PES UNIVERSITY

A mini project report on

"Multilabel Image classification of Cloud Images using Contractive Autoencoder"

Bachelor of Technology
in
Computer Science & Engineering

Submitted by :

01FB16ECS187 : Maanvi Nunna
01FB16ECS232 : Ninaad R. Rao
01FB16ECS236 : Nishant Ravi Shankar

August – December 2019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

Multi-label Cloud Formations Classification using Contractive Autoencoders

About the dataset

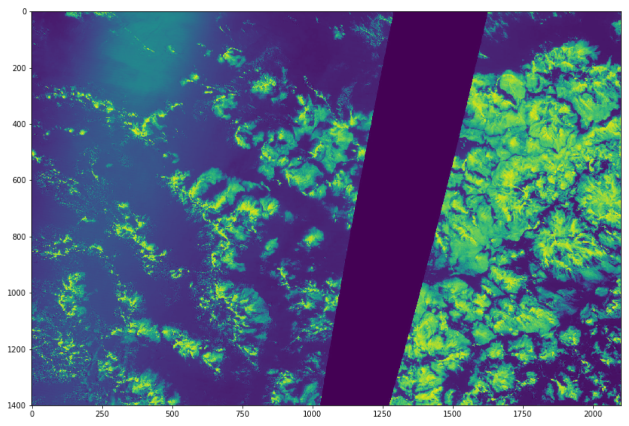
The images were downloaded from NASA Worldview. Three regions, spanning 21 degrees longitude and 14 degrees latitude, were chosen. The true-color images were taken from two polar-orbiting satellites, TERRA and AQUA, each of which passes a specific region once a day. Due to the small footprint of the imager (MODIS) onboard these satellites, an image might be stitched together from two orbits. The remaining area, which has not been covered by two succeeding orbits, is marked black.

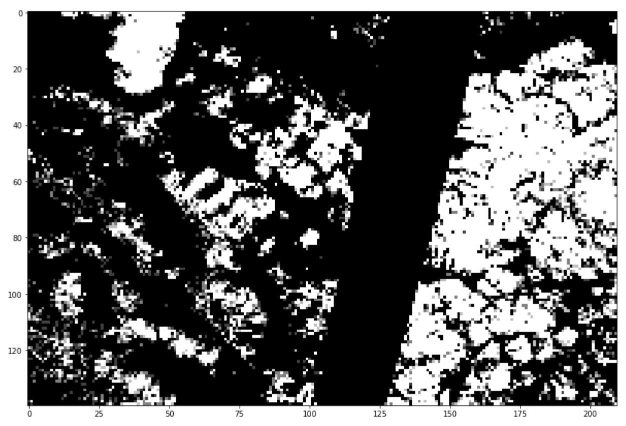
The labels were created in a crowd-sourcing activity at the Max-Planck-Institute for Meteorology in Hamburg, Germany, and the Laboratoire de météorologie dynamique in Paris, France. A team of 68 scientists identified areas of cloud patterns in each image, and each image was labeled by approximately 3 different scientists. Ground truth was determined by the union of the areas marked by all labelers for that image, after removing any black band area from the areas.

The dataset is a collection of satellite images that contain certain cloud formations - Fish, Flower, Gravel, Sugar. Each image has at least one cloud formations, and some could contain all of them. The size of the entire dataset is 5546, which has been split into 4500 for training, and the remaining for testing.

Preprocessing

In order to make sure that our model is trainable and moderately accurate, the images were resized from 2100x1400 to 210x140. After this, the images were converted to black and white images using the OTSU binarization technique.
The pixel values were normalized for faster computation. Since the classification of the images were not very obvious, instead of finding the bounding box of the classes, we found whether a particular class(of the four classes) are present in the image or not. This made it a multilabel classification problem. Here is a depiction of Original vs. Preprocessed Image





Environment

For training all the models, Kaggle was used. It has a P100 GPU with 30hours per week limit with 13GB RAM and 5GB Hard disk space. The contractive autoencoder code was written in Tensorflow and the feedforward neural network and CNN were written using Keras.

