**Satellite-to-Map Image Conversion**

Saloni Sharma   
181B179

Stuti Verma  
181B220

Trijal Singh  
181B230

ABSTRACT— Pix2Pix GAN is an approach to training CNNs for image-to-image translation tasks. Image translation is a complex problem with many variables, this architecture allows for the generation of large images as compared to previous models and the capability to perform well on a variety of image-to-image translation tasks. We’ve used a Pix2ix GAN for the translation of satellite images to Google map images and have a final generator model to translate ad-hoc satellite images.

# INTRODUCTION

Pix2Pix is a GAN model used to convert images in one domain to images in another domain. GANs are made up of a Generator and a Discriminator. The Generator tries to produce new plausible fake images, and the Discriminator tries to identify whether they are real or not. This w

Creating maps is a very expensive and time consuming process; yet it is one of the most important sources of curated data. An accurate map must reflect all changes on the ground in a timely manner. Up-to-date geospatial data is continuously collected from flyover imaging air-crafts or satellites. Automatically generating maps from satellite images is an important task. There is a body of literature which tries to address this challenge. Maps have commercial value to companies in multiple sectors of the economy: ride-sharing companies (like Uber and Lyft), food delivery companies like (like DoorDash and GrubHub), national security agencies (like CIA, NSA, and FBI), and many other sectors of the economy. We created a database of pairs of satellite images and the corresponding map of the area. Our model translates the satellite image to the corresponding standard layer map image using three main model architectures:

(i) A conditional Generative Adversarial Network (GAN) which compresses the images down to a learned embedding,

(ii) A generator which is trained as a normalizing flow (RealNVP) model, and

(iii) A conditional GAN where the generator translates via a series of convolutions to the standard layer of a map and the discriminator input is the concatenation of the real/generated map and the satellite image.

In this work, we emphasize the importance of human-readability of a map and aim to construct accurate human-readable maps directly from a satellite/aerial image of the location. The satellite/aerial image specifies the zoom level and resolution of the required map. The style of the generated human-readable map is chosen to be the style of the publicly available Google Maps API. Satellite images are obtained from the publicly available Google Earth Engine datasets and its corresponding map is obtained using the Google Maps API. The process of querying these databases and aligning the satellite/aerial image with its corresponding map is described in the section on datasets.

# THE MODEL

## CNN

A CNN is an artificial neural network with some specialization for being able to detect patterns and make sense of them. This pattern detection is what makes CNNs so useful for image analysis. CNNs preserve the spatial aspect of the dataset.

## Image to Image Translation

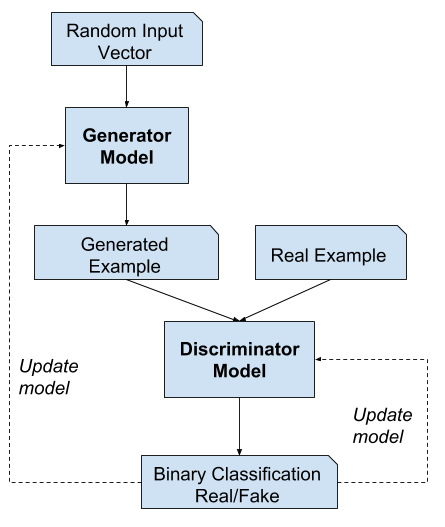
Image to image translation is the task of taking images from one domain and transforming them so they have the style (or characteristics) of images from another domain.

## Generative Modelling & Adversarial Machine Learning

Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset. Unlike discriminative models, which discriminates between different kinds of data instances, generative models can generate new instances. Adversarial machine learning is a machine learning technique that attempts to fool models by supplying deceptive input.

## Generative Adversarial Nets

Generative Adversarial nets are an approach to generative modelling using deep learning methods, such as CNNs. GAN is an architecture for automatically training a generative model by treating the unsupervised problem as supervised and using both a generative and discriminative model. They are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: a generator and a discriminator. The generator model gets trained to generate new examples. The discriminator model is responsible for classifying the examples as either real or fake (generated) (binary classification).   
The two models take part in a zero-sum game until the discriminator model is fooled nearly 50% of the time, which implies that the generator model is generating plausible examples. After training, the generator is kept and used to generate new instances.



