Image Classification Algorithm
Fruits Dataset

Krishna Agarwal
The Department of Computer Science
San Diego State University
California, USA
krishna1991krish@gmail.com

Sneha Punnatin Vadakkeveetil
The Department of Computer Science
San Diego State University
California, USA
snehapv265@gmail.com

Sonali Singhal
The Department of Computer Science
San Diego State University
California, USA
sonalisinghal05@gmail.com

Abstract— In today's digital world, machine learning has become an integral part in various domains. With the mankind moving towards robotics as a helping tool for future, the need for coming out with new machine learning strategies is required. Automated harvesting is an emerging field that utilizes computer vision and machine intelligence to gather useful information about the growth and ripeness of fruits and vegetables, and other aspects of farming.

This research is usable in monitoring, sorting and picking of fruits for ensuring faster production chain. The "Fruit Classification" model will use 41,300 images for classifying fruits into various categories and sub-categories. This model will make use of "Convolutional Neural Network" as it is designed specifically for handling image data. This research will cover the importance of using CNN model in such classification problems.

I. Introduction

The parameters of a Machine Learning model directly impact its performance. Image classification has become one of the key pilot use-cases for demonstrating machine learning. We use the Fruit Image classification dataset to explore this trend.

The fruit Image classification dataset contains images of 81 classes. Different fruit classification classes directly influence automated harvesting in more visibility of the business. So, we used relevant attributes from the dataset to train the model to classify fruit images for harvesting.

A. Challenges

The various challenges that we realized with the Fruit Image classification dataset are:

1) The dataset for image classification is huge and contains around 41,300 images for various classes of fruits. The size of the actual dataset was around 2Gb. But we need to reduce the size of the images for faster processing and model training.
2) The biggest drawbacks of the huge dataset is time taken for processing and taking in account irrelevant information and thereby wasting the resources.
3) It is tricky to pre-process the images of varying size and reduce the size of every image to 100*100.

B. Solution

We identify the fruit classification using convolutional neural networks which needs deep neural networking. Along with the various classes for fruits, there was lot of images that didn't belong to fruit category and had invalid images. Further data needs to be cleaned. All these steps will utilize less resources also reducing the time taken. Since fruit classification dataset is humongous, a rational approach is inevitable. Firstly, we studied the various classes present in the dataset. Secondly, we defined our objective to classify the fruits as per the fruit category, pre-processed the dataset by removing unwanted images, reducing the size of the images to 100*100 using **Image Resizer for Windows** (Link in references) so that every image is of same size, resources consumed during the processing will be less and training on the model will faster as the size of the dataset will be reduced tremendously. With the original dataset, the processing time to train a model was roughly around 12 hours. We processed the various files in the dataset and determined the various fruit classes. The dataset is expandable across various fruit categories

After all the pre-processing on the data the dataset got reduced to 294Mb and processing time to 2hours and 30mins* which helped us to train the model faster. Now our dataset is ready with images of resolution size 100x100 on which our model can be trained.

And finally, we divided the dataset into 80%-10%-10% ratio for training-validation-testing dataset and applied the CNN model for image classification. In the conclusion on accuracy, precision, recall-rate and runtime for the best variant of CNN model.

II. Initial Set Up

Python offers flexibility with easy implementation and minimal syntax. We use the machine learning development environment of Python on pycharm and visual studio code. Python also offers helpful libraries such as pandas for data structures and analysis, numpy for large and multi-dimensional arrays and matrices, tensorboard for plotting the graphs which will be required in the project. The installation of all these libraries is done using the pip command in Anaconda python kernel. The easy installation can be done in Anaconda python environment using below command after importing pip:

conda install -c anaconda numpy

conda install -c conda-forge tensorflow

III. MODEL DESCRIPTION

The **dataset** folder contains the images which are classified into 81 folders i.e. into 81 classes. The script **train.py** loads the pre-trained module and trains a new classifier on top of fruit images. The beauty of transfer learning is that lower layers that were trained to distinguish between some objects can be re-used for many recognition tasks without altercation. The script first reads the images from the dataset folder and then calculates and caches the bottleneck values for each image. '**Bottleneck'** is a layer just before the final output layer that does the classification. The bottleneck layer outputs a set of values which is used by the classifier to distinguish between each image class. Since every image is re-used multiple times during training and calculating each bottle neck takes a significant amount of time, we cache these values in the folder **logs/bottlenecks** to reduce the overhead of calculating the values multiple times for a single image. Since we have run the model once, the script makes use of the cached values and makes the next runs much faster.

