Published in Image Processing On Line on 2024–01–19.
Submitted on 2022–07–04, accepted on 2023–07–27.
ISSN 2105–1232 © 2024 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2024.498

Comparing Interactive Image Segmentation Models under Different Clicking Procedures

Franco Marchesoni-Acland

Université Paris-Saclay, ENS Paris-Saclay, Centre Borelli, Gif-sur-Yvette, France
marchesoniacland@gmail.com

Communicated by Pablo Musé *Demo edited by* Franco Marchesoni

Abstract

Interactive image segmentation (IIS) methods are usually evaluated in terms of segmentation performance vs. number of clicks (NoC). However, the automatic evaluation depends on a clicking procedure and its relation to the procedure used for training. In this work we compare qualitatively and quantitatively two state-of-the-art IIS methods that report the best performances but have not been compared against each other. We show i) what method is better, ii) that the performance is sensitive to clicking procedures, iii) what method is more robust to clicking procedures, and iv) that training with a specific clicking procedure does not guarantee the best performance using it.

Source Code

The reviewed source code and documentation for this algorithm are available from the web page of this article[1]. The original implementation of the method is available here[2].
This is an MLBriefs article, the source code has not been reviewed!

Keywords: interactive-image-segmentation; segmentation; survey

1 Introduction

Annotating data is a key step in the construction of a machine-learned solution. Manual annotation is the only way forward for the problems i) that lack an accurate model (that can be used to simulate target data) and ii) whose target data is not naturally contained in the original database (e.g. as in forecasting applications). Annotation is vital to solve traditional computer vision problems, i.e. classification, detection and segmentation, because it is needed by deep-learning models that are usually the best for these tasks. Annotation for classification is simple. It can be made more efficient with the use of active learning algorithms. Annotation for detection can be obtained from

[1] https://doi.org/10.5201/ipol.2024.498
[2] https://github.com/SamsungLabs/ritm_interactive_segmentation

annotation for segmentation, and in this sense, segmentation annotations are more general and have greater value.

Interactive image segmentation (IIS) methods are used to make the manual segmentation faster, and are complementary to active learning solutions. The IIS methods were at first graphs-based, but with the advent of deep learning, neural-networks based methods have become state-of-the-art. We conducted a survey of most recent IIS methods and found two deep-learning methods [1][5] reporting the best performances. These are described in Section 3. As will be checked later, their performance is very similar and no clear winner can be deduced from the results in the papers. This makes a qualitative and quantitative comparison needed and relevant.

In more detail, the inference of an IIS method is obtained by repeated application of the model. Formally, we define an IIS model as a function f_θ that maps, at step k, an input image x, auxiliary variables z_k and an encoding C_k of the history of clicks, to an estimated output mask $\hat{y}_k$ and new auxiliary variables z_{k+1}

$$\hat{y}_k, z_{k+1} = f(x, z_k, C_k). \tag{1}$$

Here, $C_k = C([c_0 = \emptyset, c_1, \ldots, c_k])$, where C is the click encoding function and $[c_1, \ldots, c_k]$ contains all past interactions, with each c_i containing potentially many clicks. Each click is a tuple (i, j, type) containing the position and type (positive or negative) information. A positive or negative click indicates a foreground or background pixel respectively. In this work, we assume each c_i contains one click only. This represents the case in which the model produces updates after each and every user interaction, i.e. there are no computational constraints. The auxiliary variable z may contain useful information such as the last estimation or the image borders. The final estimate is obtained as $\hat{y}(\hat{y}_K)$ where K is the total number of interactions and the function $\hat{y}$ could implement, for instance, a thresholding operation. Naturally, K needs to be obtained from some stopping criterion. The clicking oracle role and definition will be discussed in Section 2. The full inference procedure, including interaction, is described in Algorithm 1.

Algorithm 1: IIS inference

function $I(f_\theta, O, x, y)$
- **Input** f_θ**:** IIS model as in Equation (1)
- **Input** O**:** Oracle as in Equation (2)
- **Input** x**:** The input image
- **Input** y**:** The target mask to be estimated (only available to oracle)
- **Output** M**:** Sequence of estimations $\{\hat{y}(\hat{y}_k)\}_k$
- **Output** C**:** Sequence of clicks $\{C_k\}_k$
- **Output** K**:** Final K
- $k = 0$
- **while** $k < K$ **and** $c_k \neq$ *END* **do**
  - $\hat{y}_k, z_{k+1} = f_\theta(x, z_k, C_k)$
  - $c_{k+1} = O(x, y, \hat{y}_k, C_k)$
- $K = k$
- $\hat{y}_K, z_{K+1} = f_\theta(x, z_K, C_K)$
- **return** $\{\hat{y}(\hat{y}_k)\}_k, \{C_k\}_k, K$

2 Clicking Procedures

As seen from Algorithm 1, the performance depends on the interaction between the oracle and the IIS model. The oracle is in charge of the clicking. In other words, given a trained IIS model f_θ,

we can expect different performances for different oracles. Many oracles can be thought of, between them i) a human user, ii) an adversarial robot, iii) an optimal robot.

