Clicks as Queries: Interactive Transformer for Multi-instance Segmentation

Amit Kumar Rana* Sabarinath Mahadevan* Alexander Hermans
Bastian Leibe

RWTH Aachen University, Germany
firstname.lastname@rwth-aachen.de

Abstract

Transformers have percolated into a multitude of computer vision domains including dense prediction tasks such as instance segmentation and have demonstrated strong performances. Existing transformer based segmentation approaches such as Mask2Former generate pixel-precise object masks automatically given an input image. While these methods are capable of generating high quality masks in general, they have an inherent class bias and are unable to incorporate user inputs to either segment out-of-distribution classes or to correct bad predictions. Hence, we introduce a novel module called Interactive Transformer that enables transformers to predict and refine objects based on user interactions. Subsequently, we use our Interactive Transformer to develop an interactive segmentation network that can generate mask predictions based on user clicks and thereby widen the transformer application domains within computer vision. In addition, the Interactive Transformer can make such interactive segmentation tasks more efficient by (i) imparting the ability to perform multi-instances segmentation, (ii) alleviating the need to recompute image-level backbone features as done in existing interactive segmentation networks, and (iii) reducing the required number of user interactions by modeling a common background representation. Our transformer-based architecture outperforms the state-of-the-art interactive segmentation networks on multiple benchmark datasets.

1. Introduction

Transformers have gained in popularity for various computer vision tasks over the past few years [1,3,4,6,13]. Due to their ability to capture long-range dependencies and their general flexibility, Transformers find applicability in a wide range of 2D and 3D vision tasks. Mask2Former [3] has shown that simple Transformer modules are very effective in capturing good object representations, and can perform image and video level instance segmentation very well.

However, automatic instance segmentation methods like Mask2Former have multiple limitations: (i) they are designed to perform instance segmentation without the possibility for users to correct the output masks in case of bad predictions, (ii) such automatic segmentation methods have class biases, and hence tends to segment the object classes that they have seen during training, and (iii) all objects are segmented by default and hence it is not possible to select a relevant subset of them without additional post-processing steps. To alleviate these disadvantages while still leveraging the capability of Transformers, we propose a novel formulation called *Interactive Transformer*. Our Interactive Transformer can perform instance segmentation based on some user guidance signal, while allowing refinement of initial predictions where needed. It does so by dynamically generating queries to a Transformer module, that are conditioned on the given user interactions at a given timestep.

To show the efficacy of our new module, we develop an interactive segmentation network called DynaMITe that can annotate the objects of interest in an image using user clicks as guidance. The Interactive Transformer module also enables us to remove two major problems with existing interactive segmentation networks [2, 12, 15, 21, 22]: (i) they are designed to perform binary segmentation and hence can process only one object at a time, and (ii) these methods need to re-compute backbone features every time they have to process a new user click, thereby limiting their network sizes to achieve a good runtime performance. In contrast, the Interactive Transformer in DynaMITe can inherently handle clicks from multiple instances and requires only a forward pass through the transformer module for every new user click. Transferring clicks into queries also allows multiple objects in the image to interact through the self-attention operation, and correspondingly learn a common background representation. This enables DynaMITe to learn a better context from the input image as compared to existing interactive segmentation models.

The classic interactive segmentation benchmarks focus

*Equal contribution.

