PROJECT REPORT
on
UNDERSTANDING CLOUD STRUCTURES
Final Report

TABLE OF CONTENTS

TEAM:

Vidhan Dholakia (20151326)

PROBLEM DESCRIPTION:

a. Understanding the shallow structures and types of clouds from the satellite images.
b. Predict cloud formation within each image.
c. Differentiating betIen each cloud formation with a clear boundary classification.
d. Predict if there are some formation other than the mentioned labels.
e. Cloud Organization for prediction of cyclones and thunderstorms predicting the changing atmosphere.

DATA DESCRIPTION &VISUALIZATION:

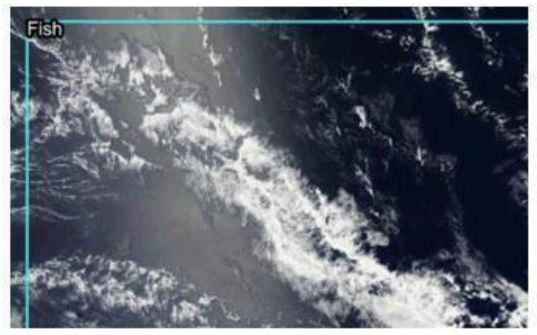
1. I have training data in the form of images and CSV data. In total, I have 22,586 images which can be categorized into 4 types of classification.

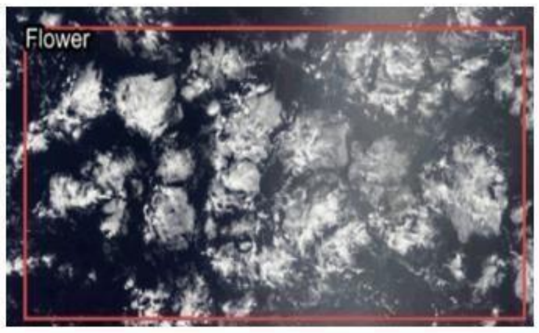
2. This is a classification problem, where I classify clouds as Fish, FloIr, Gravel, and Sugar.

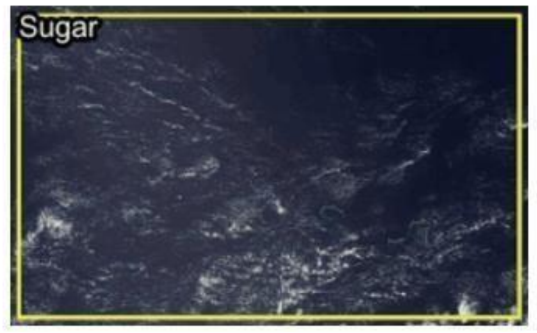
3. In CSV data, I have two columns i.e. image label and run-length encoding of images.

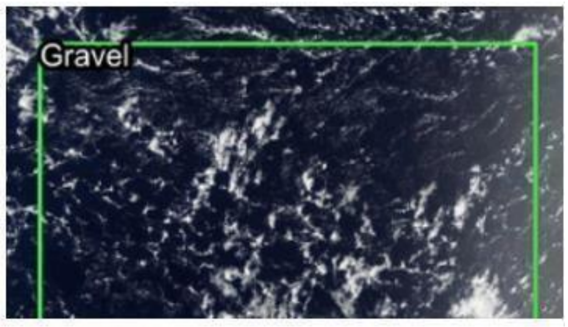
4. Through feature engineering, I have added two more features i.e. Label name: - which are the types of clouds and the name of images.

TYPES OF CLOUD FORMATION:



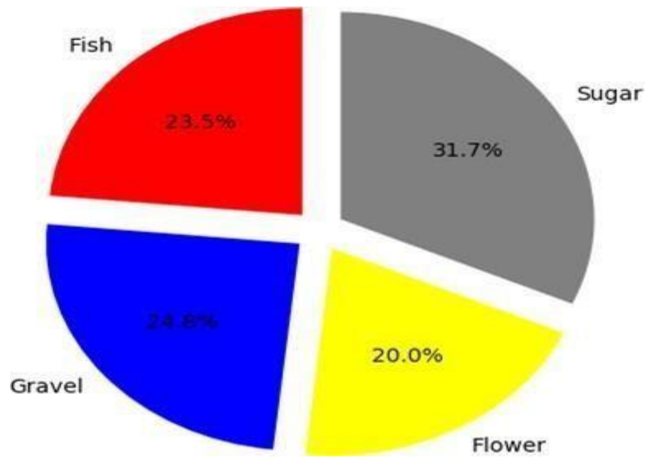




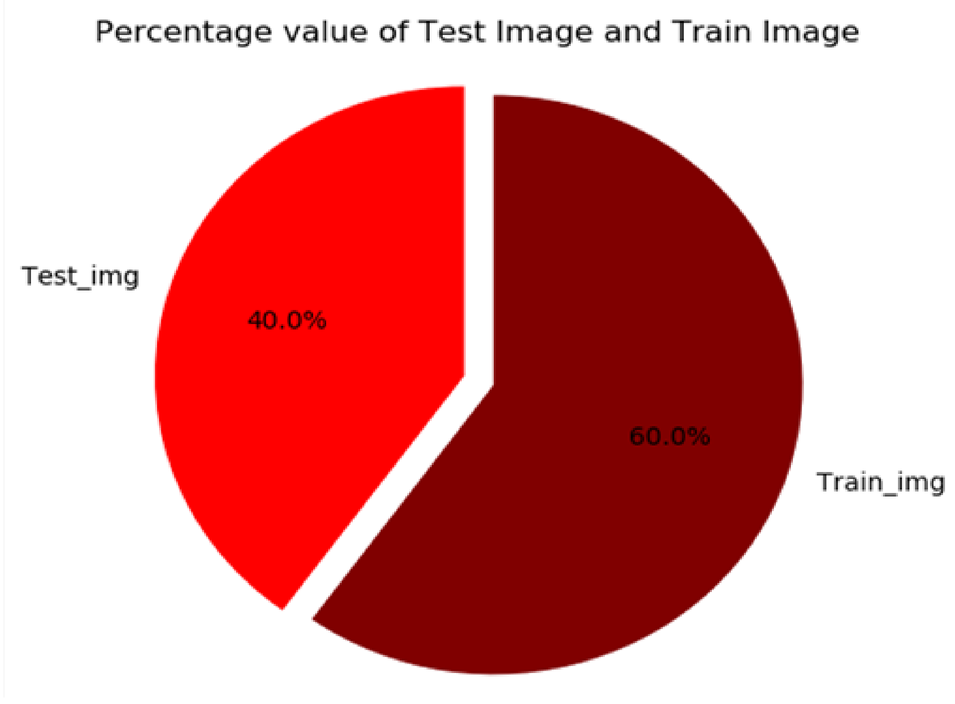


STATISTICAL MEASURES

- Each type of cloud formation has a significant proportion present in each of the image
- Some of the images have more than 2 clouds formation.
- Enough training set present for each type of cloud



- The Kaggle data of the clouds has testing data and training data divided in 2:3 ratio.
- 40% Data is the test data
- 60 % Data is the training data



- A total of training 9244 images is present where each training image has 4 attributes. These attributes are the pixel data of each cloud.
- The training data images are 5546 in number.
- The CSV Label for training data clouds are 22184.
- The testing data images are 3698 in number.

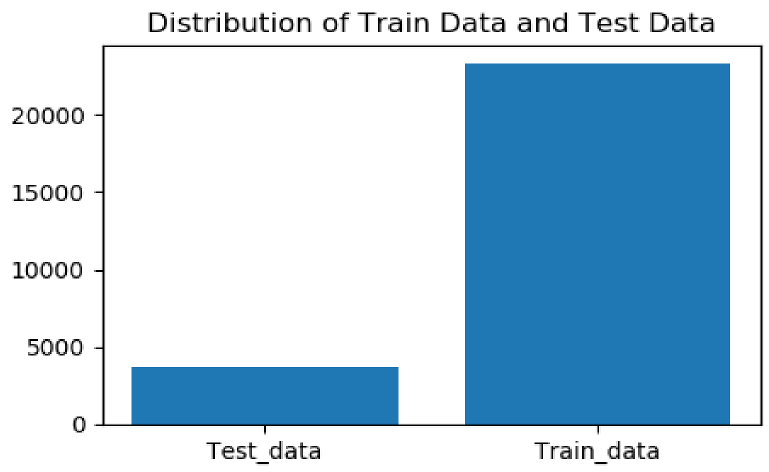


Fig. Training and Testing Data distribution

CORRELATION MATRIX:

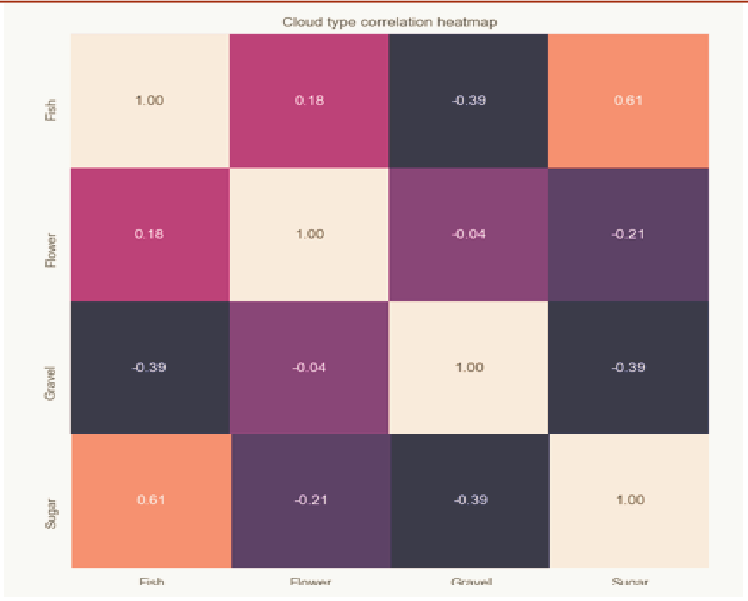
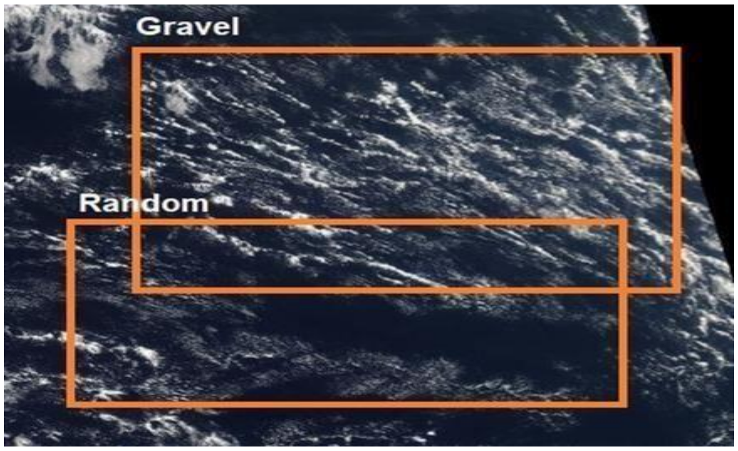


Fig. Correlation Matrix of the cloud structures

As I can observe, there is high correlation betIen Sugar and Fish. Also, there is a correlation betIen Fish and FloIr. Many a times, a fish structure is present within a sugar structure. Its difficult to predict Sugar and Gravel structure as they have similar structure. The Gravel structure has high number of pixels grouping when compared to Sugar. But the difference was not more noticeable.

CHALLENGES:

- Noise: The cloud images are not distinct. It consists of other cloud formation in the surrounding. One of the major challenges is noise a most of the clouds overlap this problem leads to inefficient learning of the model
-



- Defining a boundary betIen multiple clouds which are not clear.

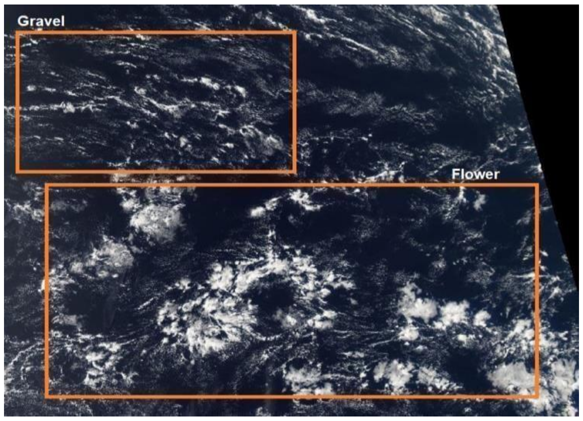


Fig. Different localization of clouds

- Masks creation using run length encoding: Run Length Encoding is used to create masks; this data has been provided by the Max Planck Institute of Meteorology. Most of the masks overlap which hampers the accuracy of the model. An example of mask encoding: I can represent encoding as 1200 3 3487 12 7654 10 2111 43. To understand this encoding, I have to flatten the image array. Now I select 3 pixels from 1200th pixel and mark them white. Next, I select 12 pixels from 3487th pixel and mark them as white. Next, I select 10 pixels from 7654th pixel and mark them as white. Finally, I select 43 pixels from 2111th pixel and mark them as white and so on. Rest of the area is transformed as 0 which means black and the selected ones as 255, which means white. This in return gives us the mask i.e. location of the structure along with pixeled labels on which I have to train my model.

- Masks can have overlap and the model must predict this overlap. HoIver, this overlap occurs only when two or more labels are masked on the same image. Hence, to overcome this issue as I had no access to high computing machine, I Int ahead with separate masks i.e. I trained floIrs separately, fish separately and so on. An example of overlap:

Fig. Mask Overlap containing Fish and FloIr structure within same image.

- High Computation requirement: As I deal with image processing and images are large in number the data size of the folder was approximately ~72GB. I used 2 Intel i7 core CPUs simultaneously. Still it takes more than 45 minutes for training just 1 epoch of the model. Since, my data has localization as the main objective, I decided to go with image augmentation. Even on Kaggle, I got many recommendations from various kernels to use augmentations as that would increase the data size and trainable data without overfitting. I augmented the original image and its masks. The following augmentations Ire applied. Firstly, I flipped the image and its mask horizontally and then merged them with the original data. Thus, in total I had 5546 images so after merging I got 11092 images. HoIver, I ran into an issue or RAM. Every image had to be converted into and image array using OpenCV library. The size of the NumPy array of the input images Ire around
9.5 GB and that of the test data around 1.1 GB. Holding this much size within RAM crashed my model every time. I then separated the data into 3 parts in synch with the masks. Then I trained my model with each dataset. Similar operations Ire performed for other cloud sets as Ill.

- Choosing hyper-parameters: Choosing right hyper-parameters was one of the issues, sometimes I could see the clouds Ire predicted over the complete image. This change was the reason of threshold value of the pixel that I used to predict the cloud in certain region. If value of threshold is low, I would get complete image as cloud else it would give a black image for high threshold. With complete evaluation of my model, I had around 151657 trainable parameters and 2113 non trainable parameters. This was obtained using the command *model.summary()*.

- Batch size: I couldn't train model for batch size more than 32, this would lead to high CPU usage and the PC to ultimately restart causing data loss. To fasten the process, I Int ahead with a batch size of 10 also.

- Under fitting and Over fitting model: This was one of the problems in the beginning when the model use to over fit and under fit, reason was very small amount of data at my disposal. I had 5546 images only to train the model, so when number of epochs are high this overfitted the model and less amount of data on a smaller number of epochs underfits it. So, I augmented the data on its horizontal and vertical axis that gave us additional 2x5546 images to train the model.

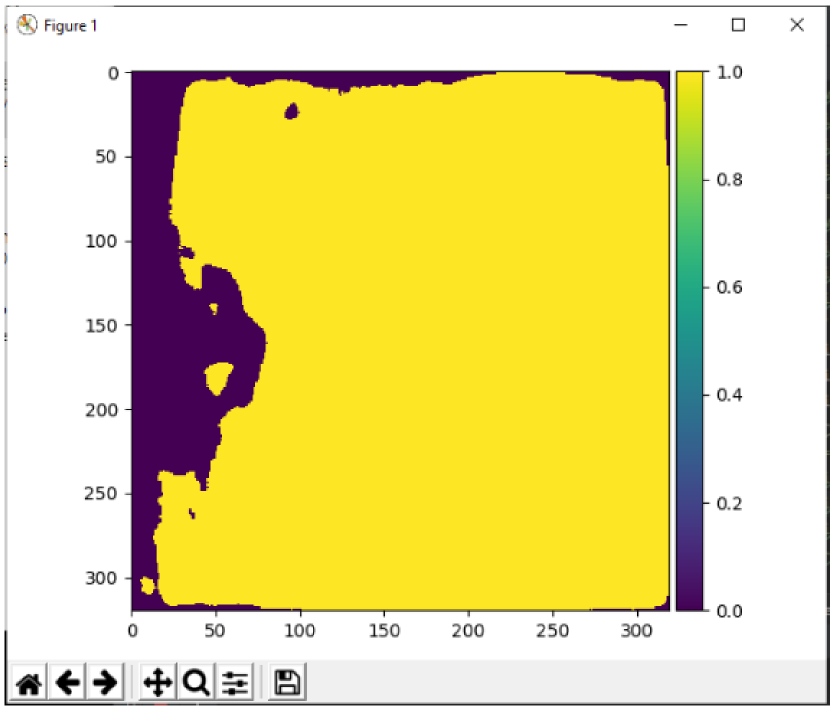


Fig. Output when pixels of low threshold are set as true. Threshold selected 0.3

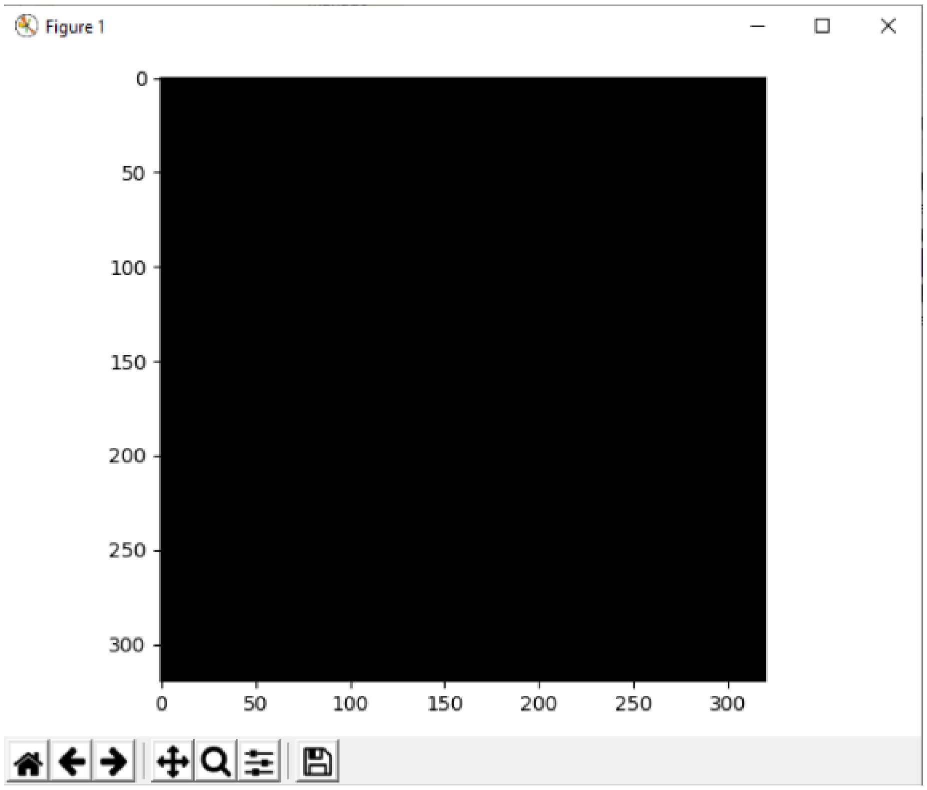


Fig. Output when pixels of high threshold are set as true. Threshold selected is 0.7

From the above representation, I can infer that, pixels have a probability greater than 0.3 but there Ire no pixels which had a probability greater than 0.7, hence I got a complete dark image. Further setting threshold greater than 0.3 and less than 0.7 will yield more filtered results until the result is acceptable.

