Promptless DCIS segmentation in first post-contrast DCE-MRI image using MedSAM

Aaron Sossin
Kontos Lab
October 24th, 2024

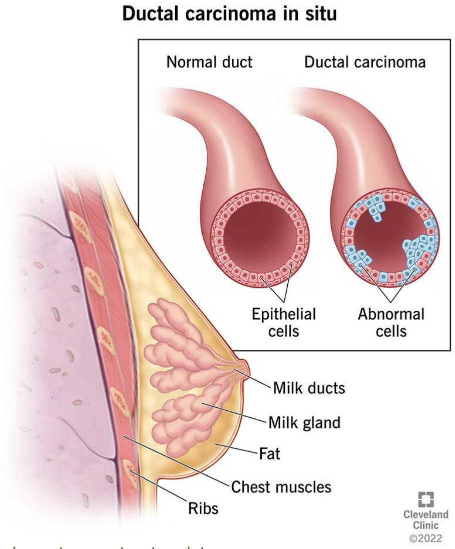
Overview

- Background + Literature Review (10min)
- Dataset (5min)
- Methods (5min)
- Results (10min)
- Future work (5min)
- Conclusion

Dual Carcinoma in Situ (DCIS)

DCIS is a non-invasive form of breast cancer. Abnormal cells are stuck in lining of the breast ducts, and have not yet spread to surrounding tissue.

In 2011, 200k new cases of invasive breast cancer were reported revealing a great need for early diagnosis



https://my.clevelandclinic.org/health/diseases/17869-ductal-carcinoma-in-situ-dcis

Why do we need automatic DCIS segmentation?

1. Since DCIS is not yet invasive, it is **hard to diagnose**, and does not always pose symptoms.
2. Identifying **low-risk DCIS** vs. **non-low-risk DCIS** is crucial in deciding treatment options
3. Treatment typically involves surgery (lumpectomy or mastectomy) and possibly radiation therapy. Over-treatment is a big cause for concern
4. Segmentation unlocks downstream radiomic analyses (such as work with Arumina)

Why do we need DCIS segmentation?

1. Since DCIS is not yet invasive, it is **hard to diagnose**, and does not necessarily pose symptoms.
2. Identifying **low-risk DCIS** vs. **non-low-risk DCIS** is crucial in deciding treatment options
3. Treatment typically involves surgery (lumpectomy or mastectomy) and possibly radiation therapy. Over-treatment is a big issue here.
4. Segmentation unlocks downstream radiomic analyses

Why do we need DCIS segmentation?

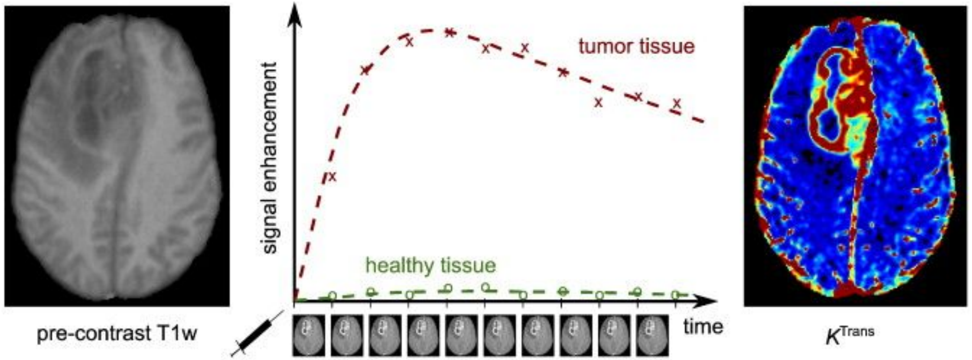
1. Since DCIS is not yet invasive, it is **hard to diagnose**, and does not necessarily pose symptoms.
2. Identifying **low-risk DCIS** vs. **non-low-risk DCIS** is crucial in deciding treatment options
3. Treatment typically involves surgery (lumpectomy or mastectomy) and possibly radiation therapy. Over-treatment is a big issue here.
4. Segmentation unlocks downstream radiomic analyses

Why do we need DCIS segmentation?

1. Since DCIS is not yet invasive, it is **hard to diagnose**, and does not necessarily pose symptoms.
2. Identifying **low-risk DCIS** vs. **non-low-risk DCIS** is crucial in deciding treatment options
3. Treatment typically involves surgery (lumpectomy or mastectomy) and possibly radiation therapy. Over-treatment is a big issue here.
4. Segmentation unlocks downstream radiomic analyses (like work with Arunima)

Why DCE-MRI (Dynamic Contrast Enhanced MRI) ?

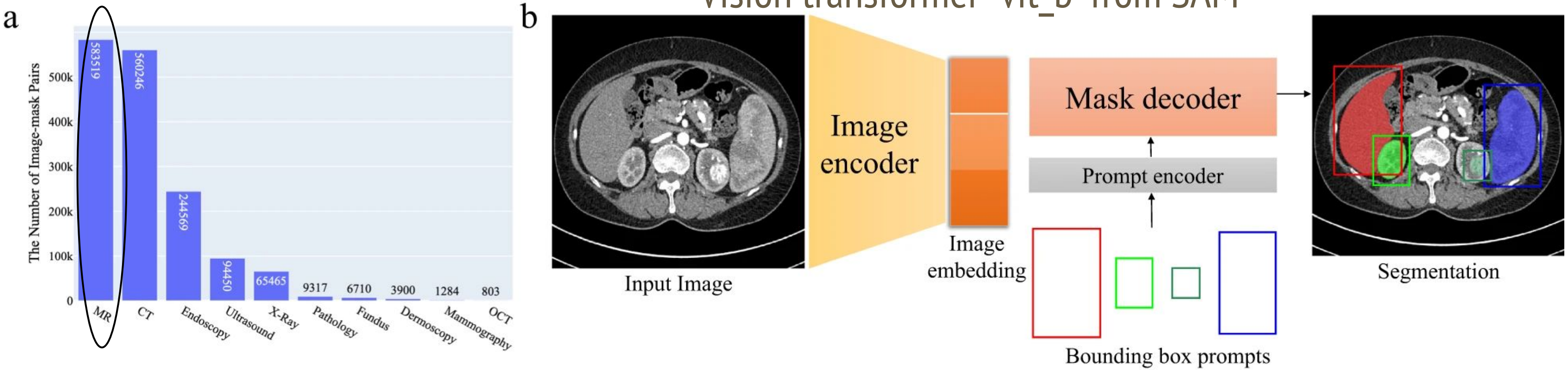
- To create a DCE-MRI dataset, a vascular contrast agent is injected intravenously, and a time series of volumetric images is made of the breast.
- Tumors = greater density of blood vessels = greater contrast faster
- As a result, voxels within a tumor in DCE-MRI show a rapid increase in signal intensity and a subsequent decrease over time, while voxels within the healthy parenchyma show a gradual increase in signal intensity.



Why MedSAM?

The “Segment Anything Model” (SAM) is highly successful foundation model for segmenting any type of image, however, it is famously poor at medical imaging tasks

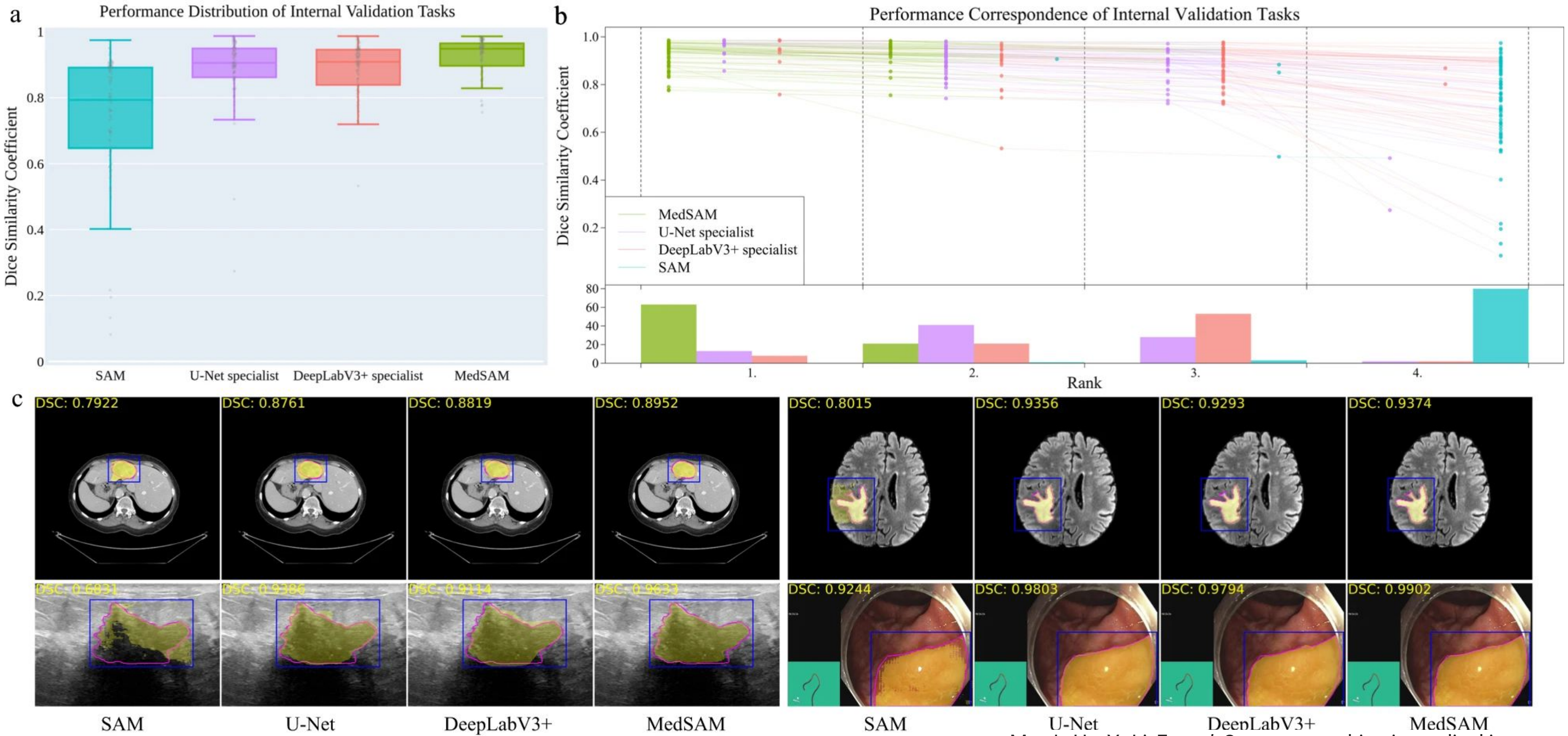
Vision transformer “vit_b” from SAM



Trained on 1,570,263 images

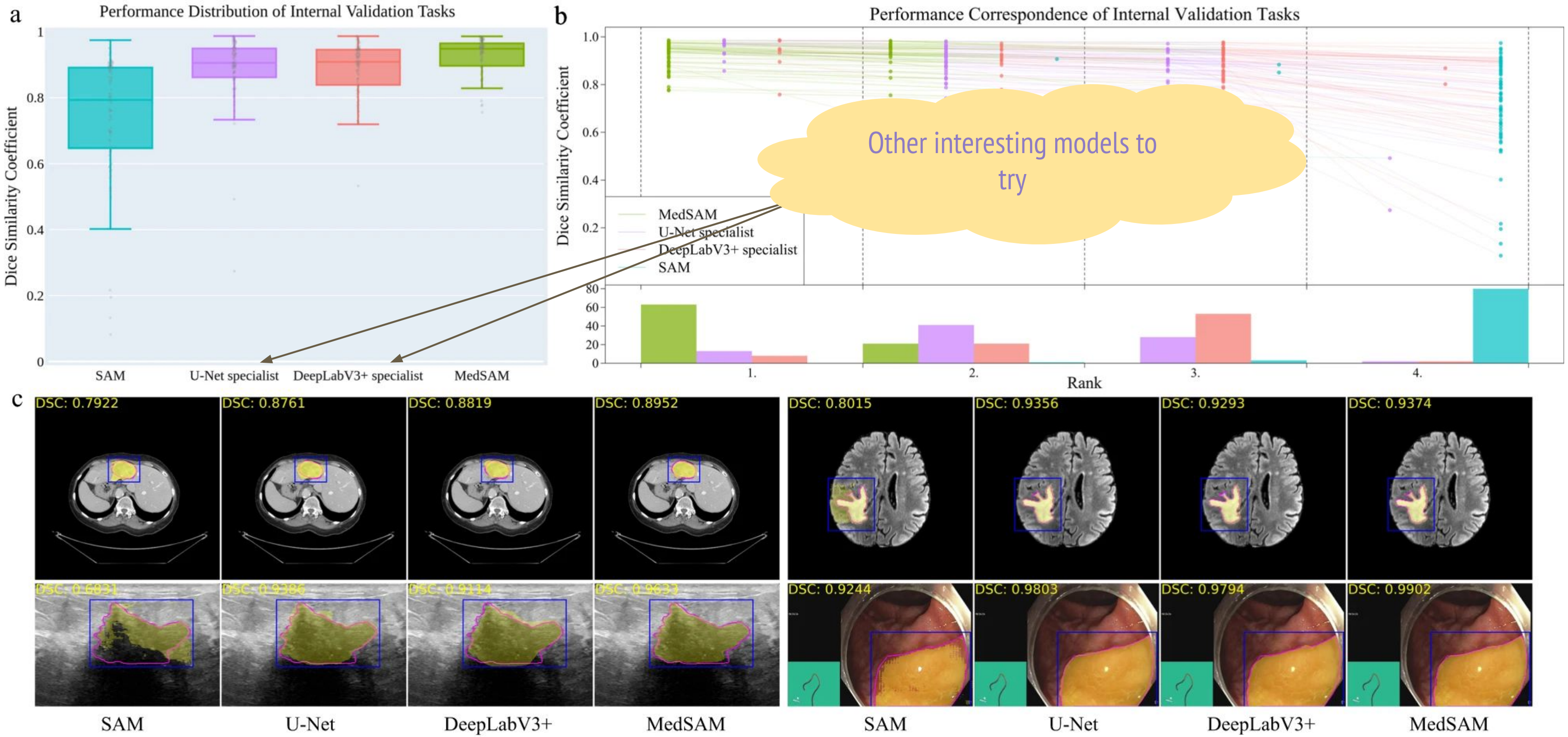
Ma, J., He, Y., Li, F. *et al.* Segment anything in medical images. *Nat Commun* 15, 654 (2024). https://doi.org/10.1038/s41467-024-44824-z

Performs favorably compared to SAM, and Unets



Ma, J., He, Y., Li, F. *et al.* Segment anything in medical images. *Nat Commun* 1
654 (2024). https://doi.org/10.1038/s41467-024-44824-z

Performs favorably compared to SAM, and Unets



Other interesting models to try

Ma, J., He, Y., Li, F. *et al.* Segment anything in medical images. *Nat Commun*
654 (2024). https://doi.org/10.1038/s41467-024-44824-z

Literature review

The majority of DCIS oriented studies focus on distinguishing low-grade DCIS from non-low-grade DCIS using radiomics

Paper	Segmentation	Automatic	NNs	Breast	DCIS	Multiple post-contrast images	2D	3D	MRI	Foundation models
Seth et al.	x	x	x	x	x		x			
SLATS	x			x	x	x		x	x	
DA-DSUnet	x	x	x				x		x	
3D AGSE-VNet	x	x	x					x	x	
Mori et al				x	x	x		x	x	
Miceli et al				x	x	x		x	x	
MA-SAM	x	x	x			x		x	x	x (sam)
RETfound		x	x				x			x (custom)
Our Approach	**x**	**x**	**x**	**x**	**x**		**x**		**x**	**x (medsam)**

How have other groups dealt with DCIS segmentation?

The majority of DCIS oriented studies focus on distinguishing low-grade DCIS from non-low-grade DCIS using radiomics

Segmentation from histopathological slides

DCE-MRI segmentation!

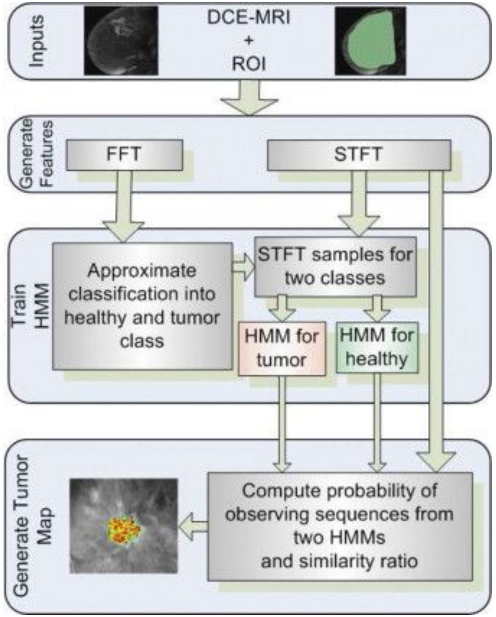
Escalation prediction

Escalation Prediction

Paper	Segment ation	Automatic	NNs	Breast	DCIS	Multiple post-contrast images	2D	3D	MRI	Foundation models
Seth et al.	x	x	x	x	x		x			
SLATS	x			x	x	x		x	x	
DA-DSUnet	x	x	x				x		x	
3D AGSE-VNet	x	x	x					x	x	
Mori et al				x	x	x		x	x	
Miceli et al				x	x	x		x	x	
MA-SAM	x	x	x			x		x		
RETfound		x	x				x			x (custom)
Our Approach	**x**	**x**	**x**	**x**	**x**		**x**		**x**	**x (medsam)**

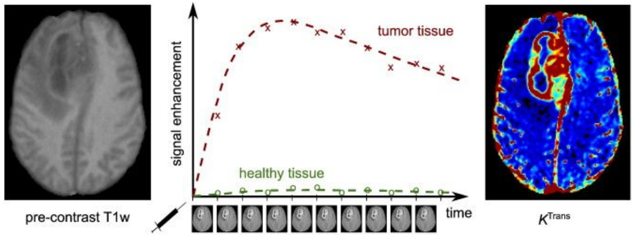
I have only found **one** other paper that segments DCIS from DCE-MRI

How does SLATs (Statistical Learning Algorithm for Tumor Segmentation) work?


