Satellite Image to High Resolution Map Translation

Mohammad Shamoon
New York University
ms12736@nyu.edu

Umesh Deshmukh
New York University
urd7172@nyu.edu

Aradhya Alamuru
New York University
aa9405@nyu.edu

Abstract

This paper outlines a hybrid architecture designed and implemented to convert the satellite images to high resolution map images. The hybrid architecture consists of a Pix2Pix cGAN model and an SRGAN model. Satellite images are used as the input dataset for the Pix2Pix model, which transforms them into map images. These map images are then used as the input dataset for the SRGAN model, improving the resolution of the map image. This paper is an in-depth study into the different architectures of cGANS, and outlines our efforts, experiments with hyper parameter tuning and the corresponding results.

1 Introduction

A Generative Adversarial Network (GAN) is a Machine Learning framework used to train generative models.A Conditional Generative Adversarial Network (cGAN) is a type of GAN that involves the conditional generation of images by a generator model.

While GANs rely on a generator that learns to generate new images, and a discriminator that learns to distinguish synthetic images from real images, in cGANs, a conditional setting is applied, meaning that both the generator and the discriminator are conditioned on some sort of auxiliary information (such as class labels or data) from other modalities. As a result, the ideal modal can learn mapping from inputs to outputs by being fed with different contextual information.

The Pix2Pix GAN is a general approach for image-to-image translation, based on the cGAN, where a target image is generated. The Pix2Pix GAN changes the loss function so that the generated image is both plausible in the content of the target domain, and is a plausible translation of the input image.

SRGAN uses a perceptual loss function which consists of an adversarial loss and a content loss, for single image super-resolution. The adversarial loss pushes the solutions to the natural image manifold using a discriminator network that is trained to differentiate between the super-resolved images and original photo-realistic images. In addition, a content loss motivated by perceptual similarity is used instead of similarity in pixel space.

Our initial paper idea was to compare the output of the different cGANS, but we soon realized that due to the differences in the dataset and the use case involved, it wasn't feasible. Finally, we decided to create a hybrid algorithm for converting Satellite images into high resolution map images.

2 Methodology

Our approach first implements the cGAN [3] based Pix2Pix [1] image to image translation model that converts satellite images to map images (like google maps). And after getting the output from Pix2Pix, we improve the resolution of the image by providing the input to SRGAN [2] model, that

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

upscales the image. Once we run the dataset, we get images that are upscaled by a factor of four.

2.1 Pix2Pix Image to Image Translation

Pix2Pix is a cGAN based generative adversarial network that has U-net as generator and Patch GAN classifier as discriminator. Both trained in tandem, the generator tries to mimic the real data distribution, while the discriminator network learns to classify real from the fake (generated).

After training, the generator inputs random noise to output realistic images like the ones in the dataset. While the generator produced realistic-looking images, we had no control over the type or class of generated images. Thus cGAN, an extension of the GAN architecture generated images conditioned on class labels.
While the generator is fed a random-noise vector conditioned on class label, the discriminator is fed real, or fake (generated) images conditioned on the class label.
Pix2Pix GAN further extends the idea of the cGAN, where images are translated from input to output image, conditioned on the input image, performing a paired Image-to-Image Translation.
The generator of every GAN is fed a random-noise vector sampled from a uniform distribution, whereas the Pix2Pix GAN eliminates the noise vector concept completely.

In Pix2Pix, an image is input to the generator network which then outputs a translated version. The discriminator is a conditional discriminator which is fed a real or fake (generated) image that has been conditioned on the same input image, that is fed to the generator.
The goal of the discriminator is to classify whether the pair of images is real (from the dataset) or fake (generated). The final objective of the Pix2Pix GAN remains the same as that of all the GANs. It too, seeks to fool the discriminator, such that the generator learns to translate images perfectly.

Given an input edge image (before translation) x, the generator G learns to produce a translated satellite photo $G(x)$ similar to the map image y. The discriminator D learns to classify the generated image $G(x)$ conditioned on input x, as fake; the real image y, conditioned on input x as real.
Unlike an uncontrolled GAN, both the generator and discriminator observe the input-edge map. The noise vector which helps generate different outputs by adding randomness in GAN architectures is not of much use in Pix2Pix GAN, so it is removed. The minor stochasticity in the output of the generator is kept, by adding Dropout in the Generator Network.

