ architecture [3] for our network, we demonstrate that applying the iterative training procedure yields significantly improved results which surpass the state-of-the-art both for interactively creating segmentations from scratch and for correcting segmentations which are automatically obtained by a video object segmentation method.

Our contributions are the following: We introduce Iteratively Trained Interactive Segmentation (ITIS), a framework for interactive click-based image segmentation and make code and models publicly available. As part of ITIS, we propose a novel iterative training strategy. Furthermore we systematically compare different design choices for representing click and mask inputs. We show that ITIS significantly improves the state of the art in interactive image segmentation.

2 Related Work

Segmenting objects interactively using clicks, scribbles, or bounding boxes has always been an interesting problem for computer vision research, as it can solve some of the problems in segmentation quality faced by fully-automatic methods.

Before the success of deep learning, graphical models were popular for interactive segmentation tasks. Boykov *et al*. [3] use a graph cut based method for segmenting objects in images. In their approach, a user first marks the foreground and background regions in an image which is then used to find a globally optimal segmentation. Rother *et al*. propose an extension of the graph cut method which they call GrabCut [25]. Here, the user draws a loose rectangle around the object to segment and the GrabCut method extracts the object automatically by an iterative optimisation algorithm. Yu *et al*. [29] further optimise the results for the problem of interactive segmentation by developing an algorithm called LooseCut.

As with most of the other computer vision algorithms, deep learning based interactive segmentation approaches [5, 17, 28] have recently become popular. Those algorithms learn a strong representation of objectness, *i.e*. which pixels belong to an object and which ones do not. Hence, they can reduce the number of user interactions required for generating high quality annotations.

Lin *et al*. [18] use scribbles as a form of supervision for a Fully Convolutional Network (FCN) to segment images. The algorithm is based on a graphical model which is jointly trained with the network. Xu *et al*. propose a deep learning based interactive segmentation approach to generate instance segmentations, called iFCN [28]. iFCN takes user interactions in the form of positive and negative clicks, where a positive click is made on the object that should be segmented (foreground) and a negative click is made on the background. These clicks are transformed into Euclidean distance maps, which are then concatenated with the input channels. The concatenated input is fed into a Fully Convolutional Network to generate

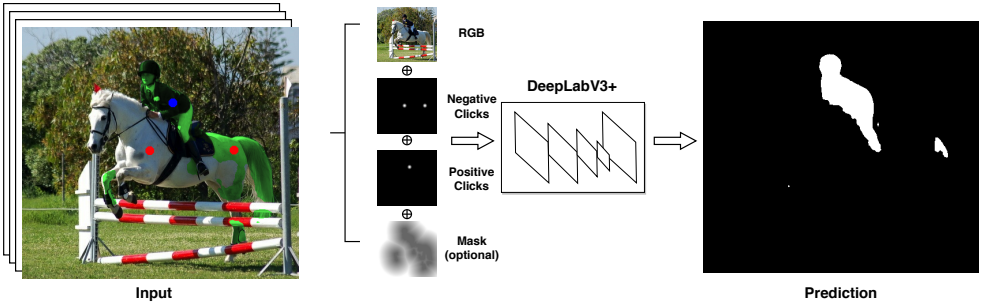


Figure 1: Overview of our method. The input to our network consists of an RGB image concatenated with two click channels representing negative and positive clicks, and also an optional mask channel encoded as distance transform.

the respective output. Our method is inspired by iFCN but extends it with a recent network architecture and a novel training procedure significantly increasing the performance.

A more recent work by Liew *et al*., called RIS-Net [17], uses regional information surrounding the user inputs along with global contextual information to improve the click refinement process. RIS-Net also introduces a click discount factor while training to ensure that a minimal amount of user click information is used and also apply graph cut optimisation to produce the final result. We show that our method achieves better results without the need for graph cuts and the relatively complicated combination of local and global context. However, these components are complementary and could be combined with our method for further improvements.

In contrast to these methods, [20, 21] use extreme points on the objects to generate the corresponding segmentation masks. DEXTR [20] is a deep learning based interactive segmentation method in which these points are encoded as Gaussians and are concatenated as an extra channel to the image which then serves as an input to a Convolutional Neural Network (CNN). While this method produces very good results, it has the restriction that exactly four clicks are used for generating the segmentations. It is difficult to refine the generated annotation with additional clicks when this method fails to produce high quality segmentations.

