IEEE CS Internship

Team Members: Ankitha C
Hita Juneja
Gagan Goutham

Team Mentor : Dr. Venugopal N

Week 1

Literature Review + Dataset Overview

- Over 10 Research Papers and Case study journals related to the Mars Rover Image Classification were studied and analysed.
- We obtained 3 major datasets with Martian Rover Images.
- 1. Curiosity Rover dataset: 6691 Raw images.

  2. Opportunity + Spirit Rover dataset: Over 350k Raw Images.

  3. AI4Mars: Over 35k Raw Images.
     Labelled through Crowdsourcing.
- After going through these datasets thoroughly, we have considered using AI4Mars images for the dataset preparation.
- Many methods/techniques such as SVMs, Fuzzy Rough Feature Selection, RNN, U Net, Vision Transformers, Deep Dilated Networks to be considered to study.

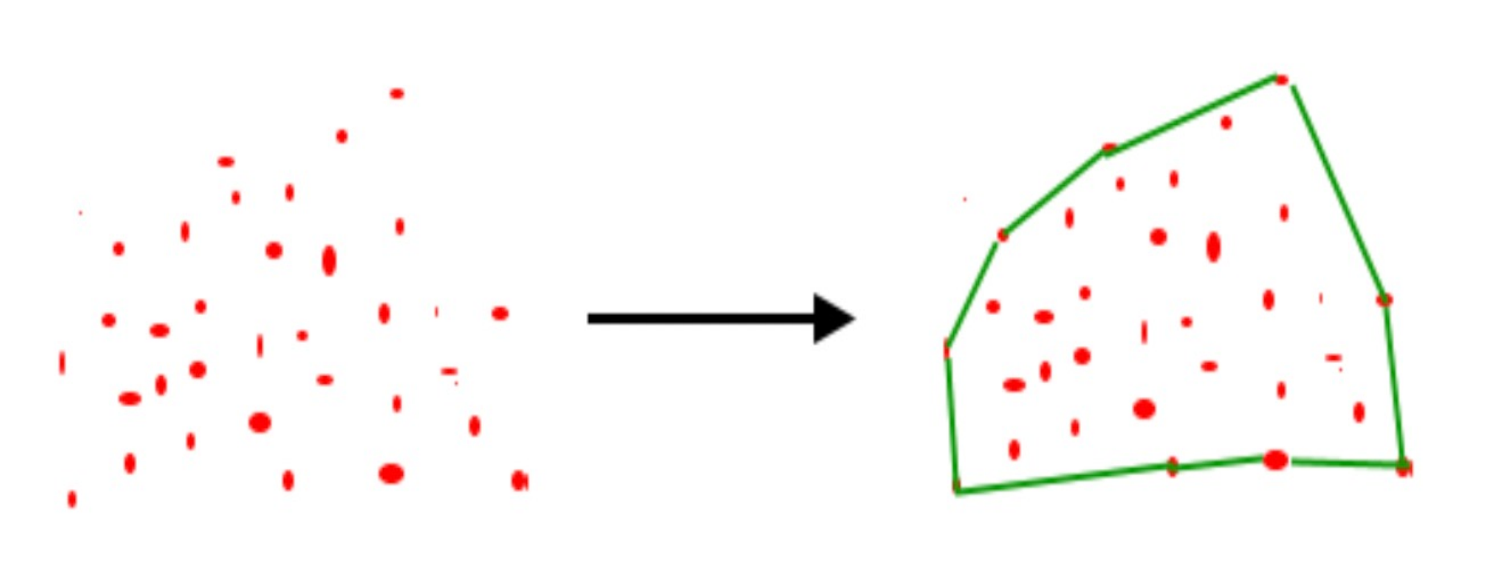
Week 2

Data Pre-processing

- This was identified as unsupervised learning. As a result, we attempted to use clustering methods to see what the raw images could give as an output.
- K-means clustering was implemented on 100 randomly sampled images. Value of k was varied(3,5,10,41,100), in order to observe the difference in image texture for different value of k.
- Contour detection(edges) and then Convex hull methods were used on the output image to identify different textures of the surface.
- We manually tried to analyze if the final output images have been identified with corresponding textures of the raw image. 49 out of 100 images were accurate enough.