Inputs
DCE-MRI
+
ROI
Generate Features
FFT
STFT
Train HMM
Approximate classification into healthy and tumor class
STFT samples for two classes
HMM for tumor
HMM for healthy
Generate Tumor Map
Compute probability of observing sequences from two HMMs and similarity ratio

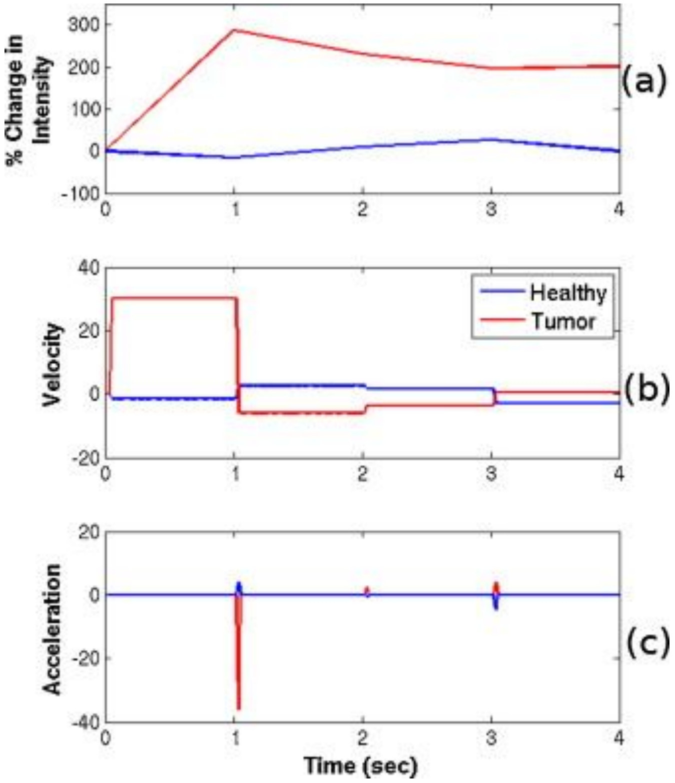
*"The SLATS algorithm has been trained to identify voxels belonging to the tumor class using the **time–intensity curve**, first and second derivatives of the intensity curves ("velocity" and "acceleration" respectively) and a composite vector consisting of a concatenation of the intensity, velocity and acceleration vectors"*

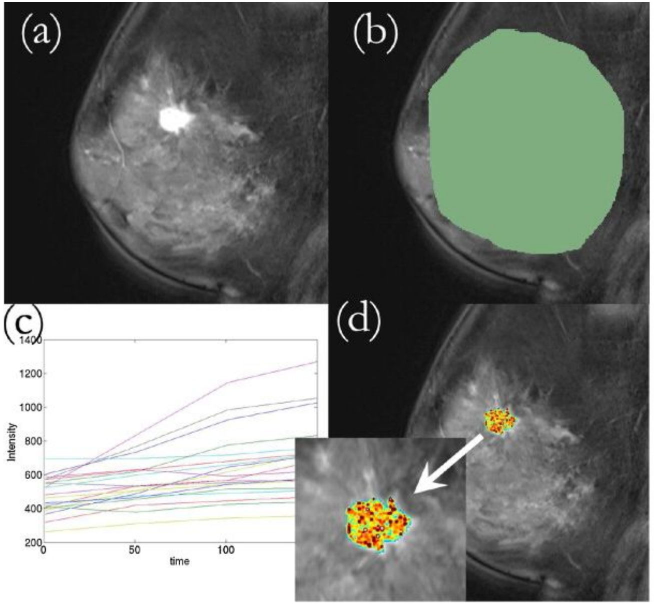
recall



Jagadeesan Jayender, Eva Gombos, Sona Chikarmane, Donnette Dabydeen, Ferenc A. Jolesz, Kirby G. Vosburgh, Statistical Learning Algorithm for in situ and invasive breast carcinoma segmentation, Computerized Medical Imaging and Graphics, Volume 37, Issue 4, 2013,

SLATs treats each voxel as *independent*





Download: Download high-res image (681KB)

Download: Download full-size image

Fig. 3. Workflow of the SLATS. (a) DCE-MRI loaded into 3D Slicer, (b) ROI delineated, (c) time–intensity curves obtained from all voxels under ROI and provided to SLATS, and (d) tumor map is generated.

SLATs results using 4 post-contrast images

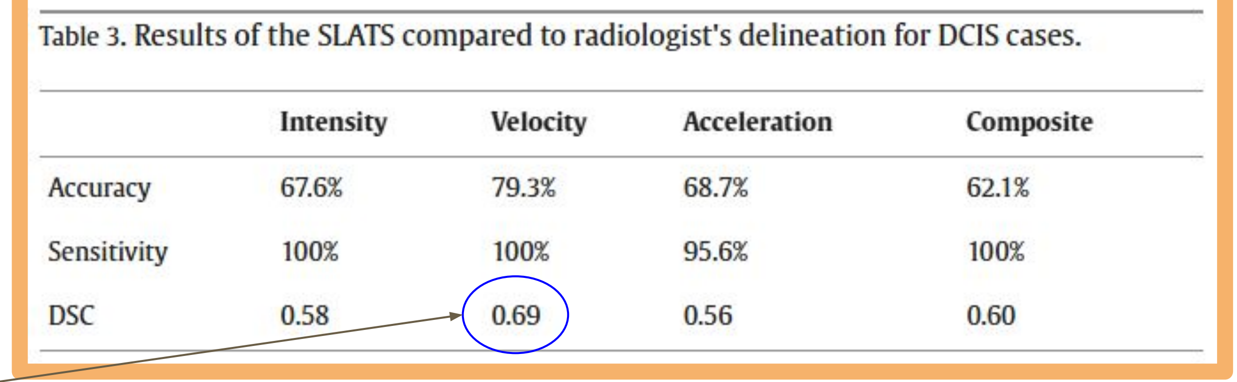


Table 3. Results of the SLATS compared to radiologist's delineation for DCIS cases.

	Intensity	Velocity	Acceleration	Composite
Accuracy	67.6%	79.3%	68.7%	62.1%
Sensitivity	100%	100%	95.6%	100%
DSC	0.58	0.69	0.56	0.60

Our ultimate comparison

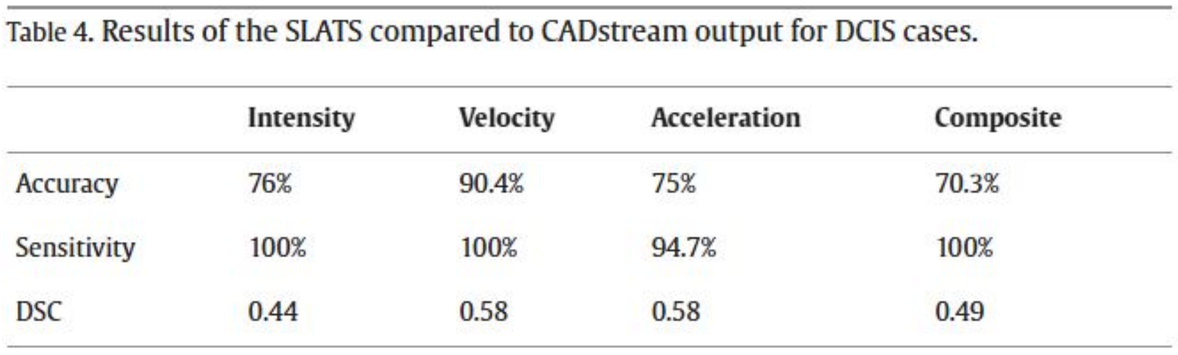


Table 4. Results of the SLATS compared to CADstream output for DCIS cases.

	Intensity	Velocity	Acceleration	Composite
Accuracy	76%	90.4%	75%	70.3%
Sensitivity	100%	100%	94.7%	100%
DSC	0.44	0.58	0.58	0.49

It is easier for SLATs to predict IDC than DCIS

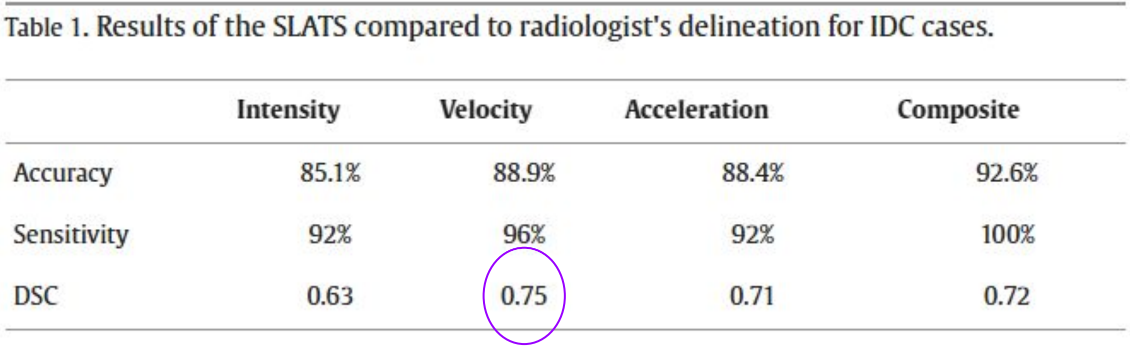


Table 1. Results of the SLATS compared to radiologist's delineation for IDC cases.

	Intensity	Velocity	Acceleration	Composite
Accuracy	85.1%	88.9%	88.4%	92.6%
Sensitivity	92%	96%	92%	100%
DSC	0.63	0.75	0.71	0.72

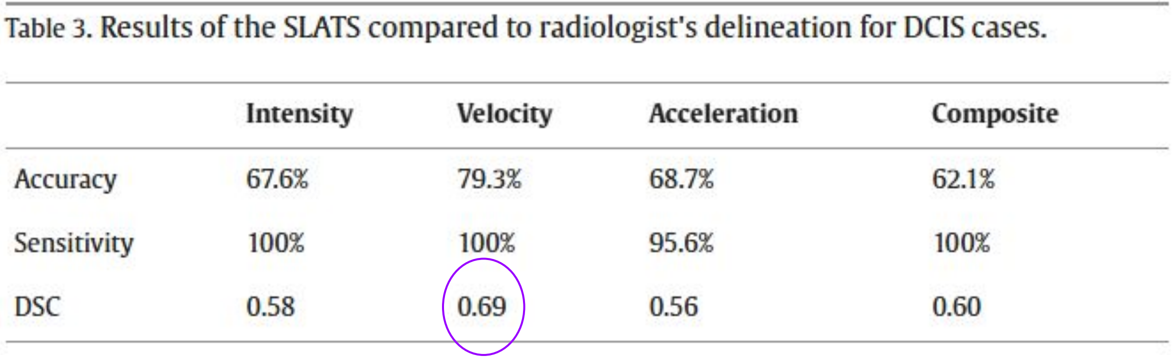


Table 3. Results of the SLATS compared to radiologist's delineation for DCIS cases.

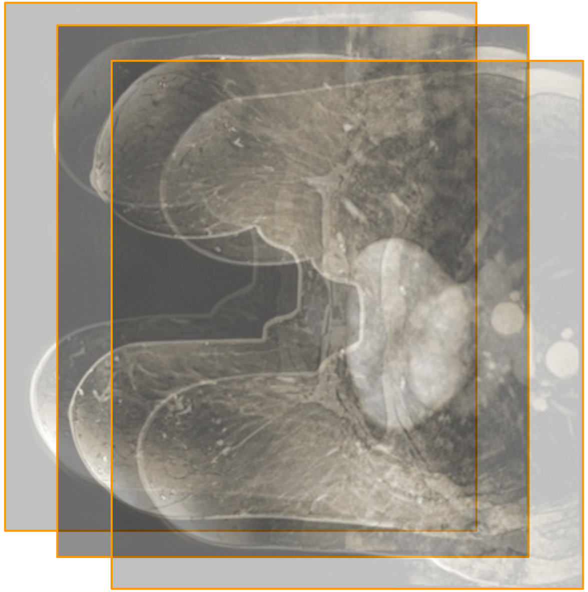
	Intensity	Velocity	Acceleration	Composite
Accuracy	67.6%	79.3%	68.7%	62.1%
Sensitivity	100%	100%	95.6%	100%
DSC	0.58	0.69	0.56	0.60

Background recap

- **Automatic DCIS segmentation** is an important, and solvable unmet healthcare need
- There is reason to believe **MedSAM** foundation model will optimize results in this task
- Only one other method (that I'm aware of) **(SLATs)** has segmented DCIS from DCE-MRI, in 2013 by jayender et al., and without the use of AI.
- Texture, shape, volume, and contrast-based features have all been shown to be *relevant* towards DCIS understanding.

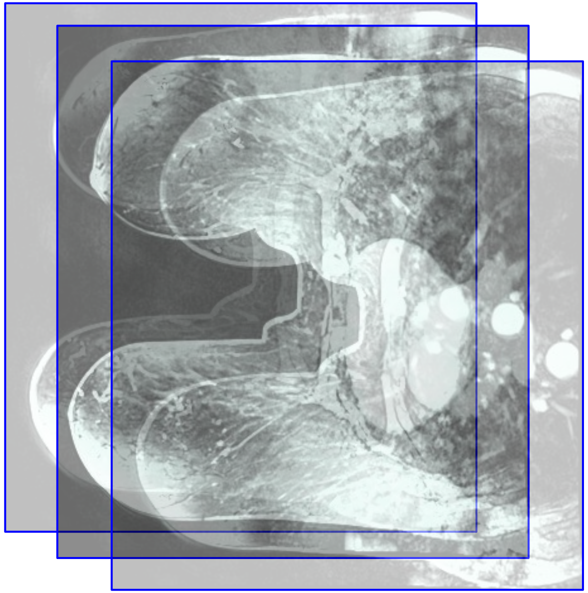
Our approach is *difficult* considering we are relying solely on texture, shape, and volume features instead of time-intensity curve

DCE-MRI dataset of 290 patients



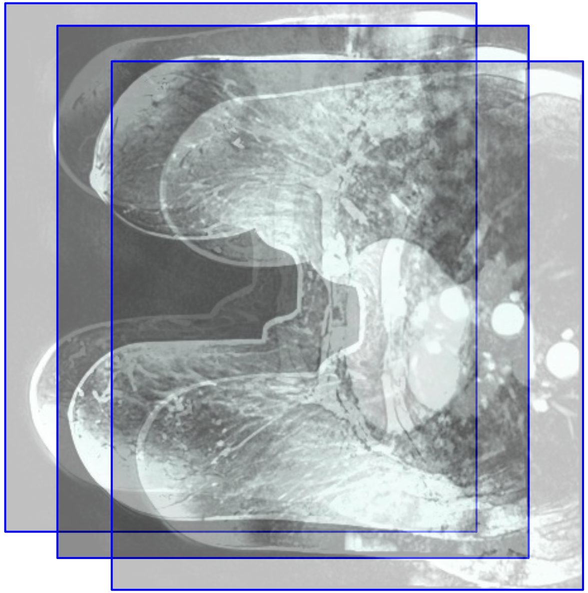
One breast MRI ~ 60 slices

Pre-contrast image



First post-contrast image

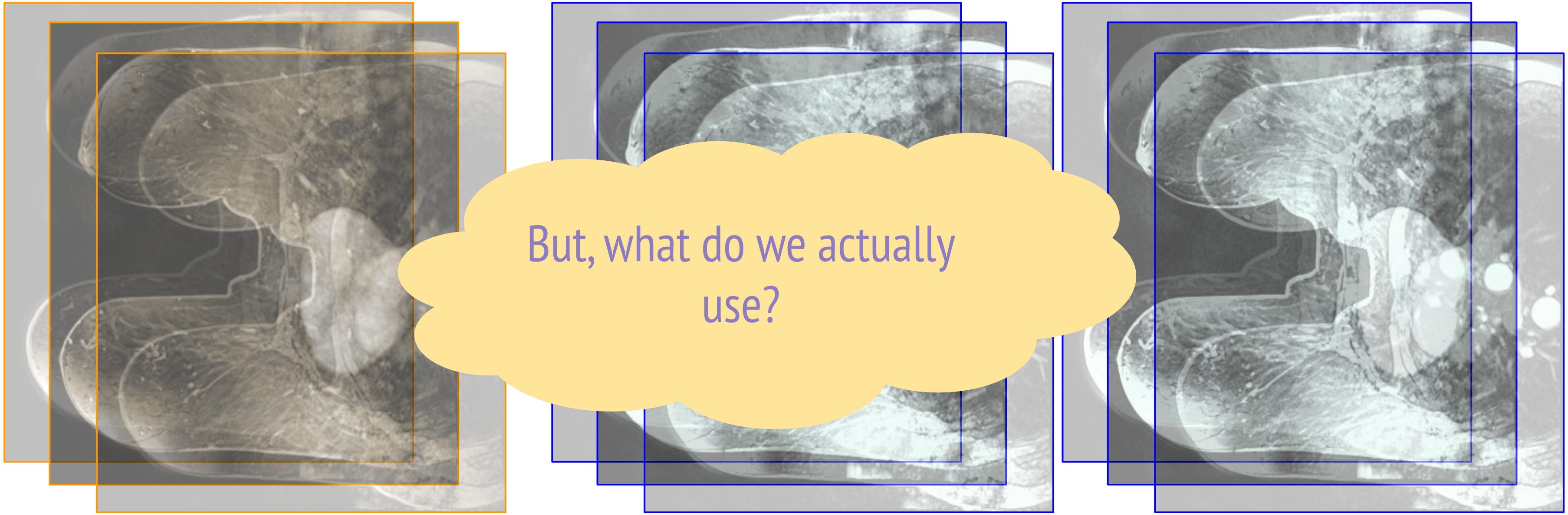
***



Last post-contrast image

There may be more post-contrast images out there, just haven’t seen more on cluster

DCE-MRI dataset of 290 patients



But, what do we actually use?

