Predicting Precipitation from Remotely Sensed Images and Weather Models using Machine Learning

Satyam Sharma
The City College of New York (CUNY), ssharma000@citymail.cuny.edu

Source code
GitHub: https://github.com/satyamsharma/capstone-weather

Includes contributions from:
Ambiorix Cruz Angeles, Yuan Zhao, Airton Prado, Sami Abood
cruzangeles.ambiorix@gmail.com, yzhou000@citymail.cuny.edu, airton.k.prado@gmail.com, abdlwahdsa@gmail.com

Abstract

In this paper, we discuss and evaluate different approaches to predicting precipitation from Remotely Sensed Images and Weather Models as input features. We use five-satellite bands, namely, *Visible*, *Shortwave window*, *Water Vapor*, *Longwave window*, and *CO2* information from GOES-13 satellite, and Weather model data, namely, *Temperature*, *Visibility*, *specific humidity*, and *relative humidity* from North-America Mesoscale (NAM). Radar images are used as the 'labeled' data for training the features in the supervised learning models. We use Multiple Linear Regression, Random Forest Regression, Pix2Pix, and Vid2Vid.

The data is collected from different sources then cropped to fit between 31N-41N longitude and 82W-102W latitude at overlapping hours of the year 2017 for proper spatial and temporal alignment. The data is then split into train and test sets. The train set is then used to create Machine Learning models by feeding them into the aforementioned algorithms. In the case of Linear Regression and Random Forest Regression, feature importance and coefficients are also evaluated to understand which features have most prominence in detecting precipitation in the models. As an attempt for building more robust regression models, input windows of different dimensions are also used to take into account of the neighboring pixels.

In a separate implementation, the created models are then loaded to predict and evaluate the corresponding Radar of the test set. Evaluation is then performed using Root Mean Square Error (RMSE), Distribution of Error on histograms, Confusion Matrix, and Receiver operating characteristic (ROC) curve. Using our evaluation, it is found that Pix2Pix performed the best. This is because of its more advance GAN architecture that does a better job of accounting for the spatial information in the data.

Keywords – Predicting Precipitation, Radar, Rainfall, GOES-13, North-America Mesoscale, Satellite, Multiple Linear Regression, Random Forest Regression, Deep Learning, Pix2Pix, Vid2Vid, Generative Adversarial Network, Supervised Learning, Machine Learning

1. INTRODUCTION

Precipitation is a key indicator of the hydrometeorological cycle and climate changes of a region. Information of precipitation is critical to performing research and forecasting of earth's atmosphere allowing monitoring of global warming and preparing for natural hazards. Moreover, Researchers greatly rely on radar and rain-gauges for accurate and up-to-date meteorological data. However, in many regions of the world, such ground-based measurement networks are sparse and insufficient for accurately capturing the spatial and temporal dimensions of precipitation data. Therefore, data from both the orbiting Satellites and the ground-based measurement networks are used together in an attempt to get an unabridged version of the meteorological data [2].

Evidently, the current algorithms used in satellite-based multispectral analysis require data from radar and rain-gauges as a reference to avoid biases. This makes it a challenge to estimate meteorological data over the open ocean regions where ground-based measurement networks are non-existent [3]. This is because it is difficult to position a network of rain gauges over the ocean. Moreover, the signals projected by the radar are mostly absorbed by the ocean's water and as a result the radar receives insufficient reflectivity information.

Developments in improving variants of PERSIANN, CMORPH, and TMPA for self-sufficient satellite-products still sustain considerable bias for it to be used in real applications [4]. Therefore, these challenges create the research opportunity for enhancing the satellite-products to better estimate the meteorological data in the absence of other sources.

Our project utilizes and evaluates several supervised learning machine learning techniques for predicting radar images using information from Satellite and Weather models. Evaluation and comparison metrics are the pre-requisites for a comprehensive and progressive research. We perform evaluation using Root Mean Square Error (RMSE), Distribution of Error on histograms, Confusion Matrix, and Receiver operating characteristic (ROC) curve.

Our contributions are novel in many approaches. The domain of the dataset is chosen because of its relatively flat geographical shape and lack of any large bodies of water. This ensures the quality of the radar data. We account for the information provided by the Weather models (NAM) and utilize it in building our prediction models. We used some of the state-of-the art Convolutional neural networks: Generative adversarial networks, namely, Vid2Vid and Pix2Pix. For evaluation we use Confusion Matrix and ROC curve by converting the regression problem into a classification problem while still keeping the traditional approaches: RMSE and Error distribution over histogram.

2. BACKGROUND

One of the works that most closely relates to our project is by Kubat, Miroslav, et.al " Machine Learning for the Detection of Oil Spills in Satellite Radar Images." It tackles oil spill detection and addresses problem formulation and data preparation. The images used for training were RADARSAT and ESRS-1 images which were trained to build a classifier using decision trees to detect oil spill [1]. This has motivated us to use Random Forest regression for our project.

Behrangi, Ali, et al. develops a multidimensional rain estimation technique called Precipitation

Estimation from Remotely Sensed Information using Artificial Neural Networks–Multispectral Analysis (PERSIANN-MSA) to estimate rain rate using multispectral data plus textural features [3]. Behrangi, Ali, et al. discusses the importance of feature extraction and use of PCA to reduce feature space in Neural Networks. Behrangi, Ali, et al. classifies comparisons in Brightness temperature of clouds, Daytime hours, and Night time hours, etc. It uses Probability of Detection(POD), False alarm ratio(FAR), Bias Error, and Equitable threat score for evaluation.

Conti, Francesco Lo, et al. does a comprehensive evaluation of four satellite products (CMORPH, PERSIANN, PERSIANN-CCS, and TMPA-RT) and two GPCP-adjusted products (TMPA and PERSIANN Adjusted). Conti, Francesco Lo, et al. discusses the use of passive microwave (PMW) data from polar-orbiting satellites (Low Earth Orbiting satellites) and the infrared (IR) data from geostationary satellites (Geosynchronous Earth Orbiting satellites) to estimate the precipitation [4]. Conti, Francesco Lo, et al. uses evaluation indices Mean Bias Error (MBE), Root Mean Square Error (RMSE), Coefficient of Variation of the RMSE (CV-RMSE), Correlation Coefficient (CC), Probability of Detection(POD), False alarm ratio(FAR), and ROC diagram. Conti, Francesco Lo, et al. reports bias in all six of the satellite products. We, however, think that simple RMSE is sufficient at this stages in the development.

To evaluate performance of Neural networks in regards to our problem, we utilized two GAN variants in our pipeline, namely, Pix2Pix [6] and Vid2Vid [5]. GAN is a relatively newer category in deep learning domain. However, due to its functioning in diverse use cases has led to its popularity.

The Pix2Pix utilizes a conditional generative adversarial network. Like other GAN, there is a generator that generates the output image, while the discriminator critiques the generated output image.

