DynaMITe: Dynamic Query Bootstrapping for Multi-object Interactive Segmentation Transformer

Amit Kumar Rana Sabarinath Mahadevan Alexander Hermans
Bastian Leibe

RWTH Aachen University, Germany
firstname.lastname@rwth-aachen.de
https://sabarim.github.io/dynamite/

Abstract

Most state-of-the-art instance segmentation methods rely on large amounts of pixel-precise ground-truth annotations for training, which are expensive to create. Interactive segmentation networks help generate such annotations based on an image and the corresponding user interactions such as clicks. Existing methods for this task can only process a single instance at a time and each user interaction requires a full forward pass through the entire deep network. We introduce a more efficient approach, called DynaMITe, in which we represent user interactions as spatio-temporal queries to a Transformer decoder with a potential to segment multiple object instances in a single iteration. Our architecture also alleviates any need to re-compute image features during refinement, and requires fewer interactions for segmenting multiple instances in a single image when compared to other methods. DynaMITe achieves state-of-the-art results on multiple existing interactive segmentation benchmarks, and also on the new multi-instance benchmark that we propose in this paper.

1. Introduction

Interactive segmentation algorithms enable a user to annotate the objects of interest within a given image with the help of user interactions such as scribbles and clicks. Such algorithms have several advantages compared to fully-automatic segmentation methods, since they enable a user to select and iteratively refine the objects of interest. Existing interactive segmentation methods [8, 28, 31, 41, 42] formulate this task as a binary instance segmentation problem, where the single object of interest can be segmented and corrected using user clicks.

Most of these approaches use deep neural networks to generate the image features that are conditioned on the user clicks and previous predictions, and they require the image level features to be re-computed for every user interaction. While such a design has been proven to be effective, the runtime for processing each interaction is proportional to the size of the feature extractor used, since a forward pass through the network is needed per interaction [8,28,31,41, 42]. Hence, these methods often have to limit their network sizes in order to achieve a good runtime performance and are thus not scalable in this respect.

In addition, the design decision to model interactive segmentation as a binary segmentation problem forces existing methods to approach multi-instance segmentation tasks as a sequence of single-instance problems, operating on separate (sometimes cropped and rescaled [8]) image regions. Consequently, such methods need additional clicks if there are multiple similar foreground instances in an image, since each of those instances has to be processed separately with a disjoint set of user interactions, specifying the foreground and background. This is inefficient, since it is often the case that one object instance has to be considered as background for a different nearby instance, such that a refinement with a negative click becomes necessary for the current object of focus.

In this work, we improve on both of the above issues by proposing a Dynamic Multi-object Interactive segmentation Transformer (DynaMITe), a novel multi-instance approach for interactive segmentation that only requires a single forward pass through the feature extractor and that processes all relevant objects together, while learning a common background representation. Our approach is based on a novel Transformer-based iterative refinement architecture which determines instance level descriptors directly from the spatio-temporal click sequence. DynaMITe dynamically generates queries to the Transformer that are conditioned on the backbone features at the click locations. These queries are updated during the refinement process whenever the network receives a new click, prompting the Transformer to output a new multi-instance segmentation. Thus, DynaMITe removes the need to re-compute image-level features for each user interaction, while making more effective use of user clicks by handling multiple object in-

Figure 1: **DynaMITe** processes multiple instances at once and models the background jointly. In this example, the false positive region on the camel in the second image is corrected automatically when the user chooses to segment it as foreground. DynaMITe is also able to correctly segment tiny structures, such as the camel's leash in the final segmentation mask.

stances together.

The attention-based formulation of learning object representations from user interactions allows multiple objects to interact with each other and with the common background representations, thus enabling the network to estimate a better context from the input image. Fig. 1 shows a typical example, highlighting DynaMITe's capability to segment all the relevant objects in the input image using few clicks. An advantage of such a network formulation can be directly seen in the third refinement iteration, where a positive click on the unsegmented camel instance automatically removes the false positive region that was spilled over after segmenting a different camel instance nearby in the previous iteration. Existing approaches that perform sequential single-instance segmentation would have to first add a negative click to remove the false positive as part of the individual object refinement in the third iteration, thereby requiring additional annotation effort.

In order to enable quantitative evaluation, we also propose a novel multi-instance interactive segmentation task (MIST) and a corresponding evaluation strategy. Compared to single-instance segmentation, MIST has the added complexity of requiring decisions which object to click on next, which is significantly harder than just deciding where to click next in a given single-instance error region. In particular, different users may apply different next-object selection strategies, and it is important that an interactive segmentation method is robust to this and always performs well. Hence, we propose to evaluate against a set of several different (but still basic) click sampling heuristics that are intended to span the expected variability of user types.

In summary, we propose DynaMITe, a novel Transformer-based interactive segmentation method which uses a query bootstrapping mechanism to learn object representations from image-level features that are conditioned on the user interactions. We also model the iterative refinement process as temporal update steps for the queries to our Transformer module, which removes the need to re-compute image-level features. We evaluate DynaMITe on the standard interactive segmentation benchmarks and show that it performs competitively in the single-instance setting, while outperforming existing state-of-the-art methods on multi-instance tasks.

2. Related Work

Instance Segmentation. Methods that perform instance segmentation automatically generate masks for every object in the input image. Mask R-CNN [19] is one of the most influential instance segmentation networks, which first generates object proposals and then segments these proposals using a mask head. Several other methods use a single-stage approach, either by grouping the pixels [11, 23, 34–36], or by employing dynamic networks [43] on top of fully convolutional object detectors [44]. After the success of Vision Transformers (ViT) [13] for image-level classification tasks, recent methods leverage Transformer-based architectures for performing instance segmentation. MaskFormer [10] adds a mask classification head to DETR [5] and models instance segmentation as a per-pixel classification task. Mask2Former [9] further extends MaskFormer by using a masked-attention Transformer decoder. Unlike interactive segmentation methods, instance segmentation networks rely on segmenting a fixed set of classes and cannot incorporate user inputs for refinement.

Interactive Segmentation. Earlier methods [4, 39, 48] that perform interactive segmentation used graph-based optimisation techniques to translate user inputs to per-pixel segmentation masks. With the advent of deep learning, recent methods [6, 8, 24–26, 28, 31, 41, 47] have been able to reduce the number of user interactions required for generating object masks. Most of these methods [8, 31, 42] use positive and negative clicks to iteratively segment a foreground object by concatenating the input image with the click maps, along with the previous mask predictions, and then sending this combined representation through a deep network. This enables the network to learn the underlying representation of objects based on the input clicks. iADAPT [24] extends ITIS [31] by considering user corrections as training examples during the testing phase, and updating the network parameters based on them, thereby aligning the training and testing domains. BRS [22] is another interactive segmenta-

