# 

# 

# Fruit Image Classification

Alistair Clarke

Andrews University

November 29th, 2018

# 

# Abstract

In this paper, we explored and compare multiple machine learning models, to effectively solve the problem of image classification for fruit images. Initially, we explored how utilizing Principal Component Analysis on RGB color pixels, could generate more informative features from the Google Downloaded Dataset, which was formulated by downloading several images of fruits within the categories of apple, banana, guava, mango, orange and pineapple. The results from the Principal Component Analysis indicated that changing the feature space of the RGB color pixels, did in fact increased the overall performance of the based line K-Nearest Neighbor model. Previous image classification work did show that utilizing data augmentation, would in fact increase models’ accuracy and it is for that reason that we explored how effective blurring, flipping (both vertically and horizontally), adding contrast, adding noise and gamma filters would be on this fruit dataset. Finally, we explored which neural network architecture (i.e. Dense Neural Network, Convolutional Neural Network, Pre-Trained CNN) when trained on this fruit dataset, gave the best performance on the validation set and then used the best neural network model to train on the entire training set with data augmentation and evaluate on the test set.

*Keywords:* Data Augmentation, Color Pixels, K-Nearest Neighbors, Principal Component Analysis, Neural Network Architecture, Pre-Trained CNN, Dense Neural Network, Convolutional Neural Network.

# 

# 

# Introduction

Classifying different classes of fruits rely mainly upon three baseline classes of features: color, texture and shape. However, effectively quantifying such features is a difficult problem and it is for that reason that in this paper we explored ways in which images of fruit could be classified into six fruit categories: apple, banana, guava, mango, orange and pineapple, while utilizing the use of Principal Components using RGB color pixel values [[4](#_edny2ngmxdqh)], Data Augmentation [[1](#_t1uv8zkl6zpl)] and Pre-Trained weighted features from Keras’ VGG16 and MobileNet convolutional networks [[2](#_9md54nvp2lvi), [3](#_3itrooe266w1)].

**Prior Relevant Work**

Previous fruit image classification research indicated that one way of improving the overall functionality of a fruit recognition system, would be to combine features such as shape and size with color and texture features. In addition, further research indicated that by adding more training images to the dataset, the overall classification performance would also improve [[7](#_2lewcwlkdv5e)].

# Methodology

**Data Source**

For this fruit classification problem, Kaggle’s Fruit 360 Dataset was used along with a formulated Google Downloaded Images Dataset. The Kaggle’s Fruit 360 Dataset consisted of 51,695 images in total with 38,695 images for training and 13,000 images for testing. Each individual image which was 100x100 pixels, had one single fruit within each image as well as a solid white background color. This initial Kaggle Fruit 360 Dataset had 77 classes of fruit but of those 77 classes of fruit, only six were used for this particular problem giving a total of 1577 images for training.

The second Google Downloaded Images Dataset consisted of approximately 300 images in total, with 210 images for training, 60 images for validation and 30 images for testing. It must be noted that this dataset was splitted using a stratified sample of 10% for testing, 20% for validation, 70% for training. Similar to the Kaggle Fruit 360 Dataset, this dataset’s images were all 100x100 pixels with solid white color backgrounds and the images were divided into six classes of fruit: apple, banana, guava, mango, orange and pineapple.

The fruit classification labels for this problem were extracted from the image directory names, and were 1-hot-encoded so that they could have been fed into the neural network architectures.

**Feature Generation**

The first approach explored was to generate features using average RGB color pixel values. This was accomplished by creating a boolean array of true values to remove color values greater than 250 (i.e.the white background) from each image. Once removed, using the numpy library, the mean or average was calculated for each color channel in each image. The second approach explored was to use Principal Component Analysis (i.e. PCA) to generate features based on the average color pixel values, that would produce more informative features and increase the overall performance of the baseline K-Nearest Neighbor model [[5](#_ojrlnzaxsj8m)]. For this particular problem, the first two principal components were used. The third approach tried, was to use the feature weights from a Pre-Trained VGG16 Convolutional Neural Network [[2](#_3itrooe266w1)], to possibly detect not just differences in color but also texture and shape.