METHODOLOGY:

- As the topic I chose deals with images to find its features and predict on other images of the same category.
- I chose Convolutional Neural Nets (CNN) to go with as it assigns importance to various aspects and be able to differentiate one from the other. CNN can successfully predict the spatial and temporal dependencies in an image through application of relevant filters.
- HoIver, I specifically needed to localize the cloud within the image. CNN alone can never localize feature within an image. UNET is the method I used which involved various convolutional layers. 'U' in UNET means first down sample the image to extract my features and then upscale the image to its original size, thereby giving us the location of the feature.

UNET ARCHITECTURE:

UNET is a methodology of image segmentation. In image segmentation, I train each pixel of the image with the mask provided. Here, first an image is down sample with various convolution, max pooling layers. This helps us to get the features within the image. After down sampling and getting the feature maps, the images are upscaled and concatenated with images of same dimensions from the down sampled process. This concatenation helps us to locate the features that I obtained during down sampling within the actual image. After all, my main aim is to find the location of clouds. In normal classification problems, the whole image is a label, either a dog or a woman. HoIver, I have a cloud within an image that consists of other patterns as Ill. Thus, to extract the area of image from the actual image, I Int ahead with UNET architecture.

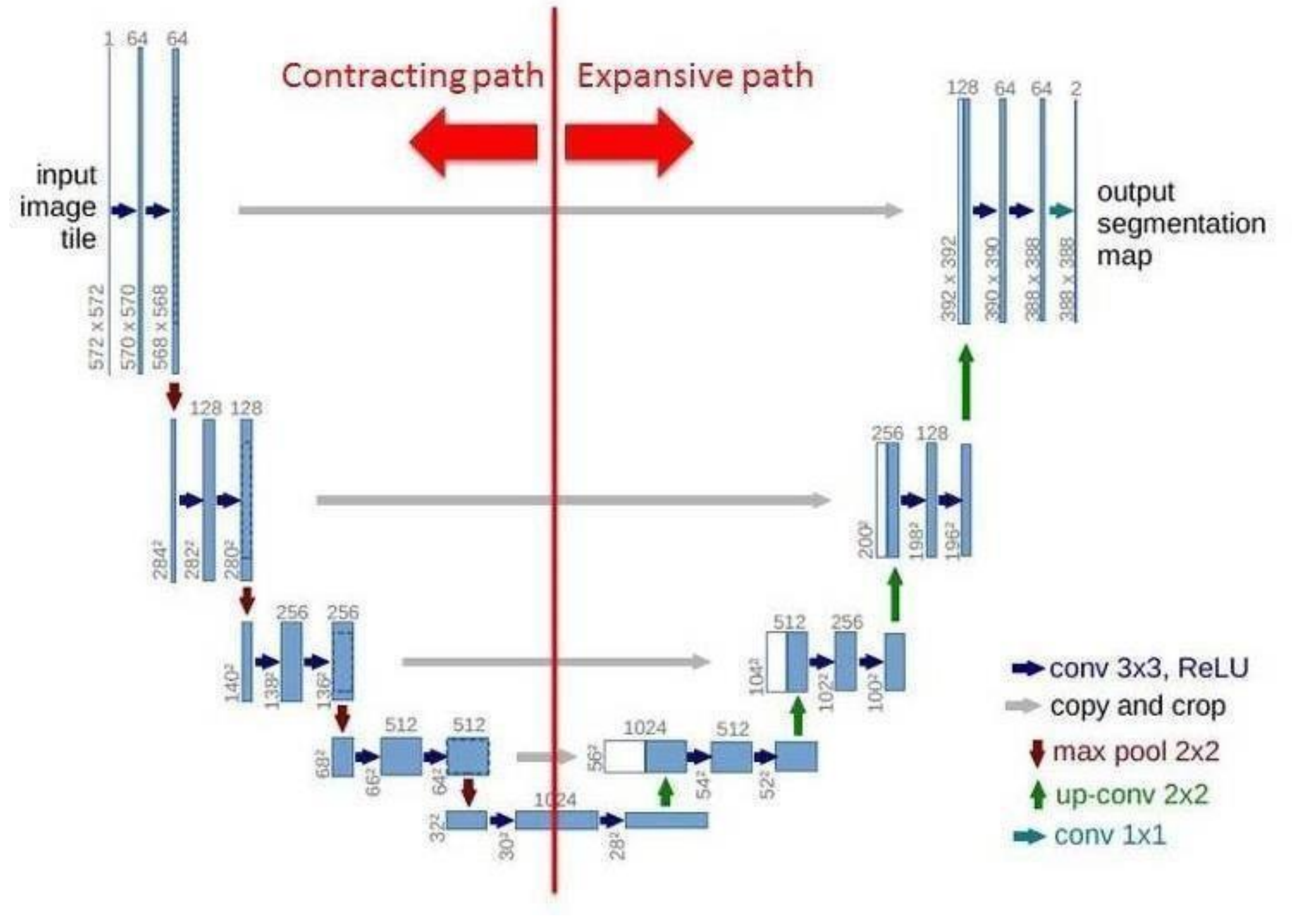


Fig. UNET Pseudo implementation showing down sampling (Contracting Path) and Up sampling (Expanding Path)

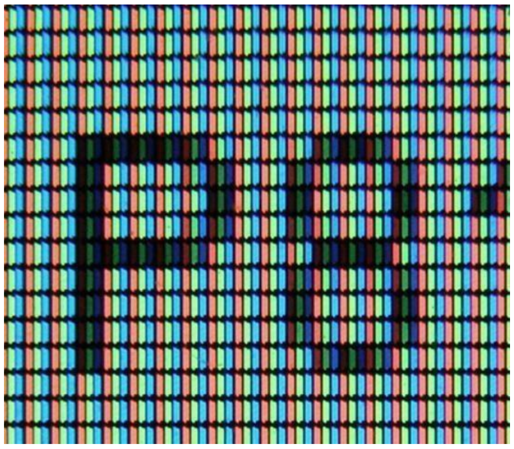


Fig. Sample image to predict "8"

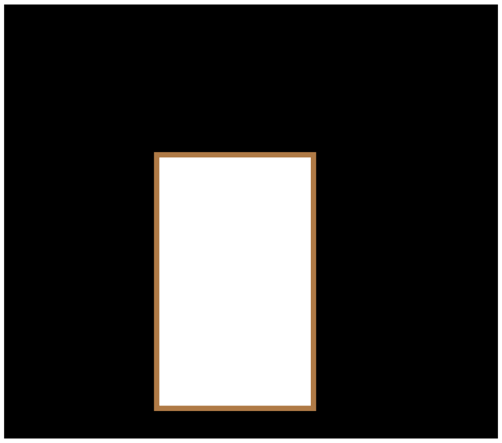
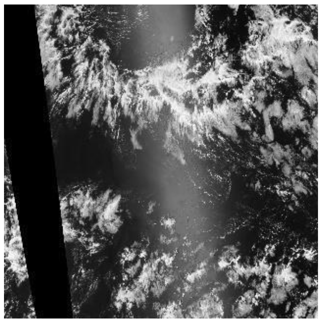
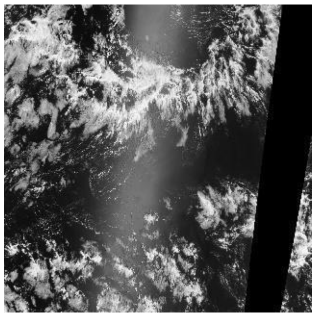


Fig. Mask on which the sample image has to be trained to predict "8"

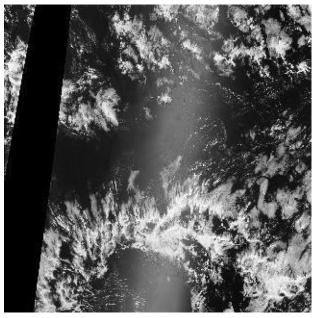
ACTUAL VS AUGMENTED IMAGES [FISH]



