Data and Image Processing Using Machine Learning Software of Space Exploration

Sam Hughes & Sumeet Kothare
CISC 489

Result Replication Procedure:

Our project was based on an example that used data from various species of Iris flowers to learn how to classify and differentiate the different species. We concluded that this idea was similar to the crater identification issue we were attempting to solve and could therefore use similar code.

- Dataset
    - To begin, we had to acquire a dataset which needed to include quantifiable features of the extra terrestrial landforms in a set of images. We began by assembling this dataset by hand and measuring the height and width of the most profound feature in each image. We did this with the idea in mind that we would eventually be able to use a software to analyze images and produce these numbers for us, but for testing purposes, measuring by hand was deemed to be suitable for now.
    - The data was compiled in a CSV file to be compatible with our example code. The header of the file must include information about the dataset. First, the number of entries, in our case, twenty-one. Second the number of data points on each entry, for us this was two. And last, the classes each entry is identified to be. For our data, we were aiming to identify images as having a crater or not so we had 0 - crater and 1 - non-crater. Each entry the had its own row with the columns being, height, width and classification.
    - Our data set can be found here
        - https://docs.google.com/spreadsheets/d/1z1_hd_wk-TCQ9qQ99EIphbH_D5tXw8nnUW5Be6bzh-c/edit?usp=sharing
    - Our next approach involved setting up a dual classification structure using Google Drive. A folder was populated with crater images exclusively whereas, a second folder was populated with images of valleys and ridges. This dataset can be found in the images folder.
- Code execution
    - We followed along almost exactly with Google's Iris Flower Identification example on the Google Colab software found here
        - https://colab.research.google.com/github/tensorflow/models/blob/master/samples/core/get_started/eager.ipynb

- Alterations had to be made to the code in order to import a local dataset rather than download one from the internet. We also had to adapt the machine learning model to work with a smaller dataset with fewer features
- Our code can be found here.
- Additionally, we executed our second project route using a program package for building Convolutional Neural Networks using tensorFlow, openCV, and Google Drive.
- The code, for this second option, was run on the Google Colaboratory Platform and is documented with the required steps for training the model and testing it using our data sets shown above. The Python code used on Google Colaboratory can be found on our GitHub repository.

- Results

```
[71]  images, cls_true  = data.train.images, data.train.cls

      # Plot the images and labels using our helper-function above.
      plot_images(images=images, cls_true=cls_true)
```