Once the bottlenecks are complete, the actual training of the top layer of the network begins. At each output step, the training accuracy, validation accuracy and cross entropy is calculated. The training accuracy shows that for each batch of 100 what percent of images were labelled with the correct class. The validation accuracy is the precision on a randomly-selected group of images from a different set. Cross entropy is a loss function which provides us information on how well the learning process is progressing. The training's objective is to reduce the cross-entropy value to as low as possible. We can observe the cross-entropy value at each 100^{th} output step and determine how well the model is learning in the training.

By default, the script uses 5000 training steps. Each step chooses 100 images at random from the training set, finds their bottlenecks from the cache, and feeds them into the final layer to get predictions. Those predictions are then compared against the actual labels to update the final layer's weights through the back-propagation process. As the process continues the accuracy keeps improving and after all the steps are done, a final test accuracy evaluation is run on a set of images kept separate from the training and validation pictures. It gives us the estimate of how well the trained model will perform on the classification task.

The script will write out the new model trained on the fruit categories in **logs/trained_graph.pb** and a text file containing the labels in **logs/trained_labels.txt**. The script also includes **TensorBoard** summaries that make it easier to understand, debug and optimize training. The **inception** folder contains all the files required for inception model to work.

Some of the important parameter/hyper-parameter values are defined below with its default values:

Parameter/Hyper-Parameter	Default Value
steps_for_training	5000
learning_rate	0.01
percentage_for_testing	10
percentage_for_validation	10
batch_size_for_training	100
batch_size_for_testing	-1
batch_size_for_validation	100

IV. Model Implementation

The data is divided for training, validation and testing. We run the data on convolutional neural network by tuning their respective parameters. Using CNN, we trained the dataset using various parameters and using various combinations for training-validation-testing data.

A. Training parameters Set1(80-10-10)

1) Training Parameters:

- Train batch size: 100
- Test batch size: 1
- Validation batch size: 100
- Learning rate: 0.01

2) Graphs:

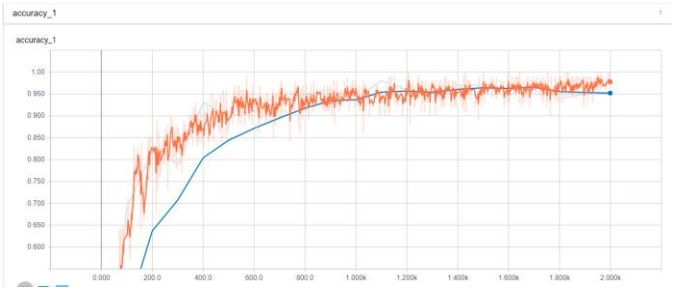

Fig. 1.1 Accuracy for train and validation data

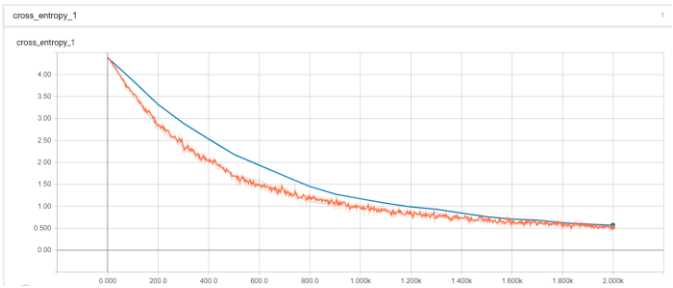

Fig. 1.2 Cross entropy for train and validation data

```
Step: 0, Train accuracy: 15.0000%, Cross entropy: 4.357937, Validation accuracy: 4.0% (N=100)
Step: 100, Train accuracy: 80.0000%, Cross entropy: 3.492501, Validation accuracy: 69.0% (N=100)
Step: 200, Train accuracy: 78.0000%, Cross entropy: 2.865616, Validation accuracy: 82.0% (N=100)
Step: 300, Train accuracy: 90.0000%, Cross entropy: 2.318330, Validation accuracy: 79.0% (N=100)
Step: 400, Train accuracy: 89.0000%, Cross entropy: 2.152843, Validation accuracy: 93.0% (N=100)
Step: 500, Train accuracy: 95.0000%, Cross entropy: 1.711209, Validation accuracy: 90.0% (N=100)
Step: 600, Train accuracy: 93.0000%, Cross entropy: 1.515374, Validation accuracy: 91.0% (N=100)
Step: 700, Train accuracy: 96.0000%, Cross entropy: 1.240842, Validation accuracy: 93.0% (N=100)
Step: 800, Train accuracy: 96.0000%, Cross entropy: 1.124755, Validation accuracy: 95.0% (N=100)
Step: 900, Train accuracy: 95.0000%, Cross entropy: 1.133910, Validation accuracy: 96.0% (N=100)
Step: 1000, Train accuracy: 99.0000%, Cross entropy: 0.909778, Validation accuracy: 94.0% (N=100)
Step: 1100, Train accuracy: 94.0000%, Cross entropy: 0.877883, Validation accuracy: 98.0% (N=100)
Step: 1200, Train accuracy: 98.0000%, Cross entropy: 0.807302, Validation accuracy: 96.0% (N=100)
Step: 1300, Train accuracy: 98.0000%, Cross entropy: 0.765718, Validation accuracy: 95.0% (N=100)
Step: 1400, Train accuracy: 95.0000%, Cross entropy: 0.740127, Validation accuracy: 97.0% (N=100)
Step: 1500, Train accuracy: 99.0000%, Cross entropy: 0.628615, Validation accuracy: 97.0% (N=100)
Step: 1600, Train accuracy: 92.0000%, Cross entropy: 0.731030, Validation accuracy: 96.0% (N=100)
Step: 1700, Train accuracy: 94.0000%, Cross entropy: 0.684476, Validation accuracy: 97.0% (N=100)
Step: 1800, Train accuracy: 99.0000%, Cross entropy: 0.527860, Validation accuracy: 94.0% (N=100)
Step: 1900, Train accuracy: 92.0000%, Cross entropy: 0.607158, Validation accuracy: 95.0% (N=100)
Step: 1999, Train accuracy: 97.0000%, Cross entropy: 0.516958, Validation accuracy: 95.0% (N=100)
Final test accuracy = 97.0% (N=8362)
```

Fig. 1.3 Execution logs

B. Training parameters Set2(70-10-20)

1) Training Parameters:

- Train batch size: 100
- Test batch size: 1
- Validation batch size: 100
- Learning rate: 0.01