## Pix2Pix

The Pix2PIx model is a type of conditional GAN, or cGAN, where the generation of the output image is conditional on an input, in this case, a source image. The discriminator is provided with both a source image and the target image and must determine whether the target is a plausible transformation of the source image. The generator is trained via adversarial loss, which encourages the generator to generate plausible images in the target domain. The generator is also updated via L1 loss measured between the generated image and the expected output image. This additional7 loss encourages the generator model to create plausible translations of the source image. Pix2Pix GANs have demonstrated excellent performance on a range of image-to-image translation tasks such as black and white photographs to colour, sketches of products to product photographs, image style transfer etc.

## Dataset

The dataset has been picked from the official Pix2Pix site and has satellite images of New York and their corresponding Google maps images. The train folder contains 1,097 images, whereas the validation dataset contains 1,099 images. Images have a digit filename and are 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.



This dataset was then prepared for training a Pix2Pix GAN model with Keras. We had to work with the images in the training dataset. Each image should be loaded, rescaled, and split into the satellite and Google map elements. The result will be 1,097 color image pairs with the width and height of 256×256 pixels. load\_images() function was made to implement this. It enumerates the list of images in a given directory, loads each with the target size of 256×512 pixels, splits each image into satellite and map elements and returns an array of each. We then called this function with the path to the training dataset. Once loaded, we saved the prepared arrays to a new file in compressed format for later use.

## Developing & Training the Pix2Pix Model

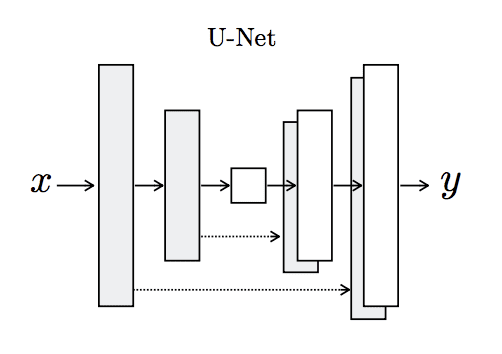
The same model architecture and configuration described in the original Pix2Pix paper was used in our implementation. This architecture was both described in the body of the paper, with additional detail in the appendix of the paper, and a [fully working implementation was provided](https://github.com/phillipi/pix2pix) as open source with the Torch deep learning framework. Our implementation uses the Keras deep learning framework and is designed to take and generate color images with the size 256×256 pixels. The architecture consists of two models: the discriminator and the generator.

The discriminator is a deep convolutional neural network that performs image classification- conditional image classification, in particular. It takes both the source image (e.g. satellite photo) and the target image (e.g. Google maps image) as input and predicts the likelihood of whether the target image is real or a fake translation of the source image.

The design of the discriminator is based on the effective receptive field of the model. The effective receptive field of the model defines the relationship between one output of the model to the number of pixels in the input image. This is also called a PatchGAN model and was carefully designed so that each output prediction of the model maps to a 70×70 square or patch of the input image. The benefit of this approach is that the same model can be applied to input images of different sizes, e.g. larger or smaller than 256×256 pixels.

The output of the model is dependent on the image but may be one value or a square activation map of values. Each value is a probability for the likelihood that a patch in the input image is real. These values can be averaged to give an overall likelihood or classification score if needed.

We’ve made define\_discriminator() function which implements the 70×70 PatchGAN discriminator model as per the design of the model in the original paper on the Pix2Pix architecture. The model takes two input images that are concatenated together and predicts a patch output of predictions. Binary cross entropy is used for optimizing the model and a weighting is used that updates to the model have half (0.5) the usual effect. The authors of Pix2Pix recommend this weighting of model updates to slow down changes to the discriminator, relative to the generator model during training.

The generator model is more complex than the discriminator model. The generator is an encoder-decoder model using a U-Net architecture. The model takes a source image (e.g. satellite photo) and generates a target image (e.g. Google maps image). It does this by first downsampling or encoding the input image down to a bottleneck layer, then upsampling or decoding the bottleneck representation to the size of the output image. The U-Net architecture means that skip-connections are added between the encoding layers and the corresponding decoding layers, forming a U-shape. The figure below makes the skip-connections clear, showing how the first layer of the encoder is connected to the last layer of the decoder, and so on.

