Continuous Adaptation for Interactive Object Segmentation by Learning from Corrections Supplementary Material

Theodora Kontogianni*‡[2], Michael Gygli*[1],
Jasper Uijlings[1], and Vittorio Ferrari[1]

[1] Google Research, Zurich
[2] RWTH Aachen University, Germany

1 Implementation Details

1.1 Subsampling for image sequence adaptation

We subsample the corrections in the guidance maps to avoid trivial solutions. If all corrections were used equally in the loss and guidance maps, the model could eventually degrade to predict the corrections given the corrections themselves. Specifically it could degrade to only use the information in the guidance maps as its prediction, without relying on image appearance. We avoid this by subsampling the clicks given to the network as guidance, but using all clicks to compute the adaptation loss (Eq. (4)). This forces the network to rely on appearance for propagating the corrections to the rest of the image, where the loss is sparsely evaluated at the pixel locations which were corrected.

1.2 Robustness to image order

Since our *image sequence adaptation* (SA) and *combined adaptation* (IA+SA) methods are processing images sequentially, we tested our method's sensitivity to the image order. We repeated all our experiments 10 times by randomising the image order and computed the variance of the results. We found the variance to be minimal (≤ 0.01 standard deviation) verifying that our adaptation methods are not sensitive to the order in which the images are processed. Hence, we only report averages to improve readability.

1.3 Adaptation parameters

Our adaptation methods use Adam optimizer [8] with learning rate of 10^{-6} and batch size 1. For single image adaptation we do 10 SGD steps and regularize with $\lambda = 1$ and $\gamma = 1$. For image sequence adaptation we do 1 SGD step and use $\lambda = 0.5$ and $\gamma = 2$. For the DRIONS-DB dataset we use a learning rate of 10^{-5}.

* Equal contribution
‡ Work done while interning at Google.

1.4 Model details

We use DeeplabV3+ [3] with Xception-65 [4] as our backbone architecture (pre-trained on ImageNet [5] and PASCAL VOC12 [6]). We extend this model with 2 extra channels for the guidance maps and train it for interactive segmentation model using SGD with momentum and an initial learning rate 0.0002 with polynomial decay. We use a batch size of 2, and atrous rates $\{12, 24, 36\}$. We use in input image resolution of 513×513 and an output stride of 8 for the encoder and 4 for the decoder, respectively. For generating corrections, we sample at most 5 foreground and 5 background corrections for stage one of training (see Sec. 3.3 in the main paper). Corrections are encoded with disks of radius 3.

2 Comparison on the COCO dataset

We have showed that all our adaptation methods are exhibiting substantial improvement compared to the frozen model in many datasets including the COCO dataset. The improvement is especially large on the unseen classes of COCO (16.8% improvement, Table 1 in the main paper) and on adaptation to a particular unseen class (44% improvement for the *donut* class, Table 2 in the main paper), two cases where adaptation is particularly useful.

While we outperform all existing methods on PASCAL VOC12, GrabCut, Berkeley and DAVIS16, some existing works report better clicks@q% than us on COCO. *e.g.* [9] reports 7.86 compared to 9.69 for our method. We however note that these results are not directly comparable. 10 instances are sampled per class to form a test set and the selected instances have not been made available by previous works. But how the selection is done is crucial, as segmenting smaller objects is more challenging. If we ignore objects smaller than 80×80 pixels as in [2], for example, our IA+SA improves from 9.9 to 5.4 (4.5 clicks less). Optimizing the architecture to better handle small objects is however not the focus of our work, as our adaptation methods work with any network architecture and can hence be combined with architectural improvements easily.

3 Single image adaptation algorithm

Algorithm 1 Single image adaptation.

function SINGLEIMAGEADAPTATION(input $\mathbf{x}$, labels $\mathbf{y}$, target iou $\mathcal{J}^t$, initial parameters $\boldsymbol{\theta}^*$, learning rate λ, number of steps k)
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^*$ ▷ Initialize adaptation model
  $\mathbf{c} \leftarrow -\mathbf{1}$ ▷ Start with no corrections
  for $i \leftarrow 0..20$ **do** ▷ Iterate predicting and correcting
    $\mathbf{p} \leftarrow \mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$ ▷ Predict mask
    $\mathcal{J} \leftarrow \text{IOU}(\mathbf{p}, \mathbf{y})$ ▷ Compute the IOU
    if $\mathcal{J} \geq \mathcal{J}^t$ **then** ▷ Stop if mask has required IOU
      return $(\mathbf{p}, |\mathbf{1}[\mathbf{c} \neq -1]|, \mathcal{J})$
    end if
    $\mathbf{c} \leftarrow \mathbf{c} \cup \text{GETCORRECTION}(\mathbf{x}, \mathbf{p}, \mathbf{c})$ ▷ User input
    $\mathbf{x} \leftarrow \text{UPDATEGUIDANCE}(\mathbf{x}, \mathbf{c})$
    for step $\leftarrow 1..k$ **do** ▷ Update model parameters
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \mathcal{L}_{\text{ADAPT}}(\mathbf{x}, \mathbf{p}, \mathbf{c}; \boldsymbol{\theta})$
    end for
  end for
  return $(\mathbf{p}, |\mathbf{1}[\mathbf{c} \neq -1]|, \mathcal{J})$
end function