2) *Graphs:*

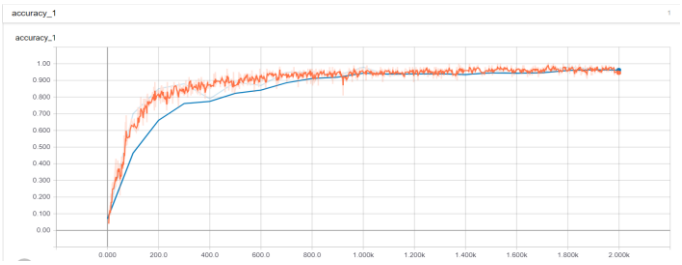

Fig. 2.1 Accuracy for train and validation data

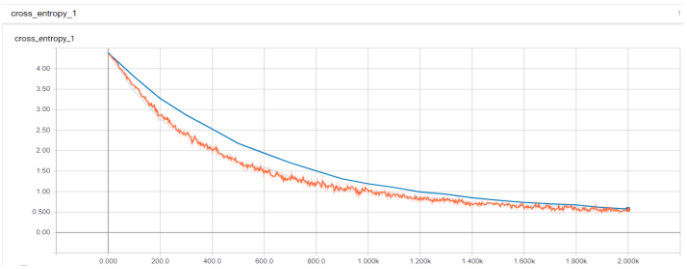

Fig. 2.2 Cross entropy for train and validation data

```
Step: 0, Train accuracy: 20.0000%, Cross entropy: 4.351013, Validation accuracy: 7.0% (N=100)
Step: 100, Train accuracy: 68.0000%, Cross entropy: 3.430526, Validation accuracy: 70.0% (N=100)
Step: 200, Train accuracy: 82.0000%, Cross entropy: 2.860171, Validation accuracy: 85.0% (N=100)
Step: 300, Train accuracy: 91.0000%, Cross entropy: 2.405802, Validation accuracy: 88.0% (N=100)
Step: 400, Train accuracy: 89.0000%, Cross entropy: 2.006136, Validation accuracy: 79.0% (N=100)
Step: 500, Train accuracy: 85.0000%, Cross entropy: 1.706185, Validation accuracy: 89.0% (N=100)
Step: 600, Train accuracy: 91.0000%, Cross entropy: 1.431476, Validation accuracy: 87.0% (N=100)
Step: 700, Train accuracy: 94.0000%, Cross entropy: 1.209351, Validation accuracy: 95.0% (N=100)
Step: 800, Train accuracy: 97.0000%, Cross entropy: 1.176396, Validation accuracy: 95.0% (N=100)
Step: 900, Train accuracy: 95.0000%, Cross entropy: 1.056891, Validation accuracy: 93.0% (N=100)
Step: 1000, Train accuracy: 95.0000%, Cross entropy: 0.949818, Validation accuracy: 98.0% (N=100)
Step: 1100, Train accuracy: 96.0000%, Cross entropy: 0.824493, Validation accuracy: 93.0% (N=100)
Step: 1200, Train accuracy: 95.0000%, Cross entropy: 0.877495, Validation accuracy: 94.0% (N=100)
Step: 1300, Train accuracy: 96.0000%, Cross entropy: 0.768118, Validation accuracy: 94.0% (N=100)
Step: 1400, Train accuracy: 96.0000%, Cross entropy: 0.605304, Validation accuracy: 93.0% (N=100)
Step: 1500, Train accuracy: 97.0000%, Cross entropy: 0.695476, Validation accuracy: 96.0% (N=100)
Step: 1600, Train accuracy: 96.0000%, Cross entropy: 0.721118, Validation accuracy: 94.0% (N=100)
Step: 1700, Train accuracy: 96.0000%, Cross entropy: 0.559216, Validation accuracy: 95.0% (N=100)
Step: 1800, Train accuracy: 96.0000%, Cross entropy: 0.509687, Validation accuracy: 98.0% (N=100)
Step: 1900, Train accuracy: 98.0000%, Cross entropy: 0.539779, Validation accuracy: 97.0% (N=100)
Step: 1999, Train accuracy: 94.0000%, Cross entropy: 0.570996, Validation accuracy: 96.0% (N=100)
Final test accuracy = 96.6% (N=16642)
```

Fig. 2.3 Execution logs

C. Training parameters Set3(70-10-20)

1) Training Parameters:

- Train batch size: 100
- Test batch size: 1
- Validation batch size: 100
- Learning rate: 0.05

2) Graphs:

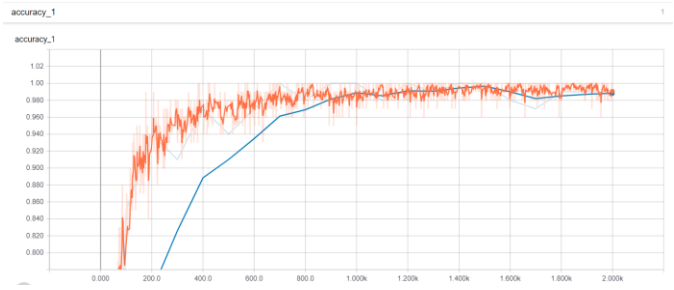

Fig. 3.1 Accuracy for train and validation data

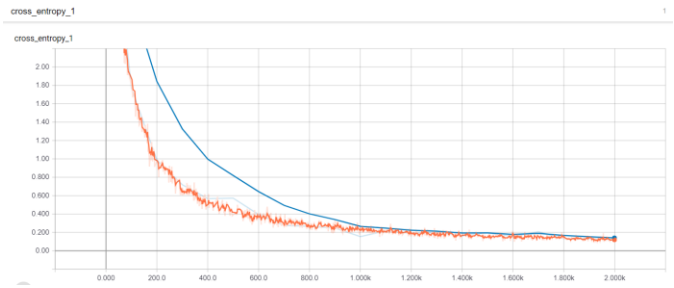

Fig. 3.2 Cross entropy for train and validation data

```
Step: 0, Train accuracy: 13.0000%, Cross entropy: 4.170862, Validation accuracy: 8.0% (N=100)
Step: 100, Train accuracy: 91.0000%, Cross entropy: 1.565411, Validation accuracy: 85.0% (N=100)
Step: 200, Train accuracy: 95.0000%, Cross entropy: 1.057176, Validation accuracy: 94.0% (N=100)
Step: 300, Train accuracy: 97.0000%, Cross entropy: 0.640518, Validation accuracy: 91.0% (N=100)
Step: 400, Train accuracy: 100.0000%, Cross entropy: 0.431093, Validation accuracy: 97.0% (N=100)
Step: 500, Train accuracy: 97.0000%, Cross entropy: 0.414319, Validation accuracy: 94.0% (N=100)
Step: 600, Train accuracy: 99.0000%, Cross entropy: 0.324740, Validation accuracy: 97.0% (N=100)
Step: 700, Train accuracy: 100.0000%, Cross entropy: 0.253460, Validation accuracy: 100.0% (N=100)
Step: 800, Train accuracy: 99.0000%, Cross entropy: 0.257307, Validation accuracy: 98.0% (N=100)
Step: 900, Train accuracy: 98.0000%, Cross entropy: 0.318767, Validation accuracy: 100.0% (N=100)
Step: 1000, Train accuracy: 98.0000%, Cross entropy: 0.239102, Validation accuracy: 100.0% (N=100)
Step: 1100, Train accuracy: 99.0000%, Cross entropy: 0.217205, Validation accuracy: 98.0% (N=100)
Step: 1200, Train accuracy: 99.0000%, Cross entropy: 0.197309, Validation accuracy: 100.0% (N=100)
Step: 1300, Train accuracy: 98.0000%, Cross entropy: 0.188450, Validation accuracy: 99.0% (N=100)
Step: 1400, Train accuracy: 100.0000%, Cross entropy: 0.176814, Validation accuracy: 100.0% (N=100)
Step: 1500, Train accuracy: 100.0000%, Cross entropy: 0.138376, Validation accuracy: 100.0% (N=100)
Step: 1600, Train accuracy: 100.0000%, Cross entropy: 0.124678, Validation accuracy: 98.0% (N=100)
Step: 1700, Train accuracy: 99.0000%, Cross entropy: 0.111533, Validation accuracy: 97.0% (N=100)
Step: 1800, Train accuracy: 98.0000%, Cross entropy: 0.155887, Validation accuracy: 99.0% (N=100)
Step: 1900, Train accuracy: 99.0000%, Cross entropy: 0.099594, Validation accuracy: 99.0% (N=100)
Step: 1999, Train accuracy: 99.0000%, Cross entropy: 0.105235, Validation accuracy: 99.0% (N=100)
Final test accuracy = 99.2% (N=16642)
```

