# Using Deep Convolutional Generative Adversarial Neural Networks to Predict Precipitation Data Over South East United States

By the Weather Project Team- Ambiorix Cruz Angeles, Satyam Sharma, Airton Prado, Yuan Zhao, and Sami Abood

*Abstract- Using goes-13 satellite images and North-America Mesoscale analysis data, several supervised machine learning models were created, trained, tested, and evaluated, to produce a corresponding radar data image. The satellite data and radar data were collected and prepared for us to use beforehand. The model data had to be requested for via a ticket through the NOAA website.*

*Once all the data was properly acquired, the data management team got to work with preprocessing the data. All three sources of data, satellite, radar, and model, had to be over the same domain of 31N-41N and 82W-102W. In addition, the frequency of the data was chosen to be every hour of the year 2017. The domain was chosen because of its relatively flat geographical shape and lack of any large bodies of water. This was to ensure the images created by the data correlated with each other as close as possible before being given to the model train and test team.*

*The model train and test team were responsible for implementing or using deep convolutional generative adversarial neural networks given the data from the data management team. A linear regression model was selected to be the "control" model for this experiment. The models created, other than linear regression, include random forest, pix2pix, and vid2vid. Once all the models were created, they were passed onto the evaluation team.*

*The evaluation team tested the models predicted radar data versus the actual radar data at that given instances. In the end, the pix2pix implementation using 5 goes-13 satellite images as input and trained with the expected corresponding radar image as the supposed output did the best. On average it was able to predict radar images with an 89% accuracy.*

## Introduction

Originally this team was given the question "Is it possible to predict radar data over the ocean given a satellite image?" but the problem had to be cut into more manageable sizes. This problem is important because current technology can't give us radar data over the ocean. The way radar works is by transmitting a radio signal aimed by an antenna in a direction towards a receiver. The only problem is that it is not possible to set up an array or matrix of receivers floating over the ocean. On land it is easy because the receivers can be manually placed into the ground and monitored via a satellite. Furthermore, radar also works by reflecting the original signal sent. In the case of projecting signals into the ocean, the water itself acts as an absorbent to the signal and hardly reflects any back. Therefore, it is not easy to get radar data over the ocean.

This is where machine learning comes into play. If a model can be trained using all available data over land, who's to say it can't do the same given some data over the ocean? Specifically speaking, using a supervised machine learning model since the expected radar data is given to us. This can

work as the “label” for the satellite data and model data.

To start things off, a baseline model had to be created. The simplest model to tackle this problem is called multi-linear regression. Given that all the data was of the same size MxN, then each individual point in every data source must have some correlation. For example, the top left most pixel from all the satellite data and all the model data must contain some information for the top left most pixel of the radar data. Using this setup, a multi-linear regression model was made using pixel to pixel mapping from several sources as input to the expected radar pixel output.

The most promising model used was pix2pix. Pix2pix takes into consideration the neighboring pixels of the given input and maps them to the one radar pixel output. This would then ensure that more information for a given instance was extracted from the input images and mapped to generate the output images. The downside being it doesn’t take into consideration the next couple of hours or the hours before.

The third model that was implemented is called vid2vid. It’s an extension of pix2pix and it considers the sequential ordering of the images. So, not only does it look at a several windows of a given hour and maps that to generate the radar data pixels, but it looks at the hour before and after as well. This would then lead to a much smoother generated output if the given never-before seen inputs are in sequential order.

The final model created was a random forest model. It’s a rather popular model used for classification, regression, and other tasks. It constructs a multitude of decisions trees at training time and outputs the mean prediction of the data. Like the linear regression model, this model would take in the several corresponding input pixels and map them to the corresponding radar pixel for training. Then, for testing, it would be given a new set of inputs and it would make several “decisions” and give out a corresponding output of what it believes the radar data would have been given the new inputs.

Once all these models were made, the evaluation team would take over. They were responsible for testing the models with never-before seen input images. These new images would then generate the predicted radar images. The predicted radar images would be analyzed and compared to the actual radar image for that specific instance. Several evaluation techniques would be used to get a general understanding of where the model succeeds and fails.