The oracle is whatever entity is doing the clicks. It is called "oracle" because it is the only function that accesses the "target mask to be estimated" y. This is a simplification: for the case in which the oracle is a human, the annotator sees the image x and imagines or knows the desired mask y. It "has access" to the target mask y because it is the annotator who defines or creates y in her head. On the other hand, for a robot clicker, accessing the annotation y of a segmentation database is a requirement to provide informative clicks. Any annotation in a segmentation database is, again, the result of what a previous annotator imagined as target mask.

In order to propose a new click, the oracle considers the current estimate $\hat{y}_k$, the desired mask y, and the click encoding C_k. The image information x can be optionally included: for our simple clicking robots, x is not needed, but for a human, borders and colors might influence where the next click is placed. Even though, for a human, y is in practice accessed by looking at x, a human annotator is also perfectly able to provide informative clicks without visualizing the image x if the target mask y were visible instead. In this paper, we regard accessing x or y as equivalent for the human and do not model the influence of the input image x because i) there are no human annotators to be dealt with and ii) our clicking robots do not use the input image x at all. The oracle is then defined as follows

$$(i, j, \text{type}) = c_k = O(y_k, y, C_k), \tag{2}$$

although in the full generality of an IIS problem, its output at each step could be a sequence of clicks itself and its input could include the image x,

$$[(i_1, j_1, \text{type}_1), \ldots, (i_N, j_N, \text{type}_N)] = c_k = O(x, y_k, y, C_k). \tag{3}$$

Now we introduce a first assumption that simplifies the problem.

Assumption 1 - True click label: Given a click at position (i, j), its label is Positive if placed inside the target mask and is Negative otherwise.

This implies that there is no label noise and we can trust all oracles. This assumption is valid for robot clickers by design. For human annotators, it holds if the annotators have in mind exactly the same target mask and have a way to undo accidental interactions.

From Assumption 1, we can simplify the oracle in Equation (2) to a function $(i, j) = \text{orcl}(y_k, \text{y}, C_k)$ and $c_k = (i, j, \texttt{type}(i, j, y))$, where $\texttt{type}(i, j, y) = y_{ij}$. If an adversarial oracle minimizes performance and an optimal oracle maximizes it, it follows that a random oracle will have a performance between these two.

2.1 Clicking Robots

We implement seven robots that respect Assumption 1 and are roughly ordered by complexity:

1. **Random:** Samples uniformly a location in the image in which to click.
2. **False region random:** Samples uniformly a location in the false region (where prediction and ground truth differ) in which to click.
3. **Largest false region random:** Samples uniformly a location in the largest false region (where prediction and ground truth differ) in which to click. The false region may consist of disconnected components and one of them is largest (we break ties arbitrarily).
4. **False region distance random:** Samples according to a probability map which is given by (up to a constant) the euclidean distance to the border of the false region.

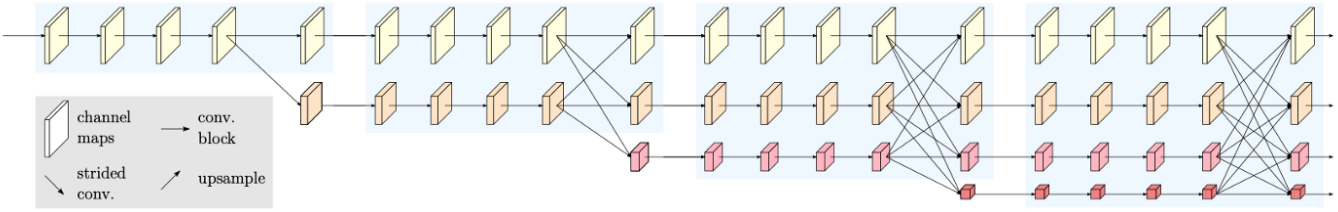


Figure 1: HRNet architecture layout. Extracted from [6].

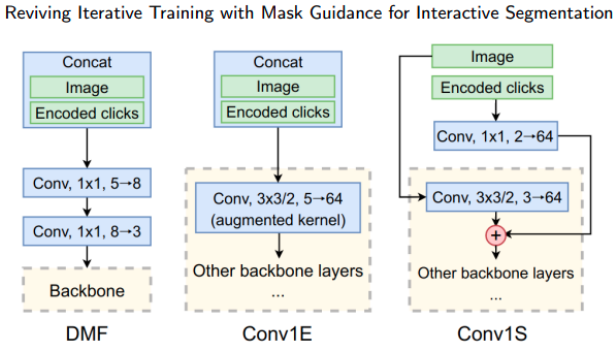


Figure 2: Early fusion variants of click maps for HRNet. The chosen one is Conv1S. Extracted from [5].