Fig. 3.3 Execution logs

D. Training parameters Set4(70-10-20)

1) Training Parameters:

- Train batch size: 100
- Test batch size: 1
- Validation batch size: 100
- Learning rate: 0.5

2) *Graphs:*

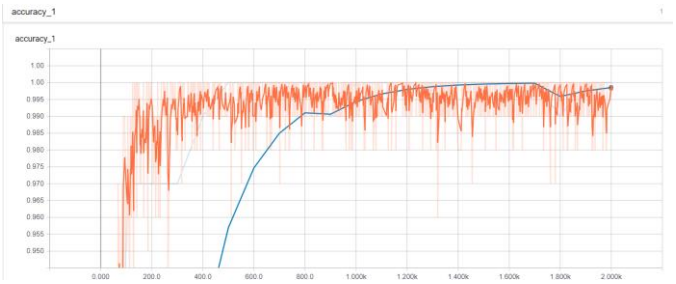

Fig. 4.1 Accuracy for train and validation data

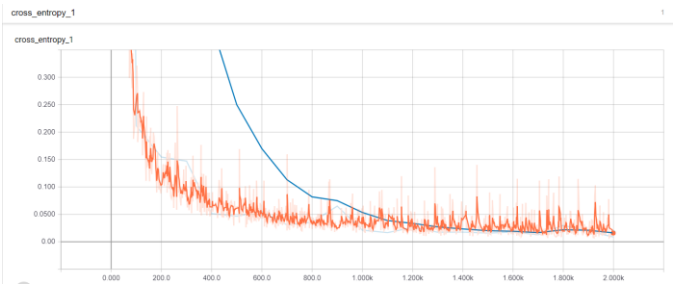

Fig. 4.2 Cross entropy for train and validation data

```
Step: 0, Train accuracy: 12.0000%, Cross entropy: 3.379567, Validation accuracy: 3.0% (N=100)
Step: 100, Train accuracy: 96.0000%, Cross entropy: 0.194114, Validation accuracy: 97.0% (N=100)
Step: 200, Train accuracy: 100.0000%, Cross entropy: 0.076052, Validation accuracy: 97.0% (N=100)
Step: 300, Train accuracy: 100.0000%, Cross entropy: 0.058425, Validation accuracy: 97.0% (N=100)
Step: 400, Train accuracy: 100.0000%, Cross entropy: 0.048083, Validation accuracy: 99.0% (N=100)
Step: 500, Train accuracy: 100.0000%, Cross entropy: 0.036200, Validation accuracy: 100.0% (N=100)
Step: 600, Train accuracy: 100.0000%, Cross entropy: 0.032634, Validation accuracy: 100.0% (N=100)
Step: 700, Train accuracy: 100.0000%, Cross entropy: 0.018299, Validation accuracy: 100.0% (N=100)
Step: 800, Train accuracy: 100.0000%, Cross entropy: 0.020107, Validation accuracy: 100.0% (N=100)
Step: 900, Train accuracy: 100.0000%, Cross entropy: 0.021511, Validation accuracy: 99.0% (N=100)
Step: 1000, Train accuracy: 100.0000%, Cross entropy: 0.023367, Validation accuracy: 100.0% (N=100)
Step: 1100, Train accuracy: 99.0000%, Cross entropy: 0.040499, Validation accuracy: 100.0% (N=100)
Step: 1200, Train accuracy: 100.0000%, Cross entropy: 0.014620, Validation accuracy: 100.0% (N=100)
Step: 1300, Train accuracy: 100.0000%, Cross entropy: 0.016544, Validation accuracy: 100.0% (N=100)
Step: 1400, Train accuracy: 100.0000%, Cross entropy: 0.013381, Validation accuracy: 100.0% (N=100)
Step: 1500, Train accuracy: 100.0000%, Cross entropy: 0.014167, Validation accuracy: 100.0% (N=100)
Step: 1600, Train accuracy: 100.0000%, Cross entropy: 0.009117, Validation accuracy: 100.0% (N=100)
Step: 1700, Train accuracy: 100.0000%, Cross entropy: 0.014677, Validation accuracy: 100.0% (N=100)
Step: 1800, Train accuracy: 100.0000%, Cross entropy: 0.014624, Validation accuracy: 99.0% (N=100)
Step: 1900, Train accuracy: 100.0000%, Cross entropy: 0.012266, Validation accuracy: 100.0% (N=100)
Step: 1999, Train accuracy: 100.0000%, Cross entropy: 0.011165, Validation accuracy: 100.0% (N=100)
Final test accuracy = 99.4% (N=16642)
```