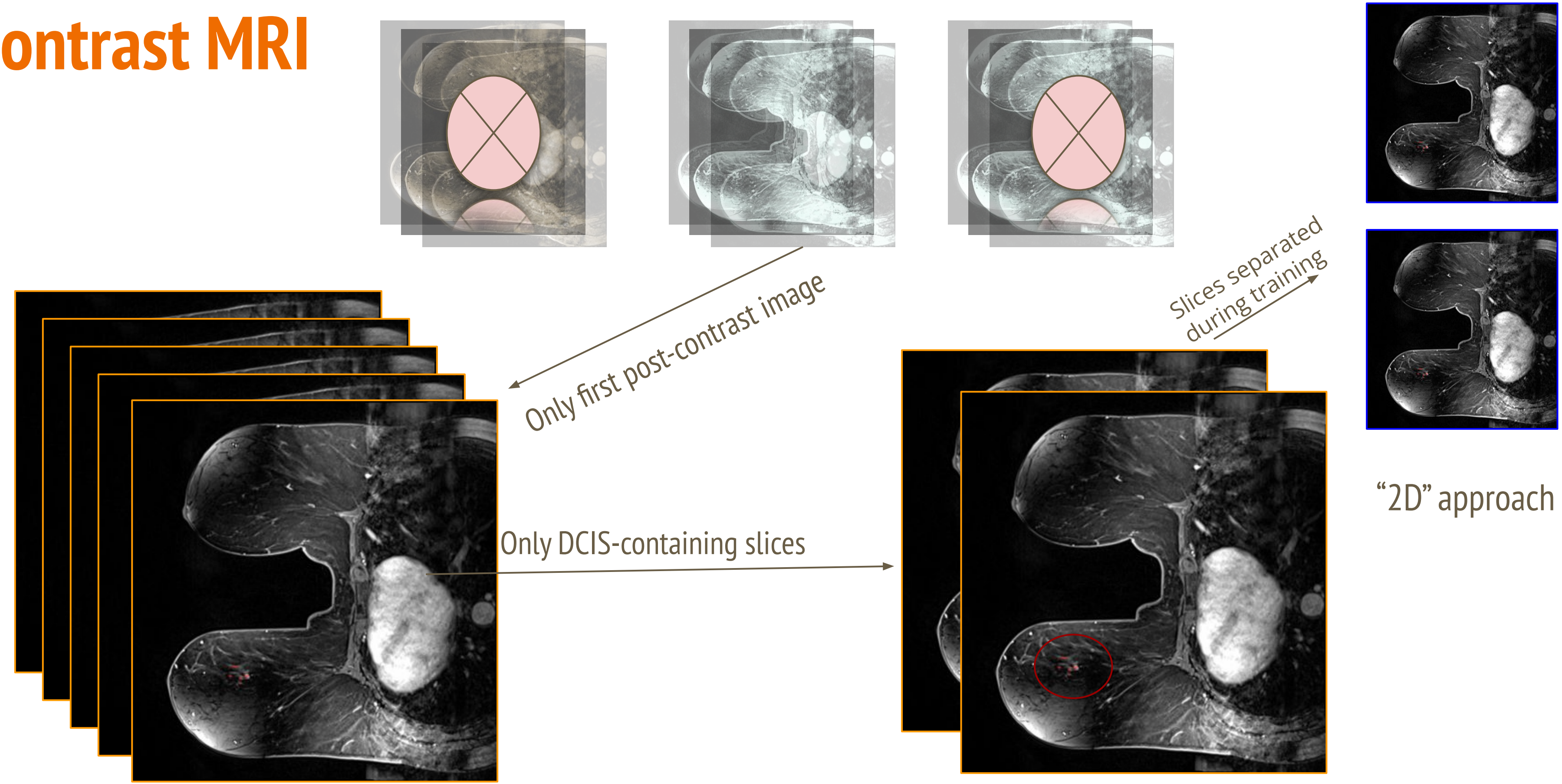
One breast MRI ~ 60 slices

Pre-contrast image

First post-contrast image

Last post-contrast image

Currently, we take DCIS-containing images from first post contrast MRI



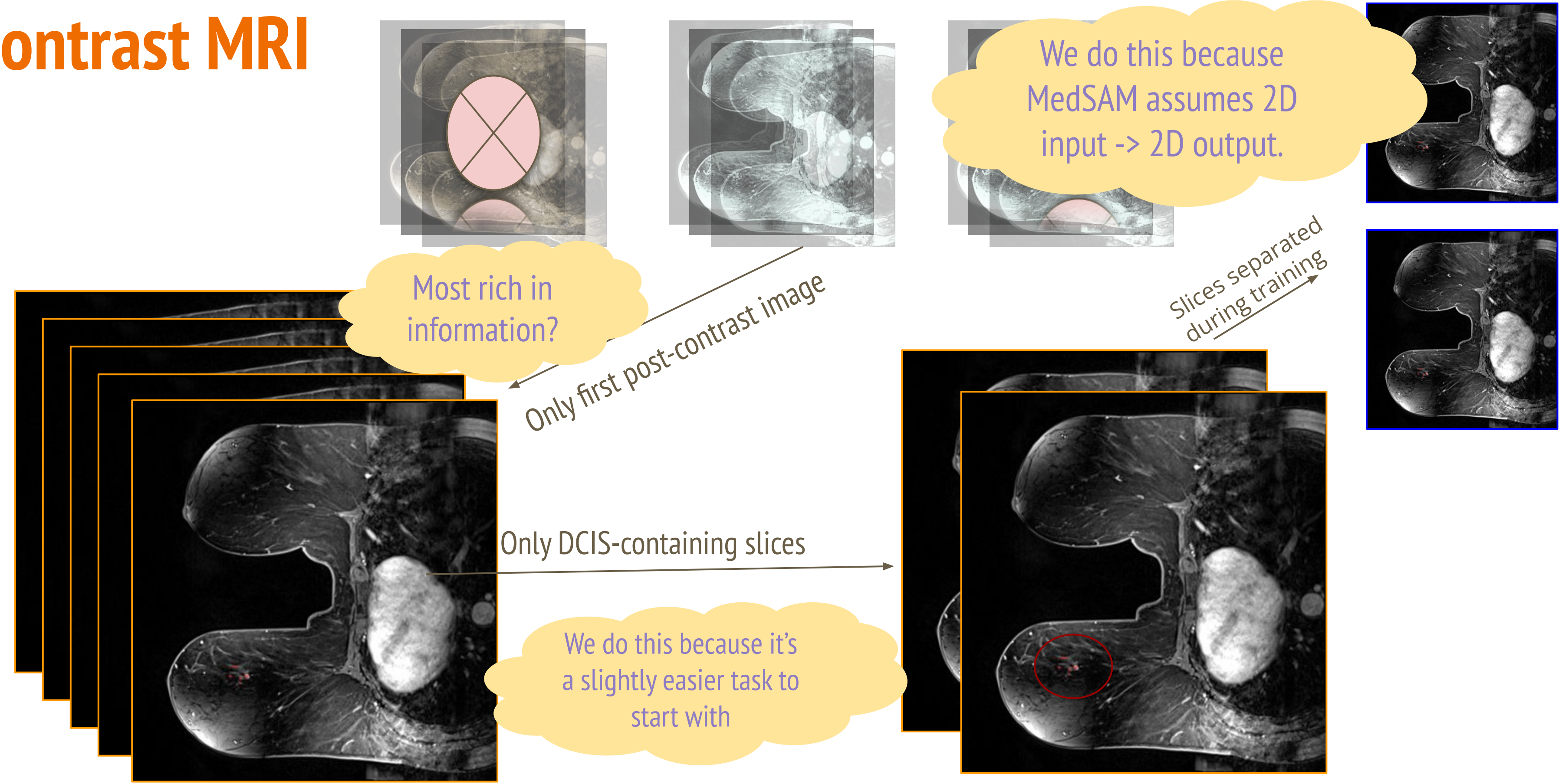
Only first post-contrast image

Only DCIS-containing slices

Slices separated during training

“2D” approach

Currently, we take DCIS-containing images from first post contrast MRI



Most rich in information?

Only first post-contrast image

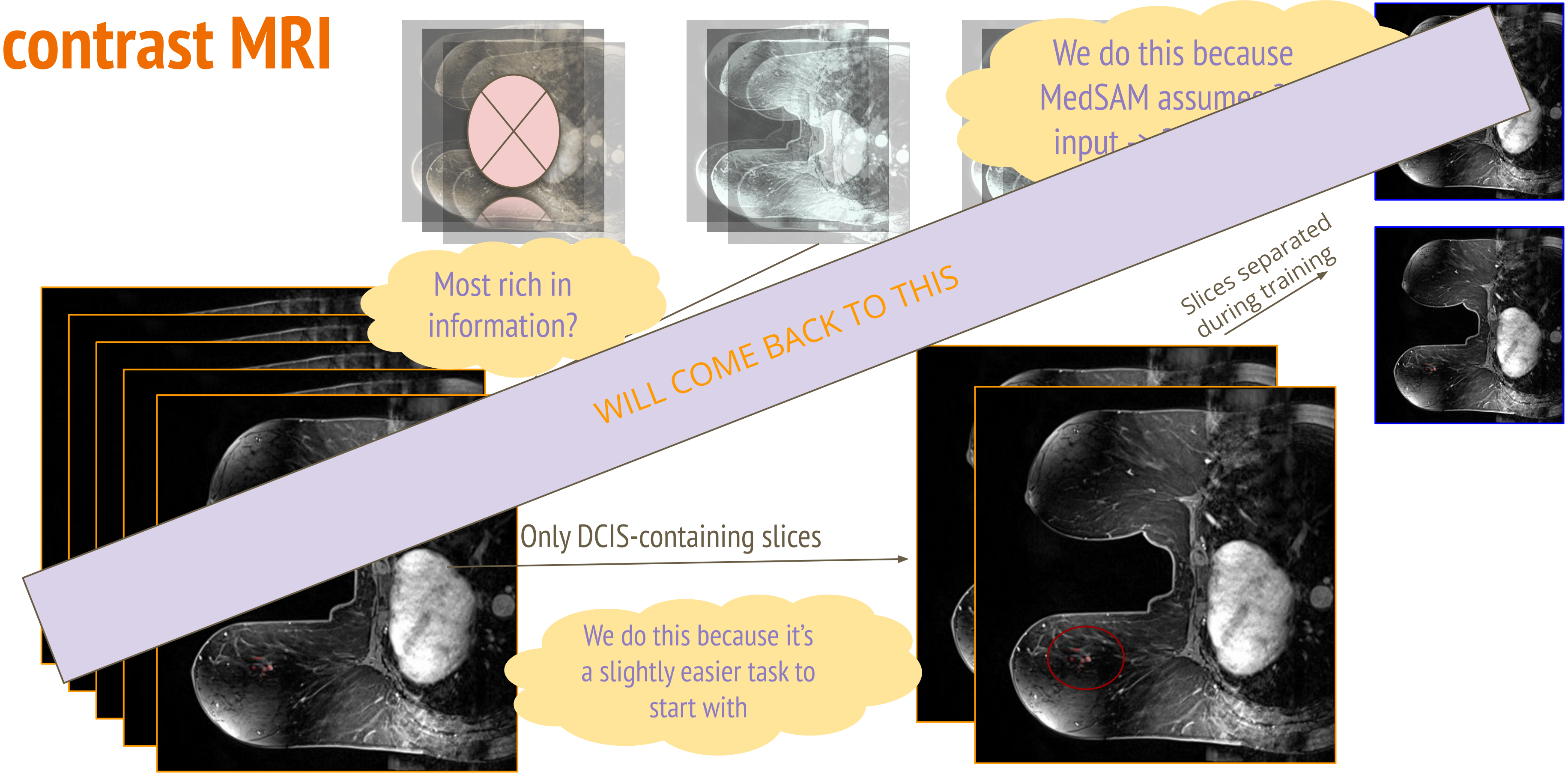
We do this because MedSAM assumes 2D input -> 2D output.

Only DCIS-containing slices

We do this because it’s a slightly easier task to start with

Slices separated during training

Currently, we take DCIS-containing images from first post contrast MRI



We do this because
MedSAM assumes
input ->

Most rich in information?

WILL COME BACK TO THIS

Slices separated during training

Only DCIS-containing slices

We do this because it’s a slightly easier task to start with

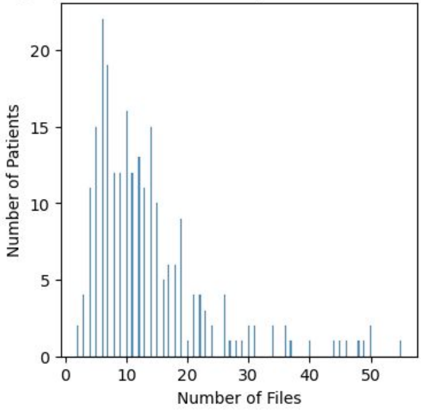
Dataset

- E4112 trial from ECOG-ACRIN (East Coast Oncology)
  - Private source

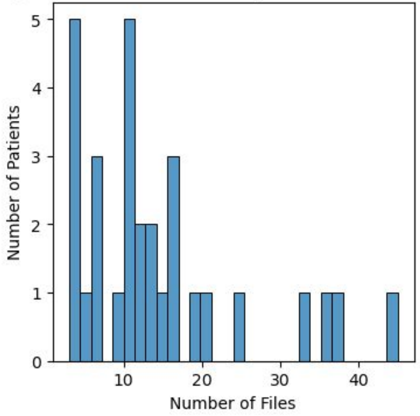
- 80%/10%/10% train/val/test split (same as MedSAM)
  - 3239/433/433 slices respectively

- Each slice has associated DCIS map **labelled by radiologists**

Histogram of Number of Files per Patient in Training Set

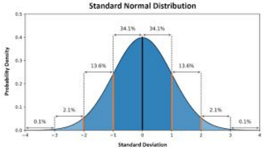


Histogram of Number of Files per Patient in Testing Set



Pre-processing steps from MedSAM

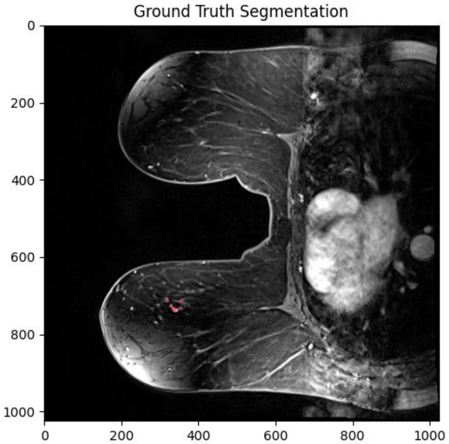
1. **Rotate** images to all have same orientation
2. *Numpy **clip*** between 50th and 99.5th percentile of pixels
3. **Normalize** each slice w.r.t itself between 0,255
4. Convert to **‘int’**
5. **Resize** image to (1024,1024,3)
6. Re-**normalize**



Kalina pre-processed everything, I have not touched this yet

Pre-processing steps

1. Rotate images to all have same orientation
2. ***Numpy clip* between 50th and 99.5th percentile of pixels**
3. Normalize each slice w.r.t itself between 0,255
4. Conv
5. Resi 24,3)
6. Re-n



Since black background isn't quite 50% of image, we may be zero-ing out some breast tissue (maybe doesn't matter, but could we do this in more informed way?)

Pre-processing steps

1. Rotate images to all have same orientation
2. *Numpy clip* between 50th and 99.5th percentile of pixels
3. **Normalize each slice w.r.t itself between 0,255**
4. Convert to ‘int’
5. Resize image to (1024,1024,3)
6. Re-normalize

We can also normalize w.r.t each patient, or entire dataset.
This deserves consideration, and literature review

Pre-processing steps

1. Rotate images to all have same orientation
2. *Numpy clip* between 50th and 99.5th percentile of pixels
3. Normalize each slice w.r.t itself between 0,255
4. **Convert to ‘int’**
5. Resize image to (1024,1024,3)
6. Re-normalize

We are losing information by discretizing our input images, but also gaining memory space.

Pre-processing steps

1. **Remove non-DCIS containing images**
2. Rotate images to all have same orientation
3. *Numpy clip* between 50th and 99.5th percentile of pixels
4. Normalize each slice w.r.t itself between 0,255
5. Convert to ‘int’
6. Resize image to (1024,1024,3)
7. Re-normalize

Will eventually change this in order to create a fully automated approach

Automated and randomized "wide" grid search

Hypothesis**:** Given a wide-enough grid search, MedSAM will successfully segment DCIS in our dataset

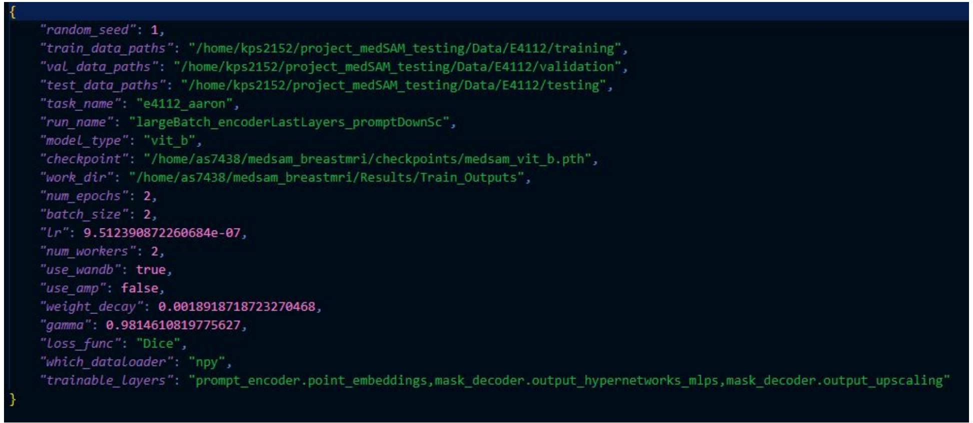
Hyper-parameter grid search for fine-tuning MedSAM

- Epochs (1,125)
- Batch size (1,10)
- Learning rate (1e-3,1e-8)
- Frozen layers (*every layer group can be frozen, with a maximum of 4 groups at a time)*
- Learning rate decay (0.96,1.0)
- Weight decay (0,0.01)
- Loss function (BCE + Dice, Dice, BCE)

Example of automated config creation

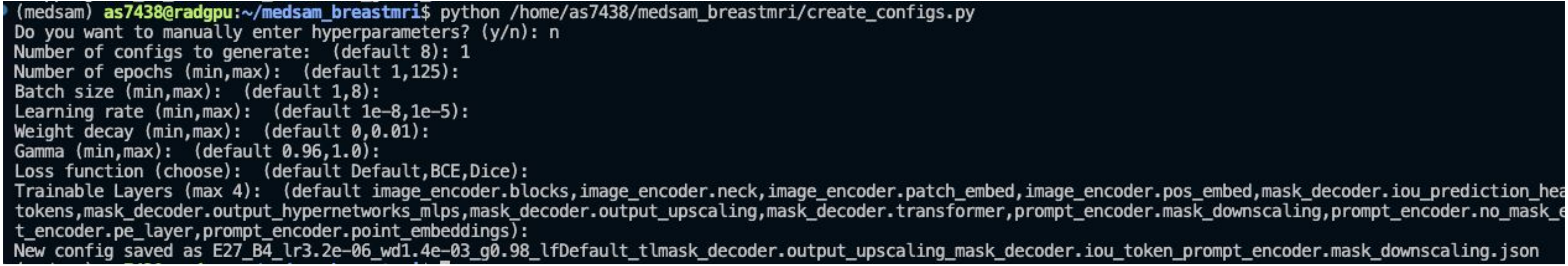
- Creating new random config files is automatic

- Controlled through creation of “config.json” files



```
{
    "random_seed": 1,
    "train_data_paths": "/home/kps2152/project_medSAM_testing/Data/E4112/training",
    "val_data_paths": "/home/kps2152/project_medSAM_testing/Data/E4112/validation",
    "test_data_paths": "/home/kps2152/project_medSAM_testing/Data/E4112/testing",
    "task_name": "e4112_aaron",
    "run_name": "largeBatch_encoderLastLayers_promptDownSc",
    "model_type": "vit_b",
    "checkpoint": "/home/as7438/medsam_breastmri/checkpoints/medsam_vit_b.pth",
    "work_dir": "/home/as7438/medsam_breastmri/Results/Train_Outputs",
    "num_epochs": 2,
    "batch_size": 2,
    "lr": 9.512390872260684e-07,
    "num_workers": 2,
    "use_wandb": true,
    "use_amp": false,
    "weight_decay": 0.0018918718723270468,
    "gamma": 0.9814610819775627,
    "loss_func": "Dice",
    "which_dataloader": "npy",
    "trainable_layers": "prompt_encoder.point_embeddings,mask_decoder.output_hypernetworks_mlps,mask_decoder.output_upscaling"
}
```

Example config file



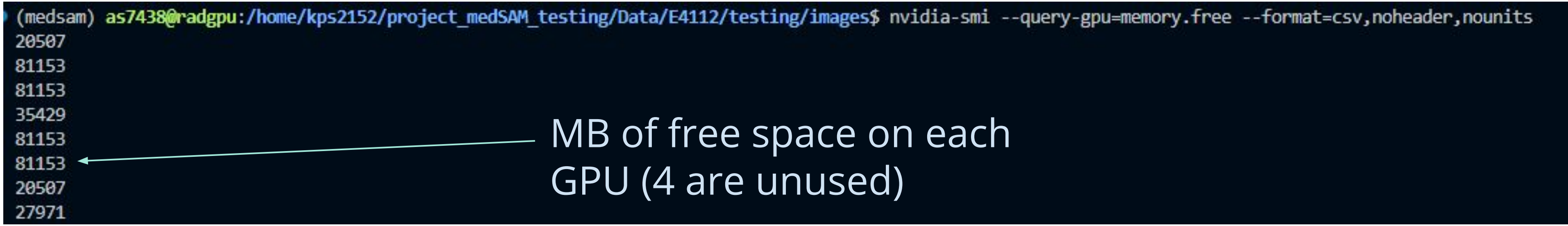
```
(medsam) as7438@radgpu:~/medsam_breastmri$ python /home/as7438/medsam_breastmri/create_configs.py
Do you want to manually enter hyperparameters? (y/n): n
Number of configs to generate:  (default 8): 1
Number of epochs (min,max):  (default 1,125):
Batch size (min,max):  (default 1,8):
Learning rate (min,max):  (default 1e-8,1e-5):
Weight decay (min,max):  (default 0,0.01):
Gamma (min,max):  (default 0.96,1.0):
Loss function (choose):  (default Default,BCE,Dice):
Trainable Layers (max 4):  (default image_encoder.blocks,image_encoder.neck,image_encoder.patch_embed,image_encoder.pos_embed,mask_decoder.iou_prediction_hea
tokens,mask_decoder.output_hypernetworks_mlps,mask_decoder.output_upscaling,mask_decoder.transformer,prompt_encoder.mask_downscaling,prompt_encoder.no_mask_e
t_encoder.pe_layer,prompt_encoder.point_embeddings):
New config saved as E27_B4_lr3.2e-06_wd1.4e-03_g0.98_lfDefault_tlmask_decoder.output_upscaling_mask_decoder.iou_token_prompt_encoder.mask_downscaling.json
```

Example of creating new config files automatically

Training/testing/analyzing workflow

As far as I know, no one else is using this?

