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 $L = \lambda_1 L_{bce} + \lambda_2 L_{dice}$ [4] on the individual mask probabilities. The network is trained end-to-end using the AdamW [2] 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. I, 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. I, 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 II and Tab. III re-

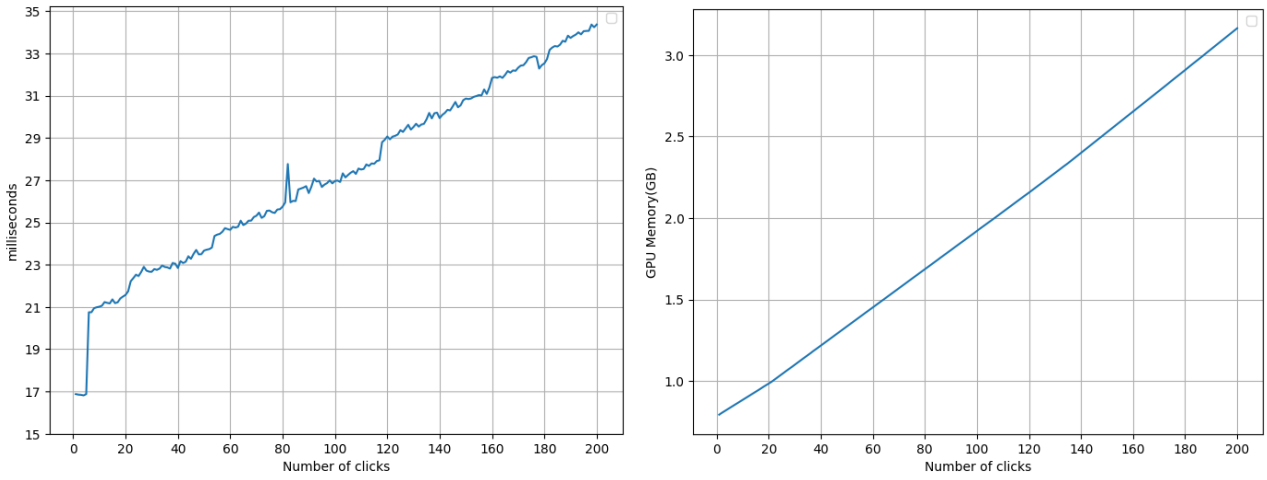
		COCO				SBD				DAVIS17			
Backbone	Strategy	NCI ↓	NFO ↓	NFI ↓	IoU ↑	NCI ↓	NFO ↓	NFI ↓	IoU ↑	NCI ↓	NFO ↓	NFI ↓	IoU ↑
Resnet50	best	6.20	15690	2508	80.9	2.87	677	352	90.0	3.42	572	380	86.9
Resnet50	random	6.13	13554	2461	84.4	2.81	559	329	90.4	3.39	580	375	87.3
Resnet50	worst	6.09	20224	2447	82.6	2.78	870	324	90.2	3.36	773	375	86.2
Resnet50	max-distance	6.82	14786	2890	85.2	3.24	762	471	90.7	3.57	627	405	87.5
Resnet50	round-robin	6.51	15534	2501	83.4	3.50	620	335	90.3	4.07	609	373	87.1
Resnet50	worst with limit	6.09	13249	2444	84.1	2.79	541	326	90.4	3.36	570	375	87.2
Resnet50	mean	6.31	15506	2542	83.4	3.00	671	356	90.3	3.53	622	380	87.0
Resnet50	std	0.30	2519	173	1.5	0.30	126	57	0.23	0.28	77	12	0.45
Segf-B0	best	6.13	15219	2485	81.3	2.83	655	342	90.2	3.29	546	364	87.5
Segf-B0	random	6.04	12986	2431	84.9	2.76	528	313	90.6	3.27	549	356	87.9
Segf-B0	worst	6.02	19758	2414	83.0	2.75	841	315	90.3	3.25	707	354	87.1
Segf-B0	max-distance	6.79	14588	2885	85.5	3.18	735	441	90.8	3.42	592	388	88.2
Segf-B0	round-robin	6.42	14608	2452	84.0	3.47	609	339	90.5	3.95	573	362	87.8
Segf-B0	worst with limit	6.03	12745	2425	84.6	2.75	519	320	90.5	3.25	526	354	87.9
Segf-B0	mean	6.24	14984	2515	83.8	2.96	648	345	90.5	3.40	582	363	87.7
Segf-B0	std	0.31	2536	183	1.5	0.30	124	48	0.21	0.27	65	13	0.38
hrnet32	best	6.14	15092	2506	81.5	2.81	640	347	90.2	3.23	539	353	87.3
hrnet32	random	6.02	12547	2417	85.1	2.74	515	316	90.6	3.20	541	347	87.7
hrnet32	worst	5.99	19419	2410	83.3	2.72	819	314	90.4	3.18	700	345	86.9
hrnet32	max-distance	6.76	14252	2850	85.6	3.16	721	441	90.8	3.35	583	375	87.9
hrnet32	round-robin	6.42	14279	2467	84.2	3.45	612	340	90.4	3.88	566	349	87.5
hrnet32	worst with limit	6.000	12191	2414	84.7	2.72	496	311	90.5	3.18	528	347	87.6
hrnet32	mean	6.22	14630	2511	84.1	2.93	634	345	90.5	3.34	576	353	87.5
hrnet32	std	0.31	2596	170	1.5	0.31	123	49	0.20	0.27	64	11	0.35
Swin-T	best	6.07	14853	2460	81.8	2.75	624	327	90.3	3.20	501	348	87.7
Swin-T	random	6.00	12710	2401	85.1	2.69	510	303	90.7	3.16	514	338	88.0
Swin-T	worst	5.94	19309	2369	83.4	2.68	798	300	90.5	3.16	704	341	87.1
Swin-T	max-distance	6.74	14277	2854	85.7	3.15	737	449	90.9	3.33	550	370	88.2
Swin-T	round-robin	6.37	14268	2438	84.3	3.40	595	325	90.6	3.84	534	339	87.9
Swin-T	worst with limit	5.9	12436	2390	84.8	2.68	492	302	90.7	3.16	503	340	88.0
Swin-T	mean	6.18	14642	2485	84.2	2.89	626	334	90.6	3.31	551	346	87.8
Swin-T	std	0.32	2478	184	1.4	0.31	122	57	0.20	0.27	77	12	0.39
Swin-L	best	5.80	13876	2305	82.4	2.47	497	266	90.7	3.06	483	330	88.4
Swin-L	random	5.70	11958	**2242**	85.3	2.42	428	**249**	91.0	3.03	479	320	88.8
Swin-L	worst	5.66	18133	2242	83.7	2.41	671	251	90.8	2.99	620	314	88.1
Swin-L	max-distance	6.53	13107	2725	**86.4**	2.87	594	371	**91.2**	3.11	498	340	**88.9**
Swin-L	round-robin	6.11	13639	2305	84.5	3.12	490	261	90.9	3.70	504	320	88.6
Swin-L	worst with limit	**5.67**	**11565**	2245	85.0	**2.41**	**422**	250	90.9	**2.99**	**461**	**315**	88.7
Swin-L	mean	5.91	13713	2344	84.5	2.62	517	275	90.9	3.15	507	323	88.6
Swin-L	std	0.35	2351	189	1.4	0.30	98	48	0.17	0.27	57	10	0.29

