**Important Details**

A CIFAR-10 picture is 32\*32 pixels. However, in the TransGan paper we start off with a smaller size. Specifically,

First Block (5 layers)– 8x8 = 64 “pixels”

Second Block (4 layers) – 16x16 = 256 “pixels”

Third Block (2 layers) – 32x32 = 1024 “pixels”

We start off with a 8x8 image and then work upwards to a 32x32 image.

Until now, I haven’t tried masking any of the layers in the first block. I have only masked the layers from the second and third block.

**Legend**

TG = Only temporal masking

TG-LM [16, 8] = Signifies masking area (16\*16 and 8\*8 for 2nd and 3rd blocks) for get\_2d\_attn\_mask()

TG-LM [8, 4] row-wise: Simply mask linearly in the flattened input (i.e., for example instead of un-masking a square of 4x4 pixels, we simply un-mask 16 pixels from that row). Uses get\_attn\_mask()

TG-LM [8, 4] random: Random un-masking of w\*\*2 other pixels from the image. Uses get\_random\_attn\_mask()

Train model 1: Test model 2 = Trained using the model 1 setting but tested with model 2 architecture

**Hyperparameters –** Batch size100 for generator, 50 for discriminator. Nothing else was changed (e.g., learning rate). Latent size used was 1024.

**Results until now**

TG (epoch 120, did not complete the training) = 28.977 FID

TG-LM [8, 4] = 46.023 FID

TG-LM [8, 8] = 25.04 FID

Below 3 rows shows that my method of square un-masking works best compared to the row-wise un-masking or some form of random un-masking. Shows un-masking the locality of each pixel is the most effective.

Train TG: Test TG-LM [8, 4] = 34.801

Train TG: TG-LM [8, 4] row-wise = 46.96

Train TG: TG-LM [8, 4] random = 129.79

Train TG: Test TG-LM [8, 8] = 29.68

Train TG: Test TG-LM [16, 8] = 29.39

Train TG-LM [8, 4]: Test TG-LM [16, 8] = 96.2337

Train TG-LM [8, 8]: Test TG = 78.xxx (cannot remember, easy to check)

**Saved Models**

Checkpoint\_1024 – Contains the TG model (at 120 epochs)

Layermask\_8\_4\_checkpoint – Contains the TG-LM [8, 4] model

Layermask\_8\_8\_checkpoint – Contains the TG-LM [8, 8] model

**How to use the code**

./models/ TransGAN\_8\_8\_1\_layermask.py – All 3 methods of masking code is implemented (get\_2d\_attn\_mask, get\_attn\_mask, get\_random\_attn\_mask) . Get\_2d\_attention\_mask(N, w) un-masks a square of width w for each pixel, where N refers to the size of the entire image at that layer. These values can be changed the self.upsample\_blocks section. The first block is contained in the self.blocks variable.

./exps/cifar10\_test.sh – Just change the gen\_model parameter to use whichever model you want to use for inference. Any of the weights can be loaded. So, for example Train TG: Test TG-LM [8, 4] setting, the weights would be from the TG checkpoint, but the model used would be the TG-LM checkpoint.

./exps/cifar10\_train\_layermask.sh – Contains the training script for running the layermask model. Not much change from cifar10\_train.sh.

**Additions**

Currently, can only seem to get the FID score and not the IS score. I am sure the code for that is written somewhere, so maybe we could get the model.

Overall, I think its best to list the type of models you want to train because you can test it on several different setting later on with ease.

Out-of-place results (Probably cannot be used)

TG [5,4, 2] 384-d = 30.9 FID