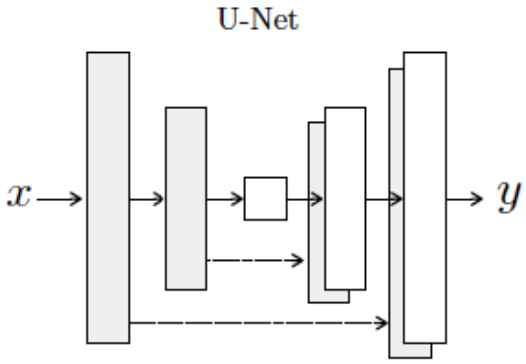


Figure 1: Encoder-Decoder like architecture called U-Net

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

Equation 1: Generator's optimizing equation

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

Equation 2: Term accounts for high frequency correctness

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

Equation 3: Term accounts for low frequency correctness

In Pix2Pix, the generator is based on an auto-encoder and decoder architecture called as U-Net [Fig. 1]. The first component of the Generator's formula accounts for the high frequency (edges) correctness [Equ. 1, 2] whereas the second term accounts for low frequency correctness [Equ. 1, 3].

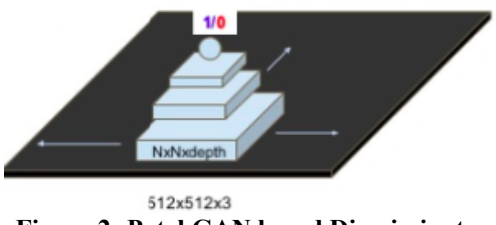


Figure 2: PatchGAN based Discriminator

The discriminator is based on a PatchGAN architecture [Fig. 2]. Basically, it is a three-layered

CNN classifier, like a pyramid, that slides over the output image generated from the Generator and then classifies if the current window is "real" or "fake." After running across the whole image, averaging all responses, it classifier the whole output as “real or fake”.

We realized that we could potentially make the GAN perform better if we could train it to account for the previous and the next frames of satellite and radar data. This as a result will provide smooth flow and generate a more realistic precipitation data. Since our data is a time sequence, we used Vid2Vid for this purpose.

Vid2Vid is a video synthesis model that aims to produce realistic videos without having to explicitly specifying scene geometry, materials, lighting, and dynamics [5].

3. METHODS

Note: The following section only comprises of the methods that were eventually included in the final presentation and the result section of this paper. Please refer to source code for all the results.

Due to the almost black-box like implementations of Pix2Pix and Vid2Vid, we trained the models using 8-bit “.png” formatted images. Therefore, there were 256 intensity levels for each pixel.

Multiple Linear Regression and Random Forest Regression, as prescribed by the main author, however, were switched to work with the raw data of type 32-bit float.

The Radar data was provided in “.dat” files. The Satellite data was provided in “.nc” files. And, the Model data was provided in “.grb” files. Since, most of the machine learning libraries we used were meant for *numpy* or *list* types, and because the conversion of the datatypes to numpy took extra time each time it was imported to jupyter notebook, we decided to remove the overhead time by converting and storing all the relevant data to “.npy” thus making the loading as a one step process.

Like Behrangi, Ali, et al. classified his dataset by the variation in intensities in the satellite imageries due to the varying exposure of the sun, we extracted the data only between 16:00 and 20:00 UTC (local time 11:00 and 15:00 EST) during sufficient sun’s exposure. This filtered 349 hours of data. (So, for each hour, we had information on the 5 satellite bands, 4 model data features, and their corresponding radar data).

Both, the Random Forest and Linear Regression, are pixels to pixel mapping. Therefore, for a given set of 256x256 images for the 9 types of input data and 1 label data resulted in 65536 rows of data. Here, each row data has 9 feature columns and a corresponding radar pixel in label-column vector. Each element contains a 32-bit type float.

```
# Test Train Split for Files
import random
import math
SEED = 42
random.seed(SEED)
random.shuffle(Sal_files_all)

Sal_files_all.sort()

size_Sal = len(Sal_files_all)

Sal_files_train_size = math.ceil(size_Sal*.75)
Sal_files_test_size = size_Sal - Sal_files_train_size

Sal_files_train = Sal_files_all[:Sal_files_train_size]
Sal_files_test = Sal_files_all[Sal_files_train_size:]
```

Figure 3: Test Train Split for files

For Test train split, we split the data at the file level itself [Fig. 3] before extracting the pixel information resulting in 262 image set for training and 87 image set. Note: 1 image set constitutes of 9 input images and 1 corresponding radar image.

In an attempt to build more robust regression models, window method with different dimensions were used: 3x3, 5x5.

In order to position the windows on top of the images without falling off the edges, 1 strip of row/column was discarded at four sides for the images that was used with 3x3 window, and 2 strips of row/column were discarded at the four sides for the images that was used with 5x5 window.

This resulted in the following configurations in the inputs for the windowed regression models.

For 3x3 window:
256x256 image was transformed to 254x254 image leading to 64516 rows of data.
Where each row has 81 columns for features and 1 column for label.

For 5x5 window
256x256 image was transformed to 252x252 image leading to 63,504 rows of data.
Where each row has 225 columns for features and 1 column for label.

Once the data is prepared, it is fed into the model to fit. We used feature importance in random forest to understand which features (bands and model) are important. We also evaluated the coefficients for the multiple linear regression to understand which features attain the largest magnitude for the coefficients. We understand that the lager the magnitude for the coefficients, the more important the model deems it to be.

After fitting the model, we immediately test it by making it predict the training set images. Then, we calculate the maximum error, histogram of the error, confusion matrix, and ROC curve.

In order to plot the Confusion Matrix, we would have to turn the regression problem into a classification problem. This was done by thresholding the values in the generated output and the actual output images. The choice of threshold value was decided in the following manner: 1) Make a copy of all the non-zero pixels from all the radar images. 2) Find the mean of the new list. 3) Find the value of the standard deviation of the list. The threshold value is the mean + the standard deviation.

