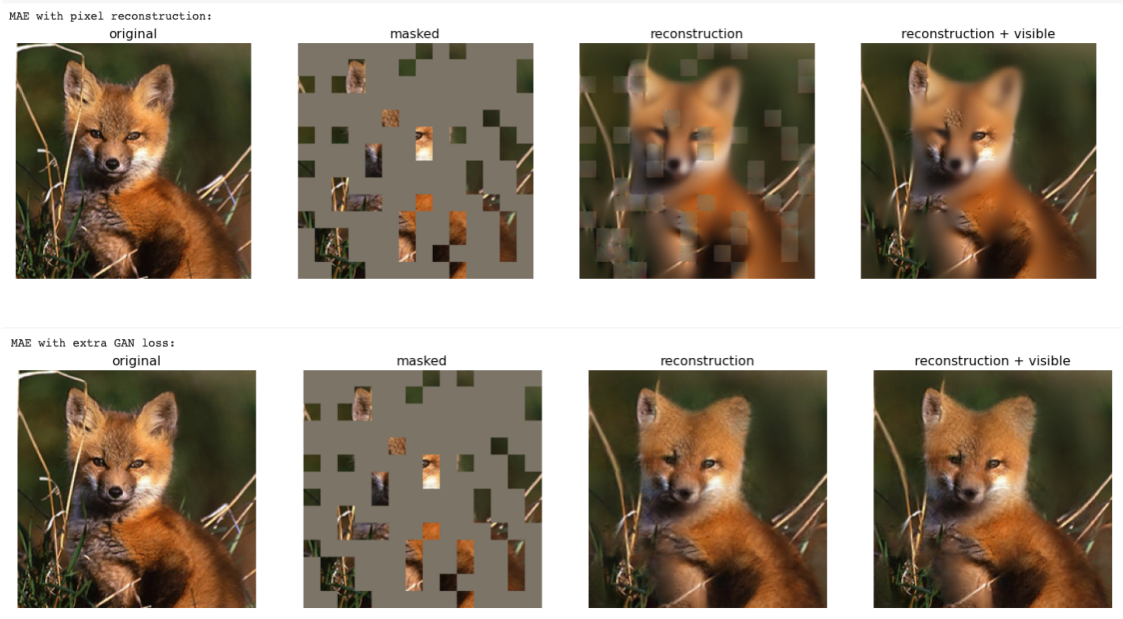
Exploring finetuning ViT MAEs for fg/bg segmentation

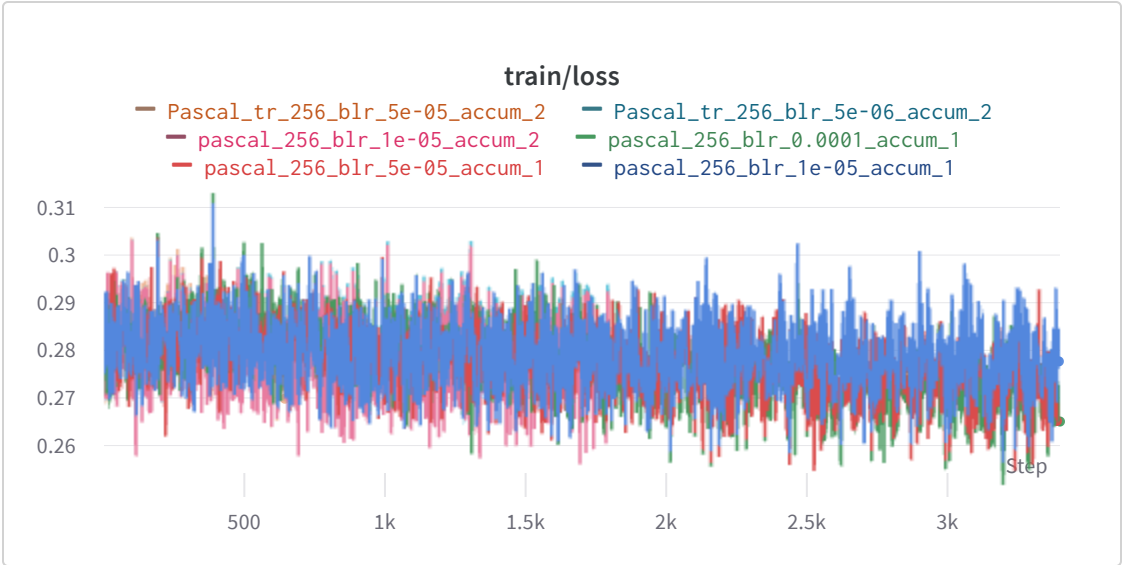
Klára Janoušková

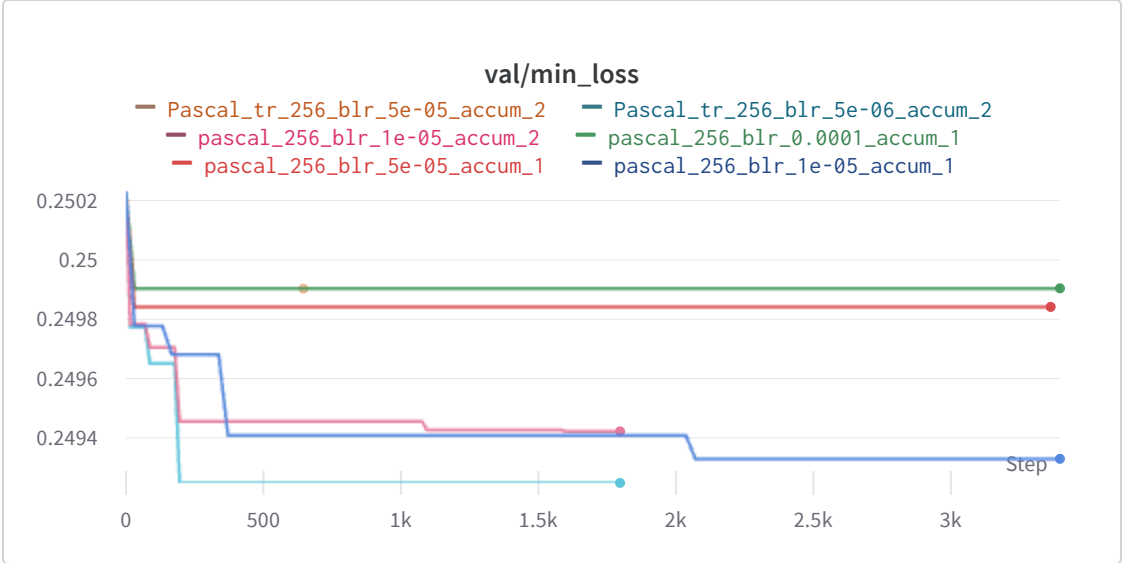
An example of the pretrained MAE model output from the authors, both with and without using GAN loss:

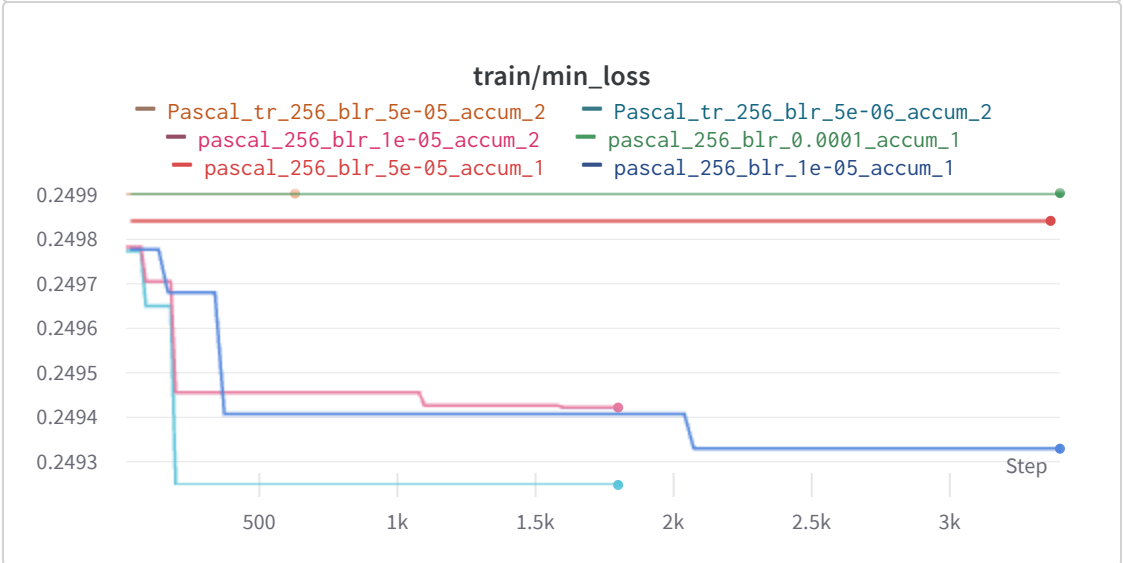


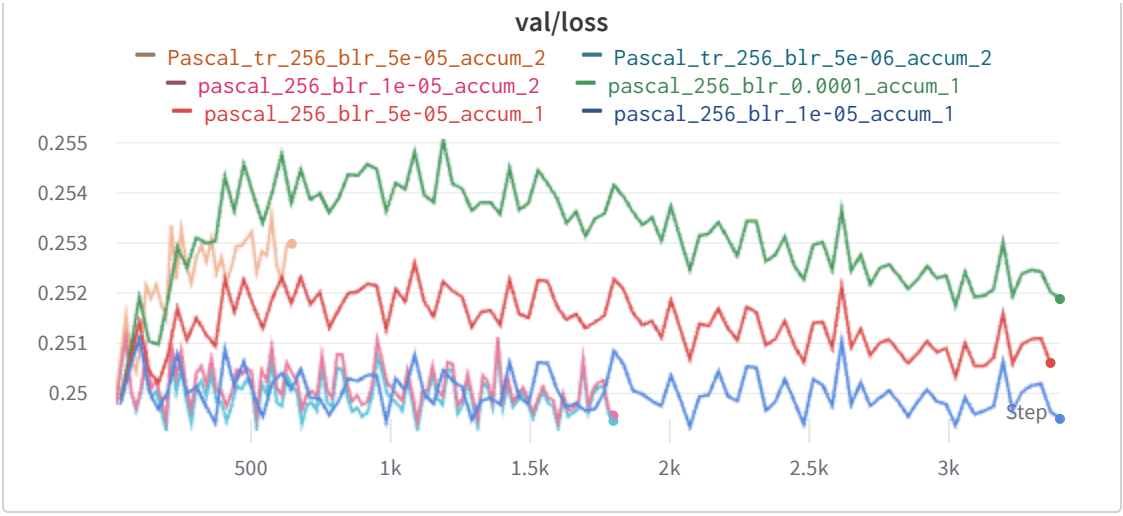
Reconstruction finetuning on pascal:

The min val loss doesn't really change. If we increase lr, it starts increasing (not sure it is visible here). This could be because the pascale dataset is very similar to image-net.









