Base Model for Cloud Cover Removal in Multispectral Satellite Imagery

Paritosh Tiwari

***Abstract*—Cloud cover in satellite images is a major problem. It either cause loss of information through complete obstruction (thick clouds) or blurry effects (semi-transparent/thin clouds). Failure to mask out clouds from images has a significant negative impact on subsequent analyses such as change detection, land-use classification, crop monitoring in agriculture etc. In this paper, I propose a solution to overcome loss of information due to full or partial obstruction by using SAR images to replace contaminated pixels in multispectral images. This image to image translation problem can be solved using conditional adversarial networks. Due to limited availability of resources, I have presented in this paper a base model, which has the potential to be applied to our problem with a few adjustments. This base model has been trained and tested on a relevant and easily available dataset and the results can easily be extrapolated to cover our problem statement.**

***Index Terms*—Generative Adversarial Network (GAN), Geospatial Analysis, Multispectral Image (MSI), Remote Sensing, Sentinel-1, Sentinel-2, Synthetic Aperture Radar (SAR)**

I. Introduction

Of the many hurdles in remote sensing and its applications, cloud cover detection and its removal is a major one. Clouds contaminating multispectral images limit their usage. Based on the percentage of cloud cover in an image, the information loss is either tolerable or renders the image useless in case the percentage exceeds 70%. This poses a major problem in remote sensing applications, especially monitoring of crops and agricultural land. In tropical and some temperate regions of the earth, which get significant rainfall every year, monitoring during the monsoon season becomes virtually impossible due to thick cloud cover over the region.

SAR or Synthetic Aperture Radar images have an advantage over multispectral images, in that they use radio waves, which can penetrate cloud cover and obtain images of earth's surface. However, they are not suitable for monitoring crop coverage and other agricultural applications.

The idea behind my proposal is to merge the properties of these two types of images, i.e. the crop monitoring properties of multispectral images and the cloud cover penetrating properties of SAR images. My model will generate a multispectral image without cloud cover of a region of land, based on or conditioned on the SAR image of the same region. For this, I will use a conditional GAN (cGAN) as a base, which will learn to generate an MSI, given an SAR image, which can subsequently fool the discriminator when compared with the ground truth MSI. When trained sufficiently well, the MSI that this model generates can be used to replace the contaminated pixels in the actual MSI to give a cloudless image.

However, due to the limited availability of resources and data, I will be presenting in this paper a model that has been trained on a different set of image pairs, generated Google map images conditioned on greyscaled RGB satellite images. Although this will not achieve the original objective of the proposal, it will be sufficient to demonstrate the capabilities of the model when applied to the relevant data.

Paritosh Tiwari, is with the Computer Science Department, International Institute of Information Technology, Naya Raipur, Chhattisgarh 493661, India (e-mail: paritosh19300@iiitnr.edu.in).

II. Background

A. Generative Adversarial Network

Generative adversarial networks or GANs consist of two competing models, a generative model called a Generator, denoted by G, that captures the data distribution, and a discriminative model called a Discriminator, denoted by D, that estimates the probability that a sample came from the training data rather than the generator.

Given a set of data instances 'a' and a set of labels 'b', a generator captures the joint probability p(a, b), or just p(a) if there are no labels. A discriminator captures the conditional probability p(b | a). In its most basic form, a GAN takes random noise 'z' as its input. The generator then transforms this noise into a meaningful output. G and D are both trained simultaneously. We adjust parameters for G to minimize $\log(1-D(G(z))$ and adjust parameters for D to minimize $\log(D(x))$, as if they are following the two-player min-max game.

