Efficient Mask Correction for Click-Based Interactive Image Segmentation

Fei Du, Jianlong Yuan, Zhibin Wang, Fan Wang
Alibaba Group
{dufei.df, gongyuan.yjl, zhibin.waz, fan.w}@alibaba-inc.com

Abstract

The goal of click-based interactive image segmentation is to extract target masks with the input of positive/negative clicks. Every time a new click is placed, existing methods run the whole segmentation network to obtain a corrected mask, which is inefficient since several clicks may be needed to reach satisfactory accuracy. To this end, we propose an efficient method to correct the mask with a lightweight mask correction network. The whole network remains a low computational cost from the second click, even if we have a large backbone. However, a simple correction network with limited capacity is not likely to achieve comparable performance with a classic segmentation network. Thus, we propose a click-guided self-attention module and a click-guided correlation module to effectively exploits the click information to boost performance. First, several templates are selected based on the semantic similarity with click features. Then the self-attention module propagates the template information to other pixels, while the correlation module directly uses the templates to obtain target outlines. With the efficient architecture and two click-guided modules, our method shows preferable performance and efficiency compared to existing methods. The code will be released at https://github.com/feiaxyt/EMC-Click.

1. Introduction

Interactive image segmentation aims to select the object of interest with minimal iterative interactions, which can benefit various computer vision tasks, such as semantic segmentation [30], instance segmentation [17], and medical image analysis [28]. As the success of these tasks often requires large-scale mask-level annotations and it is time-consuming to annotate the image manually, interactive segmentation is an attractive way to simplify the annotation process and alleviate the annotation cost.

Different interaction strategies have been studied to simplify the interactive process, including bounding boxes [23, 37, 48], clicks [33, 40, 47], scribbles [1, 2], boundary

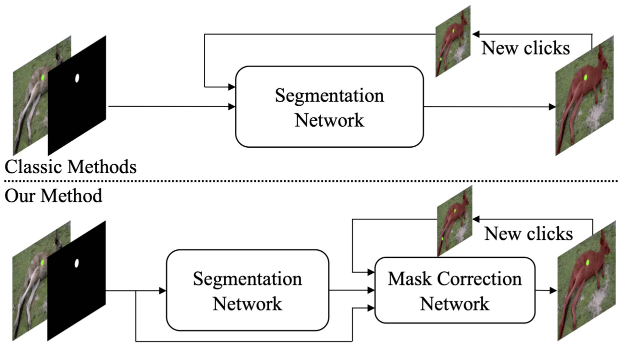

Figure 1. Comparison between the architectures of our method and classic methods. Classic methods run the segmentation network in every iteration, which is inefficient if a large network is adopted. We update the mask via a lightweight mask correction network, enabling an efficient interaction.

points [32], and extreme points [34]. However, due to the complexity of the object boundary and appearance, the performance of methods based on bounding boxes may drop if the bounding boxes are not tightly drawn [48]. Besides, it requires more effort to identify the object boundary and place bounding boxes, boundary points, or extreme points on the image. In contrast, the click-based method only requires the users to progressively place positive and negative clicks on the foreground and background areas, respectively. It has recently attracted more attention due to its simplicity and well-established training and evaluation protocols [40,47]. Hence, we focus on the click-based method.

In click-based interactive segmentation methods, the model returns a corrected prediction mask after each click from the users. Efficiency is of great importance to interactive segmentation methods since a typical segmentation process requires several interactive iterations. In recent years, deep learning-based interactive segmentation methods have achieved considerably better performance compared to traditional methods. However, some recent works [20, 22, 39] achieve state-of-the-art performance by employing inference-time optimization to refine the masks, which significantly increases the computational cost. To eliminate online optimization, RITM [40] investigates different designs for interactive segmentation and achieves state-of-the-art results with a classic segmentation network. To improve efficiency, FocalClick [8] proposes target crop and focus crop strategies to segment two selected local regions efficiently. Most click-based methods [7, 8, 27, 31, 40, 47] iteratively run the whole segmentation network to update the masks with the input of user clicks, which is time-consuming especially when we have a strong segmentation network. Majumder *et al*. [32] proposes to refine the mask with a lightweight refinement network. However, the network requires the user to place boundary points, which is not user-friendly. Therefore, more efficient click-based interactive segmentation methods are still required to reach a better trade-off between performance and efficiency.

With the above consideration, in this work, we develop an efficient click-based interactive segmentation model based on an Efficient Mask Correction network (EMC-Click). The model consists of a base segmentation network and a lightweight mask correction network. The base network takes the first click as input and outputs target-aware features and a coarse prediction. The mask correction network iteratively updates the prediction whenever a new click is placed. The difference between our network and the classic network is shown in Fig. 1. This decoupled design ensures that the computational cost of each iteration is relatively low from the second click. However, using a simply designed mask correction network to update the prediction can hardly achieve comparable performance with classic methods that use a strong segmentation network. Thus, we propose two feature augmentation modules, including a click-guided self-attention module and a click-guided correlation module, to effectively exploit the click information to augment the features in the mask correction network. We first extract the click features and enrich them by selecting several template features based on the semantic similarity between the clicks and other pixels. The self-attention module propagates the information of the template features to other pixels, and the correlation model directly learns target contours. Both modules effectively and efficiently augment the features and improve the segmentation performance. Experiments on five benchmarks demonstrate that our method achieves competitive performance and higher efficiency compared with state-of-the-art click-based interactive segmentation methods.