Note: earlier the threshold was calculated by adding 2 standard deviation above the mean, but Brian suggested to use 1 standard deviation, instead.

```
y_train_no_zero = list(filter(lambda a: a != 0, y_train))
print(np.shape(y_train_no_zero))
print(np.std(y_train_no_zero))
print(np.min(y_train_no_zero))
print(np.mean(y_train_no_zero))
print(np.max(y_train_no_zero))
```

```
(16973468,)
21.287491448825545
1
20.856037375508645
953
```

```
thershold = np.mean(y_train_no_zero) + 1 * np.std(y_train_no_zero)
thershold
```

```
42.14352882433419
```

Figure 4: Calculating a threshold value

Plotting the ROC curve was tricky.

```
from sklearn.metrics import confusion_matrix
FPR = []
TPR = []

for i in range(1, np.max(real_flattened), 50):
    y_train_copy = np.copy(real_flattened)
    y_train_copy[y_train_copy < i] = 0
    y_train_copy[y_train_copy >= i] = 1

    predictions_train_copy = np.copy(pred_flattened)
    predictions_train_copy[predictions_train_copy < i] = 0
    predictions_train_copy[predictions_train_copy >= i] = 1

    temp = confusion_matrix(y_train_copy, predictions_train_copy)

    TN = temp[0][0]
    FP = temp[0][1]
    FN = temp[1][0]
    TP = temp[1][1]

    TPR_ = TP/(TP+FN)
    FPR_ = FP/(TN+FP)

    FPR.append(FPR_)
    TPR.append(TPR_)
```

```
plt.plot(FPR, TPR, label='ROC')
plt.plot([0,1], [0,1], linestyle='dashed')
plt.title("ROC Curve")
plt.xlabel("FPR")
plt.ylabel("TPR");
plt.legend()
```

Figure 5: Generating ROC curve

The challenge was to find the values False Positive rate and True Positive rate. This was

overcome by realizing that after thresholding when we plot the confusion matrix, it actually gives us the following numbers in its quadrants: True Positive, False Positive, False Negative, and True Positive.

predicted→ real↓	Class_pos	Class_neg
Class_pos	TP	FN
Class_neg	FP	TN

$$\text{TPR (sensitivity)} = \frac{TP}{TP+FN}$$

$$\text{FPR (1-specificity)} = \frac{FP}{TN+FP}$$

Figure 6: Relationship between Confusion matrix and ROC curve

And we could calculate the False Positive rate and True Positive rate using the formula as stated in the diagram.

However, this will only provide us with 1 pair of False Positive rate and True Positive rate. To overcome this, we decided to calculate the confusion matrix across a range of threshold values and then map the True positive rate with its corresponding False Positive rate. However, we think a better and more graceful is also possible for evaluating the ROC curve.

Before we conclude the evaluation of the training set, we save the actual model on the disk for future evaluations (for the test set).

Now, to conduct evaluation of the test image, a separate notebook is created. The saved model is loaded in the notebook. This time the *.npy* files of the test set is loaded. The rows of data are created containing pixel information. The rows are then fed to the loaded model for prediction. Then, the predicted output pixel is compared to the actual pixel in the label column.

For evaluation, we use the same techniques: RMSE, distribution of error, Confusion Matrix, and ROC curve. However, this time we reconstruct the all the pixels back to the 2D images. For visualization purposes, we then separately reconstruct the 2D images with least RMSE and largest RMSE.
We also did evaluation for the GAN models using the similar techniques. For evaluation of Pix2Pix and Vid2Vid, we compared the corresponding pixels of the predicted and actual frames. Then, we calculated RMSE, distribution of error, Confusion Matrix, and ROC curve. And, displayed the frame that had the least and largest RMSE.

4. RESULTS

Below are the diagrams of confusion matrix, and ROC curve of Linear Regression, Random Forest, Pix2Pix, and Vid2Vid. We have also displayed actual and predicted images.

For Linear Regression and Random Forest, we have arbitrarily chosen: **radar.2017082219.npy**
to show the predicted and actual image.
For Pix2Pix and Vid2Vid we display the image pairs with least RMSE and largest RMSE.
Note: In regards to Confusion matrix
Class 0: No Precipitation
Class 1: Precipitation

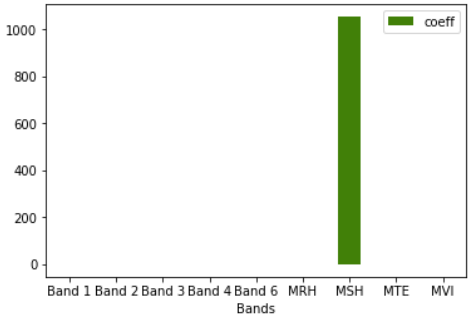


Figure 7: Absolute value of the coefficients in Linear Regression

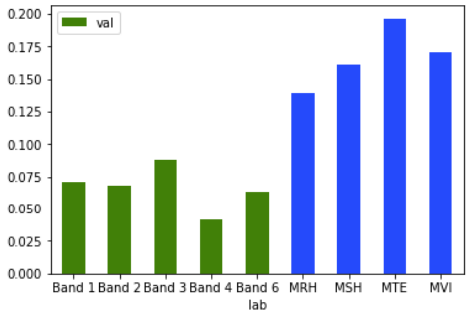
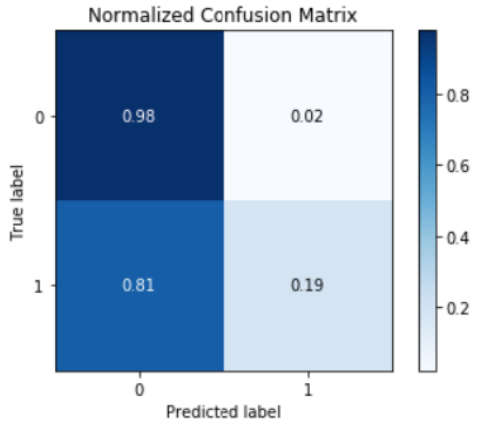
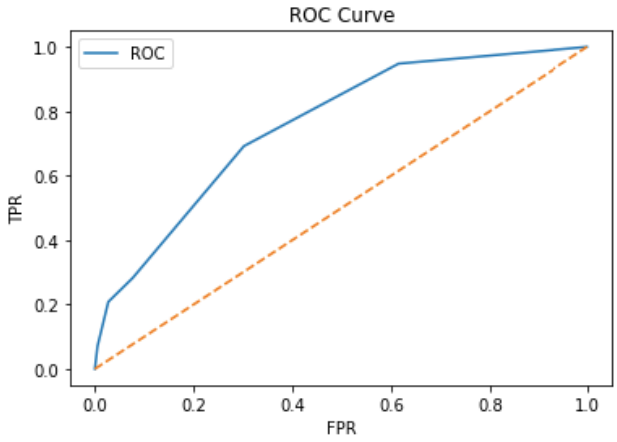


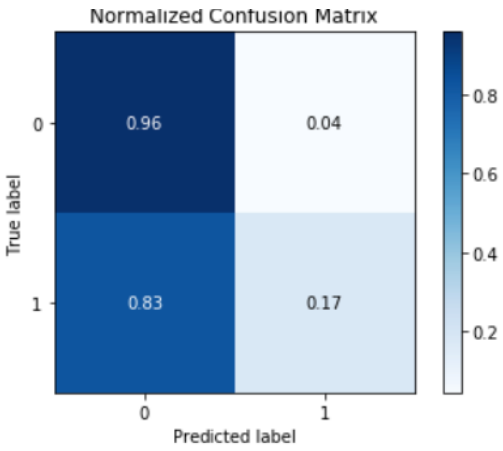
Figure 8: Feature Importance using Random Forest