Castrejon *et al*. [5] propose a Polygon-RNN which predicts a polygon outlining an object to be segmented. The polygons can then interactively be corrected. Their approach shows promising results. However, it requires the bounding box of the object to be segmented as input and it cannot easily be used to correct an existing pixel mask (*e.g*., obtained by an automatic video object segmentation system) which is not based on polygons. Another disadvantage is that it cannot easily deal with objects with multiple connected components, *e.g*., a partially occluded car.

3 Proposed Method

We propose a deep learning based interactive object segmentation algorithm that uses user inputs in the form of clicks, similar to iFCN [28] to generate a segmentation mask for a particular object instance. We propose a novel iterative training procedure which significantly improves the results. Additionally, in contrast to iFCN we encode the user clicks as Gaus-

sians with a small variance and optionally provide an existing estimate of the segmentation mask as additional information to the network.

Figure 1 shows an overview of our method. We concatenate three additional channels with the input image to form an input which contains six channels in total, where the first two non-colour channels represent the positive and the negative clicks, and the (optional) third non-colour channel encodes the mask from the previous iteration. We use the mask channel only for setups where we have an existing mask to be refined, as we found that when starting from scratch the mask channel does not yield any benefits. When an existing mask is given, we encode it as a Euclidean distance transform, which we found to perform slightly better than using the mask directly as input. As the name suggests, positive clicks are made on the foreground, which in this case is the object to segment, and negative clicks are made on the background.

3.1 Network Architecture

We adopt the recent DeepLabV3+ [8] architecture for semantic segmentation, which combines the advantages of spatial pyramid pooling and an encoder-decoder architecture. The backbone DeepLabV3 [7] structure acts as an encoder module to which an additional decoder module is added to recover fine structures. Additionally, DeepLabV3+ adopts depth-wise separable convolutions which results in a faster network with fewer parameters. DeeplabV3+ [8] produces the state of the art performance on the PASCAL VOC 2012 [11] dataset.

All network weights except for the output layer are initialised with those provided by Chen *et al*. [8], which were obtained by pre-training on the ImageNet [10], COCO [19], and PASCAL VOC 2012 [11] datasets. The output layer is replaced with a two-class softmax layer, which is used to produce binary segmentations. In contrast to iFCN [28], we directly obtain a final segmentation by thresholding the posteriors produced by the network at 0.5 and we do not use any post-processing with graphical models.

3.2 Iterative Training Procedure

In order to keep the training effort manageable, we resort to automatic click sampling techniques, for generating clicks, both during training and evaluation. We propose an iterative training procedure, where clicks are progressively added based on the errors in the predictions of the network during training, which closely aligns the network to the actual usage pattern of an annotator. This training procedure boosts the performance of our interactive segmentation model, as shown later in the experiments.

For the iterative training, we use two different kinds of sampling techniques, one for obtaining an initial set of clicks and another for adding additional correction clicks based on errors in the predicted masks. Here, the initialisation strategy initialises each object instance with a random number of clicks, which is chosen from an interval that is decided by a hyper-parameter, and helps the network to learn different notions such as negative objects or object boundaries (see below). The second strategy is useful for learning to correct errors in the predicted segmentation mask. Both strategies help the network learn properties which are useful for an interactive segmentation system.

The training starts for each object with click channels which are initialised with clicks sampled randomly based on the initial click sampling strategy as detailed below. The optional mask channel is initialised to an empty mask, if it is used.

When starting a new epoch, one of the click channels (either positive or negative) is updated with a new correction click which is sampled based on the misclassified pixels in

the predicted mask from the last epoch, according to the iterative click addition algorithm (see below). When adding one click per epoch to each object, after some time the network would only see training examples with many existing clicks and a mask which is already at a high quality. This would degrade the performance for inputs with only few clicks or low quality masks. To avoid this behaviour, at the beginning of an epoch for each object the clicks are reset with probably p_r to a new set of clicks sampled using the initial click sampling strategy (described below). When using the optional mask channel, it is then also reset to an empty mask. The reset of the clicks also introduces some randomness within the training data which reduces over-fitting.