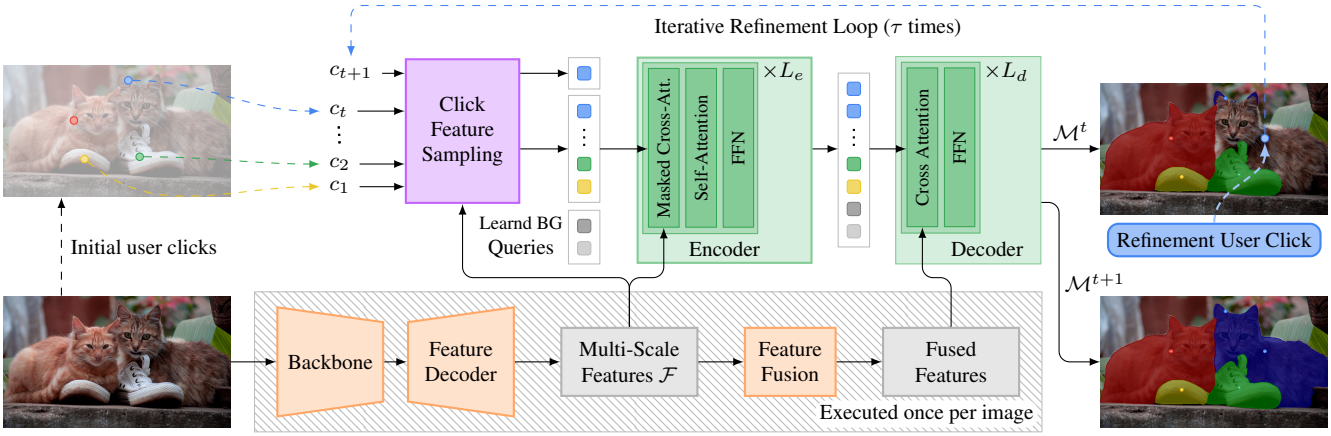


Figure 2: DynaMITe consists of a backbone, a feature decoder, and an interactive Transformer. Point features at click locations at time t are translated into queries which, along with the multi-scale features, are processed by a Transformer encoder-decoder structure to generate a set of output masks $\mathcal{M}^t$ for all the relevant objects. Based on $\mathcal{M}^t$, the user provides a new input click which is in turn used by the interactive Transformer to generate a new set of updated masks $\mathcal{M}^{t+1}$. This process is then iterated τ times until the masks are fully refined.

tion method that proposes an online update scheme for the network during testing, by constraining user click locations to have the corresponding click label. RITM [42] improves the iterative training procedure introduced in [31], and subsequently demonstrates better performance. The recent FocalClick [8] approach builds upon the RITM [42] pipeline, and uses a focus crop that is obtained based on user corrections during the refinement process. FocalClick achieves the state-of-the-art results on multiple instance segmentation datasets. Unlike DynaMITe, all of these methods are designed to work with single instances, and also need a complete forward pass for each refinement iteration.

Conceptually similar to our approach is that of Agustsson *et al*. [1], who focus on full image segmentation. This is also a form of interactive multi-instance segmentation; however, every pixel in the image has to be assigned to a segment. Their method is based on a two-stage Mask-RCNN, where the user specifies the object proposals with clicks on extreme object points, followed by scribbles for mask corrections. Our method is more general, allowing the user to click on any object pixel and only requiring a minimum of one instead of four clicks per object.

3. Method

In a typical interactive segmentation process, the model first receives an input image along with the associated foreground (positive) clicks representing the objects that the user intends to segment. Based on this set of inputs, an interactive segmentation model predicts an initial set of segmentation masks corresponding to the clicks. These initial predictions are presented to the user so that they can provide a corrective click (which can be positive or negative) that is used to refine the previous network predictions. This process is repeated until the user receives a set of non-overlapping segmentation masks of satisfactory quality.

Current state-of-the-art interactive segmentation models [8, 24, 31, 41, 42] perform this task sequentially, as their networks can handle only one foreground object at a time. These methods mostly use click maps, which are updated every time a user provides a new click and are then used to obtain a localized feature map from the feature extractor. DynaMITe, on the other hand, can process multiple objects at once, and translates clicks to spatio-temporal data that is processed by an interactive transformer. This makes our model more efficient for mainly three reasons: (i) DynaMITe just needs a single forward pass through the feature extractor to segment all the relevant foreground instances; (ii) the background is modeled jointly, and hence it reduces redundancy in negative clicks; and (iii) by annotating multiple objects jointly, these do not need to be repeatedly modeled as background for other foreground objects.

3.1. Network Architecture

Following the state-of-the-art Transformer-based segmentation methods [2, 9, 10, 49], we use three basic components in our architecture: (i) a backbone network, (ii) a feature decoder, and (iii) a Transformer structure that processes the multi-scale image features from the feature decoder (Fig. 2). Additionally, we also include a feature fusion module that fuses the multi-scale features to generate a fused feature map at the largest scale. Our main contribution lies in the Transformer structure, which learns localized object descriptors directly from the user interactions without the need to pass them through the entire feature extractor. This results in an interactive segmentation network that

is not only efficient in processing the user interactions, but also more practical since it can process multiple object instances at once. Since DynaMITe can encode relationships between multiple objects in a given scene, it is naturally capable of segmenting multiple instances at once, which is a paradigm shift for interactive segmentation networks.

Fig. 2 shows the overall architecture of our network. It takes as input an RGB image $\mathcal{I} \in \mathbb{R}^{H \times W \times 3}$, and the corresponding set of user interactions $\mathcal{S}^t = \{c_1, c_2, ..., c_t\}$ at timestep $t \in \{1, \ldots, \mathcal{T}\}$, where $|\mathcal{T}|$ is the maximum number of refinement iterations for $\mathcal{I}$. Following the classic formulation of interactive segmentation that is used by existing works, we model the user interactions as positive and negative clicks, where positive clicks are placed on the foreground objects and the negative clicks on the background. Hence $\mathcal{S}^t = \{\mathcal{S}^t_+, \mathcal{S}^t_-\}$, where $\mathcal{S}^t_+ = \{P^t_1, P^t_2, ..., P^t_n\}$ denote the positive clicks, and $\mathcal{S}^t_- = \{b^t_1, b^t_2, ..., b^t_m\}$ denote the set of negative clicks at time t. Here, $P^t_i \in S^t_+$ is a set of positive clicks that belong to object $o_i \in \mathcal{O}$. For existing interactive segmentation methods, $|\mathcal{O}|$ is always 1 since they can process only one object at a time, which need not necessarily be the case for DynaMITe as it is capable of handling multiple objects concurrently. All $P^t_i \in S_+$, as well as S_- are initialised as empty sets, and then updated when the user inputs a new click. The backbone processes $\mathcal{I}$ and extracts low-level features, which are then up-sampled by the feature decoder to produce feature maps $\mathcal{F} = \{f_{32}, f_{16}, f_8\}$ at multiple scales. These feature maps, along with the associated user interactions, up to time t, are then processed by the interactive Transformer.

3.2. Interactive Transformer

The goal of our interactive Transformer is to generate segmentation masks $\mathcal{M}^t = \{M^t_1, M^t_2, ..., M^t_n\}$ for all the relevant foreground objects at a given refinement timestep t, given the inputs $\mathcal{F}$ and the corresponding clicks $\mathcal{S}^t$. These masks should be disjoint, *i.e.* $M^t_i \cap M^t_j = \emptyset$ for all $i \neq j$.

Dynamic Query Bootstrapping. The queries used by the Transformer are dynamically generated using the input features $\mathcal{F}$ and the user clicks $\mathcal{S}^t$. To do this, we first sample the point features at every spatial location represented by each user click in $\mathcal{S}^t$ from all the feature scales in $\mathcal{F}$. Hence, if Q^t denotes the set of queries at time t, then $q_j \in Q^t$ for click c_j in $\mathcal{S}^t$ is generated as:

$$q_j = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} f_{c_j}. \tag{1}$$

During the refinement process, the network receives a new interaction c_{t+1} at the time step $t+1$, which is used to obtain an updated set of user clicks $\mathcal{S}^{t+1}$. To do this, if c_{t+1} is a positive click then it is added to the corresponding object specific click set P_j to obtain a new set of foreground clicks $\mathcal{S}^{t+1}_+$, else it is added to $\mathcal{S}_-$ to obtain $\mathcal{S}^{t+1}_-$. $\mathcal{S}^{t+1}$ is then used to obtain the updated queries Q^{t+1}, using the same process as explained above. These queries are thus dynamically updated throughout the iterative process without the need to recompute $\mathcal{F}$, and the entire interactive segmentation process can work with multiple instances at once.

