# CS597 REPORT 7
# Yash Pradeep Gupte
# MS Computer Science
# A20472798

## 1] TransGAN: Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up by ViTA group

**Repository link: [https://github.com/VITA-Group/TransGAN](https://github.com/VITA-Group/TransGAN)**

## (A) Datasets

- **Cifar10 (32*32)**
- **STL -10 (48*48)**
- **CelebA (128*128)**

**Generator and Discriminator models for Cifar10 dataset**

## (B) ViT_custom_rp.py -> contains generator and discriminator models

Parameters for test.py -> -gen_bs 128 -dis_bs 64 --world-size 1 --dataset cifar10 --bottom_width 8 --img_size 32 --max_iter 500000 --gen_model ViT_custom_rp --dis_model ViT_custom_scale2_rp_noise --df_dim 384 --d_heads 4 --d_depth 3 --g_depth 5,4,2 --dropout 0 --latent_dim 256 --gf_dim 1024 --num_workers 0 --g_lr 0.0001 --d_lr 0.0001 --optimizer adam --loss wgangp-eps --wd 1e-3 --beta1 0 --beta2 0.99 --phi 1 --eval_batch_size 8 --num_eval_imgs 50000 --init_type xavier_uniform --n_critic 4 --val_freq 20 --print_freq 50 --fade_in 0 --patch_size 2 --ema_kimg 500 --ema_warmup 0.1 --ema 0.9999 --diff_aug translation,cutout,color --load_path ./exps/cifar_checkpoint --exp_name cifar_train

Test args ->(D_downsample='avg', accumulated_times=1, arch=None, baseline_decay=0.9, beta1=0.0, beta2=0.99, bottom_width=8, channels=3, controller='controller', ctrl_lr=0.00035, ctrl_sample_batch=1, ctrl_step=30, d_act='gelu', d_depth=3, d_heads=4, d_lr=0.0001, d_mlp=4, d_norm='ln', d_spectral_norm=False, d_window_size=8, data_path='./data', dataset='cifar10', df_dim=384, diff_aug='translation,cutout,color', dis_batch_size=64, dis_model='ViT_custom_scale2_rp_noise', dist_backend='nccl', dist_url='tcp://224.66.41.62:23456', dropout=0.0, dynamic_reset_threshold=0.001, dynamic_reset_window=500, ema=0.9999, ema_kimg=500, ema_warmup=0.1, entropy_coeff=0.001, eval_batch_size=8, exp_name='cifar_train', fade_in=0.0, fid_stat='None', g_accumulated_times=1, g_act='gelu', g_depth='5,4,2', g_lr=0.0001, g_mlp=4, g_norm='ln', g_spectral_norm=False, g_window_size=8, gen_batch_size=128, gen_model='ViT_custom_rp', gf_dim=1024, gpu=None, grow_step1=25, grow_step2=55, grow_steps=None, hid_size=100, img_size=32, init_type='xavier_uniform', latent_dim=256, latent_norm=False,

load_path='./exps/cifar_checkpoint', loca_rank=-1, loss='wgangp-eps', lr_decay=False, max_epoch=200, max_iter=500000, max_search_iter=90, ministd=False, multiprocessing_distributed=False, n_classes=0, n_critic=4, num_candidate=10, num_eval_imgs=50000, num_landmarks=64, num_workers=0, optimizer='adam', patch_size=2, path_helper={'prefix': 'logs_eval/cifar_train_2021_11_19_11_02_45', 'ckpt_path': 'logs_eval/cifar_train_2021_11_19_11_02_45/Model', 'log_path': 'logs_eval/cifar_train_2021_11_19_11_02_45/Log', 'sample_path': 'logs_eval/cifar_train_2021_11_19_11_02_45/Samples'}, phi=1.0, print_freq=50, random_seed=12345, rank=-1, rl_num_eval_img=5000, seed=12345, shared_epoch=15, show=False, topk=5, val_freq=20, wd=0.001, world_size=1)