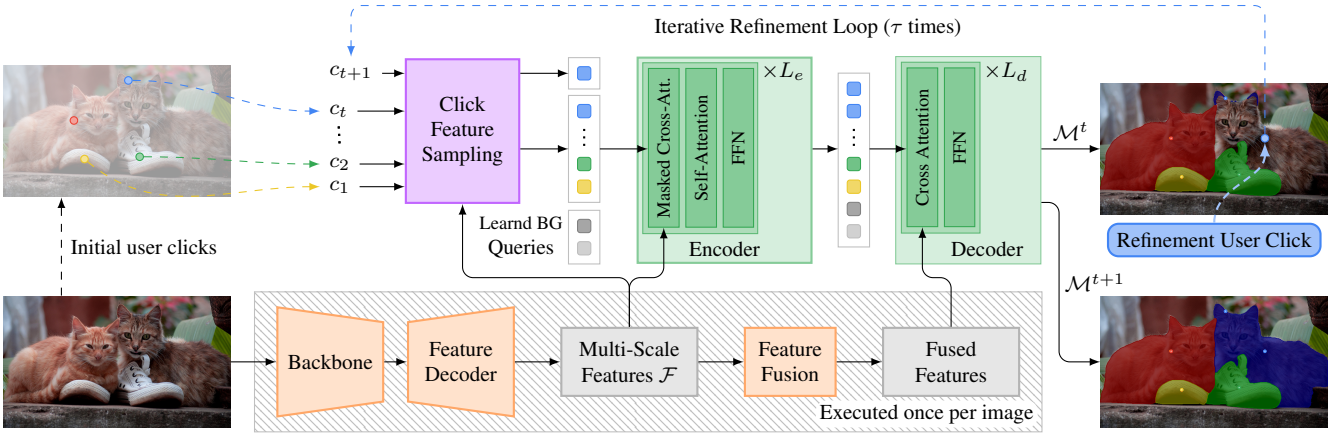


Figure 1. DynaMITe consists of a backbone, a feature decoder, and an interactive Transformer. Point features at click locations at time t are translated into queries which, along with the multi-scale features, are processed by a Transformer encoder-decoder structure to generate a set of output masks $\mathcal{M}^t$ 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 iterated τ times until the masks are fully refined.

on binary segmentation, where a single-instance is always picked and annotated iteratively using refinement clicks. Since our Interactive Transformer provides the capability to perform multi-instance interactive segmentation, we also propose a new multi-instance interactive segmentation task (MIST), and an associated evaluation strategy. Correspondingly, we evaluate DynaMITe on both MIST and the classic benchmarks, and show that our Transformer-based formulation that bootstraps queries from user-clicks outperform the state-of-the-art methods in both tasks.

2. Interactive Transformer

Existing interactive segmentation methods [2,21,22] that use a Transformer mainly use a SegFormer [23] backbone as a feature extractor to obtain a localized feature map based on click maps, that is updated when a user provides a new refinement click. Such methods can still only process one click at a time and perform the task sequentially per object, thus preventing any interaction between the object instances within an image. This limits the role of a transformer model to being just a better backbone for interactive segmentation.

We instead formulate the user clicks as a spatio-temporal sequence of data and translate them into queries that are processed by our Interactive Transformer module. In our formulation queries are derived from features at click location, each being assigned to one object within the image. These queries can interact between each other which enables a common background modeling, thereby reducing redundancy in background clicks. Hence, given the inputs $\mathcal{F}$ and the corresponding set of clicks $\mathcal{S}^t$ at refinement timestep t, our Interactive Transformer generates a disjoint set of segmentation masks $\mathcal{M}^t = \{M_1^t, M_2^t, ..., M_n^t\}$ for all the relevant foreground objects at t.

Dynamic Query Bootstrapping. At every timestep t, the queries Q^t used by the Transformer are dynamically generated from input features $\mathcal{F}$ by sampling the point features at every spatial location represented by the user clicks in $\mathcal{S}^t$.

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

Here $q_j \in Q^t$ represent the queries for clicks $c_j \in \mathcal{S}^t$, and $f \in \mathcal{F}$ are the feature maps at different scales. These queries are thus dynamically updated throughout the iterative process without the need to recompute $\mathcal{F}$. 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 are vital in reducing 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.

2.1. Network Architecture

Based on the Interactive Transformer design, we build an efficient multi-instance interactive segmentation network called *Dynamic Query Bootstrapping for Multi-object Interactive Segmentation Transformer (DynaMITe)*. Fig. 1 shows the overall architecture of DynaMITe which takes an input RGB image $\mathcal{I} \in \mathbb{R}^{H \times W \times 3}$, and the corresponding set of user clicks $\mathcal{S}^t = \{c_1, c_2, ..., c_t\}$ at timestep $t \in \{1, \ldots, \mathcal{T}\}$ as inputs. The user clicks in $\mathcal{S}^t$ can either be a positive click representing a foreground object or a negative click representing the common background. $\mathcal{S}^t$ is updated during the refinement process when the network receives a new interaction c_{t+1} at the time step $t + 1$. Unlike previous works, DynaMITe can handle multiple instances at once and hence the positive clicks in $\mathcal{S}^t$ can be on different

foreground objects. 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 fed to the Interactive Transformer.