Table I: 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.

port 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 II, they are still important for reducing the overall number of clicks especially in the single-instance setting (ref Tab. III).

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 1, we analyze the impact of such a growing query



(a) Runtime Analysis

(b) Memory Analysis

Figure 1: 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)	**6.06**	**12997**	**2458**	**2.72**	**557**	**329**	**3.20**	**541**	**356**
- static background queries	6.18	14436	2548	2.79	639	354	3.33	625	393
- Transformer decoder	6.34	14504	2652	2.90	657	384	3.24	582	371
- temporal positional encoding	6.42	14729	2704	2.94	682	402	3.35	617	388
- spatial positional encoding	6.32	14506	2632	2.90	671	395	3.24	569	370
- spatio-temporal positional encoding	6.23	13552	2569	2.86	608	376	3.34	587	379

Table II: 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 [6]		Berkeley [3]		SBD [1]		COCO MVal		DAVIS [5]	
	@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓	@85 ↓	@90 ↓
DynaMITe (Swin-T)	1.56	1.64	1.38	2.06	**3.83**	**6.39**	**2.27**	3.28	**3.75**	5.19
- static background queries	1.64	1.68	1.35	**1.87**	3.92	6.51	2.31	**3.21**	3.84	**5.15**
- Transformer decoder	1.64	1.76	**1.32**	2.28	4.18	6.89	2.40	3.50	3.77	5.33
- temporal positional encoding	**1.52**	1.64	1.51	2.27	4.17	6.89	2.42	3.48	4.04	5.43
- spatial positional encoding	1.76	1.86	1.36	2.41	4.19	6.89	2.44	3.45	3.84	5.28
- spatio-temporal positional encoding	1.56	**1.62**	1.34	2.10	3.99	6.63	2.28	3.24	4.06	5.38