***(i) Generator ->***

Act_layer = gelu , attn_drop_rate=0.0, depth=[5,4,2],embed_dim =1024, img_size=224, in_chans = 3 , mlp_ratio = 4 ,norm_layer='ln', num_classes=10, num_heads=4, patch_size=16, qk_scale = None, qkv_bias = False, optimizer = adam,

```
Generator(
  (l1): Linear(in_features=256, out_features=65536, bias=True)
  (blocks): StageBlock(
    (block): ModuleList(
      (0): Block(
        (norm1): CustomNorm(
          (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (attn): Attention(
          (qkv): Linear(in_features=1024, out_features=3072, bias=False)
          (attn_drop): Dropout(p=0.0, inplace=False)
          (proj): Linear(in_features=1024, out_features=1024, bias=True)
          (proj_drop): Dropout(p=0.0, inplace=False)
          (mat): matmul()
        )
        (drop_path): Identity()
        (norm2): CustomNorm(
          (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (mlp): Mlp(
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (act): CustomAct()
```

```
    (fc2): Linear(in_features=4096, out_features=1024, bias=True)
    (drop): Dropout(p=0.0, inplace=False)
  )
)
(1): Block(
  (norm1): CustomNorm(
    (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  )
  (attn): Attention(
    (qkv): Linear(in_features=1024, out_features=3072, bias=False)
    (attn_drop): Dropout(p=0.0, inplace=False)
    (proj): Linear(in_features=1024, out_features=1024, bias=True)
    (proj_drop): Dropout(p=0.0, inplace=False)
    (mat): matmul()
  )
  (drop_path): Identity()
  (norm2): CustomNorm(
    (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  )
  (mlp): Mlp(
    (fc1): Linear(in_features=1024, out_features=4096, bias=True)
    (act): CustomAct()
    (fc2): Linear(in_features=4096, out_features=1024, bias=True)
    (drop): Dropout(p=0.0, inplace=False)
  )
)
(2): Block(
  (norm1): CustomNorm(
    (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  )
  (attn): Attention(
    (qkv): Linear(in_features=1024, out_features=3072, bias=False)
    (attn_drop): Dropout(p=0.0, inplace=False)
    (proj): Linear(in_features=1024, out_features=1024, bias=True)
    (proj_drop): Dropout(p=0.0, inplace=False)
    (mat): matmul()
  )
  (drop_path): Identity()
  (norm2): CustomNorm(
    (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  )
  (mlp): Mlp(
    (fc1): Linear(in_features=1024, out_features=4096, bias=True)
    (act): CustomAct()
```

```
    (fc2): Linear(in_features=4096, out_features=1024, bias=True)
    (drop): Dropout(p=0.0, inplace=False)
  )
)
(3): Block(
  (norm1): CustomNorm(
    (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  )
  (attn): Attention(
    (qkv): Linear(in_features=1024, out_features=3072, bias=False)
    (attn_drop): Dropout(p=0.0, inplace=False)
    (proj): Linear(in_features=1024, out_features=1024, bias=True)
    (proj_drop): Dropout(p=0.0, inplace=False)
    (mat): matmul()
  )
  (drop_path): Identity()
  (norm2): CustomNorm(
    (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  )
  (mlp): Mlp(
    (fc1): Linear(in_features=1024, out_features=4096, bias=True)
    (act): CustomAct()
    (fc2): Linear(in_features=4096, out_features=1024, bias=True)
    (drop): Dropout(p=0.0, inplace=False)
  )
)
(4): Block(
  (norm1): CustomNorm(
    (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  )
  (attn): Attention(
    (qkv): Linear(in_features=1024, out_features=3072, bias=False)
    (attn_drop): Dropout(p=0.0, inplace=False)
    (proj): Linear(in_features=1024, out_features=1024, bias=True)
    (proj_drop): Dropout(p=0.0, inplace=False)
    (mat): matmul()
  )
  (drop_path): Identity()
  (norm2): CustomNorm(
    (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  )
  (mlp): Mlp(
    (fc1): Linear(in_features=1024, out_features=4096, bias=True)
    (act): CustomAct()
```

```
        (fc2): Linear(in_features=4096, out_features=1024, bias=True)
        (drop): Dropout(p=0.0, inplace=False)
      )
    )
  )
)
(upsample_blocks): ModuleList(
  (0): StageBlock(
    (block): ModuleList(
      (0): Block(
        (norm1): CustomNorm(
          (norm): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
        )
        (attn): Attention(
          (qkv): Linear(in_features=256, out_features=768, bias=False)
          (attn_drop): Dropout(p=0.0, inplace=False)
          (proj): Linear(in_features=256, out_features=256, bias=True)