To tackle this problem, several steps were created to keep the entire team on track and to be able to measure our progress. The first step was to find model data that corresponds with the given radar data and satellite data. Secondly, the data management team would ensure that all the data is over the same domain and time frequency. To recap, the domain must be 31N-41N and 82W-102W with the time frequency being every hour of 2017. Thirdly, once all the data is properly aligned, it was then time to create the images for all the data and organize everything into several train and test sets with specific folder and filename formatting. This was specific to what the model train and test team needed. Models like pix2pix and vid2vid require png images in several formats. While multi-linear regression and random forest could just use the actual data values. Fourthly, passing the properly

formatted train and test sets to the model train and test team so they could start training the models. Fifthly, evaluate the predicted data versus the actual expected output and find the most accurate model. Finally, give that model data over the ocean and see what it produces.

## Background

"Generative Adversarial Nets" Part of "Advances in Neural Information Processing Systems 27"
This article is part of a much bigger series of articles that go in depth with neural networks. One possible series of models we could train are generative models. They are composed of a generator and a discriminator that compete against each other to create proper output. The generator tries to capture the data distribution from the input while the discriminator estimates the probability that a sample from the generator came from the training data. If the discriminator says the output came from the sample data, the generator will go back and tweak itself until it passes the discriminators tests.

"Machine Learning for the Detection of Oil Spills in Satellite Radar Images"
This article is like what is being proposed. Human inspectors must be manually trained to distinguish oil spills in satellite images. This can be a tedious task. If a computer could to it, then there would be no need to waste human resources. The images were taken from the RADARSAT and ESRS-1 satellites to train the model. The model, in turn, uses a decision tree to try and classify if the images have an oil spill or not. This model can then be implemented into the Canadian Environmental Hazards Detection System. Relevant data preparation and evaluation measures used in this paper are well documented and could be used to help this research.

"Persiann-CDR: Daily Precipitation Climate Data Record from Multisatellite Observations for Hydrological and Climate Studies"
This article is about a satellite-based precipitation dataset that was constructed for hydrological and climate studies. This dataset was a precipitation estimation from "Remotely Sensed Information" using an Artificial Neural Network. This neural net provided daily 0.25° rainfall estimates for the latitude band 60°S-60°N for the period of January 1st, 1983 to December 31, 2012. The Persiann-CDR is generated using the Persiann algorithm using the GridSat-B1 infrared data.

"Application of Satellite Remote Sensing Imagery to Bridge Scour Evaluation"
This article is about evaluating the applicability of the HEC-18 method to bridges in Louisiana that were built on top of cohesive soil. It states that, in pier research, sandy soil is known to erode particle by particle, while cohesive soils usually erode in clumps rather than individual particles. Since the cohesive soil bonding and eroding, is a bit more complex, figuring out the actual depth using a scour depth prediction method is more difficult. An analysis of 7 bridges over various soil types over a 10-15-year period were done to determine the hydraulic properties using satellite remote sensing data. This data was then used as an input to the HEC-18. The output was then compared to real flood data.

"An Intercomparison and Validation of High-Resolution Satellite Precipitation Estimates with 3-Hourly Gauge Data"
This article is about measuring the error

when it comes to using merged geo-stationary infrared data and polar orbiting passive microwave satellite data. This led to the Program to Evaluate High Resolution Precipitation Products (PEHRPP) to evaluate and compare datasets at different spatial and temporal resolutions to help developers who plan to use different datasets in their projects. This article will come in handy because the several datasets will be used throughout this project.

"Image De-Raining Using a Conditional Generative Adversarial Network"
This article is about how severe weather conditions such as rain and snow affect the visual quality of images that were captured under those conditions. These conditions can drastically reduce the performance of visual systems. Hence, the article states that, it is important to solve this problem of de-raining/de-snowing images to achieve a better result. This criterion can be incorporated into many projects that rely on accurate inputs to achieve accurate outputs. The article goes into specifics about detection and classification algorithms and how this method can help reduce some of the artifacts.

"Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting"
The article says that the goal of precipitation nowcasting is to be able to predict the future rainfall intensity in a local region over a brief period. A LTSM network was used as the base for this project and convolutional layers were added to the input-to-state layers and the state-to-state transitions. In the article, the idea was to use this new network to generate a trainable end-to-end model for the precipitation nowcasting problem. This new ConvLTSM was compared to FC-LTSM and it was proven to consistently outperform it.