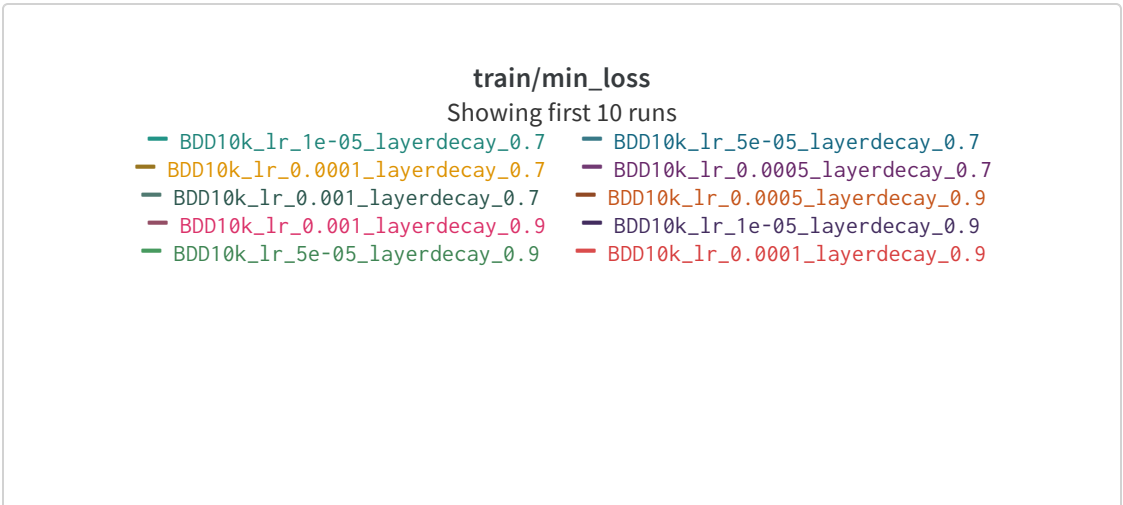
Reconstruction finetuning on Berkely Deep Drive

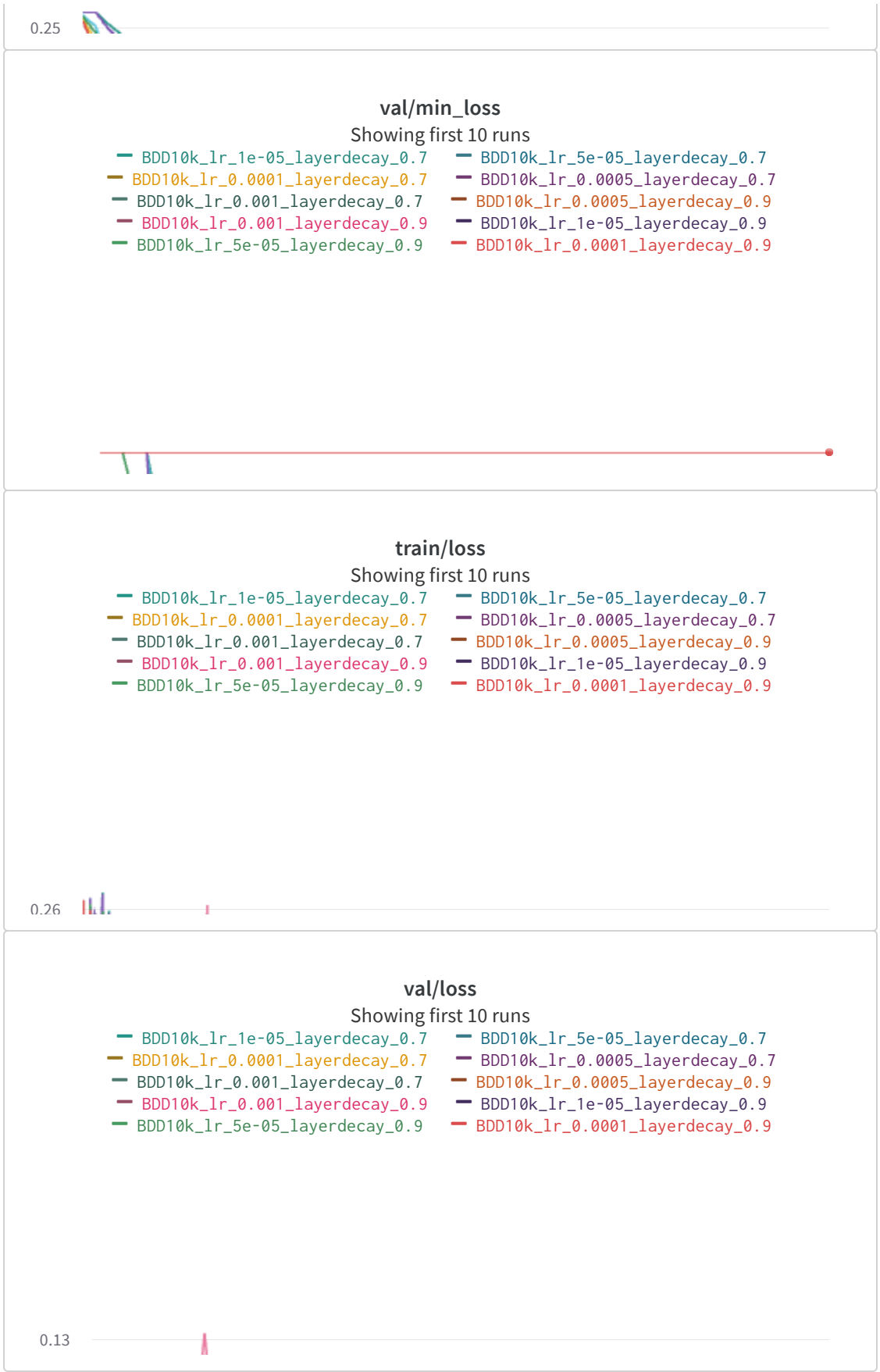
In these experiments, we try to change two things:

1) We chose a dataset that should be different enough from ImageNet, where the model was pre-trained

2) We remove optimization tricks from the finetuning code (layer decay, cosine scheduler, ...) and only use basic AdamW optimization with constant learning rate.

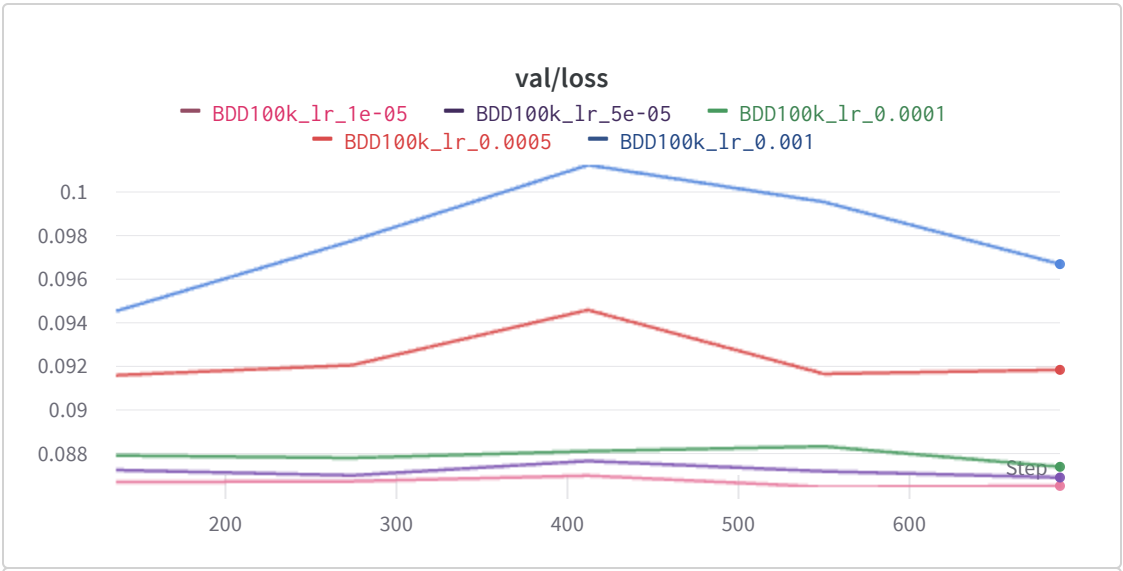
When we change the dataset, the trianing curves look good on the training dataset and the do decrease a bit on the validation set in the first few iterations. Still not ideal but better than before:

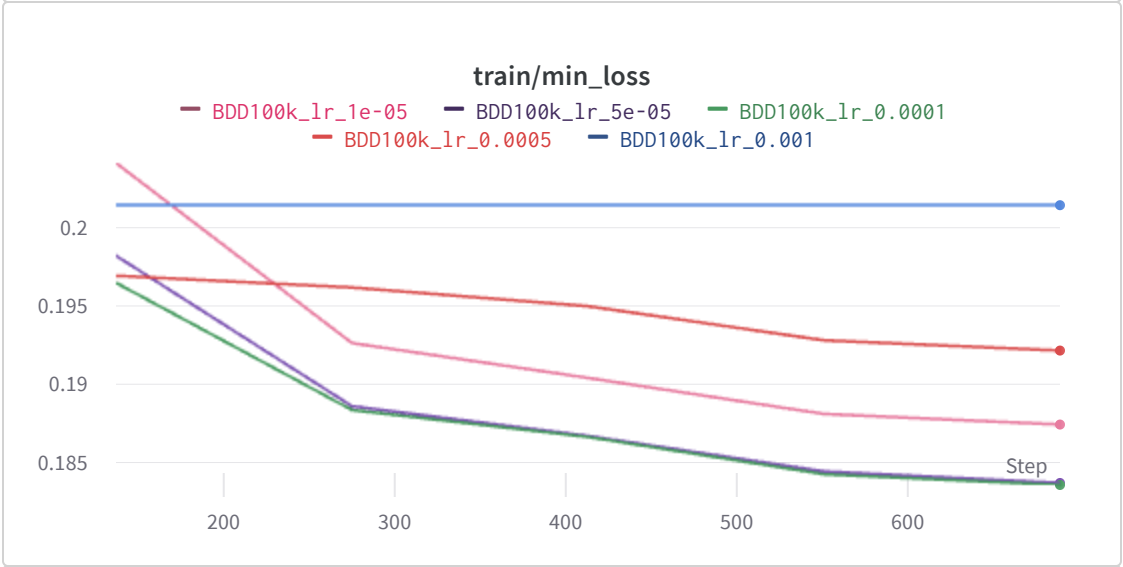


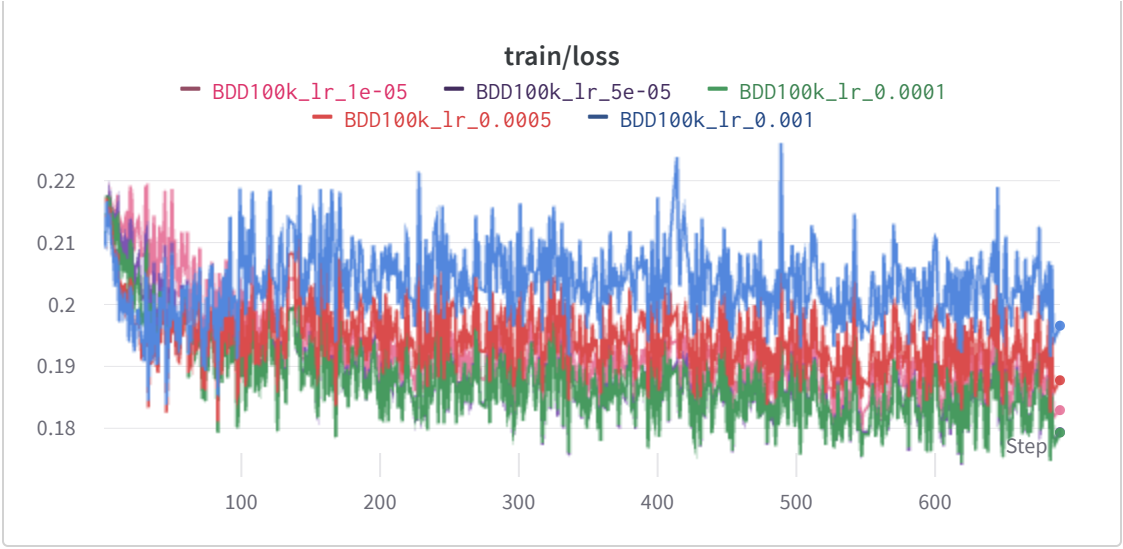


After simplifying the optimization process, we get the following results:

We see that as long as we keep the learning rate low enough, we see a healthy behaviour, if we increase it, the loss rather starts increasing on the validation dataset.