In addition to the dynamic queries, we include a set of $K = 9$ learnable queries for modeling the background without the use of any user guidance. These static background queries learn generic background representations and they reduce the background interactions that a user will have to perform. We also add a 3D positional encoding to q_j where the first two dimensions represent the spatial location of the corresponding click in the image features and the third dimension represents the refinement timestep t.

Instance Encoder. The DynaMITe instance encoder takes as input the queries Q^t and the multi-scale feature maps $\mathcal{F}$. The encoder follows the structure of the masked attention Transformer decoder presented in [9] and leverages its capability of processing multi-scale features, which is important for dense pixel prediction tasks. Its main purpose is to enhance the initial click-based queries, such that they become more discriminative. We use $L_e = 9$ layers, which are grouped into 3 blocks, each of which processes successive feature scales from $\mathcal{F}$. Every Transformer layer in the encoder consists of a masked attention module, a self-attention module and a feedforward network. Hence, our encoder block performs the following operations:

$$Q_l \leftarrow \mathrm{MaskedCrossAttn}(Q_l, F, \mathcal{M}_{l-1}) + Q_{l-1}, \tag{2}$$

$$Q_l \leftarrow \mathrm{SelfAttn}(Q_l) + Q_l, \tag{3}$$

$$Q_l \leftarrow \mathrm{FFN}(Q_l) + Q_l. \tag{4}$$

Here, Q_l represents the queries at l^{th} layer; $\mathcal{M}_{l-1}$ is the attention mask produced from the binarized mask predictions from $(l-1)^{\text{th}}$ layer; $\mathrm{SelfAttn}$ is the multi-head self-attention module introduced in [45]; and FFN denotes a feedforward network. $\mathrm{MaskedCrossAttn}$ is a variant of cross-attention, where the attention operation is restricted to the foreground region represented by each query in the previous layer. Hence, each masked attention module performs the following operation:

$$Q_l = \mathrm{softmax}(\mathcal{M}_{l-1} + Q_l K_l) V_l + Q_{l-1}, \tag{5}$$

where K_l and V_l are the keys and values derived from $\mathcal{F}$ at the corresponding feature scale.

Decoder. While the goal of the encoder is to update the instance specific queries Q, the decoder updates the fused features $\mathcal{F}^{\mathcal{M}}$. The fused features are obtained from the feature fusion module, which takes the multi-scale decoder features $\mathcal{F}$ as input and generates a fused feature map at the largest scale of $\mathcal{F}$. The feature fusion module consists of a convolutional layer, followed by an up-sampling layer with skip

connections to upsample low resolution features from $\mathcal{F}$, which are then concatenated to generate $\mathcal{F}^{\mathcal{M}}$.

The decoder processes $\mathcal{F}^{\mathcal{M}}$ using a set of $L_d = 5$ Transformer layers. Each Transformer layer in our decoder consists of a cross-attention layer, followed by a feedforward network. Hence, each of the DynaMITe decoder layers performs the following operations:

$$F_l^M = \text{softmax}(F_l^M K^T)V^T + F_{l-1}^M, \quad (6)$$

$$F_l^M = \text{FFN}(F_l^M) + F_l^M. \quad (7)$$

Here K and V are again keys and values; however, they are obtained from the instance encoder's output Q_{out} and not from $\mathcal{F}^{\mathcal{M}}$. To get the final output masks, we take a dot product of the updated mask features F_{out}^M with the click specific queries $\mathcal{Q}_{out}$, as done in [9], and obtain a set of output mask probabilities. Since we have more than one query representing both the objects and the background, we use the per-pixel $\max$ operation over the corresponding set of queries to obtain instance specific masks for all the objects and the background. In the end, the discretized output masks are obtained by taking an argmax per pixel across the different instance predictions.

4. Multi-instance Interactive Segmentation

Existing interactive segmentation approaches address multi-instance segmentation as a sequence of single-instance tasks. *I.e.*, they pick one instance at a time, and then refine it either until the mask has a satisfactory quality, or until they have exhausted a specific click budget τ for that object. If there are multiple foreground objects in a single image, these methods generate overlapping object masks which have to be merged as an additional post-processing step in order to obtain the final masks. Also, since these objects are processed individually with disjoint click sets, some clicks can be redundant at an image-level. Hence, in this work we propose a novel multi-instance interactive segmentation task (MIST), where the goal of a user is to jointly annotate multiple object instances in the same input image.

Given an input image and a common set of user clicks, the MIST expects a corresponding method to generate non-overlapping instance masks for all relevant foreground objects. A major difference in this setting is that the background, and the corresponding negative clicks, are now common for all object instances. The MIST is a more challenging problem compared to the classical single-instance setting, since every refinement step can now lead to a positive click on any of the relevant objects or to a negative (background) click. Thus, extending an existing single-instance interactive segmentation method to the MIST is not trivial.

Automatic Evaluation. It is also important to note that the user click patterns for the MIST may differ considerably between users. As a result, simulating the MIST for automatic evaluation is a challenge of its own. In contrast to single-instance interactive segmentation benchmarks that have converged onto a deterministic next-click simulation strategy [8, 24, 31, 41, 42], the refinement focus in the MIST may jump from one object to another in an arbitrary sequence, unless users are instructed to process the objects in an image according to a specific order. Since it is hard to predict what next-object/next-click selection strategies users will end up using in an actual interactive segmentation application, and since that choice will in turn depend on their impression of which strategies work best with the given segmentation method, it is not practical to assume a single, deterministic next-click simulation strategy. Instead, we postulate that a method that performs the MIST should ideally be robust against varying click patterns and next-object selection strategies. Hence, we propose a multi-fold evaluation based on three different next-object simulation patterns during refinement.

All of these click simulation strategies start by adding a single positive click to each of the foreground objects in that image to get an initial prediction. Based on this initial prediction, we choose an object o_i according to one of the following strategies: (i) *best*: choose the object that has the best IoU, compared to the ground truth mask; (ii) *worst*: choose the object that has the worst IoU; and (iii) *random*: choose a random object. In each of these strategies, only the objects that have not yet achieved the required segmentation quality will be sampled. Next, we place a simulated click c_t on the largest error region of o_i. c_t can now be (i) a positive click on o_i; (ii) a negative click on the background; or (iii) a positive click on another o_j. This process is repeated either until all the relevant objects are segmented, or until the image-level click budget τ is fully spent. We want to emphasize that we make no claim that those strategies (*best*, *worst*, *random*) are close to optimal (in fact, we discuss several more effective strategies in the supplementary material). Instead, we intend for them to span the variability of possible next-object selection strategies to ensure that evaluated approaches generalize well to different users.

Evaluation Metric. The standard metric used by existing interactive segmentation benchmarks [18, 32, 37, 40] is the average number of clicks per object (NoC). Since the MIST is quite different from annotating instances individually, the NoC metric for interactive segmentation per object would not serve as a good evaluation metric. Hence, we propose a new evaluation metric called Normalized Clicks per Image (NCI) for the multi-instance interactive segmentation task. NCI is an adaptation of NoC, where the number of clicks is now computed per image, instead of per-object, and is then normalized by the number of foreground objects in the image. For NCI, we cap the number of clicks for an image based on the number of foreground objects. If an image has

$|\mathcal{O}|$ foreground objects, then the total click budget for that image would be $\tau * |\mathcal{O}|$. Unlike the NoC metric, this cap is at an image level, and all of these clicks can be spent on a subset of objects if the corresponding algorithm so desires. Similar to the single-instance case, all objects that cannot be segmented to the desired quality level using this budget are marked as failure cases (counted as NFO), and the number of clicks for that image is set to the image-level click budget. In addition, we also mark an image as a failed image (counted as NFI) if there is at least one object within that image that could not be segmented.