"Synthetic Aperture Radar Image Synthesis by Using Generative Adversarial Nets"
This article talks about the accuracy of using Synthetic Aperture Radar images in SAR applications for automatic target recognition and image interpretation. A lot of the geometric error of these images come from the simplification of the electromagnetic calculation. This article also talks about how an end-to-end model was developed to directly synthesize images from known databases. The model was based off a generative adversarial network and the results were validated using real images and ray-tracing results. On top of that, the samples were synthesized at angles outside of the data set. This led to an improvement in the speed of convergence for the training of some of the models.

## Methods

The workload was broken up into three segments that would work on a different part of the pipeline for this project. The Data Management Team would be led by me, Ambiorix Cruz Angeles. The Model Train and Test Team are Sami Abed and Yuan Zhao. The Model Evaluation Team consist of Satyam Sharma and Airton Prado.

The Data Management Team is responsible for making sure all three sources of data, satellite, radar, and model, are over the same domain and time frequency. The data is then processed; converted into images or arrays and organized in whatever way the Model Train and Test Team needs it to be before passing it on.

**The Radar Data**

The radar data was given to use by Professor Brian Vant-Hull via a dropbox containing fodlers with ".dat" files in them. A readme and a small script to read in the data was also present. The domain for this data was 31N-41N and 82W-102W. The frequency was every hour from 2008 to 2017. There was a total of 70,000 files but since the focus was primarily on 2017, only 7856 hours were used.

The name of the files contained the month, day, and hour information of the file but the names were re-written to be "rad.yyyymmddhh.dat" for format consistency with the other data sources. Using python and the numpy python library, the following code could be used to read in the data as a numpy array.

```
import numpy as np
dims = (126,201)
data = np.fromfile(path+filename,
dtype='int16', count=-1, sep='')
data = np.reshape(data,dims)
data[data > 30] = 30
```

The little bit of code above summarizes the extent of how to read in the ".dat" radar files. Using numpy and the data dimensions given to us by Brian, it was possible to reshape the one-dimensional array of 24,321 elements into a two-dimensional array of 126 rows and 201 columns. Once the data was in the correct shape, modifying it was easy. The value at each point represented the amount of rainfall in mm/hr that was accumulated over the last hour. Brian commented and said that any data value over 30 mm/hr should be railed off as 30 mm/hr. This would make the radar data easier to play around with and would give the "png" images a max value that can be tracked. The 30 mm/hr would be mapped to a png grayscale image pixel value of 255.

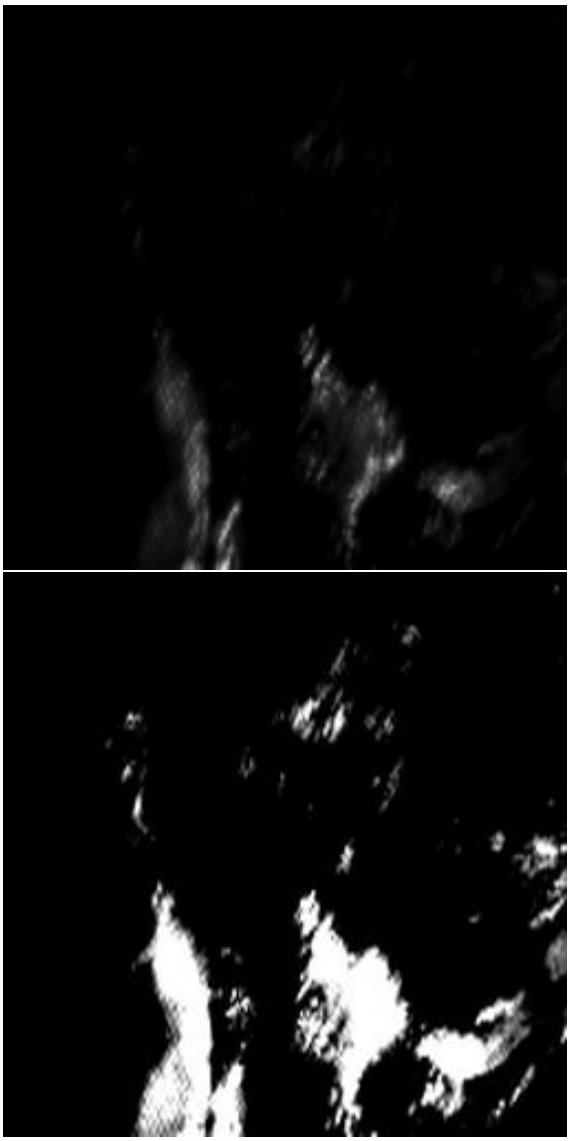

Figure 1 – The image at the top is a non-threshold radar image. The image at the bottom is at threshold 30 mm/hr.