5. **False region distance random, click-aware:** Identical to previous for this work. If placing more than 1 click at a time, places them sequentially and considering the previous locations as border.

6. `gto99`: Robot clicker from [1]. Click is placed on the exact center of the largest false region, i.e. on the internal point that is the furthest away from the border.

7. `ritm`: Robot clicker from [5]. Click is placed at the point the furthest away from borders of any false region. This is similar to the previous one, but all false regions are taken into account.

3 Methods

The two IIS models to be evaluated are here presented. Both models take as input an image and input clicks. Moreover, both use as auxiliary variable z the previously estimated mask. This is done by using a (C, H, W) tensor as input, e.g. a six-channel image. The first three channels correspond

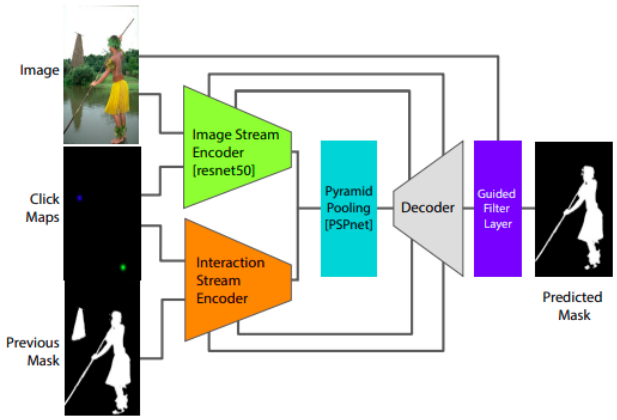


Figure 3: `gto99` architecture. Extracted from [1].

to the input image. The last channels correspond to the last estimated mask, the positive clicks and the negative clicks. The models and their particularities are:

- `ritm`: Model from [5]. The clicks are encoded as disks of radius 5. The used network is HRNet [8], shown in Figure 1, on top of the last early fusion mechanism shown in Figure 2. In this case, the number of channels C is 6. During training, some initial clicks are randomly sampled according to [7], which takes into account the distance between clicks. Then, some interactions occur in which each new click is placed uniformly and randomly in an eroded version of the false region.

- `gto99`: Model from [1]. Takes as input a 10 channel image. The first three channels correspond to the input image. The last one corresponds to the last estimated mask. The positive clicks and the negative clicks are encoded in the other 6 channels with different Gaussians of reported deviations $\{0.02L, 0.04L, 0.08L\}$ (in the code $\{0.02L, 0.08L, 0.16L\}$, $L = 352$) as in [2], using one channel per Gaussian scale and type of click. The architecture is shown in Figure 3. The training of this method is done by running interactions in which each new click is placed exactly at the center of the false region with largest area. The center is defined as the point of a region furthest away from its border.

Both methods output one value per pixel. As this value is obtained after applying a sigmoid function to the network's output and lies in $[0, 1]$, it can be interpreted as a probability. To get a final binary mask $\hat{y}(\hat{y}_K)$, a threshold at 0.5 is applied to the prediction: all values larger than 0.5 are considered foreground.

4 Experiments

4.1 Metrics

We use the usual metrics for image segmentation. As image segmentation is pixel classification, metrics for classification are also useful here. None of these metrics involve spatial relations.

4.1.1 Classification Metrics

Let TP be the number of true positive predictions, i.e. the number of pixels predicted as foreground (positive) that are indeed foreground according to the ground truth (true). Analogously define the False Positive (FP), True Negative (TN) and False Negative (FN) numbers. Then:

- **Accuracy**= $\frac{TP+TN}{TP+TN+FP+FN}$. Measures the exactitude of the predictions made.

- **Precision**= $\frac{TP}{TP+FP}$. Measures how many of the foreground predictions were correct.

- **Recall**= $\frac{TP}{TP+FN}$. Measures how many of the foreground pixels were correctly detected.

4.1.2 Segmentation Metrics

These metrics take into account

- **Dice**= $\frac{2\times TP}{(TP+FP)+(TP+FN)}$. The Dice coefficient measures the ratio between the double of the intersection of estimated and ground truth masks and the sum of their areas.

- **IoU**= $\frac{TP}{TP+FN+FP}$. Intersection over Union measures the ratio between the intersection of estimated and ground truth masks and their union.

4.2 Comparison

The most direct comparison of the two state-of-the-art methods is done with the reported weights and evaluation procedures. This comparison yields the curves in Figure 4. These curves show that **the winner depends on the metric**: `gto99` is clearly the best for Precision and Dice Coefficient but is surpassed by `ritm` in Accuracy and Recall. What the results suggest is that, for a few clicks and under both evaluation methods, `ritm` provides a slightly larger mask than `gto99`. This behavior is checked and visually exposed in Figure 5. A side conclusion of the curves is that the `gto99` clicking procedure achieves slightly better results than the evaluation methodology of `ritm`. In other words, identifying the largest false region before clicking far away from the borders is more informative than not doing so.