5. Experiments

Datasets and Metrics. We evaluate DynaMITe on an extensive range of datasets across two task settings. For the well established single-instance setting, we mainly use small-scale datasets such as GrabCut [40], Berkeley [32], COCO MVal, and DAVIS [37]. GrabCut and Berkeley are very small datasets with 50 and 96 images, respectively, mostly containing a single foreground object. COCO MVal is a subset of COCO [27] with a total of 800 images, and contains 10 objects from each object category. DAVIS [37] is a video object segmentation dataset, which consists of 50 short videos for training and 20 for validation. Each video frame consists of a single salient foreground region, where object instances that belong together share a common mask. For evaluation, we use the subset of 345 randomly sampled images [22] to be consistent with the existing interactive segmentation methods. Additionally, we also evaluate this task on SBD [18], which is an extension of the PASCAL VOC [14] dataset with 10582 images containing 24125 object instances, with 6671 instances for validation. Although SBD contains multiple instances per image, it is adapted to the single-instance task setting by considering every image-instance pair as a separate data sample.

For evaluating the MIST, we use the large-scale instance segmentation dataset COCO [27] in conjunction with DAVIS17 [38], and SBD [18]. COCO is an image dataset with annotations for multiple image level tasks with 5k images for validation. DAVIS17 is an extension of the single-instance DAVIS [37] dataset, which contains 30 validation videos with multiple segmented objects per-frame. In addition, we also use the annotations from LVIS [16] for training DynaMITe, where LVIS consists of a subset of COCO images with additional high-quality segmentation masks.

Implementation Details. For most experiments, we use a Swin Transformer [21] as backbone, with a multi-scale deformable-attention Transformer [49] on top to extract multi-scale features at 1/8, 1/16 and 1/32 scales. The encoder for our interactive Transformer follows the structure of the Transformer decoder in [9]. Specifically, there are 3 Transformer blocks in the encoder, each with 3 layers to process the feature maps at subsequent scales. For the interactive Transformer decoder, we use 5 layers of the cross-attention blocks as defined in Sec. 3.2.

The backbone is initialized with ImageNet [12] pre-trained weights, while the feature decoder and the Transformer weights are initialized randomly. The entire network is trained end-to-end on the combined COCO+LVIS [42] dataset with an input resolution of 1024×1024 px for 50 epochs, and a batch size of 32 on 16 Nvidia A100 GPUs. We follow the iterative training strategy used in [42]: we run a maximum of 3 iterative refinement steps to generate corrective clicks, based on the network output, for each object in an image during training.

5.1. Comparison with the State-of-the-art

Single Instance Setting. Although our model is designed to perform multi-instance interactive segmentation, we also apply it to the standard single-instance benchmark without any adaptations or re-training. In Tab 1 we compare our results against previous methods, which are grouped based on the underlying network architecture and the used training data. For this setting, we follow the same evaluation setting and the click sampling strategy adopted in previous works [8, 31, 41, 42] and also set the click budget τ to 20.

Early deep learning models [3, 24, 31] used larger backbones such as DeepLabV3+ [7], and were trained with small-scale image datasets such as PascalVOC [14], while state-of-the-art interactive segmentation models mostly use HRNet [46]. To be consistent with these methods, we report the results for DynaMITe using different commonly used backbone networks. Methods with comparable architectures are grouped together, and the corresponding best results within each group are marked in red. Although none of the DynaMITe models were specifically trained to perform single-instance interactive segmentation, it outperforms comparable state-of-the-art networks for a majority of the datasets. Since vision transformers have recently emerged as a competitive alternative to CNNs, we additionally report DynaMITe results with a Swin transformer [29].

Multi-instance Interactive Segmentation (MIST). For this experiment, we follow the MIST evaluation strategy described in Sec. 4 and use the proposed metrics (NCI, NFO, and NFI). In addition, we also report the average image-level IoU achieved after segmenting all the objects in an image. We use the validation sets of COCO [27], SBD [18], and DAVIS17 [38] to evaluate our models and set $\tau = 10$.

As a baseline, we adapt FocalClick [8] to the MIST setting. FocalClick is designed to work with a single object instance at a time, and processes objects sequentially to generate overlapping binary masks for each instance in an image. Hence, it cannot be directly used for automatic evaluation on the MIST, since the MIST click sampling strategy requires multi-instance segmentation masks to choose the

Method	Backbone	Train Data	GrabCut [40]		Berkeley [32]		SBD [18]		COCO MVal		DAVIS [37]	
			@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓
iFCN w/ GraphCut	-	PASCAL VOC	-	6.04	-	8.65	-	-	-	-	-	-
ITIS [31]	DeepLabV3+	SBD	-	5.6	-	-	-	-	-	-	-	-
VOS-Wild [3]	ResNet-101	-	-	3.8	-	-	-	-	-	-	-	-
iADAPT [24]	DeepLabV3+	SBD	-	3.07	-	4.94	-	-	-	-	-	-
EdgeFlow [17]	hrnet18	COCO+LVIS	1.60	1.72	-	2.40	-	-	-	-	4.54	5.77
RITM [42]	hrnet32	COCO+LVIS	1.46	1.56	-	2.10	3.59	5.71	-	-	4.11	5.34
FocalClick [8]	hrnet32	COCO+LVIS	1.64	1.80	-	2.36	4.24	6.51	-	-	4.01	5.39
f-BRS [41]	hrnet32	COCO+LVIS	1.54	1.69	1.64	2.44	4.37	7.26	2.35	3.44	5.17	6.50
PseudoClick [28]	hrnet32	COCO+LVIS	-	1.50	-	2.08	-	**5.54**	-	-	3.79	5.11
DynaMITe	hrnet32	COCO+LVIS	1.46	1.56	1.48	1.98	3.78	6.32	2.41	3.18	3.9	4.94
FocalClick [8]*	Resnet-50	COCO+LVIS	2.02	2.24	2.43	3.78	5.10	7.70	3.21	4.42	5.34	7.72.
DynaMITe	Resnet-50	COCO+LVIS	1.76	1.78	1.46	2.14	3.97	6.61	2.41	3.34	4.1	5.51
FocalClick [8]	Segformer-B0	COCO+LVIS	**1.40**	1.66	1.59	2.27	4.56	6.86	2.65	3.59	4.04	5.49.
DynaMITe	Segformer-B0	COCO+LVIS	1.50	1.60	1.52	2.02	3.97	6.58	2.39	3.36	3.92	5.16
DynaMITe	Swin-T	COCO+LVIS	1.48	1.58	**1.34**	1.97	3.81	6.38	**2.31**	3.21	3.81	5.00
FocalClick [8]	Segformer-B3	COCO+LVIS	1.44	**1.50**	1.55	1.92	3.53	5.59	2.32	**3.12**	**3.61**	**4.90**
saic-is [15]	Segformer-B?	COCO+LVIS	1.52	1.60	1.40	**1.60**	**3.44**	5.63	-	-	3.68	5.06

Table 1: NoC results on single-instance segmentation datasets grouped by the used backbone. Top results within a group are indicated in red and the overall top results in **bold**. Within groups we obtain state-of-the-art or competetive results.