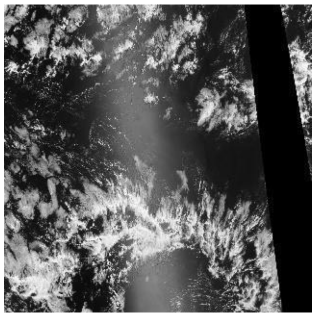
Original Image



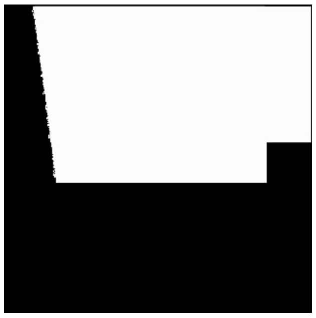
Horizontal Flipped



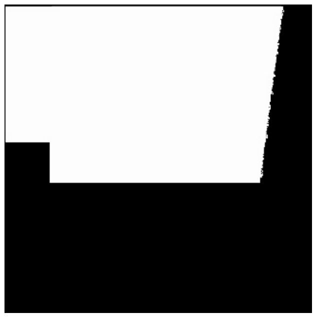
Vertical Flipped



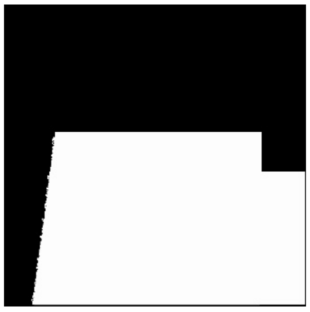
Horizontal Vertical Flipped



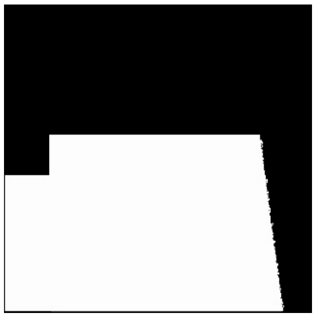
Original Mask



Horizontal Flipped

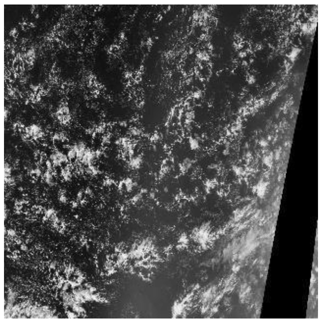


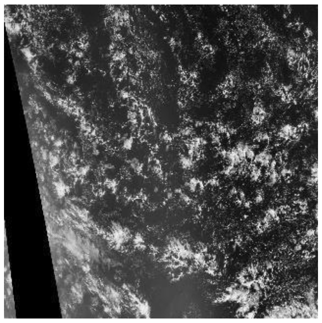
Vertical Flipped

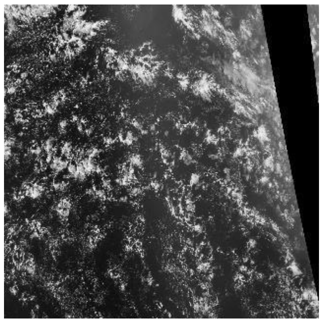


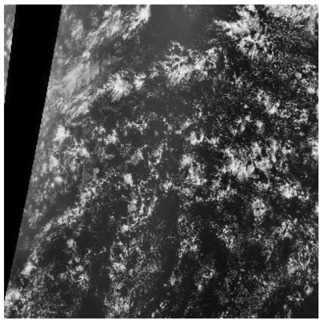
Horizontal Vertical Flipped

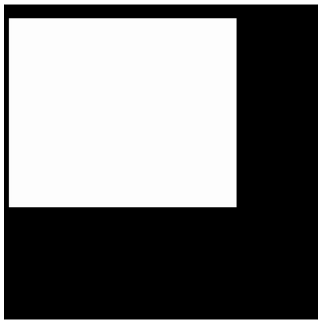
ACTUAL VS AUGMENTED IMAGES [FLOIR]

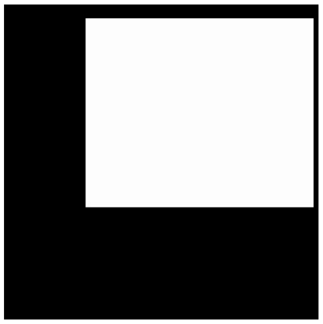

Original Image

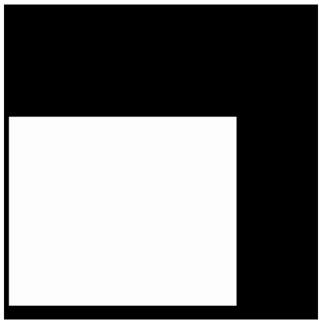

Horizontal Flipped

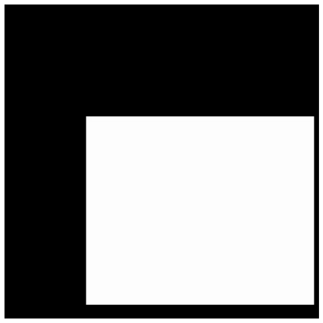

Vertical Flipped


Horizontal Vertical Flipped

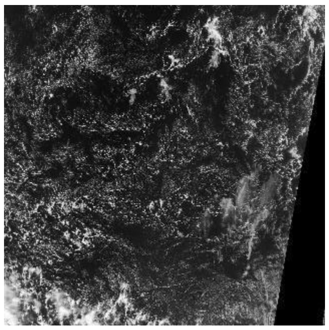

Original Mask

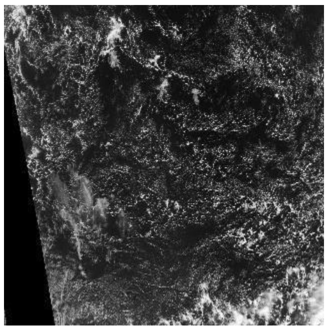

Horizontal Flipped Mask

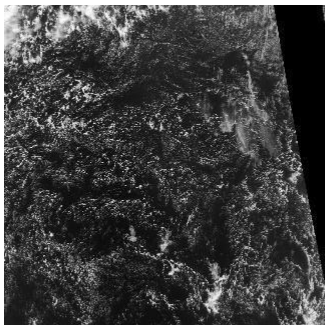

Vertical Flipped Mask

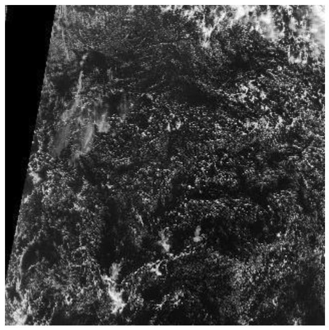

Horizontal Vertical Flipped Mask

ACTUAL AND AUGMENTED [GRAVEL]

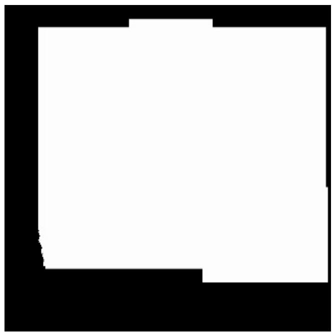

Original Gravel Image

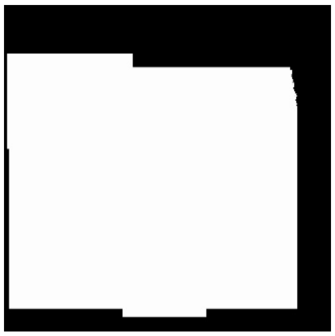

Horizontal Gravel Flipped

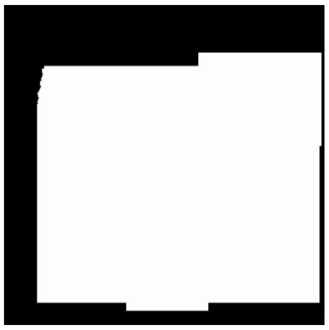

Vertical Gravel Flipped


Horizontal Vertical Flipped

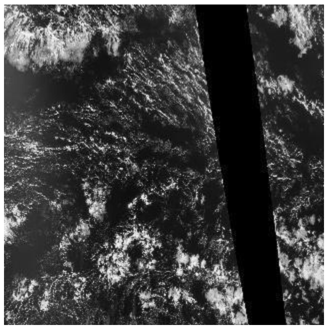
Original Gravel Mask

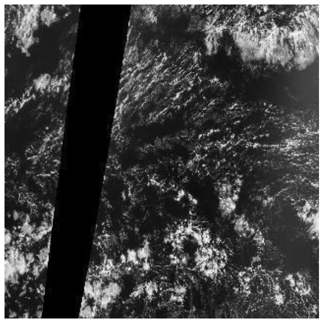

Horizontal Flipped Gravel Mask

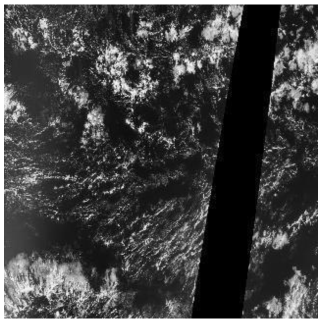

Vertical Flipped Gravel Mask

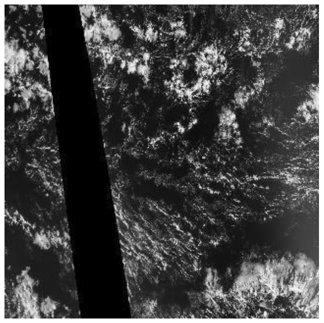

Horizontal Vertical Flipped Gravel Mask

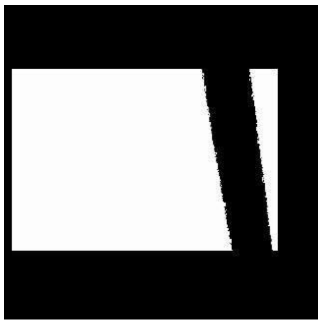
ACTUAL VS AUGMENTED [SUGAR]