$$\min_G \max_D K(D,G) = E_x[\log(D(x))] + E_z[\log(1-D(G(z)))] \quad (1)$$

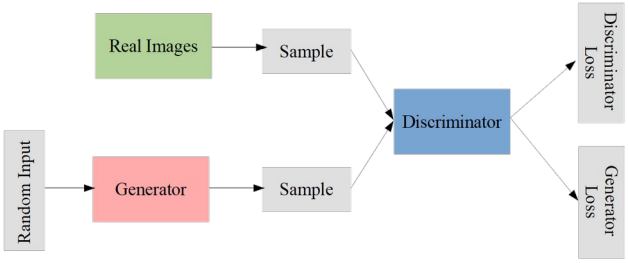

Figure 1: GAN

where, $K()$ is the value function, $D(x)$ is the discriminator's estimate of the probability that real data instance x is real, E_x is the expected value over all real data instances, $G(z)$ is the generator's output given noise z, $D(G(z))$ is discriminator's estimate of the probability that a fake data instance is real, E_z is the expected value over all random inputs to the generator (the expected value over all generated fake instances $G(z)$).

B. Conditional Adversarial Network

In an unconditioned generative model, there is no control over the general structure or type of the data being generated. In conditional GAN (cGAN), the generator learns to generate a fake sample with a specific condition or characteristics (for example, labels associated with an image) rather than a generic sample projected from an unknown noise distribution. Let us say the generator and discriminator are conditioned on some extra information 'y'. The conditioning i.e. the influencing of the model by some extra information y can be performed by pushing y as an additional input layer into both the discriminator and the generator. The objective function for this approach will be the following equation.

$$\min_G \max_D K(D,G) = E_x[\log(D(x|y))] + E_z[\log(1-D(G(z|y)))] \quad (2)$$

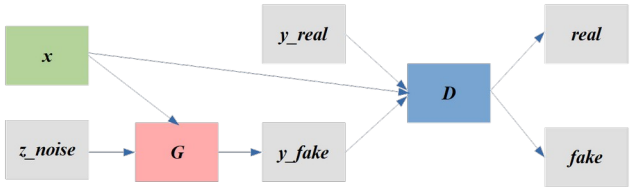

Figure 2: Conditional GAN

It is important to note that y is fed to both the generator and the discriminator instead of just the generator as one might expect. This is because the backpropagation algorithm pushes the loss function updates for the generator through the discriminator when the former is being trained. Therefore, even though the discriminator training is halted, it influences the generator updates. Hence, y is fed to D to maintain the context that the generator output is conditioned on y. Otherwise, the updates will not be adequately influenced by the fact that this is a conditional GAN.

Another observation is that the model can output images without the random noise input z, as we are feeding y as an input to G. But the pseudo randomization is necessary via z as otherwise the model will start producing deterministic outputs. The input noise z ensures that generator input is uniformly mapped over the conditioning information y.

C. U-Net

Use one A typical convolutional network with max pooling downsamples an input image to lower resolutions but at the same time increasing the processing depth in the form of filters. Subsequent convolutional layers result in an increased receptive field and the amount of "big picture" information. This approach is useful in understanding "what" is there in the image but in its operations, tends to lose the information of "where" the objects in the image are present. In the effort to recognize the image as a whole, it loses spatial information within the image.

A U-Net addresses the issue of loss of spatial information by countering the convolutional and max pooling downsampling with upsampling the image data. The architecture contains two paths. First path is the contraction path (also called as the encoder) which is used to capture the context in the image. The encoder is just a traditional stack of convolutional and max pooling layers. The second path is the symmetric expanding path (also called as the decoder) which is used to enable precise localization using transposed convolutions (upsampling). The final output can have a resolution identical to the input or any other if the user so desires.