Method	Backbone	COCO				SBD				DAVIS17			
		NCI ↓	NFO ↓	NFI ↓	IoU ↑	NCI ↓	NFO ↓	NFI ↓	IoU ↑	NCI ↓	NFO ↓	NFI ↓	IoU ↑
FocalClick [8]	Segf-B0(best)	7.31	19422	3004	73.7	4.26	1115	599	87.3	4.6	802	562	84.6
FocalClick [8]	Segf-B0(random)	7.96	29240	3463	59.3	4.81	2408	838	83.4	5.20	1278	685	82.4
FocalClick [8]	Segf-B0(worst)	8.03	31234	3505	60.7	4.91	2723	885	84.8	5.33	1433	689	81.6
DynaMITe	Segf-B0 (best)	6.17	15556	2511	81.2	2.82	679	345	90.2	3.32	535	357	87.5
DynaMITe	Segf-B0 (random)	6.07	13404	2438	84.8	2.77	551	319	90.6	3.29	537	350	87.8
DynaMITe	Segf-B0 (worst)	6.07	19935	2444	82.8	2.74	849	316	90.4	3.27	712	352	86.7
DynaMITe	Swin-T(best)	6.12	15047	2507	81.9	2.73	637	335	90.4	3.15	507	353	87.7
DynaMITe	Swin-T(random)	6.04	**12934**	2451	**85.0**	2.70	**522**	322	**90.7**	3.13	**520**	348	**88.0**
DynaMITe	Swin-T(worst)	**6.00**	19220	**2433**	83.4	**2.69**	820	**316**	90.5	**3.11**	714	**345**	87.0

Table 2: Results on the MIST using an IoU threshold of 85%. NCI: normalised clicks per image, NFO: number of failed objects, NFI: number of failed images. All reported models are trained on COCO+LVIS.

object to refine in each iteration, and the MIST expects a non-overlapping instance map as final output. We fix these issues by adapting the evaluation pipeline of FocalClick to: (i) process all relevant objects sequentially using an initial click to obtain the initial predictions for all objects in an image; (ii) store both the intermediate IoUs and predictions at each refinement step, which are then used to choose the next object to refine and the corresponding simulated next click; and (iii) fuse the final predictions by performing an $argmax$ operation on the set of final predicted probabilities. We also tried to fuse the predictions at each intermediate refinement step but found it to perform worse.

Tab. 2 shows the results of evaluating both FocalClick and DynaMITe on the MIST using the three object sampling strategies (*best*, *worst*, and *random*) explained in Sec. 4. DynaMITe outperforms FocalClick on all metrics and across all three datasets by a large margin. Additionally, the variance in performance across different sampling strategies is much smaller for DynaMITe, demonstrating that it is more robust to variable user click patterns. DynaMITe also generates segmentation masks of higher quality, as shown by the IoU values reported in Tab. 2.

5.2. Ablations

Tab. 3 reports the results of different ablations to analyze the impact of our network design choices (first group), and the positional encodings (second group) for DynaMITe. For all of our ablation experiments, we use a Swin-T [20] backbone with batch size 128, and evaluate it on the SBD [18] dataset for the MIST. Ablations on additional datasets are available in the supplementary.

Transformer Decoder. We ablate the effect of adding a Transformer decoder to the interactive transformer module. The decoder updates the fused image feature map at the

	NCI↓	NFO ↓	NFI↓
DynaMITe (Swin-T)	2.74	561	335
- Transformer decoder	2.87	640	376
- static background queries	2.80	655	356
- temporal positional encoding	2.90	631	386
- spatial positional encoding	2.75	548	330
- spatio-temporal positional encoding	2.92	637	394

Table 3: Ablation on the network design choices, always relative to the top line. All runs are repeated 3 times with random sampling and evaluated on SBD. All metrics use an IoU threshold of 85%.

(a) Examples done after a single click per object.

(b) Examples requiring refinement clicks.

Figure 3: Qualitative examples showing the annotation process with DynaMITe for high-quality masks obtained with a single click per object and for cases that require additional refinements. Clicks are represented with colored dots.

highest resolution based on the instance queries. Discarding the Transformer decoder increases DynaMITe's NCI from 2.74 to 2.87. It also adds an additional 79 failed objects, increasing the NFO from 561 to 640.

Static Background Queries. As mentioned in Sec. 3, we use a common set of 9 learnable queries that model the background in an image. These queries learn generic background representations and help in reducing the number of background clicks required for performing interactive segmentation. As seen in Tab 3, adding the static background queries reduce the NCI from 2.80 to 2.74 and also the NFO from 655 to 561.

Positional Encoding: As clicks are interpreted as spatio-temporal data, we add a 3D positional encoding to the query features Q and ablate its effect on the network performance on the MIST in the second section of Tab. 3. Removing the spatial and temporal positional encoding worsens the network performance by 0.01 and 0.16 NCI respectively. Not having any positional encoding performs the worst with 2.92 NCI as compared to 2.74 for the full network. Temporal positional encodings seem to have a more significant impact compared to the spatial counterpart. This can be partly attributed to the fact that refinement clicks are often spatially close to each other, and hence the spatial positions alone do not provide a good separation.

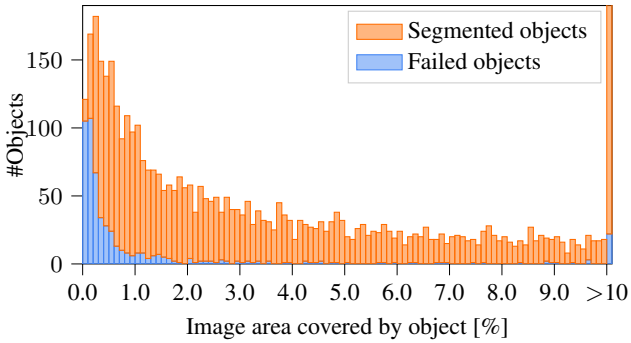


Figure 4: Failure cases analyzed by object size on the SBD dataset. The rightmost bin is truncated and contains 2582 segmented objects.

5.3. Qualitative Results

Fig. 3 shows several qualitative results produced by DynaMITe. The first row shows examples where a single click per object suffices to create well-defined segmentations for all objects. The second and third row show examples where some refinement clicks are needed to arrive at the final masks. While manually annotating images, one can notice that DynaMITe mostly works with few clicks to create sharp masks and potential mistakes are often fixed with very few refinement clicks. Notice for example the single refinement click on one of the zebra's occluded legs in Fig. 3(b) correctly fixed both legs.

5.4. Limitations

Fig. 4 shows how the failure cases are distributed as a function of the relative area of an image they cover. It can clearly be seen that the remaining failure cases of DynaMITe mostly occur on objects covering a small image area. One reason for this is that the highest resolution features map is downsampled by a factor of 4, making it harder to obtain very sharp masks. Coarser object boundaries have a larger impact on the IoU for smaller objects. Here, state-of-the-art single-instance segmentation approaches have the clear advantage that they process a zoomed-in crop around the object [8, 41] and even additionally run a per-object

mask refinement. Such a high-resolution refinement step is orthogonal to our approach and could potentially be integrated into our pipeline, which we leave as future work.

6. Conclusion

We have introduced DynaMITe, a novel Transformer-based interactive segmentation architecture that is capable of performing multi-instance segmentation, and a subsequent evaluation strategy. DynaMITe dynamically generates instance queries based on the user clicks, and uses them within a Transformer architecture to generate and refine the corresponding instance segmentation masks. Unlike existing works, DynaMITe can process user clicks for multiple instances at once without the need to re-compute image-level features. Our method achieves state-of-the-art results on multiple single-instance datasets and outperforms the FocalClick baseline on our novel MIST.

