1. Introduction

Image to Image Translation (or I2I) is the task of translating images from one domain to another. The original image ends up with the style of the image in the translated domain. I2I has been increasing in popularity and has made tremendous progress in the last few years because of its important applications in fields such as image synthesis, segmentation, style transfer, restoration, and pose estimation.

For this assignment, we use satellite images and convert them to maps images, thereby trying to retain the structural properties of the map's images formed. To aid our evaluation, the SSIM index is used to measure the structural similarity between the actual maps image and the image generated by our I2I model.

The model used to develop this aerial to maps converter is a Generative Adversarial Network (GAN). The idea of a GAN is to use a generator and discriminator which are always competing. The Generator tries to produce a fake image which is similar to the actual target while the Discriminator is a classifier which tries to distinguish between the fake image and true image. When training begins, the Generator obviously produces bad fake images. The Discriminator can easily tell these are fake and penalizes the Generator for this. As training goes on, the Generator learns to produce better fake images. The Discriminator must adapt at being able to distinguish between samples that are now more similar to each other. Since the Generator and Discriminator are both neural networks, they learn through backpropagation.

To train the GAN, we alternate between training the Discriminator and Generator. The Discriminator is trained for one or more epochs while the Generator weights are kept constant and vice versa. The Generator improves with training while the Discriminator performance gets worse. If the Generator is perfect, the Discriminator probability to distinguish between real and fake would be 50%, just like flipping a coin. The feedback given to the Generator would be less meaningful. If training is continued after this point, the Generator will be essentially training on junk feedback and its performance may deteriorate.

The rest of the report will be structured as follows: section 2 will describe the methods used to improve the model performance including model architectures and hyperparameters, section 3 will elaborate on experiments and analysis and will include dataset analysis, image visualizations, and offer some ablation studies.

2. Methods to Improve Model Performance

2.1 Model selection and parameters

To select the generator and discriminator model along with the ngf (generator filters in last layer) and ndf values (discriminator filters in last layer), 40 epochs with a learning rate of 8e-4 was used. All other parameters were kept default. Increasing the ngf and ndf values from 64*64 to 128*128 increases the SSIM score from 0.556 to 0.5716.

Next, different discriminator models were tried. By changing from the basic model to the pixel model, the SSIM increased from 0.5716 to 0.6246. Thus, pixel was selected as the discriminator model.

To select the generator model, the SSIM score of each was considered. It is important to note that the gpu farm did not allow to use the resnet models for a batch size greater than 8. Hence, to

ensure consistency, the batch size in all 4 tests was kept 8. The results are given in Table 1. The resnet_9blocks model gives the highest SSIM score of 0.6428 and is hence selected.

Model	Unet_128	Unet_256	Resnet_6blocks	**Resnet_9blocks**
SSIM	0.6246	0.623	0.634	**0.6428**