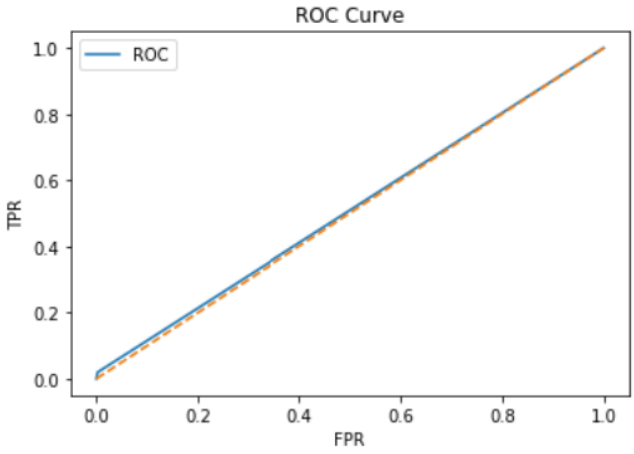
Linear Regression 1x1 Window (Training):





Linear Regression 1x1 Window (Test):

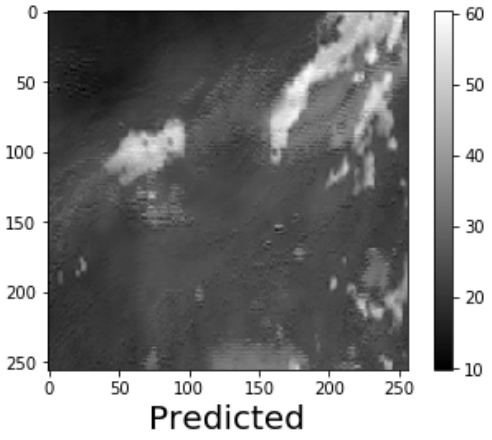


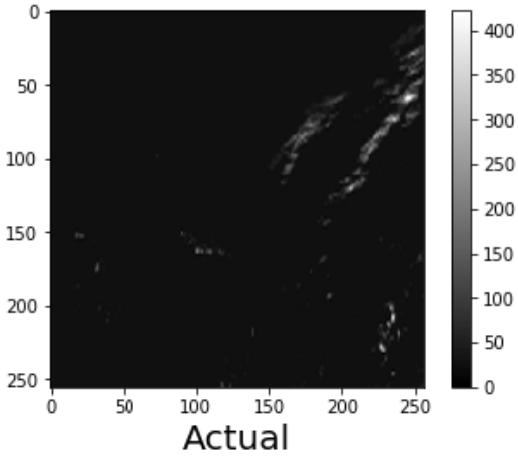


Test Image

Linear Regression

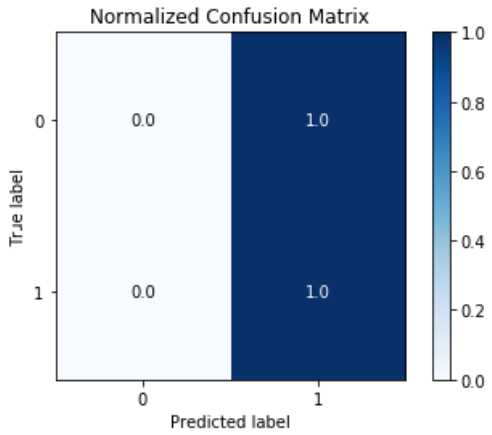
RMSE: 16.588344982683523

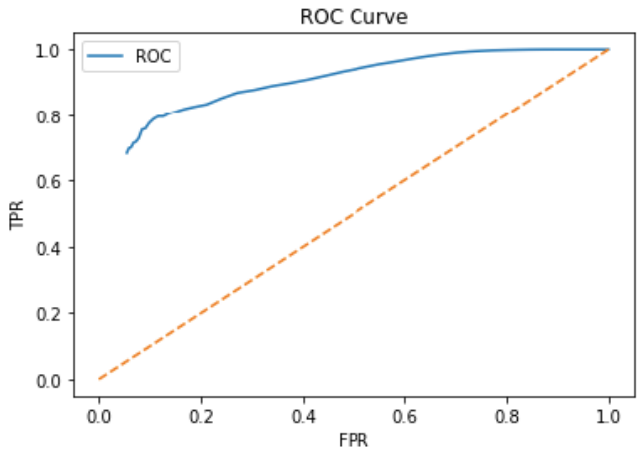




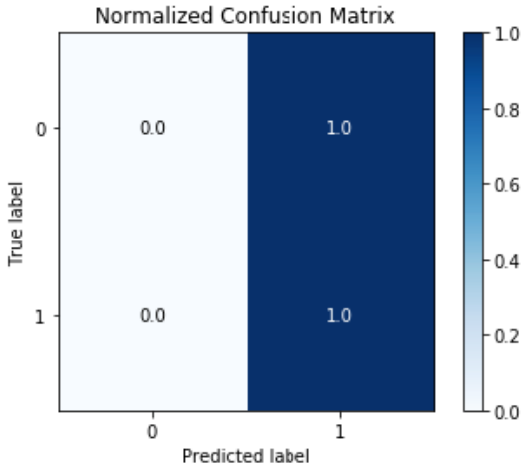
<u>Note the difference in colormap. There is a Scaling issue</u>

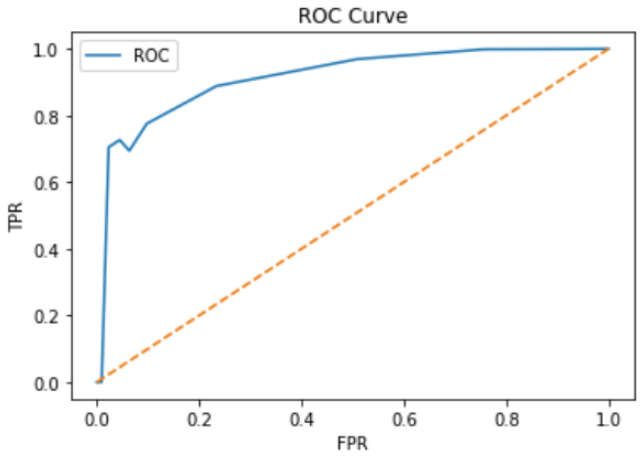
Scaling the predicted Test and Train
Linear Regression 1x1 Window (Training):