Acknowledgements. This project was funded, in parts, by ERC Consolidator Grant DeeVise (ERC-2017-COG-773161) and BMBF project NeuroSys-D (03ZU1106DA). Several experiments were performed using computing resources granted by RWTH Aachen University under project rwth1239, and by the Gauss Centre for Supercomputing e.V. through the John von Neumann Institute for Computing on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre. We would like to thank Ali Athar, and Idil Esen Zulfikar for helpful discussions.

References

[1] Eirikur Agustsson, Jasper RR Uijlings, and Vittorio Ferrari. Interactive Full Image Segmentation by Considering All Regions Jointly. In *CVPR*, 2019.

[2] Ali Athar, Jonathon Luiten, Alexander Hermans, Deva Ramanan, and Bastian Leibe. HODOR: High-level Object Descriptors for Object Re-segmentation in Video Learned from Static Images. In *CVPR*, 2022.

[3] Arnaud Benard and Michael Gygli. Interactive video object segmentation in the wild. In *arXiv preprint arXiv:1801.00269*, 2017.

[4] Yuri Boykov and Marie-pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *ICCV*, 2001.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[6] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. In *CVPR*, 2017.

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[8] Xi Chen, Zhiyan Zhao, Yilei Zhang, Manni Duan, Donglian Qi, and Hengshuang Zhao. Focalclick: Towards practical interactive image segmentation. In *CVPR*, 2022.

[9] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022.

[10] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021.

[11] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation for autonomous driving. In *CVPR Workshops*, 2017.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[13] 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.

[14] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. In *IJCV*, 2015.

[15] Boris Faizov, Vlad Shakhuro, and Anton Konushin. Interactive image segmentation with transformers. In *ICIP*, 2022.

[16] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019.

[17] 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. *ICCVW*, 2021.

[18] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

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

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[21] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019.

[22] Won-Dong Jang and Chang-Su Kim. Interactive image segmentation via backpropagating refinement scheme. In *CVPR*, 2019.

[23] Shu Kong and Charless C Fowlkes. Recurrent pixel embedding for instance grouping. In *CVPR*, 2018.

[24] Theodora Kontogianni, Michael Gygli, Jasper Uijlings, and Vittorio Ferrari. Continuous adaptation for interactive object segmentation by learning from corrections. In *ECCV*, 2020.

[25] JunHao Liew, Yunchao Wei, Wei Xiong, Sim-Heng Ong, and Jiashi Feng. Regional interactive image segmentation networks. In *ICCV*, 2017.

[26] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *CVPR*, 2016.

[27] 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*, 2014.

[28] 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.

[29] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[31] Sabarinath Mahadevan, Paul Voigtlaender, and Bastian Leibe. Iteratively trained interactive segmentation. In *British Machine Vision Conference (BMVC)*, 2018.

[32] Kevin McGuinness and Noel E O'connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 2010.

[33] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016.

[34] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *CVPR*, 2019.

[35] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *NeurIPS*, 2017.

[36] D. Novotny, S. Albanie, D. Larlus, and A. Vedaldi. Semi-convolutional operators for instance segmentation. In *ECCV*, 2018.

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

[38] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv*, 2017.

[39] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *TOG*, 2004.

[40] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.

[41] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *CVPR*, 2020.

[42] Konstantin Sofiiuk, Ilia Petrov, and Anton Konushin. Reviving iterative training with mask guidance for interactive segmentation. *arXiv preprint arXiv:2102.06583*, 2021.

[43] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020.

[44] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *Proc. Int. Conf. Computer Vision (ICCV)*, 2019.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, NIPS'17, 2017.

[46] 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*, 2019.

[47] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. Deep interactive object selection. In *CVPR*, 2016.

[48] Hongkai Yu, Youjie Zhou, Hui Qian, Min Xian, Yuewei Lin, Dazhou Guo, Kang Zheng, Kareem Abdelfatah, and Song Wang. Loosecut: Interactive image segmentation with loosely bounded boxes. In *ICIP*, 2017.

[49] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *ICLR*, 2020.

DynaMITe : Supplementary Material

Abstract

In this supplementary material, we provide some additional details, ablations and also qualitative results for our approach.

I. Additional Implementation Details

As explained in Sec. 3, DynaMITe takes an image as input, and generates a set of output masks probabilities $Y^t = \{Y_1^t, Y_2^t, ..., Y_n^t\}$ by multiplying the instance encoder's output Q_{out}^t with the output feature map F_{out}^M at timestep t. Here, each Y_i represents a set of object probabilities for $o_i \in \{\mathcal{O}, bg\}$, where bg represents the background. The final segmentation masks $\mathcal{M}^t$ are then obtained by first taking a $\max$ per pixel over each Y_i, and then an argmax over the entire Y^t.

Training. During training, we apply a weighted sum of the binary cross-entropy loss and the dice loss $L = \lambda_1 L_{bce} + \lambda_2 L_{dice}$ [33] on the individual mask probabilities. The network is trained end-to-end using the AdamW [30] optimizer for 50 epochs with a batch size of 32 and an initial learning rate of $1e-4$, which is then decayed by 0.1 after 44 and 48 epochs respectively. The models used for ablation are trained with batch size 128 and an initial learning rate of $5e-4$.

II. MIST: Additional Evaluation Strategies

In Sec. 4 we discussed a number of click simulation strategies that could potentially capture some of the user patterns for the MIST. Since these simulation strategies are not exhaustive, we discuss a few more such next-click strategies that could be used to better emulate how a user might perform a MIST. We also evaluate DynaMITe on all of these strategies in Tab. IV, and once again confirm that our model is robust against different user patterns.

Round-robin: The round-robin strategy assigns a click window of β clicks for each of the objects in an image. Here, an object is chosen randomly and after the current object of focus exhausts all the β clicks, the next random object is chosen and then refined until completion. Once all the objects in the input image are processed in this manner, the round-robin strategy revisits all the failed objects and then tries to refine their segmentation masks either until all the objects are fully segmented, or until the image-level click budget $\tau * |\mathcal{O}|$ is fully used up.

Worst with limit: Here, in each iteration we choose the object with the worst IoU, as we also do in the *worst* strategy described in Sec. 4, but we additionally add a per-object click limit β to each object. Upon selecting the next worst object, we first check if this object has not reached its click limit and if it did, we skip this object until all objects have either been segmented or reached their limit. After this is the case, we switch to the *best* strategy and try to segment the remaining objects as usual until the image budget is used up or all objects are segmented. The intuition behind this strategy is that a user will try to improve the biggest errors first, but they will notice when an object is not segmentable by the method at hand and rather spend more clicks on objects which can be segmented properly.

Max-distance: In this strategy, we again start by adding a positive click to each of the foreground objects. During refinement, the next click is simply sampled on the pixel with the maximum distance from the distance transform computed on the error region of the entire semantic map that includes the segmentation masks for all objects in an image. If the chosen pixel falls on an object, then a corresponding positive click is added to that object, and if it doesn't, then it is classified as a negative click.

For the results reported in Tab. IV, we use $\tau = 10$ and $\beta = 10$. All of the strategies work and *worst with limit* actually results in a lower number of failed objects in all cases, while having comparable NCI. The *max-distance* strategy is actually amongst the worst, resulting in the highest number of failed images. A potential reason could be that due to the joint maximum distance transform over all object errors, the clicks are no longer sampled in order to specifically correct a mistake with respect to one object and are thus less targeted. This in turn might lead to failed objects, where the other strategies that rely on a per-object distance transform actually are able to sample clicks in more useful locations.

