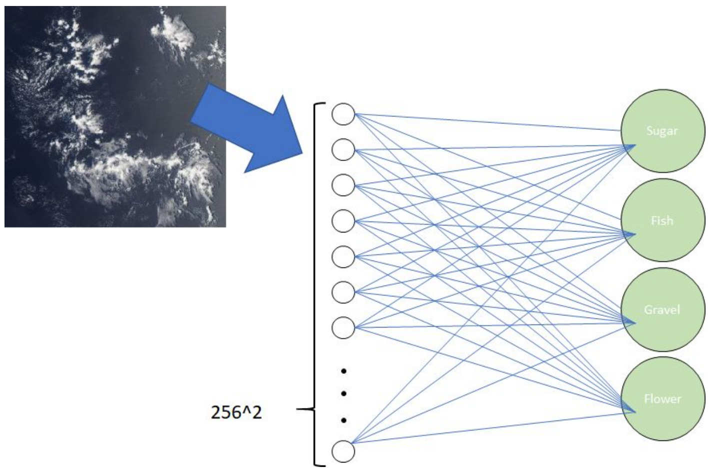
Neural network:

Naomi Tsabari 207642638
Shai Aharon 301206967
github link:https://github.com/ifryed/NLP_course/tree/Task_2

1. Simple Perceptron (Single layer neural network)



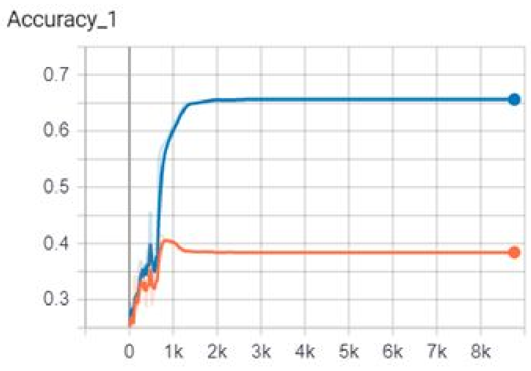
We used a softmax to train the model.

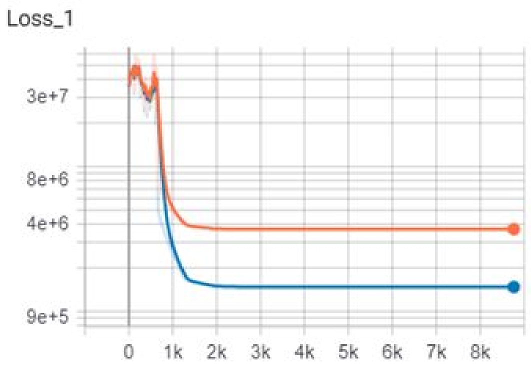
Baseline: Since we have 4 labels, the random guess accuracy is 25%.

Our first trail gave us:

Train:	65% Accuracy
Test:	40.49% Accuracy

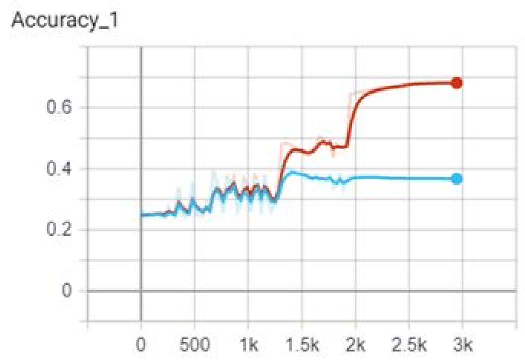
Preprocess:
The only pre-process we did was converted the image from RGB to grayscale. We figured that most of the image is on the range between blue and white, by converting it to grayscale we lose very little information, and gain a weight reduction by 3 times.
From $3 \cdot 256^2 \cdot 4 \rightarrow 256^2 \cdot 4$

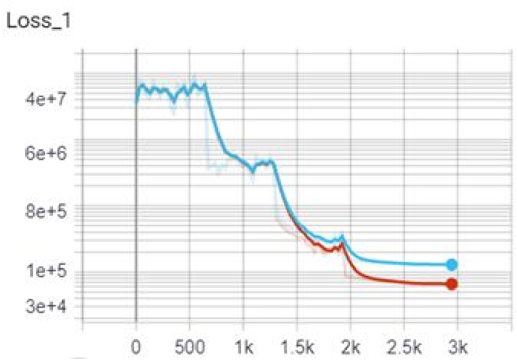




We tried to add regularization, but since the model didn't overfit on the train we were not counting on it to work.
The result:

Train :	68% Accuracy
Test:	36% Accuracy





We hypothesize that the reason for the rise of the train score, was that the regulation prevented the model from going into local maximum, that in-turn caused a bigger over-fit, which led to a lower test score. Since the input is an image, if the regularizer “turns off” some weights, that means that if found out that some pixels don’t help the classification, but since it’s a natural image, we don’t know that those same pixels won’t be helpful in the test as in the train, hence the ‘Overfit’.

