Image to Image Translation using C-GAN

Yati Katoch
ykatoch@ncsu.edu

Shubham Miglani
smiglan@ncsu.edu

I. INTRODUCTION

In this project, we have investigated a common framework for an image to image translation using conditional Generative Adversarial Networks in which we take an input image and generate the required output image using GAN conditioned on the input image. The advantage of using C-GAN in comparison to GANs which learn a generative model of data is that C-GAN learns a conditional generative model meaning that they condition to a specific input image and also learn a loss function to train this mapping. There have been many applications of GAN in image style transfer, super-resolution, etc. but they are all application-specific but the implementation in the pix2pix project is application-independent. We have explored various network architectures for generator (UNet, ResNet-6, ResNet-9, ResNet-20), Discriminator (ImageGAN, PatchGAN), loss function and training strategies and presented the qualitative and quantitative evaluations of different models.

We have used the data provided by the pix2pix dataset [2]. The plan was to experiment with the facades database first as it is the least computationally intensive and then based on results move on to other datasets. All the model training for different Architecture for Generator and Discriminator, different Loss function have been done on the facades dataset. The dataset consists of 400 images in the Training set, 100 in validation set and 100 in test set.

The original input image size is 600 X 600 X 3. We first resize the input images to 256 X 256 X 3. The pixel values of the images are also normalized between {-1,1} for easier training. The output images from the generator are of the same size as well.

II. METHODOLOGY

The conditional GAN consists of a generator and a discriminator. The aim of the generator is to produce images which are identical to the real image such that the discriminator can't identify whether the image is fake or not. The task of the discriminator is a simpler one, to classify whether the given image if fake or real. The two models are trained in an adversarial process. The discriminator is given the source and a generated image and is required to ascertain whether the generated image is identical to the source image. The generator output is conditional to the source image.

The discriminator model is updated directly while the generator model is updated based on the results of the discriminator model. The generator is trained using adversarial loss and L1 loss. The adversarial loss causes the generator to create identical images to the input data distribution. It is a sigmoid cross-entropy loss of the generated images and an array of ones. L1 loss (mean absolute error) is also used in combination with the sigmoid cross-entropy. Total generator loss is calculated as the sum of sigmoid cross-entropy loss + lambda*L1_loss where the value for lambda is decided in the paper as 100.

The discriminator has 2 losses. One loss for the real images is the cross-entropy loss of real images and an array of ones. Generated loss is the sigmoid cross-entropy loss of the generated images and an array of zeros. The total loss is the sum of real and generated losses.

The architecture used for the generator in [1] is U-Net which consists of 8 encoding and decoding layers having skip connections between layer i to layer n-i. Each block in the encoder is (Conv-Batch norm-Leaky Relu) and the decoder is (Transposed Conv-Batch norm-Dropout (first 3 blocks)-Relu). For U-Net, two models were trained based on Unet-128 and Unet-256 where 128 and 256 are the input image size. The U-Net is constructed from the innermost layer to the outermost layer in a recursive process.

For residual-based network, it consists of 2 encoding and 2 decoding layers with 6 (ResNet-6) or 9(ResNet-9) or 20(ResNet-20) residual blocks in between. The original proposal was planned for ResNet-50 but due to computational constraints, it was replaced by ResNet-20 for training.

For the discriminator, Patch GAN is used in [1] with a patch size of 70x70. A 70x70 Patch GAN classifies 70x70 patches on the input image as real or fake. Each block in the discriminator is (Conv-Batch Norm-Leaky Relu). The output shape after the last layer is (batch_size,30,30,1). Each 30x30 patch of the output classifies a 70x70 portion of the input image. The Patch GAN discriminator architecture has fewer parameters to train than a full image discriminator. The discriminator receives 2 inputs, the input and target image where the input image can be real or fake. Also, Pixel discriminator was used which classifies whether a pixel is real or not. It doesn't have any effect on spatial statistics but encourages greater color diversity as mentioned in [1].