III. Extended Ablations

Here, we extend the ablation experiments performed in Sec. 5.2 to additional datasets. Tab V and Tab. VI report the results of the ablation experiments on additional multi-instance and single-instance datasets respectively. As it can be seen from these experiments, our final model with spaio-temporal positional encoding consistently outperforms other variants, and is robust towards different task settings. Although, as stated in Sec. 5.2, the impact of the spatial embedding seems to be less significant compared to the temporal counterpart in Tab V, they are still important for reducing the overall number of clicks especially in the single-instance setting (ref Tab. VI).

Backbone	Strategy	COCO				SBD				DAVIS17			
		NCI ↓	NFO ↓	NFI ↓	IoU ↑	NCI ↓	NFO ↓	NFI ↓	IoU ↑	NCI ↓	NFO ↓	NFI ↓	IoU ↑
Segf-B0	best	6.17	15556	2511	81.2	2.82	679	345	90.2	3.32	535	357	87.5
Segf-B0	random	6.07	13404	2438	84.8	2.77	551	319	90.6	3.29	537	350	87.8
Segf-B0	worst	6.07	19935	2444	82.8	2.74	849	316	90.4	3.27	712	352	86.7
Segf-B0	max-distance	6.79	14757	2894	85.4	3.19	754	461	90.8	3.44	573	380	88.1
Segf-B0	round-robin	6.45	15194	2471	83.9	3.48	657	346	90.5	3.99	587	358	87.7
Segf-B0	worst with limit	6.07	13086	2446	84.5	2.75	519	315	90.5	3.28	511	350	87.8
Swin-T	best	6.12	15047	2507	81.9	2.73	637	335	90.4	3.15	507	353	87.7
Swin-T	random	6.04	**12934**	2451	85.0	2.70	522	322	90.7	3.13	520	348	88.0
Swin-T	worst	6.00	19220	**2433**	83.4	2.69	820	316	90.5	3.11	714	345	87.0
Swin-T	max-distance	6.74	14642	2880	**85.5**	3.13	742	454	**90.9**	3.25	555	372	88.2
Swin-T	round-robin	6.42	14635	2482	84.2	3.40	614	335	90.6	3.81	551	350	88.0
Swin-T	worst with limit	**6.01**	12940	2438	84.7	**2.68**	**516**	**313**	90.7	**3.11**	510	345	88.0
Resnet50	best	6.28	16074	2583	80.7	2.87	706	376	90.0	3.46	583	385	86.7
Resnet50	random	6.20	13993	2519	84.2	2.83	603	356	90.5	3.44	602	382	87.1
Resnet50	worst	6.17	20514	2507	82.2	2.81	936	350	90.2	3.41	778	377	86.1
Resnet50	max-distance	6.86	15276	2927	84.8	3.23	784	468	90.7	3.58	639	407	87.3
Resnet50	round-robin	6.56	15675	2536	83.3	3.52	662	365	90.4	4.12	631	386	87.0
Resnet50	worst with limit	6.17	13730	2511	83.7	2.82	569	352	90.4	3.41	588	378	87.0
hrnet32	best	6.12	15237	2510	81.6	2.75	657	348	90.3	3.21	510	349	87.7
hrnet32	random	6.05	13133	2458	84.9	2.72	564	330	90.6	3.18	524	344	87.9
hrnet32	worst	6.03	19625	2454	83.2	2.70	874	326	90.4	3.17	684	340	87.2
hrnet32	max-distance	6.74	14602	2876	85.4	3.13	762	455	90.8	3.34	563	378	**88.3**
hrnet32	round-robin	6.43	14893	2495	84.0	3.40	616	330	90.6	3.86	547	343	87.9
hrnet32	worst with limit	6.03	12862	2452	84.6	2.70	532	323	90.6	3.17	**505**	**342**	88.0

Table IV: Results on MIST using an IoU threshold of 85%. NCI: normalised clicks per image, NFO: number of failed objects, NFI: number of failed images. All reported models are trained on COCO+LVIS.

IV. Runtime and Memory Analysis

As discussed in Sec. 3, DynaMITe translates each click into a query to our interactive transformer module. Hence, the number of queries processed by the transformer increases over time during the iterative refinement process. In Figure 5, we analyze the impact of such a growing query pool in terms of runtime and GPU memory consumed during inference. Both the runtime and the memory increases as the transformer receives more queries, but the scale-up is quite slow and falls within a reasonable limit for practical usage. As shown in Fig. 5a and Fig. 5b, the runtime increases from 17ms to 34ms as the number of clicks increases from 1 to 200, and the memory used increases from around 800MB to 3.2GB. For a large scale dataset like COCO with an average of 7.3 instances per image, DynaMITe would need about 47 queries (since NCI is 6.4) in the final refinement step and hence the average maximum runtime for a refinement step would be about 23.5ms. The values reported for both of these experiments in Fig. 5 are an average over the entire GrabCut dataset on an Nvidia 3090 GPU with 24GB of memory.

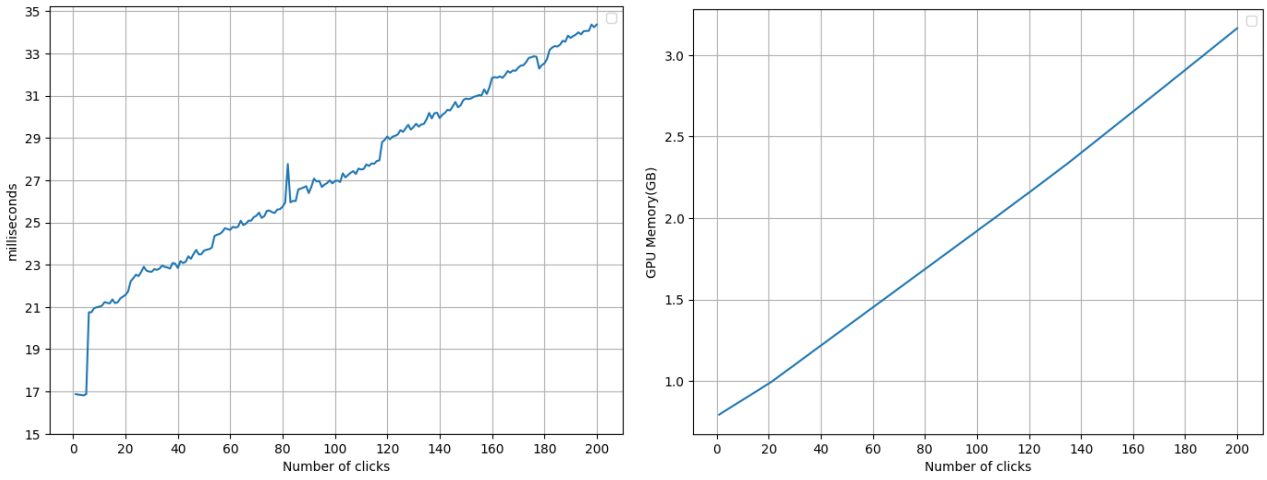
V. Refinement Analysis

In this section, we analyze the refinement quality of different variants of DynaMITe for the single-instance setting. Fig. 6 plots change in instance segmentation quality after each refinement iteration on various single-instance datasets. DynaMITe can achieve a high segmentation quality with very few clicks and can further refine the instances very well with additional clicks. Eg. for GrabCut, DynaMITe achieves 84% IoU on average with just one click, and then refines them to close to 100% IoU.

VI. Annotation Tool

For using DynaMITe in practice, we build a click based annotation tool that can perform multi-instance interactive segmentation. Our tool is built using the python based GUI toolkit *Tkinter*, and is based on the RITM [42] annotation tool. The DynaMITe annotation tool supports addition and deletion of instances within an image, and also allows a user to switch back and forth between instances to perform mask refinement. To get a glimpse of our tool, please watch the video on the project page.