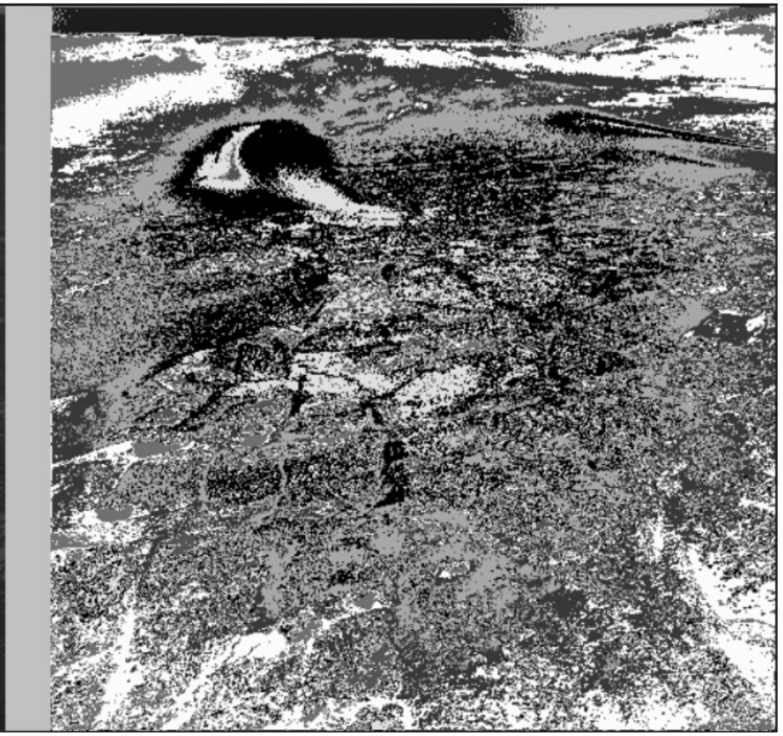
K means + Convex Hull

- Given **N** a set of points in the plane. the convex hull of the set is the smallest convex polygon that contains all the points of it.
- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters.

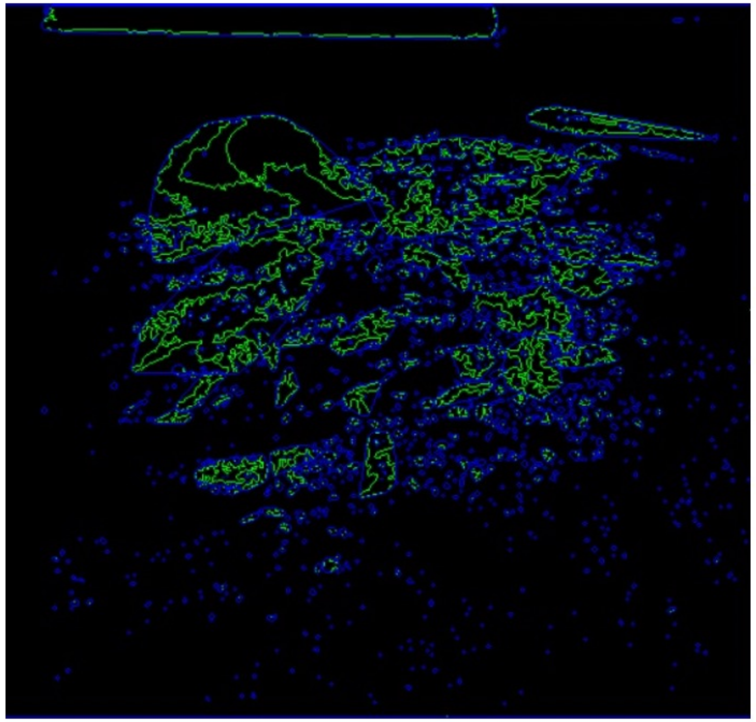


Output

RAW Image



K means output (k=100)



Final output - Green patches show the areas which are mostly sand

Contoured k means output

Observation

- We realized that a lot of images have the parts of the rovers which make it difficult for the clustering methods to accurately segment the objects.
- We also realized we have to perform data cleaning, by normalizing images, brightening it before feeding into model so that objects and terrain is accurately detected.
- We explored for sources on image data cleaning, but didn’t find any useful resources.

Week 3

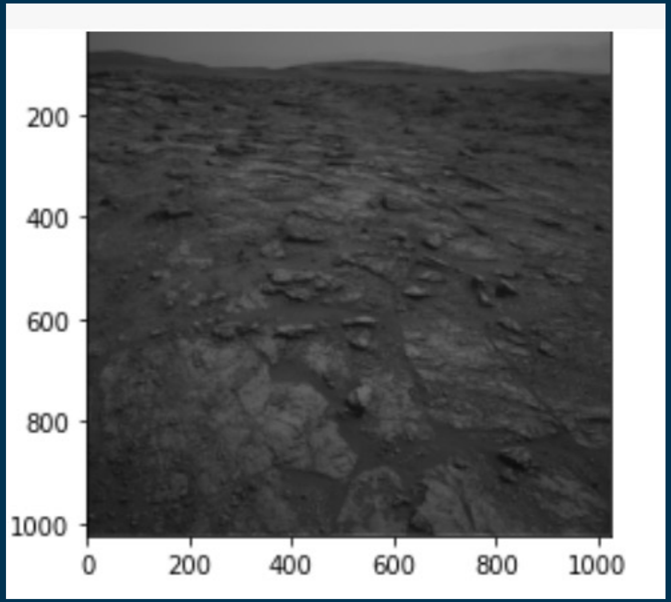
Data Cleaning

- The dataset has many unwanted images which are either overexposed or the image is mostly covered by the rover.
- It makes sense to remove such images and clean the dataset.
- We opted 2 methods to clean the data - Object Detection and Masking having a greater threshold of rover, sky, horizon components in them.

