Capstone Project 2 Final Report
Sava Dashev
Fruit Classifier

Problem statement

Retail stores move ahead. Customers started scanning items for themselves. This pattern helps stores save money.

In grocery stores, customers buy fresh fruits. The customers have the option to go to the cashier, or scan the products themselves. In the second option, the customer sometimes does not know the name of the fruit or vegetable to enter the code. To save time for customers and money for the retailer, we can use machine learning to automatically recognize types of fruit on the scale.

The goal is to program a classifier that will recognize the type of fruit on the scale.

The client is a grocery store willing to automate fruit recognition. This way, the store will increase customer satisfaction by increasing the quality of the service offered to the customers. The process will save time to customers and may attract more customers, or decrease the percent of customers leaving.

Dataset

The dataset was created for an old competition. It is hosted on kaggle website.
The link to data:
https://www.kaggle.com/chrisfilo/fruit-recognition

The dataset contains about 44, 000 images of 15 types of fruits. The dataset was collected using unconstrained conditions. Some images are with the room light on and room lights off. Some images were taken near windows of our lab, with curtains on and off. The dataset tries to simulate conditions in store. In real application, there may be illuminations, artifacts captured by camera and other objects. Some of the changed conditions below, as described in the original kaggle dataset:

- Pose Variations with different categories of fruits
- Variability on the number of elements of fruits
- Used HD camera with 5-megapixel snapshots
- Same color but different Category fruits images with illumination variation
- Cropping and partial occlusion
- Different color, same category fruit images
- Different lighting conditions (e.g. fluorescent, natural light some of the fruits shops and supermarkets are without sunshine so it can easily affect the recognition system
- Six different kind of apple fruit images
- Three categories of mango fruit with specular reflecting shading and shadows
- Three categories of Kiwi fruit images
- Natural and artificial lighting effect on images
- Partial occlusion with hand

The size of the zipped data is about 8GB. When unzipped, it is a lot more.

I used Google Colab to host my notebooks. I initially downloaded data from kaggle, and saved it on my Google Drive. When I need the data, I upload it from the drive.

The unzipped data is in 15 folders, one for each class. Three of the classes have subfolders that contain images in subcategories. These are Apple, Guava and Kiwi. The other fruits do not have subfolders. All images are hosted in one folder.

Data cleaning

We checked the names of the images. Some of the image names contain “resized” and other image names contain “resized resized”. The resized images are of size (200, 200). These images are obtained from original images.

Most of the images are png files and rest are jpg files. One example is shown below. Image `'1GuavayAB2474.png'`:

The three resized images derived from this one: `'1GuavayAB2474 resized.jpg'`, `'1GuavayAB2474 resized.png'`, `'1GuavayAB2474 resized resized.png'`:

We see that these are three versions of the same image (note the time stamp is the same).

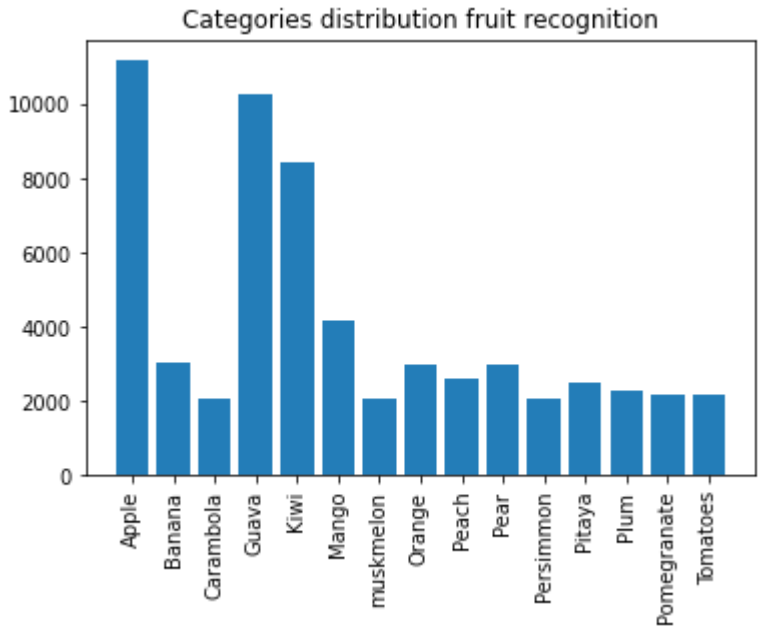
We decided to filter out these images.

The rest of the images do not contain the string “resized”. We check the sizes of these images. It turns out that 6501 are size (480, 322) and 54633 are size (320, 258). The image `'Apple/Total Number of Apples/Apple 03100.jpg'` with size (320, 240).

The names and paths of the images are saved in file_names.txt. To filter and investigate the images, we will use these paths.

We decided to use uniform size images to feed to classifier. We used the size (320, 258).