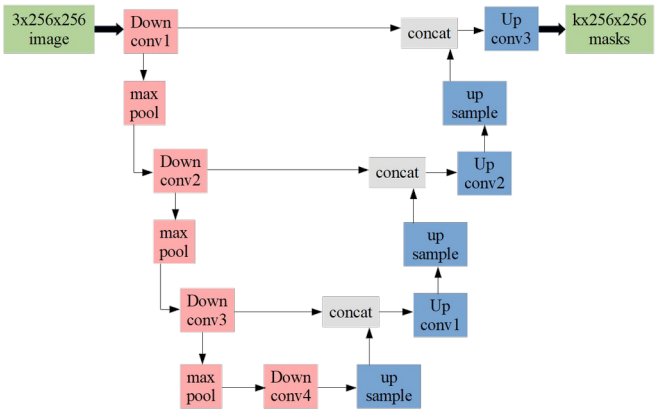

Figure 3: U-Net

To get better information on locations in the decoder, skip connections are used by concatenating the output of the upsample transpose convolutional layer with the output of the corresponding downsample convolutional layer at the same level in the encoder. This ensures that every remaining bit of spatial information in the encoder is used in the upsampling process. After every concatenation, a number of transpose convolutional layers are applied so that the model has a chance to assimilate the newly added information.

III. Architecture

Refer Figure 2 that portrays the architecture for conditional GAN. We use a U-Net as a generator (G) and a convolutional Patch-GAN as a discriminator (D).

As we are reproducing maps, it will be beneficial to use a generator that will focus on spatial information along with identification information, hence a U-Net for generation.

The Patch-GAN acts as a classifier to determine whether an image is the ground truth or generated by our model. It detects and punishes discrepancies in patches of a

fixed size instead of taking the entire image, which would result in a pixel by pixel comparison. It works on a 70x70 square or patch of the input image. The patch comparison does not hold our model back in terms of accuracy, as the results will show, while decreasing the computational cost.

I have used the “maps” dataset in this paper. This dataset is comprised of satellite images of New York and their corresponding Google maps pages. The image translation problem involves converting satellite photos to Google maps format. There are a total of 1097 images in JPEG format. Each image is 1,200 pixels wide and 600 pixels tall and contains both the satellite image on the left and the Google maps image on the right. Each image is loaded, rescaled, and split into the satellite and Google map elements. The result is 1,097 colour image pairs with a width and height of 256×256 pixels.

The generator is updated via a weighted sum of both the adversarial loss and an L1 loss, with a weighting of 100 to 1 in favour of the L1 loss. This is to encourage the generator strongly toward generating plausible translations of the input image, and not just plausible images in the target domain.

Typically, GAN models do not converge; instead, an equilibrium is found between the generator and discriminator models. As such, we cannot easily judge when training should stop. Therefore, we can save the model and use it to generate sample image-to-image translations periodically during training, such as every 10 training epochs. We train the model for 50 epochs to keep the training time low, although 200 epochs are recommended for optimal results. A batch size of 1 is used here. There are 1,097 images in the dataset. One epoch is one iteration through these images, with a batch size of one that means 1,097 training steps. The model will run for 50 epochs, i.e. a total of 98,500 training steps.

Trial runs suggest that the Gaussian noise z that we feed to the generator in order to ensure stochasticity in the output of our model, does not contribute to an effective strategy. The generator eventually learns to ignore the noise to achieve better performance, which do not improve the overall results. Instead, I found it favourable to introduce noise in the form of dropout layers in the net of the generator. However, only a minor increase in favourable results is seen with this modification.

IV. Results

The satellite image, the generated Google maps image and the ground truth image are presented here in a grid, in the same order, with the mentioned images arranged row wise and the columns containing different images from the same epoch. The images were plotted after every 10 epochs to track the improvement in the model while training it. Hence, Plot 1 represents the images after 10 epochs, Plot 2 after 20 epochs, and so on.

As we can see, the images tend to improve after 30 epochs. Though it was recommended by the community to train the model for at least 200 epochs, due to limitations stated earlier, the epochs were reduced to 50. Even then, we see that the generated images are not that far off from the ground truth images, with some even containing more information that the ground truth images. Sharpness of the generated images still remains an issue to be solved.