Fig. 4.3 Execution logs

V. CLASSIFICATION

The script **classify**.py is used to predict the probability of a random image with our model. It first reads the file **logs/trained_labels.txt** to retrieve all the labels of our model and then reads **logs/trained_graph.pb** which contains our trained results. The input image is processed, and the probability of the image being classified to one of our class is generated as shown below:

*Classification Result (**80-10-10** & Learning rate **0.01**):*

Input:

Output:

```
The test image is likely to resemble:
apple granny smith (score = 0.71807)
```

Input:

Output:

```
The test image is likely to resemble:
cantaloupe 1 (score = 0.69121)
```

Input:

Output:

```
The test image is likely to resemble:
huckleberry (score = 0.91485)
```

Input:

Output:

```
The test image is likely to resemble:
lychee (score = 0.84862)
```

*Classification Result (**70-10-20** & Learning rate **0.5**):*

Input:

Output:

```
The test image is likely to resemble:
apple granny smith (score = 0.99776)
```

Input:

Output:

```
The test image is likely to resemble:
cantaloupe 1 (score = 0.99227)
```

Input:

Output:

```
The test image is likely to resemble:
huckleberry (score = 0.99908)
```

Input:

Output:

```
The test image is likely to resemble:
lychee (score = 0.99639)
```

VI. PERFORMANCE TUNING

The model was run on both windows and mac machines to determine if there were any performance impact due to operating systems.

The windows machine had the below configuration:

- **RAM** – 12 GB
- **OS** – Windows 10 (64-bit)
- **CPU** – Inter Core i5-8250U CPU @ 1.60 GHz 1.90 GHz

The mac machine had the below configuration:

- **RAM** – 16GB
- **OS** – Mac OS Mojave
- **CPU** – 2.3GHz Inter Core i5

Model Execution Time Analysis with different parameters on Windows and Mac OS	Windows	Mac
Training=80% Testing=10% Validation=10% Learning Rate=0.01	2hrs25mins	2hrs15mins
Training=70% Testing=20% Validation=10% Learning Rate=0.01	2hr23mins	2hrs30mins
Training=70% Testing=20% Validation=10% Learning Rate=0.05	2hrs22mins	2hrs40mins
Training=70% Testing=20% Validation=10% Learning Rate=0.5	2hrs23mins	3hrs00mins

VII. CONCLUSION

Based on the above data, we were able to conclude that a higher learning rate and lower learning rate results resulted in approximately the same duration for training the model. However, the model with highest learning rate had a better accuracy level in training, validation and testing. Also, the entropy value was the least in this scenario after 2000 training steps. The classify.py results were also higher for the training model with higher learning rate.

VIII. Link to Code

Below is the google drive url for accessing the files and folder required for running the Image Classification-Fruit Dataset model:
https://drive.google.com/drive/folders/1XEQ5tQmiwU2U_ye5We8wwEjWC_TfP8vw?usp=sharing

IX. REFERENCES

https://www.bricelam.net/ImageResizer/

https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

https://www.tensorflow.org/tutorials/images/image_recognition

https://datascience.stackexchange.com/questions/25315/why-should-softmax-be-used-in-cnn

https://www.tensorflow.org/guide/summaries_and_tensorboard

https://codelabs.developers.google.com/codelabs/cpb102-txf-learning/index.html?index=..%2F..index#0

https://ai.googleblog.com/2016/03/train-your-own-image-classifier-with.html