Data is all in “RadGPU” server, which contains 8 GPUs each with a maximum of 80GB of memory

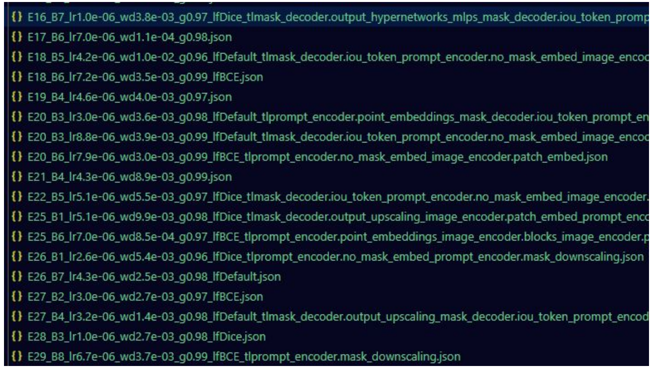


```
(medsam) as7438@radgpu:/home/kps2152/project_medSAM_testing/Data/E4112/testing/images$ nvidia-smi --query-gpu=memory.free --format=csv,noheader,nounits
20507
81153
81153
35429
81153
81153
20507
27971
```

MB of free space on each GPU (4 are unused)

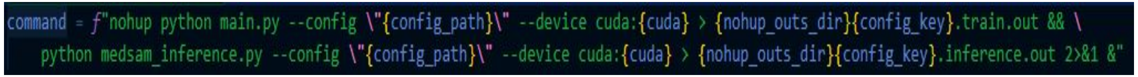
Training/testing/analyzing workflow

1. *Loop through config files*

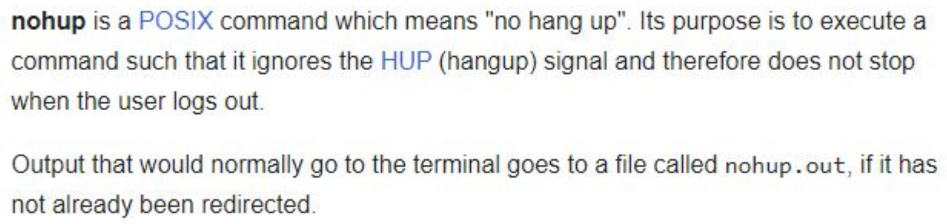


```
{} E16_B7_lr1.0e-06_wd3.8e-03_g0.97_lfDice_tlmask_decoder.output_hypernetworks_mlps_mask_decoder.iou_token_promp
{} E17_B6_lr7.0e-06_wd1.1e-04_g0.98.json
{} E18_B5_lr4.2e-06_wd1.0e-02_g0.96_lfDefault_tlmask_decoder.iou_token_prompt_encoder.no_mask_embed_image_encod
{} E18_B6_lr7.2e-06_wd3.5e-03_g0.99_lfBCE.json
{} E19_B4_lr4.6e-06_wd4.0e-03_g0.97.json
{} E20_B3_lr3.0e-06_wd3.6e-03_g0.98_lfDefault_tlprompt_encoder.point_embeddings_mask_decoder.iou_token_prompt_en
{} E20_B3_lr8.8e-06_wd3.9e-03_g0.99_lfDefault_tlmask_decoder.iou_token_prompt_encoder.no_mask_embed_image_encod
{} E20_B6_lr7.9e-06_wd3.0e-03_g0.99_lfBCE_tlprompt_encoder.no_mask_embed_image_encoder.patch_embed.json
{} E21_B4_lr4.3e-06_wd8.9e-03_g0.99.json
{} E22_B5_lr5.1e-06_wd5.5e-03_g0.97_lfDice_tlmask_decoder.iou_token_prompt_encoder.no_mask_embed_image_encoder.
{} E25_B1_lr5.1e-06_wd9.9e-03_g0.98_lfDice_tlmask_decoder.output_upscaling_image_encoder.patch_embed_prompt_enco
{} E25_B6_lr7.0e-06_wd8.5e-04_g0.97_lfBCE_tlprompt_encoder.point_embeddings_image_encoder.blocks_image_encoder.p
{} E26_B1_lr2.6e-06_wd5.4e-03_g0.96_lfDice_tlprompt_encoder.no_mask_embed_prompt_encoder.mask_downscaling.json
{} E26_B7_lr4.3e-06_wd2.5e-03_g0.98_lfDefault.json
{} E27_B2_lr3.0e-06_wd2.7e-03_g0.97_lfBCE.json
{} E27_B4_lr3.2e-06_wd1.4e-03_g0.98_lfDefault_tlmask_decoder.output_upscaling_mask_decoder.iou_token_prompt_encod
{} E28_B3_lr1.0e-06_wd2.7e-03_g0.98_lfDice.json
{} E29_B8_lr6.7e-06_wd3.7e-03_g0.99_lfBCE_tlprompt_encoder.mask_downscaling.json
```

2. Find GPU with most free space, and if it exceeds 70GB, launch job using **nohup**



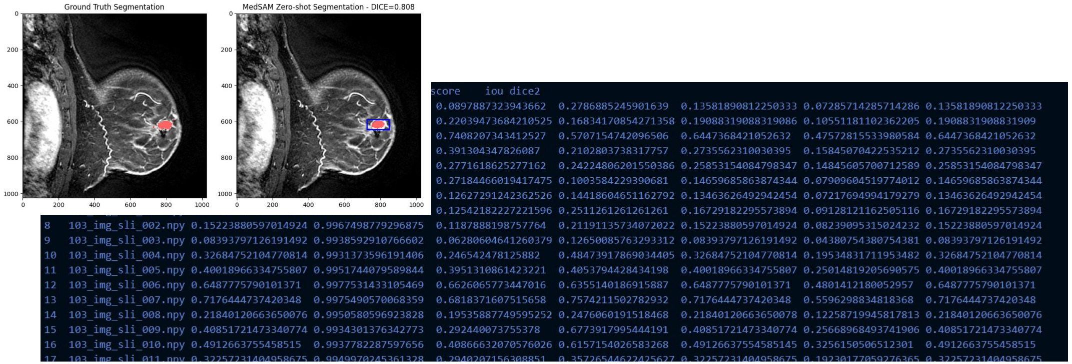
```
command = f"nohup python main.py --config \"{config_path}\" --device cuda:{cuda} > {nohup_outs_dir}{config_key}.train.out && \
    python medsam_inference.py --config \"{config_path}\" --device cuda:{cuda} > {nohup_outs_dir}{config_key}.inference.out 2>&1 &"
```



nohup is a POSIX command which means "no hang up". Its purpose is to execute a command such that it ignores the HUP (hangup) signal and therefore does not stop when the user logs out.

Output that would normally go to the terminal goes to a file called `nohup.out`, if it has not already been redirected.

3. Each configuration gets an output folder, with quantitative scores per image, visualization of results, trained model, and training loss curves

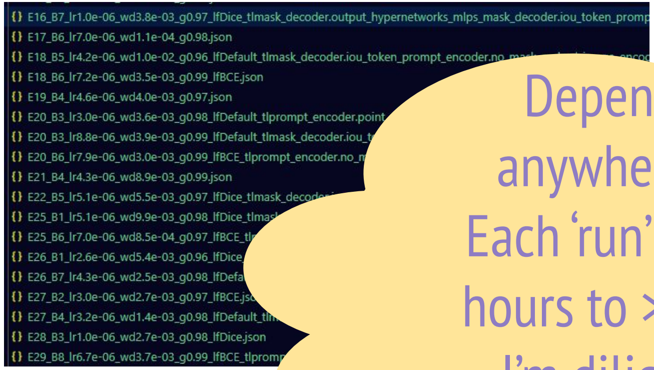


Ground Truth Segmentation
MedSAM Zero-shot Segmentation - DICE=0.808
0
200
400
600
800
1000
0
200
400
600
800
1000

```
                                                  score      iou dice2
                                                   0.0897887323943662  0.2786885245901639  0.13581890812250333 0.07285714285714286 0.13581890812250333
                                                   0.22039473684210525 0.16834170854271358 0.19088319088319086 0.10551181102362205 0.1908831908831909
                                                   0.7408207343412527  0.570715474209650 6 0.6447368421052632  0.47572815533980584 0.6447368421052632
                                                   0.391304347826087   0.2102803738317757  0.2735562310030395  0.15845070422535212 0.2735562310030395
                                                   0.2771618625277162  0.24224806201550386 0.25853154084798347 0.14845605700712589 0.25853154084798347
                                                   0.27184466019417475 0.1003584229390681  0.14659685863874344 0.07909604519774012 0.14659685863874344
                                                   0.12627291242362526 0.14418604651162792 0.13463626492942454 0.07217694994179279 0.13463626492942454
                                                   0.12542182227221596 0.2511261261261261  0.16729182295573894 0.09128121162505116 0.16729182295573894
8   103_img_sli_002.npy 0.15223880597014924 0.9967498779296875  0.1187888198757764  0.21191135734072022 0.15223880597014924 0.08239095315024232 0.15223880597014924
9   103_img_sli_003.npy 0.08393797126191492 0.9938592910766602  0.06280604641260379 0.12650085763293312 0.08393797126191492 0.04380754380754381 0.08393797126191492
10  103_img_sli_004.npy 0.32684752104770814 0.9931373596191406  0.246542478125882   0.48473917869034405 0.32684752104770814 0.19534831711953482 0.32684752104770814
11  103_img_sli_005.npy 0.40018966334755807 0.9951744079589844  0.3951310861423221  0.4053794428434198  0.40018966334755807 0.25014819205690575 0.40018966334755807
12  103_img_sli_006.npy 0.6487775790101371  0.9977531433105469  0.6626065773447016  0.6355140186915887  0.6487775790101371  0.4801412180052957  0.6487775790101371
13  103_img_sli_007.npy 0.7176444737420348  0.9975490570068359  0.6818371607515658  0.7574211502782932  0.7176444737420348  0.5596298834818368  0.7176444737420348
14  103_img_sli_008.npy 0.21840120663650076 0.9950580596923828  0.19535887749595252 0.247606019151846 8 0.21840120663650078 0.12258719945817813 0.21840120663650076
15  103_img_sli_009.npy 0.40851721473340774 0.9934301376342773  0.29244007375537 8  0.6773917995444191  0.40851721473340774 0.25668968493741906 0.40851721473340774
16  103_img_sli_010.npy 0.4912663755458515  0.9937782287597656  0.40866632070576026 0.6157154026583268  0.49126637554585145 0.3256150506512301  0.4912663755458515
17  103_img_sli_011.npy 0.32257231404958675 0.9949970245361328  0.29402071563088 51 0.35726544622425627 0.32257231404958675 0.19230177059276365 0.32257231404958675
```

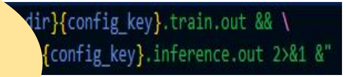
Training/testing/analyzing workflow

1. *Loop through config files*



```
{} E16_B7_lr1.0e-06_wd3.8e-03_g0.97_lfDice_tlmask_decoder.output_hypernetworks_mlps_mask_decoder.iou_token_promp
{} E17_B6_lr7.0e-06_wd1.1e-04_g0.98.json
{} E18_B5_lr4.2e-06_wd1.0e-02_g0.96_lfDefault_tlmask_decoder.iou_token_prompt_encoder.no_mas
{} E18_B6_lr7.2e-06_wd3.5e-03_g0.99_lfBCE.json
{} E19_B4_lr4.6e-06_wd4.0e-03_g0.97.json
{} E20_B3_lr3.0e-06_wd3.6e-03_g0.98_lfDefault_tlprompt_encoder.point
{} E20_B3_lr8.8e-06_wd3.9e-03_g0.99_lfDefault_tlmask_decoder.iou_t
{} E20_B6_lr7.9e-06_wd3.0e-03_g0.99_lfBCE_tlprompt_encoder.no_m
{} E21_B4_lr4.3e-06_wd8.9e-03_g0.99.json
{} E22_B5_lr5.1e-06_wd5.5e-03_g0.97_lfDice_tlmask_decoder
{} E25_B1_lr5.1e-06_wd9.9e-03_g0.98_lfDice_tlmask
{} E25_B6_lr7.0e-06_wd8.5e-04_g0.97_lfBCE_tl
{} E26_B1_lr2.6e-06_wd5.4e-03_g0.96_lfDice
{} E26_B7_lr4.3e-06_wd2.5e-03_g0.98_lfDefa
{} E27_B2_lr3.0e-06_wd2.7e-03_g0.97_lfBCE.js
{} E27_B4_lr3.2e-06_wd1.4e-03_g0.98_lfDefault_tl
{} E28_B3_lr1.0e-06_wd2.7e-03_g0.98_lfDice.json
{} E29_B8_lr6.7e-06_wd3.7e-03_g0.99_lfBCE_tlpromp
```

2. Find GPU with most free space, and if it exceeds 7 using **nohup**



```
dir}{config_key}.train.out && \
{config_key}.inference.out 2>&1 &"
```

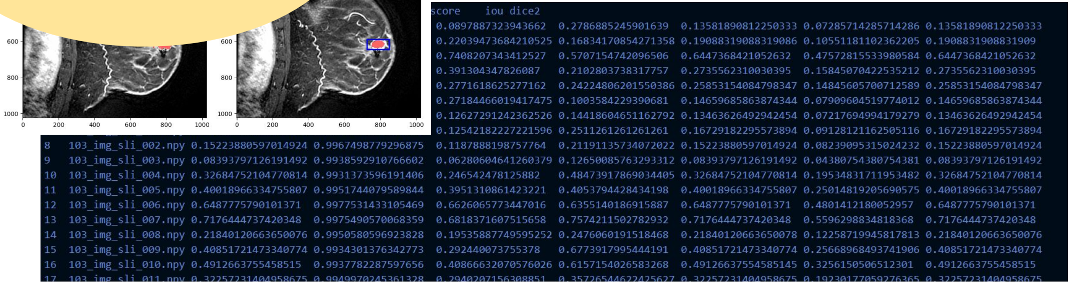
se is to execute a

does not stop

ed nohup.out, if it has

Depending on batch sizes, can run anywhere from 8-15 runs in parallel. Each 'run' can take anywhere from a few hours to >24hrs depending on epochs. If I'm diligent about it, can do up to 25 experiments per day

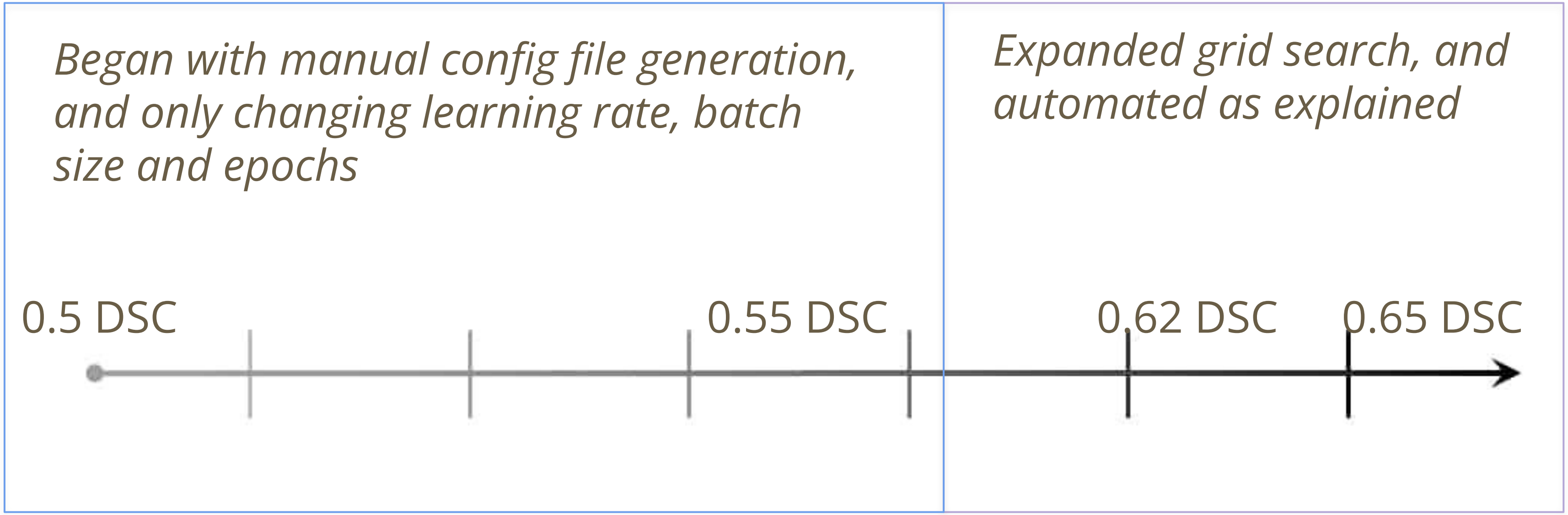
3. Each configuration ge output folder, with quantitative scores per image, visualization of results, trained model, and training loss curves



```
score    iou dice2
 0.0897887323943662  0.2786885245901639  0.13581890812250333 0.07285714285714286 0.13581890812250333
 0.22039473684210525 0.16834170854271358 0.19088319088319086 0.10551181102362205 0.190883190883l909
 0.7408207343412527  0.5707154742096506  0.6447368421052632  0.47572815533980584 0.6447368421052632
 0.391304347826087   0.2102803738317757  0.2735562310030395  0.15845070422535212 0.2735562310030395
 0.277161862527716 2 0.24224806201550386 0.25853154084798347 0.14845605700712589 0.25853154084798347
 0.27184466019417475 0.1003584229390681  0.14659685863874344 0.07909604519774012 0.14659685863874344
 0.12627291242362526 0.14418604651162792 0.13463626492942454 0.07217694994179279 0.13463626492942454
 0.12542182227221596 0.2511261261261261  0.16729182295573894 0.09128121162505116 0.16729182295573894
8   103_img_sli_002.npy 0.15223880597014924 0.9967498779296875  0.1187888198757764  0.21191135734072022 0.15223880597014924 0.08239095315024232 0.15223880597014924
9   103_img_sli_003.npy 0.08393797126191492 0.993859291076660 2 0.06280604641260379 0.12650085763293312 0.08393797126191492 0.04380754380754381 0.08393797126191492
10  103_img_sli_004.npy 0.32684752104770814 0.9931373596191406  0.24654247812588 2  0.48473917869034405 0.32684752104770814 0.19534831711953482 0.32684752104770814
11  103_img_sli_005.npy 0.40018966334755807 0.9951744079589844  0.3951310861423221  0.4053794428434198  0.40018966334755807 0.25014819205690575 0.40018966334755807
12  103_img_sli_006.npy 0.6487775790101371  0.9977531433105469  0.6626065773447016  0.6355140186915887  0.6487775790101371  0.4801412180052957  0.6487775790101371
13  103_img_sli_007.npy 0.7176444737420348  0.9975490570068359  0.6818371607515658  0.7574211502782932  0.7176444737420348  0.5596298834818368  0.7176444737420348
14  103_img_sli_008.npy 0.21840120663650076 0.9950580596923828  0.19535887749595252 0.2476060191518468  0.21840120663650078 0.12258719945817813 0.21840120663650076
15  103_img_sli_009.npy 0.40851721473340774 0.9934301376342773  0.29244007375537 8  0.6773917995444191  0.40851721473340774 0.25668968493741906 0.40851721473340774
16  103_img_sli_010.npy 0.491266375545851 5 0.9937782287597656  0.40866632070576026 0.6157154026583268  0.49126637554585145 0.3256150506512301  0.491266375545851 5
17  103_img_sli_011.npy 0.32257231404958675 0.9949970245361328  0.294020715630885 1 0.35726544622425627 0.32257231404958675 0.19230177059276365 0.32257231404958675
```

Results

Since the first attempt over a month ago, Dice scores on test set have improved from **0.5** to **0.65**.