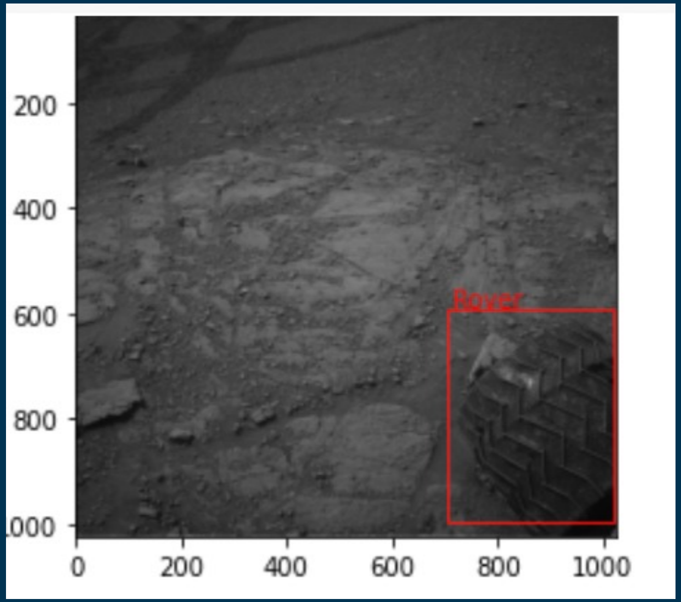
Object Detection - Detecting the rover

- We read 4 research papers focused on labelling of images, detailed working and information about the AI4MARS Dataset, and detection of objects/robots.
- We took about 100 images to manually label using an online labeling tool (makesense.ai), and also generated annotation files for each labelled image in xml, csv form.
- Used detecto python package for implementation.
- Used the Ran into many configuration errors, fixed them, and finally implemented.
- The logic was it discard images which have the area of the bounding box, when rover is detected, greater than 1/4th of the size of the image.
- Overall loss 10% in 20 epochs.
- Object Detection Images: ~ 10,000

Output



200
400
600
800
1000
0
200
400
600
800
1000



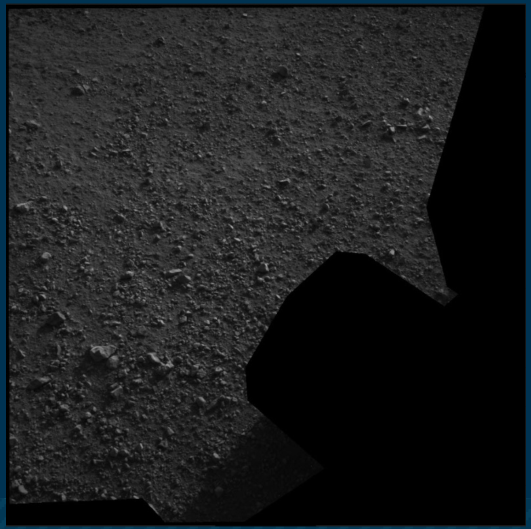
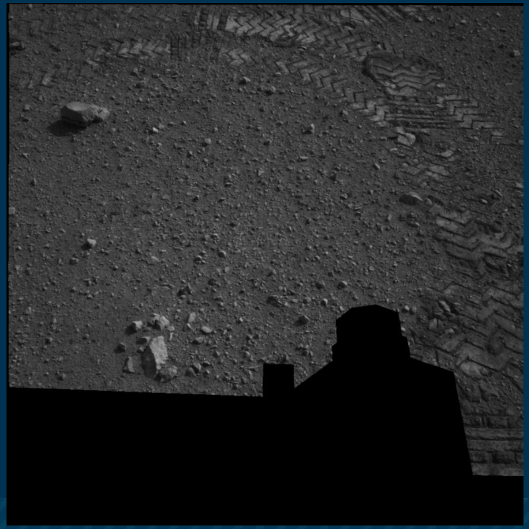
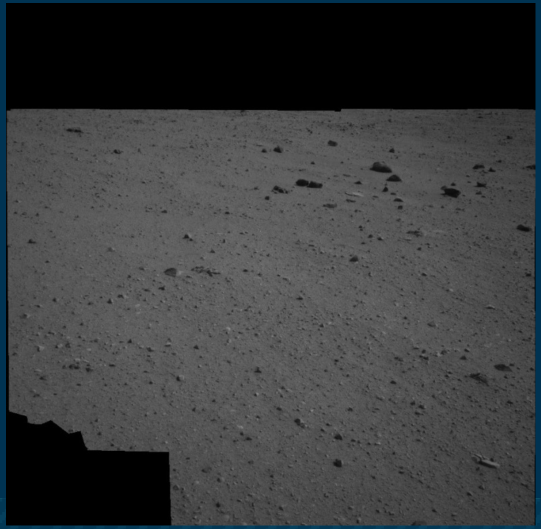
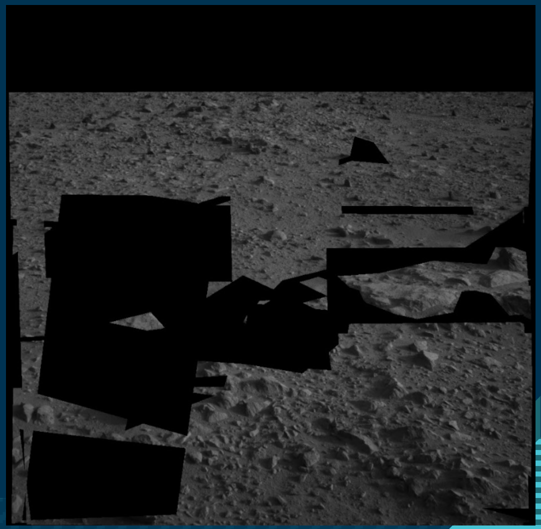
200
400
600
800
Rover
0
200
400
600
800
1000