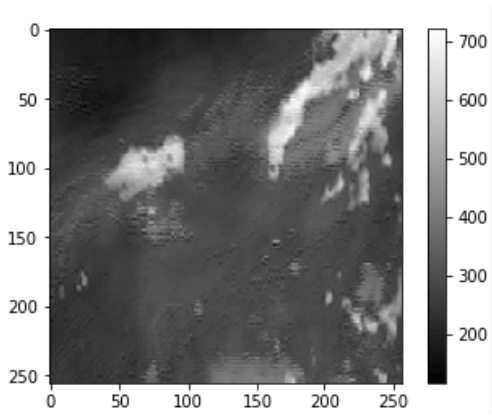


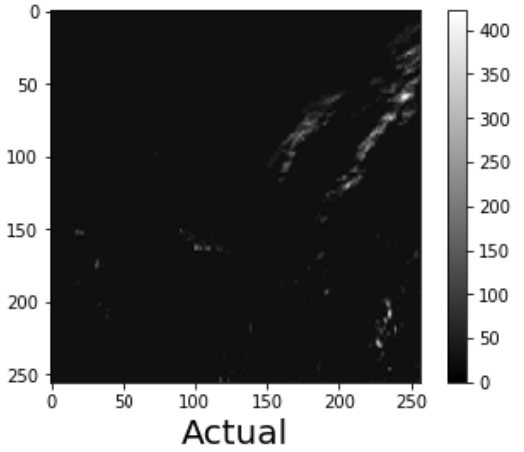
Linear Regression 1x1 Window (Test):





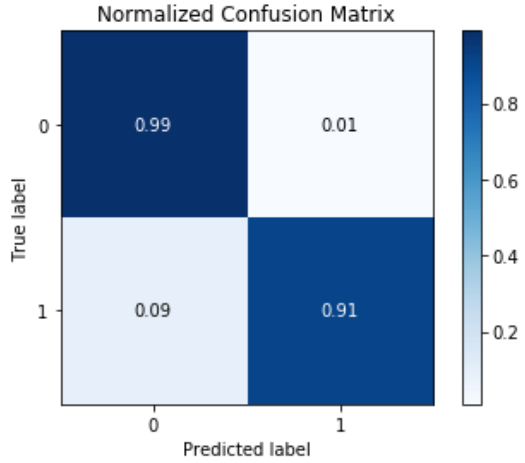
Test Image
Linear Regression
RMSE: 295.491889

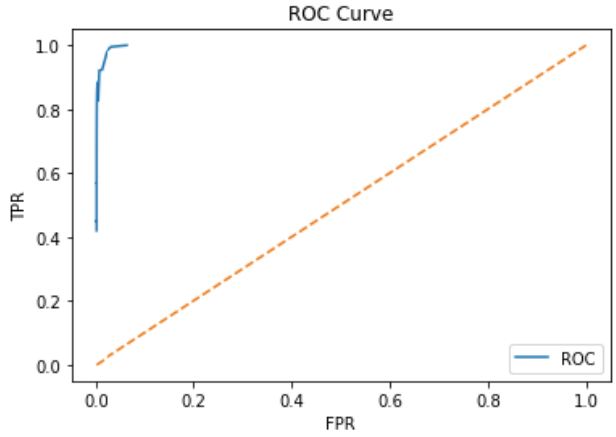




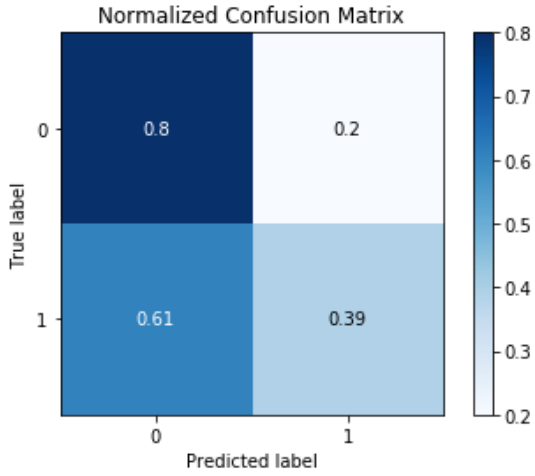
<u>Resolution to the issue presented before, we</u>
<u>scale the predicted output to 0 – max(actual radar)</u>

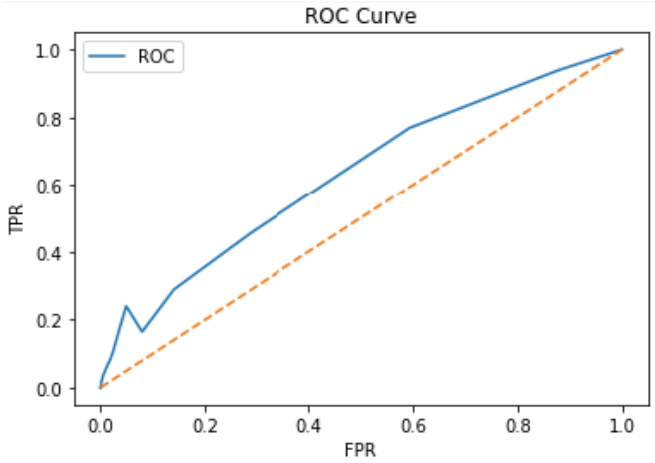
Random Forest Regression 1x1 Window (Training):





Random Forest Regression 1x1 Window (Test):