Began with manual config file generation, and only changing learning rate, batch size and epochs

Expanded grid search, and automated as explained

In total, hundreds of models have been trained

How does 0.65 DSC compare to Jayender et al.

Recall:

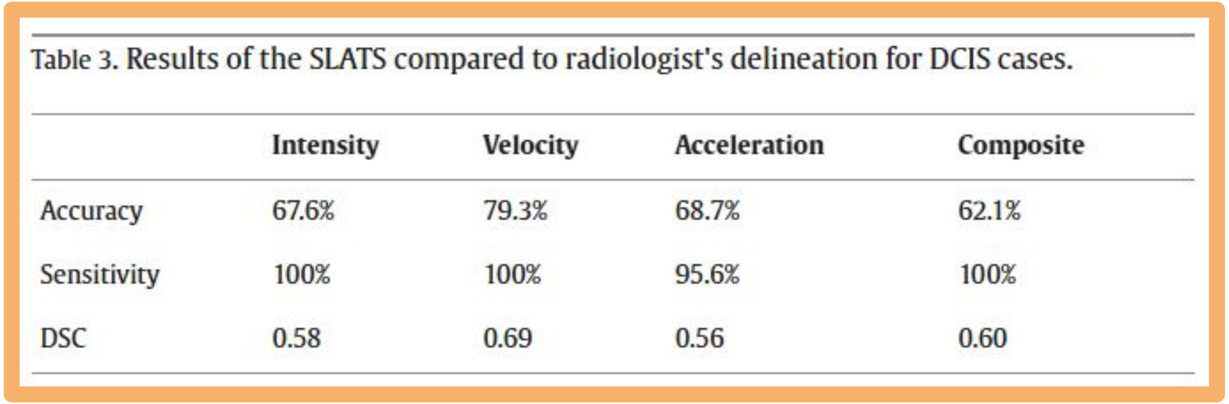
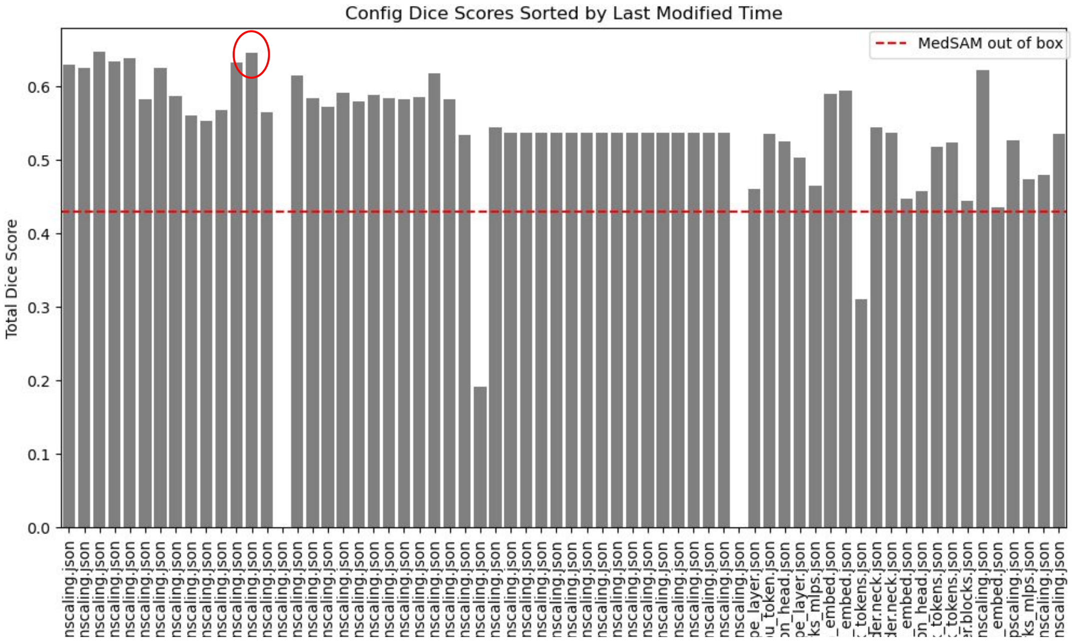


Table 3. Results of the SLATS compared to radiologist's delineation for DCIS cases.

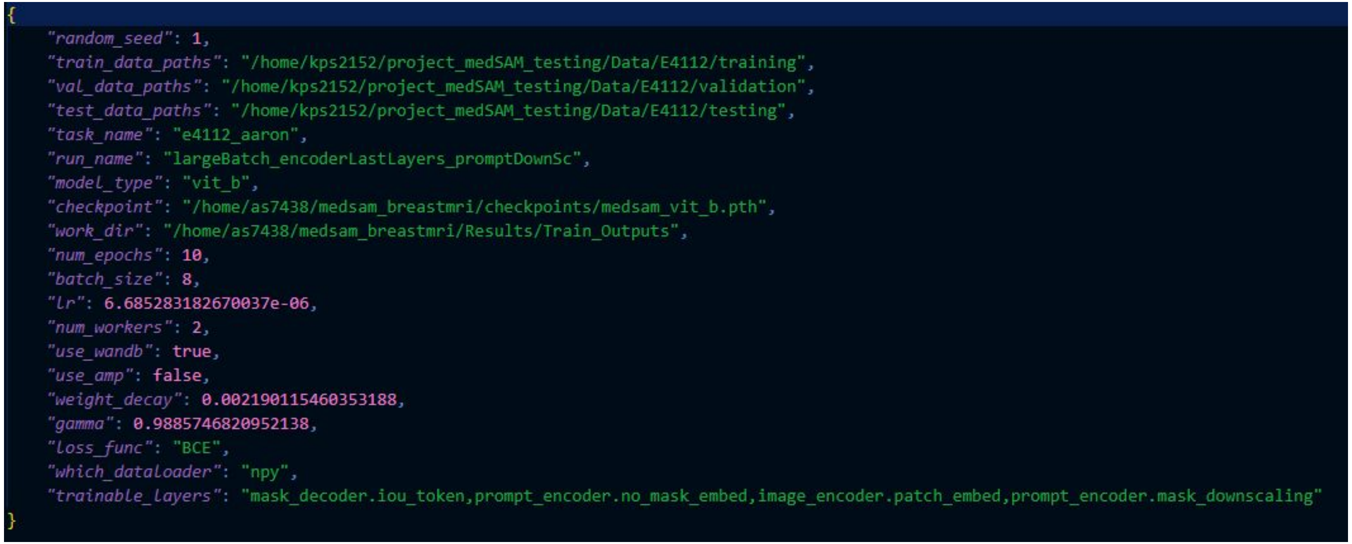
	Intensity	Velocity	Acceleration	Composite
Accuracy	67.6%	79.3%	68.7%	62.1%
Sensitivity	100%	100%	95.6%	100%
DSC	0.58	0.69	0.56	0.60

Same ballpark

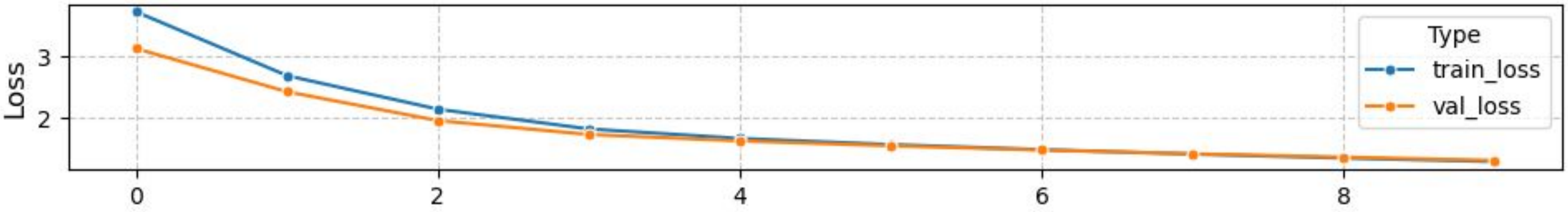
In the last week



Best run 0.65 DSC in more detail



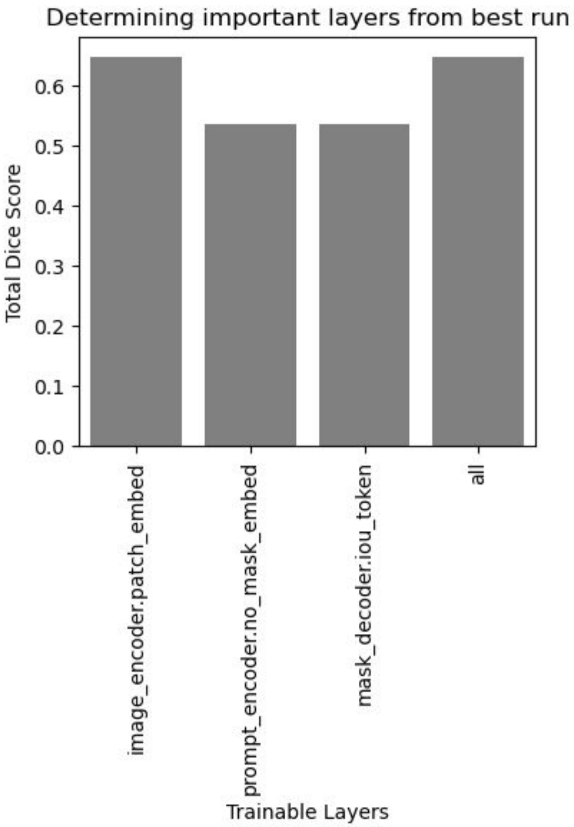
```
{
    "random_seed": 1,
    "train_data_paths": "/home/kps2152/project_medSAM_testing/Data/E4112/training",
    "val_data_paths": "/home/kps2152/project_medSAM_testing/Data/E4112/validation",
    "test_data_paths": "/home/kps2152/project_medSAM_testing/Data/E4112/testing",
    "task_name": "e4112_aaron",
    "run_name": "largeBatch_encoderLastLayers_promptDownSc",
    "model_type": "vit_b",
    "checkpoint": "/home/as7438/medsam_breastmri/checkpoints/medsam_vit_b.pth",
    "work_dir": "/home/as7438/medsam_breastmri/Results/Train_Outputs",
    "num_epochs": 10,
    "batch_size": 8,
    "lr": 6.685283182670037e-06,
    "num_workers": 2,
    "use_wandb": true,
    "use_amp": false,
    "weight_decay": 0.002190115460353188,
    "gamma": 0.9885746820952138,
    "loss_func": "BCE",
    "which_dataloader": "npy",
    "trainable_layers": "mask_decoder.iou_token,prompt_encoder.no_mask_embed,image_encoder.patch_embed,prompt_encoder.mask_downscaling"
}
```