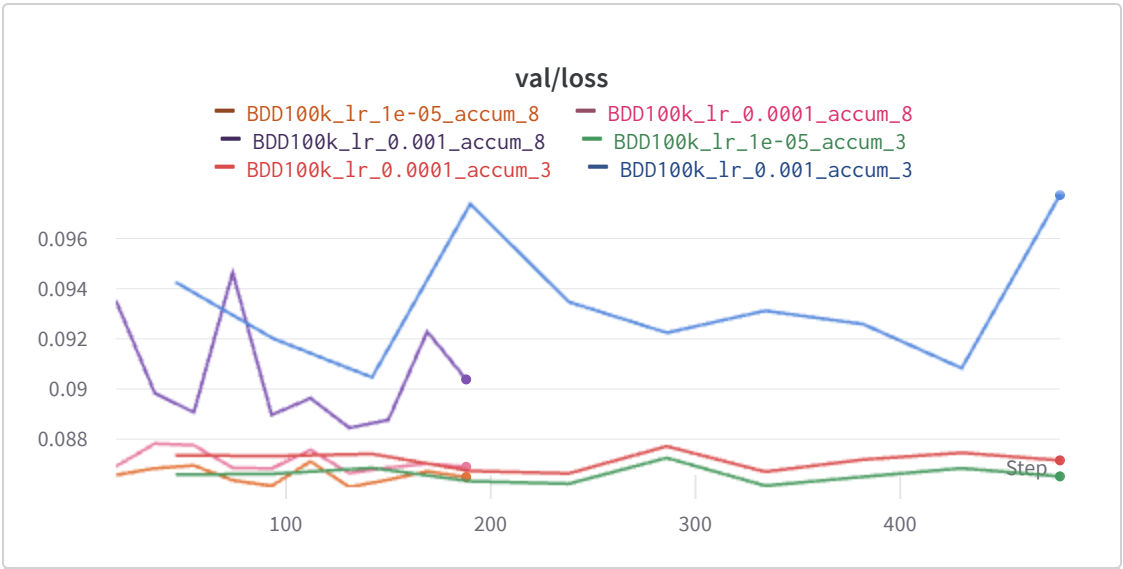


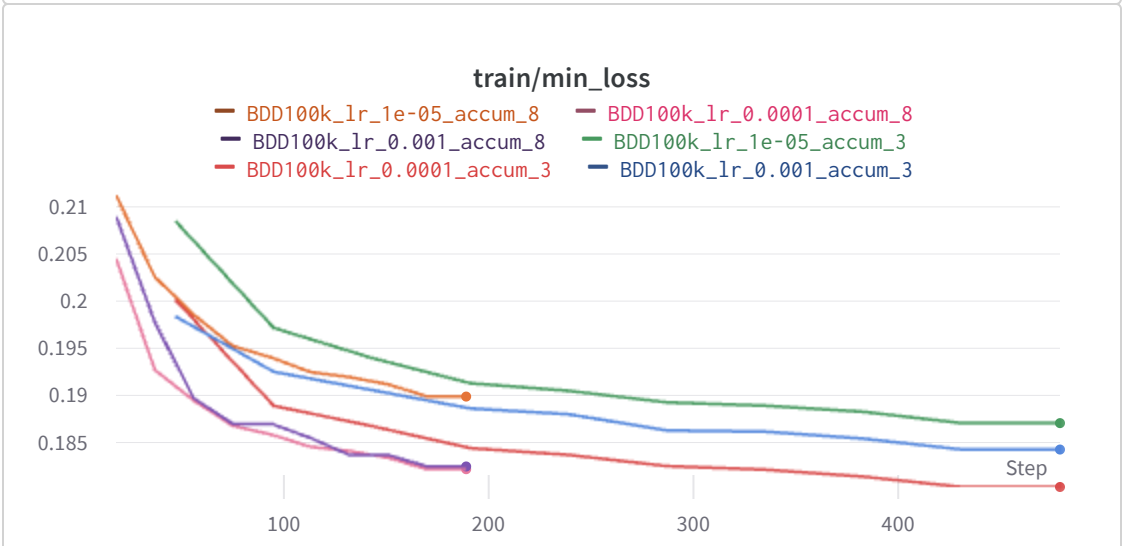


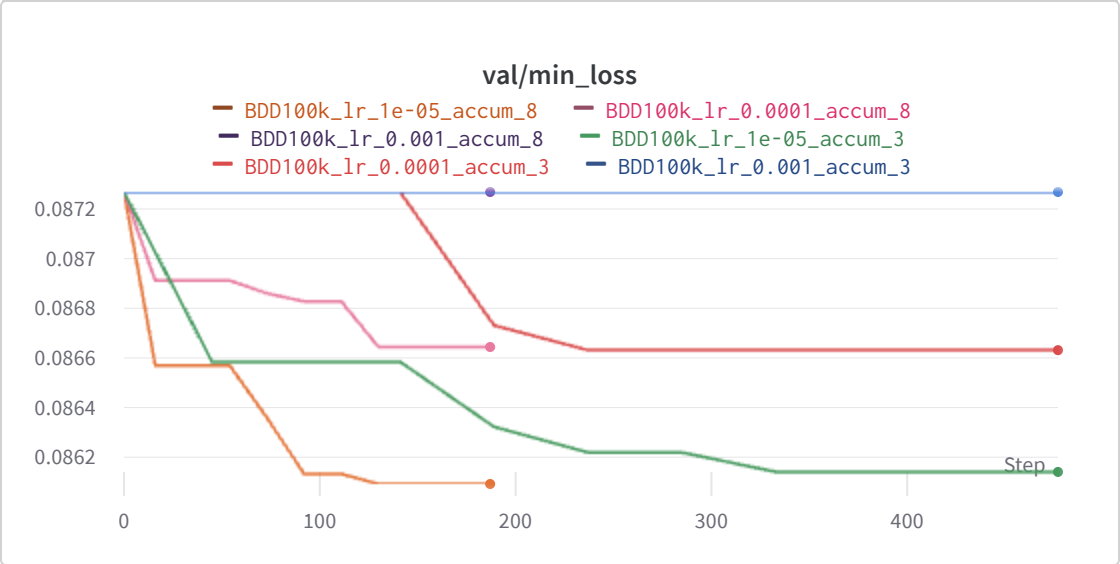


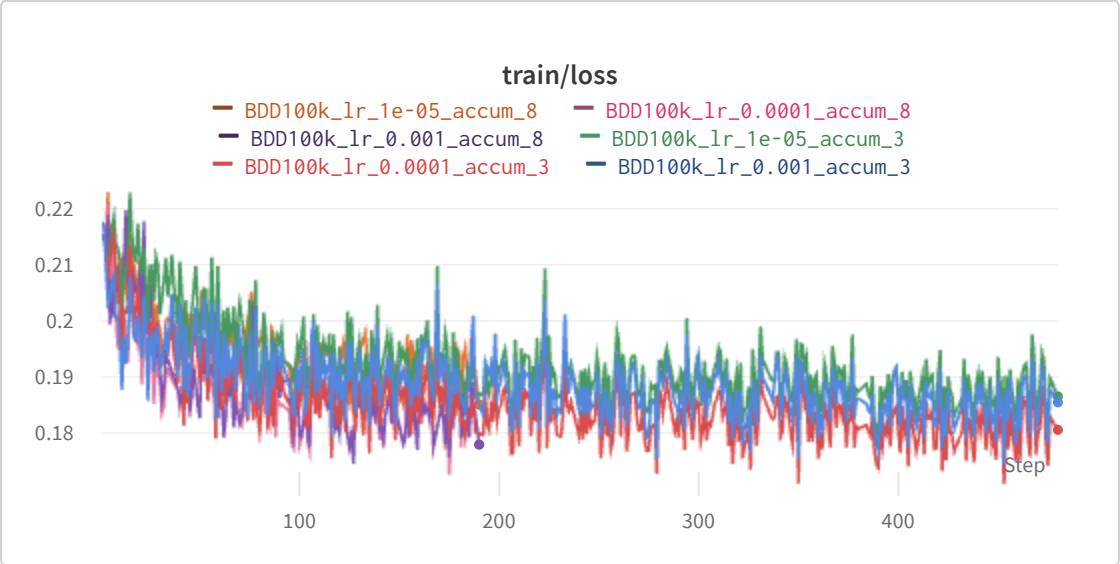
Playing around with more gradient accumulation steps (effective increasing batch size):