In the following, we describe the initial click sampling and iterative click addition algorithms.

Initial Click Sampling. For this, we initialise the click channels using multiple sampling strategies that try to reproduce the click patterns of a human annotator during training, as done in iFCN [28]. Briefly, iFCN samples positive clicks on the object, and negative clicks based on three different strategies which try to cover multiple patterns such as encoding the object boundary or removing false-positive predictions from background objects. For more details on the sampling strategies, we refer the reader to our supplementary material.

Iterative Click Addition. After the initial set of clicks are sampled using the above strategies, we generate all subsequent clicks for an image with respect to the segmentation mask which was predicted by the network at the previous iteration, as explained below

- First, the mislabelled pixels from the output mask of the previous iteration m_{i-1} are identified by comparing the output mask with the ground truth mask (see Fig. 2 a).
- These pixels are then grouped together into multiple clusters using connected component labelling (see Fig. 2 b).
- The largest of these clusters is selected based on the pixel count.
- A click is sampled on the largest cluster (see Fig. 2 c) such that the sampled pixel location has the maximum Euclidean distance from both the cluster boundary and the other click points within the same cluster. This corresponds to the centre of the cluster if no previous clicks were sampled on it. Here sampling is only used to break ties if multiple pixels have the same distance.
- Finally, the sampled click is considered as positive if the corresponding pixel location in the target image lies on the object, or as negative otherwise. A Gaussian is subsequently added to the corresponding click channel at the sampled location.

4 Experiments

We conduct experiments on four different datasets and compare our approach to other recent methods. On the PASCAL, GrabCut, and KITTI datasets, we consider a scenario where objects are segmented using clicks from scratch. On the DAVIS dataset, we start with the results obtained by an automatic method for video object segmentation and correct the worst results using our method.

4.1 Datasets

PASCAL VOC. We use the 1,464 training images from the PASCAL VOC 2012 dataset [11] plus the additional instance annotations from the semantic boundaries dataset (SBD) [14] provided by Hariharan *et al*. for training our network. This provides us with more than

(a) (b) (c)

Figure 2: An example of the proposed click sampling strategy. a) From all mislabelled pixels (shown in green), b) clusters of mislabeled pixels are identified. c) A click is added on the largest mislabelled cluster after each training round.

20,000 object instances across 20 categories. For our experiments, we use all 1,449 images of the validation set.

GrabCut. The GrabCut dataset [25] consists of 50 images with the corresponding ground truth segmentation masks and is used traditionally by interactive segmentation methods. We evaluate our algorithm on GrabCut to compare our method with other interactive segmentation algorithms.

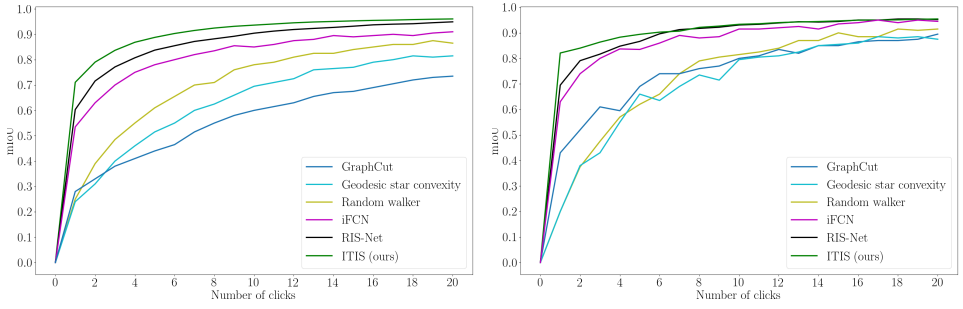
KITTI. For the experiments on KITTI [12], we use 741 cars annotated at the pixel level provided by [6].

DAVIS. DAVIS [23] is a dataset for video object segmentation. It consists of 50 short videos from which 20 are in the validation set which we use in our experiments. In each video, the pixel masks of all frames for one object are annotated.