#### Architecture of the U-Net Generator Model

The encoder and decoder of the generator consist of standardized blocks of convolutional, [batch normalization](https://machinelearningmastery.com/how-to-accelerate-learning-of-deep-neural-networks-with-batch-normalization/), [dropout](https://machinelearningmastery.com/how-to-reduce-overfitting-with-dropout-regularization-in-keras/), and activation layers. This standardization gives us the ability to develop helper functions to create each block of layers and call it repeatedly to build-up the encoder and decoder parts of the model.

We’ve made define\_generator() function which implements the U-Net encoder-decoder generator model. It uses the define\_encoder\_block() helper function to create blocks of layers for the encoder and the decoder\_block() function to create blocks of layers for the decoder. The tanh activation function is used in the output layer, meaning that pixel values in the generated image will be in the range [-1,1].

The discriminator model is trained directly on real and generated images, whereas the generator model is not.

Instead, the generator model is trained via the discriminator model and is updated to minimize the loss predicted by the discriminator for generated images marked as “real.” As such, it is encouraged to generate more real images. The generator is also updated to minimize the L1 loss or mean absolute error between the generated image and the target image.

The generator is updated via a weighted sum of both the adversarial loss and the L1 loss, where the authors of the Pix2Pix model recommend a weighting of 100 to 1 in favor 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.

This was achieved by defining a new logical model comprising the weights in the existing standalone generator and discriminator model. This logical or composite model involves stacking the generator on top of the discriminator. A source image is provided as input to the generator and to the discriminator, although the output of the generator is connected to the discriminator as the corresponding “target” image. The discriminator then predicts the likelihood that the generator was a real translation of the source image.

The discriminator is updated in a standalone manner, so the weights are reused in this composite model but are marked as not trainable. The composite model is updated with two targets, one indicating that the generated images were real (cross entropy loss), forcing large weight updates in the generator toward generating more realistic images, and the executed real translation of the image, which is compared against the output of the generator model (L1 loss).

We’ve made the define\_gan() function to implement this, taking the already-defined generator and discriminator models as arguments and using the Keras API to connect them together into a composite model. Both loss functions are specified for the two outputs of the model and the weights used for each are specified in the loss\_weights argument to the compile() function. Next, we’d loaded our paired images dataset in compressed NumPy array format.

This returned a list of two NumPy arrays: the first for source images and the second for corresponding target images.

Training the discriminator required batches of real and fake images. The generate\_real\_samples() function prepared a batch of random pairs of images from the training dataset, and the corresponding discriminator label of class=1 to indicate they are real. The generate\_fake\_samples() function was defined next, which uses the generator model and a batch of real source images to generate an equivalent batch of target images for the discriminator. These are returned with the label class-0 to indicate to the discriminator that they are fake.

Typically, GAN models do not converge; instead, an equilibrium is found between the generator and discriminator models. As such, it wasn’t easy to 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 then reviewed the generated images at the end of training and used the image quality to choose a final model.

The summarize\_performance() function was made to implement this, taking the generator model at a point during training and using it to generate a number, in this case three, of translations of randomly selected images in the dataset. The source, generated image, and expected target are then plotted as three rows of images and the plot saved to file. Additionally, the model is saved to an H5 formatted file that makes it easier to load later. Both the image and model filenames include the training iteration number, allowing us to easily tell them apart at the end of training.

Finally, we trained the generator and discriminator models. The train() function implemented this, taking the defined generator, discriminator, composite model, and loaded dataset as input. The number of epochs was set at 100 to keep training times down, although 200 was used in the paper. A batch size of 1 is used as was recommended in the paper. Training involved a fixed number of training iterations. There were 1,097 images in the training dataset. One epoch was one iteration through this number of examples, with a batch size of one means 1,097 training steps. The generator was saved and evaluated every 10 epochs or every 10,970 training steps, and the model was run for 100 epochs, or a total of 109,700 training steps.