Table 1: Generator Model Selection

2.2 Hyper Parameters

i. Epochs: 40

ii. Learning Rate: Selected as 8e-4 based on Figure 1

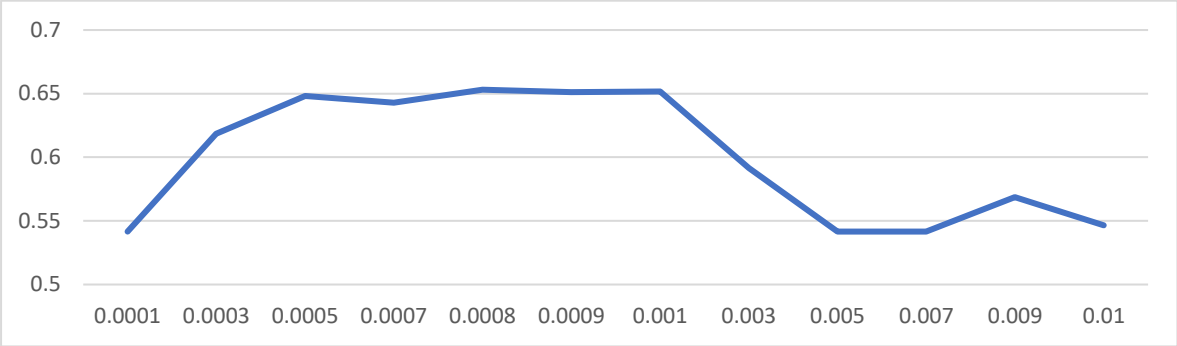

Figure 1: SSIM vs Learning Rate

iii. Batch Size: As seen in Table 2, decreasing the batch size offers an SSIM improvement, albeit small. Hence the batch size is selected as 2

Batch Size	8	4	**2**
SSIM	0.6428	0.65	**0.6517**

Table 2: Batch Size

For other hyperparameters, we use unet_128 as the generator model to save time. Unet models train 3 times faster than resnet models. Hence, to ensure efficiency, the unet_128 model is used for hyperparameter selection with a batch size of 8.

iv. Init_type: The model used for network initialization. Normal is selected by comparing the results shown in Table 3.

Init_type	**Normal**	Xavier	Orthogonal	Kaiming
SSIM	**0.6246**	0.608	0.61	0.594

Table 3: Init_type Selection

v. Load height and Width: These parameters had almost no effect on the SSIM score with the test results fluctuating between 0.615 and 0.625 with no clear trend. The 32*32 combination offered the best result of 0.625 and was hence selected.

vi. GAN Mode: The type of GAN objective function. lsgan is selected based on the results shown in Table 4.

GAN Mode	**lsgan**	Vanilla	Wgangp
SSIM	**0.6254**	0.62	0.62

Table 4: GAN Mode Function Selection

vii. LR Policy: Linear policy is selected based on results shown in Table 5.

LR Policy	**Linear**	Step	Cosine	Plateau

SSIM	**0.6254**	0.62	0.586	0.55

Table 5: LR Policy vs SSIM

viii. Beta1 Value for Adam: Selected as 0.8 based on Figure 2.

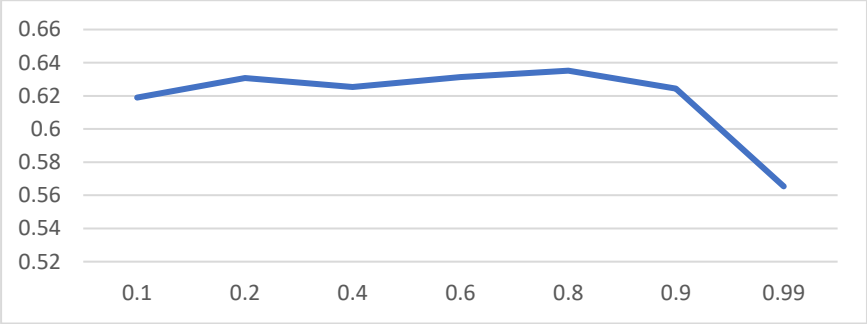

Figure 2: SSIM vs Beta1

ix. Lambda L1: Selected as 100 based on Figure 3.

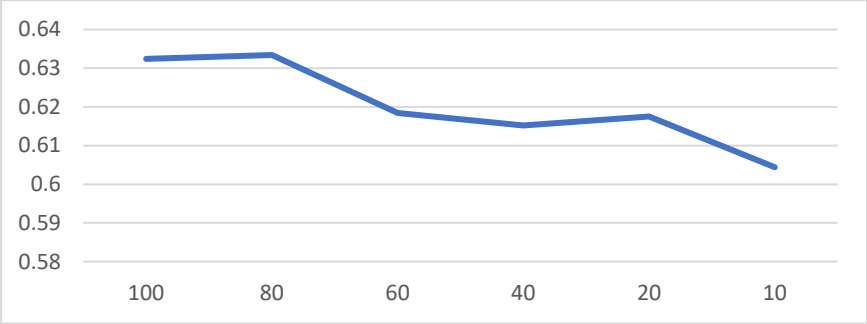

Figure 3: SSIM vs Lambda1

x. LR Decay Iterations: The learning rate gets multiplied by a gamma every n epochs. Here, n is selected as 10 based on Table 6.