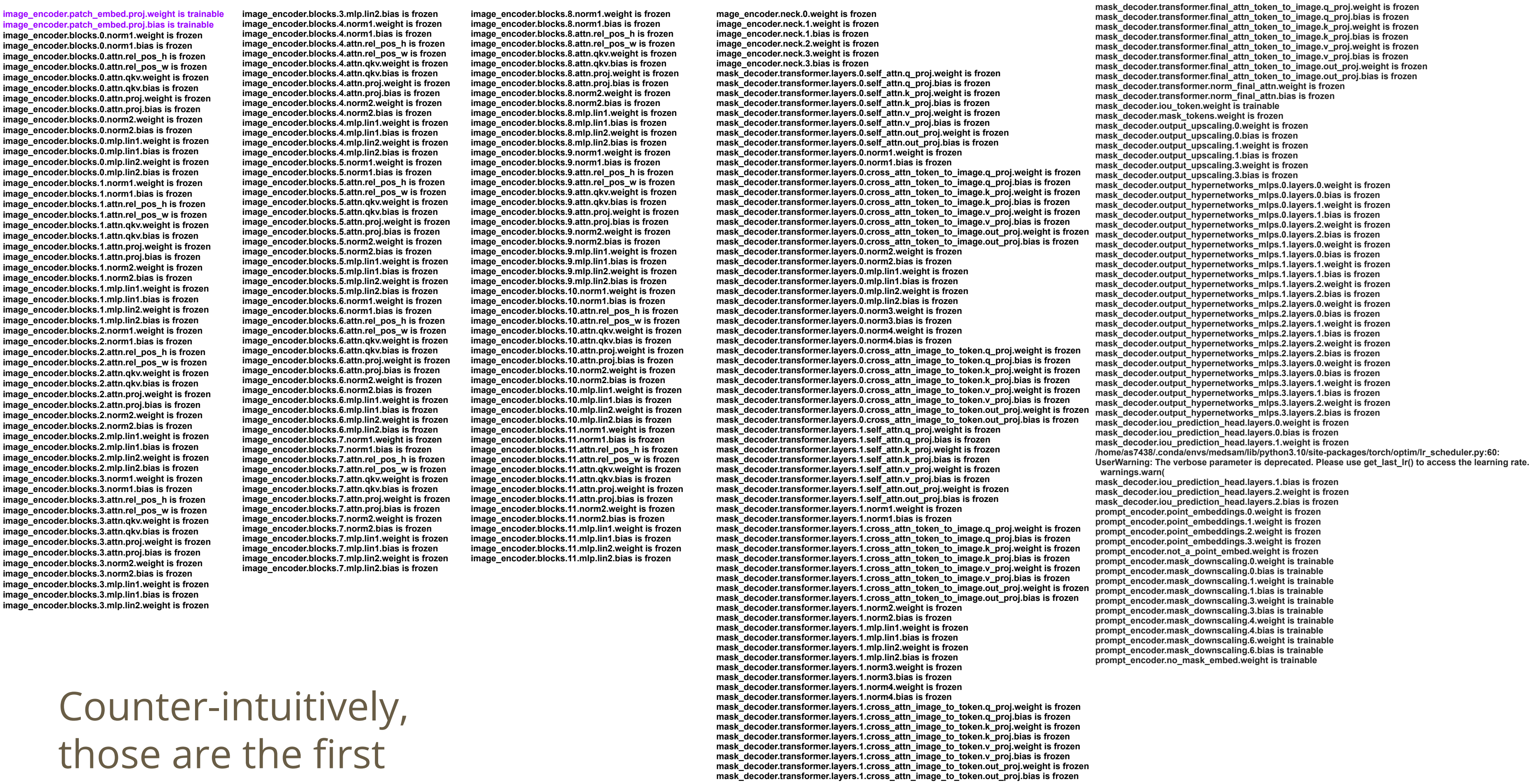
Trainable layers of best model, are all these necessary? Still experimenting with this…

```
image_encoder.patch_embed.proj.weight is trainable
image_encoder.patch_embed.proj.bias is trainable
image_encoder.blocks.0.norm1.weight is frozen
image_encoder.blocks.0.norm1.bias is frozen
image_encoder.blocks.0.attn.rel_pos_h is frozen
image_encoder.blocks.0.attn.rel_pos_w is frozen
image_encoder.blocks.0.attn.qkv.weight is frozen
image_encoder.blocks.0.attn.qkv.bias is frozen
image_encoder.blocks.0.attn.proj.weight is frozen
image_encoder.blocks.0.attn.proj.bias is frozen
image_encoder.blocks.0.norm2.weight is frozen
image_encoder.blocks.0.norm2.bias is frozen
image_encoder.blocks.0.mlp.lin1.weight is frozen
image_encoder.blocks.0.mlp.lin1.bias is frozen
image_encoder.blocks.0.mlp.lin2.weight is frozen
image_encoder.blocks.0.mlp.lin2.bias is frozen
image_encoder.blocks.1.norm1.weight is frozen
image_encoder.blocks.1.norm1.bias is frozen
image_encoder.blocks.1.attn.rel_pos_h is frozen
image_encoder.blocks.1.attn.rel_pos_w is frozen
image_encoder.blocks.1.attn.qkv.weight is frozen
image_encoder.blocks.1.attn.qkv.bias is frozen
image_encoder.blocks.1.attn.proj.weight is frozen
image_encoder.blocks.1.attn.proj.bias is frozen
image_encoder.blocks.1.norm2.weight is frozen
image_encoder.blocks.1.norm2.bias is frozen
image_encoder.blocks.1.mlp.lin1.weight is frozen
image_encoder.blocks.1.mlp.lin1.bias is frozen
image_encoder.blocks.1.mlp.lin2.weight is frozen
image_encoder.blocks.1.mlp.lin2.bias is frozen
image_encoder.blocks.2.norm1.weight is frozen
image_encoder.blocks.2.norm1.bias is frozen
image_encoder.blocks.2.attn.rel_pos_h is frozen
image_encoder.blocks.2.attn.rel_pos_w is frozen
image_encoder.blocks.2.attn.qkv.weight is frozen
image_encoder.blocks.2.attn.qkv.bias is frozen
image_encoder.blocks.2.attn.proj.weight is frozen
image_encoder.blocks.2.attn.proj.bias is frozen
image_encoder.blocks.2.norm2.weight is frozen
image_encoder.blocks.2.norm2.bias is frozen
image_encoder.blocks.2.mlp.lin1.weight is frozen
image_encoder.blocks.2.mlp.lin1.bias is frozen
image_encoder.blocks.2.mlp.lin2.weight is frozen
image_encoder.blocks.2.mlp.lin2.bias is frozen
image_encoder.blocks.3.norm1.weight is frozen
image_encoder.blocks.3.norm1.bias is frozen
image_encoder.blocks.3.attn.rel_pos_h is frozen
image_encoder.blocks.3.attn.rel_pos_w is frozen
image_encoder.blocks.3.attn.qkv.weight is frozen
image_encoder.blocks.3.attn.qkv.bias is frozen
image_encoder.blocks.3.attn.proj.weight is frozen
image_encoder.blocks.3.attn.proj.bias is frozen
image_encoder.blocks.3.norm2.weight is frozen
image_encoder.blocks.3.norm2.bias is frozen
image_encoder.blocks.3.mlp.lin1.weight is frozen
image_encoder.blocks.3.mlp.lin1.bias is frozen
image_encoder.blocks.3.mlp.lin2.weight is frozen
image_encoder.blocks.3.mlp.lin2.bias is frozen
image_encoder.blocks.4.norm1.weight is frozen
image_encoder.blocks.4.norm1.bias is frozen
image_encoder.blocks.4.attn.rel_pos_h is frozen
image_encoder.blocks.4.attn.rel_pos_w is frozen
image_encoder.blocks.4.attn.qkv.weight is frozen
image_encoder.blocks.4.attn.qkv.bias is frozen
image_encoder.blocks.4.attn.proj.weight is frozen
image_encoder.blocks.4.attn.proj.bias is frozen
image_encoder.blocks.4.norm2.weight is frozen
image_encoder.blocks.4.norm2.bias is frozen
image_encoder.blocks.4.mlp.lin1.weight is frozen
image_encoder.blocks.4.mlp.lin1.bias is frozen
image_encoder.blocks.4.mlp.lin2.weight is frozen
image_encoder.blocks.4.mlp.lin2.bias is frozen
image_encoder.blocks.5.norm1.weight is frozen
image_encoder.blocks.5.norm1.bias is frozen
image_encoder.blocks.5.attn.rel_pos_h is frozen
image_encoder.blocks.5.attn.rel_pos_w is frozen
image_encoder.blocks.5.attn.qkv.weight is frozen
image_encoder.blocks.5.attn.qkv.bias is frozen
image_encoder.blocks.5.attn.proj.weight is frozen
image_encoder.blocks.5.attn.proj.bias is frozen
image_encoder.blocks.5.norm2.weight is frozen
image_encoder.blocks.5.norm2.bias is frozen
image_encoder.blocks.5.mlp.lin1.weight is frozen
image_encoder.blocks.5.mlp.lin1.bias is frozen
image_encoder.blocks.5.mlp.lin2.weight is frozen
image_encoder.blocks.5.mlp.lin2.bias is frozen
image_encoder.blocks.6.norm1.weight is frozen
image_encoder.blocks.6.norm1.bias is frozen
image_encoder.blocks.6.attn.rel_pos_h is frozen
image_encoder.blocks.6.attn.rel_pos_w is frozen
image_encoder.blocks.6.attn.qkv.weight is frozen
image_encoder.blocks.6.attn.qkv.bias is frozen
image_encoder.blocks.6.attn.proj.weight is frozen
image_encoder.blocks.6.attn.proj.bias is frozen
image_encoder.blocks.6.norm2.weight is frozen
image_encoder.blocks.6.norm2.bias is frozen
image_encoder.blocks.6.mlp.lin1.weight is frozen
image_encoder.blocks.6.mlp.lin1.bias is frozen
image_encoder.blocks.6.mlp.lin2.weight is frozen
image_encoder.blocks.6.mlp.lin2.bias is frozen
image_encoder.blocks.7.norm1.weight is frozen
image_encoder.blocks.7.norm1.bias is frozen
image_encoder.blocks.7.attn.rel_pos_h is frozen
image_encoder.blocks.7.attn.rel_pos_w is frozen
image_encoder.blocks.7.attn.qkv.weight is frozen
image_encoder.blocks.7.attn.qkv.bias is frozen
image_encoder.blocks.7.attn.proj.weight is frozen
image_encoder.blocks.7.attn.proj.bias is frozen
image_encoder.blocks.7.norm2.weight is frozen
image_encoder.blocks.7.norm2.bias is frozen
image_encoder.blocks.7.mlp.lin1.weight is frozen
image_encoder.blocks.7.mlp.lin1.bias is frozen
image_encoder.blocks.7.mlp.lin2.weight is frozen
image_encoder.blocks.7.mlp.lin2.bias is frozen
image_encoder.blocks.8.norm1.weight is frozen
image_encoder.blocks.8.norm1.bias is frozen
image_encoder.blocks.8.attn.rel_pos_h is frozen
image_encoder.blocks.8.attn.rel_pos_w is frozen
image_encoder.blocks.8.attn.qkv.weight is frozen
image_encoder.blocks.8.attn.qkv.bias is frozen
image_encoder.blocks.8.attn.proj.weight is frozen
image_encoder.blocks.8.attn.proj.bias is frozen
image_encoder.blocks.8.norm2.weight is frozen
image_encoder.blocks.8.norm2.bias is frozen
image_encoder.blocks.8.mlp.lin1.weight is frozen
image_encoder.blocks.8.mlp.lin1.bias is frozen
image_encoder.blocks.8.mlp.lin2.weight is frozen
image_encoder.blocks.8.mlp.lin2.bias is frozen
image_encoder.blocks.9.norm1.weight is frozen
image_encoder.blocks.9.norm1.bias is frozen
image_encoder.blocks.9.attn.rel_pos_h is frozen
image_encoder.blocks.9.attn.rel_pos_w is frozen
image_encoder.blocks.9.attn.qkv.weight is frozen
image_encoder.blocks.9.attn.qkv.bias is frozen
image_encoder.blocks.9.attn.proj.weight is frozen
image_encoder.blocks.9.attn.proj.bias is frozen
image_encoder.blocks.9.norm2.weight is frozen
image_encoder.blocks.9.norm2.bias is frozen
image_encoder.blocks.9.mlp.lin1.weight is frozen
image_encoder.blocks.9.mlp.lin1.bias is frozen
image_encoder.blocks.9.mlp.lin2.weight is frozen
image_encoder.blocks.9.mlp.lin2.bias is frozen
image_encoder.blocks.10.norm1.weight is frozen
image_encoder.blocks.10.norm1.bias is frozen
image_encoder.blocks.10.attn.rel_pos_h is frozen
image_encoder.blocks.10.attn.rel_pos_w is frozen
image_encoder.blocks.10.attn.qkv.weight is frozen
image_encoder.blocks.10.attn.qkv.bias is frozen
image_encoder.blocks.10.attn.proj.weight is frozen
image_encoder.blocks.10.attn.proj.bias is frozen
image_encoder.blocks.10.norm2.weight is frozen
image_encoder.blocks.10.norm2.bias is frozen
image_encoder.blocks.10.mlp.lin1.weight is frozen
image_encoder.blocks.10.mlp.lin1.bias is frozen
image_encoder.blocks.10.mlp.lin2.weight is frozen
image_encoder.blocks.10.mlp.lin2.bias is frozen
image_encoder.blocks.11.norm1.weight is frozen
image_encoder.blocks.11.norm1.bias is frozen
image_encoder.blocks.11.attn.rel_pos_h is frozen
image_encoder.blocks.11.attn.rel_pos_w is frozen
image_encoder.blocks.11.attn.qkv.weight is frozen
image_encoder.blocks.11.attn.qkv.bias is frozen
image_encoder.blocks.11.attn.proj.weight is frozen
image_encoder.blocks.11.attn.proj.bias is frozen
image_encoder.blocks.11.norm2.weight is frozen
image_encoder.blocks.11.norm2.bias is frozen
image_encoder.blocks.11.mlp.lin1.weight is frozen
image_encoder.blocks.11.mlp.lin1.bias is frozen
image_encoder.blocks.11.mlp.lin2.weight is frozen
image_encoder.blocks.11.mlp.lin2.bias is frozen
mage_encoder.neck.0.weight is frozen
image_encoder.neck.1.weight is frozen
image_encoder.neck.1.bias is frozen
image_encoder.neck.2.weight is frozen
image_encoder.neck.3.weight is frozen
image_encoder.neck.3.bias is frozen
mask_decoder.transformer.layers.0.self_attn.q_proj.weight is frozen
mask_decoder.transformer.layers.0.self_attn.q_proj.bias is frozen
mask_decoder.transformer.layers.0.self_attn.k_proj.weight is frozen
mask_decoder.transformer.layers.0.self_attn.k_proj.bias is frozen
mask_decoder.transformer.layers.0.self_attn.v_proj.weight is frozen
mask_decoder.transformer.layers.0.self_attn.v_proj.bias is frozen
mask_decoder.transformer.layers.0.self_attn.out_proj.weight is frozen
mask_decoder.transformer.layers.0.self_attn.out_proj.bias is frozen
mask_decoder.transformer.layers.0.norm1.weight is frozen
mask_decoder.transformer.layers.0.norm1.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.q_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.q_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.k_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.k_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.v_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.v_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.out_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.out_proj.bias is frozen
mask_decoder.transformer.layers.0.norm2.weight is frozen
mask_decoder.transformer.layers.0.norm2.bias is frozen
mask_decoder.transformer.layers.0.mlp.lin1.weight is frozen
mask_decoder.transformer.layers.0.mlp.lin1.bias is frozen
mask_decoder.transformer.layers.0.mlp.lin2.weight is frozen
mask_decoder.transformer.layers.0.mlp.lin2.bias is frozen
mask_decoder.transformer.layers.0.norm3.weight is frozen
mask_decoder.transformer.layers.0.norm3.bias is frozen
mask_decoder.transformer.layers.0.norm4.weight is frozen
mask_decoder.transformer.layers.0.norm4.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.q_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.q_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.k_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.k_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.v_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.v_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.out_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.out_proj.bias is frozen
mask_decoder.transformer.layers.1.self_attn.q_proj.weight is frozen
mask_decoder.transformer.layers.1.self_attn.q_proj.bias is frozen
mask_decoder.transformer.layers.1.self_attn.k_proj.weight is frozen
mask_decoder.transformer.layers.1.self_attn.k_proj.bias is frozen
mask_decoder.transformer.layers.1.self_attn.v_proj.weight is frozen
mask_decoder.transformer.layers.1.self_attn.v_proj.bias is frozen
mask_decoder.transformer.layers.1.self_attn.out_proj.weight is frozen
mask_decoder.transformer.layers.1.self_attn.out_proj.bias is frozen
mask_decoder.transformer.layers.1.norm1.weight is frozen
mask_decoder.transformer.layers.1.norm1.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.q_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.q_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.k_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.k_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.v_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.v_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.out_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.out_proj.bias is frozen
mask_decoder.transformer.layers.1.norm2.weight is frozen
mask_decoder.transformer.layers.1.norm2.bias is frozen
mask_decoder.transformer.layers.1.mlp.lin1.weight is frozen
mask_decoder.transformer.layers.1.mlp.lin1.bias is frozen
mask_decoder.transformer.layers.1.mlp.lin2.weight is frozen
mask_decoder.transformer.layers.1.mlp.lin2.bias is frozen
mask_decoder.transformer.layers.1.norm3.weight is frozen
mask_decoder.transformer.layers.1.norm3.bias is frozen
mask_decoder.transformer.layers.1.norm4.weight is frozen
mask_decoder.transformer.layers.1.norm4.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.q_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.q_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.k_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.k_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.v_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.v_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.out_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.out_proj.bias is frozen
mask_decoder.transformer.final_attn_token_to_image.q_proj.weight is frozen
mask_decoder.transformer.final_attn_token_to_image.q_proj.bias is frozen
mask_decoder.transformer.final_attn_token_to_image.k_proj.weight is frozen
mask_decoder.transformer.final_attn_token_to_image.k_proj.bias is frozen
mask_decoder.transformer.final_attn_token_to_image.v_proj.weight is frozen
mask_decoder.transformer.final_attn_token_to_image.v_proj.bias is frozen
mask_decoder.transformer.final_attn_token_to_image.out_proj.weight is frozen
mask_decoder.transformer.final_attn_token_to_image.out_proj.bias is frozen
mask_decoder.transformer.norm_final_attn.weight is frozen
mask_decoder.transformer.norm_final_attn.bias is frozen
mask_decoder.iou_token.weight is trainable
mask_decoder.mask_tokens.weight is frozen
mask_decoder.output_upscaling.0.weight is frozen
mask_decoder.output_upscaling.0.bias is frozen
mask_decoder.output_upscaling.1.weight is frozen
mask_decoder.output_upscaling.1.bias is frozen
mask_decoder.output_upscaling.3.weight is frozen
mask_decoder.output_upscaling.3.bias is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.0.weight is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.0.bias is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.1.weight is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.1.bias is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.2.weight is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.2.bias is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.0.weight is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.0.bias is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.1.weight is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.1.bias is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.2.weight is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.2.bias is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.0.weight is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.0.bias is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.1.weight is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.1.bias is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.2.weight is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.2.bias is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.0.weight is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.0.bias is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.1.weight is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.1.bias is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.2.weight is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.2.bias is frozen
mask_decoder.iou_prediction_head.layers.0.weight is frozen
mask_decoder.iou_prediction_head.layers.0.bias is frozen
mask_decoder.iou_prediction_head.layers.1.weight is frozen
/home/as7438/.conda/envs/medsam/lib/python3.10/site-packages/torch/optim/lr_scheduler.py:60:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to access the learning rate.
  warnings.warn(
mask_decoder.iou_prediction_head.layers.1.bias is frozen
mask_decoder.iou_prediction_head.layers.2.weight is frozen
mask_decoder.iou_prediction_head.layers.2.bias is frozen
prompt_encoder.point_embeddings.0.weight is frozen
prompt_encoder.point_embeddings.1.weight is frozen
prompt_encoder.point_embeddings.2.weight is frozen
prompt_encoder.point_embeddings.3.weight is frozen
prompt_encoder.not_a_point_embed.weight is frozen
prompt_encoder.mask_downscaling.0.weight is trainable
prompt_encoder.mask_downscaling.0.bias is trainable
prompt_encoder.mask_downscaling.1.weight is trainable
prompt_encoder.mask_downscaling.1.bias is trainable
prompt_encoder.mask_downscaling.3.weight is trainable
prompt_encoder.mask_downscaling.3.bias is trainable
prompt_encoder.mask_downscaling.4.weight is trainable
prompt_encoder.mask_downscaling.4.bias is trainable
prompt_encoder.mask_downscaling.6.weight is trainable
prompt_encoder.mask_downscaling.6.bias is trainable
prompt_encoder.no_mask_embed.weight is trainable
```

Which of the 3 layer groups was most important?



