Supervised Image to Image Translation Using Paired Images

Team 9
Vamshi Saggurthi - vs734
Vignesh Rajesh Singh -vrs60
Qingyang Yu - qy116
Chaitanya Vallabhaneni - cv346

Abstract

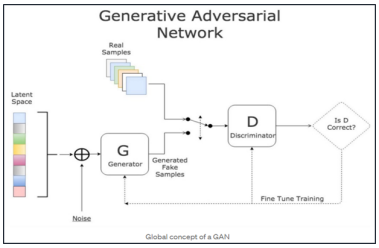
*The goal of the Task is to learn and implement Supervised Image to Image translation using Paired images using the popular **Pix2Pix** model on the Dayton dataset*
As the name indicates "Pixel to Pixel" which means, in an image it takes a pixel, then convert that into another pixel, goal is to learn the mapping from an input image to an output image. The application on areas of such models is in number of image-to-image translation tasks such as, translating maps to satellite photographs, converting black-and-white photos to color, and translating product designs to product photos etc.
Hence we try to train a Pix2Pix from ground up to learn more on internal workings of such models while qualitatively evaluating them using FID and Inception score metrics

1. Introduction

The GAN architecture is comprised of two models:

1. Generator model in order to generate new convincing synthetic images

2. Discriminator model that classifies between images that are real (from the dataset) and those that are not (generated).



The generator and discriminator are trained simultaneously as we train it over number of iterations the generator learns to fool the discriminator better(like a zero-sum game)

Pix2Pix is a general-purpose image-to-image translation model based on the Generative Adversarial Network (GAN).We use these special GAN's to synthesize photos from one space to another.The approach was presented by **Phillip Isola, et al**. in their 2016 paper titled **"Image-to-Image Translation with Conditional Adversarial Networks"** and presented at CVPR in 2017.[1]

The Major comparsion between GAN and Pix2Pix(cGAN) is that a basic GAN generates images from a random distribution vector(as in task1 pizzaGan) with no condition applied Where as a the other generates the image-to-to mapping under the conditiontal settings of translating from Street to Overview and vice versa

Figure 1. Conditional setting (Overview → Street)

1.1. Working Steps of cGAN

- Training data pairs (x: input domain image and y: target domain image)
- Pix2Pix GAN → G: x, z → y. (z → noise vector, x → input image, y → output image)
- Generator Network is a Encode- decode architecture which has image as the input, where we learn the deep

representational features of the input image , decode it and pass on the to Discriminator Network (PatchGAN) for prediction.

- CGAN loss function are optimised.

2. Model Architectures

The Paper specifies Generator and Discriminator Architectures as below

2.1. Generator

Generator is an encoder-decoder network first a series of down sampling layers then we have bottle neck layer then a series of upsampling layers. The Encoder-Decoder network, the authors employed the "U-Net" architecture with skip connections. The U-Net skip connections are especially appealing since they do not need any scaling, projections, or other adjustments because the spatial resolution of the layers being connected is already compatible.

The name U-Net highlights the structure of the "U" shaped network. Both generator and discriminator uses **Convolution-BatchNorm-ReLu** like module that it is the unit block of the generator and discriminator Skip connections are added between each layer i and layer " n - i ", where n is the total number of layers. At each skip connection all the channels from current layer i are concatenated with all the channels at " n - i " layer.

Encoder:C64-C128-C256-C512-C512-C512-C512-C512
Decoder:CD512-CD1024-CD1024-C1024-C1024-C512-C256-C128
All convolutions are 4×4 spatial filters applied with stride 2 and Convolutions in the encoder downsample by a factor of 2

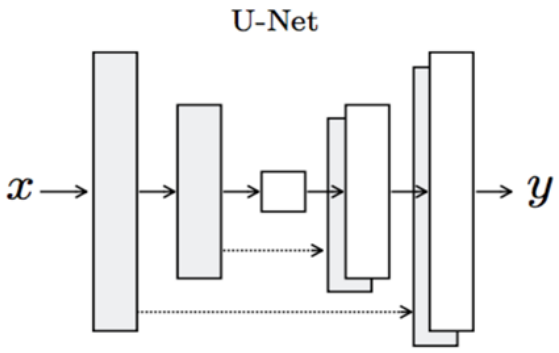


Figure 2. U-Net

2.2. Discriminator

The PatchGAN network is used in the discriminator network. Major difference between a PatchGAN and regular GAN discriminator is that rather the regular GAN maps from 256*256 image to a single Scalar output , which is of class "0"(fake) or real "1" , whereas the PatchGAN maps from 256*256 to a N*N patch image (array of outputs X, where each X_{ij} signifies whether the path i,j in the image is fake or real. X_{ij} is just a Neuron in a convolution neural net