Our contribution can be summarized as follows:

- We propose EMC-Click, an efficient click-based interactive method, to correct the masks via a lightweight mask correction network interactively. Our method significantly reduces computational cost from the second click especially when we have a large backbone.
- We propose two feature augmentation modules to improve the segmentation accuracy by effectively exploiting the click information.
- We build EMC-Click on different base segmentation networks and evaluate our models on several benchmarks. The results show that our method achieves preferable performance and efficiency compared to state-of-the-art methods.

2. Related works

2.1. Interactive image segmentation

Interactive image segmentation is a longstanding research topic in computer vision. In the early years, methods based on optimization on a graph over image pixels were the mainstream. GrabCut [37] is a classic method that extend [5] to an iterative energy minimization problem. Methods based on random walk [13, 21] build an undirected graph on the image to estimate the probability of each pixel. Since early methods use handcrafted features, they can hardly generalize to complex scenes.

Deep learning has been applied in many fields due to its high representation power and the ability to model complex structures. Xu *et al*. [47] are the first to propose a CNN-based interactive segmentation method. They encode the positive/negative clicks as positive/negative distance maps and concatenate them with the image to feed them into the network. They also design strategies to sample clicks for training, which are adopted by many subsequent works [8, 15, 40]. Later, Mahadevan *et al*. [31] add an iterative sampling strategy on [47] during training. BRS [20] proposes optimizing the network online based on user-specified location information during the annotation process. f-BRS [39] accelerates BRS [20] by optimizing a small part of the network. Recently, RITM [40] conducts comprehensive experiments to investigate the design principle of an end-to-end click-based interactive segmentation method. Although the click-based method is the mainstream due to its simplicity and well-established training and evaluation protocols, other interactive methods are also exploited, such as extreme points [34], boundary points [32], and bounding boxes [46]. Despite the impressive progress made by deep learning-based methods, most of them suffer from a high computational cost especially when they have a strong backbone like ResNet101 [18].

2.2. Mask refinement in interactive segmentation

Mask refinement is commonly used in interactive segmentation to get fine-level results. They often follow the coarse-to-fine schema and crop a local region to make local corrections. For example, RIS-Net [24] samples multiple regions based on the click positions for local refinement.

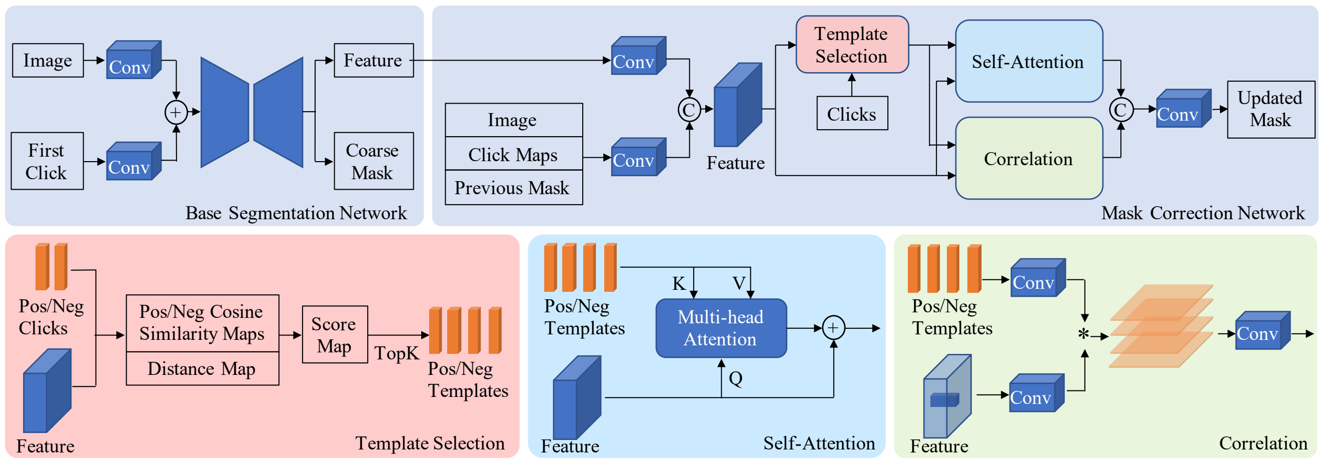

Figure 2. The framework of the proposed EMC-Click. We adopt a base segmentation network to extract target-aware features and make the first prediction by taking the first click as input. The mask correction network is responsible for updating the mask when a new click is placed. In this mask correction network, we first map the click location to the feature map and extract the click features. Then we sample several template features that are similar to the click features. The click-guided self-attention module propagates the template information to other pixels, while the click-guided correlation module extracts target outlines via the correlation between the templates and the feature map. Both modules augment the features, and the combined features are used to predict the mask.

FocalClick [8] crops a region based on the difference between the previous mask and the coarsely predicted mask to refine the details. Zhang *et al*. [50] refines a local area centered on the target based on the coarse segmentation results. There are also methods to refine the global mask, such as EdgeFlow [15] and 99%AccuracyNet [12]. One common property of these refinement methods is that they run the whole segmentation and refinement networks to get the results in each interactive iteration. It is too costly compared with a simple feedforward segmentation network. FocusCut [26] trains a segmentation network to not only segment the target object but also refine local details. However, it is still not efficient enough to run the whole segmentation network in each iteration. Our mask correction network is somewhat like a refinement network. The difference is that we directly use this network to correct masks when a new click is placed, and the base segmentation network only runs for the first click. This architecture reduces the inference time significantly compared with traditional methods.