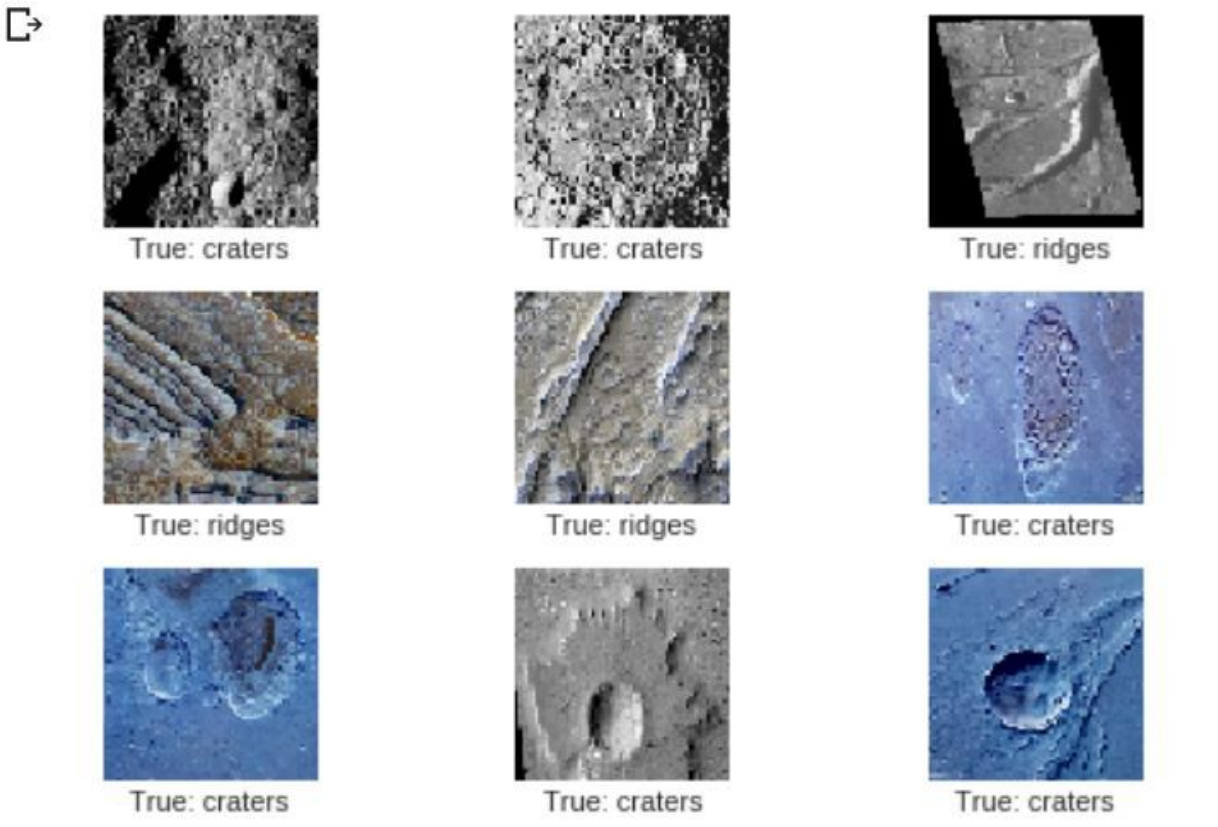

Figure 1: Successful Training of the CNN using the training dataset of 52 images, 13 validations, and a batch size of 10 images.

```
[237] #THIS STEP RUNS YOUR MODEL TO TEST TRAINING AND VALIDATION ACCURACY

      #Improving accuracy here is quite difficult as you need to have an indepth understanding of how and why different convolutional, fully connection and pooli
      #Depending on the image types you use it will converge faster than 100 iterations e.g. 25

      optimize(num_iterations=100)
```

```
Epoch 1 --- Training Accuracy:  50.0%, Validation Accuracy:  40.0%, Validation Loss: 0.726
Epoch 2 --- Training Accuracy:  50.0%, Validation Accuracy:  40.0%, Validation Loss: 0.798
Epoch 3 --- Training Accuracy:  60.0%, Validation Accuracy:  60.0%, Validation Loss: 0.699
Epoch 4 --- Training Accuracy:  70.0%, Validation Accuracy:  40.0%, Validation Loss: 0.750
Epoch 5 --- Training Accuracy:  60.0%, Validation Accuracy:  40.0%, Validation Loss: 0.813
Epoch 6 --- Training Accuracy:  70.0%, Validation Accuracy:  40.0%, Validation Loss: 0.778
Epoch 7 --- Training Accuracy:  70.0%, Validation Accuracy:  40.0%, Validation Loss: 0.766
Epoch 8 --- Training Accuracy:  70.0%, Validation Accuracy:  40.0%, Validation Loss: 0.796
Epoch 9 --- Training Accuracy:  70.0%, Validation Accuracy:  40.0%, Validation Loss: 0.814
Epoch 10 --- Training Accuracy:  70.0%, Validation Accuracy:  40.0%, Validation Loss: 0.809
Epoch 11 --- Training Accuracy:  80.0%, Validation Accuracy:  40.0%, Validation Loss: 0.829
Epoch 12 --- Training Accuracy:  80.0%, Validation Accuracy:  40.0%, Validation Loss: 0.850
Epoch 13 --- Training Accuracy:  80.0%, Validation Accuracy:  30.0%, Validation Loss: 0.857
Epoch 14 --- Training Accuracy:  90.0%, Validation Accuracy:  30.0%, Validation Loss: 0.873
Epoch 15 --- Training Accuracy:  90.0%, Validation Accuracy:  30.0%, Validation Loss: 0.894
Epoch 16 --- Training Accuracy:  90.0%, Validation Accuracy:  30.0%, Validation Loss: 0.904
Epoch 17 --- Training Accuracy:  90.0%, Validation Accuracy:  30.0%, Validation Loss: 0.920
Epoch 18 --- Training Accuracy:  90.0%, Validation Accuracy:  30.0%, Validation Loss: 0.937
Epoch 19 --- Training Accuracy:  90.0%, Validation Accuracy:  30.0%, Validation Loss: 0.951
Epoch 20 --- Training Accuracy:  90.0%, Validation Accuracy:  30.0%, Validation Loss: 0.963
Time elapsed: 0:00:01
```

Figure 2: Training and Validation Accuracy

```
#Print Test Accuracy
msg_test = "Test Accuracy: {0:>6.1%}"
print(msg_test.format(val_acc))
```

```
Test Accuracy:  38.5%
```

Figure 3: Final Result of the trained Model

Conclusively, the final result of the model of 38.5% accuracy in predicting the correct feature, crater or a ridge, is mediocre. This accuracy is heavily dependent on sampling bias and the kind of training data set used to train the model. This project used 52 images to train the mode and then use 10 images to test it. The accuracy may be improved by selecting more uniform, consistent, cleaner, and numerous images of topographical features. Clearly, a well-trained model will require a large dataset. Furthermore, several images sourced for the project from the internet and from NASA were not clean pictures with always distinguishable or exclusive topographical features. Hence, such training sets may be created in the future to supplement minimally bias-hampered treatment.

Project Enhancement Plans:

We think that the AI could be trained further by sourcing crater images from the internet which would very dramatically broaden our data set and possibly allow it to categorize more types of surface features. Software that read data from our images may also be able to pick up more data point on each feature other than just length and width of the feature and feed them into our machine learning algorithm.