Same hyper-parameters except trainable layer



```
image_encoder.patch_embed.proj.weight is trainable
image_encoder.patch_embed.proj.bias is trainable
image_encoder.blocks.0.norm1.weight is frozen
image_encoder.blocks.0.norm1.bias is frozen
image_encoder.blocks.0.attn.rel_pos_h is frozen
image_encoder.blocks.0.attn.rel_pos_w is frozen
image_encoder.blocks.0.attn.qkv.weight is frozen
image_encoder.blocks.0.attn.qkv.bias is frozen
image_encoder.blocks.0.attn.proj.weight is frozen
image_encoder.blocks.0.attn.proj.bias is frozen
image_encoder.blocks.0.norm2.weight is frozen
image_encoder.blocks.0.norm2.bias is frozen
image_encoder.blocks.0.mlp.lin1.weight is frozen
image_encoder.blocks.0.mlp.lin1.bias is frozen
image_encoder.blocks.0.mlp.lin2.weight is frozen
image_encoder.blocks.0.mlp.lin2.bias is frozen
image_encoder.blocks.1.norm1.weight is frozen
image_encoder.blocks.1.norm1.bias is frozen
image_encoder.blocks.1.attn.rel_pos_h is frozen
image_encoder.blocks.1.attn.rel_pos_w is frozen
image_encoder.blocks.1.attn.qkv.weight is frozen
image_encoder.blocks.1.attn.qkv.bias is frozen
image_encoder.blocks.1.attn.proj.weight is frozen
image_encoder.blocks.1.attn.proj.bias is frozen
image_encoder.blocks.1.norm2.weight is frozen
image_encoder.blocks.1.norm2.bias is frozen
image_encoder.blocks.1.mlp.lin1.weight is frozen
image_encoder.blocks.1.mlp.lin1.bias is frozen
image_encoder.blocks.1.mlp.lin2.weight is frozen
image_encoder.blocks.1.mlp.lin2.bias is frozen
image_encoder.blocks.2.norm1.weight is frozen
image_encoder.blocks.2.norm1.bias is frozen
image_encoder.blocks.2.attn.rel_pos_h is frozen
image_encoder.blocks.2.attn.rel_pos_w is frozen
image_encoder.blocks.2.attn.qkv.weight is frozen
image_encoder.blocks.2.attn.qkv.bias is frozen
image_encoder.blocks.2.attn.proj.weight is frozen
image_encoder.blocks.2.attn.proj.bias is frozen
image_encoder.blocks.2.norm2.weight is frozen
image_encoder.blocks.2.norm2.bias is frozen
image_encoder.blocks.2.mlp.lin1.weight is frozen
image_encoder.blocks.2.mlp.lin1.bias is frozen
image_encoder.blocks.2.mlp.lin2.weight is frozen
image_encoder.blocks.2.mlp.lin2.bias is frozen
image_encoder.blocks.3.norm1.weight is frozen
image_encoder.blocks.3.norm1.bias is frozen
image_encoder.blocks.3.attn.rel_pos_h is frozen
image_encoder.blocks.3.attn.rel_pos_w is frozen
image_encoder.blocks.3.attn.qkv.weight is frozen
image_encoder.blocks.3.attn.qkv.bias is frozen
image_encoder.blocks.3.attn.proj.weight is frozen
image_encoder.blocks.3.attn.proj.bias is frozen
image_encoder.blocks.3.norm2.weight is frozen
image_encoder.blocks.3.norm2.bias is frozen
image_encoder.blocks.3.mlp.lin1.weight is frozen
image_encoder.blocks.3.mlp.lin1.bias is frozen
image_encoder.blocks.3.mlp.lin2.weight is frozen
```

```
image_encoder.blocks.3.mlp.lin2.bias is frozen
image_encoder.blocks.4.norm1.weight is frozen
image_encoder.blocks.4.norm1.bias is frozen
image_encoder.blocks.4.attn.rel_pos_h is frozen
image_encoder.blocks.4.attn.rel_pos_w is frozen
image_encoder.blocks.4.attn.qkv.weight is frozen
image_encoder.blocks.4.attn.qkv.bias is frozen
image_encoder.blocks.4.attn.proj.weight is frozen
image_encoder.blocks.4.attn.proj.bias is frozen
image_encoder.blocks.4.norm2.weight is frozen
image_encoder.blocks.4.norm2.bias is frozen
image_encoder.blocks.4.mlp.lin1.weight is frozen
image_encoder.blocks.4.mlp.lin1.bias is frozen
image_encoder.blocks.4.mlp.lin2.weight is frozen
image_encoder.blocks.4.mlp.lin2.bias is frozen
image_encoder.blocks.5.norm1.weight is frozen
image_encoder.blocks.5.norm1.bias is frozen
image_encoder.blocks.5.attn.rel_pos_h is frozen
image_encoder.blocks.5.attn.rel_pos_w is frozen
image_encoder.blocks.5.attn.qkv.weight is frozen
image_encoder.blocks.5.attn.qkv.bias is frozen
image_encoder.blocks.5.attn.proj.weight is frozen
image_encoder.blocks.5.attn.proj.bias is frozen
image_encoder.blocks.5.norm2.weight is frozen
image_encoder.blocks.5.norm2.bias is frozen
image_encoder.blocks.5.mlp.lin1.weight is frozen
image_encoder.blocks.5.mlp.lin1.bias is frozen
image_encoder.blocks.5.mlp.lin2.weight is frozen
image_encoder.blocks.5.mlp.lin2.bias is frozen
image_encoder.blocks.6.norm1.weight is frozen
image_encoder.blocks.6.norm1.bias is frozen
image_encoder.blocks.6.attn.rel_pos_h is frozen
image_encoder.blocks.6.attn.rel_pos_w is frozen
image_encoder.blocks.6.attn.qkv.weight is frozen
image_encoder.blocks.6.attn.qkv.bias is frozen
image_encoder.blocks.6.attn.proj.weight is frozen
image_encoder.blocks.6.attn.proj.bias is frozen
image_encoder.blocks.6.norm2.weight is frozen
image_encoder.blocks.6.norm2.bias is frozen
image_encoder.blocks.6.mlp.lin1.weight is frozen
image_encoder.blocks.6.mlp.lin1.bias is frozen
image_encoder.blocks.6.mlp.lin2.weight is frozen
image_encoder.blocks.6.mlp.lin2.bias is frozen
image_encoder.blocks.7.norm1.weight is frozen
image_encoder.blocks.7.norm1.bias is frozen
image_encoder.blocks.7.attn.rel_pos_h is frozen
image_encoder.blocks.7.attn.rel_pos_w is frozen
image_encoder.blocks.7.attn.qkv.weight is frozen
image_encoder.blocks.7.attn.qkv.bias is frozen
image_encoder.blocks.7.attn.proj.weight is frozen
image_encoder.blocks.7.attn.proj.bias is frozen
image_encoder.blocks.7.norm2.weight is frozen
image_encoder.blocks.7.norm2.bias is frozen
image_encoder.blocks.7.mlp.lin1.weight is frozen
image_encoder.blocks.7.mlp.lin1.bias is frozen
image_encoder.blocks.7.mlp.lin2.weight is frozen
image_encoder.blocks.7.mlp.lin2.bias is frozen
```

```
image_encoder.blocks.8.norm1.weight is frozen
image_encoder.blocks.8.norm1.bias is frozen
image_encoder.blocks.8.attn.rel_pos_h is frozen
image_encoder.blocks.8.attn.rel_pos_w is frozen
image_encoder.blocks.8.attn.qkv.weight is frozen
image_encoder.blocks.8.attn.qkv.bias is frozen
image_encoder.blocks.8.attn.proj.weight is frozen
image_encoder.blocks.8.attn.proj.bias is frozen
image_encoder.blocks.8.norm2.weight is frozen
image_encoder.blocks.8.norm2.bias is frozen
image_encoder.blocks.8.mlp.lin1.weight is frozen
image_encoder.blocks.8.mlp.lin1.bias is frozen
image_encoder.blocks.8.mlp.lin2.weight is frozen
image_encoder.blocks.8.mlp.lin2.bias is frozen
image_encoder.blocks.9.norm1.weight is frozen
image_encoder.blocks.9.norm1.bias is frozen
image_encoder.blocks.9.attn.rel_pos_h is frozen
image_encoder.blocks.9.attn.rel_pos_w is frozen
image_encoder.blocks.9.attn.qkv.weight is frozen
image_encoder.blocks.9.attn.qkv.bias is frozen
image_encoder.blocks.9.attn.proj.weight is frozen
image_encoder.blocks.9.attn.proj.bias is frozen
image_encoder.blocks.9.norm2.weight is frozen
image_encoder.blocks.9.norm2.bias is frozen
image_encoder.blocks.9.mlp.lin1.weight is frozen
image_encoder.blocks.9.mlp.lin1.bias is frozen
image_encoder.blocks.9.mlp.lin2.weight is frozen
image_encoder.blocks.9.mlp.lin2.bias is frozen
image_encoder.blocks.10.norm1.weight is frozen
image_encoder.blocks.10.norm1.bias is frozen
image_encoder.blocks.10.attn.rel_pos_h is frozen
image_encoder.blocks.10.attn.rel_pos_w is frozen
image_encoder.blocks.10.attn.qkv.weight is frozen
image_encoder.blocks.10.attn.qkv.bias is frozen
image_encoder.blocks.10.attn.proj.weight is frozen
image_encoder.blocks.10.attn.proj.bias is frozen
image_encoder.blocks.10.norm2.weight is frozen
image_encoder.blocks.10.norm2.bias is frozen
image_encoder.blocks.10.mlp.lin1.weight is frozen
image_encoder.blocks.10.mlp.lin1.bias is frozen
image_encoder.blocks.10.mlp.lin2.weight is frozen
image_encoder.blocks.10.mlp.lin2.bias is frozen
image_encoder.blocks.11.norm1.weight is frozen
image_encoder.blocks.11.norm1.bias is frozen
image_encoder.blocks.11.attn.rel_pos_h is frozen
image_encoder.blocks.11.attn.rel_pos_w is frozen
image_encoder.blocks.11.attn.qkv.weight is frozen
image_encoder.blocks.11.attn.qkv.bias is frozen
image_encoder.blocks.11.attn.proj.weight is frozen
image_encoder.blocks.11.attn.proj.bias is frozen
image_encoder.blocks.11.norm2.weight is frozen
image_encoder.blocks.11.norm2.bias is frozen
image_encoder.blocks.11.mlp.lin1.weight is frozen
image_encoder.blocks.11.mlp.lin1.bias is frozen
image_encoder.blocks.11.mlp.lin2.weight is frozen
image_encoder.blocks.11.mlp.lin2.bias is frozen
```

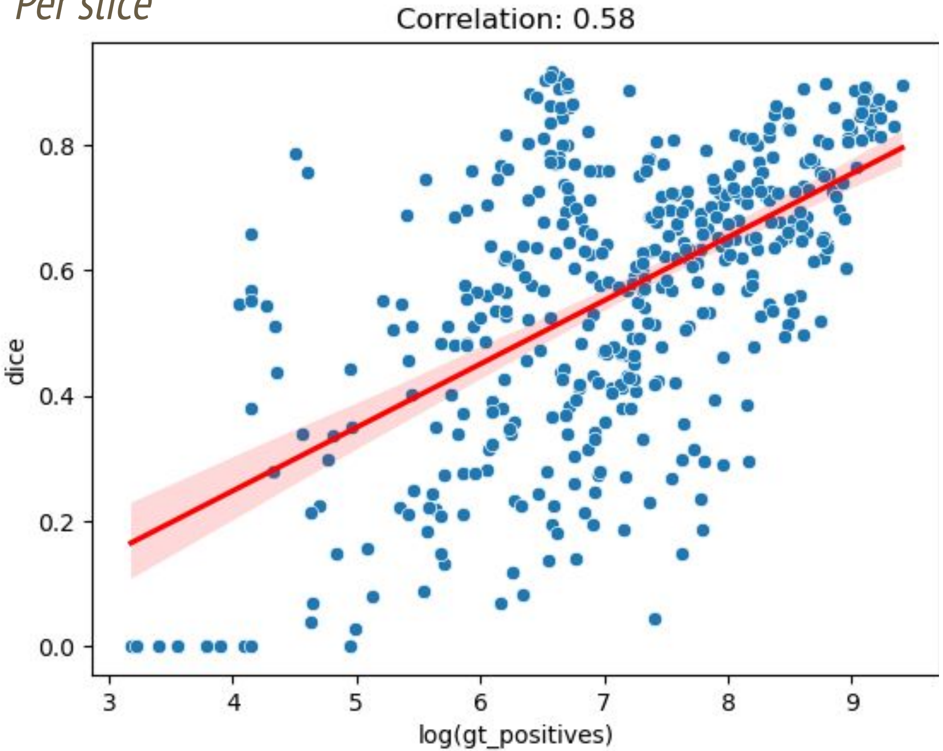
```
mage_encoder.neck.0.weight is frozen
image_encoder.neck.1.weight is frozen
image_encoder.neck.1.bias is frozen
image_encoder.neck.2.weight is frozen
image_encoder.neck.3.weight is frozen
image_encoder.neck.3.bias is frozen
mask_decoder.transformer.layers.0.self_attn.q_proj.weight is frozen
mask_decoder.transformer.layers.0.self_attn.q_proj.bias is frozen
mask_decoder.transformer.layers.0.self_attn.k_proj.weight is frozen
mask_decoder.transformer.layers.0.self_attn.k_proj.bias is frozen
mask_decoder.transformer.layers.0.self_attn.v_proj.weight is frozen
mask_decoder.transformer.layers.0.self_attn.v_proj.bias is frozen
mask_decoder.transformer.layers.0.self_attn.out_proj.weight is frozen
mask_decoder.transformer.layers.0.self_attn.out_proj.bias is frozen
mask_decoder.transformer.layers.0.norm1.weight is frozen
mask_decoder.transformer.layers.0.norm1.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.q_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.q_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.k_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.k_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.v_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.v_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.out_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_token_to_image.out_proj.bias is frozen
mask_decoder.transformer.layers.0.norm2.weight is frozen
mask_decoder.transformer.layers.0.norm2.bias is frozen
mask_decoder.transformer.layers.0.mlp.lin1.weight is frozen
mask_decoder.transformer.layers.0.mlp.lin1.bias is frozen
mask_decoder.transformer.layers.0.mlp.lin2.weight is frozen
mask_decoder.transformer.layers.0.mlp.lin2.bias is frozen
mask_decoder.transformer.layers.0.norm3.weight is frozen
mask_decoder.transformer.layers.0.norm3.bias is frozen
mask_decoder.transformer.layers.0.norm4.weight is frozen
mask_decoder.transformer.layers.0.norm4.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.q_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.q_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.k_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.k_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.v_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.v_proj.bias is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.out_proj.weight is frozen
mask_decoder.transformer.layers.0.cross_attn_image_to_token.out_proj.bias is frozen
mask_decoder.transformer.layers.1.self_attn.q_proj.weight is frozen
mask_decoder.transformer.layers.1.self_attn.q_proj.bias is frozen
mask_decoder.transformer.layers.1.self_attn.k_proj.weight is frozen
mask_decoder.transformer.layers.1.self_attn.k_proj.bias is frozen
mask_decoder.transformer.layers.1.self_attn.v_proj.weight is frozen
mask_decoder.transformer.layers.1.self_attn.v_proj.bias is frozen
mask_decoder.transformer.layers.1.self_attn.out_proj.weight is frozen
mask_decoder.transformer.layers.1.self_attn.out_proj.bias is frozen
mask_decoder.transformer.layers.1.norm1.weight is frozen
mask_decoder.transformer.layers.1.norm1.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.q_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.q_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.k_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.k_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.v_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.v_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.out_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_token_to_image.out_proj.bias is frozen
mask_decoder.transformer.layers.1.norm2.weight is frozen
mask_decoder.transformer.layers.1.norm2.bias is frozen
mask_decoder.transformer.layers.1.mlp.lin1.weight is frozen
mask_decoder.transformer.layers.1.mlp.lin1.bias is frozen
mask_decoder.transformer.layers.1.mlp.lin2.weight is frozen
mask_decoder.transformer.layers.1.mlp.lin2.bias is frozen
mask_decoder.transformer.layers.1.norm3.weight is frozen
mask_decoder.transformer.layers.1.norm3.bias is frozen
mask_decoder.transformer.layers.1.norm4.weight is frozen
mask_decoder.transformer.layers.1.norm4.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.q_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.q_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.k_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.k_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.v_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.v_proj.bias is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.out_proj.weight is frozen
mask_decoder.transformer.layers.1.cross_attn_image_to_token.out_proj.bias is frozen
```

```
mask_decoder.transformer.final_attn_token_to_image.q_proj.weight is frozen
mask_decoder.transformer.final_attn_token_to_image.q_proj.bias is frozen
mask_decoder.transformer.final_attn_token_to_image.k_proj.weight is frozen
mask_decoder.transformer.final_attn_token_to_image.k_proj.bias is frozen
mask_decoder.transformer.final_attn_token_to_image.v_proj.weight is frozen
mask_decoder.transformer.final_attn_token_to_image.v_proj.bias is frozen
mask_decoder.transformer.final_attn_token_to_image.out_proj.weight is frozen
mask_decoder.transformer.final_attn_token_to_image.out_proj.bias is frozen
mask_decoder.transformer.norm_final_attn.weight is frozen
mask_decoder.transformer.norm_final_attn.bias is frozen
mask_decoder.iou_token.weight is trainable
mask_decoder.mask_tokens.weight is frozen
mask_decoder.output_upscaling.0.weight is frozen
mask_decoder.output_upscaling.0.bias is frozen
mask_decoder.output_upscaling.1.weight is frozen
mask_decoder.output_upscaling.1.bias is frozen
mask_decoder.output_upscaling.3.weight is frozen
mask_decoder.output_upscaling.3.bias is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.0.weight is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.0.bias is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.1.weight is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.1.bias is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.2.weight is frozen
mask_decoder.output_hypernetworks_mlps.0.layers.2.bias is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.0.weight is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.0.bias is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.1.weight is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.1.bias is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.2.weight is frozen
mask_decoder.output_hypernetworks_mlps.1.layers.2.bias is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.0.weight is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.0.bias is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.1.weight is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.1.bias is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.2.weight is frozen
mask_decoder.output_hypernetworks_mlps.2.layers.2.bias is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.0.weight is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.0.bias is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.1.weight is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.1.bias is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.2.weight is frozen
mask_decoder.output_hypernetworks_mlps.3.layers.2.bias is frozen
mask_decoder.iou_prediction_head.layers.0.weight is frozen
mask_decoder.iou_prediction_head.layers.0.bias is frozen
mask_decoder.iou_prediction_head.layers.1.weight is frozen
/home/as7438/.conda/envs/medsam/lib/python3.10/site-packages/torch/optim/lr_scheduler.py:60:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to access the learning rate.
  warnings.warn(
mask_decoder.iou_prediction_head.layers.1.bias is frozen
mask_decoder.iou_prediction_head.layers.2.weight is frozen
mask_decoder.iou_prediction_head.layers.2.bias is frozen
prompt_encoder.point_embeddings.0.weight is frozen
prompt_encoder.point_embeddings.1.weight is frozen
prompt_encoder.point_embeddings.2.weight is frozen
prompt_encoder.point_embeddings.3.weight is frozen
prompt_encoder.not_a_point_embed.weight is frozen
prompt_encoder.mask_downscaling.0.weight is trainable
prompt_encoder.mask_downscaling.0.bias is trainable
prompt_encoder.mask_downscaling.1.weight is trainable
prompt_encoder.mask_downscaling.1.bias is trainable
prompt_encoder.mask_downscaling.3.weight is trainable
prompt_encoder.mask_downscaling.3.bias is trainable
prompt_encoder.mask_downscaling.4.weight is trainable
prompt_encoder.mask_downscaling.4.bias is trainable
prompt_encoder.mask_downscaling.6.weight is trainable
prompt_encoder.mask_downscaling.6.bias is trainable
prompt_encoder.no_mask_embed.weight is trainable
```

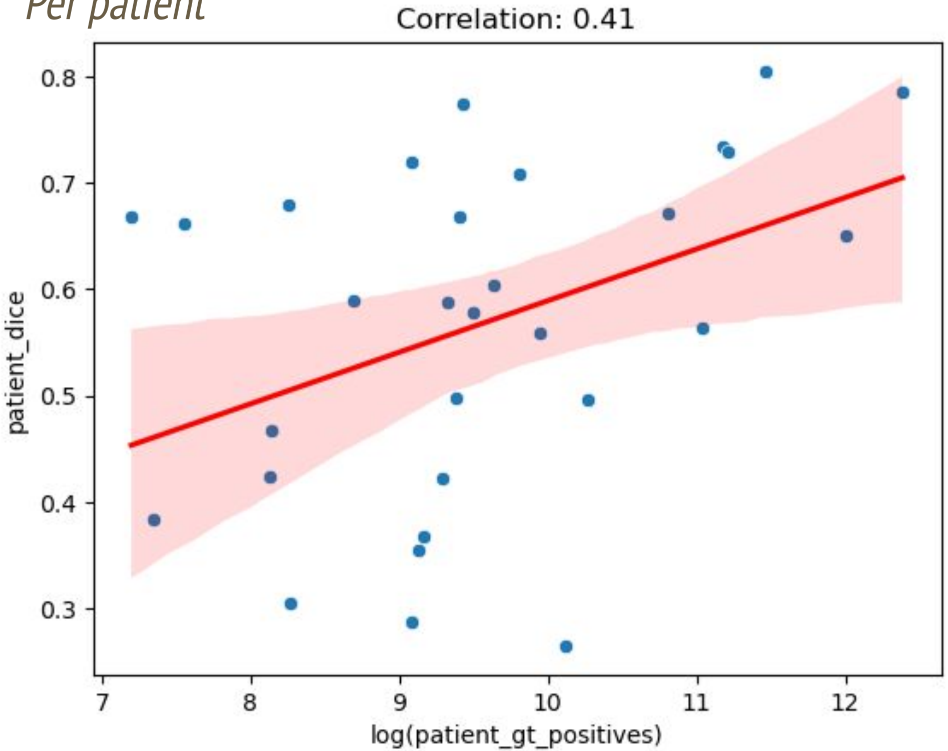
Counter-intuitively, those are the first layers

Results are better in scans with more DCIS

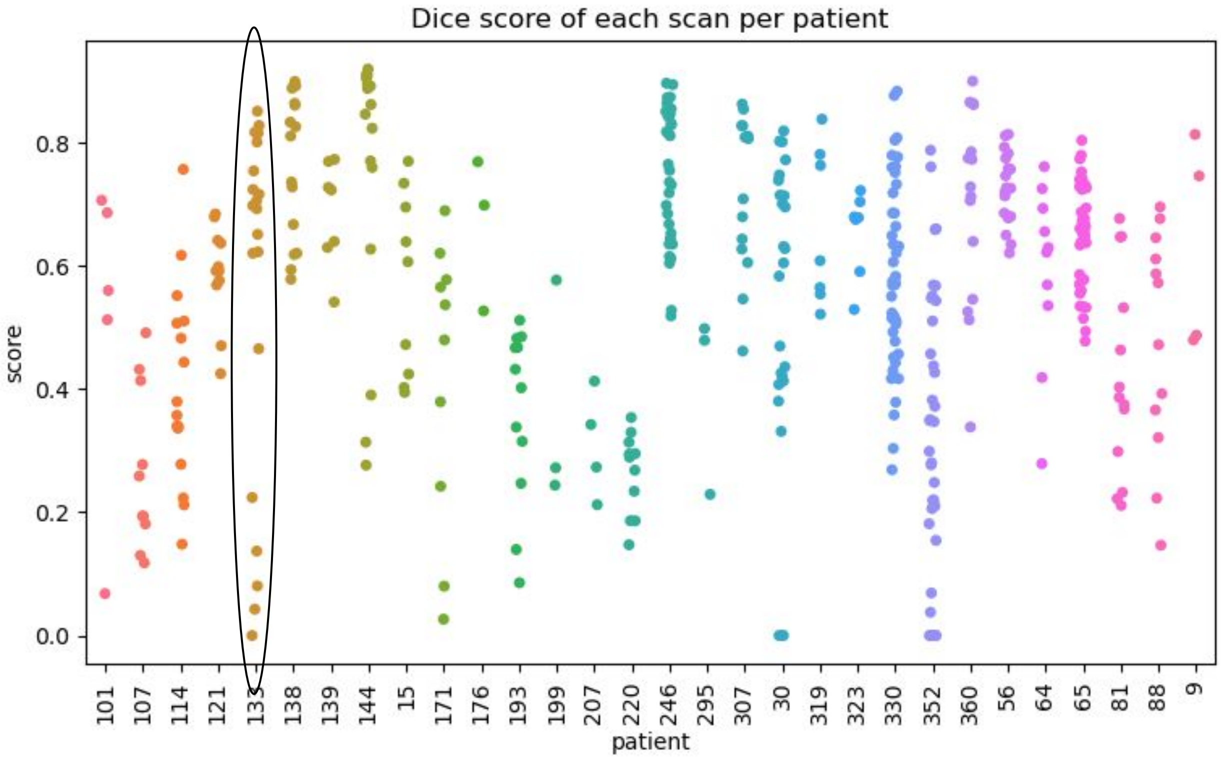
Per slice



Per patient



Even though there are some ‘zeros’ model never misses a patient’s entire DCIS lesion



How should we quantify success with Dice score?