2.1.1 Pix2Pix Generator Architecture — U-Net

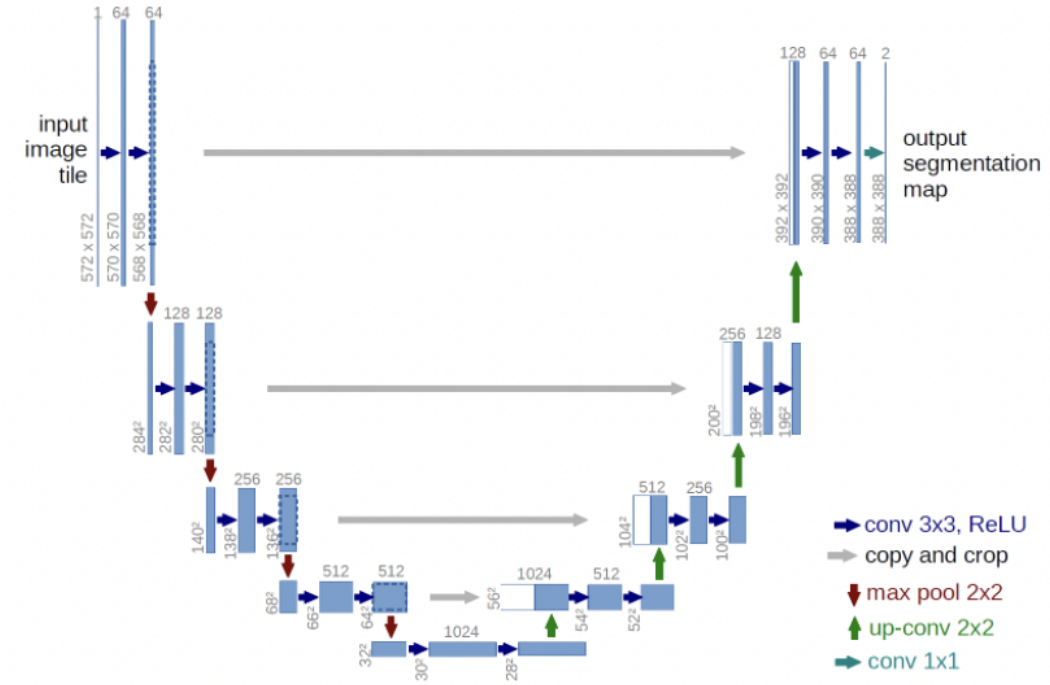
The neural architecture that the Generator uses is the U-Net, which is basically Encoder-Decoder with skip connections between layers in the Encoder and corresponding layers in the Decoder. We have used a U-Net 256.

The Encoder is the convolutional layers to the left of the network. The role for those layers is to extract core features of the image and map those features to the bottleneck latent space at the middle of the network (an 1024-D array). This latent space is the encoded form which contains the most important information of the input image.

The Decoder is the transpose convolutional layers to the right of the network. Those layers map encoded latent space oft he image back to a full-size image. Those layers do not simply output the same input image, but they are trained to map the encoded features to an image with another representation. For example, the Encoder encodes the information of a dog photo, and then the Decoder maps the encoded information for he dog to a drawing of the same dog. Both input and output have the same information, but they have a different representation: a photo, and a drawing.

To make the training more stable, extracted information from the Encoder network is concatenated to the corresponding layers in the Decoder network. This ensures that the Decoder has sufficient information to map the latent space to a realistic image.

Figure 1: Pix2Pix U-Net Generator Architecture



Since the Encoder and Decoder are in the same network, we can train U-Net end-to-end.

2.1.2 Convolutional Neural Network

In Pix2Pix and in cGAN, the Discriminator is still a binary Convolutional Neural Network. The difference between Discriminator in Pix2Pix with that of the original GAN is that in Pix2Pix not only takes the examined image y but also the conditional image x as the inputs.

2.1.3 Loss Function

In defining the objective for the training process, in the Image Translation task, the GAN training scheme is almost the same as the original GAN, except now, we have conditional input and an additional L1 loss to ensure the generated image is not too different from the expected output.

GAN Loss

Just like the original GAN, optimizing this loss will force the generator to produce results with overall distribution close to that of the image representation in the dataset and thus improve the structural quality of the Generators output.

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(x, z))] \tag{1}$$

L1 Loss

By using pixel-wise loss between two images, this loss forces the output image to be as close to the expected output as possible. I.e improving the minor details of the output.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1] \tag{2}$$

Final Loss

Combining GAN loss and the L1 Loss for the final Loss for the entire algorithm.

$$G^* = arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

2.2 SRGAN (Super Resolution Generative Adversarial Network)

The motive of the SRGAN architecture is to recover the finer textures from the image when its upscaled, so that its quality is not compromised. Other methods such as Bilinear Interpolation can be used to perform this task, but they suffer from image information loss and smoothing.

2.2.1 Training Procedure

: The high resolution images are processed to get down-sampled low resolution images. Now both the high and low resolution images are used in the training data set. Passing the low resolution images through the Generator which up-samples and gives the Super Resolution Images. The Discriminator distinguishes the high resolution images and back propagates the GAN loss to train the discriminator and the generator.

2.2.2 Network Architecture

- **Residual Blocks**: Since deeper networks are more difficult to train, the residual learning framework eases the training of these networks, and enables them to be substantially deeper, leading to improved performance. 16 residual blocks are used in the Generator.
- **Pixel Shuffler x2**: This is feature map upscaling. 2 sub-pixel CNNs are used in the Generator. Upscaling of Up sampling are the same.
- **PRelu (Parameterized Relu)**: Using Relu in place of Relu or LeakyRelu, which introduces learnable parameters that makes it possible to adaptively learn the negative part coefficient. $k3n64s1$ is 3 Kernel, 64 channels and 1 stride.
- **Loss Function**: The perceptual loss is a combination of both adversarial loss and content loss. The formulation of this loss can be interpreted as perpetual loss function LSR which is critical to the performance of the generator network. While lsr is commonly modeled based on the MSE[10, 48], that is an improvement on Johnson [7], and Bruna [4], designed to assess a solution with respect to perceptual relevant characteristics.