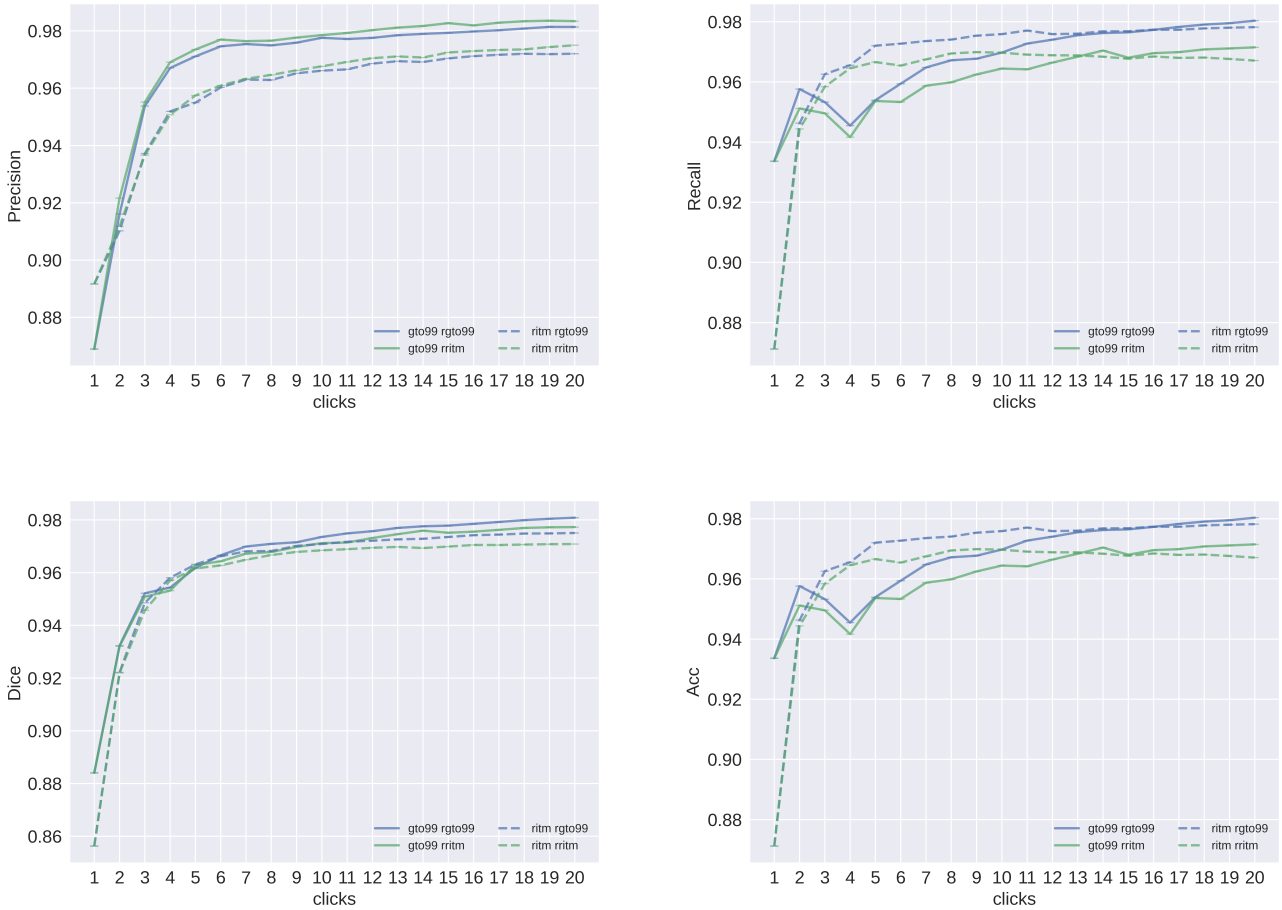
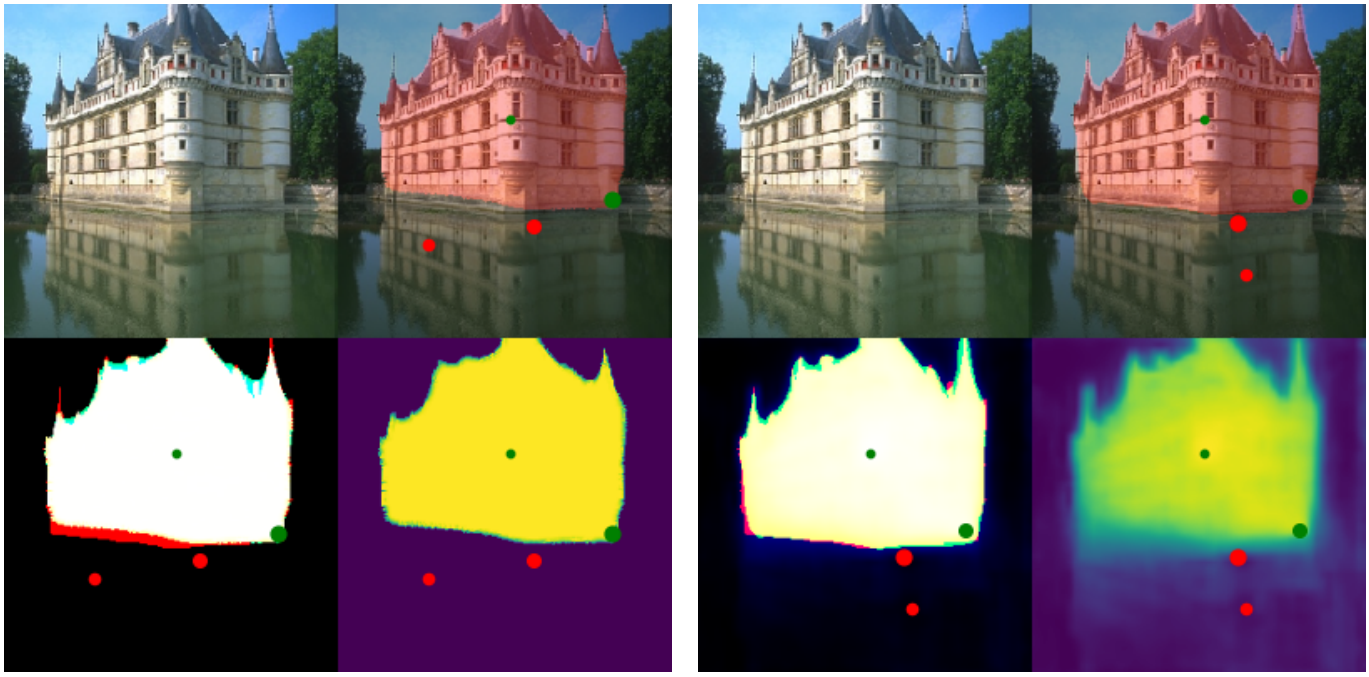