TP_{ij} Is true positive count of **i-th patient** and **j-th slice**

“Each slice weighted equally”

$$D = \frac{1}{\text{number of slices}} \sum_i \sum_j \left(\frac{2 \cdot TP_{ij}}{2 \cdot TP_{ij} + FP_{ij} + FN_{ij}} \right)$$

“Each patient weighted equally”

$$D = \frac{1}{\text{number of patients}} \sum_i \left(\frac{2 \cdot \sum_j TP_{ij}}{2 \cdot \sum_j TP_{ij} + \sum_j FP_{ij} + \sum_j FN_{ij}} \right)$$

“Each voxel weighted equally”

$$D = \frac{2 \cdot \sum_i \sum_j TP_{ij}}{2 \cdot \sum_i \sum_j TP_{ij} + \sum_i \sum_j FP_{ij} + \sum_i \sum_j FN_{ij}}$$

How should we quantify success with Dice score?

TP_{ij} Is true positive count of **i-th patient** and **j-th slice**

“Each slice weighted equally”

$$D = \frac{\quad}{\text{nu}}$$

Because of artificial zeros, and biases towards patients with more/less dice, I like *total dice approach* (still feels arbitrary)

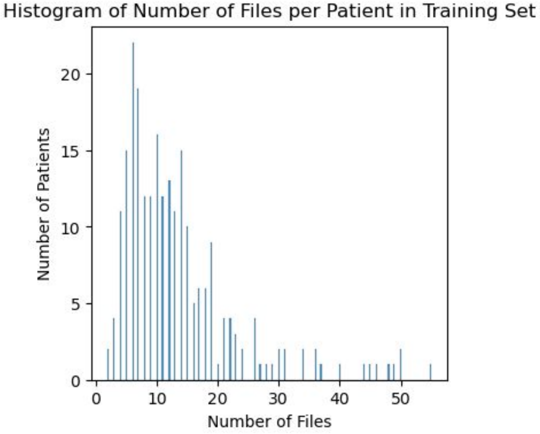
“Each patient weighted e

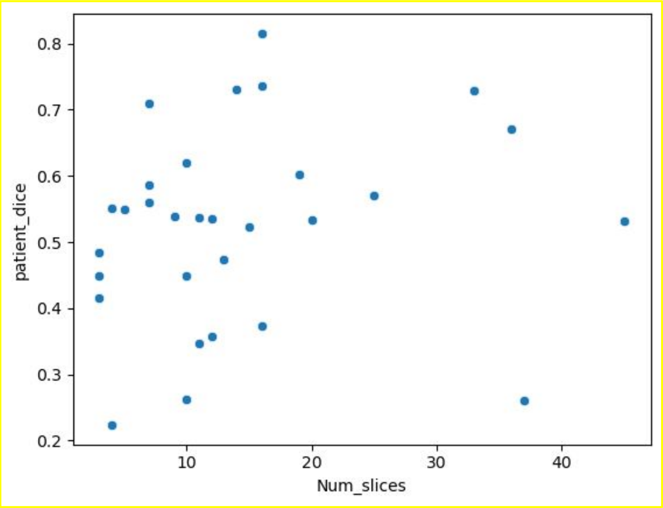
$$D = \frac{1}{\text{number of patients}} \sum_i \left(\frac{\ldots_{ij}}{\ldots_{ij} + \sum_j FP_{ij} + \sum_j FN_{ij}} \right)$$

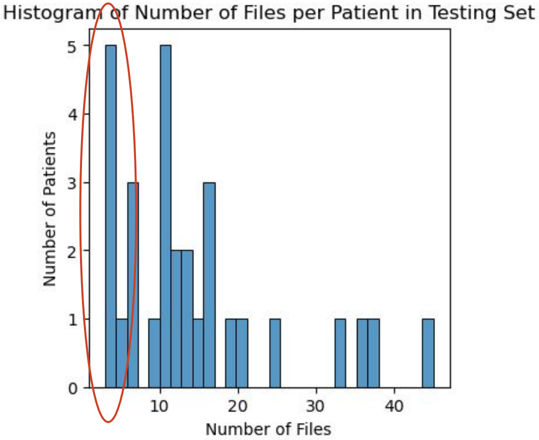
“Each voxel weighted equally” (my favorite)

$$D = \frac{2 \cdot \sum_i \sum_j TP_{ij}}{2 \cdot \sum_i \sum_j TP_{ij} + \sum_i \sum_j FP_{ij} + \sum_i \sum_j FN_{ij}}$$

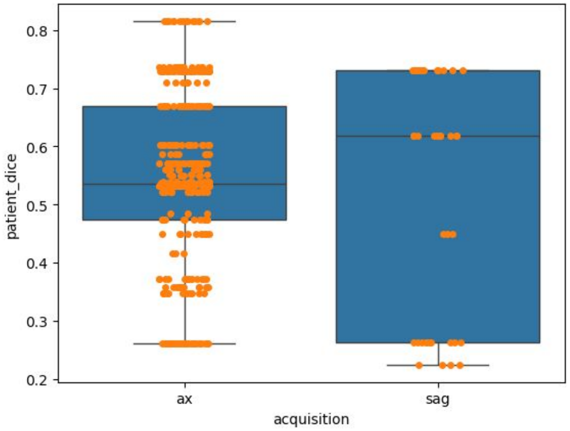
We need to be careful about #slices per patient being balanced between train/test

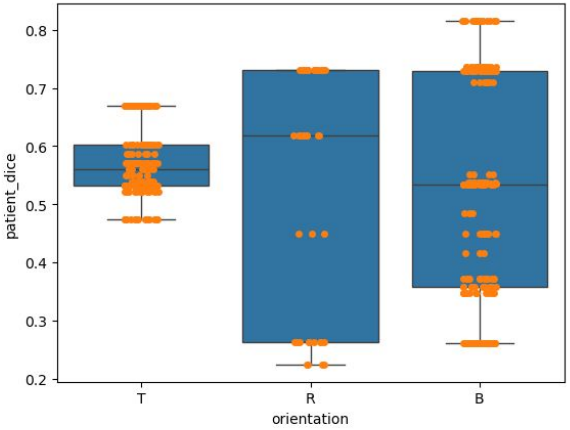


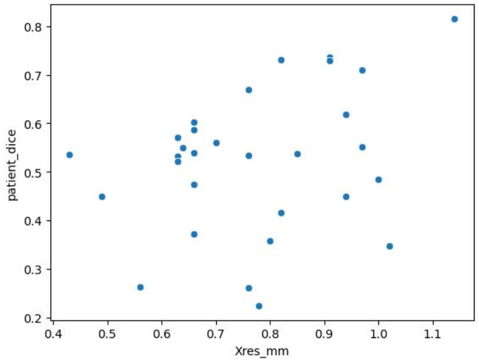


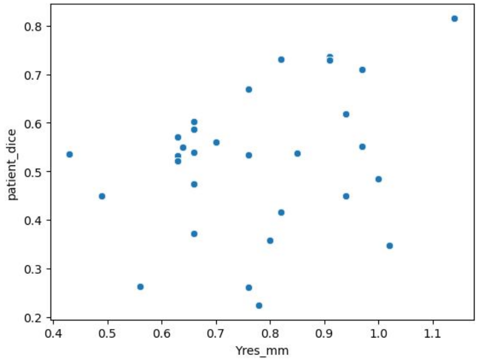


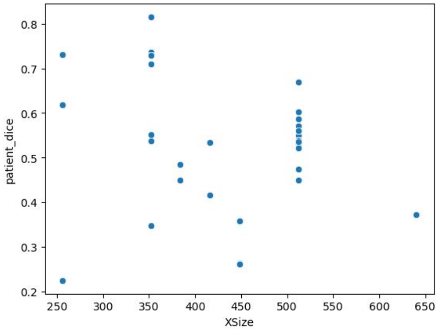
Biases from acquisition?

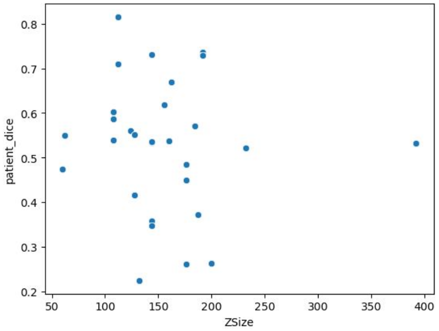


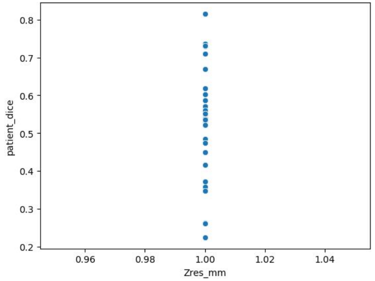






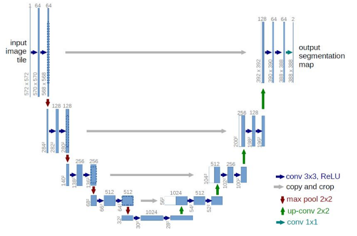




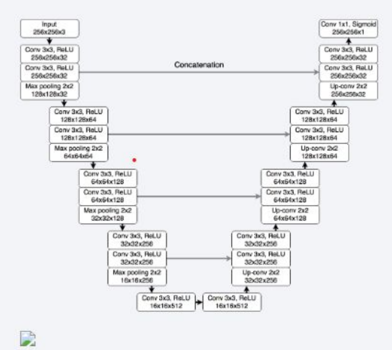


Control studies with Unets

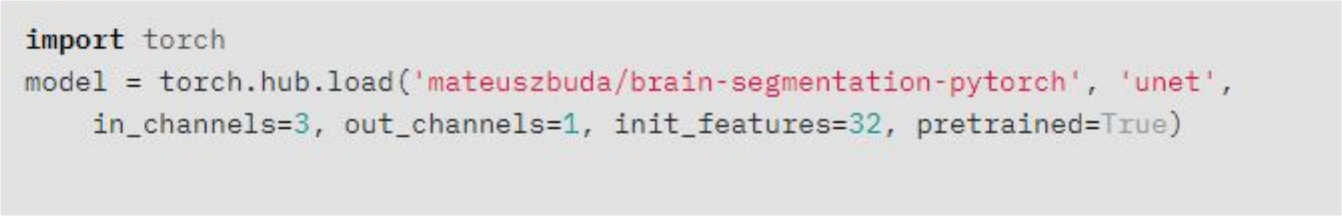
U-Net: Convolutional Networks for Biomedical Image Segmentation



1. Base PyTorch Unet found here: https://github.com/milesial/Pytorch-UNet



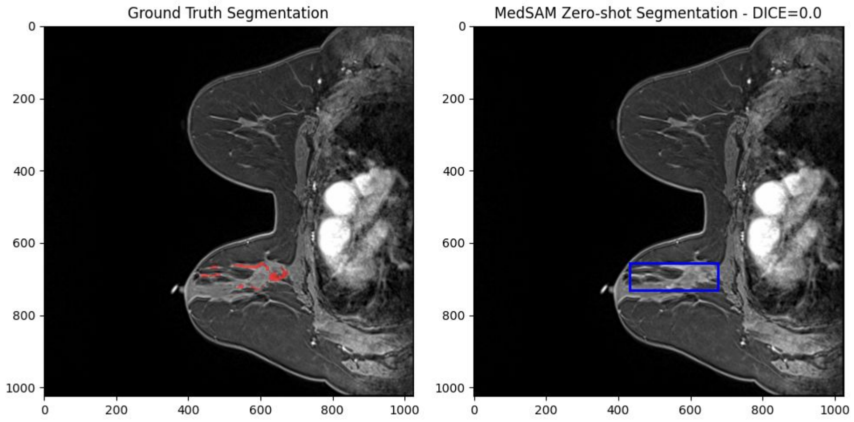
2. Pretrained Unet for brain MRI segmentation: https://pytorch.org/hub/mateuszbuda_brain-segmentation-pytorch_unet/

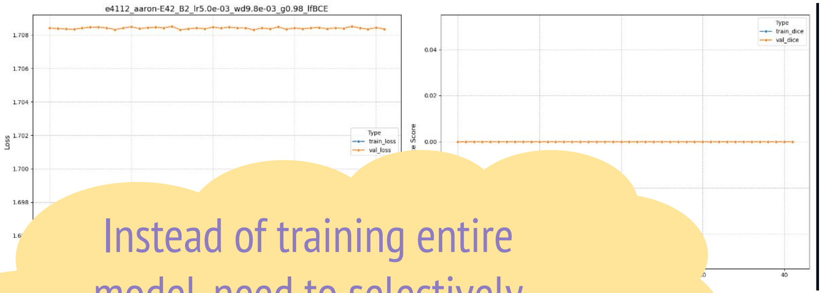


```
import torch
model = torch.hub.load('mateuszbuda/brain-segmentation-pytorch', 'unet',
    in_channels=3, out_channels=1, init_features=32, pretrained=True)
```

Starting with Pretrained Unet for brain MRI segmentation

Long story short.. Training has proven difficult. 20 configs tried.. No progress yet.


Ground Truth Segmentation
MedSAM Zero-shot Segmentation - DICE=0.0
0
200
400
600
800
1000



Instead of training entire model, need to selectively train output layers like in MedSAM

General Future Ideas? Feedback?

- If we want to leverage full abilities of DCE-MRI, we should **include more post-contrast images**
- If we want to create a solution that does not require DCE-MRI, we should replace our dataset with **pre-contrast image** (analogous to standard MRI)

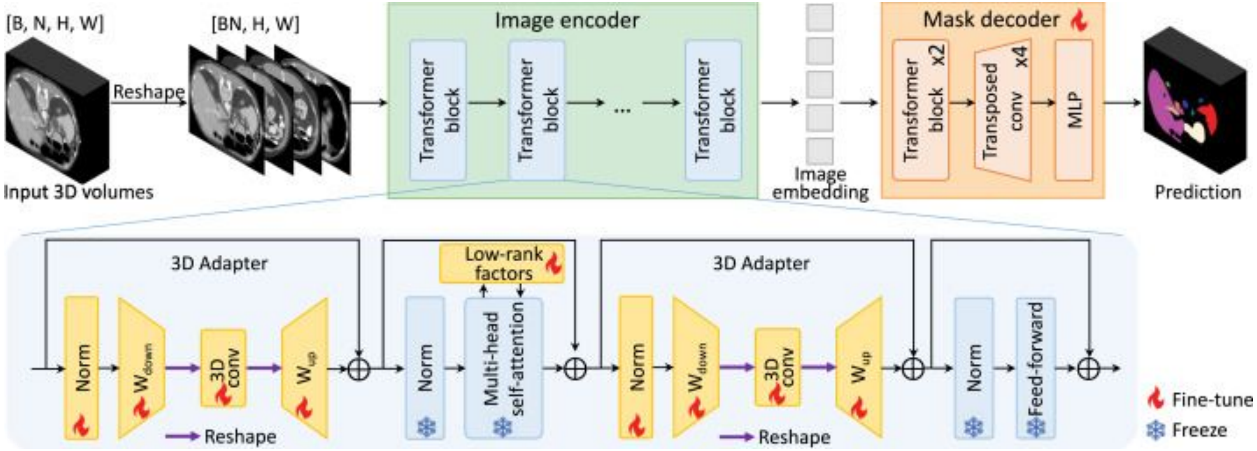
- However, if we want to see how far the current approach can take us, then I propose the following...

Future work in optimizing MedSAM approach

1. Run more training experiments
   a. Some completely random to create new breakthroughs
   b. Some based on previous breakthroughs to optimize
2. Preliminary experimentation with Unets as controls
3. Preprocessing modifications (as discussed)
4. Inclusion of DCIS-free images from same patients in order to create fully automated method for clinicians
5. 3D/4D approach (most interesting) using **MA-SAM**
   a. 3D represents 3D MRI input -> 3D MRI output
   b. 4D represents multiple post-contrast images
6. Can we transfer segmentation models to non-low-risk vs. low-risk classification / treatment information?

MA-SAM

We know SAM is bad with medical images. MedSAM addresses this by fine-tuning it on millions of medical images. MA-SAM addresses this by allowing 3D inputs, since these are so important for medical modalities.



Cheng Chen, Juzheng Miao, Dufan Wu, Aoxiao Zhong, Zhiling Yan, Sekeun Kim, Jiang Hu, Zhengliang Liu, Lichao Sun, Xiang Li, Tianming Liu, Pheng-Ann Heng, Quanzheng Li, MA-SAM: Modality-agnostic SAM adaptation for 3D medical image segmentation, Medical Image Analysis,Volume 98,2024,103310,ISSN 1361-8415,