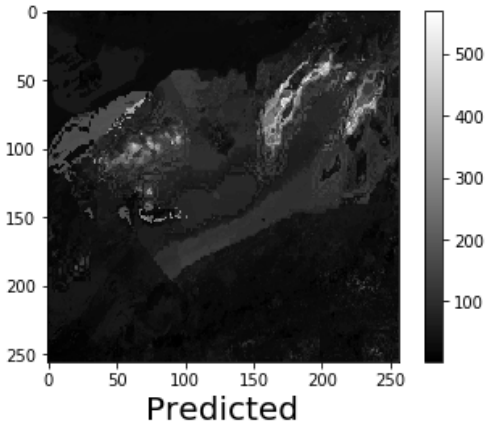


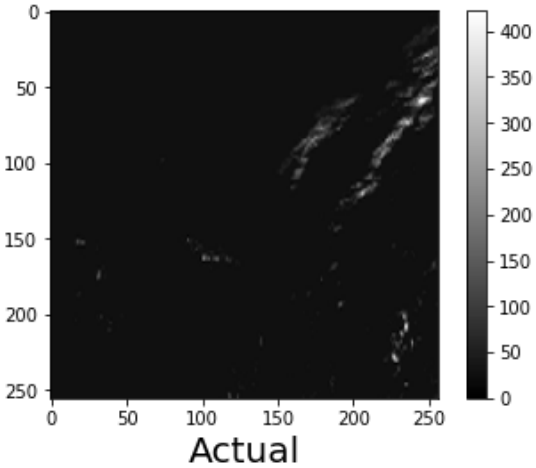


Test Image

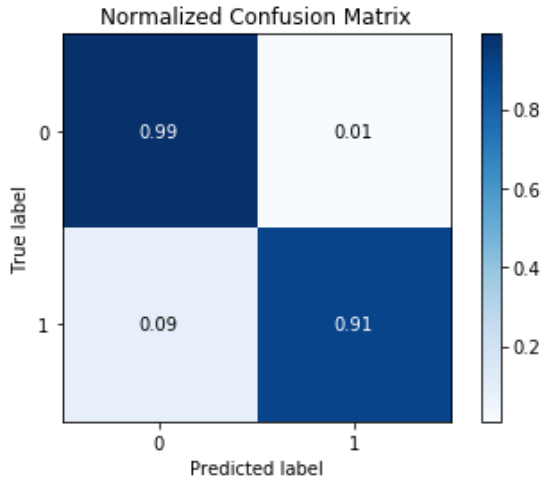
Linear Regression

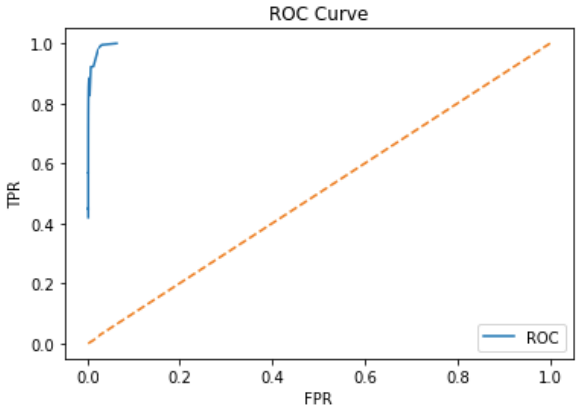
RMSE: 69.52103852333681



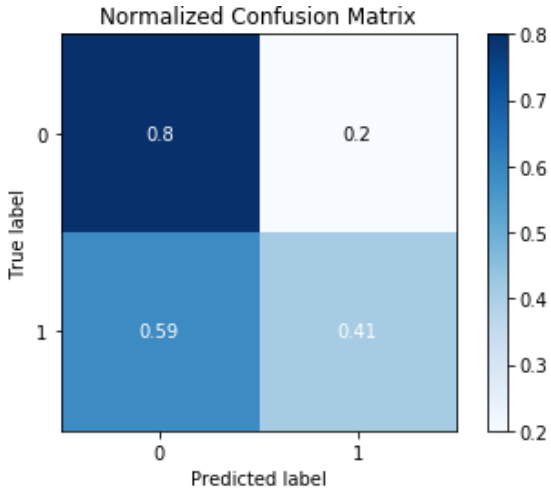


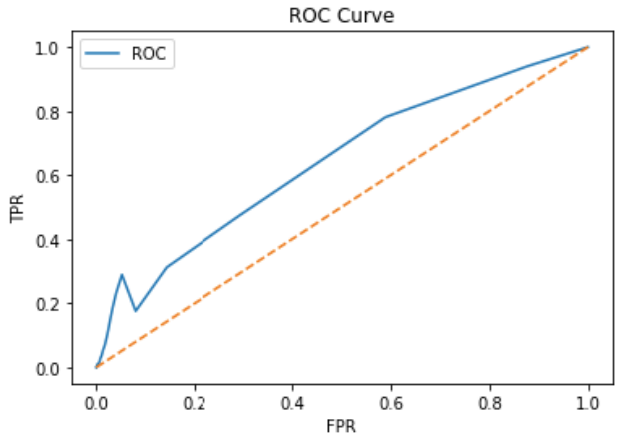
Random Forest Regression 3x3 Window (Training):





Random Forest Regression 3x3 Window (Test):