The Interactive Transformer has an encoder-decoder structure. The encoder follows the Transformer decoder architecture from [3], and also uses their masked cross-attention module to restrict the attention operation to the predicted target region from the previous layers. It takes as input the queries Q^t and the multi-scale feature maps $\mathcal{F}$, and generates descriptors for each click locations. The keys and values for the encoder are derived from $\mathcal{F}$ at the corresponding feature scale similar to [3]. The decoder on the other hand uses the click descriptors generated by the encoder to update the image features via cross-attention. Hence, the click descriptors form the keys and values to the decoder, while the queries are obtained by fusing the multi-scale features $\mathcal{F}$ obtained from the feature extractor.

3. Experiments

We evaluate DynaMITe on an extensive range of datasets across two interactive segmentation task settings: (i) the well established single-instance setting using small-scale datasets mostly containing one object instance per image such as GrabCut [20], Berkeley [16], COCO MVal, and DAVIS [18] and (ii) our novel multi-instance segmentation task (MIST) on larger multi-instance datasets like COCO [11], DAVIS17 [19], and SBD [7].

Implementation Details. We use either a Swin Transformer [9] or a SegFormer as backbone, with a multi-scale deformable-attention Transformer [24] on top. The encoder for our interactive Transformer has 9 transformer layers while the decoder uses 5 cross-attention layers. We initialize the backbone with ImageNet [5] pretrained weights, and the entire network is trained end-to-end on the combined COCO+LVIS [22] dataset with an input resolution of 1024 $\times$ 1024 px for 50 epochs. We use a batch size of 128 and train it on 64 Nvidia A100 GPUs. We always train our network for multi-instance segmentation, even when evaluated in the single instance setting.

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

Single Instance Setting. In Tab 1 we compare our results against state-of-the art methods on the classic single-instance interactive segmentation setting. Here, we follow the same evaluation setting adopted in previous works [2, 15, 21, 22], *I.e.* segment one instance at a time and then refine it either until the mask has a satisfactory quality, or until a specific click budget $\tau = 20$ has been exhausted [15]. We also use the average number of clicks per object (NoC) metric, similar to previous works, for comparison. Under this setting, DynaMITe outperforms all comparable state-of-the-art networks, on a majority of the reported datasets. This includes both methods that use Transformer backbones, as well the traditionally popular HRNet backbones. Our method with a smaller backbone also performs comparably with FocalClick [2] that uses a larger Segformer-B3 backbone. This shows the ability of our Interactive Transformer module to generalize to a task setting that it was not specifically trained for.

Multi-instance Interactive Segmentation Task. For our novel MIST, we use the following next-click simulation strategy for automatic evaluation: (i) generate initial predictions for each object by placing a positive click in the center, (ii) choose a random object from this prediction set that has not achieved the required segmentation quality, and (ii) place a click on the largest error region for the chosen object. The sampling process is repeated until either the entire image is segmented, or until we exhaust an image-level click budget $\tau * |\mathcal{O}|$, where $|\mathcal{O}|$ is the number of foreground objects in an image and $\tau = 10$. For this task, we use a new metric called Normalized Clicks per Image (NCI), which is obtained by normalizing the total number of clicks used for an input image by the number of foreground objects in that image. Additionally, we also mark the average number of failed objects (NFO), number of failed images (NFI), and the average IoU achieved at the end of the evaluation process. As a baseline, we adapt state-of-the-art FocalClick [2] to the MIST setting. Tab. 2 compares the performance of DynaMITe against the adapted FocalClick. Our method outperforms the baseline on all metrics for this task and achieves a better final segmentation quality as shown by the IoU values, highlighting the benefit of jointly predicting multiple instance segmentations. For a more detailed explanation on MIST, please refer to the supplementary material.