$$\ell^{SR} = \underbrace{\ell^{SR}_{X}}_{\text{content loss}} + \underbrace{10^{-3}\ell^{SR}_{Gen}}_{\text{adversarial loss}}$$

- **Content Loss**: The pixel wise MSE is calculated as:

$$\ell^{SR}_{MSE} = \frac{1}{r^2WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} (I^{HR}_{x,y} - G_{\theta_G}(I^{LR})_{x,y})^2$$

  This is the most widely used optimization target for image SR on which state of the art approaches rely. While achieving particularly high PSNR, solutions for MSE optimization problems often lack high frequency content which results in perceptually unsatisfying solutions with overly smooth textures.
  Instead of relying on pixel-wise losses, a loss function which is closer to perceptual similarity, a GFF loss based on the ReLU activation layers of the pre-trained 19-layer VGG network.

[6][4][7]

$$\ell^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR})_{x,y}))^2$$

The VGG loss is defined [2] as the Euclidean distance between the feature representation of a reconstructed image G_{θ_G} (ILR) and the reference image IHR. Here $W_{i,j}$ and $H_{i,j}$ describe the dimensions of the respective feature maps within the VGG network.

- **Adversarial Loss**: In addition to the content losses, a generative component of our GAN is added to the perceptual loss. This encourages our network to favor solution that reside on the manifold of natural images, by trying to fool the discriminator network. The generative loss is defined based on the probabilities of the discriminator over all the training samples as:

$$\ell^{SR}_{Gen} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$

2.2.3 Generator Architecture

The generator architecture of the SRRESNET generator network consists of the low-resolution input, passed through an initial convolutional layer of 9x9 kernels and 64 feature maps followed by a parametric Relu Layer. The entire generator architecture makes use of the Parametric Relu as a the primary activation function, for the reason that it is one fo the best non-linear functions for the task of mapping low resolution images to high resolution images. Although ReLU can also be used, issues such as dead neurons when values less than zero are mapped directly to zero arise. Alternatively the Leaky ReLU, where values less than zero are mapped to a number set by the user can be chosen. However, in the case of parametric ReLU, we can let the neural network choose the best value by itself and is hence preferred in this scenario.

The next layer of the feed-forward fully convolutional SRRESNET model utilizes a bunch of residual blocks, where each of these contains a convolutional layer of 3x3 kernels and 64 feature maps, followed by a convolutional layer with batch normalization, and a final element wise sum method. The element wise sum method uses the feed-forward output along with the skip connection output for providing the final resulting output.

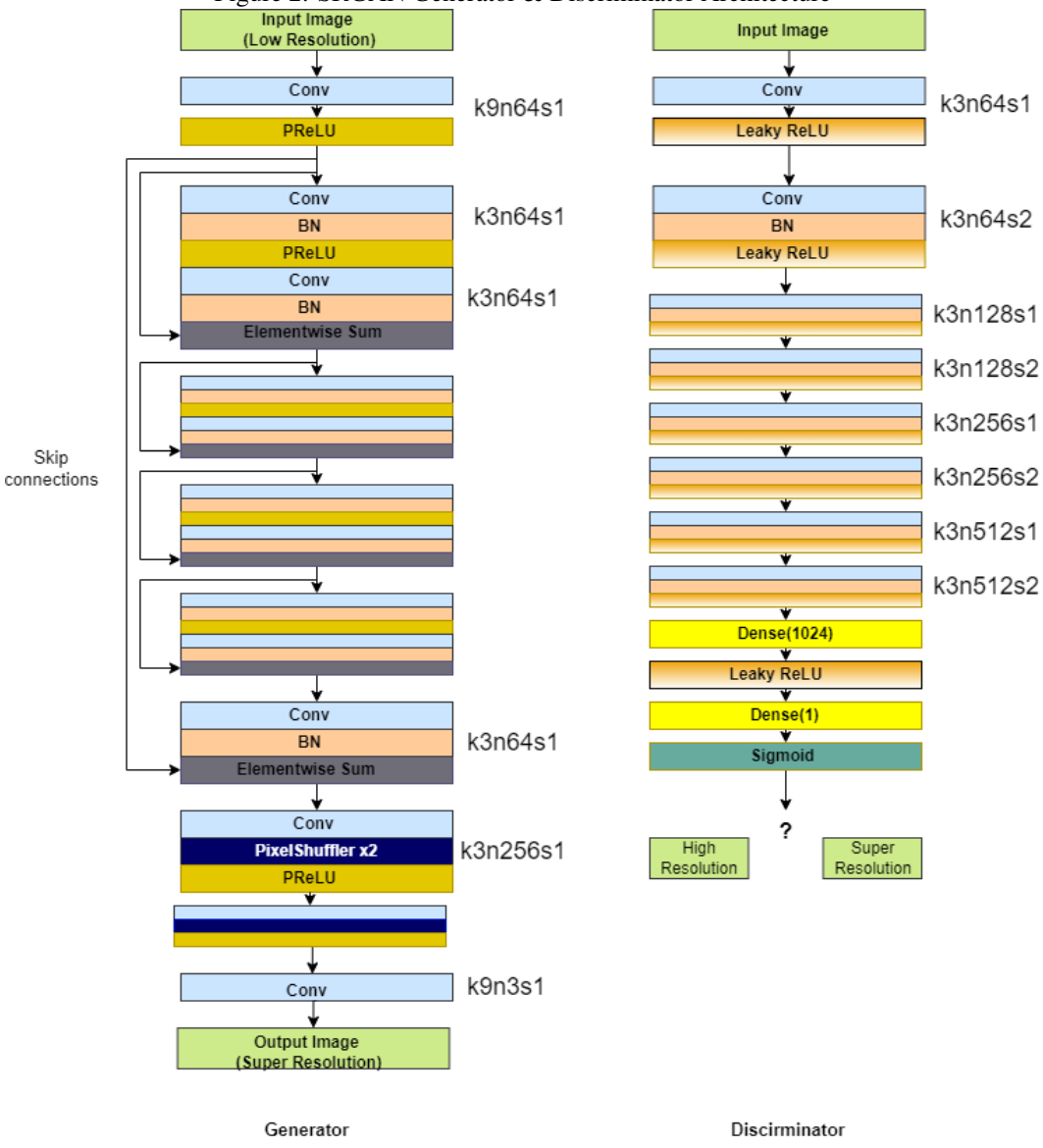
A key aspect of the neural network architecture is that each of the convolutional layers make use of similar padding sot hat the size of the following inputs and outputs are not varied. Unlike other fully convolutional networks like the U-Net architecture [1], often utilize pooling layers for reducing the image size. This is not required for our SRGAN model because the image size does not need to be reduced. And in fact, the opposite is what we're trying to achieve. Once the residual blocks are constructed, the rest of the generator model is built, as shown in the figure [2]. A pixel shuffler in this generator model architecture is used after the 4x up sampling of the convolutional layer to produce the super-resolution images. The pixel shufflers take values from he channel dimension and stick them into the height and width dimensions. Both the height and width are multiplied by two, while the channel is divided by two in our case.