2.3. Exploitation of click information

The clicks provide key information for locating the foreground and background areas. Most methods encode the clicks as click maps where the regions around the clicks are highlighted. However, the click information is not explicitly exploited. To better exploit the click information, CDNet [7] propagates the click information to other pixels via two non-local networks [43]. It constructs weight maps based on the click locations to re-weight the affinity matrix, which is not computational efficiency since it needs to model the affinity between every two pixels. IFPN [50] learns a sparse graph model to perform click feature propagation. It only models the dependencies between the clicks and other pixels. Inspired by the two methods, we construct a multi-head self-attention module to propagate the click information to other pixels. Different from CDNet and IFPN, we select several templates to enrich the click features. Thus, our method is more efficient compared to CDNet [7] and can exploit more information compared to IFPN [50]. Besides, we also propose a correlation module to exploit the click information explicitly.

3. Proposed method

We propose to iteratively correct the segmentation mask using a lightweight mask correction network, which exploits the information of the clicks to boost the performance via a click-guided self-attention module and a click-guided correlation module. We first briefly revisit the traditional pipeline of the interactive segmentation model, then introduce our pipeline and elaborate on the mask correction network.

3.1. The standard pipeline

Interactive segmentation is often regarded as a special type of segmentation task. Different from classic segmentation, the input of the interactive segmentation network contains not only the image but also the click maps and previously predicted mask. When segmenting the object of interest, the user decides where to place the next click based on the current segmentation mask. All positive/negative clicks are encoded as positive/negative click maps that are usually defined as distance maps [47] or binary disks [3]. The click maps are then concatenated with the image and previously predicted mask to be fed to the segmentation network. After each click, the user obtains an updated segmentation mask by running the whole segmentation network. Usually, a strong segmentation network like DeepLabv3+ [6] is required to achieve satisfactory results since the segmentation accuracy directly depends on it. However, running a strong segmentation network several times is often time-consuming, especially on low-power devices, which makes this standard pipeline inefficient in practice.

3.2. Efficient pipeline

To avoid running the whole segmentation network for each click, we build a lightweight network to correct the segmentation mask in each iteration. The pipeline is shown in Fig. 2. Our network consists of a base segmentation network and a lightweight mask correction network. As observed in FCA-Net [27], the first click is responsible for locating the target object. Thus, we input the first click into the segmentation network to generate target-aware features. When the user places the first click, the base segmentation network extracts the target-aware feature and generates a coarse mask, and the mask correction network corrects the coarse mask. Starting from the second click, we no longer run the base segmentation network. Instead, we directly feed all clicks into the mask correction network to correct the previously predicted mask.

Specifically, we denote the input image as I and the click set at the t-th iteration as C^t. The base segmentation network takes the input image and the first click C^1 as input and generates target-aware features F_t and a coarse mask M_c, which can be formulated as follows,

$$F_t, M_c = \Phi_S(I, Enc(C^1); \theta_s), \tag{1}$$

where Enc denotes the click encoding function, and Φ_S and θ_s denote the base segmentation network and its parameters, respectively. In this work, we adopt disks with a small radius [3, 40] to encode the clicks. The mask correction network takes the click set C^t, the target-aware feature F_t, the original image I, and the previously predicted mask M^{t-1} as input and outputs an updated mask, which can be written as follows,

$$M^t = \Phi_R(I, Enc(C^t), M^{t-1}, F_t; \theta_r), \tag{2}$$

where Φ_R and θ_r denote the mask correction network and its parameters, respectively. For the first click (*i.e.* $t = 1$), the previous mask M^0 input to the mask correction network is equal to the coarse mask M_c. The mask correction network can also take as input a preexisting mask generated from other forms of pre-processing.

Note that our pipeline differs from FCA-Net [27] which also treats the first click differently from other clicks. Our network only runs the mask correction network after the first click, while FCA-Net still needs to run the whole network for each click.

3.3. Efficient and effective mask correction network

A simple mask correction network with limited parameters and computational cost can hardly achieve comparable performance with a traditional segmentation network. Therefore, we propose to exploit the click information better to boost performance with the click-guided self-attention module and the click-guided correlation module. Before performing correlation and self-attention, we first concatenate the image, positive/negative click maps, and previous mask to extract the low-level features F_l from them. We also adjust the channel number of target-aware features F_t via a convolutional layer with a kernel size of 1×1. Then we concatenate F_l with F_t from the backbone to learn the fused features F_u.

To exploit the click information, we map the location of clicks onto the feature maps and extract the features with a shape of $1 \times 1 \times C$ for each click, where C is the number of channels. These click features are regarded as templates. The click-guided self-attention module propagates the information of these templates to other locations via multi-head attention, while the click-guided correlation attention module uses them to learn target outlines via correlation. Since the number of clicks used to obtain a satisfactory segmentation accuracy is often less than 20, it may not be sufficient only to extract templates from the location of clicks. Thus, we first enrich the templates by selecting some informative locations based on their semantic similarity to the clicks.