Figure 4: Comparison of `gto99` [1] vs. `ritm` [5] for evaluation procedures in respective papers. In the legend, the first name refers to the network, and the second one, starting with an "r", refers to the robot used to evaluate the network. Curves are the mean of the curves for each image on the Berkeley dataset [4].

In Figure 6(a) one can see that `gto99` surpasses `ritm` in IoU for both evaluation procedures. Although the difference is small, it is consistently in favor of `gto99`, what makes us conclude that it is the very best method. This is for an evaluation conducted over a subset of the Berkeley dataset BSD300 [3] originally introduced by [4] and used by [5] for testing, available in the ritm repository[3]. This conclusion is in sharp contrast with the conclusion taken from Figure 6(b), where `ritm` consistently surpasses `gto99` for virtually all the clicking procedures, being slightly worse on the

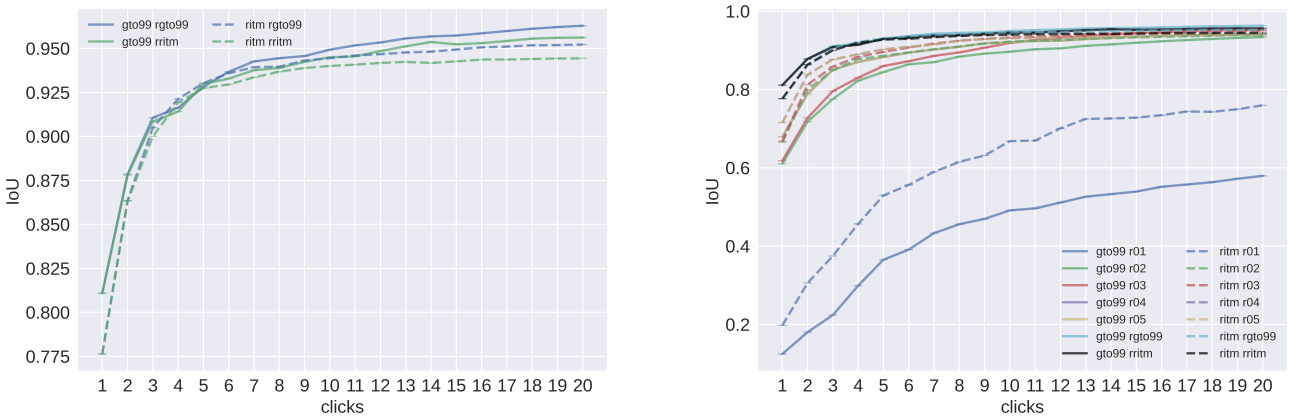
[3] https://github.com/saic-vul/fbrs_interactive_segmentation/releases/download/v1.0/Berkeley.zip