**Models**

The first experimental model used in this classification problem, was the famous non-parametric k-nearest-neighbor model [[5](#_ojrlnzaxsj8m)]. From this model we were able to hypertune the variable ‘k’, to see which value increased the overall validation accuracy of the knn model. The second model explored was a dense regular neural network, with 3 hidden layers, 64 units for each hidden layer and a dropout layer of 50% to remedy cases of overfitting. The third model used for this classification experiment was a convolutional neural network, which consisted of three max-pooling layers, one dropout layer and two 2D convolutional layers with a filter size of 3x3x3 with 16 and 64 channels respectively [[8](#_e11pyr8d45ez)]. The fourth and final model used for this experiment was Keras’ Pretrained MobileNet convolutional network [[3](#_3itrooe266w1)]. For this Pretrained network, the learned weights were frozen so that the weights would not be relearned and because those weights would have had very informative features which could be used to classify images of fruit.

**Forms of Evaluation**

For this fruit classification problem, the loss or cost function used was categorical\_crossentropy which is commonly used for multi-class classification problems. What the loss function does is it calculates how much each weight at each unit or node of the neural network ought to be adjusted by, through the use of gradient descent and backpropagation. The metric used to evaluate the correctness of the model predictions was that of accuracy.

**Data Augmentation**

Theoretical and empirical evidence has shown that data augmentation increases the performance of machine learning classification models, simply because there are more images in the training set to learn and such transformations enables the model to learn better informative features [[6](#_8ntlbjmyhx6x)]. It is for that reason that for this classification problem we augmented or transformed the Google Downloaded Dataset images by, blurring, flipping vertically and horizontally, adding noise, adding a gamma filter which desaturated the RGB colors and adding contrast, to increase the overall validation test accuracies of the neural net model architectures.

# 

# 

# Experiments

The following experiments were tried:

1. Generating feature shape using sobel edge detection.
2. Generating the average RGB color pixel value.
3. Generating the first two Principal Components from the average RGB color pixel feature vector of values.
4. Removing the background from an image using sobel edge detection and watersheds.
5. Calculating the dominant color of an image
6. Generating image augmentation by blurring, flipping, adding contrast, adding noise and gamma filters.

# Results

|  |  |  |
| --- | --- | --- |
| Without Data Augmentation | | |
| Architecture | Training Accuracy | Validation Accuracy |
| Dense Neural Network | 100% | 80% |
| Conv2D Network | 54% | 60% |
| Pre-Trained MobileNet Network | 90% | 82% |

# Table 1.0 - Model Accuracy Results Without Data Augmentation After 30 Epochs

|  |  |  |  |
| --- | --- | --- | --- |
| With Data Augmentation | | | |
| Architecture | Training Accuracy | Validation Accuracy | Test Accuracy |
| Dense Neural Network | 97% | 80% | N/A |
| Conv2D Network | 50% | 64% | N/A |
| Pre-Trained MobileNet Network | 78% | 85% | 66% |