Template selection. The strategy to select templates is based on the observation that pixels on the same object show higher similarity than pixels from different objects. We separately select positive and negative templates by identifying pixels that are more likely to be the object and the background, respectively. To select positive templates, we assign a score for each pixel to indicate the likelihood of being the object and select the top K_p pixels. The score is calculated by considering three factors: the similarity to positive clicks, the similarity to negative clicks, and the distance from the positive clicks. A pixel with a short distance and a high similarity to one of the positive clicks and a low similarity to one of the negative clicks is more likely to be located in the object region and vice versa. We use the Cosine similarity between features of two pixels to measure their similarity and the Euclidean distance to measure the distance. Let $H_f \times W_f$ denote the size of the feature map, where H_f and W_f are respectively the height and width, and we can obtain a positive score map $O_p \in \mathbb{R}^{H_f \times W_f}$ as follows,

$$O_p = (V_p + (1 - V_n)) * (1 - D_p), \tag{3}$$

where $V_p \in \mathbb{R}^{H_f \times W_f}$ and $V_n \in \mathbb{R}^{H_f \times W_f}$ respectively denote the positive and negative Cosine similarity maps, and $D_p \in \mathbb{R}^{H_f \times W_f}$ denotes the normalized distance map. Note that the similarity scores $V_p^{u,v}$/$V_n^{u,v}$ respectively represent the maximum similarity between the pixel (u, v) and positive/negative clicks, while $D_p^{u,v}$ represents the normalized minimum distance between (u, v) and positive clicks. Similarly, we can obtain a negative score map O_n by

$$O_n = (V_n + (1 - V_p)) * (1 - D_n), \tag{4}$$

where D_n denotes the normalized negative distance map, which measures the normalized minimum distance between each pixel and negative clicks. After obtaining the score maps, we select positive templates from the top K_p locations of the positive score map and K_n negative templates from the top K_n locations of the negative score map. The positive and negative template features are represented as $F_{pt} \in \mathbb{R}^{K_p \times C}$ and $F_{nt} \in \mathbb{R}^{K_n \times C}$, respectively.

Click-guided self-attention. Our first way to exploit the click information is to propagate their information to other pixels via a self-attention module. Multi-head attention is widely used in computer vision to capture long-range dependencies among different pixels [9, 19]. We adopt the standard *qkv* multi-head attention [41] to augment the representation of each pixel with the selected templates. Multi-head attention helps each pixel attend to the templates, which is beneficial to the identification of the target. To perform multi-head attention, we first reshape the feature F_u to a size of $H_f W_f \times C$, then we learn the self-attention-based augmented feature F_s by

$$F_s = F_u + \mathit{Attention}(Q, K, V), \tag{5}$$

$$Q = \phi_Q(F_u), K = \phi_K(F_{pnt}), V = \phi_V(F_{pnt}), \tag{6}$$

where ϕ_Q, ϕ_K, and ϕ_V are linear transformations, and F_{pnt} is the concatenation of positive templates F_{pt} and negative templates F_{nt}. F_s is further reshaped to $H_f \times W_f \times C$. As the template features are selected from F_u, and the attention is guided by the click information, we denote this attention module as a click-guided self-attention module.

Click-guided correlation. To further exploit the click information, we propose a click-guided correlation attention module to learn the contour of the object of interest explicitly. Correlation is widely used in the Siamese object tracking archietecture [4] to locate the position of the target in a search area. Siamese tracking calculates similarity maps between a target template and the search area. In video object segmentation, RANet [44] treats target features from the first frame as the template to learn pixel-wise similarity maps. In RANet, pixel-wise correlation is shown to be able to learn the contour of the target with the target features as templates. In this work, we extract target semantic features based on the location of the clicks and employ the pixel-wise correlation to learn target outlines to improve the segmentation performance. For the j-th positive template feature $F_{pt}^j \in \mathbb{R}^{1 \times C}$, we can get a similarity map via the correlation between it and $F_u \in \mathbb{R}^{H_f W_f \times C}$. Thus, we have

$$S_p = \{S_p^j | S_p^j = \phi_q(F_{pt}^j) * \phi_k(F_u)\}_{j \in \{1,\dots,K_p\}}, \tag{7}$$

$$S_n = \{S_n^j | S_n^j = \phi_q(F_{nt}^j) * \phi_k(F_u)\}_{j \in \{1,\dots,K_n\}}, \tag{8}$$

where $*$ denotes the correlation operation, S_p and S_n respectively denote the set of positive and negative correlation similarity maps, and ϕ_q and ϕ_k denote two linear transformations. We then transform the positive/negative similarity maps into features for segmentation via a shared convolutional layer. Then positive and negative features are concatenated and followed by the second convolutional layer to fuse them. In this manner, we get the correlation-based augmented feature F_c as follows,

$$F_c = \phi_f(\mathit{Concat}(\phi_s(S_p), \phi_s(S_n))), \tag{9}$$

where ϕ_s and ϕ_f denote two convolutional layers.

After learning the two augmented features F_s and F_c, we concatenate them and use four convolutional layers to predict the segmentation mask. Our mask correction network is efficient since the resolution of the features is just 1/4 of the image, and the whole structure is lightweight. It is also effective since the click information is exploited explicitly via two modules to improve performance.

4. Experiments

4.1. Experimental Configuration

Implementation details. Our pipeline can be built on different existing segmentation networks. Following FocalClick [8], we choose HRNet+OCR [42, 49] and SegFormer [45] series as our base segmentation networks to show the effectiveness of the proposed method. For lightweight models like SegFormerB0 and HRNet18s, the mask correction network first transforms the number of backbone feature channels to 64. For larger models like SegFormerB3, HRNet18, and HRNet32, the channel is transformed to 96. The number of selected templates K_p and K_n is set to be equal to the channel number.

