CS 597 REPORT 3

Yash Pradeep Gupte
MS COMPUTER SCIENCE
CWID: A20472798

1] CODE – TransGAN – Transformer GAN

(A) Image resolution = 64*64

(B) Feature extractor = Inception v3

(C) Generator_parameters = 19933952

(D) Discriminator_parameters = 2777856

(E)Train data – 1324 images

(F) Optimizers:
Adam (lr=1e-3, betas=(0.9, 0.999), eps=1e-8)

(G) Batch Size = 32

(H) Training loop

Train dataset – coco2014
Test dataset – coco2014_test

Sets a timestamp to save model and other supporting files.

Fetch image pairs ->
train_set, test_set :
rootdir - '/Users/yashgupte21/Desktop/CS597/week2/coco_data/images16/train'
dirnames - ['class_01_zebra']
filenames – ['COCO_train2014_000000456010.png', 'COCO_train2014_000000160741.png', 'COCO_train2014_000000481760.png',

'COCO_train2014_000000563929.png', 'COCO_train2014_000000162181.png', 'COCO_train2014_000000294877.png', 'COCO_train2014_000000526534.png', 'COCO_train2014_000000090317.png', 'COCO_train2014_000000443390.png', 'COCO_train2014_000000369736.png', …..]

input_data(captions16) -
'/Users/yashgupte21/Desktop/CS597/week2/coco_data/captions16/train/class_01_zebra/COCO_train2014_000000456010.txt'

target_data(image) -
'/Users/yashgupte21/Desktop/CS597/week2/coco_data/images16/train/class_01_zebra/COCO_train2014_000000456010.png'

target1_data(image_mask) -
'/Users/yashgupte21/Desktop/CS597/week2/coco_data/images_mask/train/class_01_zebra/COCO_train2014_000000456010.png'

target2_data(images256) -
'/Users/yashgupte21/Desktop/CS597/week2/coco_data/images256/train/class_01_zebra/COCO_train2014_000000456010.png'

train_samples.append([input_data, target_data, target1_data, target2_data]) -
[['/Users/yashgupte21/Desktop/CS597/week2/coco_data/captions16/train/class_01_zebra/COCO_train2014_000000456010.txt',
'/Users/yashgupte21/Desktop/CS597/week2/coco_data/images16/train/class_01_zebra/COCO_train2014_000000456010.png',
'/Users/yashgupte21/Desktop/CS597/week2/coco_data/images_mask/train/class_01_zebra/COCO_train2014_000000456010.png',
'/Users/yashgupte21/Desktop/CS597/week2/coco_data/images256/train/class_01_zebra/COCO_train2014_000000456010.png'],
['/Users/yashgupte21/Desktop/CS597/week2/coco_data/captions16/train/class_01_zebra/COCO_train2014_000000160741.txt',
'/Users/yashgupte21/Desktop/CS597/week2/coco_data/images16/train/class_01_zebra/COCO_train2014_000000160741.png',
'/Users/yashgupte21/Desktop/CS597/week2/coco_data/images_mask/train/class_01_zebra/COCO_train2014_000000160741.png',
'/Users/yashgupte21/Desktop/CS597/week2/coco_data/images256/train/class_01_zebra/COCO_train2014_000000160741.png'],…]]

train_dataset = ListDataset(train_samples, input_transform)

Same we will perform for test_dataset

Create_model ->
For training it will go in the else loop and create the model by setting network_data =None