4.2 Experimental Setup

For training our network, we use bootstrapped cross-entropy [27] as the loss function, which takes an average over the loss values at the pixels that represent the worst k predictions. We train on the worst 25% of the pixel predictions and use Adam [16] to optimize our network. We use a reset probability p_r of 0.3 (*cf*. Section 3.2). The clicks are encoded as Gaussians with a standard deviation of 10 pixels that are centred on each click. We clip the Gaussians to 0 at a distance of 20 pixels from the clicks. Using Gaussians with a small scale localises the clicks well and boosts the system performance, as shown in our experiments. Training is always performed on PASCAL VOC for about 20 epochs. More details are given in the supplementary material.

We use the mean intersection over union score (mIoU) calculated between the network prediction and the ground truth, to evaluate the performance of our interactive segmentation algorithm. For a fair comparison with the other interactive segmentation methods, we also report the number of clicks used to reach a particular mIoU score. For this, we run the same setup that is used in other interactive segmentation methods [17, 28], where clicks are simulated automatically to correct an existing segmentation. The algorithm used to sample the clicks is the same as for iterative click addition during training (*cf*. 3.2). Clicks are repeatedly added until 20 clicks are sampled, and the mIoU score is calculated against the number of clicks that are sampled to achieve it. If a particular IoU score cannot be reached for an instance, then the number of clicks is thresholded to 20 [28].



(a) PASCAL VOC (b) GrabCut

Figure 3: Mean IoU score against the number of clicks used to achieve it on the PASCAL VOC [11] and GrabCut [25] datasets.

4.3 Comparison to State of the Art

We consider two different methodologies to evaluate the amount of interaction required to reach a specific mIoU value. The first method is to count for each object individually, how many clicks are required to obtain a specific IoU value. If the IoU value cannot be achieved within 20 clicks, the number of clicks for this object is clipped to 20 [28]. The results for PASCAL and GrabCut using this method are shown in Table 1 (left). It can be clearly seen from the results that our iteratively trained model requires the least number of clicks on the PASCAL VOC validation set giving a huge advantage over the previous state-of-the-art interactive segmentation methods. An interesting observation here is that our model requires 0.2 clicks less than DEXTR [20], which in fact requires all four extreme points for segmenting an object and hence requires much more human effort compared to our method. Figure 3 a) complements our observations by showing that our model consistently outperforms the other methods on the PASCAL VOC dataset [11]. The curve in Figure 3 b) shows however, that our method produces the best result for the initial few clicks and afterwards performs similar to RIS-Net [17]. To reach the high threshold of 90% on GrabCut [25], our method needs slightly more clicks than RIS-Net [17]. However we argue that this is mainly an effect of the very high threshold which is for many instances very slightly not reached.

The second way of evaluation is to use the same number of clicks for each object instance over the whole validation set, and to increase the number of clicks until the target IoU value is reached. Hence, the number of clicks reported here is actually the average clicks required to attain a particular mIoU score over the validation set of the corresponding dataset. The results for this evaluation are shown in Table 1 (right). With this evaluation strategy, ITIS performs slightly better than RIS-Net [17] on GrabCut and again shows the strongest result on the PASCAL VOC [11] dataset.

4.4 Generalisation to Other Sampling Strategies

To show that our training strategy does not overfit to one particular sampling pattern, we evaluate our method with different click sampling strategies at test time. For this, we use two additional click sampling strategies for correcting the segmentation masks, which we call *cluster sampling*, and *random sampling*. In cluster sampling, first the set of mislabelled clusters is identified using connected components labelling, as described in Section 3.2. A

Method	PASCAL @ 85%	GrabCut @90%
Graph cut [9]	15.0	11.1
Geodesic matting [1]	14.7	12.4
Random walker [13]	11.3	12.3
iFCN[28]	6.8	6.0
RIS-Net[17]	5.1	5.0
DEXTR [20]	4.0	**4.0**
ITIS (ours)	**3.8**	5.6

Method	PASCAL @ 85%	GrabCut @90%
Graph cut [9]	>20	>20
Geodesic matting [1]	>20	>20
Random walker [13]	16.1	15
iFCN[28]	8.7	7.5
RIS-Net[17]	5.7	6.0
DEXTR [20]	4.0	**4.0**
ITIS (ours)	**3.4**	5.7

Table 1: The average number of clicks required to attain a particular mIoU score on PASCAL VOC 2012 and GrabCut datasets. The table on the left shows the values calculated per object instance, and the one on the right shows the corresponding values over the whole validation set.