Contractive Autoencoder

Contractive Autoencoders are regularization autoencoders, with the goal of creating encodings that are less susceptible to variations in the training data. This is achieved by adding a penalty or a regularization term to the traditional reconstruction loss function. As described in [1], the loss function to be minimized can be written as:

$$\mathcal{J}_{\mathrm{CAE}}(\theta) = \sum_{x \in D_n} \left(L(x, g(f(x))) + \lambda \|J_f(x)\|_F^2 \right)$$

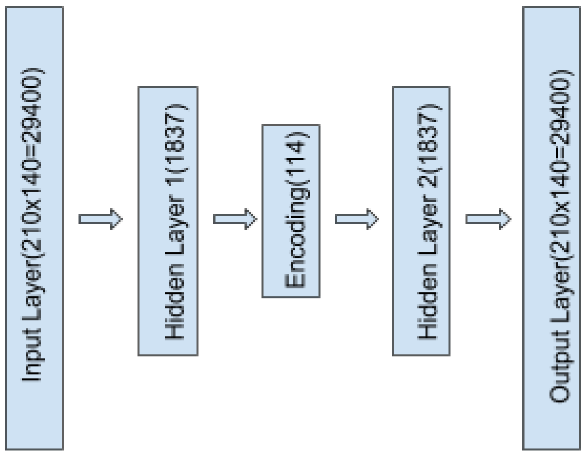
Intuitively, the motivation behind the loss function is to learn only the important features of the input data, which is similar to the objective of Principle Component Analysis. We can break down this loss function into two parts:

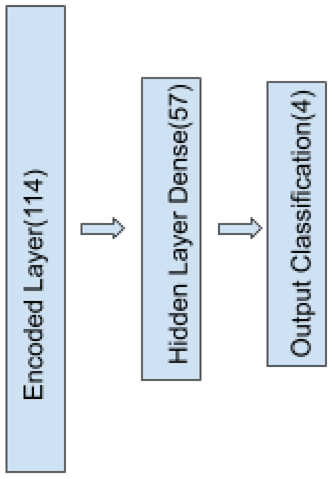
1) Reconstruction Loss: This is the traditional loss function of an auto-encoder.
2) Frobenius Norm of the Jacobian Matrix: This is a regularization term that aims to only learn the important features of the training data. A jacobian matrix is obtained by the differentiation between two vectors, which in this case is the encoder activations and the input, where each term of the matrix is a partial derivative between the jth activation and the ith input to the encoding layer.

$$\|J_f(x)\|_F^2 = \sum_{ij} \left(\frac{\partial h_j(x)}{\partial x_i} \right)^2 .$$

In an ideal scenario, when the derivative becomes zero, it means that the hidden layer output is not dependent on the input at all. This term, when used in conjunction with the reconstruction loss ensures that only the robust features of the training data are learned and the small variations are ignored.

Architecture Used For Contractive Autoencoder:



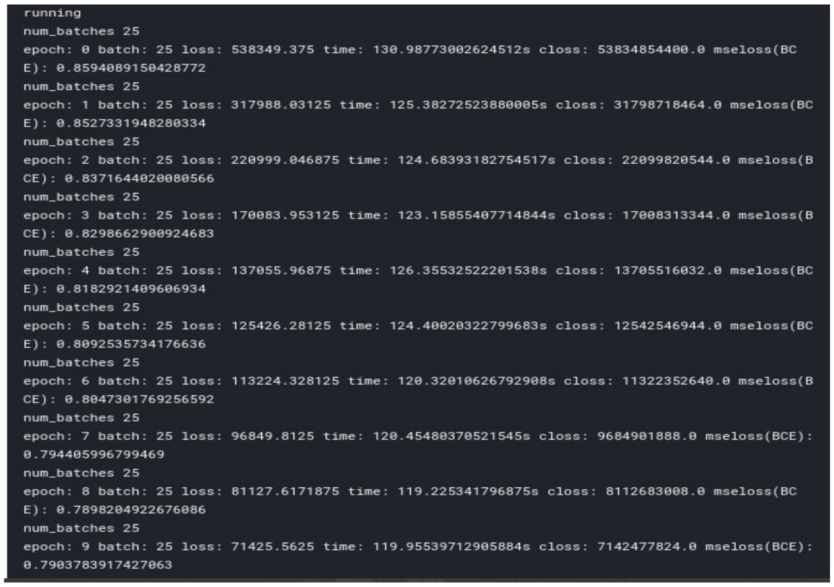


Feed Forward Architecture

All the layers except the output layer had a ReLU activation unit. Since the contractive autoencoder predicts if a given pixel value is a 0 or 1(for all the 29400 pixels), sigmoid activation function was used in the final layer. The error metric was binary cross-entropy. The main reason for using binary cross-entropy was because the gradient of the BCE error is directly proportional to the update made at that point.

- For training the contractive auto-encoder, Adam optimizer was used so that the effects of both momentum and RMSProp is considered.
- Lambda's value for the Frobenius norm of the Jacobian Matrix in the error term was 10^-5(experimental).
- The model was trained for 100 epochs and each epoch took about 5 minutes to run. The model took around 6 hours to train

Training Results:



```
running
num_batches 25
epoch: 0 batch: 25 loss: 538349.375 time: 130.98773002624512s closs: 53834854400.0 mseloss(BC
E): 0.8594089150428772
num_batches 25
epoch: 1 batch: 25 loss: 317988.03125 time: 125.38272523880005s closs: 31798718464.0 mseloss(BC
E): 0.8527331948280334
num_batches 25
epoch: 2 batch: 25 loss: 220999.046875 time: 124.68393182754517s closs: 22099820544.0 mseloss(B
CE): 0.8371644020080566
num_batches 25
epoch: 3 batch: 25 loss: 170083.953125 time: 123.15855407714844s closs: 17008313344.0 mseloss(B
CE): 0.8298662900924683
num_batches 25
epoch: 4 batch: 25 loss: 137055.96875 time: 126.35532522201538s closs: 13705516032.0 mseloss(BC
E): 0.8182921409606934
num_batches 25
epoch: 5 batch: 25 loss: 125426.28125 time: 124.40020322799683s closs: 12542546944.0 mseloss(BC
E): 0.8092535734176636
num_batches 25
epoch: 6 batch: 25 loss: 113224.328125 time: 120.32010626792908s closs: 11322352640.0 mseloss(B
CE): 0.8047301769256592
num_batches 25
epoch: 7 batch: 25 loss: 96849.8125 time: 120.45480370521545s closs: 9684901888.0 mseloss(BCE):
0.794405996799469
num_batches 25
epoch: 8 batch: 25 loss: 81127.6171875 time: 119.225341796875s closs: 8112683008.0 mseloss(BC
E): 0.7898204922676086
num_batches 25
epoch: 9 batch: 25 loss: 71425.5625 time: 119.95539712905884s closs: 7142477824.0 mseloss(BCE):
0.7903783917427063
```

A fully connected feedforward neural network was used. The input to this was the encoding of the contractive autoencoder. The activation units were ReLU(except the final output layer) and Sigmoid for the output Multi-Label classification. Adam Optimizer was used to train the model. Batch normalization and Dropout layer was added. This model had 4 nodes in the output layer indicating the four classes(Fish, Flower, Gravel, Sugar). An 80-20 split was done.

Metrics for Evaluating the model:

For a multilabel classification, just measuring the accuracy is not sufficient. The metrics that were used were:
Accuracy: Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. It refers to the closeness of the measurements to a specific value. A simple comparison between the actual and the predicted value gives us the accuracy of the model.

