Exploring Latent Space of GAN architectures for Image-to-Image Translation

Step-1

Qingyang Yu
qingyang.yu@rutgers.edu

Vamshi Saggurthi
vamshi.saggurthi@rutgers.edu

VIGNESH RAJESH SINGH
vrs60@scarletmail.rutgers.edu

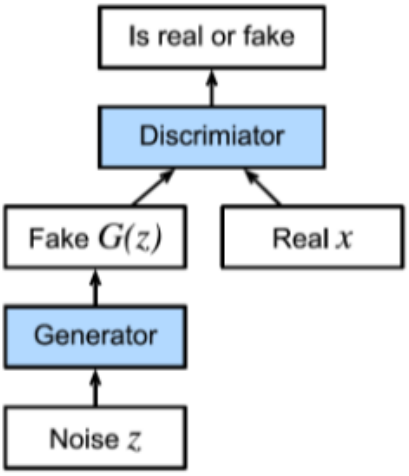
CHAITANYA VALLABHANENI
chaitanya.vallabhaneni@rutgers.edu

1. **Problem statement:-**
   The first task is to build a GAN model from image to image without any label so that we can capture the features of the images concisely by the data and explore the inner working theory of mapping functions.

2. **The basic structure of a GAN:-**
   GANs are a clever way of training a generative model by framing the generative problem as a supervised learning problem. It consists of two sub-models:-
   a) The generator model that we train to generate new examples.

   b) The discriminator model that tries to classify examples as either real (from the domain) or fake (generated).

   The two models are trained together in a zero-sum game, both try to beat/fool each other adversarial until the discriminator model is fooled about half the time when generating examples are plausible.



3. **Models Used:-**
   For this project, we used two models:-
   a) Based on the architecture provided. The generator model is based on the architecture created by J. Johnson, A. Alahi and L. Fei-Fei.
   For discriminator, we used the PatchGAN model created by P. Isola, J.-Y. Zhou, and A. A. Efros.
   b) In this model, we used a similar discriminator architecture, but we added some dropout in residual networks and gave an activation function of tanh.
4. **Architecture of Given Models**

   a) **Generator:-**
   Use 9 residual blocks for 256 ×256

training images. The network with 9 residual blocks consists of: c7s1-64,d128,d256,R256,R256,R256,R256, R256,R256,R256,R256,R256,u128,u64,c7s1

b) **Discriminator:-**
For discriminator networks, we use 70 ×70 PatchGAN.We use leaky ReLUs with a slope of 0.2.
The discriminator architecture is: C64 -C128 -C256 -C512.

Datasets :

We have 2 different data sets . Where each image is of about 256*256

Real Pizza - 6500+ images
Synthetic Pizza- 1300+ images

Data Preprocessing and loading:

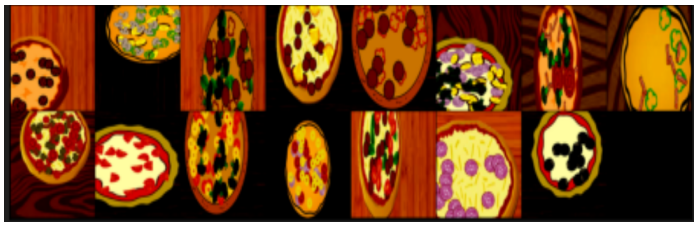
Transforms :
1. We resize the image to required 256*256 since some images may not suit our exact dimensions.
2. Center crop the image
3. Load it to a Tensor
4. Normalize the values (0.5,0.5,0.5)

We use the dataloader of pytorch and batch size of 16 for real images (2 GPU's) for training(hence higher batch size.

Synthetic images Batch size of 10(1 GPU) was available for training

Sample RealPizza Data :

Sample Synthetic Pizza:



Training Method :

Code Structure :

1. We have initial configurations on the top with options like criterion (loss functions, batch size,epoch sizes..)
2. device configuration for GPUS
3. Data loaders
4. Architectures/Network classes
5. training loops
6. Data collection using Tensorboard scalars

Real pizza data and Synthetic pizza data :

We pass in the real pizza data folder path of the same copy .ipynb file.
The results would be saved along with model weights and sample images during each training epoch

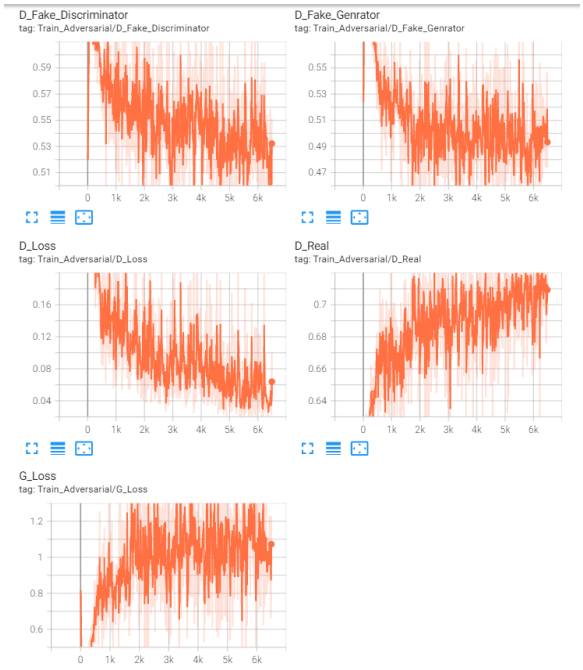
We run 50 epochs for both models for both real and synthetic images.
In total, we run 2 GAN models and store the models at each epoch. The metrics that we use to compare the models are.