Same results as before, we see the smaller learning rate is important









TODO: Add images comparing pretrained/finetuned/gan models

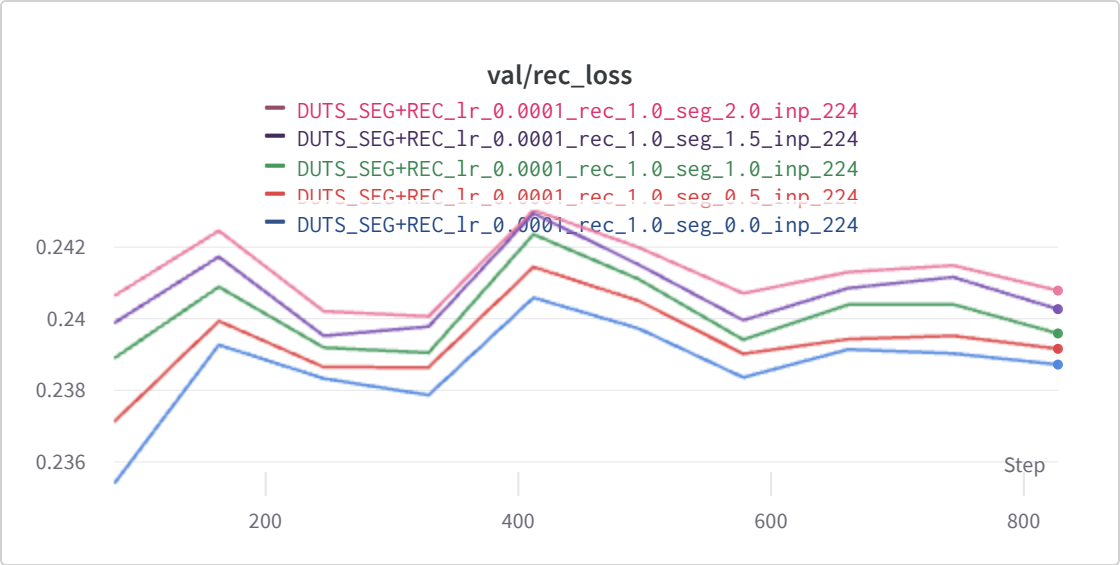
Binary segmentation training

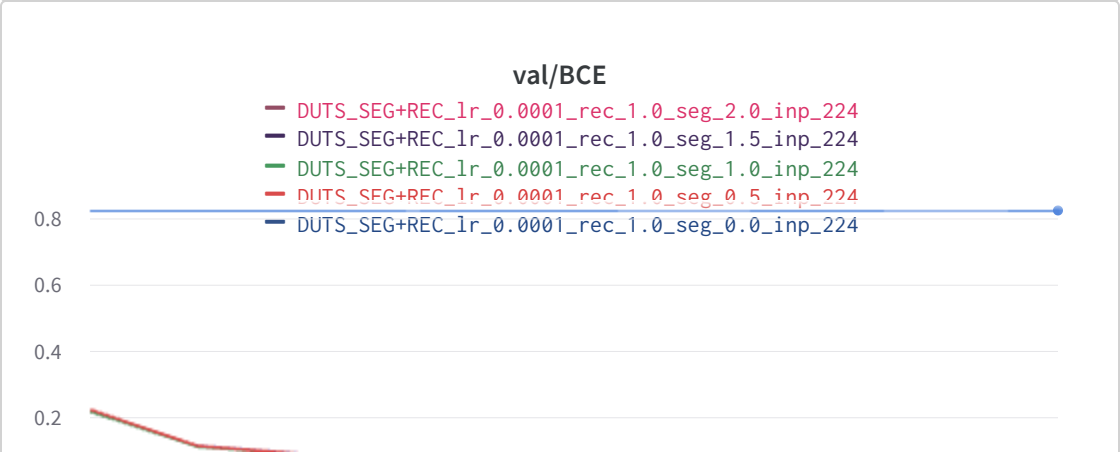
All the experiments here are on the DUTS dataset, which is based on image-net.

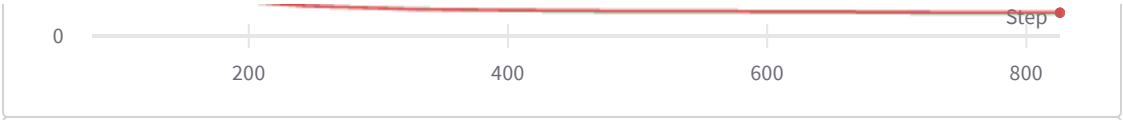
We are mostly exploring the qualitative results and how the two tasks interact.

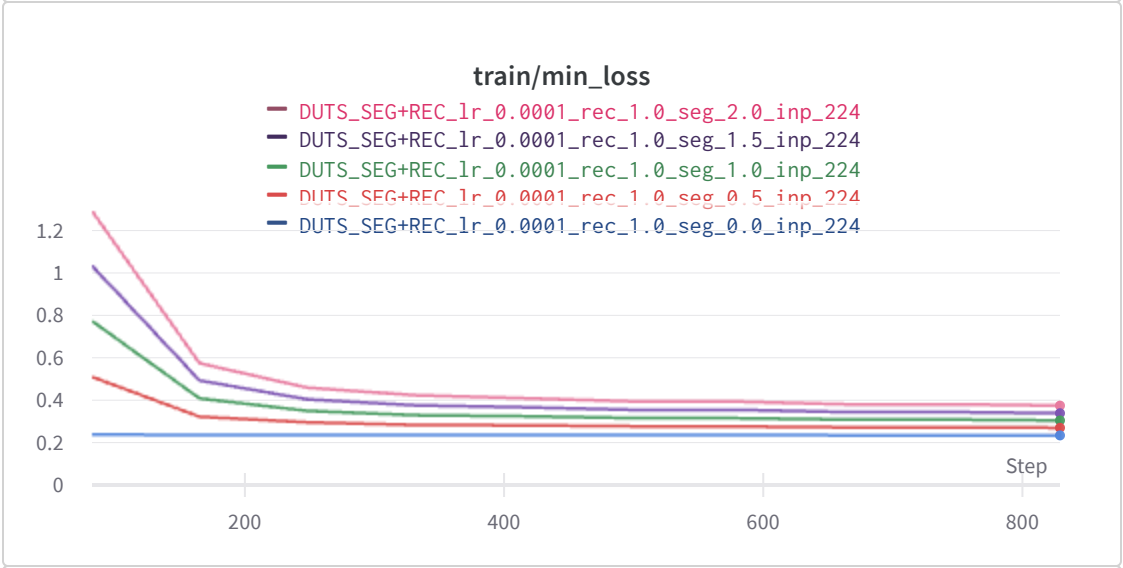
Reconstruction x segmentation loss interaction according to weight

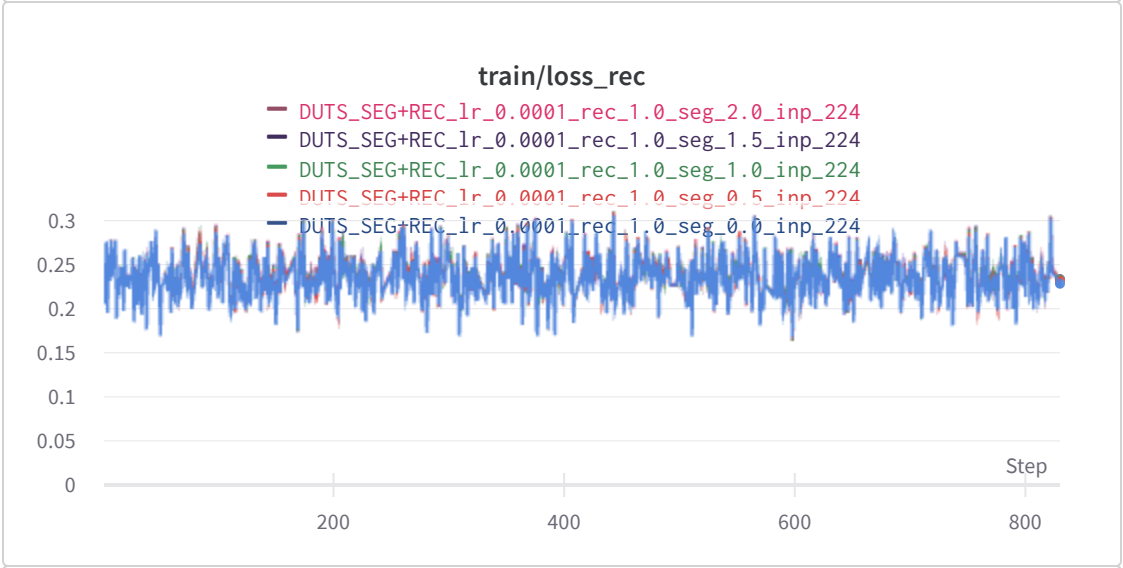
We can see that the weight doesn't really have a big effect, the reconstruction is mostly constant while the segmentation loss is always decreaisng at the same rate, as long as its weight is non-zero.