III. MODEL TRAINING

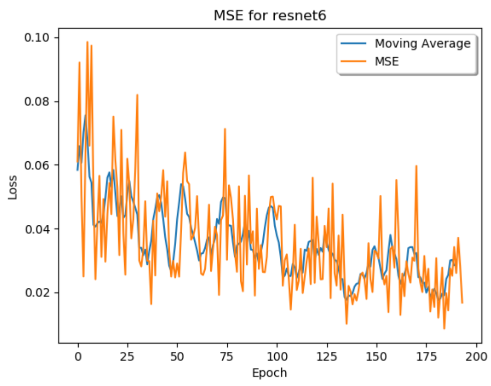
Before training as suggested in [1], Random jitter is applied by resizing the 256x256 image to 286x286 and then randomly cropping back to 256x256 with random flipping. The task for the discriminator is easier as compared to that for the generator so the discriminator learns quickly as compared to the generator. Therefore, as suggested in [1], the objective for the discriminator is divided by 2 while optimizing which slows the rate at which the discriminator learns. Adam was used for optimization with a starting learning rate of 0.0002 with a linear decay policy. The batch size used for training is 1.

For training comparison, MSE (Mean Square Error) and SSIM (Structural Similarity) were calculated for various models. The MSE plots along with images from minimum MSE and maximum SSIM are plotted. As the losses do not constantly decrease, moving average with a window of 5 was also plotted to get a clearer picture.

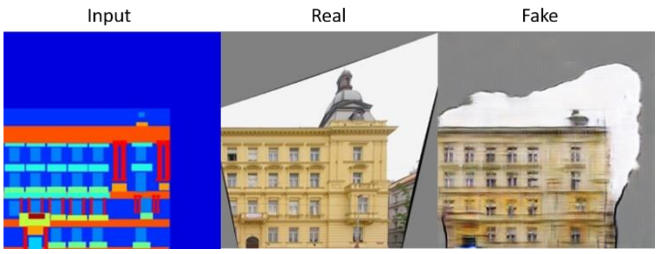
Also, the training for these types of losses is a little complicated as it must be made sure that neither model has won. If either the generator loss or the discriminator loss gets very low, it is an indicator that one model is dominating the other. The value of 0.69 (log (2)) is taken as a reference point for these losses as it indicates a perplexity of 2. As the training progresses, generator loss should go down. Sometimes during training, the discriminator loss would get stuck at zero and the entire training process had to be started again which is computationally intensive and time consuming as one model takes on average around 5-6 hours to train on Google Colab. The training was done for 200 epochs for all models.

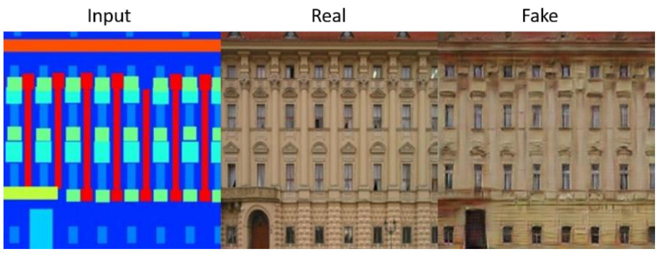
A). GENERATOR ARCHITECTURE

1) The results for ResNet-6 model are as follows:
MSE:

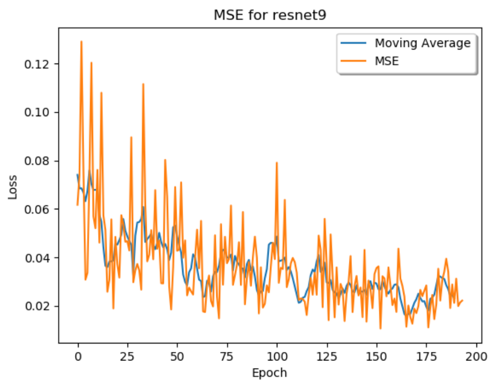


The minimum MSE was recorded at epoch 185 and the max SSIM was recorded at Epoch 190. The images for the maximum SSIM and minimum MSE are shown as follows in the respective order:

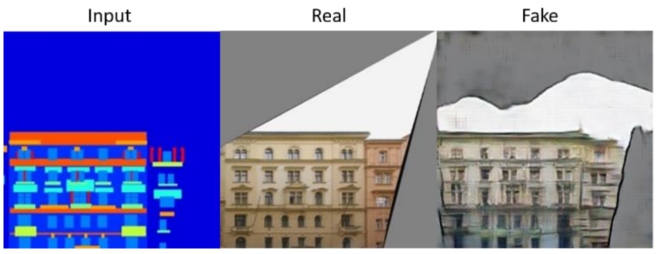


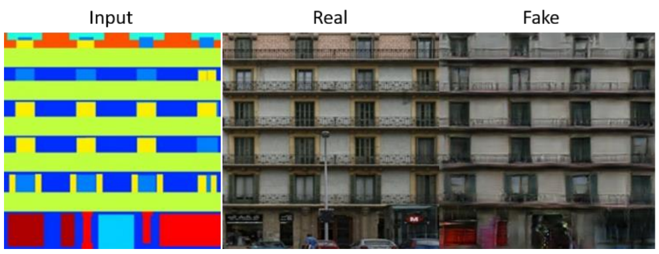


2) The results for ResNet-9 Model are as follows:

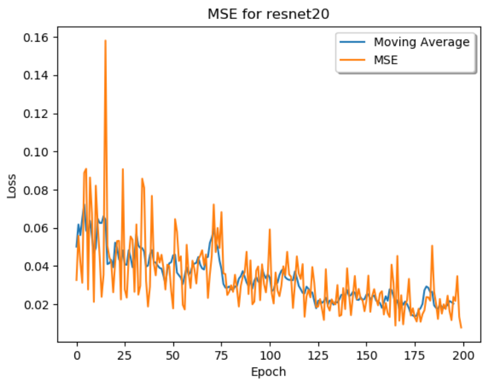


The minimum MSE was recorded at epoch 153 and the max SSIM was recorded at Epoch 141. The images for maximum SSIM and minimum MSE are as shown:

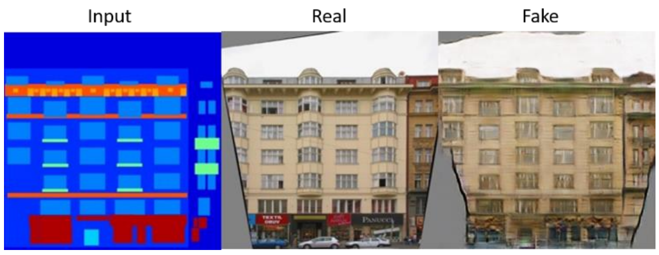


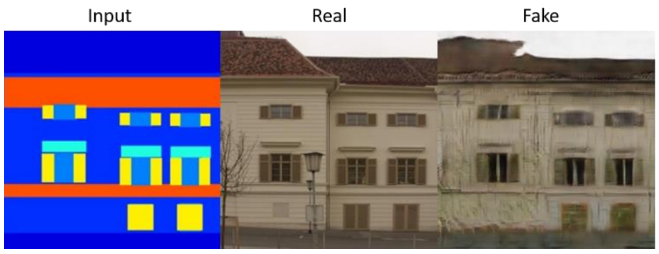


3) The results for ResNet-20 Model are as follows:

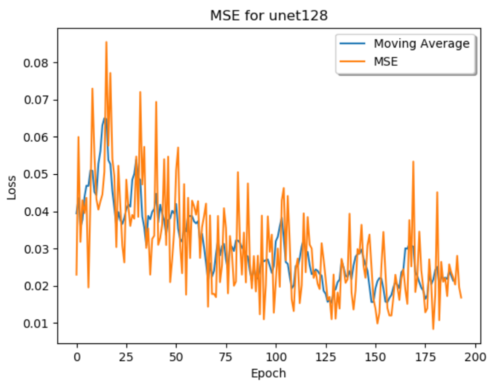


The minimum MSE was recorded at epoch 200 and the max SSIM was recorded at Epoch 135. The images for the maximum SSIM and minimum MSE are shown