Mathematically we could say as to down scaling the image using the U-Net down sampling blocks into 70*70 overlapping patches , running regular GAN classifier on each and averaging the results.
$n -> 32 * n/2 -> 64 * n/4 -> 128 * n/8 -> 1 * n/8$

The authors claim that this creates additional constraints, encouraging sharp high-frequency detail. PatchGAN also contains less parameters and is faster than whole picture classification.

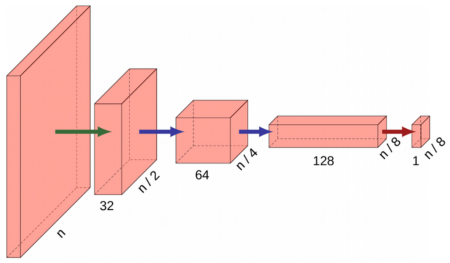


Figure 3. Discriminator

2.3. Loss functions

2.3.1 Generator Loss

The Generator Loss is responsible of "fooling" the discriminator by attempting to convince it that the generated picture is real, while also allowing the output image to be near to the target image. As a result, the total loss is of two components

1. First component of the loss is a Binary Crossentropy loss of the discriminator output for the generated images, against labels of 1. This section is in charge of "tricking" the discriminator believing the image to be real

2. The L1 loss, on the other hand, will make the output comparable to the target image.

2.3.2 Discriminator Loss

Discriminator loss is also combined of two parts

1. The first will make the discriminator output predict a value close to 1 for all pictures originating from real targets,

2. Second part will make the discriminator predict a value close to 0 for all images originating from the generator

For this aim, Binary Crossentropy Loss will be used for both losses.

The objective function of Pix2Pix model can be framed as

$L_{cGAN}(D,G) = E_{xy}[logD(x,y)]+E_y[log(1-D(y,G(y)))]$

L1 distance $L_{l1}(G) = E_{xy}[||x - G(y)||_1]$

The final objective then is framed as

arg $\min_G max_D L_{cGAN}(D,G) + lambda * L_{l1}(G)$

3. Implementation

3.1. Dataset Preparation

The Dataset is publicly available at https://www.mediafire.com/folder/f4gga3h86d659/GTCrossView and we have downloaded into to the local machine for preparation. The Dayton dataset contains individual images of the street images and it's coresponding overview(aerial) images tagged with the same names . We first resize the images to 256*256 and stack them one after the other for both street $\rightarrow$ aerial view vice versa The data set orignally contains about 152096 individual tagged images after stacking up as required it reduces to 76048 images of which have split int trainsplit of 550001048 images respectively

Figure 4. Merged Data set

3.2. Image Transformations

Data loaders have been prepared with transfomations to load the images for processing . The Paper suggests transforms of the orginal data as

- Resize to scale - (256*256)
- Random crop of the original image(256*256)
- Random Jittering
- Normalize values to [-1,1] (this helps learning lower numbers which is quite a advantage)

Finally a parser to separate to return target and conditonal image while parsing data batches

3.3. Optimizers

Adam optimizers have been used for both Generator and Discriminator models

3.4. Training

We train both Generator and Discriminator over several epochs passing on the data set . Both of them are trained simultaneously

3.4.1 Training Discriminator

A 6*256*256 image is passed to the discriminator and outputs a 1*30*30 tensor

Discriminator Training Loop

1. Generate a fake image from using the Generator
2. Now first pass in the Discriminator Real images and target domain from the training data and get the real prediction output
3. Make tensor of label 1 similar to the discriminator and compute the BCE loss (part 1)
4. Secondly pass in the fake image batch along the with source domain and get the fake prediction output
5. Make a tensor of label 0 similar to the discriminator and Compute the BCE loss of fake label
6. Average the both losses fake and real (this is the total loss of the discriminator)
7. Back propagate the losses and optimize

3.4.2 Training Generator

A 3*256*256 tensor(image) is given as input for the generator and it down-samples and up-samples the input data as specified in the architecture of the model and outputs a 3*256*256 translated image .