**The Satellite Data**

The satellite data was given to us by Paul. A scrip to display the images was also given. Since the data came in as a ".nc" file extension, the netCDF4 python library had to be used. This data came in two batches. The first batch had the required domain

needed but the frequency was every 30 minutes starting at 00:15 and 00:45 of some hours of September, October, and November of 2017. This was unacceptable. Only about 400 actual hours per satellite band was available. The second batch given to us to use had the same frequency of every 30 minutes starting at 00:15 and 00:45 of every hour of 2017. Unfortunately, there were only 4200 instances per band. If which only 1800 were usable since 00:15 was used to match the hours of radar. Furthermore, the domain was increased to 28N-45N and 78W-112W. Some preprocessing was needed before creating the images.

**GOES Imager band nominal wavelengths (GOES-12 through 15)**

| GOES Imager Band | Name | Central Wavelength (µm) | Objective |
|---|---|---|---|
| 1 | Visible | 0.63 | Cloud cover and surface features during the day, smoke, etc. |
| 2 | Shortwave window | 3.9 | Low cloud/fog, fire detection, winds, etc. |
| 3 | Water vapor | 6.48 | Upper-level water vapor, winds, etc. |
| 4 | Longwave window | 10.7 | Surface or cloud-top temperature, precipitation, etc. |
| 5 | N/A | N/A | N/A |
| 6 | $CO_2$ band | 13.3 | $CO_2$ band: Cloud detection, etc. |

Figure 2 – the goes 13 satellite band names and overall objective.

All 5 available satellite bands take a picture of the same area but represent very different data. For example, band 1 is a high-resolution camera. During night times band 1 images would be entirely gray or black because there is no light coming from the sun for the camera to capture.

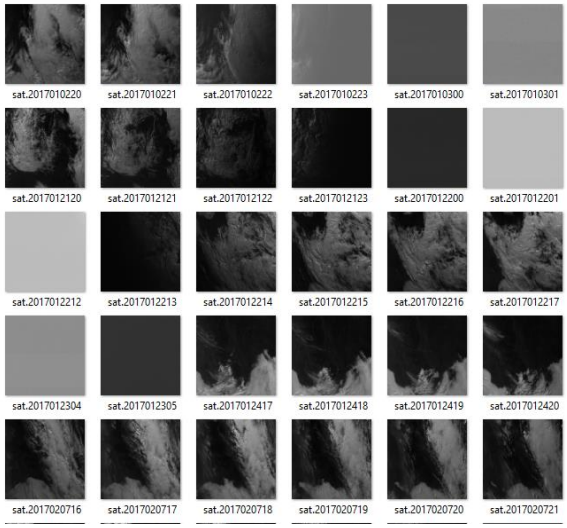


Figure 3 – Showcasing satellite band 1 only showing gray or black images during night time.

This fact about band 1 was kept in mind when creating datasets for the other teammates. Since the bottle neck for the correlating data is already so low, band 1 would be excluded for many of the datasets.

**The Model Data**

The model data was taken from the North American Mesoscale dataset from the NOAA website. One of the only datasets that had hourly files for 2017. This dataset required the use of the pygrib python library since the data came in as ".grb" and ".grb2" file extension. A ticket had to be put into the website and several days later the data would be ready for download. This process took about a week before an ftp link was emailed with the data. The domain for this source was 12N-51N and 50W-155W, the entire North American Region. This was no big deal since the pygrib library would allow cropping of the data. The frequency was good though, every hour of 2017 was just what was required. The total number of files turned out to be about 7,180 instances.