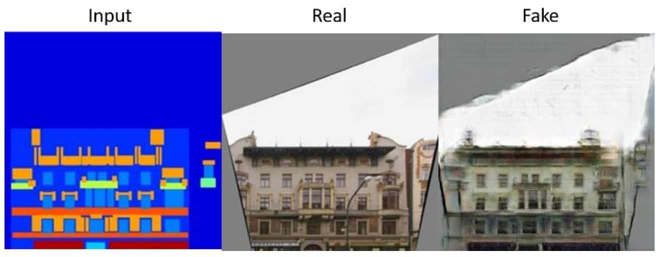


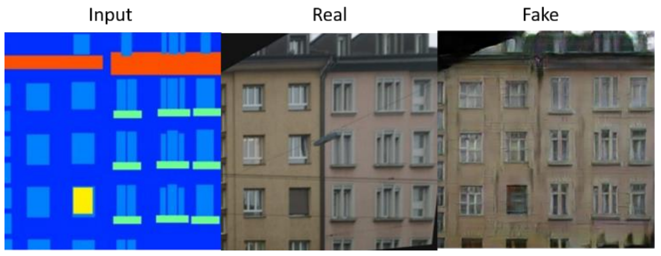


4) The results for Unet-128 model are as follows:

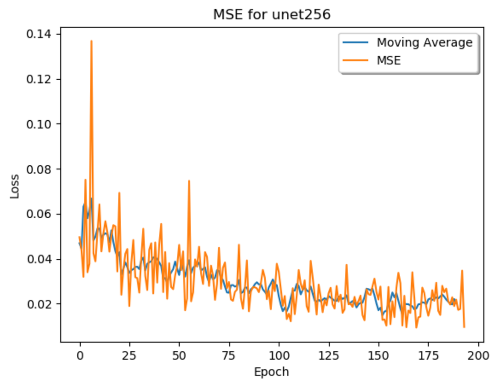


The minimum MSE was recorded at epoch 180 and the max SSIM was recorded at Epoch 127. The images for the maximum SSIM and minimum MSE are shown:

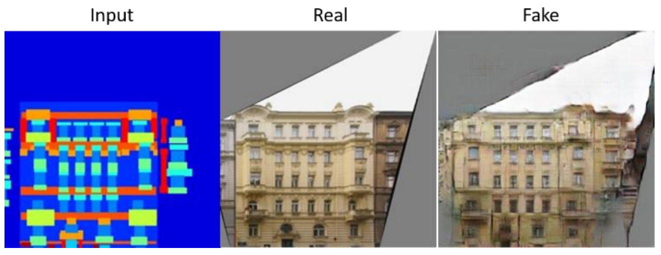


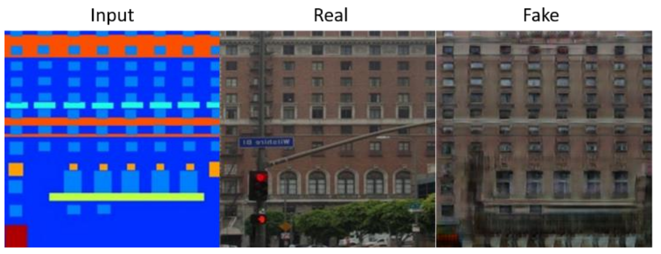


5) The results for Unet-256 model are as follows:



The minimum MSE was recorded at epoch 170 and the max SSIM was recorded at Epoch 174. The images for the maximum SSIM and minimum MSE are shown:





The summary of different generator architecture is as follows:

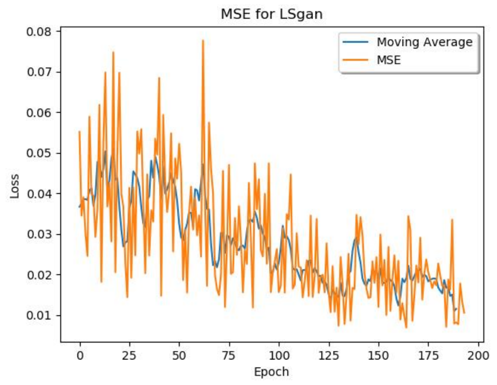
Table 1
Generator Architecture

Type	Minimum MSE	Epoch for Minimum MSE	Maximum SSIM	Epoch for Maximum SSIM
ResNet-6	0.0086	185	0.54	190
ResNet-9	0.0106	153	0.58	141
ResNet-20	0.0077	200	0.61	135
UNet-128	0.0083	180	0.55	127
UNet-256	0.0093	170	0.51	174

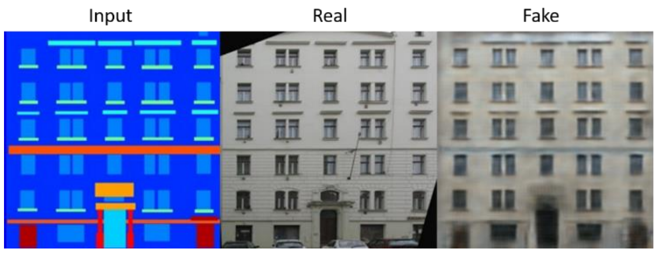
As can be seen, ResNet-20 outperforms the baseline and other models both in terms of MSE and SSIM. The performance of ResNet in terms of SSIM improves with an increase in residual layers. Strangely, UNet-128 performance was found to be better than UNet-256 which means that the model trained better with an image size of 128 by 128. It can be since the model trained faster with smaller image size.
It can also be observed that the images with minimum MSE have no cropping in them whereas the images with maximum SSIM have cropping in them. This may be because the cropping increases the MSE value significantly but doesn’t affect the SSIM as much.

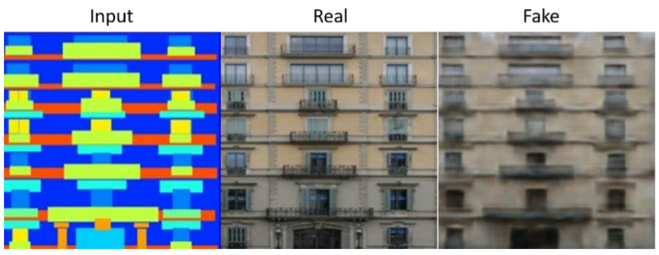
B). LOSS FUNCTION

In addition to vanilla GAN, the loss function used in the pix2pix paper, LSGAN (Least squares Generative Adversarial Network) loss was tried which is basically the L2 loss objective. A default generator of ResNet-9 was used and the discriminator was a PatchGAN. As can be seen, MSE is lower as compared to the vanilla loss which makes sense as the objective function is the same as our performance measure. But the resulting images turn out to be blurry.



The minimum MSE was recorded at epoch 165 and the max SSIM was recorded at Epoch 135. The images for the maximum SSIM and minimum MSE are shown:





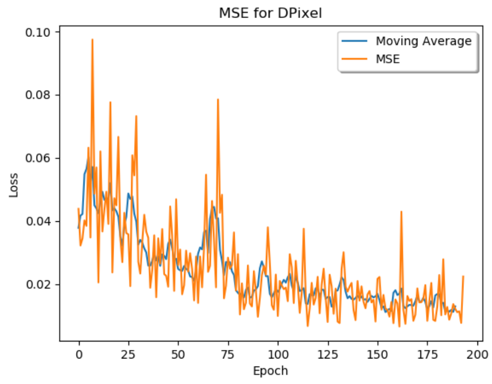
As can be seen in the table below, the LSGAN improved the SSIM and MSE quite significantly