LR Decay	**10**	20	30	40	50
SSIM	**0.651**	0.6414	0.6479	0.6454	0.648

Table 6: LR Decay Selection

xi. Epochs Decay: Number of epochs to linearly decay learning rate to 0. Selected as 20 based on Table 7.

Epochs Decay	5	10	**20**	30	40
SSIM	0.6324	0.6434	**0.651**	0.6464	0.6464

Table 7: Epochs Decay Selection

xii. Norm: Batch norm gives a better result than instance and is hence selected over instance.

3. Experiments and Analysis

3.1 Dataset Analysis: The dataset consists of 1096 training images, and 549 validation and testing images respectively. This gives a train:val:test split of 50:25:25. Each of the images are aerial images and their corresponding target image will be the map image. For the train and val datasets, the target images have been provided.

3.2 Failure Case Analysis: The images with the lowest SSIM score with their target validation image have been shown below in Figure 4. The first 2 images clearly have similar features- zig-zag roads and highways. The model mistakenly captures the highway as the zig-zag road and provides its colour as orange. On the other hand, it hardly captures the zigzag, replacing the flyover's area with blank/ blurred spaces. This could be attributed to the fact that such features, or the combination of the 2 are underrepresented in the dataset and the model does not have enough training opportunity to deal with such features. The third image is a very dense residential area with extremely narrow roads. The model is unable to distinguish the narrow roads from the buildings and hence provides a very faint representation (if you look at the image very closely) of such roads.