2.2.4 Discriminator Architecture

The Discriminator architecture is constructed in the best way to support a typical GAN procedure: both the generator and discriminator are competing, and they are both improving simultaneously. Which the discriminator tries to find the fake images, the generator tries to produce realistic images so that it can escape the detection from the discriminator. The working in the case of SRGANs is similar, where the generative model G with the goal of fooling a differentiable discriminator D that is trained to distinguish super-resolved images from real images.
The discriminator model that is constructed aims to solve the adversarial min-max problem. The general idea for the formulation of this equation can be interpreted as:

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{l^{HR} \sim p_{train}(I^{HR})}[\log D_{\theta_D}(I^{HR})] + \mathbb{E}_{l^{HR} \sim p_G(I^{LR})}[\log(1 - D_{\theta_D}(G_{\theta_G}(I^{HR})))]$$

Figure 2: SRGAN Generator & Discriminator Architecture



An initial convolutional layer is used, followed by a Leaky ReLU activation function. The alpha value for the Leaky ReLU is set to 0.2 for this structure. Then we have a bunch of repeating blocks of convolutional layers, followed by the batch normalization layer and the Leaky ReLU activation function. Once we assemble five of these repetitive blocks, we have the dense layers followed by the sigmoid activation function for performing the classification action. The initial starting convolutional size is 64x64, which is multiplied by 2 after two complete blocks each until we reach we reach the 8x upscaling factor of 512x512. This discriminator model helps the generator to learn more efficiently and produce better results.

2.3 Architectural Guidelines for Stable GANs:

- Any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator) are replaced
- Batchnorm is used in both generator and the discriminator

- Fully connected hidden layers are removed for deeper architectures
- ReLU activation is used for all layers except for the output, which uses Tanh
- LeakyReLU activation is used in the discriminator for all the layers

2.4 Difficulties of Training GANs

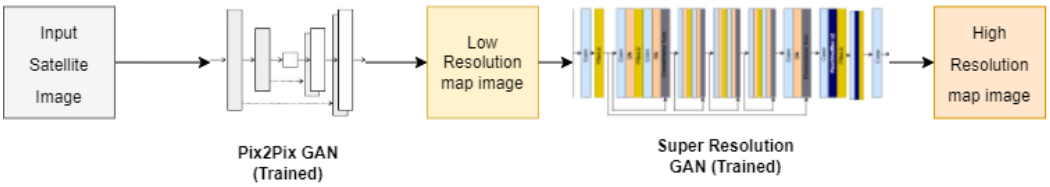
GANs are difficult to train because both the generator and the discriminator model are trained simultaneously. Meaning, improvements to once model come at the expense of the other model. The goal of training two models rests on finding a point of equilibrium between the two competing concerns.

Every time the parameters of one of the models are updated, the nature of the optimization problem that is being solved changes. This has the effect of creating a dynamic system. But in GAN, every optimization process is seeking not a minimum, but an equilibrium.

The largest problem is the issue of non-convergence, where GANs may suffer from finding a point of equilibrium, and oscillating between the generation. Another challenging issue is where multiple inputs lead to the generation of the same output, referred to as mode collapse.