Table *2*
Loss Function

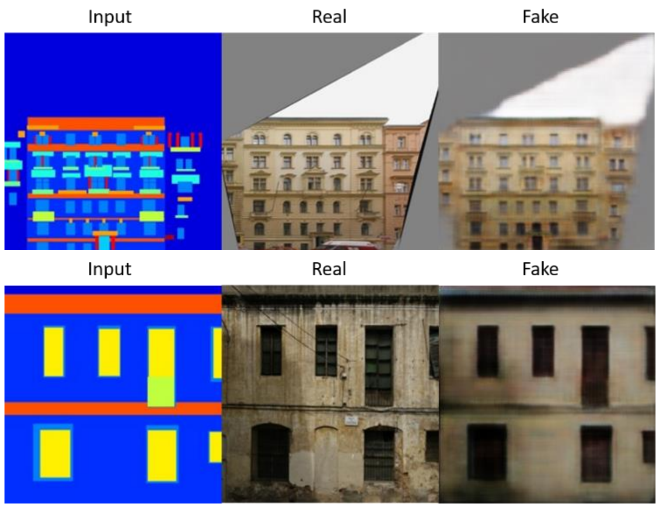
Type	Minimum MSE	Epoch for Minimum MSE	Maximum SSIM	Epoch for Maximum SSIM
Vanilla	0.0106	153	0.58	141
LSGAN	0.0068	165	0.61	135

C). Discriminator architecture

The Discriminator architecture was changed to 1x1 PixelGAN which can classify whether a pixel is real or not. This helps in generating greater color diversity but has no effect on spatial statistics.



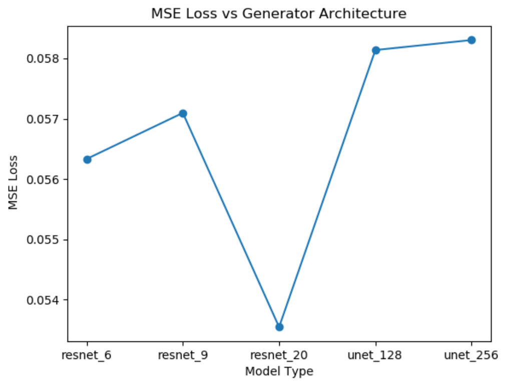
The minimum MSE was 0.0065 at epoch 162, maximum SSIM was 0.67 at epoch 150. Whereas for PatchGAN it was 0.0106 at epoch 153 and 0.58 at epoch 141. The images for the maximum SSIM and minimum MSE are shown:

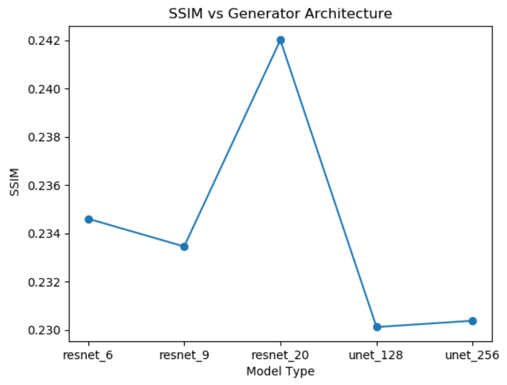


IV. EVALUATION

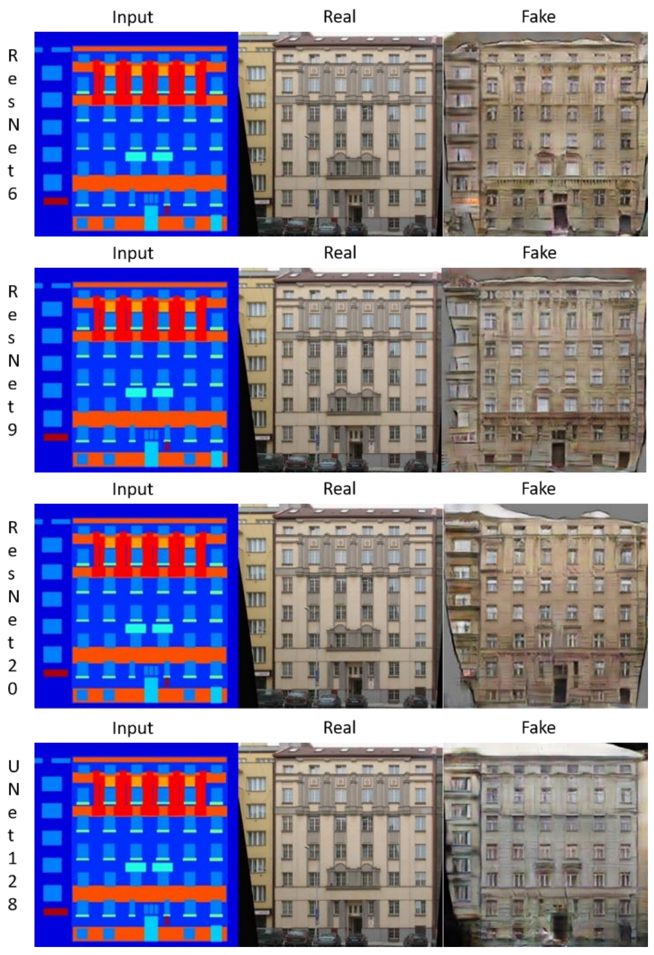
The evaluation was done on the test set of the facades database. The generator model used for testing were for the last epoch in training.