(a) gto99 (b) ritm

Figure 5: Qualitative comparison of gto99 [1] vs. ritm [5] with 4 clicks following the gto99 clicking procedure. Top-left: input image. Top-right: input image overlayed with thresholded prediction and clicks. Bottom-left: TP (white), TN (black), FP (blue), FN (red) when assuming thresholded prediction, intermediate colors otherwise. Green clicks are positive and red clicks are negative. Larger clicks are more recent clicks.

most informative ones, which are presumably unrealistic. In other words, **gto99 is slightly superior under the most sophisticated evaluation protocols, but ritm presents substantially better robustness to the clicking procedure**, which makes it presumably better for dealing with the variability of real human users (that sometimes do click in the borders of the objects). This is also confirmed under all the metrics in Figure 7.



(a) evaluation as in respective papers (b) evaluation under all clicking procedures

Figure 6: Comparison of gto99 [1] vs. ritm [5] for different clicking procedures. Curves are the mean of the curves for each image on the Berkeley dataset [4]. Each curve corresponds to an IIS method, i.e. gto99 or ritm, evaluated with a clicking procedure, i.e. one of r01, r02, r03, r04, r05, rgto99 or rritm. These clicking procedures correspond to the robots 1 to 7 of Section 2.1, respectively.

One extreme example of the huge difference for the random clicking procedure is observable in Figure 8, where 20 random clicks were placed (same location for both methods) and `gto99` and `ritm` did terrible and terrific jobs respectively. This behavior suggests that **the training procedure impacts the final robustness**. On the one hand, `ritm` was trained with some initial randomly placed clicks and subsequent clicks had some noise, although they were close to the center of the false region. Because click ordering information is not provided to the models, `ritm` does not have many priors suggesting "a positive click is in the exact center of a region to be labeled". But this prior is totally built into `gto99`. This makes `gto99` estimate regions that place positive clicks close to the center much more than `ritm`. This behavior can also be seen in Figure 9, where `ritm` does not preserve as much symmetry around the positive click as `gto99` does.