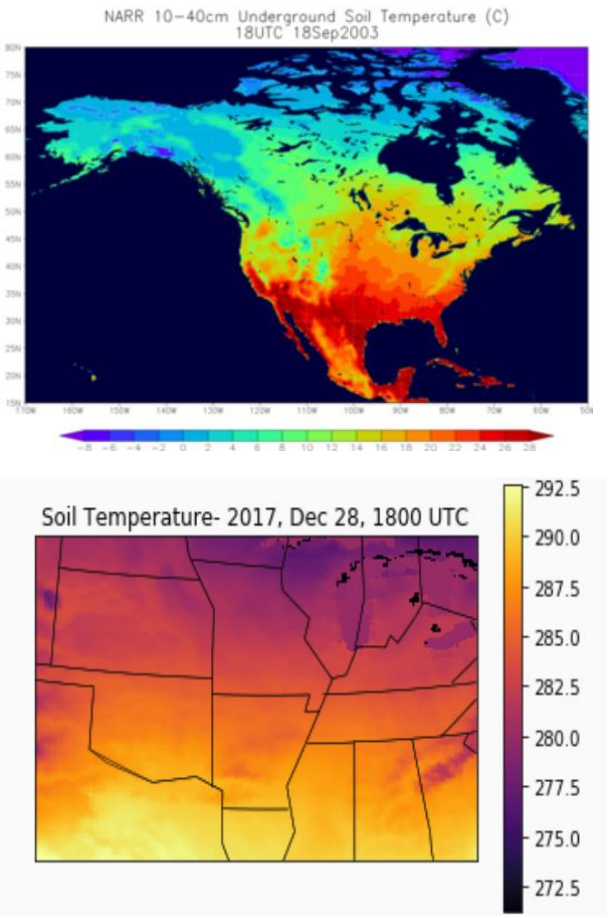


Figure 4 – Top image shows the entire domain of the model data while the bottom image shows the cropped-out version with basemap state lines for domain verification.

**Table 1:** Overview of the amount of raw data used for this project.

| | Ideal | Rad-ar | Satellite 2nd | Model |
|---|---|---|---|---|
| Domain | 41N 31N 102W 82W | YES | Crop Able Using Base-map | Crop Able Using Base-map |
| Freq-uency | Hourly | YES | 00:15 as hourly | YES |
| Total Hours 2017 | 8760 | 7856 | 1812 per band | 7180 |

Looking at table 1, it is apparent that the satellite data we received is bottle necking every other dataset. The models that will be used require the input and output to correspond. Since satellite only contains 1812 hours for 2017 per band, then the most images that could be used for training is that amount. This is incredibly low for training any real models, but the project needed to be done.

**Datasets**

Several datasets were created depending on which teammate was creating what model at the time. Some models required png images of specific dimensions with specific names, others required sequential png images with a different name, and some required the actual numbers not png images.

All these datasets were rsync'd onto a node computer in the colleges basement. This node contained a very powerful computer with gpu's available for model training.

**The pix2pix Dataset for Sami**

Sami's implementation of pix2pix went through several stages. Each stage was given a different dataset. There were 4 main dataset changes for Sami. The first dataset only had satellite band 2, satellite band 6, model data temperature, model data visibility, model data specific humidity, and model data relative humidity. At the time, the model data was only available 4 times a day since it was still being downloaded. This set only contained 195 images. The second dataset had all hours of the day available.

Bumping the number of images to about 1620 from 195. A much bigger improvement. Then, the first important trained model called “sparseless” was created by taking about any radar image that was 80% black or more. This change downgraded the 1620 images to 820 images. In turn, it made the model generate more accurate radar images since it was not trained with completely black images. Furthermore, sparseless2 was created. This model was trained with the same dataset except radar images that were 95% black were removed. Increased the dataset from 820 to 1420. This dataset was further improved by rotating each image 3 extra times, 90 degrees counterclockwise generated more instances. Finalizing the total number of correlating images to be 5680 instances.

### The vid2vid Dataset for Yuan

Since there was a lot of trouble getting vid2vid to run properly, a multiple image input to 1 expected image output was not feasible. In this case, only the corresponding satellite band 2 images that matched the radar data set were used to create this dataset. Vid2vid required 4 folders: train_A, train_B, test_A, and test_B. Each folder should contain sequential subfolders: “000, 001, 002, etc.” These sequential folders should contain hourly images with sequences longer than 30 back-to-back hours. The dimensions for these images were blown up to 2080x1040 and a script was given to Yuan to change the dimensions of the images to whatever she wanted.

### The Random Forest and Feature Importance Dataset for Satyam

This is the most intricate dataset of all the others. It contains corresponding images of all available sources: all satellite bands, radar, and 4 model data features. This dataset also came in two versions; png and npy. Originally it was given to Satyam as png images, but it was realized that his method did not require images. Thus, the npy version of all the data without the png limitations was created to get closer to the data.

## Results and Evaluation

### Linear Regression

Using the same dataset as pix2pix, a baseline was created using linear regression. This baseline, untouched, is no better than flipping a coin to generate the radar data pixel by pixel. If the output image of the linear regression is threshold at about 15, the accuracy shoots up to about 82%, better than flipping a coin.