3.2. Ablations

We ablate the impact of our design choices for our Interactive Transformer module in Tab. 3. For these ablations, we use a Swin-T [8] backbone, and evaluate it on MIST. First, we analyze the advantages of having an additional decoder in the Interactive Transformer. As it can be seen from Tab. 3, removing this decoder reduces the performance of DynaMITe from 2.74 NCI to 2.87 NCI and also increases the number of failed objects by 79. Likewise, removing the static background queries reduces the NCI from 2.74 to 2.80 and also increases the NFO from 561 to 655.

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

Method	Backbone	Train Data	GrabCut [20]		Berkeley [16]		SBD [7]		COCO MVal		DAVIS [18]	
			@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓
RITM [22]	hrnet32	COCO+LVIS	1.46	1.56	-	2.10	3.59	5.71	-	-	4.11	5.34
f-BRS [21]	hrnet32	COCO+LVIS	1.54	1.69	1.64	2.44	4.37	7.26	2.35	3.44	5.17	6.50
PseudoClick [12]	hrnet32	COCO+LVIS	-	1.50	-	2.08	-	**5.54**	-	-	3.79	5.11
DynaMITe (Ours)	hrnet32	COCO+LVIS	1.46	1.56	1.48	1.98	3.78	6.32	2.41	3.18	3.9	4.94
FocalClick [2]	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 (Ours)	Segformer-B0	COCO+LVIS	1.50	1.60	1.52	2.02	3.97	6.58	2.39	3.36	3.92	5.16
FocalClick [2]	Segformer-B3	COCO+LVIS	1.44	**1.50**	1.55	1.92	3.53	5.59	2.32	3.12	**3.61**	**4.90**
DynaMITe (Ours)	Swin-T	COCO+LVIS	1.48	1.58	**1.34**	1.97	3.81	6.38	2.31	3.21	3.81	5.00
DynaMITe (Ours)	Swin-L	COCO+LVIS	1.62	1.72	1.39	**1.90**	**3.32**	5.64	**2.19**	**2.88**	3.8	5.09

Table 1. NoC results on single-instance segmentation datasets grouped by the backbone used. 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 competitive results.

Method	Backbone	COCO				SBD				DAVIS17			
		NCI ↓	NFO ↓	NFI ↓	IoU ↑	NCI ↓	NFO ↓	NFI ↓	IoU ↑	NCI ↓	NFO ↓	NFI ↓	IoU ↑
FocalClick [2]	Segformer-B0	7.31	19422	3004	73.7	4.26	1115	599	87.3	4.6	802	562	84.6
DynaMITe (Ours)	Segformer-B0	6.07	13404	2438	84.8	2.77	551	319	90.6	3.29	537	350	87.8
DynaMITe (Ours)	Swin-T	6.04	12934	2451	85.0	2.70	522	322	90.7	3.13	520	348	88.0
DynaMITe (Ours)	Swin-L	**5.74**	**11976**	**2259**	**85.2**	**2.45**	**422**	**247**	**90.9**	**3.07**	**501**	**339**	**88.7**

Table 2. 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. DynaMITe produces better segmentations while requiring fewer clicks.

	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

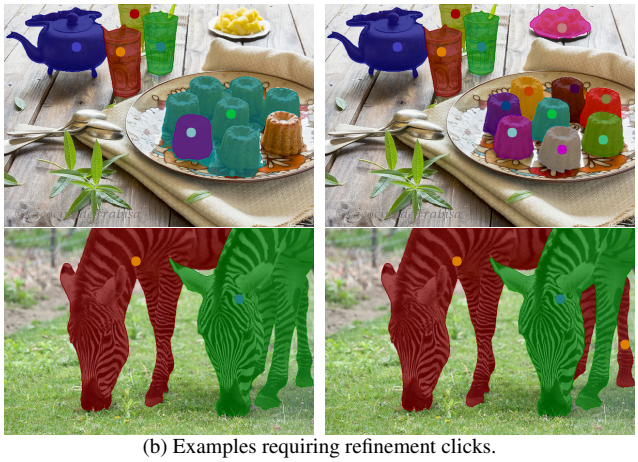
Table 3. Ablation on the network design choices, always relative to the top line. All runs are repeated 3 times with random sampling and evaluated on SBD. All metrics use an IoU threshold of 85%.