Generator Training Loop

1. Get the source domain image and pass it to generator to output the target domain image
2. Now pass in the fake images along the actual target domain image to the discriminator and collect the output
3. Compute the adeversial(BCE) loss of the fake images
4. Compute the reconstruction loss
5. compute the total loss (adversial + lambda*reconstruction loss)
6. Back-propagate and optimize

3.5. Training Experiment's

We have trained over diffrent batch sizes of the same models fine tuned our results over the results

We have used the constant hyper params of
Learning Rate = 0.002
lambda = 100
Batch Size = [varying over experiments]

Experiment-1 : Street to Aerial/Overview (batch size 128)(3 GPU's)
We passed in the initial experiment and trained the model until 174 epochs . We have observed the images being pixalated after 142 epoch and saw the model is over-fitting the data , there was no new learning curve . To support our results we saw the losses being fluctuated drastically in during these training sets , hence stopped it there
Tranining Time Each epoch : 16 mins

Experiment-2 : Aerial/Overview to Street(batch size: 48)(2 GPUs)
Based on our previous learning we have reduced our batch size drastically from 128 to 48 and aimed to train until 200 epochs. With this change we observed the intial epochs had better results compared to the batch size of 128 , this helped us learn the U-Net architecture works best for lower number of params which means lower batch size
Training Time each epoch : 24 mins

Experiment-3 : Street to Aerial/Overview(batch size: 24)(1 GPU) Now we have further reduced the batch size and results are much better within very few epochs.
Training time for each epoch :32 mins

Experiment-4 : Street to Aerial/Overview(batch size: 1)(1 GPU) We tried passing each image , but the training time for each Epoch was about 1 hr , this was not feasible for given the competiveness of the iLab GPU's. We needed about 200 hours 8.3 Days ,Hence we discarded the approach

4. Qualitative Evaltuation

4.1. Aerial to steet View

The images are good until epoch 95 and the getting pixlated after this , we can see the similarity in the outputs

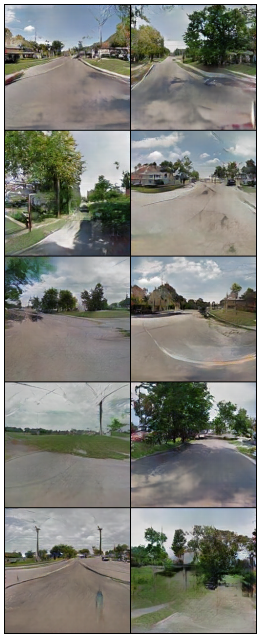


Figure 5. Epoch 33

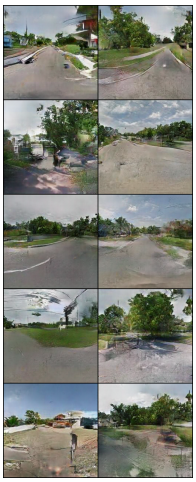


Figure 6. Epoch 140

4.2. Street to aerial

The images are good until epoch 106 and the getting pixlated after this , we can see the similarity in the outputs

Figure 7. Epoch 35

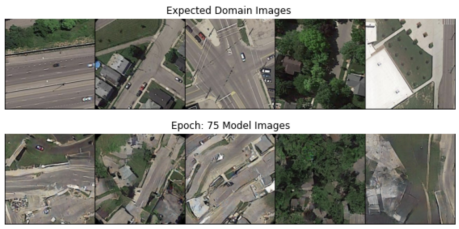


Figure 8. Epoch 75

Figure 9. Epoch 95

5. Quantitative Evaluation

5.1. FID score

The FID score is used to evaluate the quality of images generated by generative adversarial networks, and lower scores have been shown to correlate well with higher quality images.FID compares the distribution of generated images with the distribution of real images that were used to train the generator