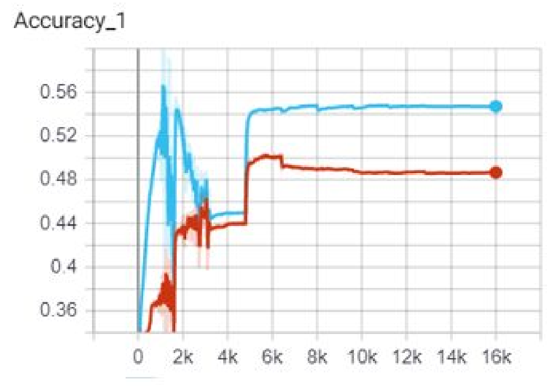
L1 and L2 was tried, L1 gave slightly better results but still lesser than no regullazation at all.

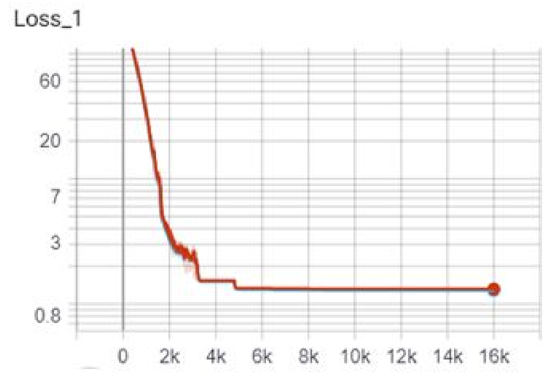
Next, we tried to pre-process the data. We figured that if we convert the image to a binary image, it will help the model learn better. We tried a few different thresholds, the best one was 127 (The image range is [0,255])

The result:

- Train : 54% Accuracy
- Test: 50% Accuracy

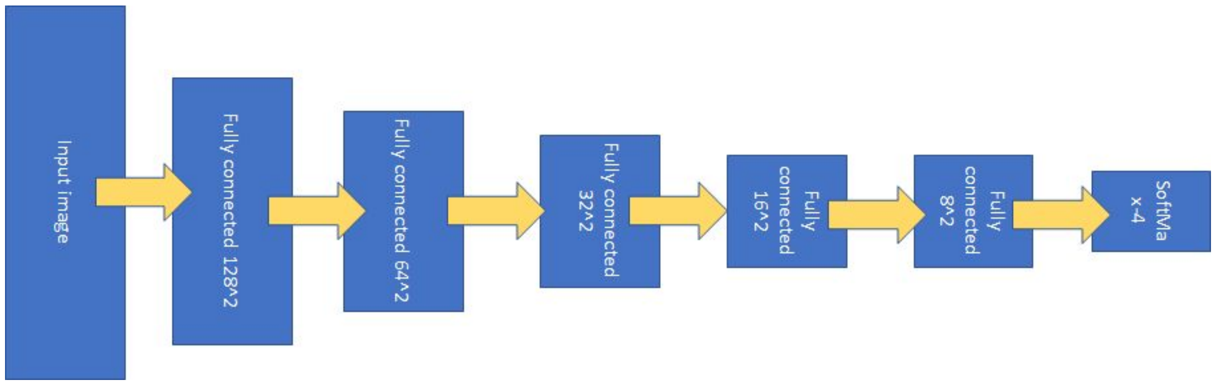
* L1 regularizer was used



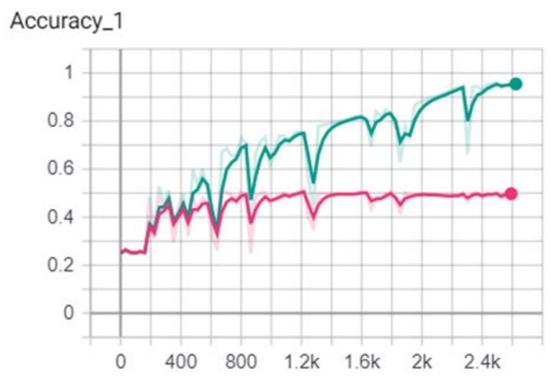


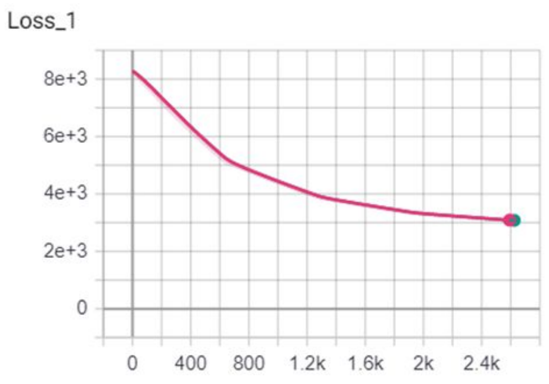
2. ANN (Multi-layer perceptron)

We built a deep NN as follows:

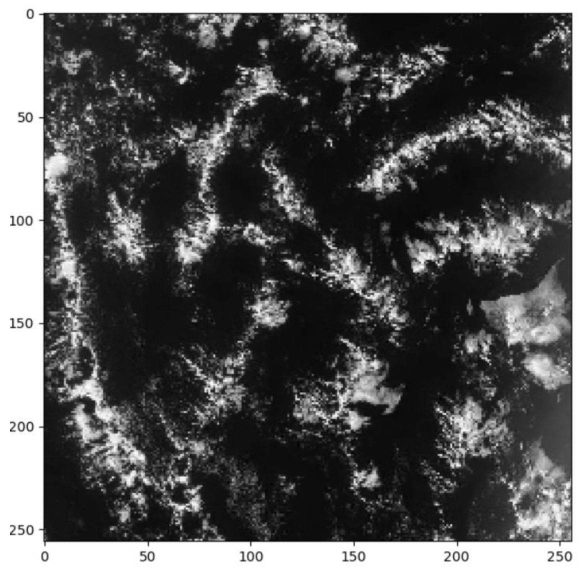


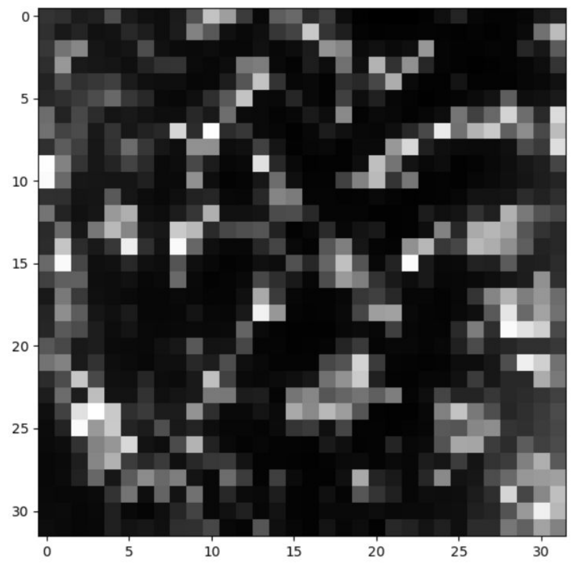
Since the input (256^2) was very big, the net had a lot of parameters to fit. This cause the trials to be very long. We tried all sorts of models for the net, with smaller fully connected layers and larger, deeper nets, and shallower. But we could not beat the 50% on the test. It usually over fitted on the train, while the test never passed the 50%.





Due to computer limitations, we could not run larger nets, so our only option was to resize the input image. We refrained from doing so in the beginning, since we were afraid that we would lose too important information, but after we reached a dead end, we decided to try it out.
We resized the image from 256x256 to 32x32. The main formation of the clouds still remains in tact.

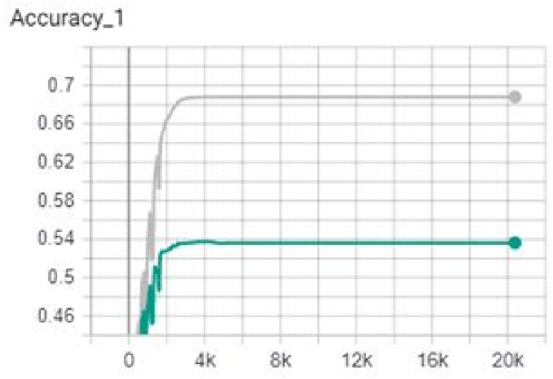


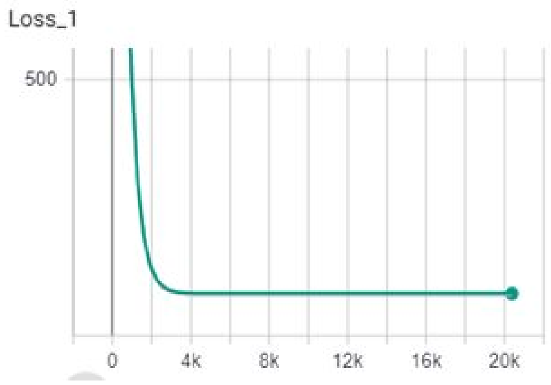


We used the net with the dense layers with the following sizes: 128->64->64->32->16

The result:

- Train : 67% Accuracy
- Test: 53% Accuracy





Conclusion:
Although we weren't surprised to see that the **multi-layer** neural network outperformed the **signal layer** network, we did expect it to perform significantly better, and not but a mere 2-3%. We blame the structure of the network (fully connected), that does not take advantage of the image properties, such as that neighbor pixels in the image have mutual information and a higher correlation then non neighbor pixels. Such information could have saved a lot of weights, and made the NN smaller and easier to train. Even when we re-sized the image to 32x32, the model took a long time to learn and did not achieve significant results. Since the task is to learn clouds patterns, the focus of the net should be spatial, while the fully-connected structure cannot learn spatial information. We expect that a Convolutional Network, which is built to incorporate spatial information will perform significantly better.