cluster is then chosen based on a probability proportional to the size of the cluster, and a click is added to the centre of this cluster. For the Random Sampling strategy, we consider the whole misclassified region as a single cluster, and randomly sample a pixel from it. Figure 4 shows the results of our methods with all three sampling strategies. Although smarter sampling strategies, such as cluster sampling, or choosing the largest mislabelled cluster has some advantages for lower number of clicks, this gets neutralised as more clicks are added. The plot shows that our method can achieve similar mIoU scores even with a totally random click sampling strategy further demonstrating that ITIS is robust against user click patterns.

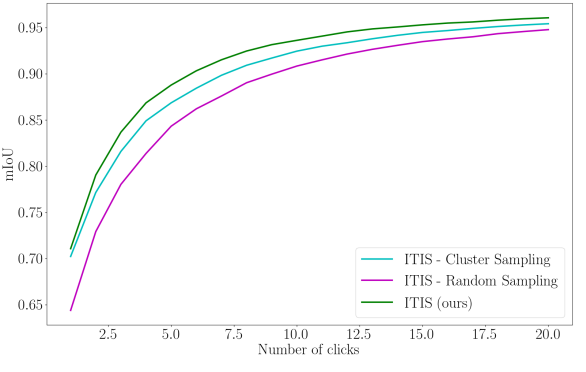


Figure 4: Effect of different click sampling strategies at test time. It can be seen that our method generalizes to alternative sampling methods with only a small loss in performance.

4.5 Ablation Study

We compare different variants of the proposed method in Figure 5. In particular, we investigate the effect of the representation of the clicks on the PASCAL VOC [11] dataset. We use the same evaluation strategy as in Section 4.2 to compare the different models. iFCN [28] uses a distance transform to encode clicks, while DEXTR [20] and Benard *et al.* [2] found that encoding clicks by Gaussians yields better results. Our results also confirm this

Method	OSVOS	1 click	4 clicks	10 clicks
GrabCut[17]	50.4	46.6	53.5	68.8
iFCN[28]	50.4	55.7	71.3	79.9
IVOS [2]	50.4	63.8	75.7	82.2
ITIS - VOS (ours)	50.4	**67.0**	**77.1**	**82.8**

Table 2: Refinement of the worst predictions from OSVOS [4] (performance measured in % mIoU). Our method with an additional mask channel refines the predictions significantly with a few number of clicks.

finding: When we replace the Gaussians by a distance transform, the number of clicks that is required increases from 5.4 to 6.5. The table on the left of Figure 5 also shows that the iterative training strategy greatly reduces the number of clicks needed to reach 85% mIoU on PASCAL VOC from 5.4 to 3.8 clicks. When the optional mask channel is added, which in our case is used to evaluate video object segmentation, the model performs on similar lines in terms of the click evaluation. However, this reduces the performance for the initial 10 clicks as seen in Figure 5 (right). It is also worthwhile to note that the iterative training scheme boosts the maximum mIoU achieved by the model at 20 clicks. Also, the model that uses clicks encoded as distance transform corresponds to the baseline iFCN [28] trained with our DeeplabV3+ [8] network, and hence shows that the strong performance shown by our algorithm is not just a result of using a better network architecture.

Distance Transform	Gaussian	Iterative Training	Mask	Clicks
✓				6.5
	✓			5.4
	✓	✓		3.8
	✓	✓	✓	3.7

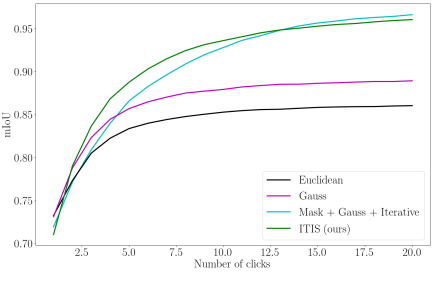


Figure 5: Ablation study on PASCAL VOC. It can be seen, both from the table on the left and the plot on the right, that the proposed iterative training procedure significantly improves the results.