          (proj_drop): Dropout(p=0.0, inplace=False)
          (mat): matmul()
        )
        (drop_path): Identity()
        (norm2): CustomNorm(
          (norm): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
        )
        (mlp): Mlp(
          (fc1): Linear(in_features=256, out_features=1024, bias=True)
          (act): CustomAct()
          (fc2): Linear(in_features=1024, out_features=256, bias=True)
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
      (1): Block(
        (norm1): CustomNorm(
          (norm): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
        )
        (attn): Attention(
          (qkv): Linear(in_features=256, out_features=768, bias=False)
          (attn_drop): Dropout(p=0.0, inplace=False)
          (proj): Linear(in_features=256, out_features=256, bias=True)
          (proj_drop): Dropout(p=0.0, inplace=False)
          (mat): matmul()
        )
        (drop_path): Identity()
        (norm2): CustomNorm(
```

```
    (norm): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
  )
  (mlp): Mlp(
    (fc1): Linear(in_features=256, out_features=1024, bias=True)
    (act): CustomAct()
    (fc2): Linear(in_features=1024, out_features=256, bias=True)
    (drop): Dropout(p=0.0, inplace=False)
  )
)
(2): Block(
  (norm1): CustomNorm(
    (norm): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
  )
  (attn): Attention(
    (qkv): Linear(in_features=256, out_features=768, bias=False)
    (attn_drop): Dropout(p=0.0, inplace=False)
    (proj): Linear(in_features=256, out_features=256, bias=True)
    (proj_drop): Dropout(p=0.0, inplace=False)
    (mat): matmul()
  )
  (drop_path): Identity()
  (norm2): CustomNorm(
    (norm): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
  )
  (mlp): Mlp(
    (fc1): Linear(in_features=256, out_features=1024, bias=True)
    (act): CustomAct()
    (fc2): Linear(in_features=1024, out_features=256, bias=True)
    (drop): Dropout(p=0.0, inplace=False)
  )
)
(3): Block(
  (norm1): CustomNorm(
    (norm): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
  )
  (attn): Attention(
    (qkv): Linear(in_features=256, out_features=768, bias=False)
    (attn_drop): Dropout(p=0.0, inplace=False)
    (proj): Linear(in_features=256, out_features=256, bias=True)
    (proj_drop): Dropout(p=0.0, inplace=False)
    (mat): matmul()
  )
  (drop_path): Identity()
  (norm2): CustomNorm(
```

```
      (norm): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
    )
    (mlp): Mlp(
      (fc1): Linear(in_features=256, out_features=1024, bias=True)
      (act): CustomAct()
      (fc2): Linear(in_features=1024, out_features=256, bias=True)
      (drop): Dropout(p=0.0, inplace=False)
    )
  )
 )
)
(1): StageBlock(
 (block): ModuleList(
  (0): Block(
    (norm1): CustomNorm(
      (norm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
    )
    (attn): Attention(
      (qkv): Linear(in_features=64, out_features=192, bias=False)
      (attn_drop): Dropout(p=0.0, inplace=False)
      (proj): Linear(in_features=64, out_features=64, bias=True)
      (proj_drop): Dropout(p=0.0, inplace=False)
      (mat): matmul()
    )
    (drop_path): Identity()
    (norm2): CustomNorm(
      (norm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
    )
    (mlp): Mlp(
      (fc1): Linear(in_features=64, out_features=256, bias=True)
      (act): CustomAct()
      (fc2): Linear(in_features=256, out_features=64, bias=True)
      (drop): Dropout(p=0.0, inplace=False)
    )
  )
  (1): Block(
    (norm1): CustomNorm(
      (norm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
    )
    (attn): Attention(
      (qkv): Linear(in_features=64, out_features=192, bias=False)
      (attn_drop): Dropout(p=0.0, inplace=False)
      (proj): Linear(in_features=64, out_features=64, bias=True)
      (proj_drop): Dropout(p=0.0, inplace=False)
```

```
          (mat): matmul()
        )
        (drop_path): Identity()
        (norm2): CustomNorm(
          (norm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
        )
        (mlp): Mlp(
          (fc1): Linear(in_features=64, out_features=256, bias=True)
          (act): CustomAct()
          (fc2): Linear(in_features=256, out_features=64, bias=True)
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
    )
  )
 )
 (deconv): Sequential(
   (0): Conv2d(64, 3, kernel_size=(1, 1), stride=(1, 1))
 )
)
```