One main concern is there are no good objective metrics for evaluating whether a GAN is performing well during training. E.g reviewing loss is not sufficient. Instead, the best approach is to visually inspect generated examples and use subjective evaluation. [8]

3 Our Experiments



3.1 Data Pre-processing

Pix2Pix Data Pre-processing

The input data for Pix2Pix is resized to 256x512, random crop with a probability of 0.5, normalized with $[0.5, 0.5, 0.5]$. The dataset images are a combination of satellite images and its corresponding map images stacked side by side. The images are separated into two for training. This is required to ensure that when we apply random flip, the input and the output image do not get swapped.

SRGAN Data Pre-processing

We had to create a brand new dataset for the SRGAN input, by removing the satellite map images. We processed the dataset used for Pix2Pix and removed the satellite image which is the left half. Then we normalized the image with mean of $[0.485, 0.456, 0.406]$ and standard deviation of $[0.229, 0.224, 0.225]$. And then we reduced the resolution by a factor of 4. This resulting image is used as the input for the SRGAN model. The right half image becomes our target iamge and the low resolution image is the input to our SRGAN model.

3.2 Hyper Parameter Tuning

Pix2Pix Hyperparameter Tuning

The hyperparameters are chosen after various trial and error runs. Firstly, the hyperparameters chosen were based on the preliminary result mentioned in [1] that indicate the batch size, learning rate, β_1,

Table 1: Pix2Pix Hyperparameter Tuning

Batch Size	Learning Rate (Lr)	β_1	β_2	L1 Lambda	Epochs
1	0.0002	0.5	0.999	100	200 (Final)
2	0.0001	0.5	0.009	100	300
3	0.00001	0.5	0.999	100	150
4	0.00002	0.5	0.999	50	50
5	0.00002	0.5	0.900	100	150
6	0.00001	0.5	0.999	100	150

Table 2: SRGAN Hyperparameter Tuning

Batch Size	Learning Rate (LR)	β_1	β_2	Decay Epoch	Epochs
16	0.000001	0.9	0.999	100	200
2	0.0001	0.5	0.009	100	100
3	0.0001	0.5	0.099	100	100
4	0.000001	0.5	0.099	100	100
5	0.00001	0.5	0.099	100	100
6	0.000002	0.5	0.09	100	100
8	0.00008	0.5	0.99	100	500 (Final)

β_2 for Adam optimizer and L1_lambda as indicated in Table 1.

According to the original author's recommendation, we have tested our model with the hyperparmeters as follows: a batch size of 1, learning rate of 0.0002, $\beta_1 = 0.5$, $\beta_2 = 0.999$, L1_labda as 100.

This gave us enough results, sufficient to translate our image A to image B. And this output (image B) is used as our input to the SRGAN model, which is trained solely on image B, that is originally preprocessed from the map dataset.

From the original paper [1], a mix of GAn and L1 showed result of less blurred image and then later on, in our next model (based on [2]), this is improved by using a feature matching loss. This feature matching loss is related to perceptual loss, allowing for better performances by our SRGAN model.

Using a number of iterations while using various values for batch size, learning rate, β_1, β_2, L1 and also comparing the generator loss against the discriminator loss, we reach the conclusion that the original hyperparameters mentioned in the paper [1], work well on our map dataset.

SRGAN Hyperparameter Tuning

In SRGAN, we have generated our LR images from down samfpflfing the HR images with a factor of 4 and scale the range of the LR images between $[0, 1]$ and HR images $[-1, 1]$, as originally mentioned in [2], the proposed paper.

In SRGAN, VGG feature maps were also rescaled by a factor of $\frac{1}{12.75}$, to obtain VGG losses of a scale. For the optimizer, we have used Adam with $\beta_1 = 0.9$.

The proposed model uses the hyperparameters which are marked as *Final* in the table [2], which have been used: a batch size of 8, learning rate of 0.00008, β_1 of 0.5, β_2 of 0.009, decay epoch after 100 iterations. After some experimentation, we are not able to tune these hyperparameters better.

3.3 Joint Optimization Parameters for Pix2Pix & SRGAN

Learing Rate

Learning rate which we choose for our model is based on generator loss and discriminator loss that we reached with a smaller number of epochs but as we know GANs are notoriously difficult to train so we need to find optimal number of epoch value while choosing learning rate that help us in converging our model fast, that is important for running high resolution images of satellite maps in our Pix2Pix model.

This also important for SRGAN model where we need convergence faster, as here, we are training our model with map images, which do not have satellite images so to improve the resolution we need up sampling. Learning rate of SRGAN model which we define as 0.00008 provides a better result as compared to the original paper[2] recommended learning rate.

Optimizer

We have used Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ for discriminator and generator respectively in our Pix2Pix GAN model.

And Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ for discriminator and generator respectively in our SRGAN model. Again, we have tried to implement various values of β_1 and β_2, but none of these have provided any improvement. These values still have relevant improvement in the images as compared to the other combinations of hyperparameters.