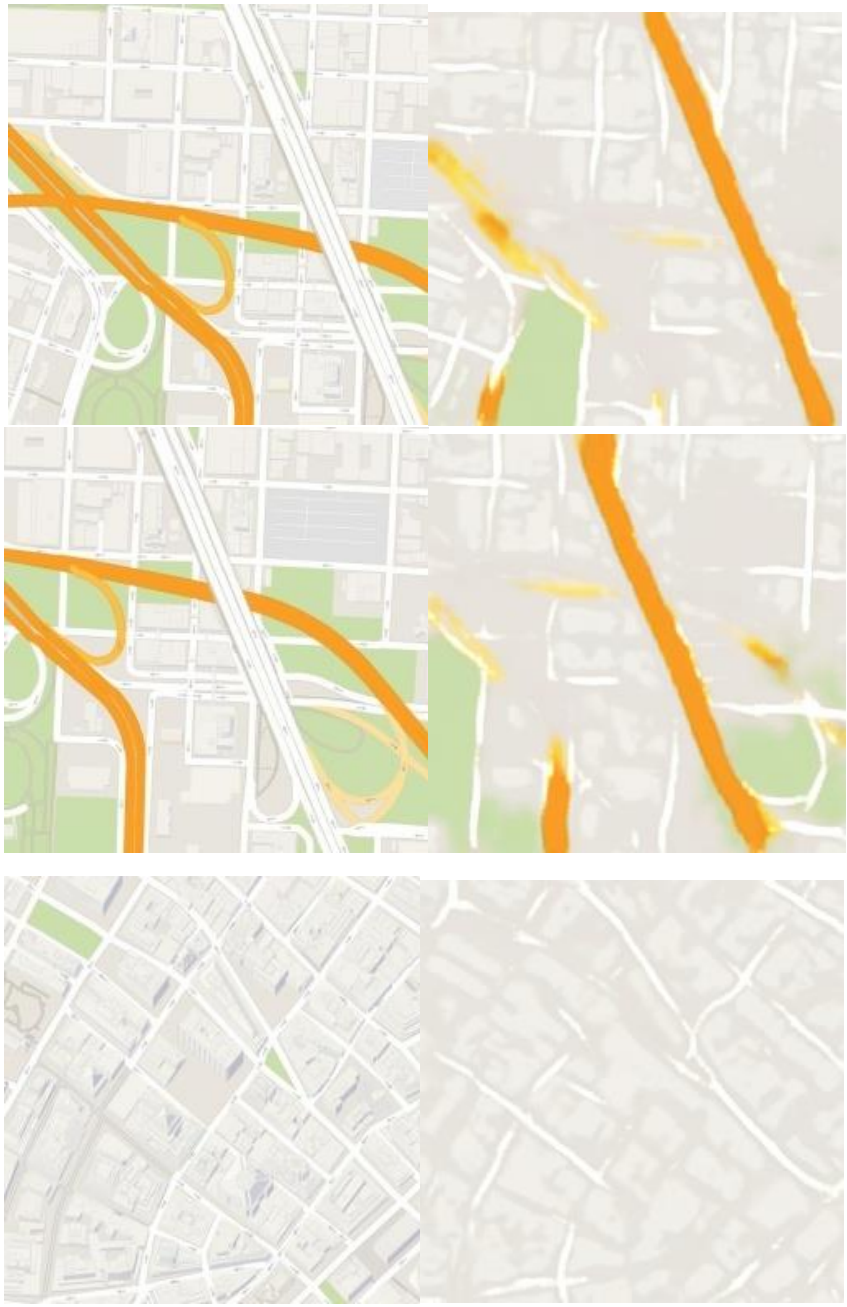

Figure 4: Target Image vs Generated Image

3.3 Ablation Studies

i. Effect of number of filters in the last convolution layer of generator and discriminator: Again, to ensure efficiency, the unet_128 model was used for this experiment. Table 8 shows the relationship between the number of ngf and ndf on the SSIM score. The first column is the number of filters in the last generator layer (ngf), the first row is the number of filters in the last discriminator layer (ndf), the values in the middle show the corresponding SSIM score for the combination of NGF and NDF value for that row and column. A clear increasing trend can be observed when moving from left to right, i.e. increasing the number of ndf filters. This effect is most pronounced when ngf = 128, as we see an SSIM improvement from 0.4536 to 0.5716. Going from top to bottom also offers improvements in some cases, although these are more subtle. Overall, it suggests that having more filters in the last layer of both discriminator and generator can offer the highest SSIM score. However, this comes at the expense of increased training time: moving from a 32*32 to 128*128 more than triples the training time per epoch (from 35 seconds to 115 seconds).

NGF / NDF	**16**	**32**	**64**	**128**
16	-	-	-	0.5616
32	-	0.5319	0.5225	0.57
64	-	0.5301	-	0.5505
128	0.4536	0.53	0.55	0.5716

Table 8: NGF/ NDF Values and SSIM Score

ii. Impact of Learning Rate on SSIM Score: Figure 1 showcases the graphical relation between the SSIM Score and Learning Rate. Initially, increasing the learning rate offers drastic improvements in the SSIM score. This indicates that the model is unable to adapt to the dataset when the learning rate is too low and hence may undertfit. The SSIM score converges between 7e-4 and 1e-3 and going beyond this deteriorates model performance. A very high learning rate (more than 1e-3) may result in overfitting as the model only captures certain key frequently occurring features but doesn't capture overall model generality. Therefore, a value in the range 7e-4 and 1e-3 may be most suitable. Increasing the number of training epochs may however change this result- more epochs may require a smaller learning rate value.

iii. Unet vs Resnet: As shown in Table 9, the Resnet Architecture performs better than the Unet architecture. This result is consistent regardless of the network depth- both 6 and 9 resnets perform better than 128 and 256 unet architectures. However, the resnet architectures require significantly more training time (3x more) than Unet architectures. Maybe having a deeper resnet architecture could offer even better performance but this will require more training and hyperparameter tuning due to increased model complexity.

Model	Unet_128	Unet_256	Resnet_6blocks	Resnet_9blocks
SSIM	0.6246	0.623	0.634	0.6428

Table 9: Comparing Unet and Resnet