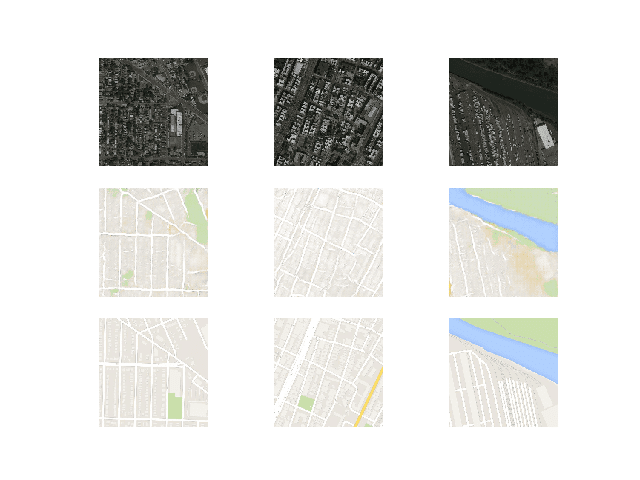
Each training step involved first selecting a batch of real examples, then using the generator to generate a batch of matching fake samples using the real source images. The discriminator was then updated with the batch of real images and then fake images.

Next, the generator model was updated providing the real source images as input and providing class labels of 1 (real) and the real target images as the expected outputs of the model required for calculating loss. The generator had two loss scores as well as the weighted sum score returned from the call to train\_on\_batch(). We were only interested in the weighted sum score (the first value returned) as it was used to update the model weights.

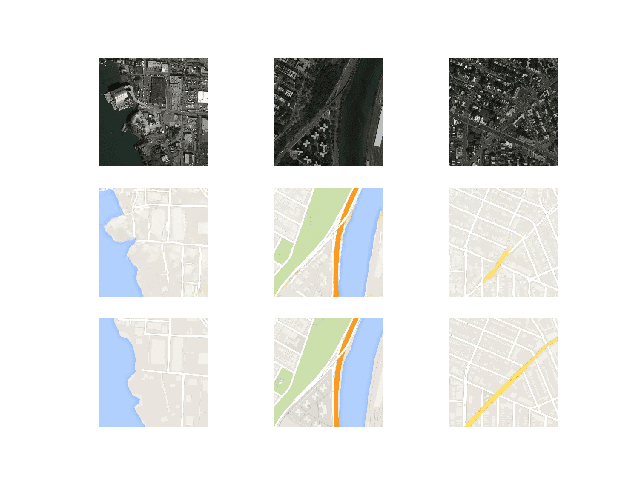
Finally, the loss for each update was reported to the console each training iteration and model performance was evaluated every 10 training epochs. The loss was reported for each training iteration, including the discriminator loss on real examples (d1), discriminator loss on generated or fake examples (d2), and generator loss, which is a weighted average of adversarial and L1 loss (g).

Models were saved every 10 epochs and saved to a file with the training iteration number. Additionally, images were generated every 10 epochs and compared to the expected target images. These plots were assessed at the end of the run and used to select a final generator model based on generated image quality. At the end of the run, we had 10 saved model files and 10 plots of generated images.

After the first 10 epochs, map images were generated that look plausible, although the lines for streets are not entirely straight and images contained some blurring. Nevertheless, large structures were in the right places with mostly the right colors.



Generated images after about 50 training epochs began to look very realistic, at least to mean, and quality appeared to remain good for the remainder of the training process. For instance, the first generated image example below (right column, middle row) included more useful detail than the real Google map image.



Now that the Pix2Pix model has been developed and trained, we could explore how it can be used in a standalone manner.