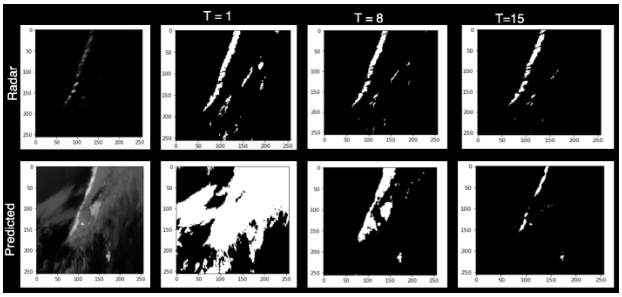


Figure 5 – the output of the linear regression model threshold at various values and compared to the expected radar image.

### Random Forest

Using the dataset given to Satyam, that included every input resource mapped to the radar data, a random forest model was created. This model had an accuracy of about 76% when the contrast was adjusted.

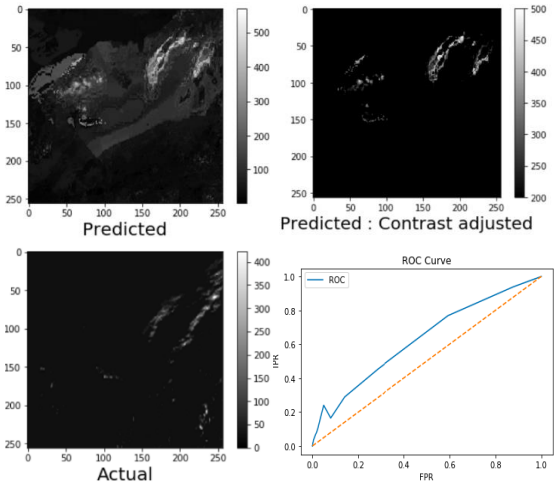


Figure 6 – The top left image contains a predicted output of the random forest model. The top right image is the contrast adjusted version of the predicted image. The bottom left image is the actual radar image. The bottom right image is the ROC curve of the random forest model using the contrast adjusted images.

**Pix2Pix**

Looking at two variations of the sparseless model yields different accuracies. One version of the sparseless model was trained using model data. This model had an accuracy of 92.9%. It is believed that the model data contains radar information when it is generated. A deeper look into the North American Mesoscale by Brian confirms this. He says that no matter where you get the model data, it will contain some form of radar data in it. However, if model data that is close enough to the surface is used to train the pix2pix model, it is acceptable. Regardless, another pix2pix model was trained only using satellite data and the accuracy only went down to 89.1%.

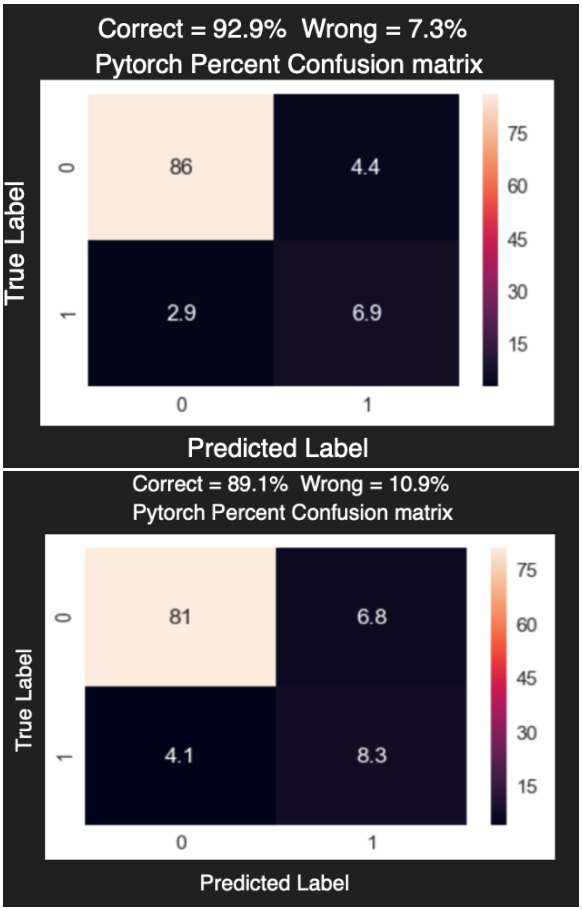


Figure 7 – the confusion matrix at the top is from the pix2pix model trained using model data. The confusion matrix at the bottom is from the pix2pix model trained with only satellite images.