5.1.1 FID Aerial to street

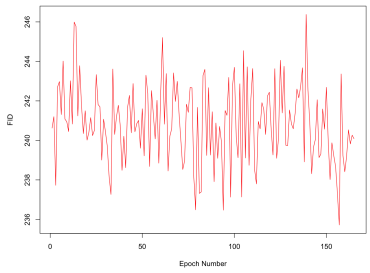


Figure 10. Aerial to Street FID

5.1.2 FID Street to Aerial

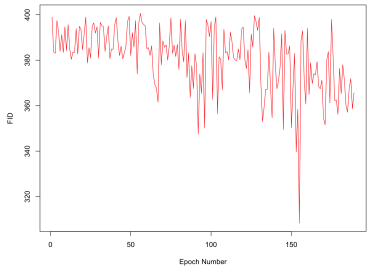


Figure 11. Street to Aerial FID

5.2. Inception score

Inception Score is a measure of how realistic a GAN's output is. A higher score is better. It means your GAN can generate many different distinct images. The lowest score possible is zero. Mathematically the highest possible score is infinity, although in practice there will probably emerge a non-infinite ceiling

Inception score is measure on Inception v3 classifier model by Google, By passing the images from our GAN through the classifier, we can measure properties of our generated images

The score is limited by what the Inception (or other network) classifier can detect, which is directly linked to the training data.

5.2.1 Steet to Aerial Inception Score

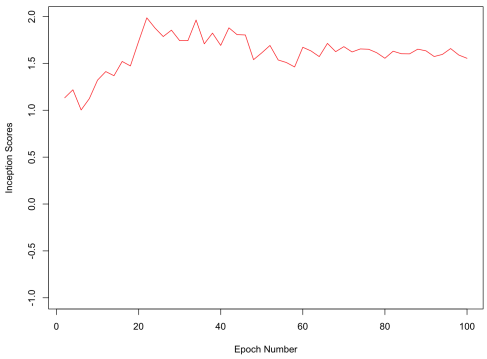


Figure 12. Street to Aerial IS

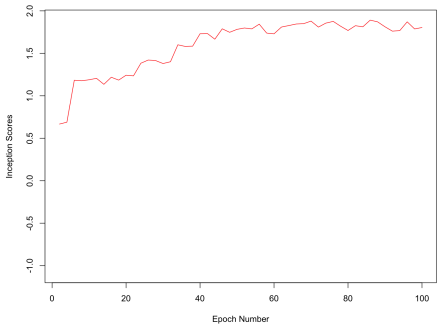


Figure 13. Aerial to Street IS

5.2.2 Aerial to Street Inception Score

5.3. Conclusion

The FID scores of both models have been decreasing and Inception scores have been increasing which is the expected . Hence the models are wokring good

5.4. More on Aerial to street

Figure 14. Aerial to Street Epoch 5

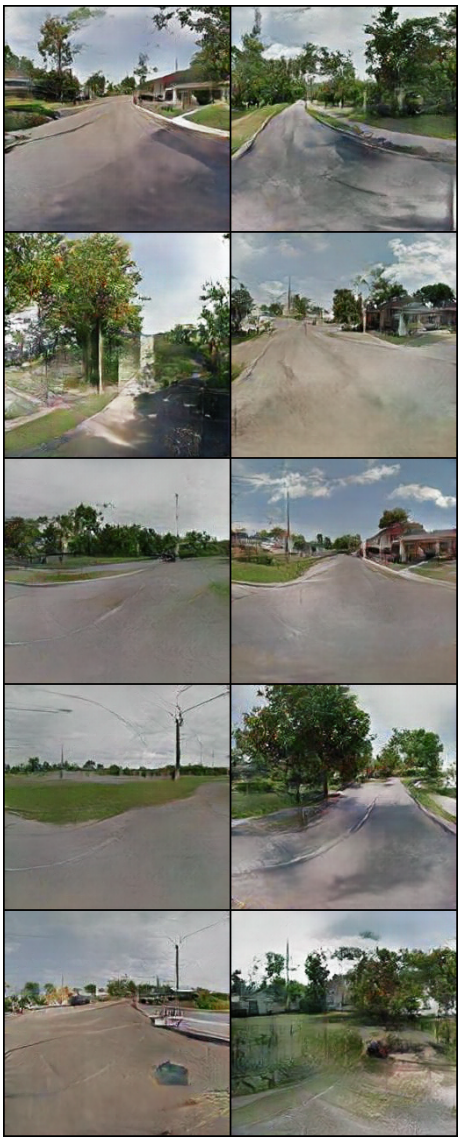


Figure 15. Aerial to Street Epoch 63

Figure 16. Aerial to Street Epoch 107