# CONCLUSION

In this work, we proposed a GAN model for satellite-to-map image conversion. Considering the challenge that some objects might not be easily identified visually from the satellite image, we novelty integrate the external geographic data into a GAN structure to build the conversion. GANs (Generative Adversarial Networks) are the newest set of deep learning models that have a lot of potential in improving image conversion. They fit into the wider framework of unsupervised and semi-supervised learning, which means they can help solve a couple of broader problems in this area. It consists of two networks - the generator and discriminator. The generator tries to generate realistic images from the input, while the discriminator judges how real the generated image is compared to the input image. It's basically a duel between a generator and discriminator trying to fool each other. This project is a combination of many other works on the internet. This method generates images from a latent space representation, which can use image form of data as a seed. The objective is to get a better translation between input images and the output, in our case, satellite images. This estimated semantics help the translation working towards region-alike and hence reduce many pixel-wise noises. This machine learning system is being designed and developed for mapping the conversion of image pixels using a satellite image taken from space. The system would be able to distinguish the different types of ground cover through analyzing the images of the Earth. Humans are not able to distinguish many types of ground cover when isolating or photographing an area; however, it may be possible with the use of AI algorithms to analyze multiple images from space and then pinpoint which types of ground. The proposed GPS integration and semantic-estimation can be easily embedded into various backbone GAN structures.

# FUTURE WORK

Many factors can influence the appearance of a satellite image, including the angle at which the image is taken and the direction that the satellite orbits. With a change in position, an image may appear differently; for example, it can be captured during different times of the day or night which can result in varying colors and brightness. To solve this problem, a computer-vision algorithm is being designed to analyze scatter. The future of Satellite mapping will be a lot more than just images. Future work could also include training on more bands, such as IR to help the map distinguish between visually ambiguous regions such as lakes and vegetation. On the higher end, future of satellite mapping and remote sensing data has a 3D (3 dimension) product where you might have access to 3D representations of your data, precise measurements in your buildings, even historical views from the past. Integrating 3D data with infrastructure and buildings in real time using satellites, integrated 3D products in real time, regardless of distance or weather conditions on the ground, opening unprecedented possibilities for all users are a few areas where potential future work can be done. Meet the innovations that are transforming Earth observation and land management, highlighting the latest examples of how satellite data is changing our planet and influencing strategic decision-making in government and enterprise businesses across the globe. The aim of this can go to providing a high-level overview of the future of satellite mapping technology, revenue models, business applications, and involvement of the commercial space industry.

REFERENCES

* *V. Badrinarayanan, A. Kendall, and R. Cipolla, SegNet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 12, pp.2481\_2495, Dec. 2017.*
* *S. Barratt and R. Sharma, ``A note on the inception score,'' 2018, arXiv:1801.01973.*
* *M. Bierlaire, J. Chen, and J. Newman, ``A probabilistic map matching method for smartphone GPS data,'' Transp. Res. C, Emerg. Technol., vol. 26, pp. 78\_98, Jan. 2013.*
* *C. Chen, C. Lu, Q. Huang, Q. Yang, D. Gunopulos, and L. Guibas, ``City- scale map creation and updating using GPS collections,'' in Proc. KDD, 2016, pp. 1465\_1474.*
* *D. Costea, A. Marcu, M. Leordeanu, and E. Slusanschi, ``Creating roadmaps in aerial images with generative adversarial networks and smoothing-based optimization,'' in Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW), Oct. 2017, pp. 2100\_2109.*
* *X. Zhang, X. Han, C. Li, X. Tang, H. Zhou, and L. Jiao, ``Aerial image road extraction based on an improved generative adversarial network,'' Remote Sens., vol. 11, no. 8, p. 930, 2019.*
* *T. Sun, Z. Di, P. Che, C. Liu, and Y. Wang, ``Leveraging crowdsourced GPS data for road extraction from aerial imagery,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 7509\_7518.*
* *P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, ``Image-to-image translation with conditional adversarial networks,'' in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 1125\_1134.*
* *X. Huang, Y. Yin, S. Lim, G. Wang, B. Hu, J. Varadarajan, S. Zheng, A. Bulusu, and R. Zimmermann, ``Grab-Posisi: An extensive real-life GPS trajectory dataset in southeast Asia,'' in Proc. PredictGIS, 2019, pp. 1\_10.*
* *J. Gu, Y. Zhang, W. Zhang, H. Yu, S. Wang, Y. Wang, and L. Wang, ``Aerial image and map synthesis using generative adversarial networks,'' in Proc. IEEE Int. Geosci. Remote Sens. Symp., Jul. 2019, pp. 9803\_9806.*