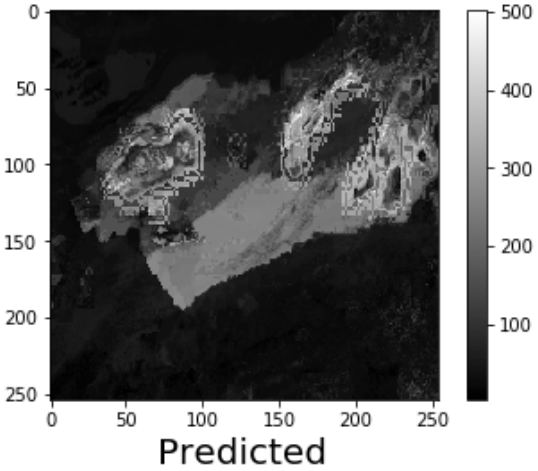


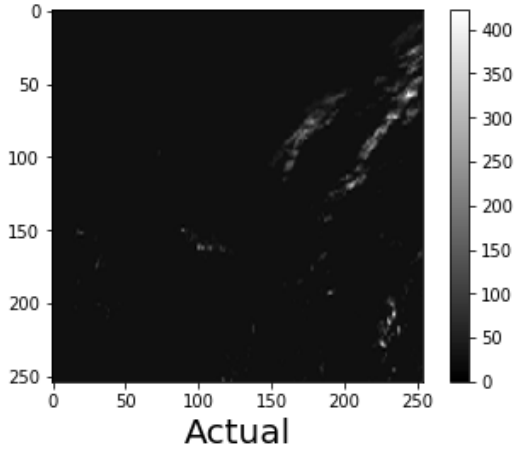


Test Image

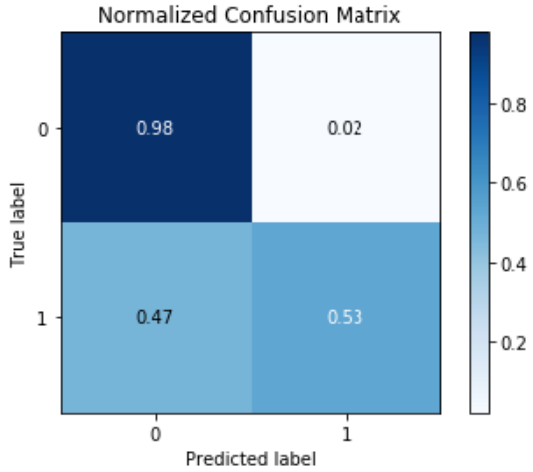
Linear Regression

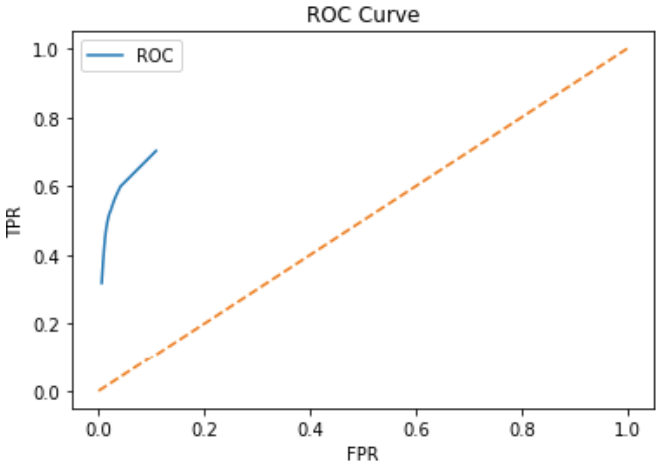
RMSE: 110.08173994833993



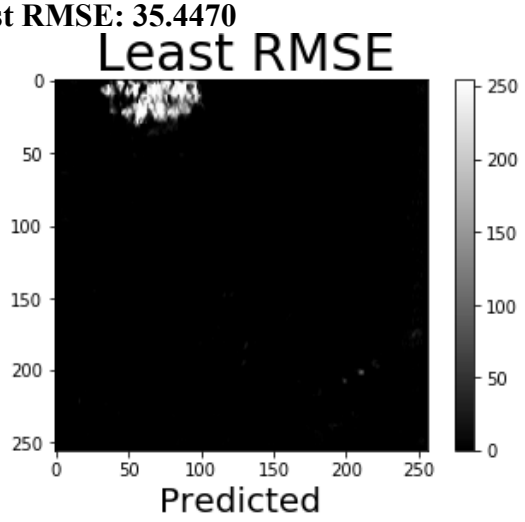


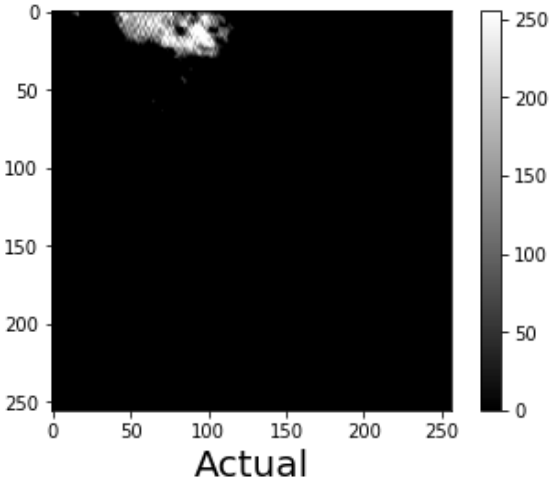
<u>Test set result</u> of Pix2Pix (Trained using satellite only, Sparse-less, latest implementation of pix2pix: runaug1)