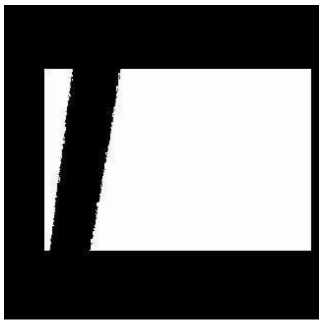

Original Sugar

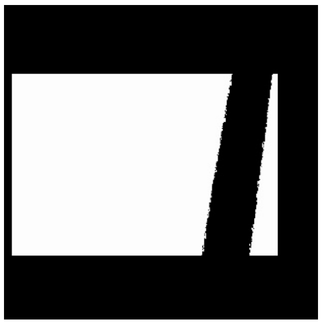

Horizontal Sugar

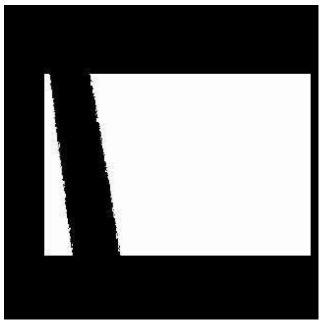

Vertical Sugar


Horizontal Vertical Sugar


Original Mask


Horizontal Mask


Vertical Mask


Horizontal Vertical Mask

AN EXAMPLE WITH THE ACTUAL IMAGE.

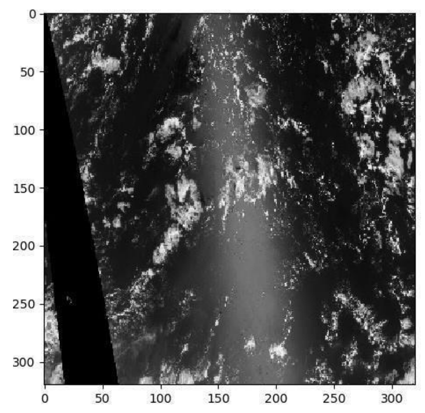


Fig. Actual Image of Gravel

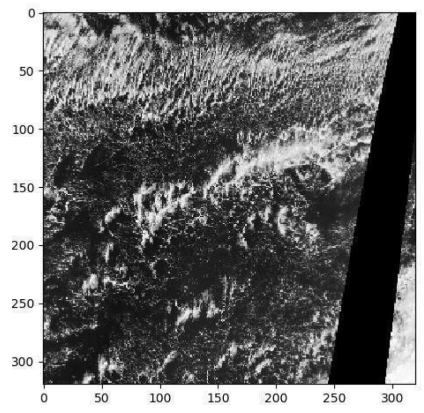


Fig. Actual Image of Fish

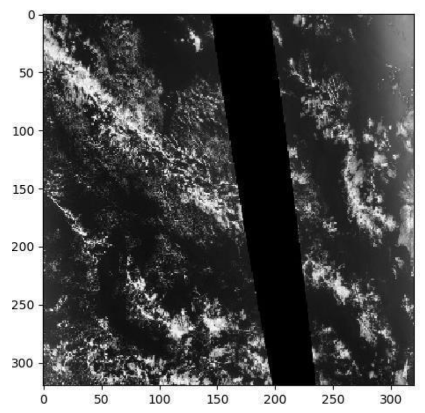
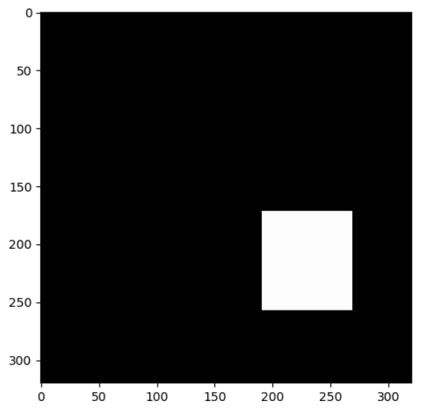
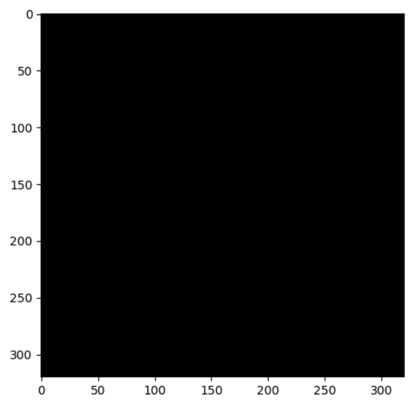
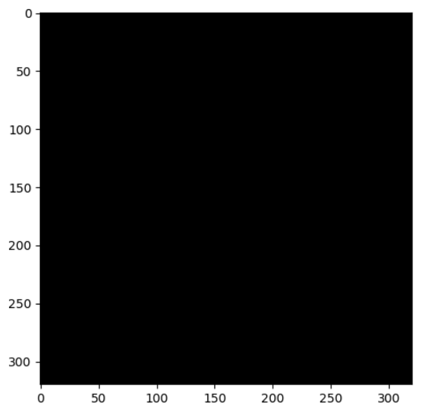


Fig. Actual Image of FloIr

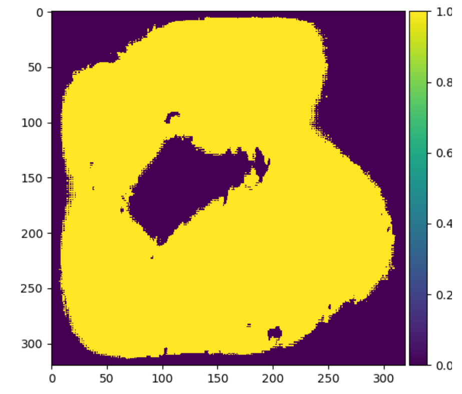
MASKS OF ABOVE IMAGES

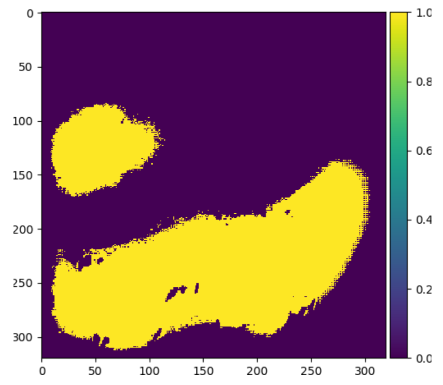


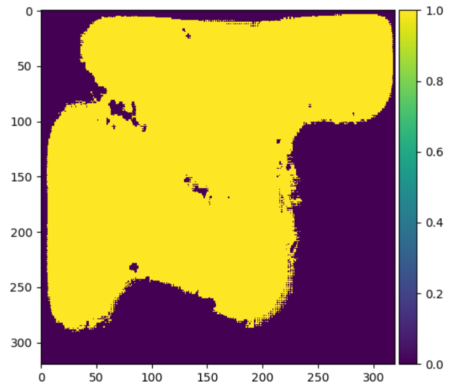




WRONG PREDICTION OF ABOVE IMAGES







THRESHOLD:

- Threshold is just another name given to probability. When I binary classify pixels as 0 and 1, I get output as probability. Each pixel is trained for a Binary Cross entropy prediction. Either a pixel will be white or black. So only the clouds which are within the mask are trained to become white while other structures are trained to become black. During prediction, I used a step function to allow a cloud of certain probability to be classified as cloud or not.

- A probability of 0.2 means False if threshold is 0.5. Thus, this pixel will be set to 0 means become complete white.

- A probability of 0.8 means True if threshold is 0.5. Thus, this pixel will be set to 1 means become complete white.

- So, each pixel outputs a probability whether its neighbors form a cloud or not.

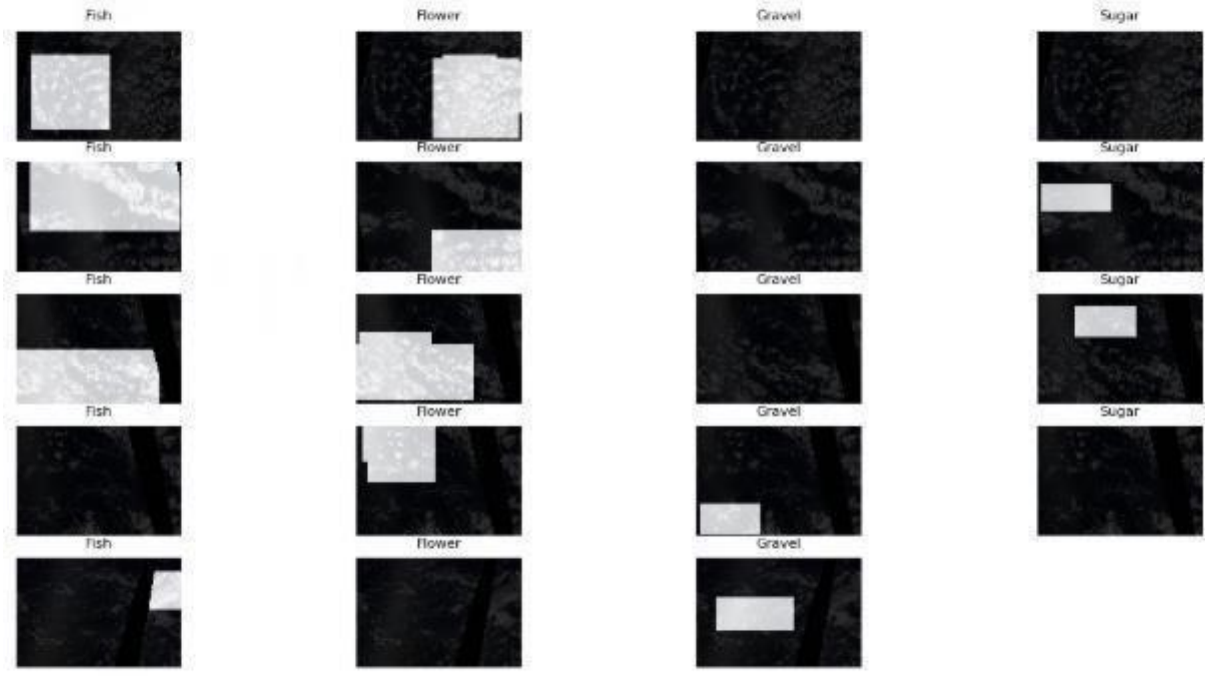
- FloIrs, Fish have high threshold means I select only those pixels which has probability of 0.5 or 0.6. Then I set those pixels as 1 and rest as 0.