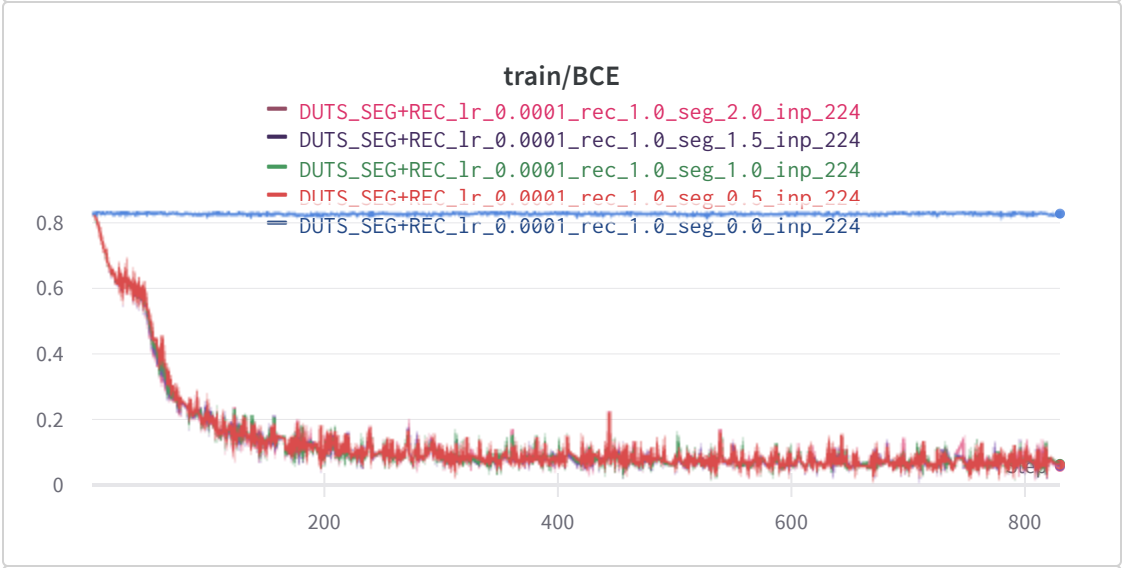




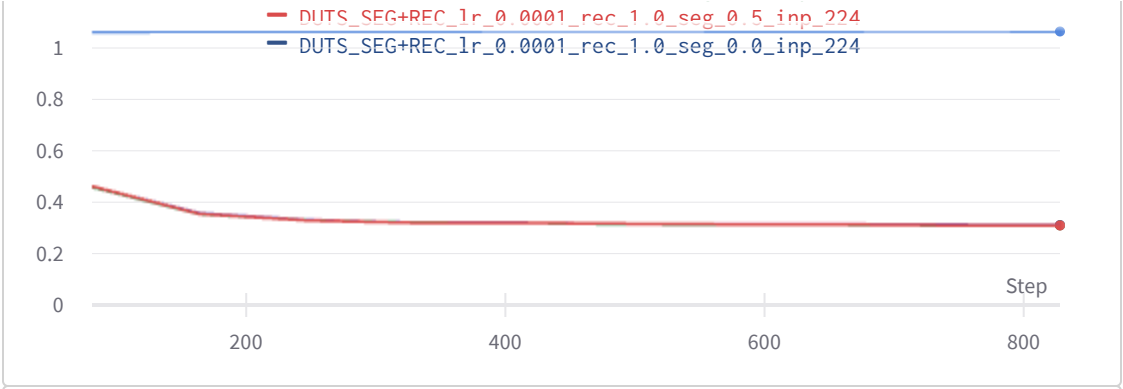

0
200
400
600
800
Step

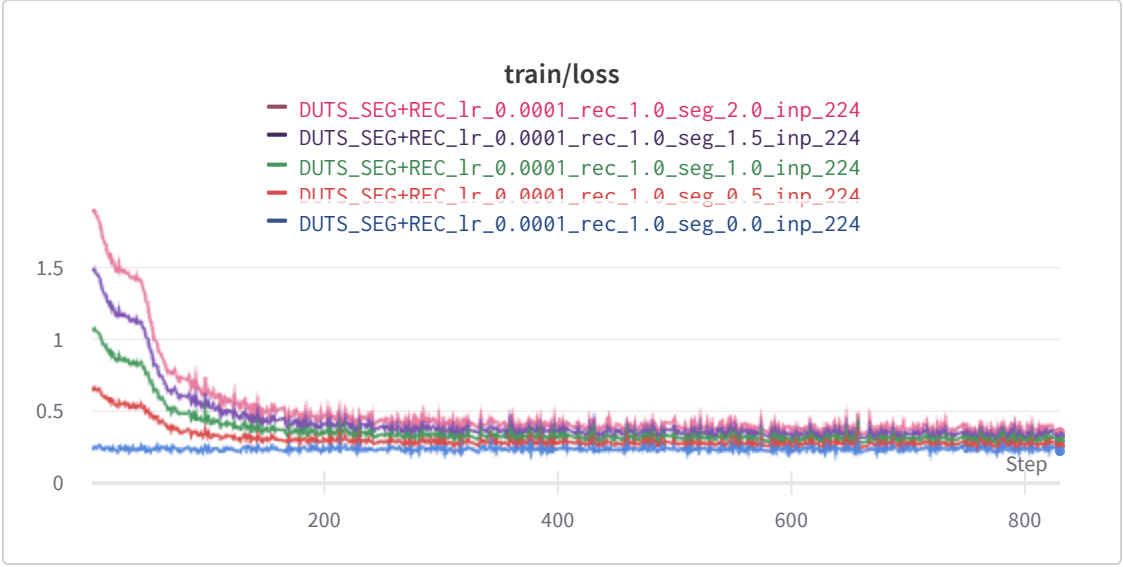

train/min_loss
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_2.0_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_1.5_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_0.5_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_0.0_inp_224
1.2
1
0.8
0.6
0.4
0.2
0
200
400
600
800
Step


train/loss_rec
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_2.0_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_1.5_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_0.5_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_0.0_inp_224
0.3
0.25
0.2
0.15
0.1
0.05
0
200
400
600
800
Step


train/BCE
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_2.0_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_1.5_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_0.5_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_0.0_inp_224
0.8
0.6
0.4
0.2
0
200
400
600
800


val/loss
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_2.0_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_1.5_inp_224
DUTS_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_224



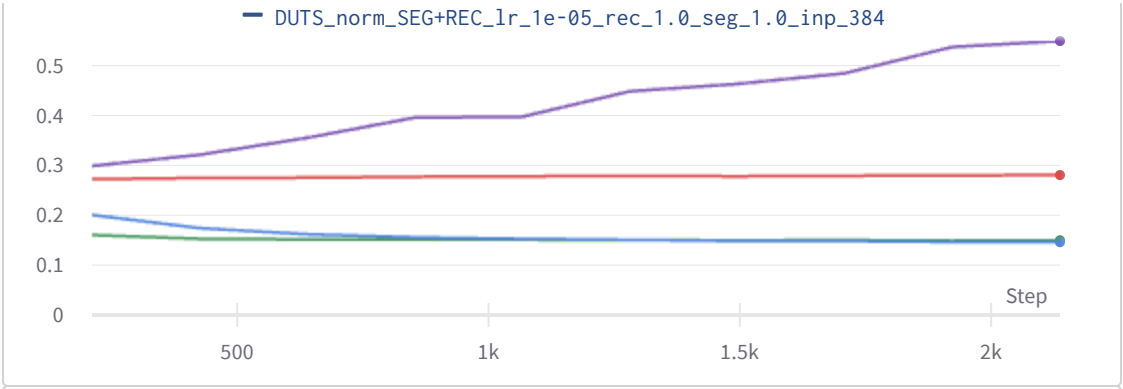


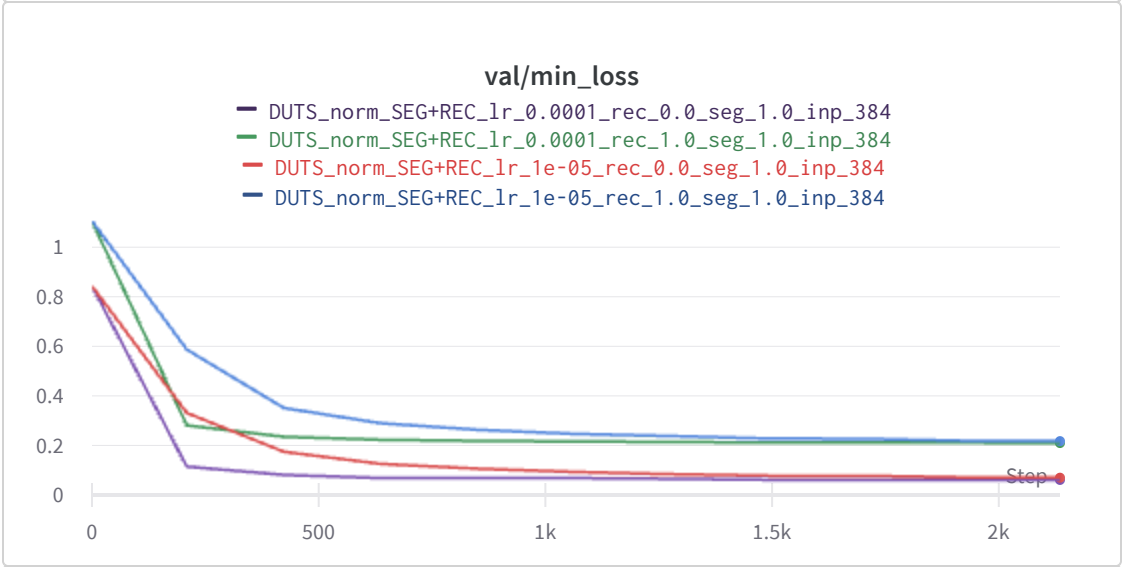
Increased resolution to 384x384

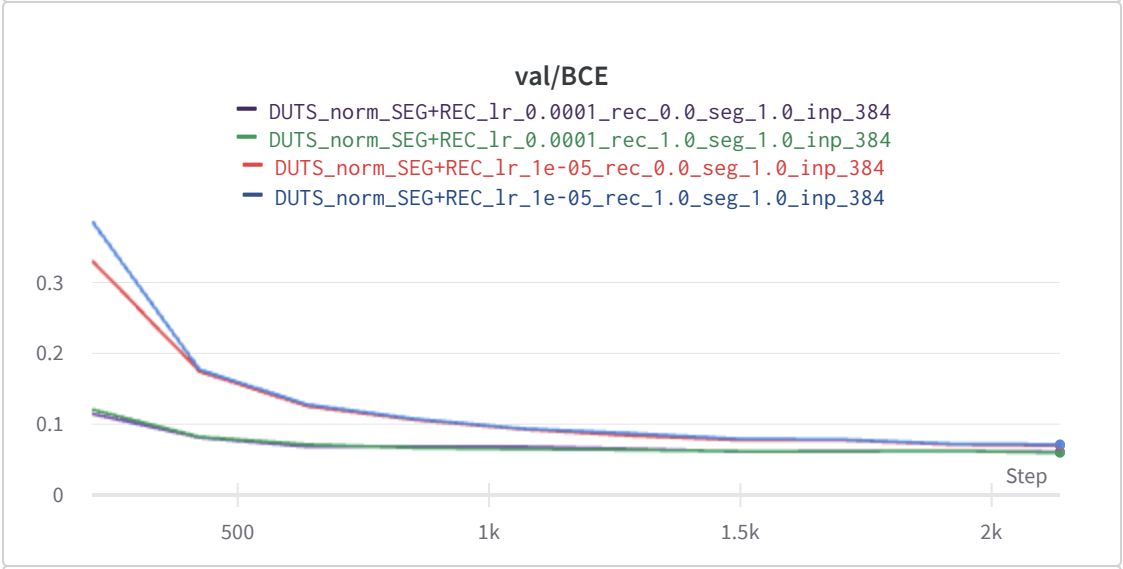
Results when increasing the resolution to 384x384:

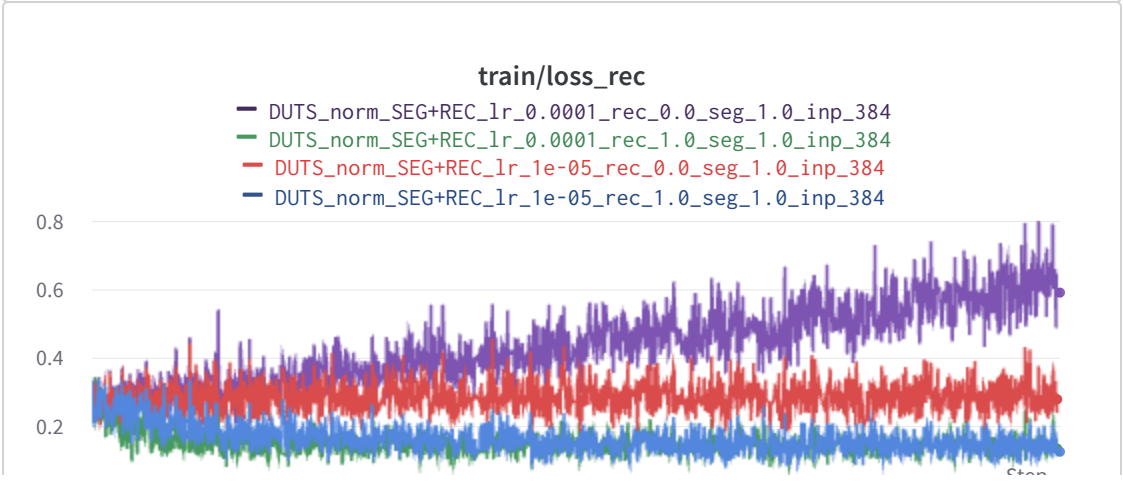
We see the results are a bit better thanks to the increased resolution, but we still have the issues with patch inconsistency, we will need to do something about the model architecture.

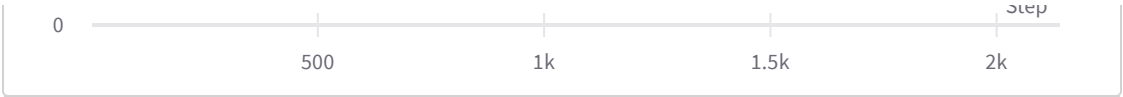
We can actually see the patch boundaries even in the reconstruction output, but it is not so visible a sit is not binary.