Week 4

Masking the unwanted regions

- The original AI4Mars dataset has another folder which contains the masked images for the respective raw images.
- A Raw image was merged with the corresponding Masked image.
- This resulted in an image where rover and sky were masked as black.

Output

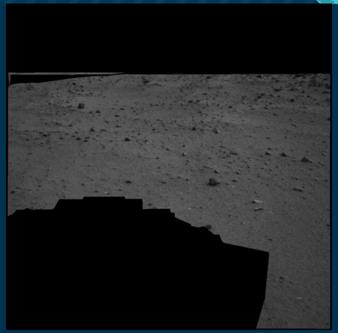
Final Datasets

- The data cleaning processing was challenging for us as we had to deal with nearly 18k images.
- Since we did not have high computational resource, it was a major issue for us to pass 18k images in loops to clean the data.
- Thereby, the entire dataset was divided into 4 batches, where each batch consisted of approximately 4,500 images.
- Involved primarily 4 stages – Object detection, Masking, Resizing, Labeling.
- Further, the process was carried out in batches. Eventually, each batch output was combined to form the final dataset.

Pre-Processing Steps

Raw Image

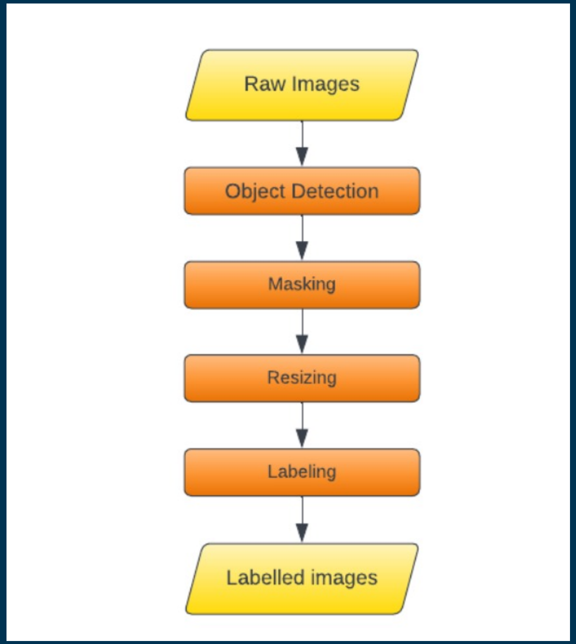
Mask



Masked Output

Resizing

Labeling



Week 5

Multilabel Image Classification

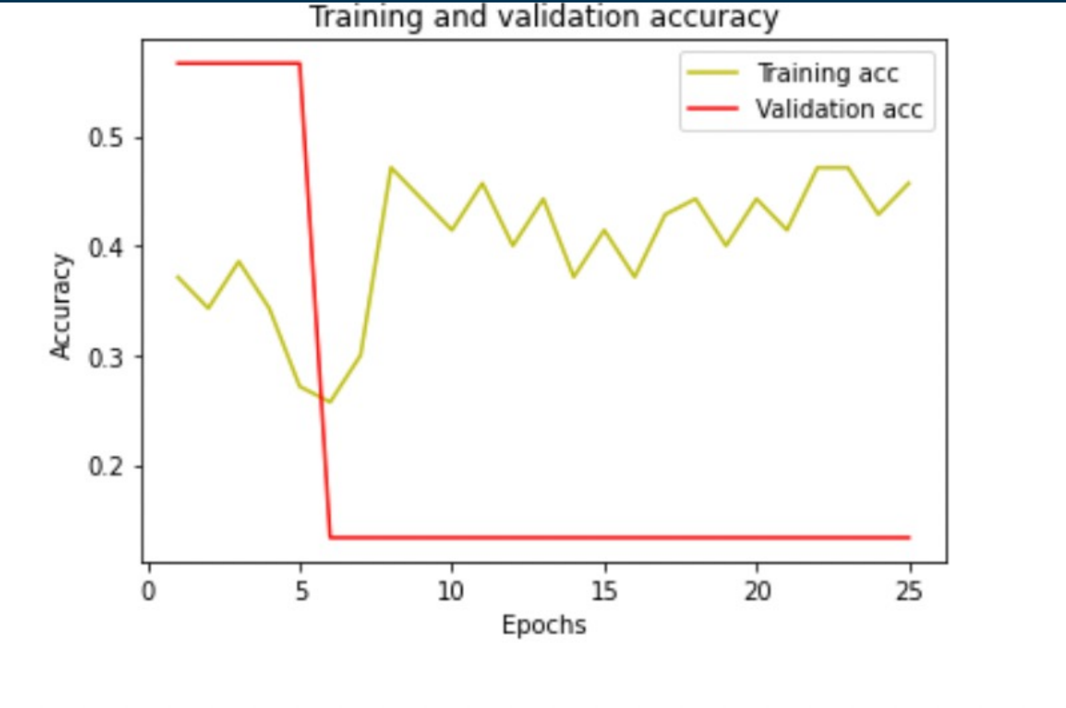
- Each image contains parts belonging to at least 1 class (out of a minimum of 3 – soil, sand, rock)
- Manually labelled 100 images in csv file.