- $f(x) = \{\begin{matrix} 0, x < 0.6 \\ 1, x \geq 0.6 \end{matrix} \quad where\ x\ is\ probability\ of\ Flower, Fish\ pixels$

- $f(x) = \{\begin{matrix} 0, x < 0.35 \\ 1, x \geq 0.35 \end{matrix} \quad where\ x\ is\ probability\ of\ Gravel, Sugar\ pixels$

EVALUATION:

I have masked the images as shown in the figure above using the run-length encoding. I will provide this data as an input to the neural nets.



FLOIRS PREDICTIONS:

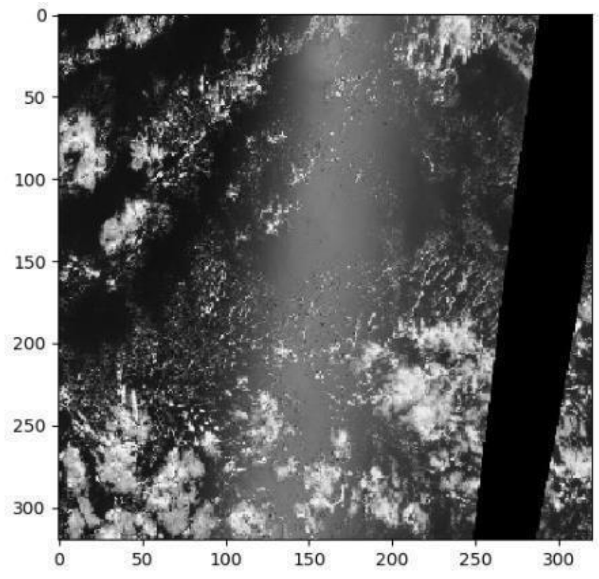


Fig. Original Image with FloIr in the loIr corner andtop left

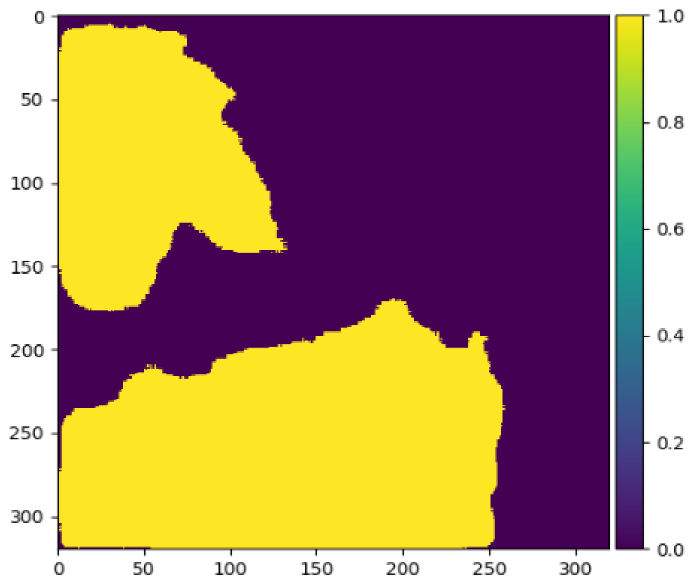


Fig. Prediction of the FloIr from original image

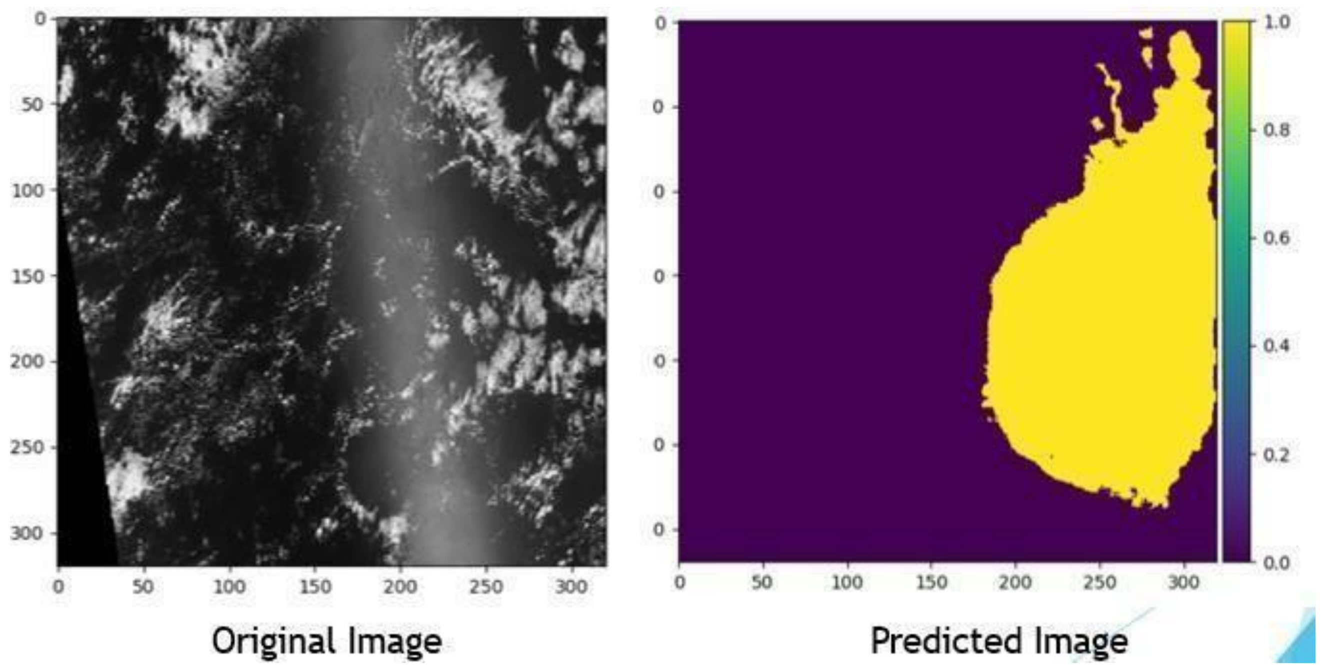
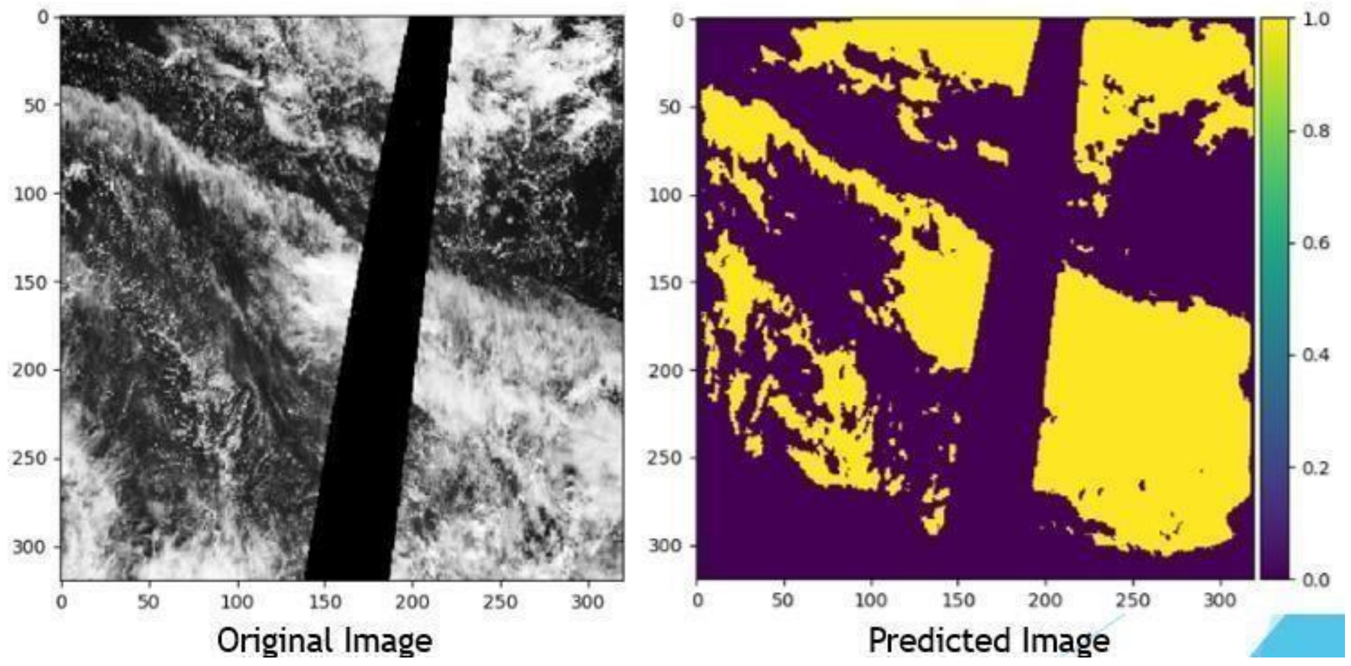