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.

3.3. Qualitative Results

Fig. 2 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 few refinement clicks. Notice for example the single refinement click on one of the zebra's occluded legs in Fig. 2(b) correctly fixed both legs. We refer the reader to the supplementary material for more qualitative results.

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



(b) Examples requiring refinement clicks.

Figure 2. 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.

4. Conclusion

We have introduced a novel Interactive Transformer module that can segment and refine object masks based on user clicks. To achieve this, queries to the Trans-

former are dynamically initialized from the corresponding click-features. We show its effectiveness by developing a multi-instance interactive segmentation architecture called DynaMITe. Unlike existing works, DynaMITe can process user clicks for multiple instances at once without the need to re-compute image-level features and achieves state-of-the-art results on multiple interactive segmentation benchmarks. These results are enabled by the Transformer attention operations, highlighting another interesting use-case.

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] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1

[2] Xi Chen, Zhiyan Zhao, Yilei Zhang, Manni Duan, Donglian Qi, and Hengshuang Zhao. Focalclick: Towards practical interactive image segmentation. In *CVPR*, 2022. 1, 2, 3, 4, 6

[3] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. 1, 3

[4] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021. 1

[5] 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. 3

[6] 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. 1

[7] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 3, 4

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[9] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019. 3

[10] Theodora Kontogianni, Michael Gygli, Jasper Uijlings, and Vittorio Ferrari. Continuous adaptation for interactive object segmentation by learning from corrections. In *ECCV*, 2020. 6

[11] 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. 3

[12] 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. 1, 4

[13] 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. 1

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6

[15] Sabarinath Mahadevan, Paul Voigtlaender, and Bastian Leibe. Iteratively trained interactive segmentation. In *British Machine Vision Conference (BMVC)*, 2018. 1, 3, 6

[16] Kevin McGuinness and Noel E O'connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 2010. 3, 4

[17] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016. 6

[18] 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. 3, 4

[19] 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. 3

[20] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004. 3, 4

[21] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *CVPR*, 2020. 1, 2, 3, 4, 6

[22] Konstantin Sofiiuk, Ilia Petrov, and Anton Konushin. Reviving iterative training with mask guidance for interactive segmentation. *arXiv preprint arXiv:2102.06583*, 2021. 1, 2, 3, 4, 6

[23] 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*, 2021. 2

[24] 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. 3

Supplementary Material

I. Additional Implementation Details

As explained in 2, 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 $L = \lambda_1 L_{bce} + \lambda_2 L_{dice}$ [17] on the individual mask probabilities. The network is trained end-to-end using the AdamW [14] optimizer with a batch size of 128 and an initial learning rate of $5e-4$, which is then decayed by 0.1 after 44 and 48 epochs respectively.

II. Multi-instance Interactive Segmentation (MIST)

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.