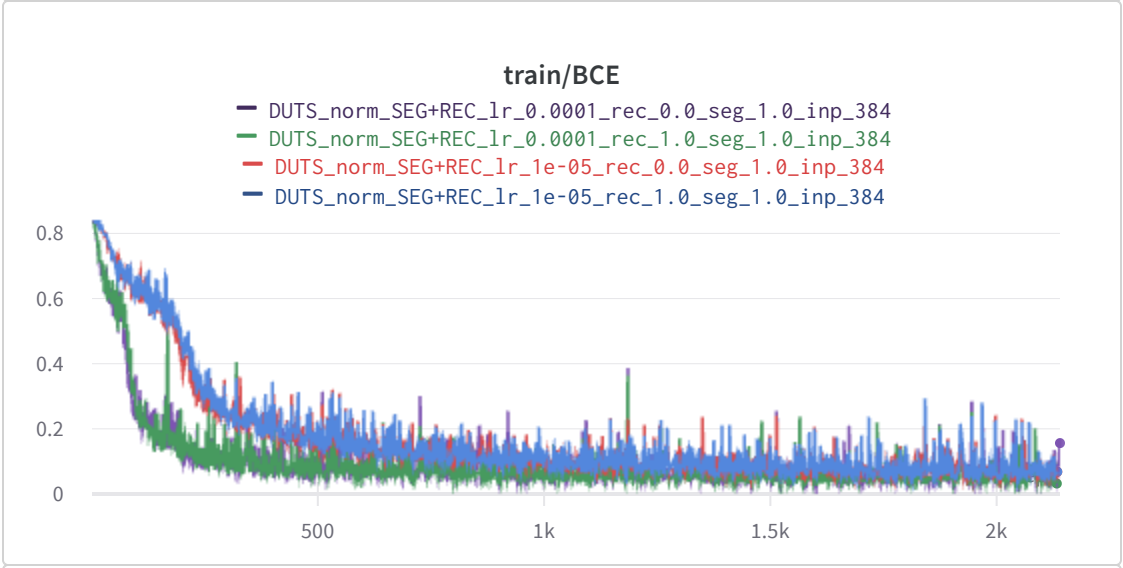

DUTS_norm_SEG+REC_lr_1e-05_rec_1.0_seg_1.0_inp_384
0.5
0.4
0.3
0.2
0.1
0
Step
500
1k
1.5k
2k

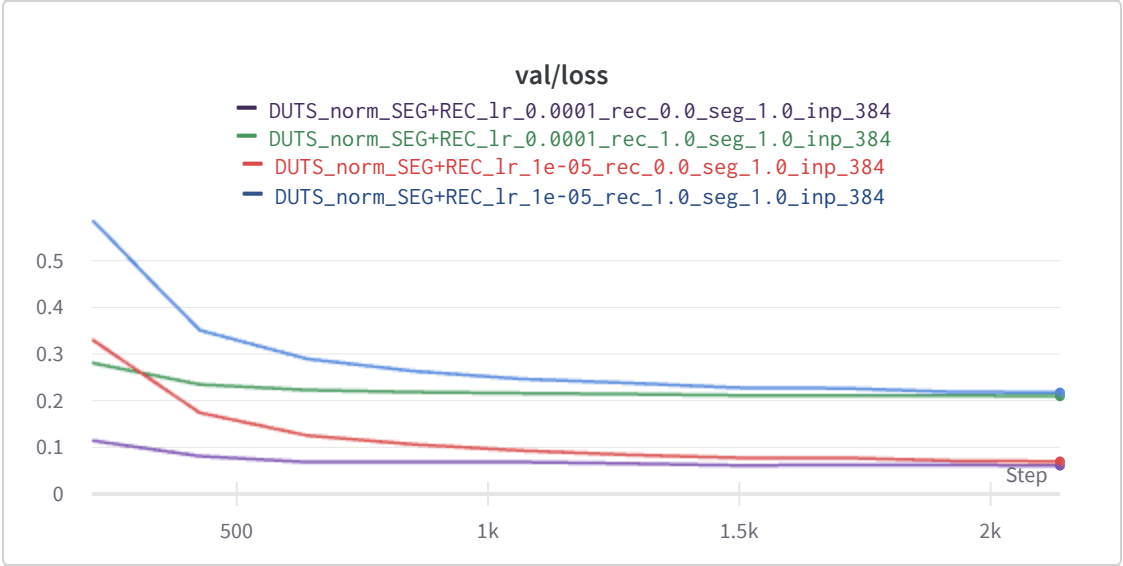

val/min_loss
DUTS_norm_SEG+REC_lr_0.0001_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_1.0_seg_1.0_inp_384
1
0.8
0.6
0.4
0.2
0
Step
0
500
1k
1.5k
2k

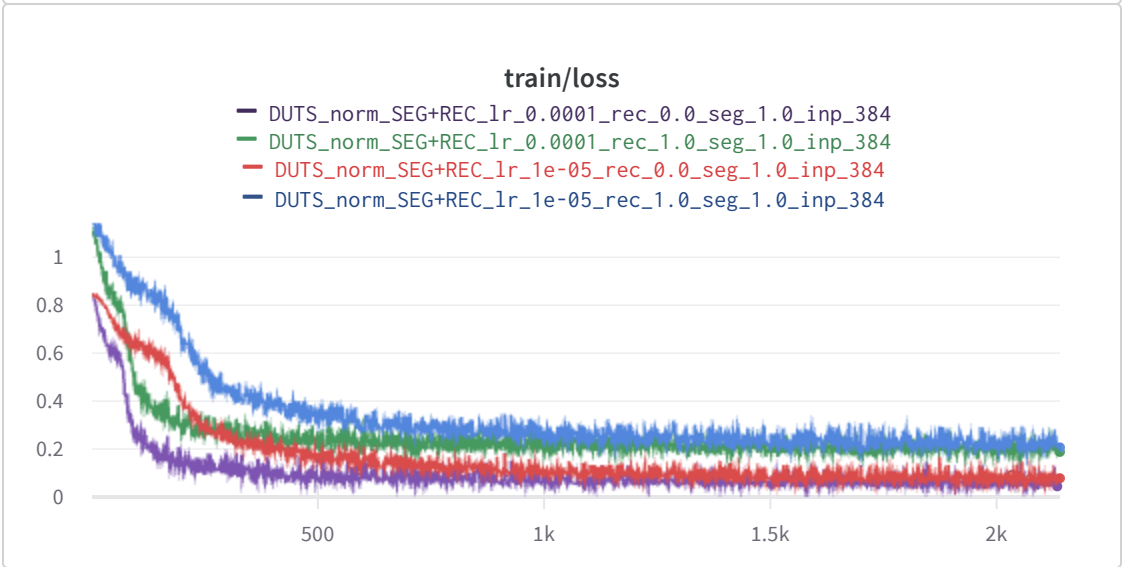

val/BCE
DUTS_norm_SEG+REC_lr_0.0001_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_1.0_seg_1.0_inp_384
0.3
0.2
0.1
0
Step
500
1k
1.5k
2k


train/loss_rec
DUTS_norm_SEG+REC_lr_0.0001_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_1.0_seg_1.0_inp_384
0.8
0.6
0.4
0.2

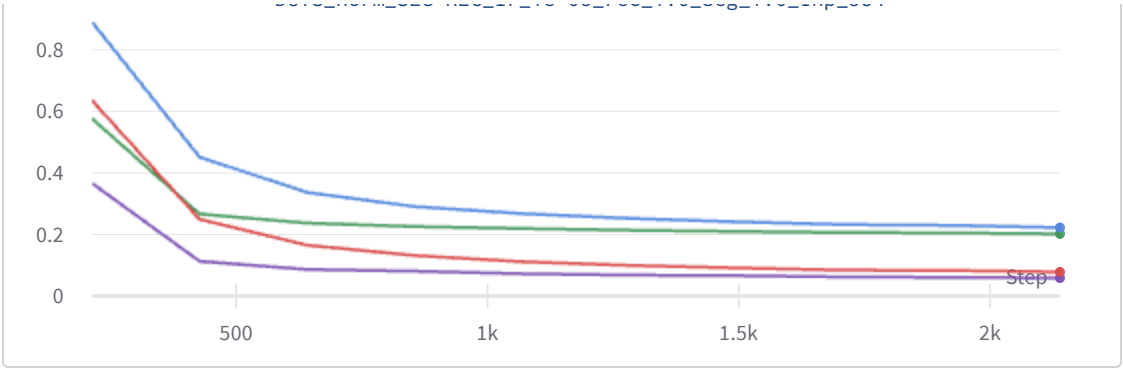

Step
0
500
1k
1.5k
2k

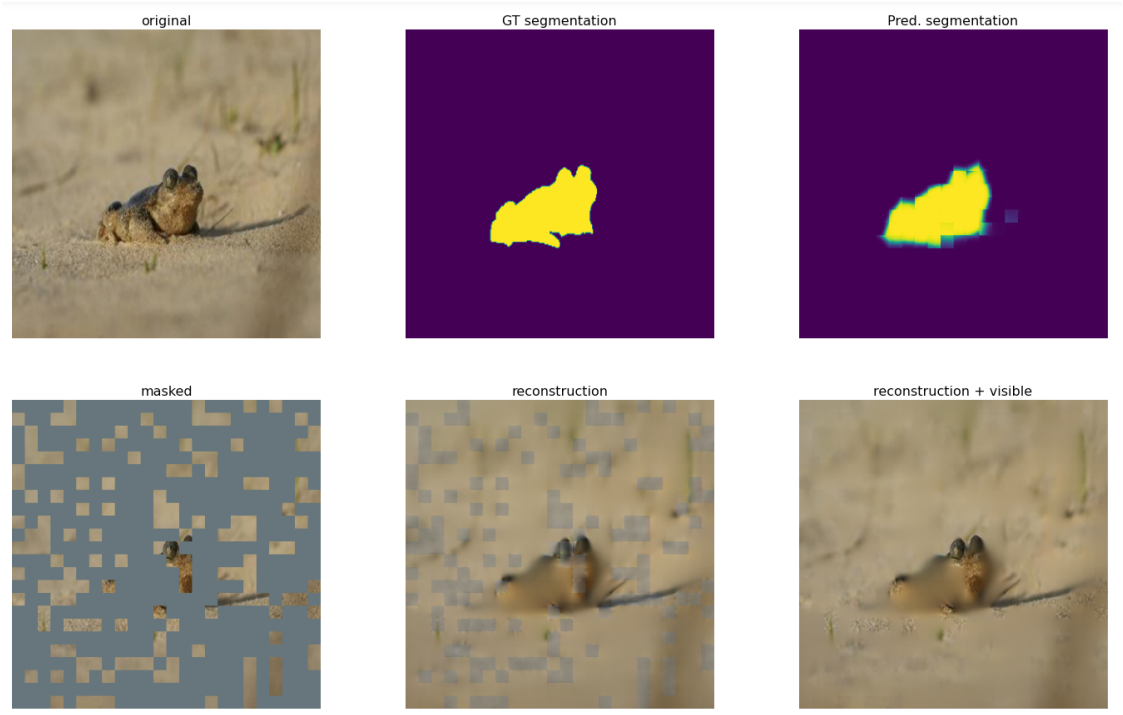

train/BCE
DUTS_norm_SEG+REC_lr_0.0001_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_1.0_seg_1.0_inp_384
0.8
0.6
0.4
0.2
0
500
1k
1.5k
2k


val/loss
DUTS_norm_SEG+REC_lr_0.0001_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_1.0_seg_1.0_inp_384
0.5
0.4
0.3
0.2
0.1
0
Step
500
1k
1.5k
2k