It should be noted that this tool is only a prototype and cannot be seen as a proper tool that was optimized for the



(a) Runtime Analysis

(b) Memory Analysis

Figure 5: Runtime and memory scaling with respect to the number of clicks for the interactive transformer.

	COCO			SBD			DAVIS17		
	NCI ↓	NFO ↓	NFI ↓	NCI ↓	NFO ↓	NFI ↓	NCI ↓	NFO ↓	NFI ↓
DynaMITe (Swin-T)	6.11	13383	2484	2.74	561	335	3.19	564	356
- static background queries	6.19	14477	2542	2.80	655	356	3.19	561	359
- Transformer decoder	6.32	14118	2640	2.87	640	376	3.32	590	373
- temporal positional encoding	6.28	13837	2599	2.90	631	386	3.34	603	378
- spatial positional encoding	6.10	13161	2479	2.74	548	330	3.23	540	358
- spatio-temporal positional encoding	6.33	13964	2630	2.92	637	394	3.37	593	380

Table V: Ablation on the network design choices, always relative to the top line. NCI: normalised clicks per image, NFO: number of failed objects, NFI: number of failed images. All reported models are trained on COCO+LVIS.

	GrabCut [40]		Berkeley [32]		SBD [18]		COCO MVal		DAVIS [37]	
	@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓
DynaMITe (Swin-T)	1.54	1.58	1.34	1.98	3.83	6.47	2.29	3.10	3.84	5.11
- static background queries	1.42	1.56	1.33	1.98	4.02	6.70	2.35	3.37	3.80	4.99
- Transformer decoder	1.64	1.70	1.35	2.31	4.17	6.86	2.47	3.52	4.02	5.44
- temporal positional encoding	1.44	1.58	1.52	2.09	4.00	6.72	2.31	3.29	4.14	5.44
- spatial positional encoding	1.40	1.52	1.39	2.21	3.88	6.50	2.32	3.27	3.78	5.23
- spatio-temporal positional encoding	1.60	1.66	1.50	1.96	4.10	6.79	2.28	3.32	4.10	5.69

Table VI: Ablation on network design choice, on single-instance segmentation datasets, always relative to the top line.

best possible user experience. Many improvements could be thought of, *e.g.* one could optimize the switching between objects by right-clicking on existing masks and keyboard shortcuts could be included for actions such as creating a new object. We could also easily extend the tool with additional functionalities such as the removal of existing clicks, since this is supported out of the box by DynaMITe. A detailed exploration of this design space is outside of our expertise and the scope of this paper.

VII. Qualitative Results

In Fig. 7, we show additional multi-instance segmentation results for sequential segmentation process using DynaMITe. Here we follow the *random* strategy, where we first sample a single click per object, after which we iteratively select a random object to refine. In most cases, DynaMITe starts out with a high average IoU after a single click per object and the resulting masks are often arguably better than the corresponding ground truth segmentation, *e.g.* row 3, 5, and 6. Nevertheless, in most cases we can also adjust to arbitrary mistakes present in the ground truth anno-

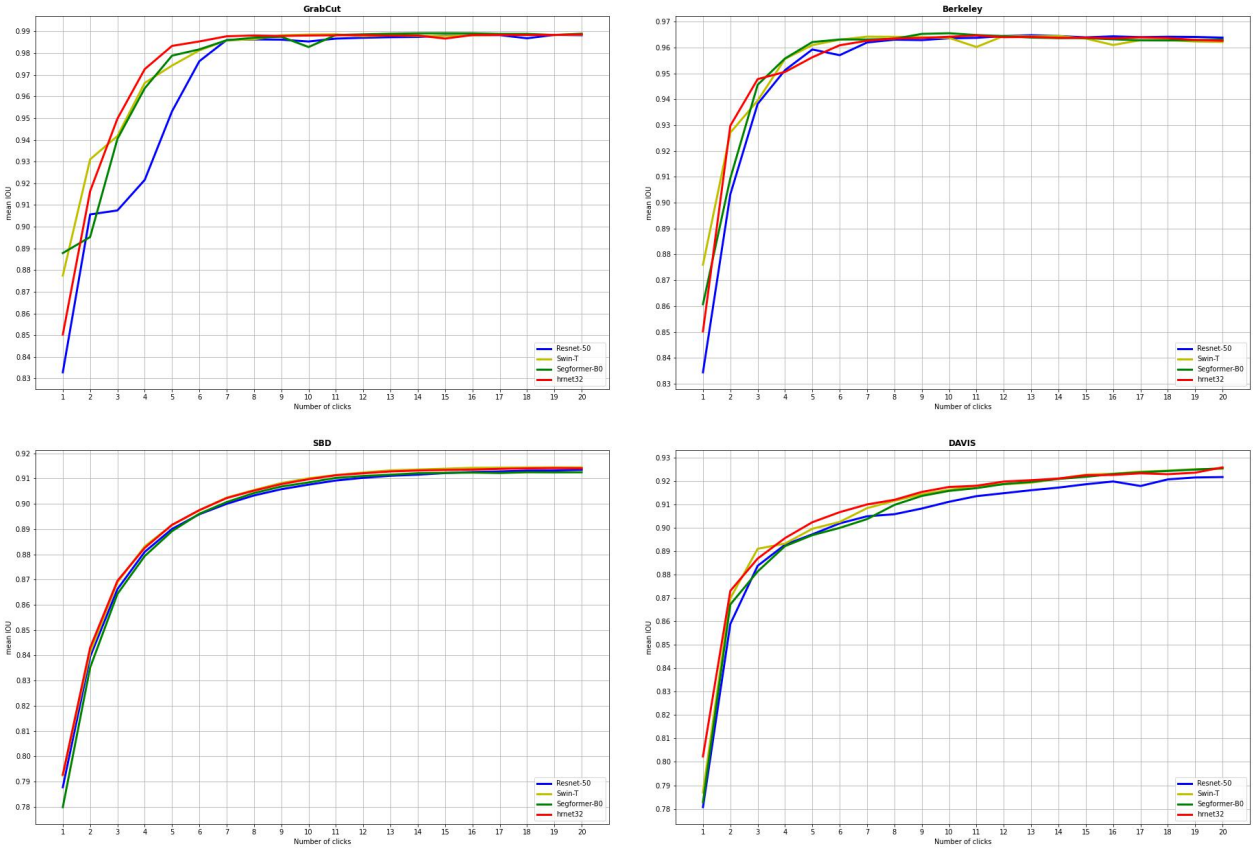


Figure 6: IoU vs. number of clicks for multiple single-instance datasets.

tations. There are also some interesting failure cases such as the one shown in Fig. 8, where DynaMITe fails to capture the thin ropes of the kite. Although DynaMITe can segment fairly thin structures in practice, the automatic click sampling fails to sample the necessary additional clicks for DynaMITe to segment the ropes in this particular case.

(a) Ground truth (b) $\tau = 1$ (c) $\tau = 3$ (d) $\tau = 5$

Figure 7: Qualitative examples based on our automatic random click sampling strategy. We show the ground truth and how the segmentation looks after a click budge of $\tau * |\mathcal{O}|$. For $\tau = 1$ we click on each object exactly once. The bottom left corner of each image shows the average IoU.

(a) Ground truth (b) $\tau = 1$ (c) $\tau = 3$ (d) $\tau = 5$

Figure 8: A qualitative example of a negative result. Even though both the board and the ropes of the kite are segmented badly, the board can be recovered with a few additional clicks. After a total of 15 clicks, the refinement is not able to segment the ropes though. Given that the refinement clicks are sampled based on a maximum distance transform, no clicks are sampled for the very thin structure, even though DynaMITe might actually be able to segment such structures.