4 Number of corrections as proxy for segmentation time

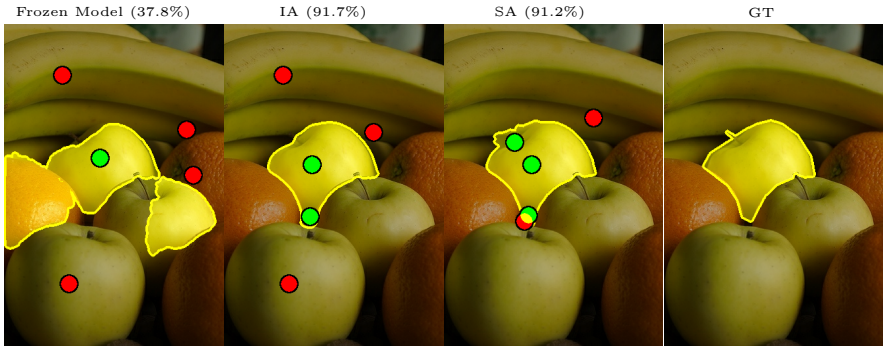
As is common practice [15,10,1,12,9,7], we rely on simulated user corrections to evaluate our method. The number of corrections required to reach a certain segmentation quality serves as a proxy for the total time a user requires to segment an object. When less corrections are needed, segmenting an object is generally faster. But the time for making a correction might vary, as it comprises user interaction time (the time it takes for a user to make a correction) and computation time. We now contrast these two factors for our method.

Benenson [2] performed a rigorous user study on interactive segmentation. There, they find that a user spends ≈ 3 seconds per click correction (not including computation time). For the network used in our work, a single update step takes 0.16 seconds (Sec. 4.6 in the main paper) and the forward pass takes 0.04 seconds. Thus, single image adaptation requires a computation time of $\approx 0.5s$ with 3 update steps. Image sequence adaptation is even faster as it only requires a single update step, which is done after an object is segmented. Hence, given these timings, user interaction time is the dominating factor for the total segmentation time. Reducing the number of corrections required, as in our work, is therefore an effective way to save user time.

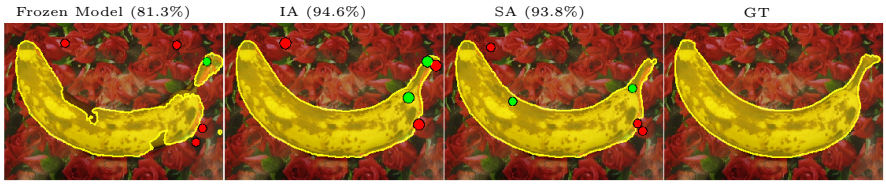
5 Additional Qualitative Results

We show additional results for our two adaptation methods compared to the frozen model in Fig. 1.

(a) Rooftop Aerial dataset [14]



(b) COCO dataset [11]



(c) GrabCut [13]

Fig. 1: **Additional results.** For each image we show results for our two adaptation methods (IA and SA) and the frozen model for the same number of user corrections (IoU@5 is given in parenthesis). When the frozen model is applied to classes that are unseen during training, it sometimes produces segmentation masks that span multiple objects (1a & 1b) or do not respect object boundaries (1c). Single image adaptation (IA) handles such cases much better, by adapting the model parameters to that specific object and its background. This allows it to correctly segment objects even when the foreground and background have similar appearance (1b). Image sequence adaptation (SA) optimizes the model parameters for the test sequence. This allows it to produce good masks from very few clicks, and additional clicks are only required close to the object to refine the exact boundary (1a & 1c).

References

1. Benard, A., Gygli, M.: Interactive video object segmentation in the wild. arXiv (2017)
2. Benenson, R., Popov, S., Ferrari, V.: Large-scale interactive object segmentation with human annotators. In: CVPR (2019)
3. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV (2018)
4. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: CVPR (2017)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-fei, L.: ImageNet: A large-scale hierarchical image database. In: CVPR (2009)
6. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html (2012)
7. Jang, W.D., Kim, C.S.: Interactive image segmentation via backpropagating refinement scheme. In: CVPR (2019)
8. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization. In: ICLR (2015)
9. Li, Z., Chen, Q., Koltun, V.: Interactive image segmentation with latent diversity. In: CVPR (2018)
10. Liew, J., Wei, Y., Xiong, W., Ong, S.H., Feng, J.: Regional interactive image segmentation networks. In: ICCV (2017)
11. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.: Microsoft COCO: Common objects in context. In: ECCV (2014)
12. Mahadevan, S., Voigtlaender, P., Leibe, B.: Iteratively trained interactive segmentation. In: BMVC (2018)
13. Rother, C., Kolmogorov, V., Blake, A.: GrabCut - Interactive Foreground Extraction using Iterated Graph Cut. SIGGRAPH **23** (2004)
14. Sun, X., Christoudias, C.M., Fua, P.: Free-shape polygonal object localization. In: ECCV (2014)
15. Xu, N., Price, B., Cohen, S., Yang, J., Huang, T.: Deep interactive object selection. In: CVPR (2016)