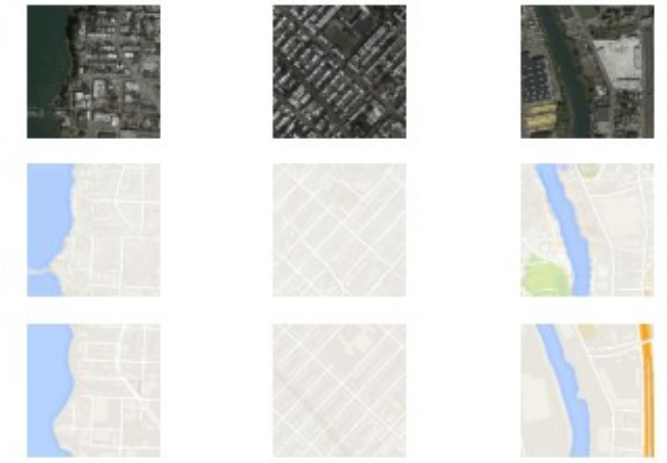

Figure 4: Plot 1 (10 epochs)

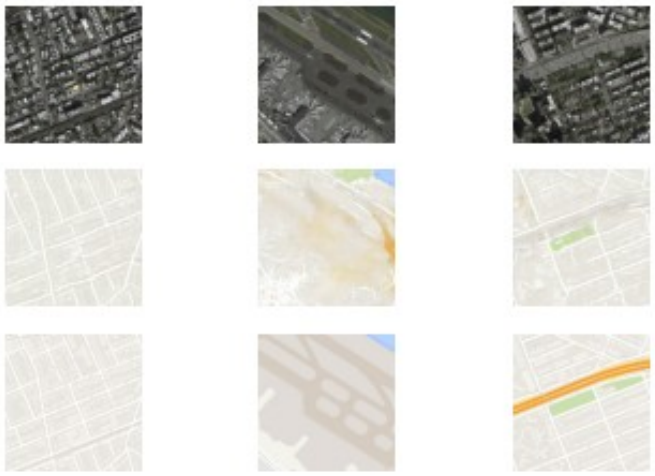

Figure 5: Plot 2 (20 epochs)

Figure 6: Plot 3 (30 epochs)

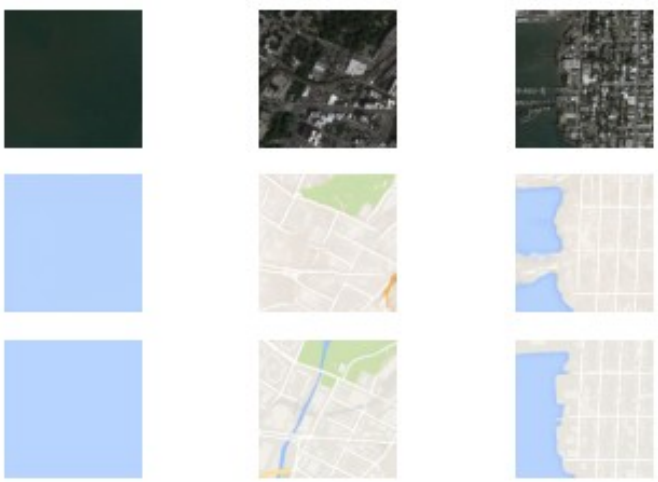

Figure 7: Plot 4 (40 epochs)

Figure 8: Plot 5 (50 epochs)

Next we have RGB histogram plots of selected pairs of generated and real map images from every plot presented previously. These histogram plots will help us get an idea of how similar two images in question are on a pixel level.