**Fixed_z = Tensor(4,256)**
**Fid_stat = 'fid_stat/fid_stats_cifar10_train.npz'**

**Epoch = 317 from checkpoint**

**Input to generator is (fixed_z,epoch) -> gen_net(Tensor(4,256), 317)**

**O/P of**
**line 300: x = Tensor(4,64,1024)**

**line 302 : B ={size:3} torch.Size([4, 64, 1024])**
**line 303: H= 8 and W  =8**

**line 304 : x = Stageblock -> Block = blk**

```
blk = Block(
 (norm1): CustomNorm(
   (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
 )
 (attn): Attention(
   (qkv): Linear(in_features=1024, out_features=3072, bias=False)
   (attn_drop): Dropout(p=0.0, inplace=False)
   (proj): Linear(in_features=1024, out_features=1024, bias=True)
```

```
    (proj_drop): Dropout(p=0.0, inplace=False)
    (mat): matmul()
  )
  (drop_path): Identity()
  (norm2): CustomNorm(
    (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  )
  (mlp): Mlp(
    (fc1): Linear(in_features=1024, out_features=4096, bias=True)
    (act): CustomAct()
    (fc2): Linear(in_features=4096, out_features=1024, bias=True)
    (drop): Dropout(p=0.0, inplace=False)
  )
)
```

**In block line 158 : i/p : x=(4,64,1024) -> goes to Attention**

**In attention: i/p : x=(4,64,1024)**

Line 104 : B=4, C=1024, N = 64

LINE 105 qkv(x).reshape(B, N, 3, self.num_heads, C // self.num_heads).permute(2, 0, 3, 1, 4) : qkv = Tensor(3,4,4,64,256)

Line 106 q, k, v = qkv[0], qkv[1], qkv[2] : q=Tensor(4,4,64,256) , k = Tensor(4,4,64,256), v= Tensor(4,4,64,256).

Line 107 : attn =Tensor(4,4,64,64)

Line 109: relative_position_bias = Tensor(64,64,4)

Line 111 : relative_position_bias = Tensor(4,64,64)

Line 112: attn =Tensor(4,4,64,64)

Attention module returns x with shape Tesnor(4,64,1024)

```
Attention(
    (qkv): Linear(in_features=1024, out_features=3072, bias=False)
    (attn_drop): Dropout(p=0.0, inplace=False)
    (proj): Linear(in_features=1024, out_features=1024, bias=True)
      (proj_drop): Dropout(p=0.0, inplace=False)
    (mat): matmul()
    )
```

**In MLP: i/p : x=(4,64,1024)**

Line 63: x = Tensor(4,64,4096)

Line 66: x = Tesnor(4,64,1024)

```
Mlp(
    (fc1): Linear(in_features=1024, out_features=4096, bias=True)
    (act): CustomAct()
    (fc2): Linear(in_features=4096, out_features=1024, bias=True)
```

(drop): Dropout(p=0.0, inplace=False)
)

**Therefore, output of line 184 in Stageblock is : x= Tensor(4,64,1024) which indirectly the output of line 304**

**Now back to the generator we are on line 308 inside for loop which iterates over upsample blocks**

**i/p to line 308 : x= Tensor(4,64,1024), H=8,W=8**

THEN WE GO INSIDE PIXEL_UPSAMPLE
o/p of line 118 : B=4, N=64, C=1024

line 191: x = Tensor(4,1024,64) -> permuting(0,2,1)
line 192: x = Tensor(4,1024,8,8)
line 193: x = Tensor(4,256,16,16)
line 194: B=4, C=256, H=16, W=16
line 195: x= Tesnor(4,256,256)
line 196: x = Tesnor(4,256,256)
returns x,H,W = Tensor(4,256,256) , 16, 16

**BACK to generator, o/p of line 308 : x,H,W = Tensor(4,256,256) , 16, 16**

**Line 309: x = Tesnor(4,256,256)**

Line 310: we go inside stage block again
Line 183 : we go inside block
Line 158 : we go inside attention module
Line 104 : B,N,C = 4,256,256
Attention module retunrs x= Tesnor(4,256,256)
And we're back to line 159
Line 159 goes inside MLP
Line 63: x= Tesnor(4,256,1024)
Line 66: x= Tesnor(4,256,256)

**BACK to generator, for loop continues the same process**
**Now in 2nd iteration - o/p of lines:**

Line 308 o/p after pixel_upsample : x =Tensor(4,1024,64), H=32, W=32
LINE 309 : x =Tensor(4,1024,64)
LINE 310: x =Tensor(4,1024,64)

Then the control comes out of for loop and the output is
Line 314: x= Tensor(4,3,32,32)

**Now we're back to test.py file where we have output of line 49 as Sample_imgs = Tensor(4,3,32,32)**

**Line 50 : img_grid = Tensor(3,36,138)**

**Line 53 eval_iter = 6250**

**Line 55 is a for loop to iterate over 6250 instances**

**Line 56: z =(8,256)**

**Line 59 : generates images in an ndarray format of shape(8,32,32,3)**

**Where batch_size = 8 images**

## (C) Execution :