- We used a sequential CNN with Relu activation because each classifying to each label is a binary cross entropy function - either that image belongs to that particular label(1) or not(0).

```
                                  Image  Soil   Sand  Bedrock
0  NLA_397681520EDR_F0020000AUT_04096M1.jpg     1      0        1
1  NLA_397681893EDR_F0020000AUT_04096M1.jpg     1      1        0
2  NLA_398919855EDR_F0030078NCAM00303M1.jpg     0      1        1
3  NLA_398920122EDR_F0030078NCAM00303M1.jpg     1      0        1
4  NLA_399365236EDR_F0030100NCAM00403M1.jpg     0      0        1
Index(['Image', 'Soil ', 'Sand', 'Bedrock'], dtype='object')
```

- Code

Model Results

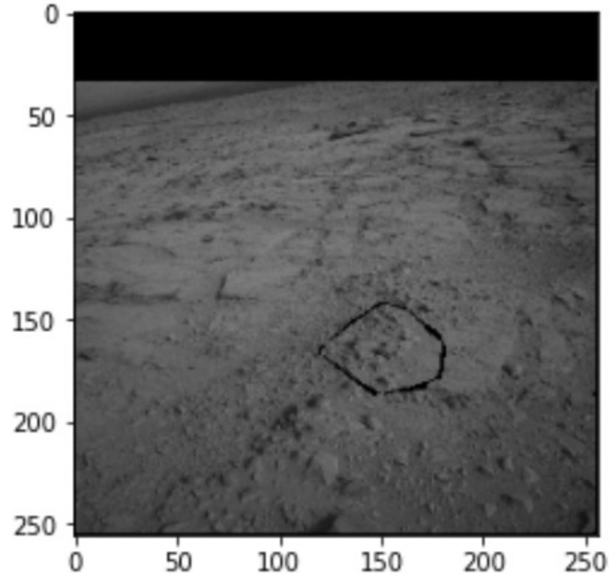


- Probability of picture belonging to either of 3 classes ->
- Accuracy – 13.33%

```
_, acc = model.evaluate(X_test, y_test)
print("Accuracy = ", (acc * 100.0), "%")
```

```
1/1 [==============================] - 1s 711ms/step - loss: 0.6725 - accuracy: 0.1333
Accuracy =  13.333334028720856 %
```

```
Soil 0.7541554
Sand 0.32735533
Bedrock 0.8767582
```



Week 6

Implementation of the Model

- Following a week of intensive research, it was decided to attempt **multi-label image segmentation** on the outputs of masked images using a transformer-based model called **SegFormer**.
- The model needs a corresponding label file, which represents each pixel value with regard to multiple labels, in order to be trained. Almost **10%** of the masked output images were manually chosen in the classes of Sand, Soil, and Rock. **932 images** were included in the final sample size as a result. With the help of the versatile segmentation tool "**Segments.ai**," these photos were given labels. Before the labeling procedure, a few pre-processing steps including resizing and normalizing were performed.
- Further, after feature extraction from the images and labels, **'SegFormer'** was implemented. SegFormer is a powerful semantic segmentation framework which comprises hierarchically structured **Transformer encoder** and **Multilayer perceptron (MLP) decoders**, pre-trained on ImageNet-1k. We have deployed multiple series of the SegFormer. **B0, B1 and B2** are the 3 variants that have been implemented. Each variant of the model differs in the parameters, hidden sizes, depths of the transformer encoder backbone introduced in SegFormer.