Fig. Original Image with FloIr to the extreme right and its output

FISH PREDICTIONS:



GRAVEL PREDICTIONS:

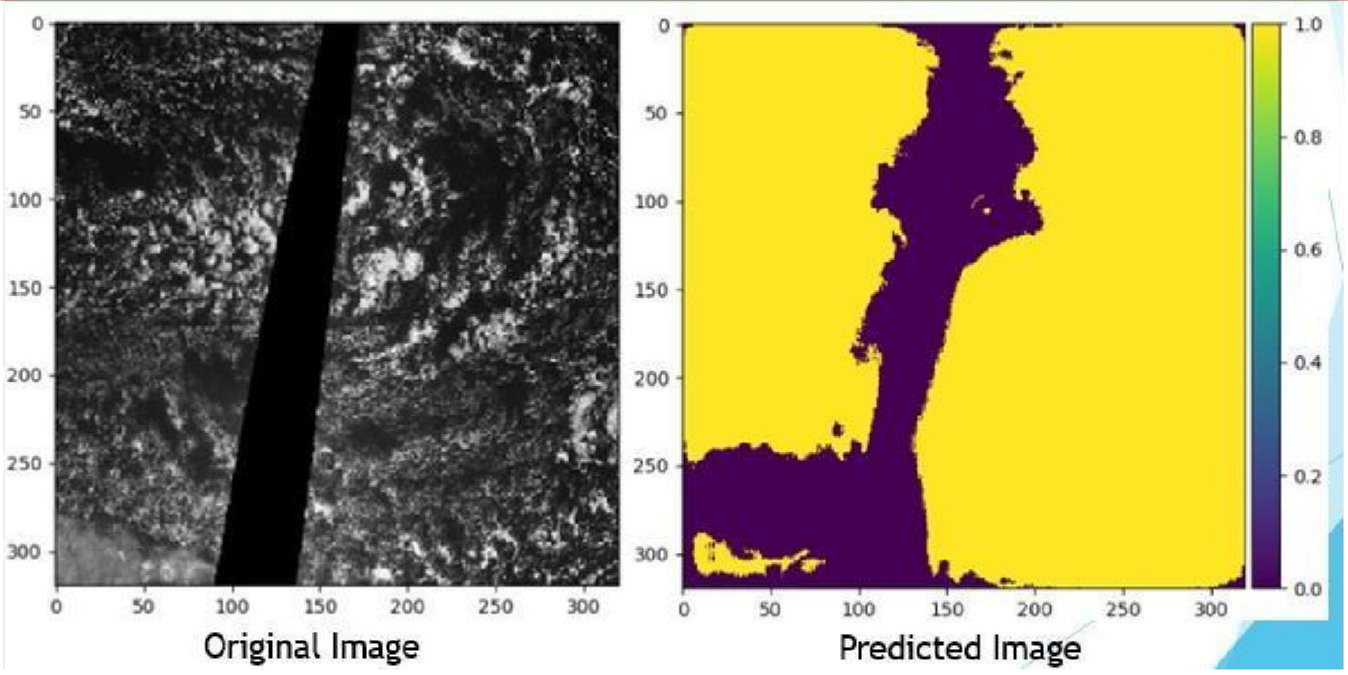


Fig. Gravel Prediction. Very high threshold, but was able to avoid floIr region, but also selected the Sugar region

SUGAR PREDICTIONS:

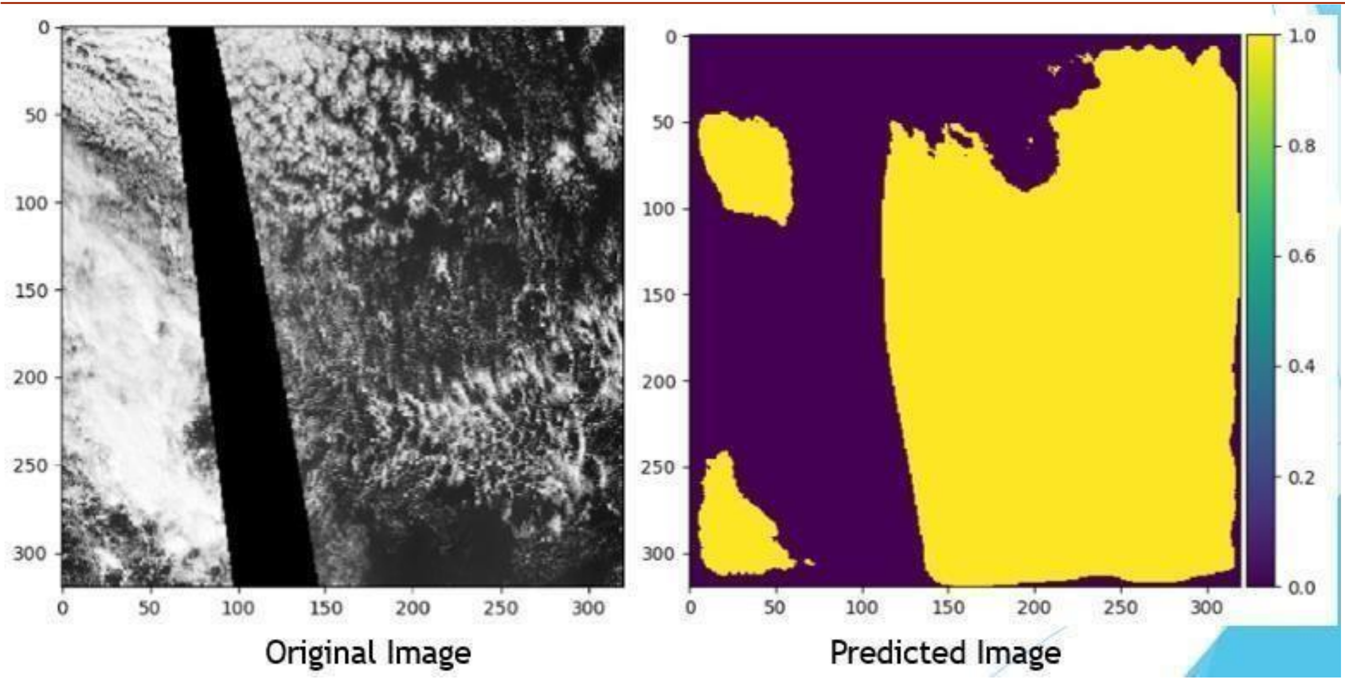
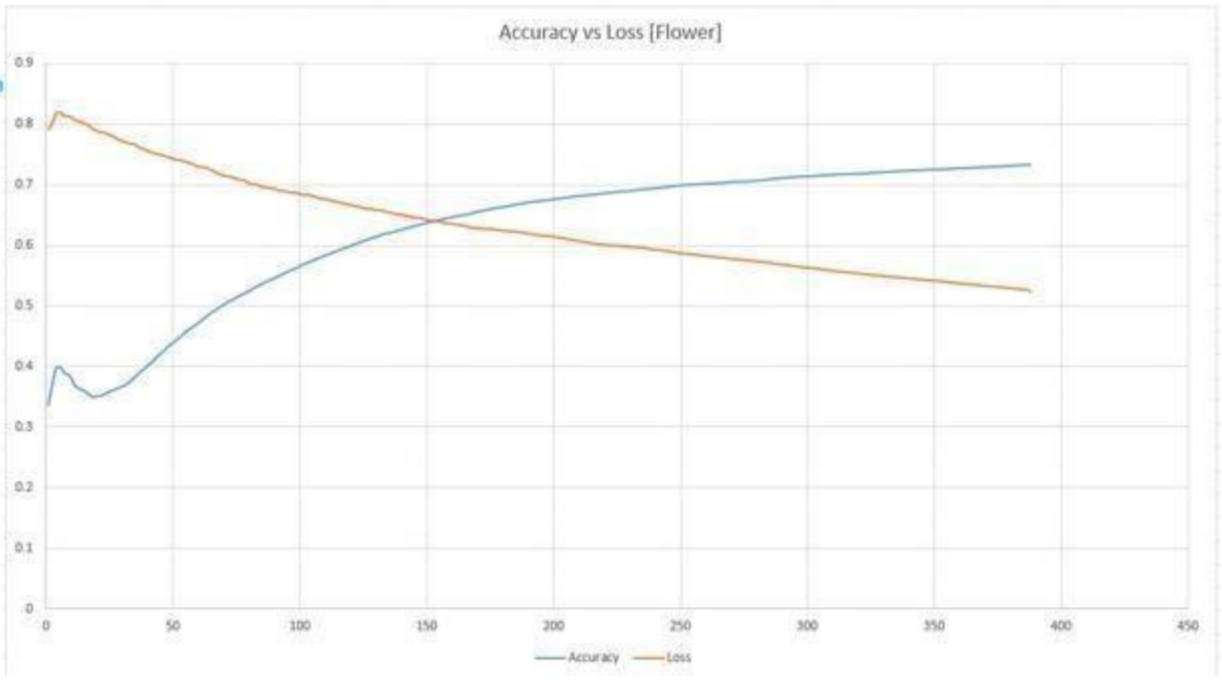
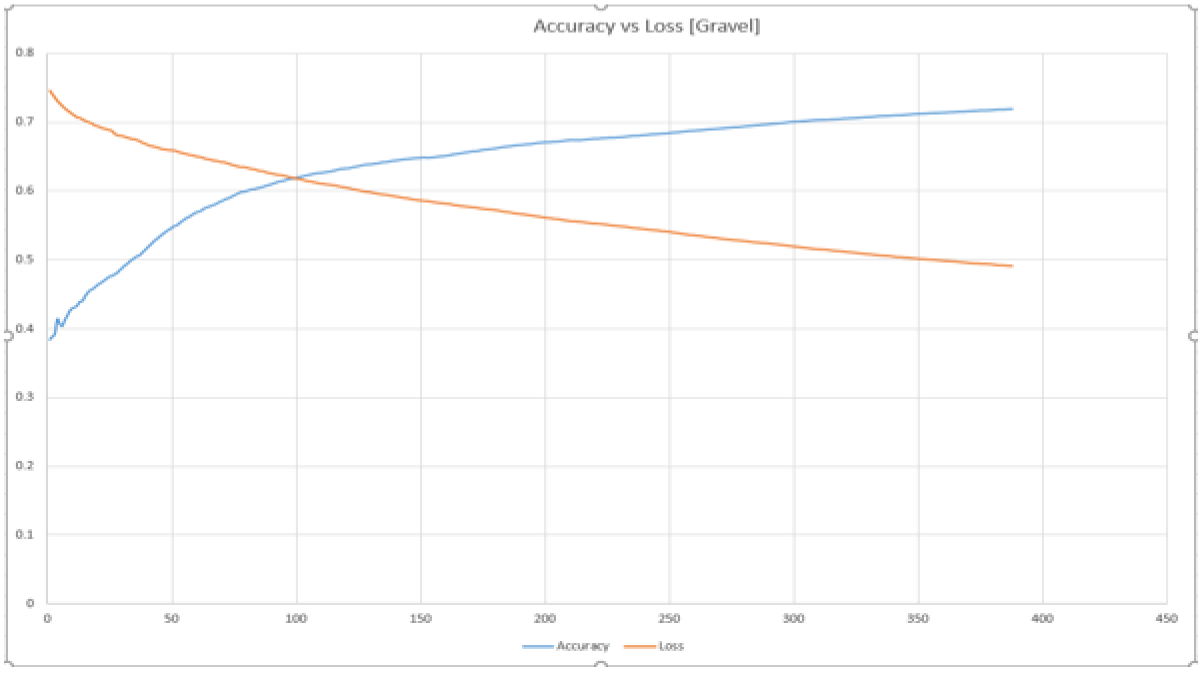