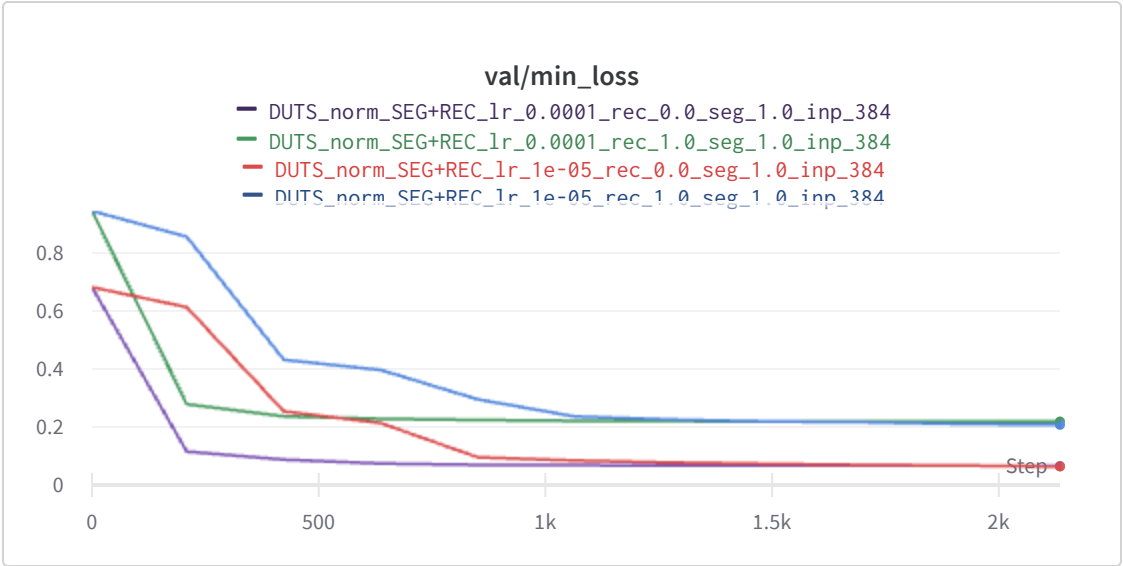

train/loss
DUTS_norm_SEG+REC_lr_0.0001_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_1.0_seg_1.0_inp_384
1
0.8
0.6
0.4
0.2
0
500
1k
1.5k
2k

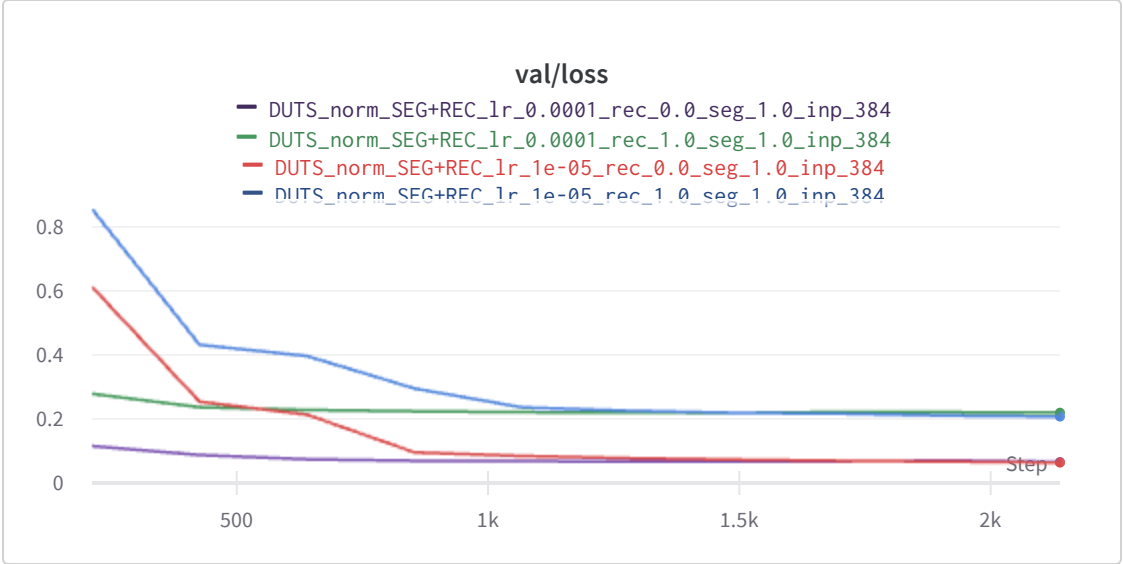

train/min_loss
DUTS_norm_SEG+REC_lr_0.0001_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_0.0_seg_1.0_inp_384

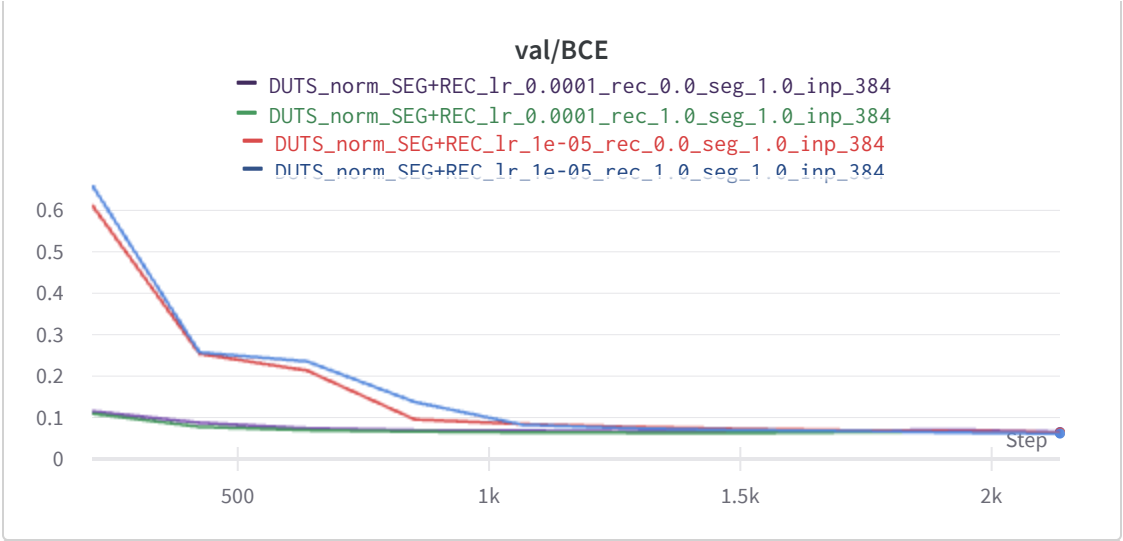

0.8
0.6
0.4
0.2
0
Step
500
1k
1.5k
2k

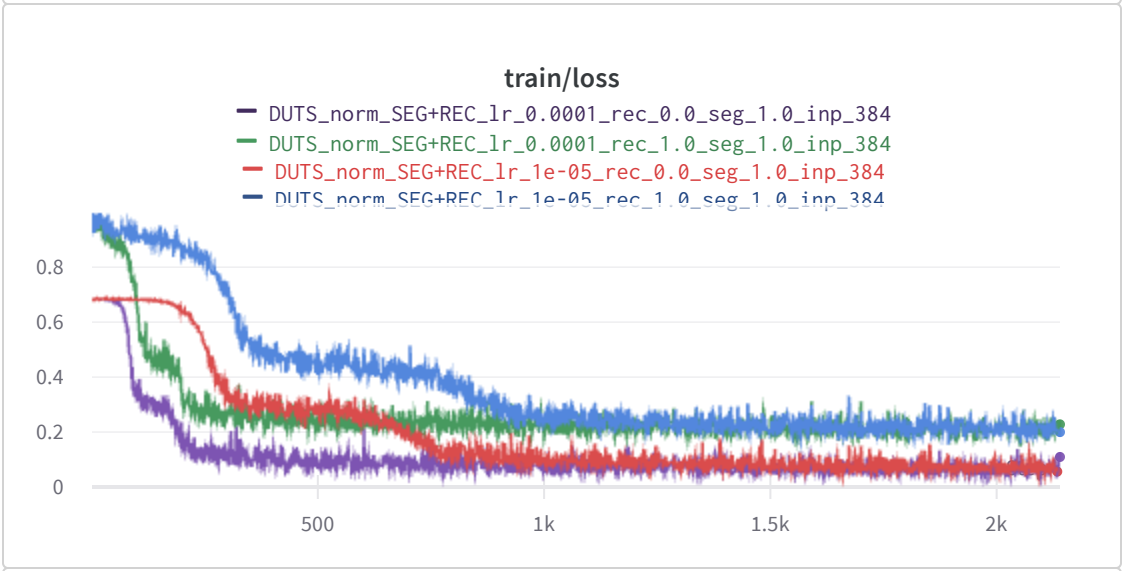

original
GT segmentation
Pred. segmentation
masked
reconstruction
reconstruction + visible