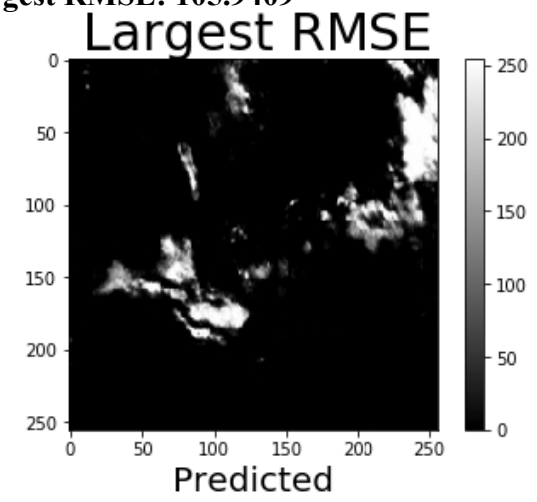


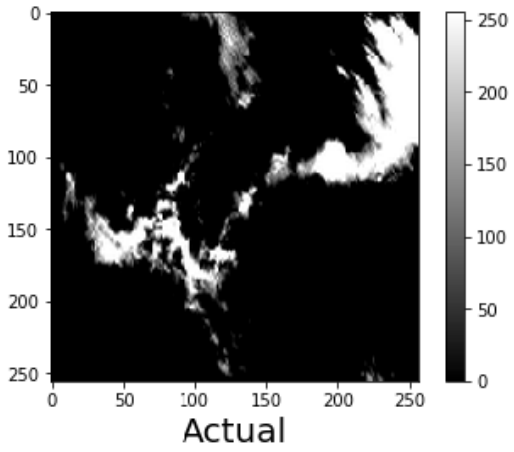
Least RMSE: 35.4470



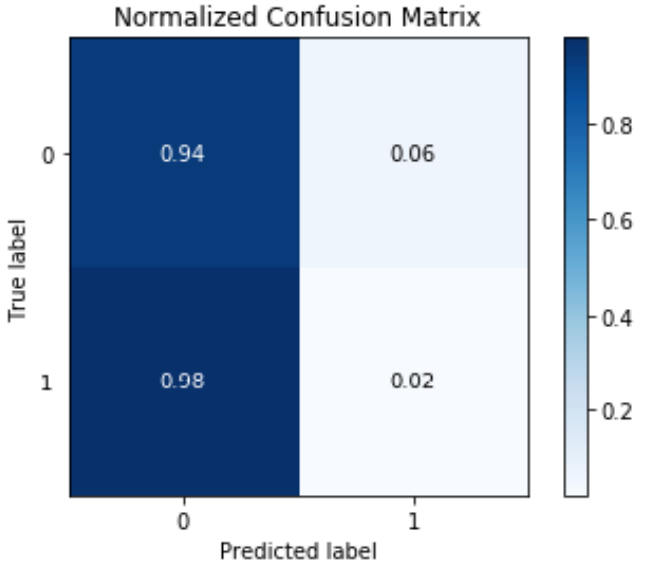


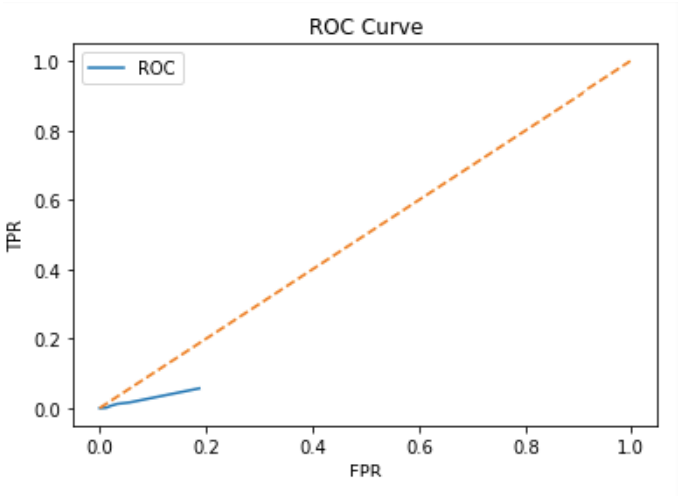
Largest RMSE: 105.9409



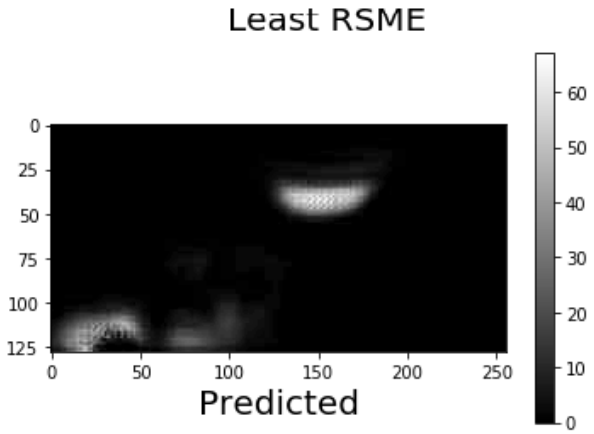


Test set result of Vid2Vid

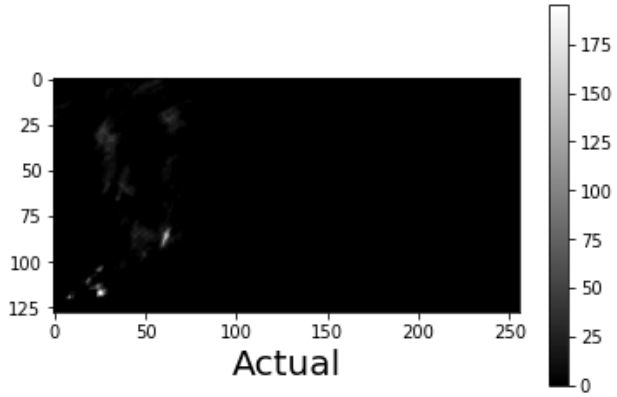




Least RMSE: 9.2666

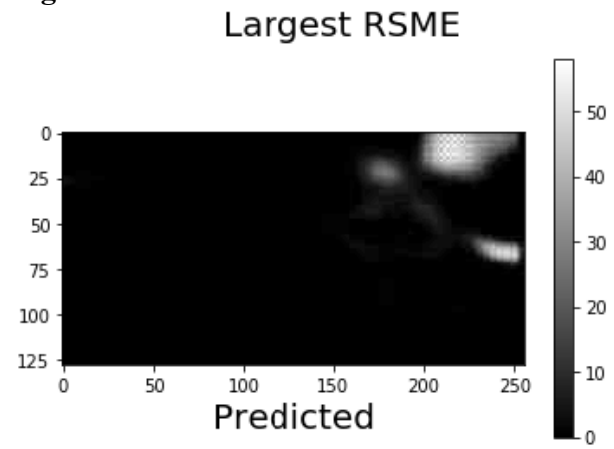


000/output_grayscalefake_B_07.jpg

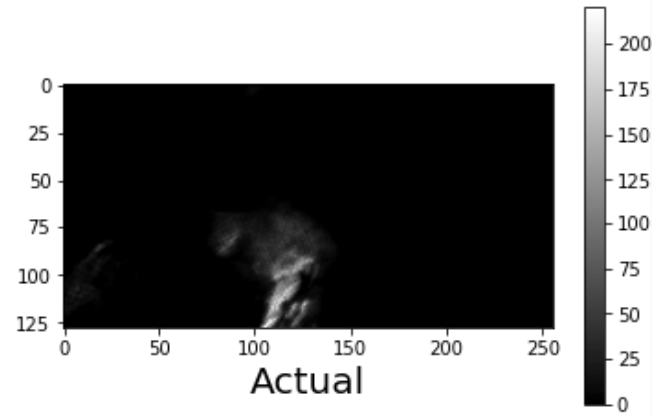


realradar/rad.20170402.00.png

Largest RMSE: 16.63874



000/output_grayscalefake_B_28.jpg



realradar/rad.20170402.21.png

A flaw was found in the Linear Regression workflow.

We realized that there was a disagreement in results for confusion matrix for Linear Regression among team members.

After further investigation of the confusion matrix, a flaw was, in fact, found in the confusion matrix for the predicted radar images.

The intensity range for the actual radar is [0 - ~800]. However, intensity range for the predicted radar is [0 - ~100]. As can be seen in the results of page 7.

When the threshold is calculated, using the procedure described in the Methods section, it only looks into the statistics of the actual radar. In the experiments, the threshold is evaluated as 42.14. As a result, the relative high threshold value classifies most of the pixels as Class 0: No Precipitation. To resolve, we could multiply the predicted pixels with max of radar and then divide by previous max of predicted pixels (as opposed to diving by the max of predicted pixels, first, and then multiplying with the max of radar as this could lead to error due to truncation).

The resolutions is applied to the Linear Regression and the results are presented again on page 8. The is obvious improvement in the ROC curve. However, the resulting confusion matrix seems unusual. This is because after scaling the predicted pixels to [0 and max of radar], most of the pixels fall over threshold = 42.14 and are classified as Class 1: Precipitation.

Moreover, the scaling issue was not apparent for Random Forest regression because the predicted radar images were close to the same range as of the actual radar image.

5. CONCLUSION

The Linear Regression model has the scaling issue for the predicted pixels. Even if pixels are scaled the low variance of the actual predicted pixels pushes most of the pixels to over the threshold. This could be potentially resolved using dynamic or adaptive scaling of the predicted pixels.

During Feature selection [Fig. 7, 8], it is concluded that Model data plays a significantly higher role in determining the pixel value of the radar. There is a possibility that Model data at NAM has already been corrected and biased to embed the radar information. However, according to Brain, “they are not affected by much.”

Below are the score for correctness in the Testing set for the models. Linear Regression is not included because more work needs to be done for properly scaling, defining, and classifying the pixels.

Model	Correctness (True Positive %)
Random Forest Regression 1x1 Window (Satellite + Model)	39%
Random Forest Regression 3x3 Window (Satellite + Model)	41%
Pix2Pix (Satellite only)	53%
Vid2Vid (Satellite)	2%

Using our evaluation, it is concluded that Pix2Pix performs the best due to its more advance GAN architecture that does a way better job of accounting for the spatial information in the data than regression models.

6. REFERENCES:

[1] Kubat, Miroslav, Robert C. Holte, and Stan Matwin. "Machine learning for the detection of oil spills in satellite radar images." Machine learning 30.2-3 (1998): 195-215.

[2] Moazami, Saber, et al. "Comparison of PERSIANN and V7 TRMM Multi-satellite Precipitation Analysis (TMPA) products with rain gauge data over Iran." International Journal of Remote Sensing 34.22 (2013): 8156-8171.

[3] Behrangi, Ali, et al. "PERSIANN-MSA: A precipitation estimation method from satellite-based multispectral analysis." Journal of Hydrometeorology 10.6 (2009): 1414-1429.

[4] Conti, Francesco Lo, et al. "Evaluation and comparison of satellite precipitation estimates with reference to a local area in the Mediterranean Sea." Atmospheric Research 138 (2014): 189-204.

[5] Wang, Ting-Chun, et al. "Video-to-video synthesis." arXiv preprint arXiv:1808.06601 (2018).

[6] Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." arXiv preprint (2017).

Author's note: At the time of the submission of the paper, 5x5 windowed Random Forest, and 3x3 and 5x5 windowed Linear Regression models are still being trained (after being in 4 days in NOAA's node) and could not be included in the paper. Though the source code already has the respective runnable code, the evaluated python notebook will be placed in the repo once the node finishes evaluating. 5x5 windowed Random Forest may potentially perform better than 3x3 windowed Random Forest presented in the paper, but it is unlikely it performs better than pix2pix. Also, note that Pix2Pix was trained on Satellite only making it, inherently, a more robust model.