Batch Size

In choosing batch size in GAN we try to implement it with lower batch size because training in GAN [8] is difficult, so we focus in choosing less batch size even though we have GPU computation power that reduced the effect of increase computation time due to batch size[5]. So as described in [1] base Pix2Pix paper we have use batch size of 1 in Pix2Pix and batch size of 8 in SRGAN model due to the constraint of training time.

Data Augmentation

In data augmentation we have used different transformation method for our different models. For Pix2Pix, to make our image suitable for input, we resize it to 256x512 as this data has two images embedded in single image which we have already mentioned the in data processing phase where we have separated our image A and image B for our generator and discriminator architecture. Then we use CentreCrop, Vertical flip and normalization of images. One thing we need consider, is that we need to make sure that image A and image B should remain at their respected position so that while splitting we need to make sure that ImageA must be satellite image and ImageB must be map image.

Data Regularization

Regularization basically helps to avoid overfitting, as we know that the weights of the generator will be updated via both adversarial loss via the discriminator output and L1 loss via the direct image output. The loss scores are added together, where the L1 loss is treated as a regularizing term and weighted via a hyperparameter called lambda, set to 100 and we have also used drop out of 0.5 after first three residual layers in generator architecture in Pix2Pix model.

4 Results

Our paper outlines our experimentation and implementation of our novel idea of creating hybrid architecture that consists of a Pix2Pix model and SRGAN model.
In Figure [1], the left image is the satellite image which is the input to our Pix2Pix model, the center image is the output of our Pix2Pix model, which we use as the input for the SRGAN model, that upscales our images by a factor of 4, the image on the right is the output image from the SRGAN model.
By visual obeservation, we can say that the image has be upscaled and the resolution is improved, but there still are some artifacts in the output image, which is not observed in the high resolution image.

In Pix2Pix, the generated output images have distortions which can be improved in future research. There is still scope of improvement in this hybrid approach.
Also, we can implement diffusion networks, which are considered an advanced version of GANs.

Figure 3: Satellite image (left), Generated Map LR (center), Super Resolution Map (right)

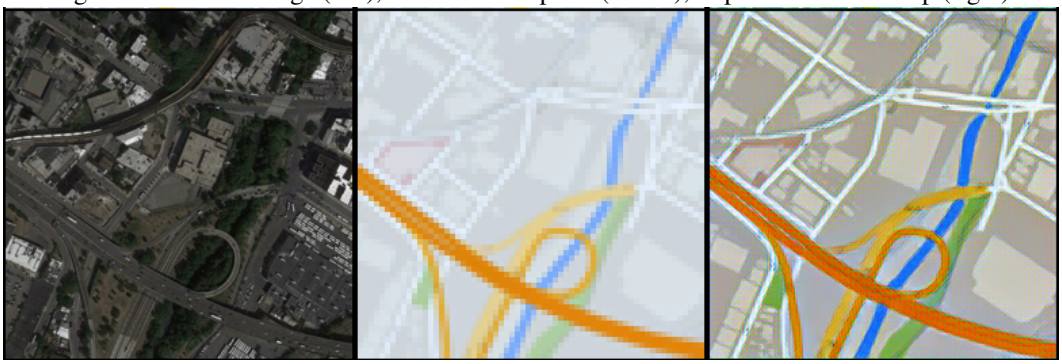


Figure 4: Satellite image (left), Generated Map LR (center), Super Resolution Map (right)

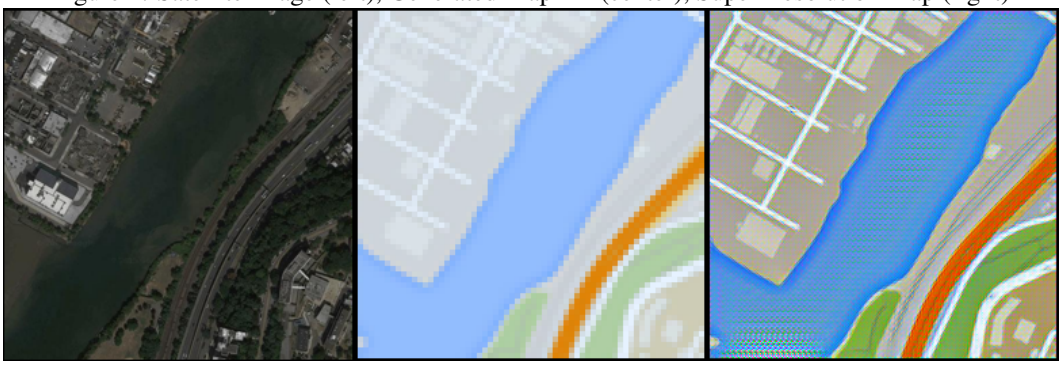


Figure 5: Satellite image (left), Generated Map LR (center), Super Resolution Map (right)

References

[1]. Isola, P., Zhu, J.Y., Zhou, T. and Efros, A.A., 2017. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).

[2]. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z. and Shi, W., 2017. Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4681-4690).

Figure 6: Train vs Test Discriminator Loss

Figure 7: Train vs. Test Generator Loss

[3]. Mirza, M. and Osindero, S., 2014. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

[4]. J. Bruna, P. Sprechmann, and Y. LeCun. Super-resolution with deep convolutional sufficient statistics. In International Conference on Learning Representations (ICLR), 2016. 2, 3, 5

[5]. N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” CoRR, vol. abs/1609.04836, 2016.