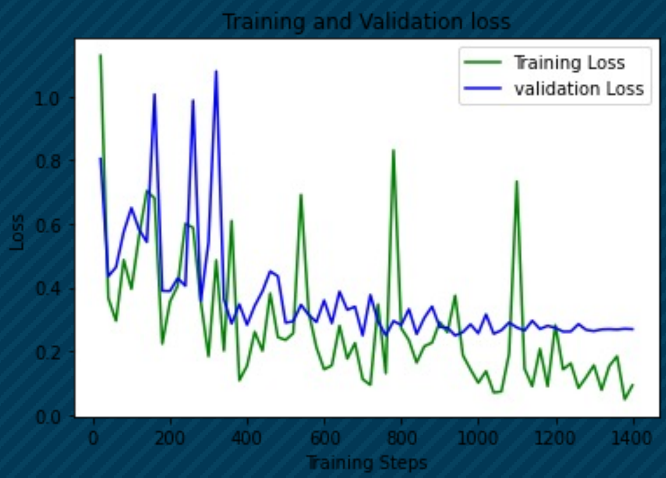
Week 7

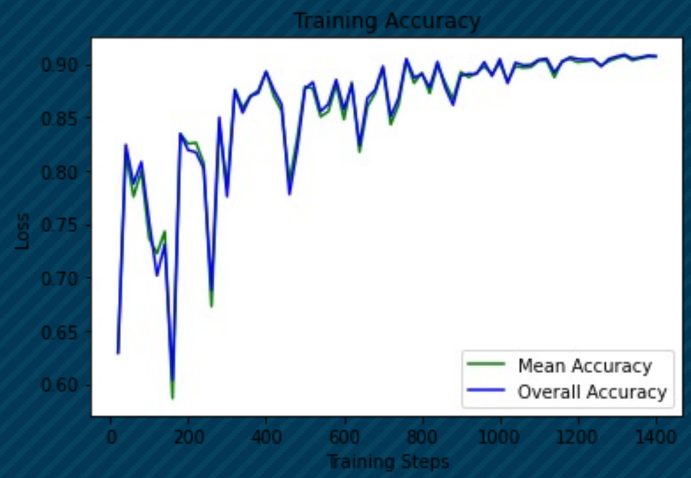
Plotting Model Results

The training set contained 746 photos, and the testing set contained 186 images after the dataset was split in half in the ratio 80:20 for the train and test sets, respectively. Using the Pytorch framework, we implement the Segformer - B0, B1 and B2 fine tuned on ADE20k. With a **batch size of 8** and a **learning rate of 0.0006**, we train our network over **15 epochs** covering **1400 steps**.

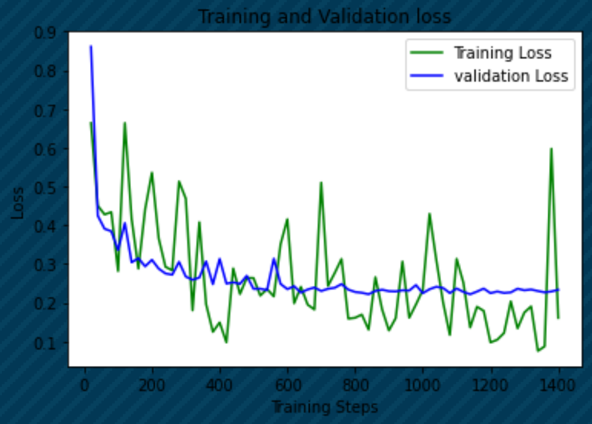
The results of the models were carefully analyzed.