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

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. 1a and Fig. 1b, 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. 1 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. 2 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 in-

stances 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 [7] 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 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. 3, 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. There are also some interesting failure cases such as the one shown 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.

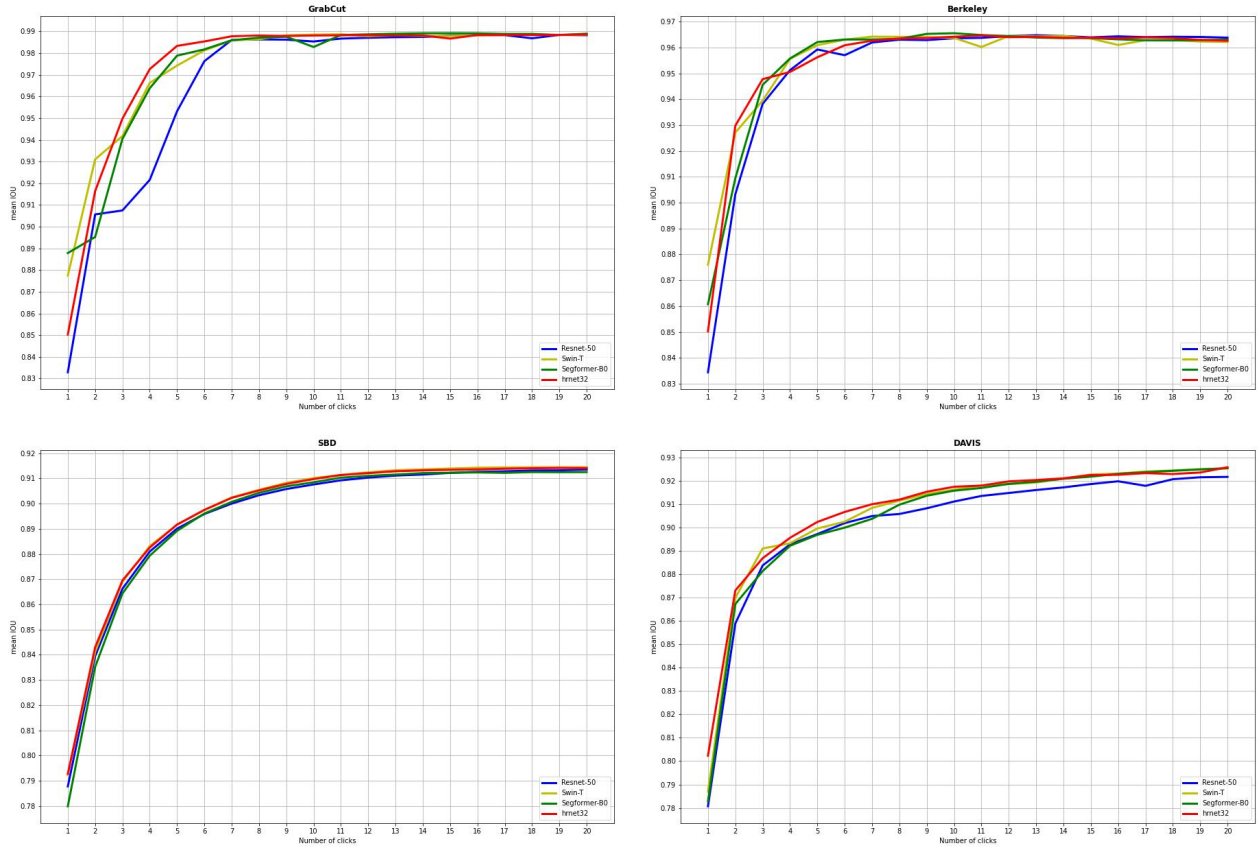


Figure 2: IoU vs. number of clicks for multiple single-instance datasets.

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

Figure 3: 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 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 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.

References

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

[2] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 1

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

[4] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016. 1

[5] 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

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

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