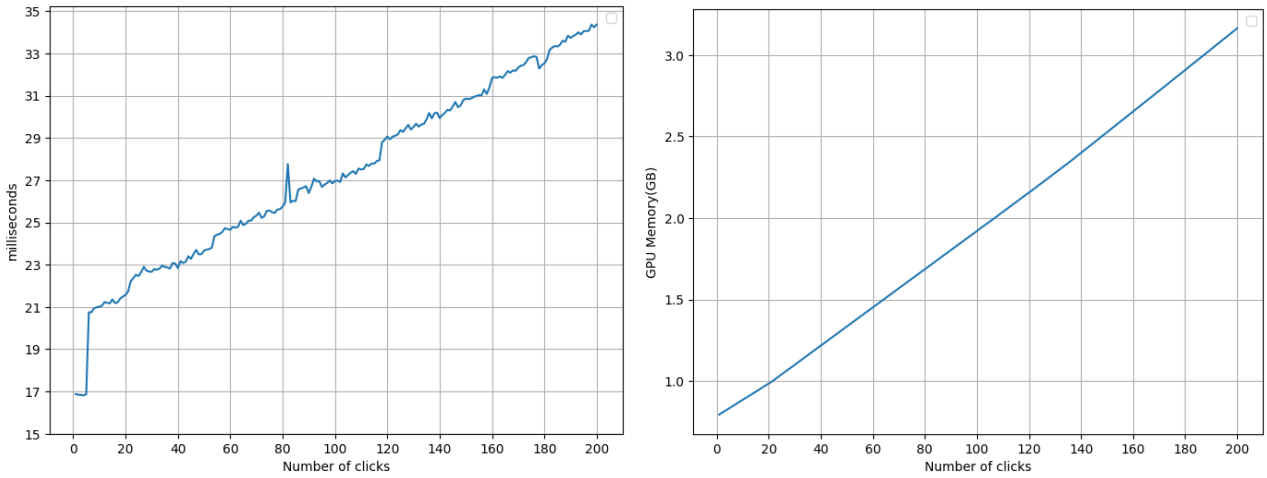
Given an input image and a common set of user clicks, MIST expects a corresponding method to generate non-overlapping instance masks for all relevant foreground objects. A major difference in this setting is that the background, and the corresponding negative clicks, are now common for all object instances. 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 MIST is not trivial.

Automatic Evaluation. It is also important to note that the user click patterns for MIST may differ considerably between users. As a result, simulating 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 [2, 10, 15, 21, 22], the refinement focus in MIST may jump from one object to another in an arbitrary sequence, unless users are instructed to process the objects in a specific order. Since it is hard to predict what next-object/next-click selection strategies users will end up picking 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 MIST should ideally be robust against varying click patterns and next-object selection strategies.

We start by adding a single positive click to each of the foreground objects in that image to get an initial prediction. For evaluating FocalClick, we choose the next object for refinement with quality closest to the desired segmentation quality i.e choose the object that has the best IoU, compared to the ground truth mask. To showcase the robustness of our method against varying click patterns, we select a random object for refinement in each step while evaluating our method. We also tried the same strategy for FocalClick but since it is not designed for multi-instance segmentation, this setting gives poor results. In each of the refinement steps, 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.

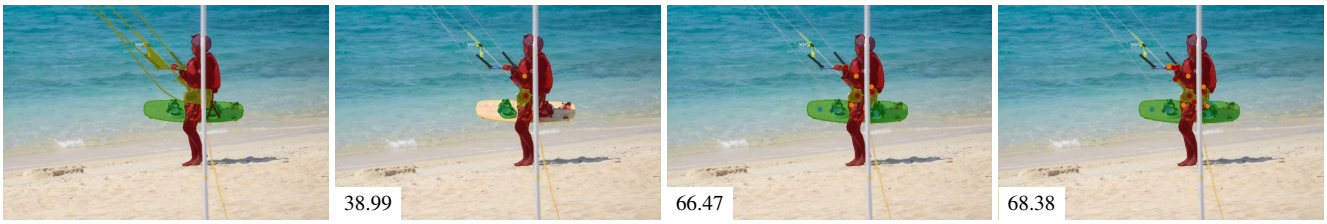
III. Runtime and Memory Analysis

As discussed in 2, 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 3, 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. 3a and Fig. 3b, 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 av-



(a) Runtime Analysis (b) Memory Analysis

Figure 3. Runtime and memory scaling with respect to the number of clicks for the interactive transformer.