Precision: Precision is the ratio of correctly predicted positive observations of the total predicted positive observations.
Precision = TP/(TP+FP)

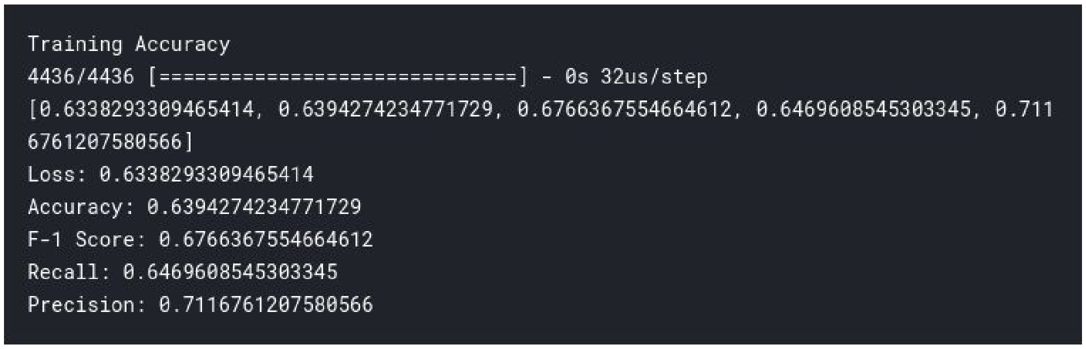
Recall: Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.
Recall = TP/TP+FN

F1-score: F1 Score is the Harmonic mean of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have a similar cost. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall.

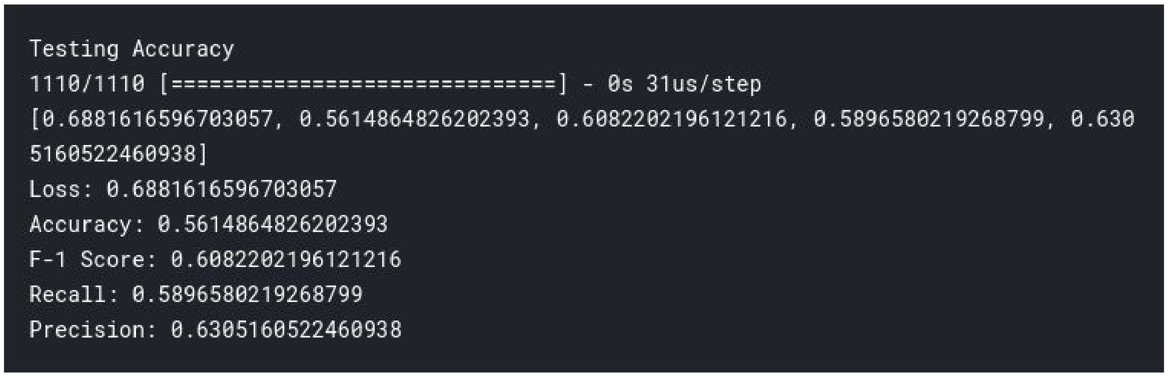
F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Results

Two models were designed for this dataset. One was with a contractive autoencoder with a feed-forward neural network that took around 8 hours of computation power on the environment specified. The other model was a Convolutional Neural Network(CNN) with a feed-forward neural network. Both the models’ performances are very close and further training can improve results.
The results for the contractive autoencoder with a feed-forward neural network are as follows:

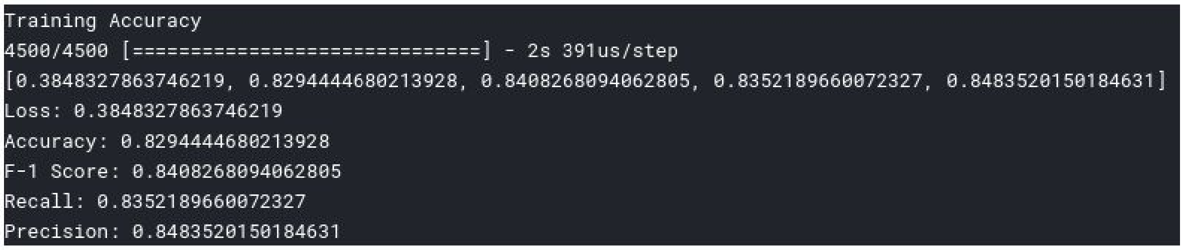


```
Training Accuracy
4436/4436 [==============================] - 0s 32us/step
[0.6338293309465414, 0.6394274234771729, 0.6766367554664612, 0.6469608545303345, 0.711
6761207580566]
Loss: 0.6338293309465414
Accuracy: 0.6394274234771729
F-1 Score: 0.6766367554664612
Recall: 0.6469608545303345
Precision: 0.7116761207580566
```

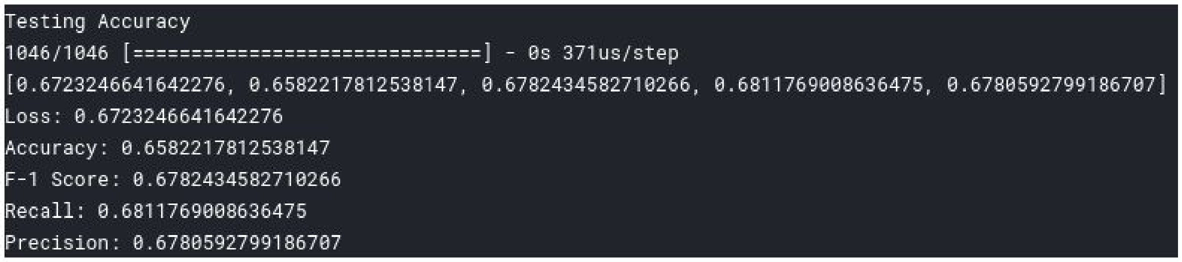


```
Testing Accuracy
1110/1110 [==============================] - 0s 31us/step
[0.6881616596703057, 0.5614864826202393, 0.6082202196121216, 0.5896580219268799, 0.630
5160522460938]
Loss: 0.6881616596703057
Accuracy: 0.5614864826202393
F-1 Score: 0.6082202196121216
Recall: 0.5896580219268799
Precision: 0.6305160522460938
```

The results for the Convolutional Neural Network with the feed-forward network for multilabel classification are as follows:



```
Training Accuracy
4500/4500 [==============================] - 2s 391us/step
[0.3848327863746219, 0.8294444680213928, 0.8408268094062805, 0.8352189660072327, 0.8483520150184631]
Loss: 0.3848327863746219
Accuracy: 0.8294444680213928
F-1 Score: 0.8408268094062805
Recall: 0.8352189660072327
Precision: 0.8483520150184631
```



```
Testing Accuracy
1046/1046 [==============================] - 0s 371us/step
[0.6723246641642276, 0.6582217812538147, 0.6782434582710266, 0.6811769008636475, 0.6780592799186707]
Loss: 0.6723246641642276
Accuracy: 0.6582217812538147
F-1 Score: 0.6782434582710266
Recall: 0.6811769008636475
Precision: 0.6780592799186707
```

The architecture used for CNN consists of 3 convolutional layers and a max-pool after every convolution. The network has been trained over 10 and 20 epochs. After training for 20 epochs we realized that the training and testing accuracies have a huge disparity compared to the network trained on 10 epochs. Clearly, training for longer is causing overfitting and that is not desirable. Hence, we settled for this network run over 10 epochs, with a training accuracy of 82% and a testing accuracy of 66%.

REFERENCES

[1] Rifai, Salah, et al. "Contractive auto-encoders: Explicit invariance during feature extraction." *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, 2011.