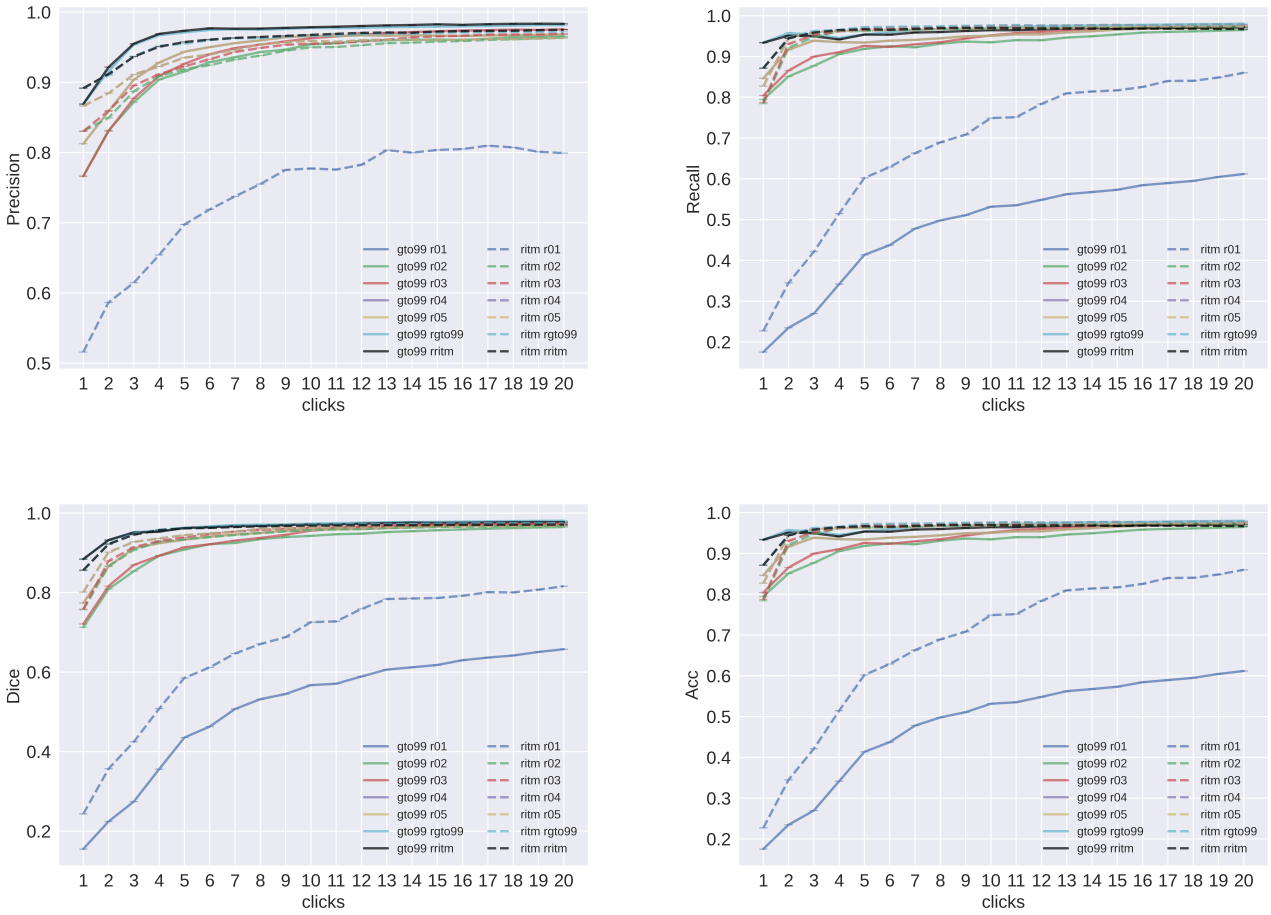
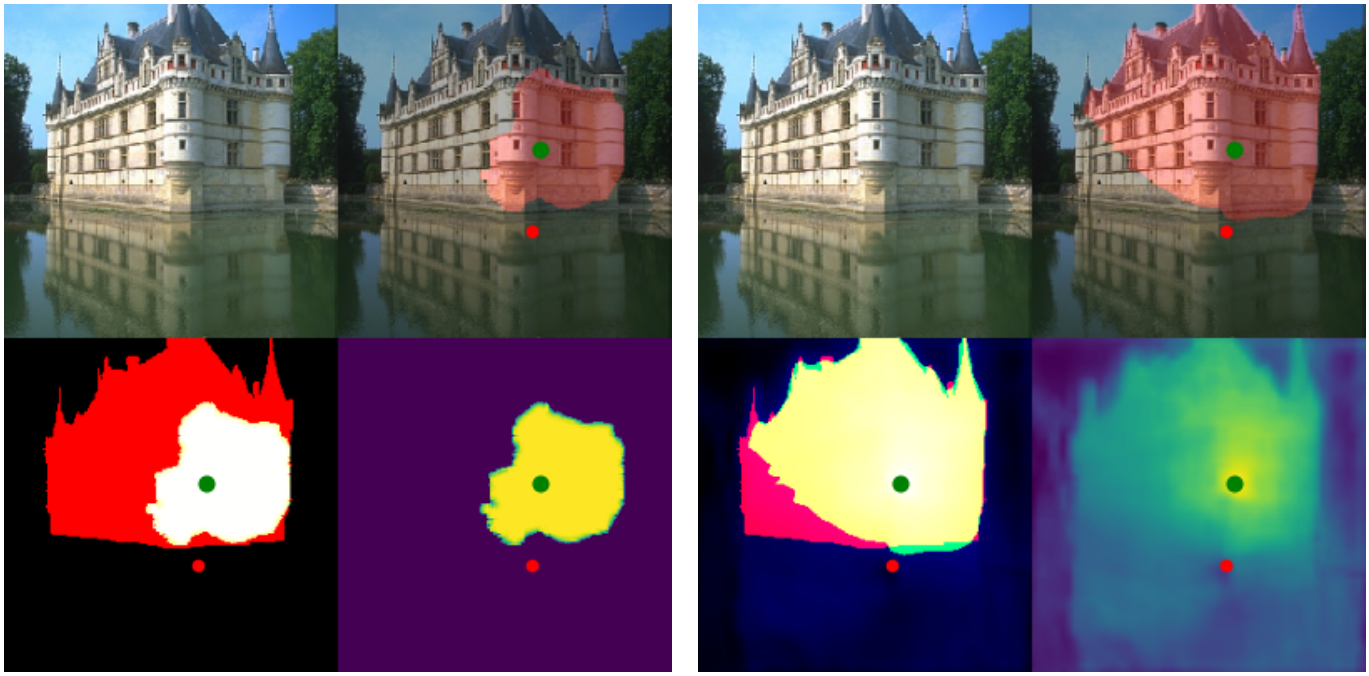


Figure 7: Comparison of `gto99` [1] vs. `ritm` [5] for different clicking procedures under different metrics. Curves are the mean of the curves for each image on the Berkeley dataset [4]. Each curve corresponds to an IIS method, i.e. `gto99` or `ritm`, evaluated with a clicking procedure, i.e. one of r01, r02, r03, r04, r05, rgto99 or rritm. These clicking procedures correspond to the robots 1 to 7 of Section 2.1, respectively.