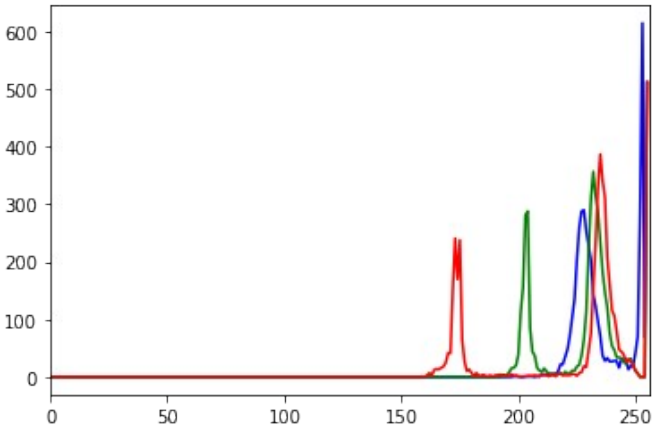

Figure 9: Plot 1, column 1, generated image

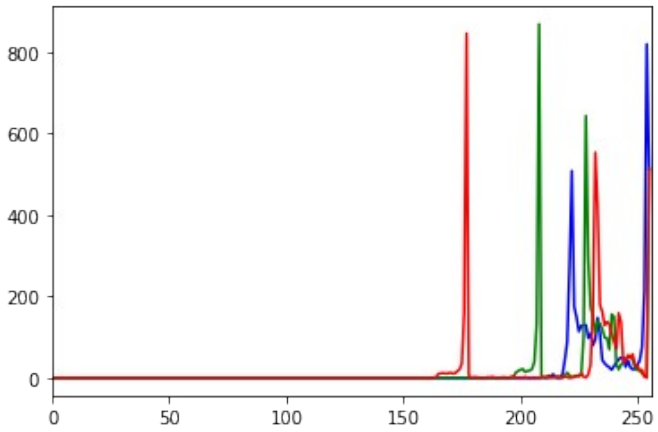

Figure 10: Plot 1, column 1, ground truth image

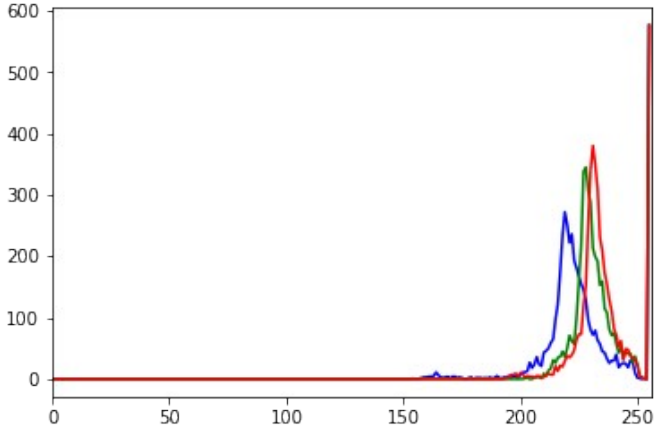

Figure 11: Plot 2, column 3, generated image

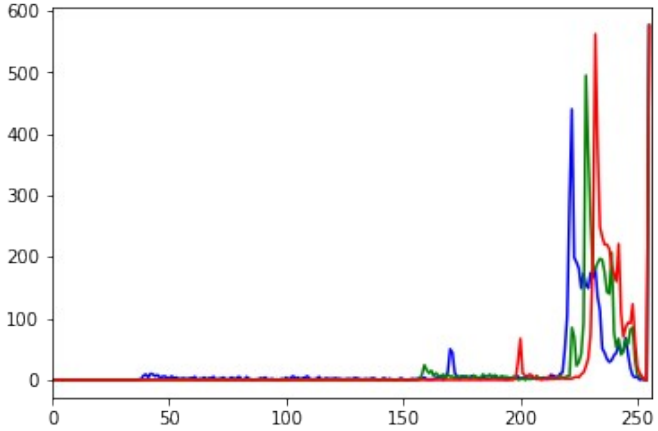

Figure 12: Plot 2, column 3, ground truth image

The histograms plot the pixel intensity values for the respective bands (red, green and blue), on the x-axis and the frequencies (number of pixels) on the y-axis. By looking at the histograms we can get an idea of the contrast and intensity distribution of the image.