We train our model on a combination of COCO [25] and LVIS [14] datasets, following RITM [40]. For a fair comparison of some previous methods, we also report the performance of our model trained on SBD [16]. We use the normalized focal loss [38] for both the base segmentation

Method	Params/MB	FLOPs/G	Speed/ms
CDNet-ResNet34$_{384}$ [7]	23.5	56.7	3339
f-BRS-ResNet50$_{400}$ [39]	31.4	84.6	2373
RITM-hrnet18s$_{400}$ [40]	4.22	8.96	634
RITM-hrnet18$_{400}$ [40]	10.0	15.4	1103
RITM-hrnet32$_{400}$ [40]	31.0	40.4	1635
FocalClick-hrnet18s$_{256}$ [8]	4.23	3.82	358
FocalClick-hrnet32$_{256}$ [8]	31.0	17.1	728
FocalClick-SegB0$_{256}$ [8]	3.74	1.94	207
FocalClick-SegB3$_{256}$ [8]	45.6	12.9	805
Ours-hrnet18s-FirstClick$_{384}$	4.33	9.35	752
Ours-hrnet18-FirstClick$_{384}$	10.3	15.3	1237
Ours-hrnet32-FirstClick$_{384}$	31.2	40.5	1745
Ours-SegB0-FirstClick$_{384}$	3.84	5.38	528
Ours-SegB3-FirstClick$_{384}$	45.9	32.3	2006
Ours-MaskCorrection-C64$_{384}$	0.11	1.09	183
Ours-MaskCorrection-C96$_{384}$	0.22	2.25	280

Table 1. Comparison of the computational cost for different methods. The speed represents the average inference time per click and is measured on a CPU laptop with 2 GHz, 4×Intel Core i5. From the second click, we only run the mask correction network, significantly reducing the inference time compared to other methods.

	NoC@90	Time@90	NoC@95	Time@95
FocalClick-hrnet18s	6.79	34min	12.78	62min
FocalClick-hrnet32	6.51	49min	12.50	85min
FocalClick-SegB0	6.86	23min	12.73	39min
FocalClick-SegB3	5.59	34min	11.55	63min
Ours-hrnet18s	6.16	19min	12.47	30min
Ours-hrnet32	5.65	21min	11.90	32min
Ours-SegB0	6.21	16min	12.45	27min
Ours-SegB3	5.57	20min	11.65	29min

Table 2. Comparison of the total inference time to reach different IoU thresholds on the largest SBD dataset. The time is measured on an NVIDIA V100 GPU.

and mask correction networks. We adopt the iterative training strategy [31] by randomly sampling the first positive click in the target area and progressively adding clicks based on the errors in the network's predictions during training. Our lightweight mask correction network enables efficient iterative training. We use the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to train the network. As in FocalClick [8], we sample 30000 images in one epoch and train our model for 230 epochs. We use an initial learning rate of $1e-3$ for the HRNet+OCR series and $2e-3$ for the SegFormer series, and decay them by ten times on the 200th and 220th epochs. We train the network with an image size of 320×480 and a batch size of 64. We add data augmentation strategies, including random resizing in a range of [0.75, 1.40], horizontal flip, random jittering of brightness, contrast, and RGB values. All our models are implemented in Python using Pytorch.

Evaluation datasets and metrics. We evaluate our method on five commonly used datasets, including GrabCut [37], Berkeley [35], DAVIS [36], SBD [16], and Pascal VOC [10]. The GrabCut dataset contains 50 images with one single target in each image. The test set of the Berkeley dataset contains 100 object masks for 96 images. The DAVIS dataset is originally constructed for the evaluation of video object segmentation methods. We sample 345 frames to evaluate our method as in [8, 40]. The SBD dataset is the largest and has been widely used to evaluate the interactive segmentation methods since [47]. It contains 6671 object masks for 2820 images. The validation set of the Pascal VOC dataset is also used, which consists of 1449 images with 3427 object masks.

We use the commonly used evaluation protocol to make a fair comparison. During the evaluation, the clicks are simulated based on the difference between the current prediction and the ground truth. The first click is placed in the center of the target, and other clicks are placed in the center of the largest erroneous region. Following [26], the image is resized to a fixed length of 384 as the short side during evaluation. We use the standard Number of Clicks (NoC) metric to compare different methods. The NoC measures the number of clicks required to reach a predefined Intersection over Union (IoU) threshold between the predicted mask and the ground truth. Following most methods, the thresholds of 85% and 90% are used, and the metrics are denoted as NoC@85 and NoC@90.

4.2. Comparison with state-of-the-art methods

Computational comparison. We compare the parameters, FLOPs, and inference speed on CPUs of representative methods in Tab. 1. *MaskCorrection-C64* denotes that we set the channel of the mask correction network to 64, which is used for HRNet18s and SegFormerB0. *MaskCorrection-C96* is used for HRNet18, HRNet32, and SegFormerB3. The computational cost of our method is different between the first click and other clicks. For the first click, our method needs to run the base segmentation network and the mask correction network, resulting in slightly higher inference time compared with RITM [40]. However, our method shows higher efficiency from the second click even compared with FocalClick [8] which uses a small resolution since our mask correction network is lightweight. The computational cost of the mask correction network will not change with the base segmentation network if we fix the configuration, which is a good property since we can use a stronger segmentation network to reduce the number of interactions without a significant increase in the inference