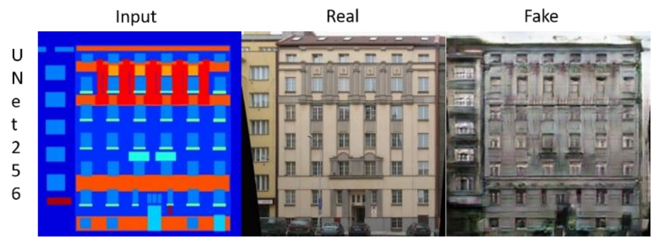
1) Generator Architecture: Firstly, average MSE and SSIM are plotted for different generator architectures and comparison is made. As can be seen from both the plots for SSIM and MSE, ResNet-20 outperforms the other models significantly. The performance of ResNet-6 and ResNet-9 were found to be better than UNet-128 and UNet-256. The ResNet models had higher SSIM and lower MSE values as compared to the UNet which was the baseline in [1]





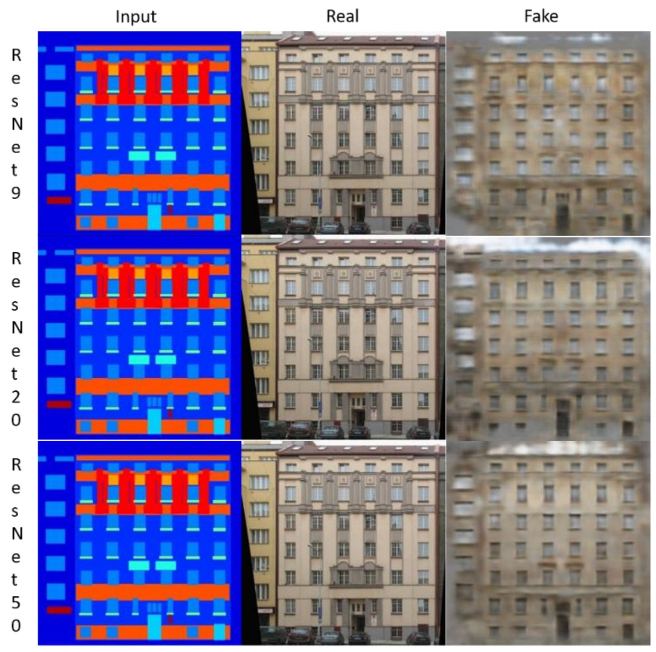
The images for test image 1 for different generator architectures are as shown:





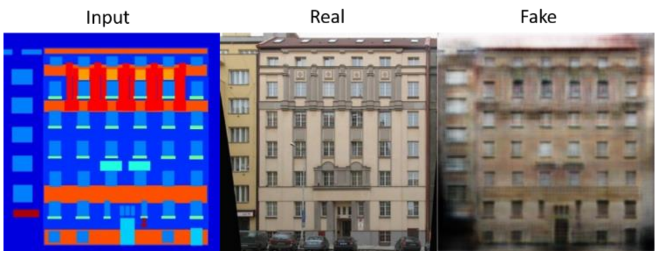
The performance of ResNet-20 in visual analysis also seems better as compared to other models. In UNet models, the color obtained can be observed to be different as compared to the real image and the ResNet Models.