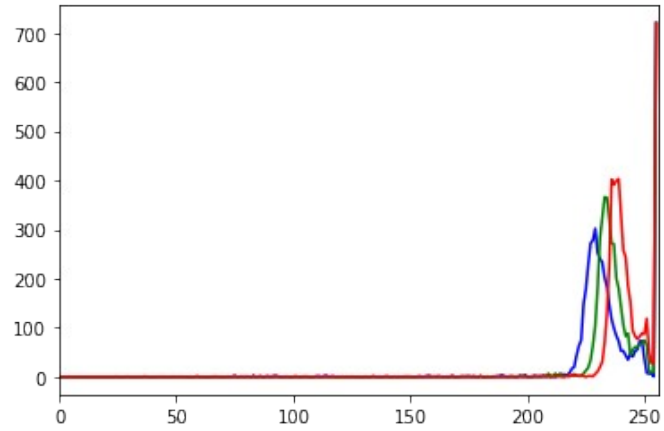

Figure 13: Plot 3, column 2, generated image

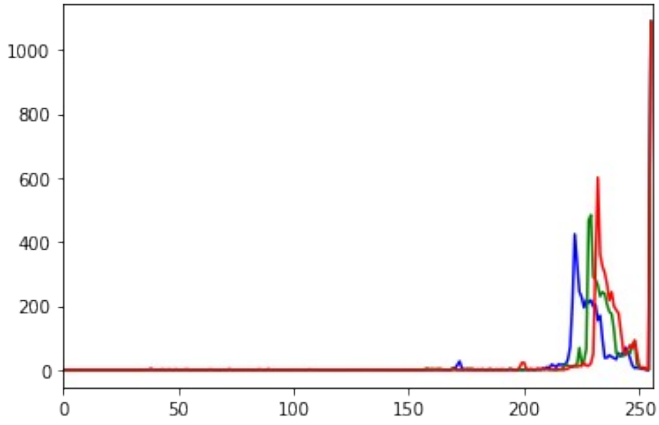

Figure 14: Plot 3, column 2, ground truth image

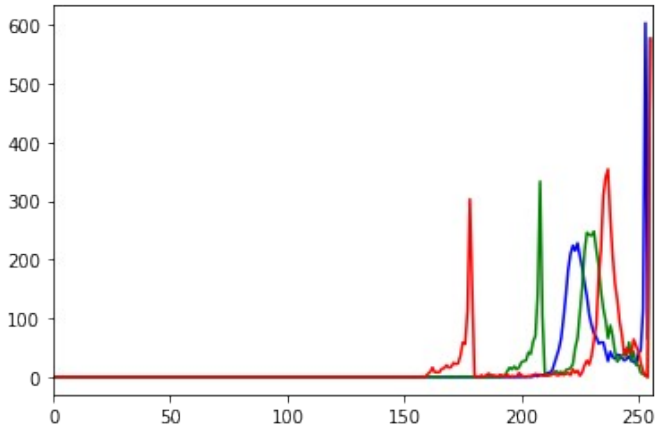

Figure 15: Plot 4, column 3, generated image

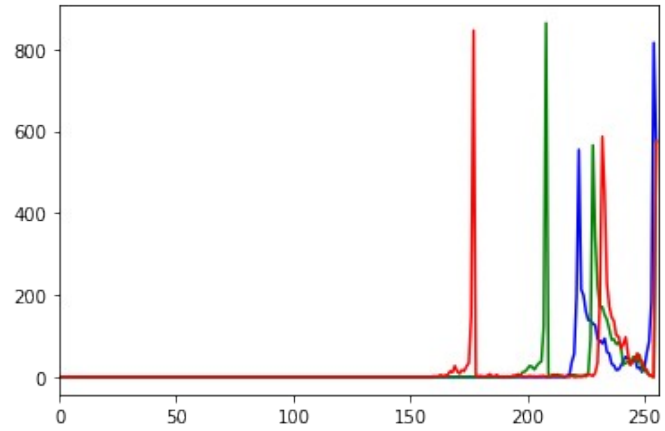

Figure 16: Plot 4, column 3, ground truth image

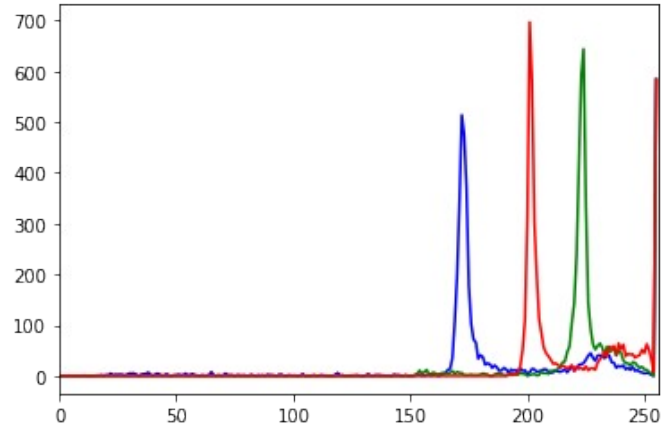

Figure 17: Plot 5, column 1, generated image

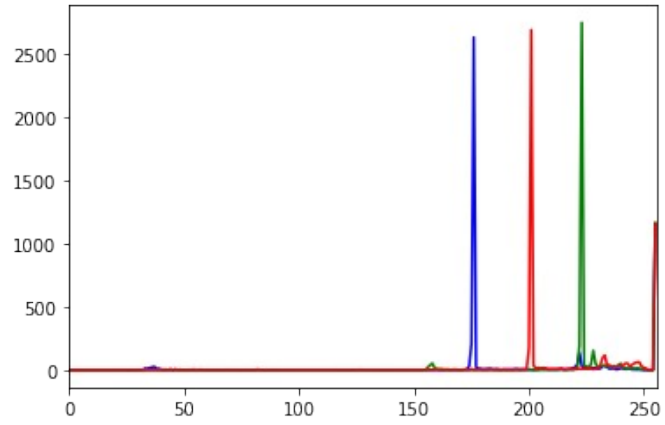

Figure 18: Plot 5, column 1, ground truth image

V. Conclusion

As we can see from the histogram plots, the ground truth images have sharp peaks on certain intensity levels whereas the generated images have more distributed intensity levels in comparison. This suggests that the actual Google map images give preference to sharp contrasts in order to distinguish features on a map. The generated incline more towards their parent satellite images as can be deduced from the intensity distribution. The satellite images represent real world optical data, and we don't see sharp changes in intensities in the real world.

This is due to the low number of epochs that the model was trained for in this paper. It can be safely said that the generated images would start mirroring the histograms of the ground truth map images if we were to run the model for at least 200 epochs. However, the intensities do peak in roughly the same areas and hence, we can say that our model performs as expected and can be used for the original proposal with some modifications.

The stochasticity of the model does remain an issue and we need to figure out a better input to project pseudo randomness in the generator inputs.