Table 2.0 - Model Accuracy Results With Data Augmentation After 30 Epochs

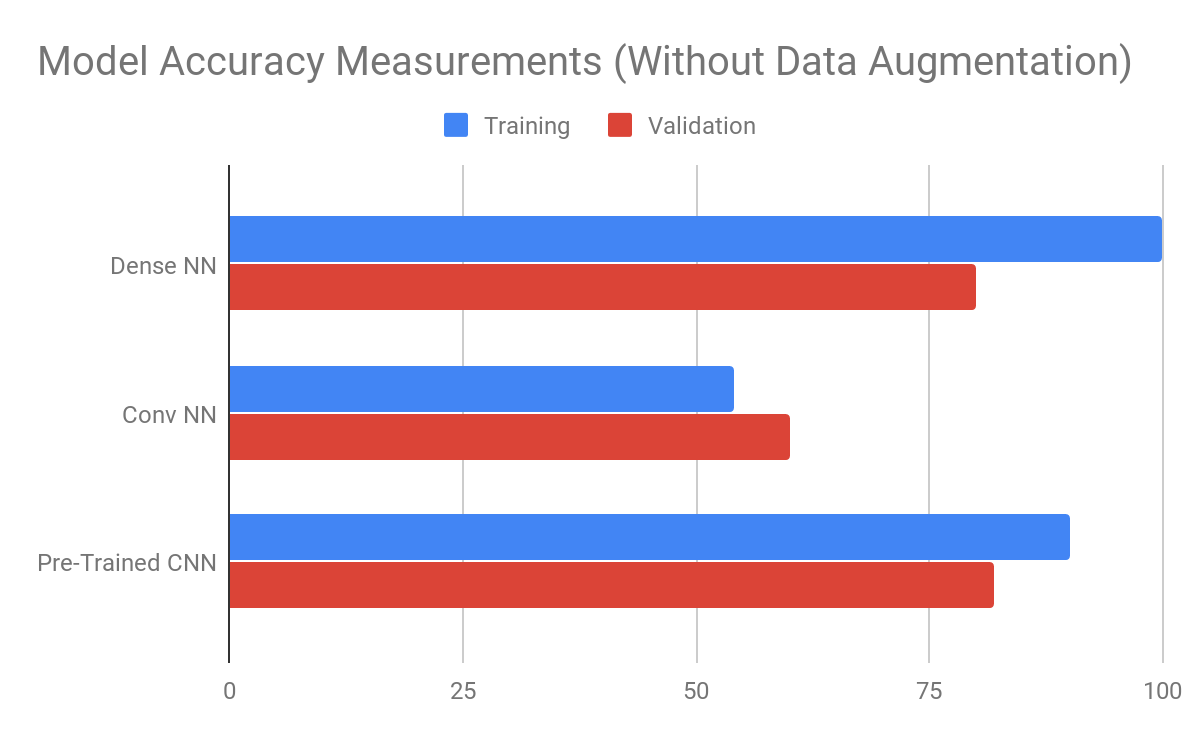


Figure 1. Model Accuracy Measurements without Data Augmentation

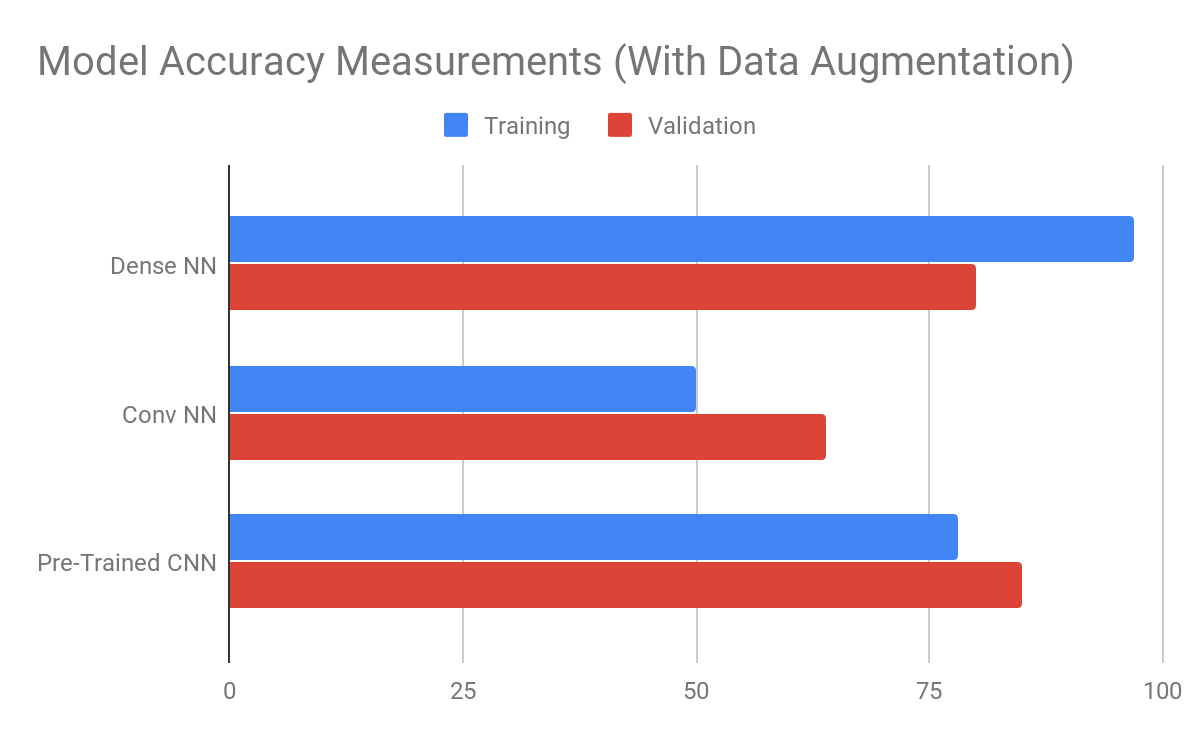


Figure 2. Model Accuracy Measurements with Data Augmentation

From the chart show in Figure 2, it can be concluded that the Pre-Trained MobileNet network when trained using data augmentation gave the best overall performance, with a validation accuracy of 85% and a test accuracy of 66%. With that being said, is can be concluded that data augmentation did in fact improve the overall accuracy of the neural network model architectures.

# 

# Conclusions

As previously mentioned, data augmentation did in fact improved the overall accuracy of all the explored models within this classification problem. In addition is must be stated that the reason why the Pretrained MobileNet network out performed the Regular Dense and Convolutional networks, was because it had already learnt higher level feature weights from previous training and so it was better able to identify feature differences from the fruit images.

From this experiment, it can be concluded that having large sums of data to train, validate and test with is very important to the success of any machine learning problem. This research experiment has also shown that generating features for image classification is a difficult task and the majority of work in the field of data science comes from data cleaning, visualization and manipulation. Through this experience I was able to fully understand and apply the machine learning pipeline and I now have an appreciate work of machine learning feature engineering.