2) LSGAN:



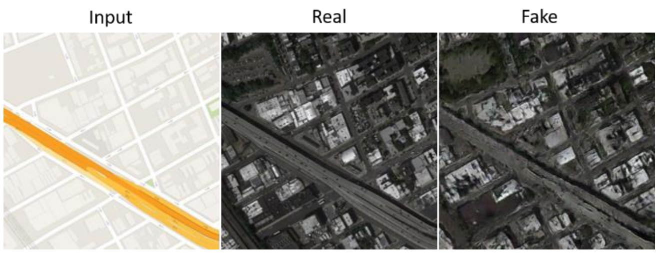
While the SSIM and MSE improved for LSGAN loss but as can be seen and was expected it produces blurry images as it is just a mean squared error loss. The default generator architecture was ResNet-9 but the results were also checked with ResNet-20 and ResNet-50. All the models produce better results in terms of MSE and SSIM but visually are blurry. Thus, the original vanilla loss is quite effective at generating fake images.

3) 1x1 Pixel Discriminator

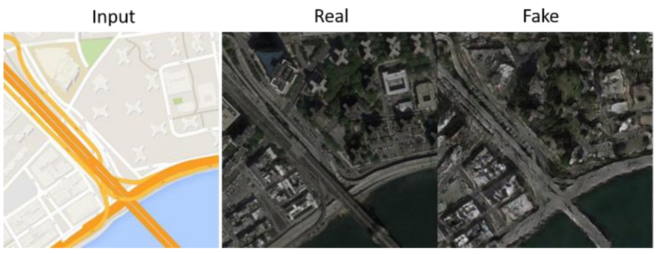


While the 1x1 Pixel Discriminator had better MSE and SSIM performance but the images produced are quite blurry. It is better at producing different colors but has no effect on spatial statistics.

The ResNet-9 Generator was also trained on maps and edges 2shoes database. For maps, due to computational constrains, the model was trained for 100 epochs only. The training image for last epoch is as shown:

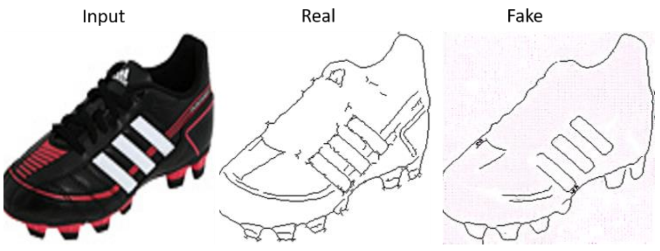


An example of test image is as shown:

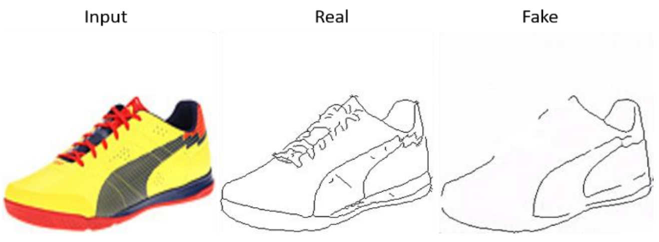


The model performance seemed comparable to the one in baseline even with training of 100 epochs whereby the original baseline was trained for 200 epochs.

Due to further computational constraints, edges2shoes dataset was also trained but only using 8000 of the training images for 50 epochs only. With this configuration, the model took around 11 hours to train. The training image for the last epoch is as shown:



Two examples of test images are also shown. The first image has good results but the second one does not.



I. VI. REFERENCES

[1] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image translation with conditional adversarial networks, 2016.

[2]https://people.eecs.berkeley.edu/~tinghuiz/projects/pix2pix/ datasets/

[3] Pros and Cons of GAN Evaluation Measures, Ali Borji

[4] https://machinelearningmastery.com/how-to-implement-pix2pix-gan-models-from-scratch-with-keras/