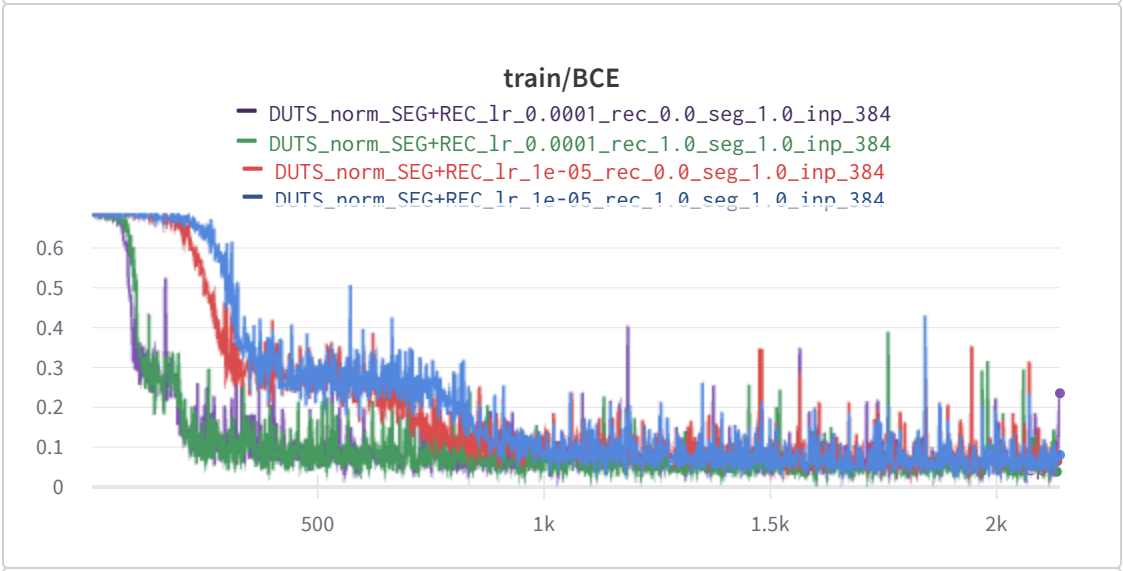
Basic Sweep with ViT + ConvNet

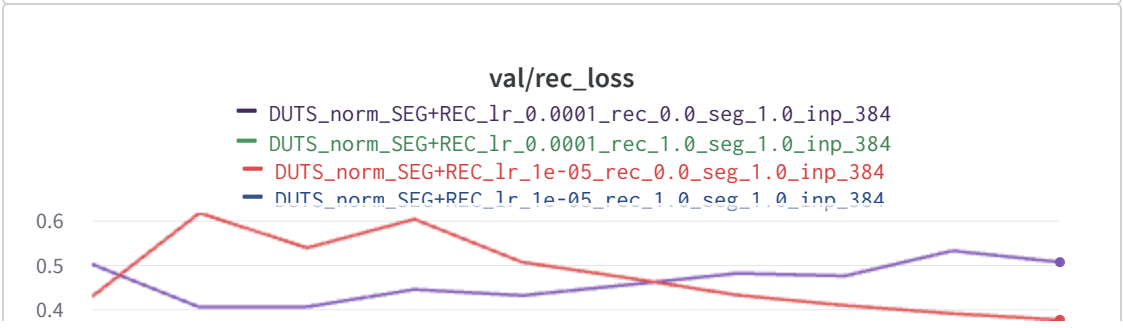


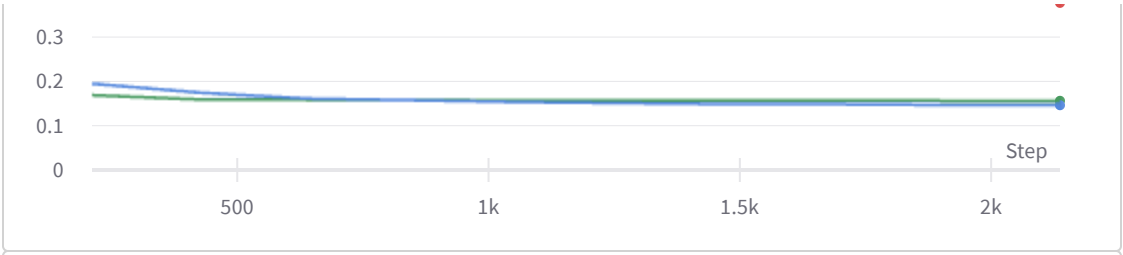


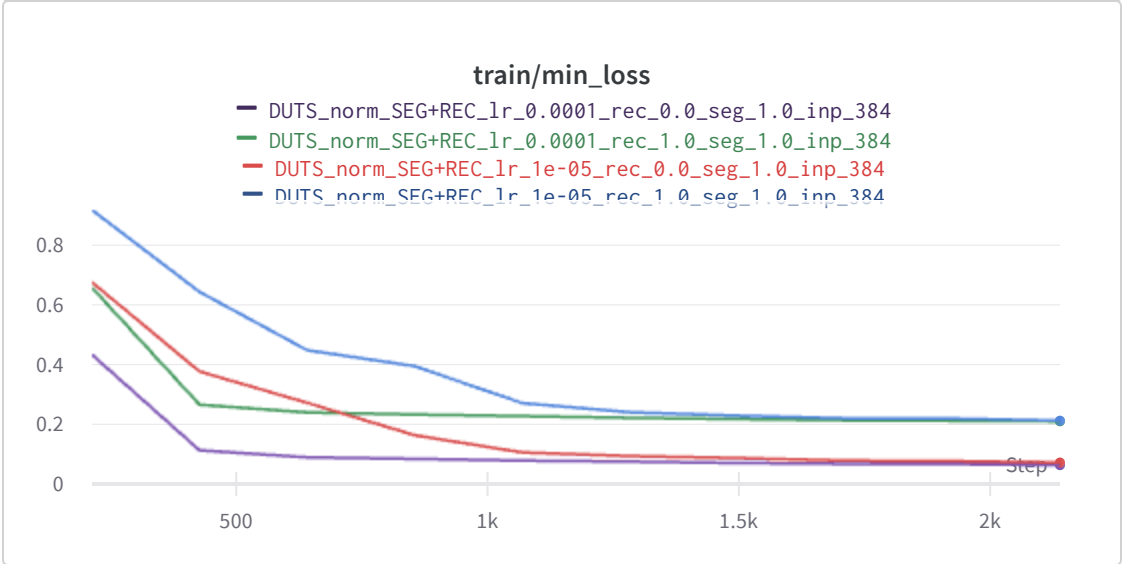

val/BCE
DUTS_norm_SEG+REC_lr_0.0001_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_1.0_seg_1.0_inp_384
0.6
0.5
0.4
0.3
0.2
0.1
0
Step
500
1k
1.5k
2k


train/loss
DUTS_norm_SEG+REC_lr_0.0001_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_1.0_seg_1.0_inp_384
0.8
0.6
0.4
0.2
0
500
1k
1.5k
2k

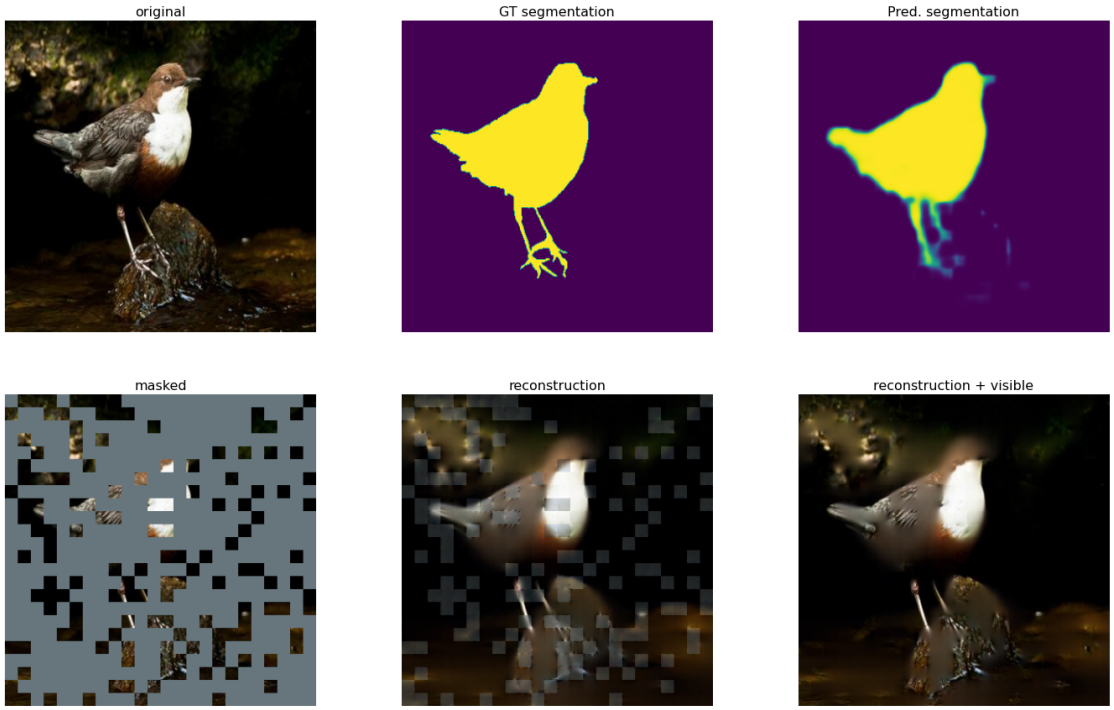

train/BCE
DUTS_norm_SEG+REC_lr_0.0001_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_1.0_seg_1.0_inp_384
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1k
1.5k
2k

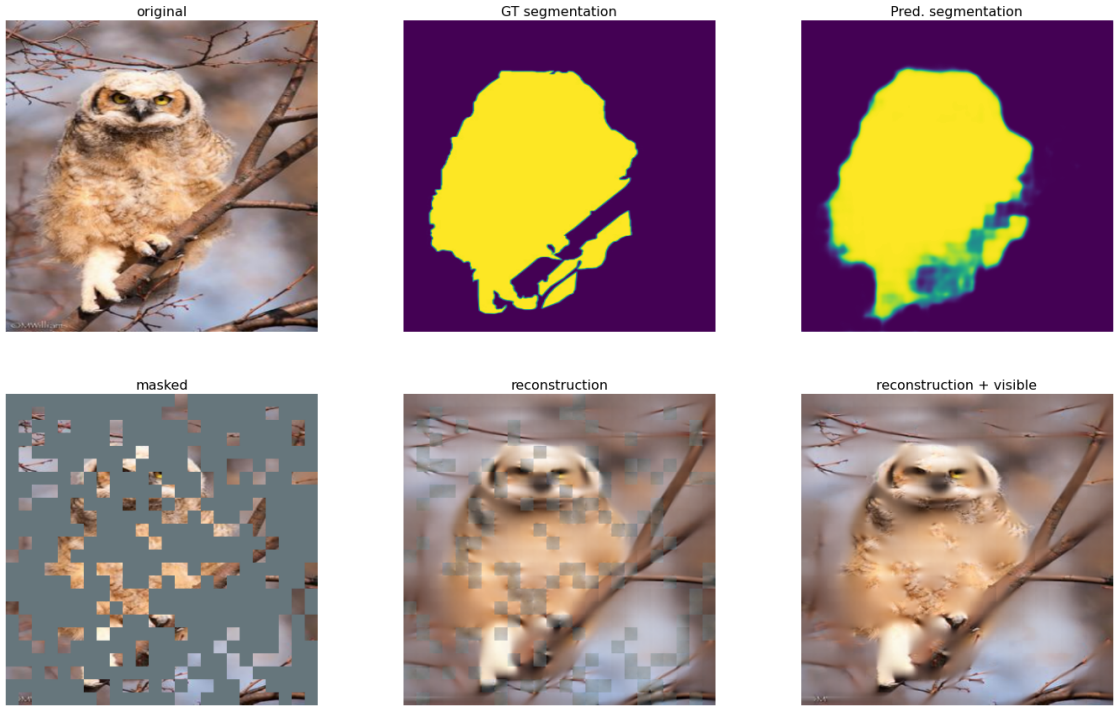

val/rec_loss
DUTS_norm_SEG+REC_lr_0.0001_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_0.0001_rec_1.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_0.0_seg_1.0_inp_384
DUTS_norm_SEG+REC_lr_1e-05_rec_1.0_seg_1.0_inp_384
0.6
0.5
0.4





Ae can still see some artifacts due to patches, but it has improved a lot. We might want to try a deeper convnet (should help with performance, the transformer decoder wa smuch bigger), but most likely, we need to add a skip ocnnection from the input image to the convnet part.





Created with ❤️ on Weights & Biases.

https://wandb.ai/klara/MAE-finetune/reports/Exploring-finetuning-ViT-MAEs-for-fg-bg-segmentation--VmlldzoyOTY5NjM4