[6]. L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In Advances in Neural Information Processing Systems (NIPS), pages 262–270, 2015. 3, 5

[7]. J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super- resolution. In European Conference on Computer Vision (ECCV), pages 694–711. Springer, 2016. 2, 3, 4, 5, 7

[8]. Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[9]. GitHub Code Repository: https://github.com/Shamoonmohd/Gans-For-Image-Generation.

Figure 8: Pix2Pix - Input & Output of Generator (1)

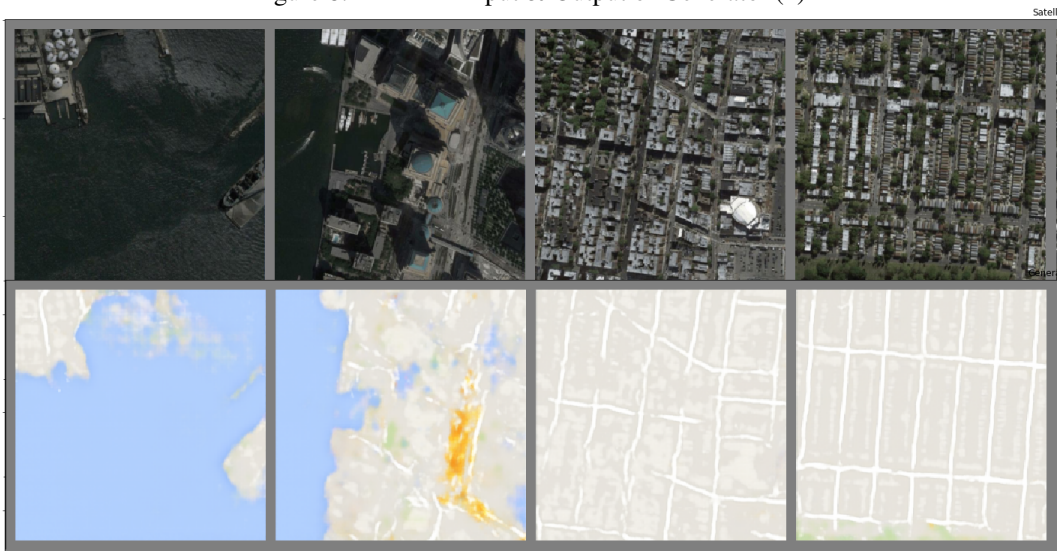


Figure 9: Pix2Pix - Input & Output of Generator (2)