		GrabCut		Berkeley	SBD		DAVIS		Pascal
Methods	Train Data	NoC@85	NoC@90	NoC@90	NoC@85	NoC@90	NoC@85	NoC@90	NoC@85
f-BRS-B-ResNet50 [39]	SBD	2.50	2.98	4.34	5.06	8.08	5.39	7.81	
CDNet-ResNet50 [7]	SBD	2.22	2.64	3.69	4.37	7.87	5.17	6.66	-
FocusCut-ResNet50 [26]	SBD	1.60	1.78	3.44	3.62	5.66	5.00	6.38	-
FocalClick-hrnet18s [8]	SBD	1.86	2.06	3.14	4.30	6.52	4.92	6.48	-
RITM-hrnet18 [40]	SBD	1.76	2.04	3.22	3.39	5.43	4.94	6.71	-
Ours-hrnet18s	SBD	1.82	1.92	3.26	3.58	5.79	5.23	6.88	2.47
Ours-hrnet18	SBD	1.74	1.84	3.03	3.38	5.51	5.05	6.71	2.37
TransClick-segformerB4 [11]	C+L	1.52	1.60	1.60	3.44	5.63	3.68	5.06	2.08
FocalClick-segformerB0 [8]	C+L	**1.40**	1.66	**2.27**	4.56	6.86	**4.04**	**5.49**	2.97
Ours-segformerB0	C+L	1.56	**1.64**	2.40	**3.95**	**6.21**	4.48	5.53	**2.65**
FocalClick-segformerB3 [8]	C+L	1.44	1.50	**1.92**	3.53	5.59	**3.61**	**4.90**	2.46
Ours-segformerB3	C+L	**1.42**	**1.48**	2.35	**3.44**	**5.57**	4.49	5.69	**2.23**
RITM-hrnet18s [40]	C+L	1.54	1.68	**2.60**	4.04	6.48	4.70	5.98	2.57
FocalClick-hrnet18s [8]	C+L	1.48	1.62	2.66	4.43	6.79	**3.90**	**5.25**	2.93
Ours-hrnet18s	C+L	**1.40**	**1.52**	2.68	**3.86**	**6.16**	4.42	5.66	**2.37**
RITM-hrnet18 [40]	C+L	1.42	1.54	**2.26**	3.80	6.06	4.36	5.74	**2.28**
EdgeFlow-hrnet18 [15]	C+L	1.60	1.72	2.40	-	-	4.54	5.77	2.50
Ours-hrnet18	C+L	**1.38**	**1.50**	2.30	**3.69**	**5.93**	**4.34**	**5.59**	2.37
RITM-hrnet32 [40]	C+L	1.46	1.56	2.10	3.59	5.71	4.11	5.34	2.57
FocalClick-hrnet32 [8]	C+L	1.64	1.80	2.36	4.24	6.51	**4.01**	5.39	2.80
PseudoClick-hrnet32 [29]	C+L	-	1.50	**2.08**	-	5.68	4.09	**5.27**	**1.94**
Ours-hrnet32	C+L	**1.30**	**1.42**	2.35	**3.55**	**5.65**	4.29	5.33	2.22

Table 3. Comparison results on five benchmarks. NoC@85 and NoC@90 respectively denote the number of clicks required to reach the IoU of 85% and 90%, and lower is better for the value. C+L means the combination of COCO and LVIS.

time. We demonstrate the total inference time to reach 90% and 95% IoU on the largest dataset SBD in Tab. 2. To save evaluation time, all models are evaluated on an NVIDIA V100 GPU. Our method spends less time reaching a predefined IoU threshold for all versions. Notably, although the number of clicks required to reach 90% (NoC@90) and 95% (NoC@95) IoUs is similar for FocalClick-SegB3 and Ours-SegB3, the inference time of our method is significantly lower because the computational cost of Ours-SegB3 from the second click is considerably lower than FocalClick-SegB3. And the gap between our method and FocalClick is larger when the threshold changes from 90% to 95%. Thus, our method is more suitable for challenging images and scenarios where we want to get a high IoU.

Performance Comparison. We compare our method with existing methods on five benchmarks in Tab. 3. Some methods use the SBD [16] dataset to train their network, while RITM suggests training on COCO [25] and LVIS [14]. We report the results of our method equipped with different base segmentation networks and trained on different datasets. Compared with some early methods like CDNet [7] and f-BRS [39], different versions of our method show superior performance and efficiency. Compared with recent state-of-the-art methods RITM [40], FocalClick [8], FocusCut [26], PseudoClick [29], and TransClick [11], our method demonstrates competitive performance. Especially, our method achieves outstanding performance on the largest SBD dataset. Overall, the proposed method demonstrates preferable performance considering that the inference time is significantly low from the second click.

4.3. Method analysis

Ablation study. We conduct ablation studies to verify the effectiveness of each component in our method. The results on three challenging datasets in presented in Tab. 4. We first construct a network with only a simple convolutional mask correction network as our baseline. The first point is not fed to the base segmentation network (HRNet18s) in this baseline. As shown in Tab. 4, a simple mask correction network can hardly achieve high performance. The input of the first click to the base segmentation network helps

1st Click	Corr	Self-Att	TS	DAVIS NoC@90	SBD NoC@90	Pascal NoC@90
				7.11	8.32	4.38
✓				6.98	6.83	3.19
✓	✓			6.32	6.30	3.09
✓		✓		6.02	6.33	2.98
✓	✓	✓		5.86	6.24	2.89
✓	✓	✓	✓	5.66	6.16	2.84

Table 4. Ablation study on three challenging benchmarks. HRNet18s is used as the base segmentation network. *1st Click* denotes that we input the first click to the base segmentation network, *Corr* denotes the correlation module, *Self-Att* denotes the self-attention module, and *TS* denotes the template selection module.