Fig. Sugar Predictions. Due to high correlation, the Fish structure also got selected by the model.

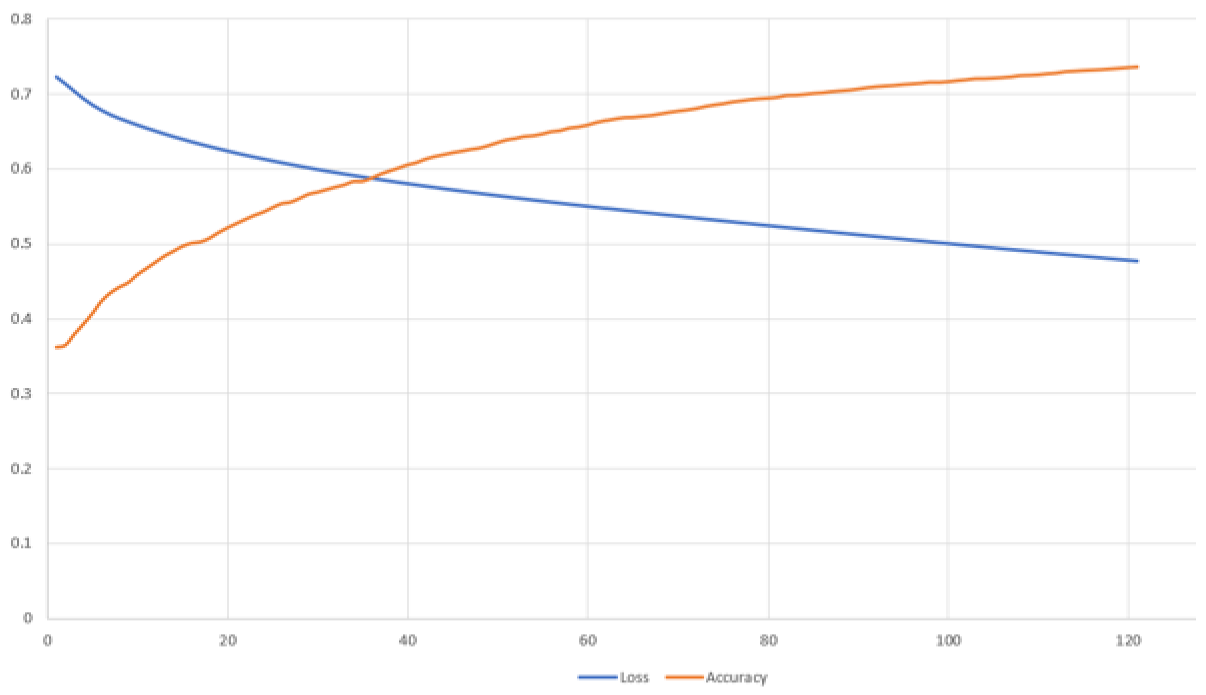
Flower Predictions :

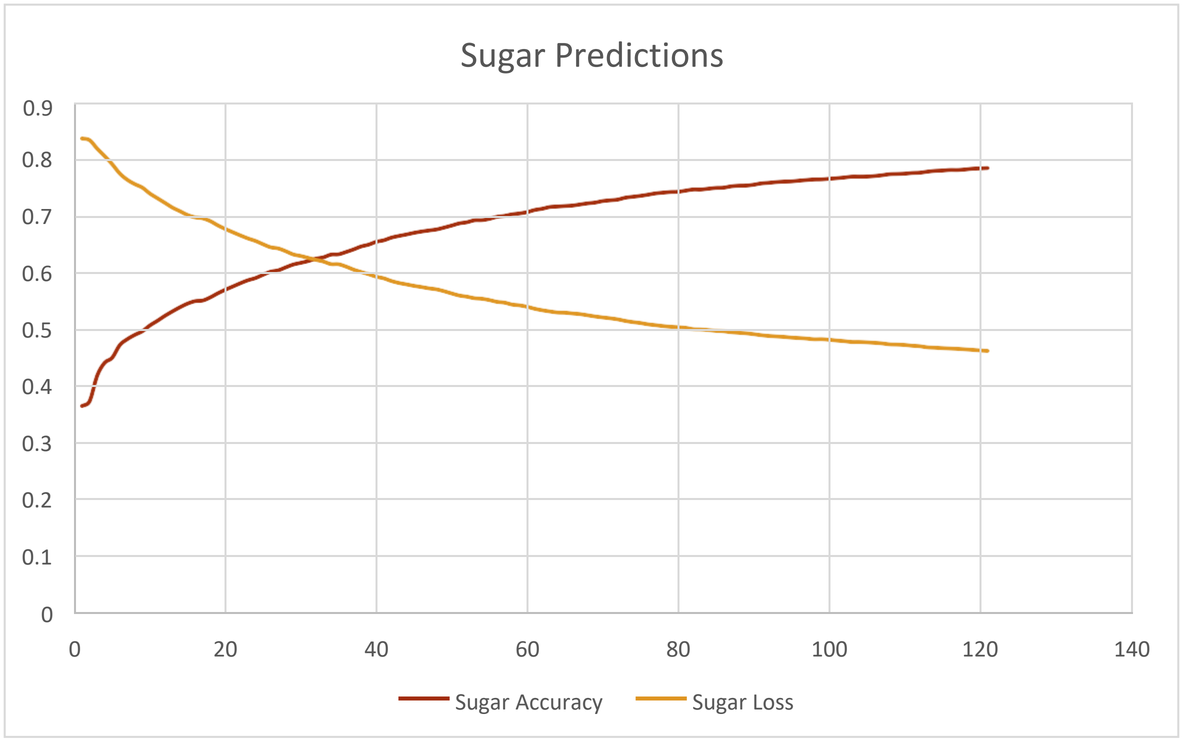


Gravel Predictions :



Fish Predictions :





CONCLUSION:

I Ire successful in finding out most of the clouds. Higher correlated clouds Ire seemingly difficult but with higher epochs got cleared out. CNN still relies much on similar structure within similar labels and it becomes tough to train the samples having slightly different structures having same label. Low resolution data clearly had an impact on my accuracy as it involved less features but on also took longer to process. The segmentation could have been easier and more accurate had I received more accurate masks. The masks Ire mostly generalized and lack a formal shape as Ill. As per Kaggle, output should involve run length encoding of the masks which I didn't apply here. According to them, getting 25% accurate test data is set as qualified. I `can therefore say that I Ire qualified as I predicted and looked on each sample. Unfortunately, it was not possible to show the entire prediction in the presentation or the report. The testing model had huge size which also took time and resmyces to run.

FUTURE WORK:

Currently, my final shot would be to convert my predicted masks in form on run length encoding in order to compress the output.

REFERENCES:

1. https://www.hellocodings.com/2017/10/convolution-neural-network-i.html
2. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
3. https://towardsdatascience.com/image-classification-python-keras-tutorial-kaggle-challenge-45a6332a58b8?gi=e8eb060142f3
4. http://deeplearning.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/
5. https://towardsdatascience.com/u-net-b229b32b4a71