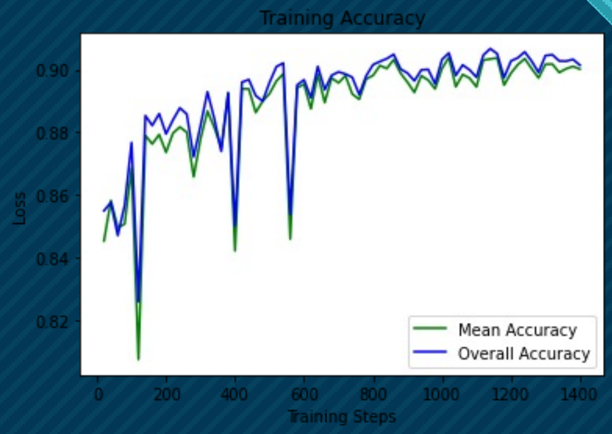
SegFormer - B0



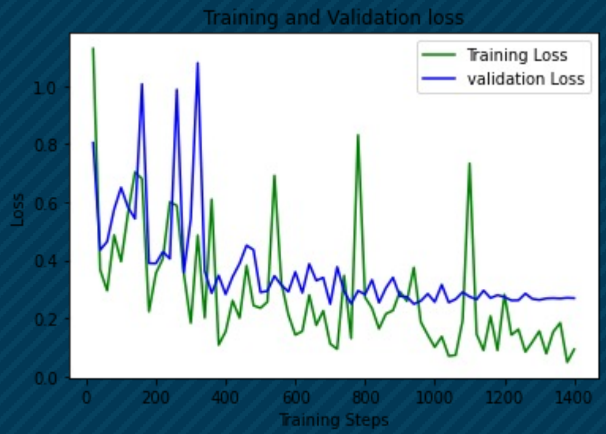


SegFormer - B1





SegFormer - B2



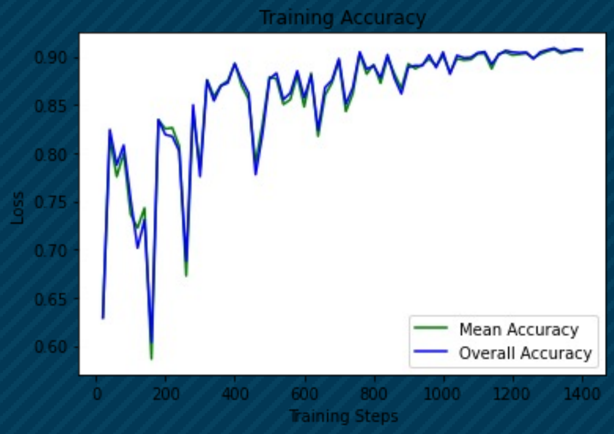


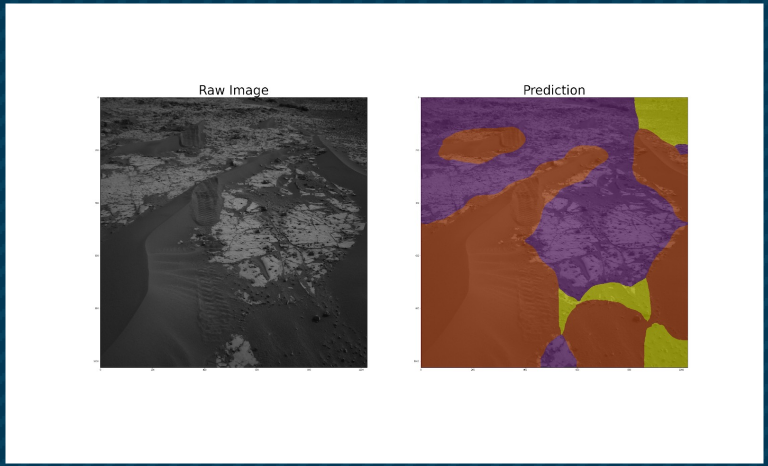
TABLE I: Final Results

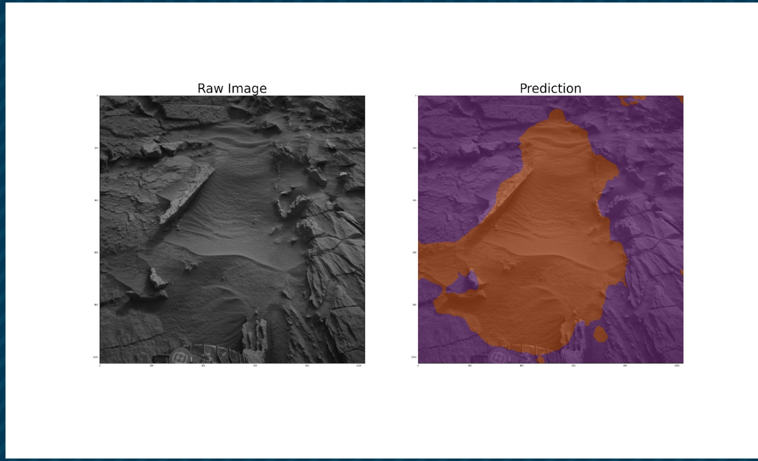
Model	Mean IoU	Mean Accuracy	Overall Accuracy
SegFormer - B0	81.15%	88.61%	89.92%
SegFormer - B1	83.02%	90.33%	90.66%
SegFormer - B2	83.55%	90.75%	90.86%

TABLE II: Per Class Results of SegFormer - B2

Class	Per Class IoU	Per Class Accuracy
Sand	85.67%	90.65%
Soil	79.35%	89.06%
Rock	85.64%	92.54%

Final Segmented Outputs:

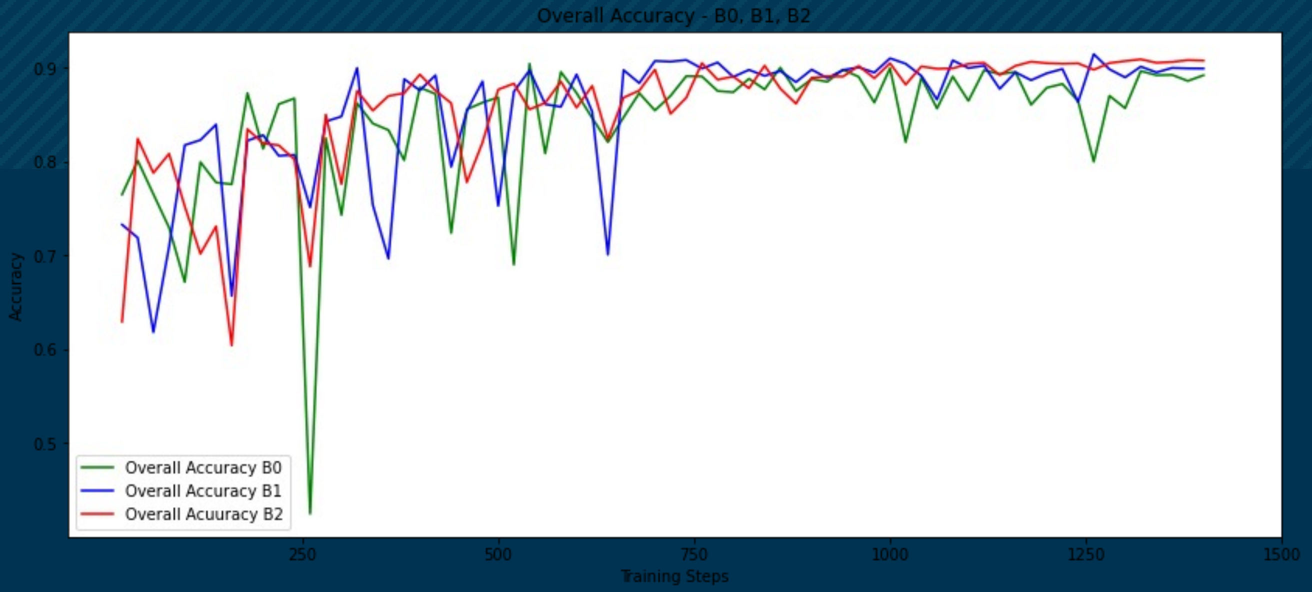


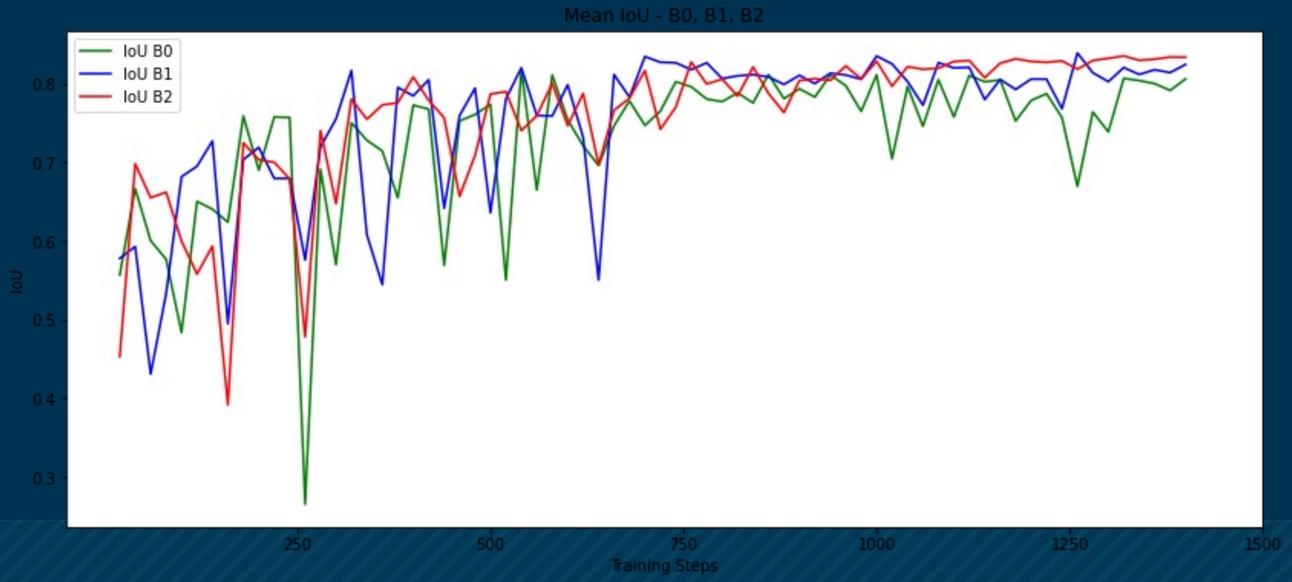


Week 8

Comparative Analysis

- Variations of the models, such as the SegFormer B1 and B2 models, are currently being deployed to perform a comparison-based research between the SegFormer versions for this problem statement. The model training uses parameters and epochs that are comparable to those of the B0 model.
- It was concurrently organized, systematically recorded, and examined in order to write a research report on the complete project. A draft paper that examines the application of the SegFormer-B0 model is being finished. The training and validation loss from the B0, B1, and B2 models are being compared. We have made an effort to examine the results and create a structured research paper from the comparison of the three variants. The paper is being completed to be submitted for a conference in the coming weeks.
- Following plots compare the **Mean IoU** and **Overall Accuracy** of B0, B1, B2 versions of the SegFormer. It is observed that B2 marginally is the highest in both, followed by B1 and B0.


Overall Accuracy - B0, B1, B2
Accuracy
Training Steps
Overall Accuracy B0
Overall Accuracy B1
Overall Acuuracy B2
0.9
0.8
0.7
0.6
0.5
250
500
750
1000
1250
1500


Mean IoU - B0, B1, B2
IoU
Training Steps
IoU B0
IoU B1
IoU B2
0.8
0.7
0.6
0.5
0.4
0.3
250
500
750
1000
1250
1500