In this presentation, we did not force a loss function on the model but allowed it to adapt via the min max adversarial game construct with inherent binary crossentropy losses in each of the neural net families. We need to further study the effects of forcing a loss function like Wasserstein loss, or a tailored loss function, onto the model.

Enhancing the quality of the satellite images can be considered as an option. Pre-processing steps such as de-noising and histogram equalization of the satellite images are sure to, in turn, increase the quality of the generated images in earlier epochs.

In conclusion, I would like to say that I have successfully demonstrated the capabilities of my model in achieving the task at hand and also to further extrapolate it to the proposed solution. A few performance improvements could be made but the model on its own has showcased its required capabilities.

References

[1] Harshall Lamba. https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47/. Accessed, 2020-04-21.

[2] Paul-Louis Prove. https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d/. Accessed, 2020-04-21.

[3] https://developers.google.com/machine-learning/gan/. Accessed, 2020-04-20.

[4] https://scihub.copernicus.eu/. Accessed, 2020-04-15.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D.Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *arXiv:1406.2661*, 2014.

[6] M. Mirza, S. Osindero. Conditional Generative Adversarial Networks. *arXiv:1411.1784*, 2014.

[7] O. Ronneberger, P. Fischer, T. Brox. U-Net: Convolutional Netowrks for Biomedical Image Segmentation. *arXiv:1505.04597*, 2015.

[8] P. Isola, Jun-Yan Zhu, T. Zhou, A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv:1611.07004v3*, 2018