	Negative Templates	DAVIS NoC@90	SBD NoC@90	Pascal NoC@90
Correlation Module		6.10	6.39	3.01
	✓	6.09	6.26	2.96
Self-Attention Module		6.04	6.55	3.22
	✓	5.92	6.29	2.99

Table 5. The performance of the correlation and self-attention modules with and without the adoption of the negative templates.

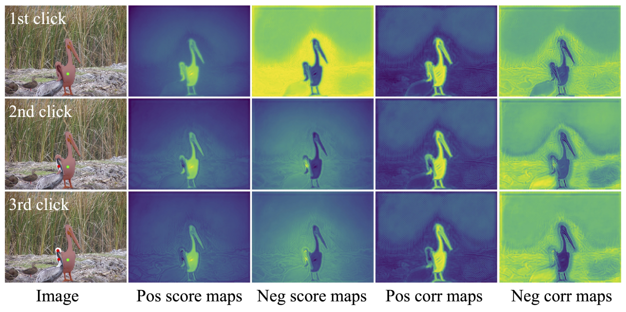

Figure 3. Illustration of the positive/negative score maps for template selection and positive/negative correlation similarity maps in click-guided correlation module.

extract target-aware features to boost the performance. However, the performance is still relatively low. Both the introduction of the click-guided self-attention module and the click-guided correlation module bring significant improvements over the simple mask correction network, indicating that they both effectively exploit the click information. Their combination achieves the best performance, showing that they learn complementary augmented features and can be combined to boost performance. The template selection module further improves performance by sampling several templates to enrich the click features.

The necessity of negative templates. All positive and negative templates are input to the self-attention and correlation modules. The adoption of positive templates is a natural choice since they contain rich information regarding the target. However, the negative templates are also vital since negative clicks are often selected in distractor areas, and they contain information that can discriminate the target from the background. Tab. 5 demonstrates the performance of the two modules with and without the negative templates. The employment of the negative templates effectively improves the performance of both the correlation and self-attention modules.

Qualitative analysis. We show the change of the positive/negative score maps for template selection and positive/negative correlation similarity maps in Fig. 3. After placing the first positive click, the model predicts an erroneous area that shows high semantic similarity to the target. After placing two more negative clicks in the erroneous area, the model successfully segments the target. In the interactive process, the score maps change with the clicks and can effectively discriminate the areas of the target of interest and other objects, which helps select meaningful and correct templates. The change in the correlation similarity maps shows that we can learn the target contours by correlating the templates with the feature maps. Fig. 3 only shows the similarity maps of two templates. Since we select several templates, the correlation results provide rich information to discriminate the target from other areas. Directly employing the correlation results as features helps better segment the target.

Limitation analysis. Our method still requires a relatively high computational cost in the first click, which may not be suitable for some simple annotation scenarios where only one or two clicks are needed. Maybe this can be solved by reducing the resolution of the base segmentation network as in FocalClick [8] and locally updating the mask with the mask correlation network. We leave this for future research.

5. Conclusion

In this paper, we propose an efficient EMC-Click method for click-based interactive segmentation. A lightweight mask correction network is constructed to iteratively update the mask, which significantly reduces the computational cost from the second click. A click-guided correlation module and a click-guided self-attention module are proposed to effectively exploit the click information to boost the performance of the mask correction network. Experiments on five benchmarks demonstrate that our method achieves competitive performance and higher efficiency compared with state-of-the-art methods.

References

[1] Eirikur Agustsson, Jasper RR Uijlings, and Vittorio Ferrari. Interactive full image segmentation by considering all regions jointly. In *CVPR*, pages 11622–11631, 2019. 1

[2] Junjie Bai and Xiaodong Wu. Error-tolerant scribbles based interactive image segmentation. In *CVPR*, pages 392–399, 2014. 1

[3] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. In *CVPR*, pages 11700–11709, 2019. 4

[4] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV Workshops*, volume 9914, pages 850–865, 2016. 5

[5] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *ICCV*, volume 1, pages 105–112, 2001. 2

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, volume 11211, pages 833–851, 2018. 4

[7] Xi Chen, Zhiyan Zhao, Feiwu Yu, Yilei Zhang, and Manni Duan. Conditional diffusion for interactive segmentation. In *ICCV*, pages 7345–7354, 2021. 2, 3, 6, 7

[8] Xi Chen, Zhiyan Zhao, Yilei Zhang, Manni Duan, Donglian Qi, and Hengshuang Zhao. Focalclick: Towards practical interactive image segmentation. In *CVPR*, pages 1300–1309, 2022. 2, 3, 5, 6, 7, 8

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 5

[10] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010. 6

[11] Boris Faizov, Vlad Shakhuro, and Anton Konushin. Interactive image segmentation with transformers. In *ICIP*, pages 1171–1175, 2022. 7

[12] Marco Forte, Brian Price, Scott Cohen, Ning Xu, and François Pitié. Getting to 99% accuracy in interactive segmentation. *arXiv:2003.07932*, 2020. 3

[13] Leo Grady. Random walks for image segmentation. *IEEE TPAMI*, 28(11):1768–1783, 2006. 2