4.6 Correcting Masks for Video Object Segmentation

Many recent works [4, 15, 23, 26] focus on segmenting objects in videos since such object annotations are expensive. These fully-automatic methods produce results which are of good quality but still contain some errors. In this scenario, we are given existing segmentation masks with errors, which can then be corrected by our method using additional clicks. In order to account for the existing mask, we use the optional mask channel as input to the network in this setting. Following [2], we refine the results obtained by OSVOS [4] and report the segmentation quality at 1, 4 and 10 clicks in Table 2. The table shows that our extended network, referred to as ITIS - VOS, produces better results compared to the other methods, especially at clicks 1 and 4.

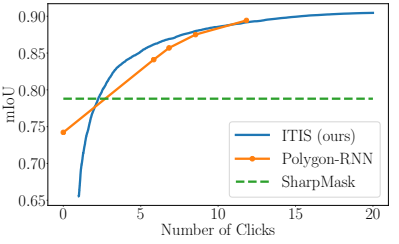


Figure 6: Interactive segmentation performance for segmenting 741 cars on KITTI. For a large range of number of clicks our method performs better than Polygon-RNN although Polygon-RNN uses the ground truth bounding box and requires more manual effort per click.

4.7 Annotating KITTI Instances

In order to compare to Polygon-RNN and to show that our method generalizes to other datasets, we segment 741 cars on the KITTI dataset. The results are shown in Fig. 6, where we also added the result of the fully-automatic SharpMask [24] method for comparison. For the results from Polygon-RNN, clicks are added until all vertices are closer than a specific threshold to their ground truth positions. To create a comparable setup, we define an IoU threshold which should be reached per instance and add up to 20 clicks to each instance until the IoU value is reached. We then vary the target IoU to generate a curve. Note that the shown mIoU in the curve is not the used threshold, but the actual obtained value. Polygon-RNN needs the ground truth bounding box in order to crop the instance which allows it to already produce reasonable results at 0 clicks. In contrast, we work on the whole image without needing the bounding box, which in turn means that ITIS takes a couple of clicks to catch up with Polygon-RNN and from there performs better than Polygon-RNN, converging to a similar value for many clicks. Additionally, correcting a polygon by a click requires significant effort since the click needs to be exactly at the outline of the object while for our method the user just needs to click somewhere in a region which contains errors. Moreover, Polygon-RNN was trained on the Cityscapes [9] dataset, with an automotive setup closer to KITTI, while we focus on a generic model trained on Pascal VOC.

5 Conclusion

We introduced ITIS, a framework for interactive click-based segmentation with a novel iterative training procedure. We have demonstrated results better than the current state-of-the-art on a variety of tasks. We will make our code publicly available at https://github.com/sabarim/itis and hope that it will be used for annotating large datasets.

Acknowledgements. This project was funded, in parts, by ERC Consolidator Grant DeeVise (ERC-2017-COG-773161) and EU project CROWDBOT (H2020-ICT-2017-779942). We would like to thank István Sárándi for helpful discussions.

References

[1] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *ICCV*, 2007.

[2] A. Benard and M. Gygli. Interactive video object segmentation in the wild. *arXiv preprint arXiv: 1801.00269*, 2018.

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

[4] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *CVPR*, 2017.

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

[6] L-C. Chen, S. Fidler, and R. Urtasun. Beat the MTurkers: Automatic image labeling from weak 3d supervision. In *CVPR*, 2014.

[7] L-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[8] L-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018.

[9] 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.

[10] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 2010.

[12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[13] L. Grady. Random walks for image segmentation. *PAMI*, 2006.

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

[15] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele. Lucid data dreaming for object tracking. In *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2017.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

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

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

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

[20] K-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2017.

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

[22] 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.

[23] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A. Sorkine-Hornung. Learning video object segmentation from static images. In *CVPR*, 2017.

[24] P.H.O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, 2016.

[25] C. Rother, V. Kolmogorov, and A. Blake. GrabCut - interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.

[26] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*, 2017.

[27] Z. Wu, C. Shen, and A. van den Hengel. Bridging category-level and instance-level semantic image segmentation. *arXiv preprint arXiv: 1605.06885*, 2016.

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

[29] H. Yu, Y. Zhou, H. Qian, M. Xian, and S. Wang. Loosecut: Interactive image segmentation with loosely bounded boxes. In *ICIP*, 2017.