(a) gto99 (b) ritm

Figure 8: Qualitative comparison of gto99 [1] vs. ritm [5] with 20 clicks following the random clicking procedure. Top-left: input image. Top-right: input image overlayed with thresholded prediction and clicks. Bottom-left: TP (white), TN (black), FP (blue), FN (red) when assuming thresholded prediction, intermediate colors otherwise. Green clicks are positive and red clicks are negative. Larger clicks are more recent clicks.



(a) gto99 (b) ritm

Figure 9: Qualitative comparison of gto99 [1] vs. ritm [5] with 2 clicks following the random clicking procedure. Top-left: input image. Top-right: input image overlayed with thresholded prediction and clicks. Bottom-left: TP (white), TN (black), FP (blue), FN (red) when assuming thresholded prediction, intermediate colors otherwise. Green clicks are positive and red clicks are negative. Larger clicks are more recent clicks.

5 Demo

The demo provided allows one to try the automatic evaluation of one of these state-of-the-art methods (`gto99`, `ritm`) using one of the provided clicking procedures (a total of 6 different) over one specific (image, selected mask) pair. The mask uploaded should be an image of the same size than the input image, and all the pixels where the channel mean is smaller (larger) than 128 will be considered background (foreground). The clicking robots are run until 20 clicks are placed.

The demo provides as output the performance vs. NoC curve for each metric, e.g. Figure 10, and a visualization similar to Figure 5.

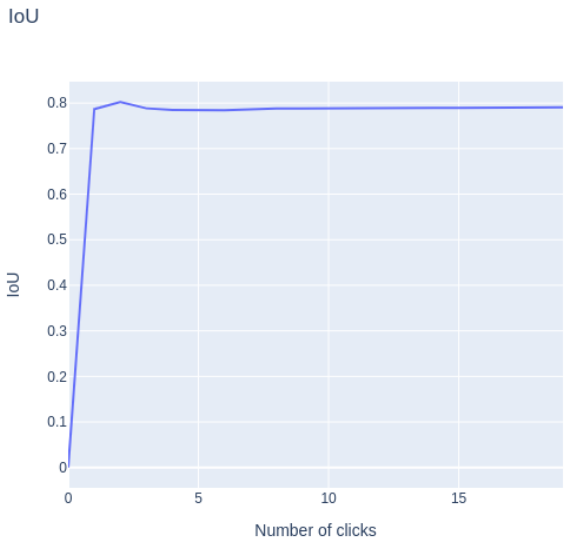
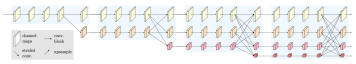


Figure 10: Example output of the demo. In this particular case, IoU is reported for each step over a single image. The demo also yields curves for Acc, Dice, Fscore, Precision, and Recall metrics.

Image Credits

 Extracted from [6].

 Extracted from [5].

 Extracted from [1].

 Extracted from [3].

References

[1] M. Forte, B. Price, S. Cohen, N. Xu, and F. Pitié, *Getting to 99% Accuracy in Interactive Segmentation*, 2020, https://doi.org/10.48550/arXiv.2003.07932.

[2] H. Le, L. Mai, B. Price, S. Cohen, H. Jin, and F. Liu, *Interactive Boundary Prediction for Object Selection*, in European Conference on Computer Vision (ECCV), 2018, pp. 18–33, https://doi.org/10.1007/978-3-030-01264-9_2.

[3] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics*, in IEEE International Conference on Computer Vision (ICCV), vol. 2, IEEE, 2001, pp. 416–423, https://doi.org/10.1109/ICCV.2001.937655.

[4] K. McGuinness and N. E. O'connor, *A Comparative Evaluation of Interactive Segmentation Algorithms*, Pattern Recognition, 43 (2010), pp. 434–444, https://doi.org/10.1016/j.patcog.2009.03.008.

[5] K. Sofiiuk, I. A. Petrov, and A. Konushin, *Reviving Iterative Training with Mask Guidance for Interactive Segmentation*, 2021, https://doi.org/10.48550/arXiv.2102.06583.

[6] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang, *High-Resolution Representations for Labeling Pixels and Regions*, 2019, https://doi.org/10.48550/arXiv.1904.04514.

[7] N. Xu, B. Price, S. Cohen, J. Yang, and T. S. Huang, *Deep Interactive Object Selection*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 373–381, https://doi.org/10.1109/CVPR.2016.47.

[8] Y. Yuan, X. Chen, and J. Wang, *Object-Contextual Representations for Semantic Segmentation*, in European Conference on Computer Vision (ECCV), Springer, 2020, pp. 173–190, https://doi.org/10.1007/978-3-030-58539-6_11.