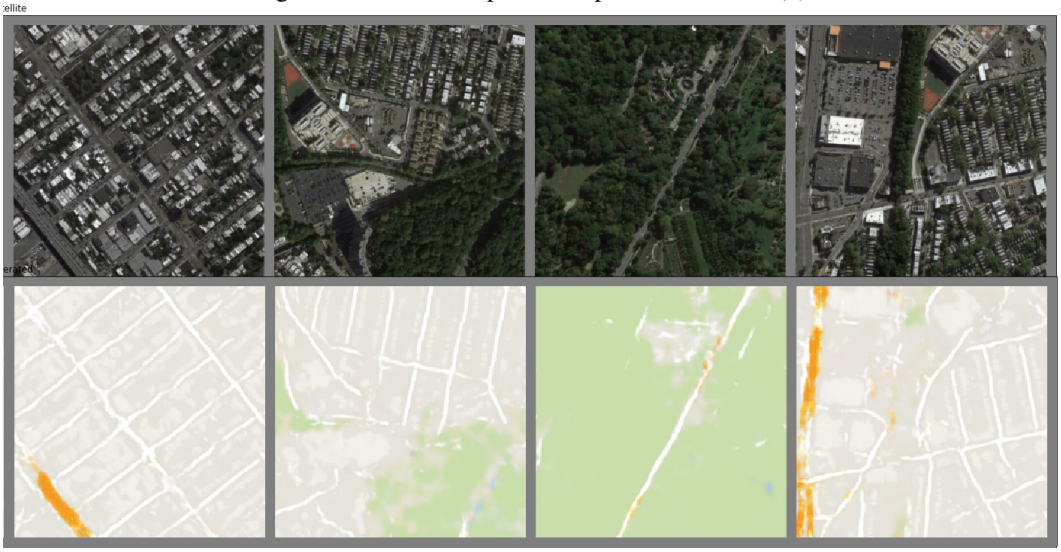


Figure 10: SRGAN Output: After 50 Epochs

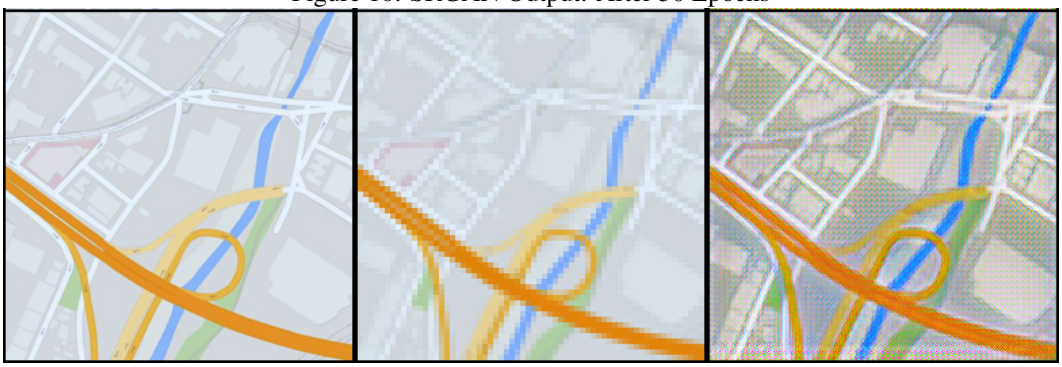


Figure 11: SRGAN Output: After 100 Epochs

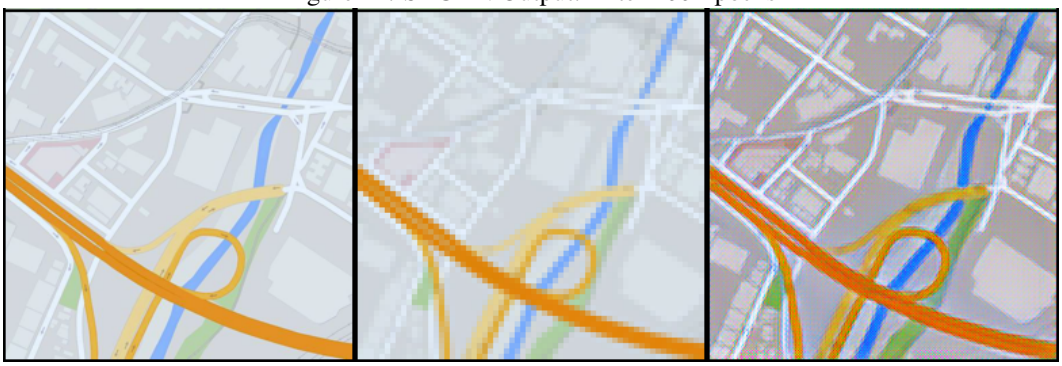


Figure 12: SRGAN Output: After 150 Epochs

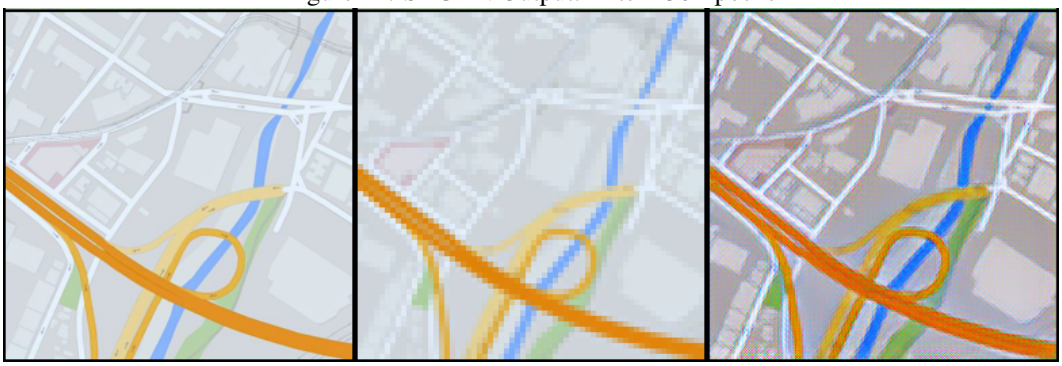


Figure 13: SRGAN Output: After 200 Epochs

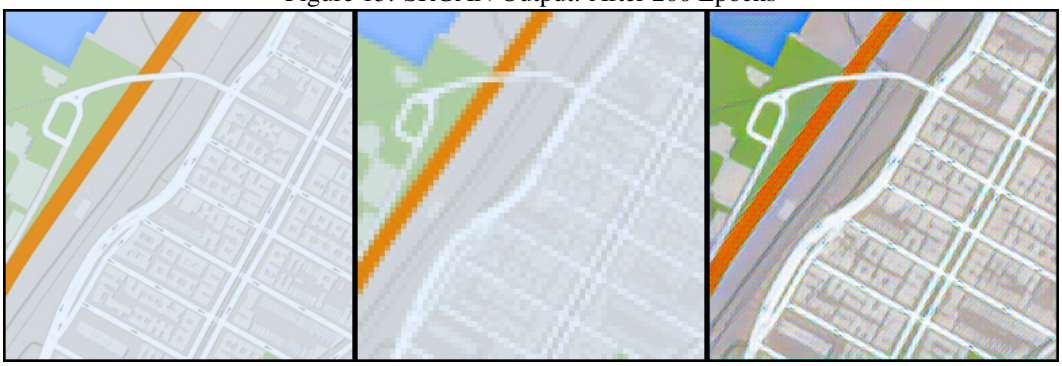


Figure 14: SRGAN Output: After 250 Epochs

Figure 15: SRGAN Output: After 300 Epochs

Figure 16: SRGAN Output: After 350 Epochs

Figure 17: SRGAN Output: After 400 Epochs

Figure 18: SRGAN Output: After 450 Epochs