[14] Agrim Gupta, Piotr Dollár, and Ross B. Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *CVPR*, pages 5356–5364, 2019. 5, 7

[15] Yuying Hao, Yi Liu, Zewu Wu, Lin Han, Yizhou Chen, Guowei Chen, Lutao Chu, Shiyu Tang, Zhiliang Yu, Zeyu Chen, et al. Edgeflow: Achieving practical interactive segmentation with edge-guided flow. In *ICCV workshops*, pages 1551–1560, 2021. 2, 3, 7

[16] Bharath Hariharan, Pablo Arbelaez, Lubomir D. Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, pages 991–998, 2011. 5, 6, 7

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, pages 2980–2988, 2017. 1

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2

[19] Khawar Islam. Recent advances in vision transformer: A survey and outlook of recent work. *arXiv:2203.01536*, 2022. 5

[20] Won-Dong Jang and Chang-Su Kim. Interactive image segmentation via backpropagating refinement scheme. In *CVPR*, pages 5297–5306, 2019. 1, 2

[21] Tae Hoon Kim, Kyoung Mu Lee, and Sang Uk Lee. Generative image segmentation using random walks with restart. In *ECCV*, pages 264–275, 2008. 2

[22] Theodora Kontogianni, Michael Gygli, Jasper R. R. Uijlings, and Vittorio Ferrari. Continuous adaptation for interactive object segmentation by learning from corrections. In *ECCV*, volume 12361, pages 579–596, 2020. 1

[23] Victor S. Lempitsky, Pushmeet Kohli, Carsten Rother, and Toby Sharp. Image segmentation with a bounding box prior. In *ICCV*, pages 277–284, 2009. 1

[24] JunHao Liew, Yunchao Wei, Wei Xiong, Sim-Heng Ong, and Jiashi Feng. Regional interactive image segmentation networks. In *ICCV*, pages 2746–2754, 2017. 2

[25] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, volume 8693, pages 740–755, 2014. 5, 7

[26] Zheng Lin, Zheng-Peng Duan, Zhao Zhang, Chun-Le Guo, and Ming-Ming Cheng. Focuscut: Diving into a focus view in interactive segmentation. In *CVPR*, pages 2637–2646, 2022. 3, 6, 7

[27] Zheng Lin, Zhao Zhang, Lin-Zhuo Chen, Ming-Ming Cheng, and Shao-Ping Lu. Interactive image segmentation with first click attention. In *CVPR*, pages 13339–13348, 2020. 2, 4

[28] Qin Liu, Zhenlin Xu, Yining Jiao, and Marc Niethammer. isegformer: Interactive segmentation via transformers with application to 3d knee MR images. In *MICCAI*, volume 13435, pages 464–474, 2022. 1

[29] Qin Liu, Meng Zheng, Benjamin Planche, Srikrishna Karanam, Terrence Chen, Marc Niethammer, and Ziyan Wu. Pseudoclick: Interactive image segmentation with click imitation. In *ECCV*, 2022. 7

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 1

[31] Sabarinath Mahadevan, Paul Voigtlaender, and Bastian Leibe. Iteratively trained interactive segmentation. In *BMVC*, 2018. 2, 6

[32] Soumajit Majumder, Abhinav Rai, Ansh Khurana, and Angela Yao. Two-in-one refinement for interactive segmentation. In *BMVC*, 2020. 1, 2

[33] Soumajit Majumder and Angela Yao. Content-aware multi-level guidance for interactive instance segmentation. In *CVPR*, pages 11602–11611, 2019. 1

[34] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, pages 616–625, 2018. 1, 2

[35] Kevin McGuinness and Noel E. O'Connor. A comparative evaluation of interactive segmentation algorithms. *PR*, 43(2):434–444, 2010. 6

[36] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus H. Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, pages 724–732, 2016. 6

[37] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM TOG*, 23(3):309–314, 2004. 1, 2, 6

[38] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin. Adaptis: Adaptive instance selection network. In *ICCV*, pages 7355–7363, 2019. 5

[39] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *CVPR*, pages 8623–8632, 2020. 1, 2, 6, 7

[40] Konstantin Sofiiuk, Ilya A Petrov, and Anton Konushin. Reviving iterative training with mask guidance for interactive segmentation. In *ICIP*, pages 3141–3145, 2022. 1, 2, 4, 5, 6, 7

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017. 5

[42] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *IEEE TPAMI*, 43(10):3349–3364, 2021. 5

[43] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, pages 7794–7803, 2018. 3

[44] Ziqin Wang, Jun Xu, Li Liu, Fan Zhu, and Ling Shao. Ranet: Ranking attention network for fast video object segmentation. In *ICCV*, pages 3978–3987, 2019. 5

[45] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, pages 12077–12090, 2021. 5

[46] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas Huang. Deep grabcut for object selection. In *BMVC*, 2017. 2

[47] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. Deep interactive object selection. In *CVPR*, pages 373–381, 2016. 1, 2, 4, 6

[48] Hongkai Yu, Youjie Zhou, Hui Qian, Min Xian, and Song Wang. Loosecut: Interactive image segmentation with loosely bounded boxes. In *ICIP*, pages 3335–3339. IEEE, 2017. 1

[49] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *ECCV*, volume 12351, pages 173–190, 2020. 5

[50] Chuyu Zhang, Chuanyang Hu, Yongfei Liu, and Xuming He. Intention-aware feature propagation network for interactive segmentation. *arXiv:2203.05145*, 2022. 3