1. **Loss Function (Both):-**
   a. MSE Loss
2. **Real Score (Discriminator only)**:- How likely is the Validation image classified correctly.
3. **Fake Score (Discr & Generator)**:- How likely is the fake image generated classified as fake. Should be converging to 0.5

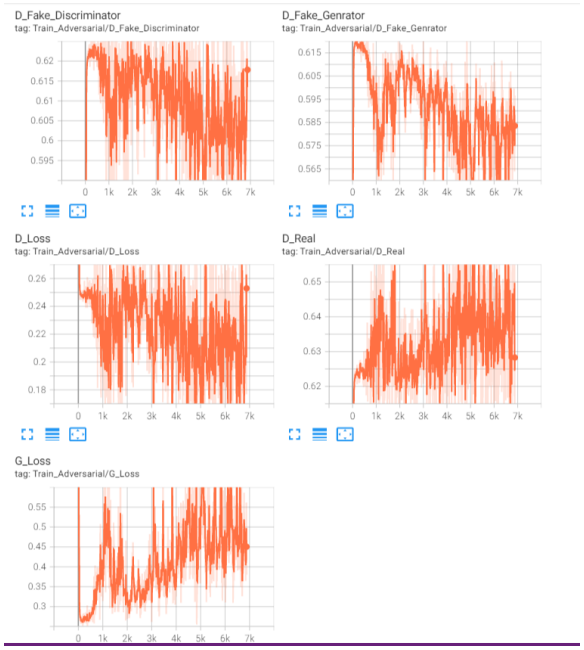
Exploring Latent Space of GAN architectures for Image-to-Image Translation

Results:-

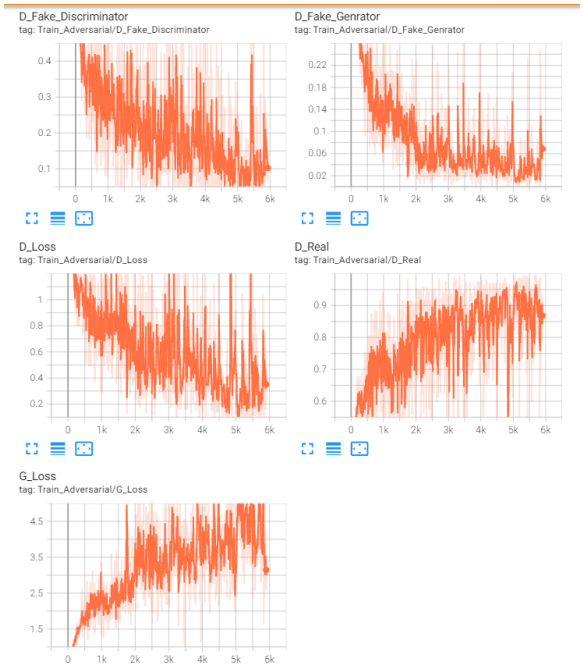
Synthetic Images :-



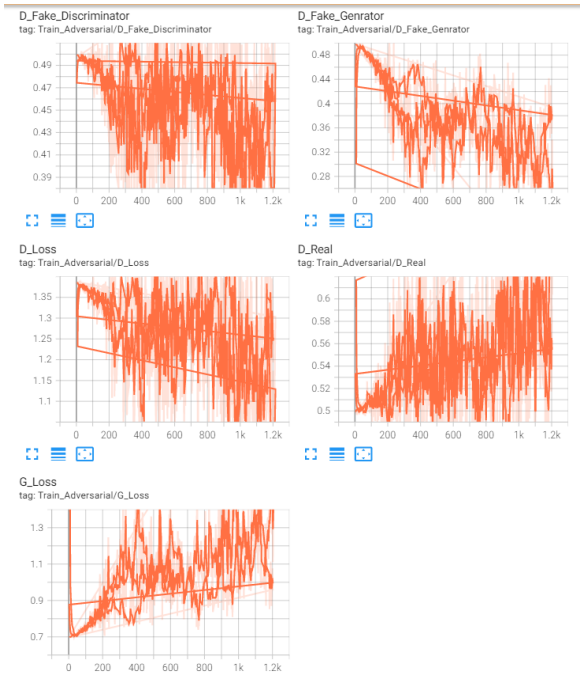
Real Images -



Experiment : Synthetic Images with tanh Activation:-



Experiment : Real Images with tanh Activation:-



Exploring Latent Space of GAN architectures for Image-to-Image Translation

GAN Images
Epoch : 38
Synthetic Pizza Generated

Real pizza Generated :

Epoch 37:

Interpretation :
Given Architecture:
We see the **Fake_score** of the discriminator being converging around 0.50-0.52 which is expected

As we see the **Loss** it is being going down which means the weights are being optimized as we train

Experiments :
We have experimented with some changes in the architecture
Generator:

1. Adding dropout of 0.2 in Residual block
2. Using Tanh activation in the final layer of Generator

Discriminator :
Same as give 70*70 patch GAN

Results: As we see the oscillations in the convergence the fake scores are not being correct

The images generated are not so good. This architecture didn't work well for us.

Conclusion:

We are able to generate good results with 50 epochs of training . If we train more epochs 100+, Augment more data (blurring , rotating images) , change model to buffer load data rather than training on same should increase the image quality more