# Future Work

1. Experiment with K-Means Image Clusterization
2. Experiment with Different Regular Dense Neural Network, Convolutional Neural Network and Pre-Trained CNN Architectures
3. Collect more images to increase the Google Downloaded Dataset
4. Experiment with Localization

# 

# 

# References

## [1] J. Wang and L. Perez, *“*The Effectiveness of Data Augmentation in Image Classification using Deep Learning*,”* in CVPR, 2017.

## [2] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, *“*MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*,”* in CVPR, 2017.

## [3] H. Kataoka, K. Iwata and Y. Satoh, *“*Feature Evaluation of Deep Convolutional Neural Networks for Object Recognition and Detection*,”* in CVPR in 2015.

## [4] A. Abadpour and S. Kasaei, *“*An Efficient PCA–Based Color Transfer Method*,”* in *Journal of Visual Communication and Image Representation*, vol. 18, no. 7, pp. 1527– 1554, 2006.

## [5] O. Boiman, E. Shechtman, and M. Irani, *“*In defense of Nearest-Neighbor based image classification*,”* *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1-8.

## [6] D. A. VAN DYK and X. MENG, *“*The Art of Data Augmentation*,”* in *Journal of Computational and Graphical Statistics*, vol. 10, Issue-1, 2001.

## [7] S. Arivazhagan, R. N. Shebiah, S. S. Nidhyanandhan , L. Ganesan, *“*Fruit Recognition using Color and Texture Features*,”* in *Journal of Emerging Trends in Computing and Information Sciences*, vol. 1, no. 2, 2010

## [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105