**Vid2Vid**

After a fierce semester lengthy battle to get vid2vid to work on the node in the colleges basement, results came in. There was an issue with the gpu compatibility when running vid2vid. Vid2vid requires at least 8 gpu's to properly train a model with high resolution images. 4 gpu's are used for the generator and the 4 others are used for the discriminator. 4 images at a time are loaded up individually into the gpu's for fast computation. This is how it differentiates from pix2pix. It takes into account multiple hours before and after the current instance as

well as several convolutions over the images to retain the spaciel information from input to output. Since time ran out, the vid2vid model was only trained on 180 or so images. So, the results are not good but the visualization of predicted output to the next predicted output is way smoother than pix2pix. Leading us to believe that vid2vid has the potential to be better if given more data and computing power.

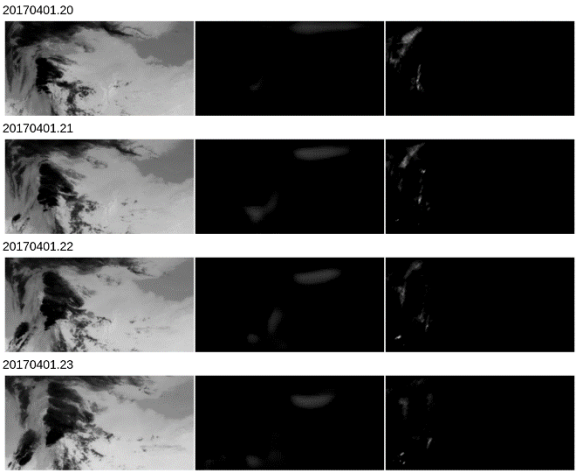


Figure 8 – the satellite band 2 image is at the far left, the predicted is in the middle, and the actual is on the right-hand side.

The results are nowhere near as good as pix2pix.

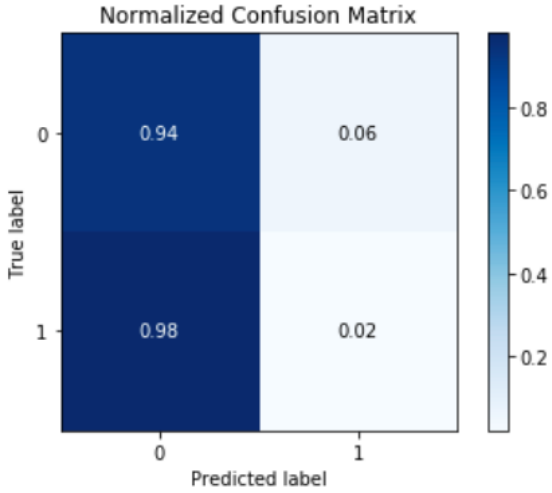


Figure 9 – The confusion matrix for the 180-image trained vid2vid model. Only a 2% accuracy.

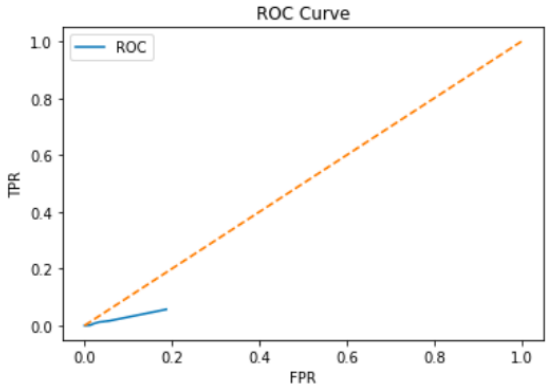


Figure 10 – The ROC Curve for the trained vid2vid model also shows how bad it is when trained with such a small amount of images.

## Conclusion

With the current data we have, pix2pix generated the best results. This is since it takes into consideration the spatial information in the input images and maps them to the output image. It is possible that vid2vid could yield better results, but given the time constraint, lack of budget, lack of computing power, and lack of data, a proper vid2vid run was not possible. In regard to the original problem, gathering more data over the ocean to test the trained pix2pix model would take time. So, it was not possible to test the best trained pix2pix model with ocean satellite data to see if it could generate radar data.