Net_G –>

Class Generator_transformer :
In_ch =4208
Out_ch = 3

Generator_transformer architecture

```
Generator_transformer(
  (linear): Sequential(
    (0): Linear(in_features=768, out_features=128, bias=True)
    (1): ReLU(inplace=True)
  )
  (position_embedding): Embedding(16, 128)
  (encoder_layers): TransformerEncoderLayer(
    (self_attn): MultiheadAttention(
      (out_proj): NonDynamicallyQuantizableLinear(in_features=128, out_features=128,
bias=True)
    )
    (linear1): Linear(in_features=128, out_features=128, bias=True)
    (dropout): Dropout(p=0.2, inplace=False)
    (linear2): Linear(in_features=128, out_features=128, bias=True)
    (norm1): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
    (norm2): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
    (dropout1): Dropout(p=0.2, inplace=False)
    (dropout2): Dropout(p=0.2, inplace=False)
  )
  (transformer_encoder): TransformerEncoder(
    (layers): ModuleList(
      (0): TransformerEncoderLayer(
        (self_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=128, out_features=128,
bias=True)
        )
        (linear1): Linear(in_features=128, out_features=128, bias=True)
        (dropout): Dropout(p=0.2, inplace=False)
        (linear2): Linear(in_features=128, out_features=128, bias=True)
        (norm1): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
        (norm2): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0.2, inplace=False)
        (dropout2): Dropout(p=0.2, inplace=False)
      )
      (1): TransformerEncoderLayer(
        (self_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=128, out_features=128,
bias=True)
        )
        (linear1): Linear(in_features=128, out_features=128, bias=True)
        (dropout): Dropout(p=0.2, inplace=False)
        (linear2): Linear(in_features=128, out_features=128, bias=True)
        (norm1): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
        (norm2): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0.2, inplace=False)
        (dropout2): Dropout(p=0.2, inplace=False)
      )
    )
  )
  (deconv0): Sequential(
    (0): ConvTranspose2d(2048, 512, kernel_size=(4, 4), stride=(2, 2), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (deconv1): Sequential(
    (0): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
bias=False)
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (deconv2): Sequential(
    (0): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
bias=False)
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (deconv3): Sequential(
    (0): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
bias=False)
```

```
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (deconv4): outdeconv1(
    (conv): Sequential(
      (0): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
bias=False)
      (1): Tanh()
    )
  )
)
```

M = BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

Net_d –>

Class Discriminator :

Initializer - d=64, h=256, w=256
Nc = 3
Ndf = 64

Discriminator architecture

```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (1): LeakyReLU(negative_slope=0.2, inplace=True)
  (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (4): LeakyReLU(negative_slope=0.2, inplace=True)
  (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (7): LeakyReLU(negative_slope=0.2, inplace=True)
  (8): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (10): LeakyReLU(negative_slope=0.2, inplace=True)
```

Loses ->

Epoch_size = 42

```
# (1) Update D network: maximize D(x) - D(G(z))
#train with real
        netd.zero_grad()
        errD_real = netd(real_var, cap_vectors_var).mean(0).view(1)
        errD_real.backward(one)


# train with fake
        fake = netg(cap_vectors_var, mask_var)
        errD_fake = netd(fake.detach(), cap_vectors_var).mean(0).view(1)
        errD_fake.backward(mone)

# total discriminator loss
     errD = errD_real - errD_fake
     netd_losses.update(errD.mean().view(1).data, batch_size)
     netd_optimizer.step()

     netd_writer.add_scalar('net_err_minibatch', errD.mean().view(1), n_iter)


# (2) Update G network: maximize D(G(z))
        netg.zero_grad()
        fake = netg(cap_vectors_var, mask_var)

        feature_real = featureExtractorNet(real_var)
        feature_real = feature_real.detach()

        feature_fake = featureExtractorNet(fake)
        feature_fake = feature_fake.detach()

        adv_loss = netd(fake, cap_vectors_var).mean(0).view(1)
        content_loss = contentLoss(fake, real_var)

        errG = adv_loss + 100.0 * content_loss

        errG.backward(one)

         netg_losses.update(errG.mean().view(1).data, batch_size)
         netg_optimizer.step()
```

```
netg_writer.add_scalar('net_err_minibatch', errG.mean().view(1), n_iter)
```

(I) Running time:

It took approximately 5 minutes to run one epoch on my machine