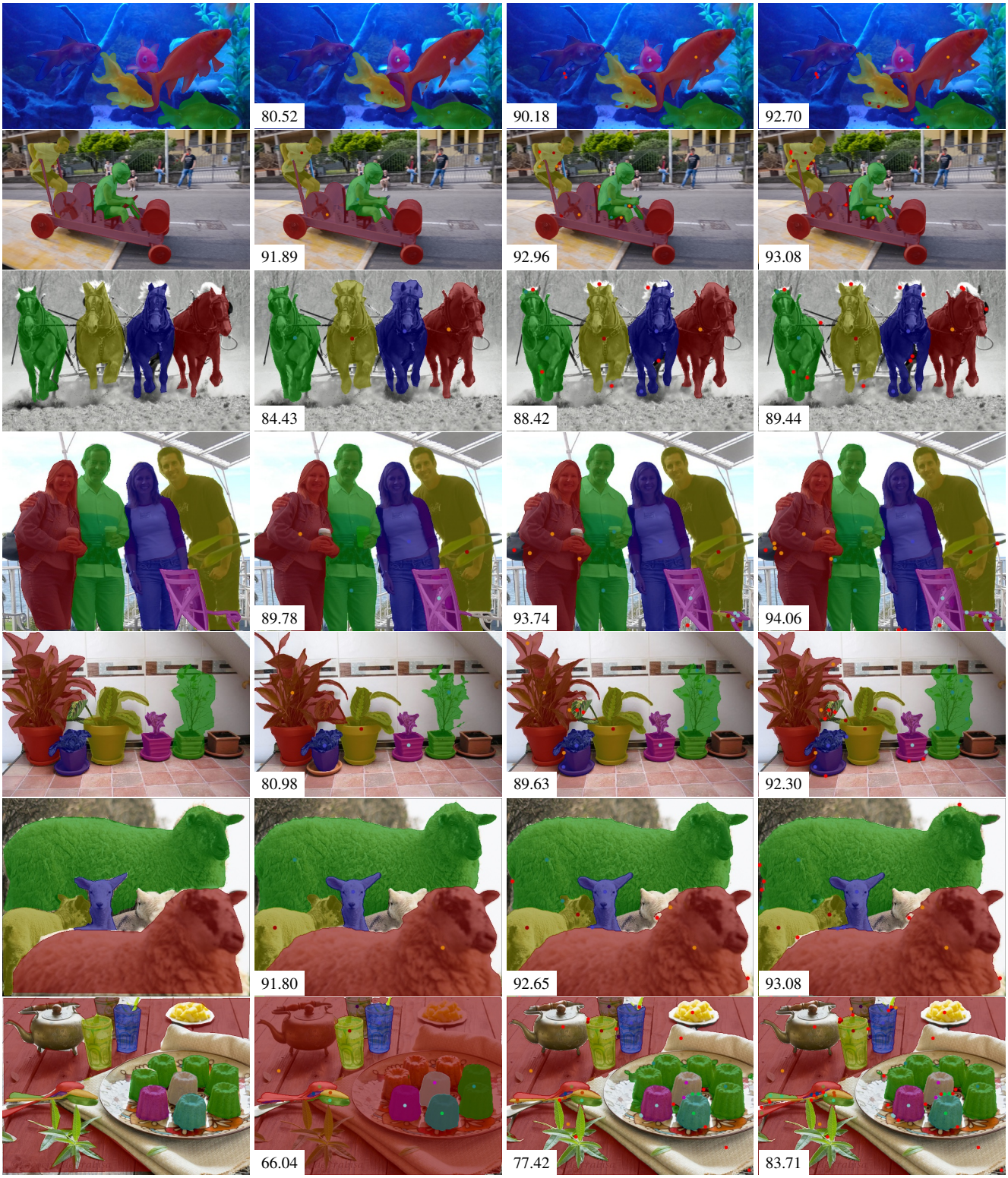
(a) Ground truth (b) $\tau = 1$ (c) $\tau = 3$ (d) $\tau = 5$

Figure 4. 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 recover with 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.

erage 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. 3 are an average over the entire GrabCut dataset on an Nvidia 3090 GPU with 24GB of memory.

IV. Qualitative Results

We first show an interesting failure case in Fig. 4, 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, due to the use of a maximum distance transform. In Fig. 5, 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 annotations.



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

Figure 5. 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.