We will have only 15 classes. We count the images in the filtered list to see how many images are in each class.



We see that Apple has more than 10000 images, Guava and Kiwi have about 10 and 8 k images. All other classes have between about 4000 and 2000 images in the set we consider for machine learning.

The images were screened using the size of the image. This means none of the images is an empty file.

Preparation

Usually the images are placed in one directory when processed. We will use one size images and will not distort them. We will use the original zip file to save space and time.

We shuffle the paths of the files and corresponding classes. Next, we use train test split to create train and test sets for machine learning.

These variables are saved in files.

To feed the classifier, we will use the original zip file and the files with paths and y values for machine learning.

Machine learning

One common approach to teach a classifier to recognize images is to use convolutional neural networks (CNN). We can create CNN from scratch and fit it to data. Another approach is to use a pre-trained network and replace its dense layers to recognize images from classes in our dataset. We used both approaches to solve the problem.

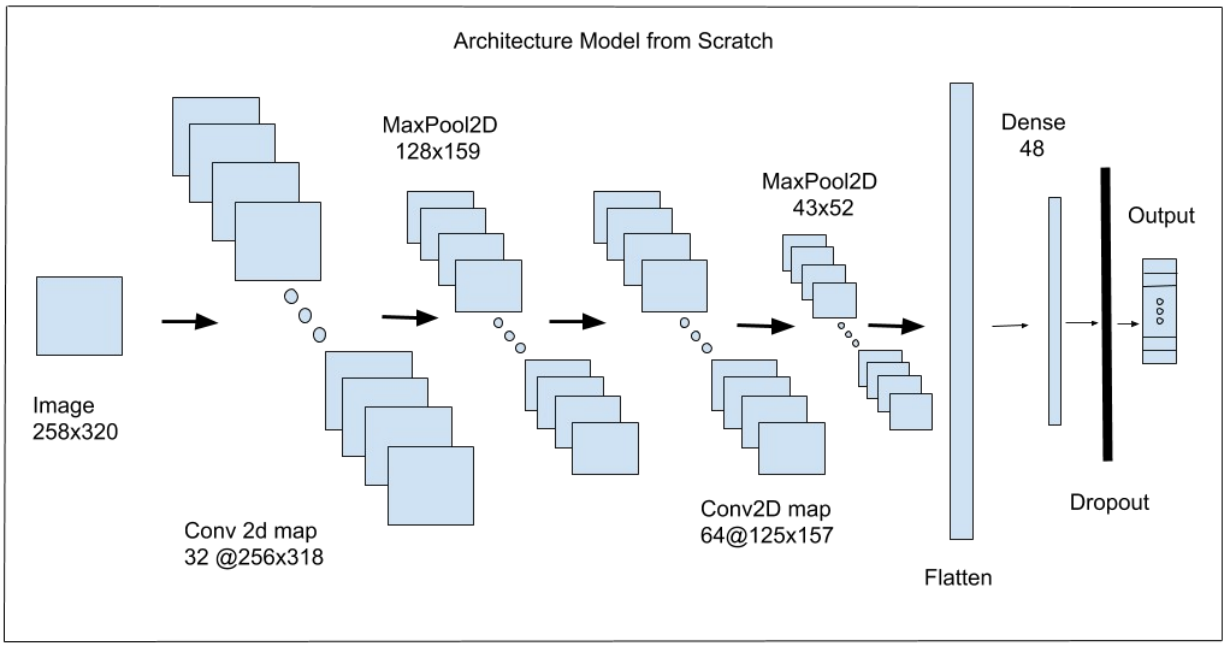
Creating classifier from scratch

The model we use is sequential. This simply means that we start with an image. We transform the image using convolution or other type of transformation. The output of the first transformation is the input of the second transformation and so on. There is no connection between non-adjacent layers in the model.

CNN-s usually contain one or more convolution layers, flatten layer, and two or more dense layers. We use the convolution layers to extract features from images. The role of the maxpool layer is to change the scale of the image to find features at different scales. Our model contains a flatten layer. We use the flatten layer to change the shape of the output to one dimensional tensor. The dense layer is used to “learn” the classes. The final dense layer contains one value for each class. This layer calculates the probability of that particular set of features to be pointing to that particular class.

The model may contain a dropout layer which prevents the model from overfitting. We place this layer just before the final dense layer.

Last layer in the model uses the values from the layer before to predict the class the image belongs to.



The CNN learns to recognize images by changing the weights each particular feature and scale has in the calculation of the probabilities. The process of changing the weights is memory expensive - each image tensor representation is multiplied by the number of filters we use at each stage.

The process is computationally expensive too. Each pixel of each feature image is calculated using element by element multiplication of small matrix, or matrix multiplication. The maxpool layer adds to the computational cost by finding the max value for the part of the matrix or averaging the values of the elements to be placed.

Our model contains a dense layer that is fully connected. Each pixel of the flatten layer is connected to all points of the dense layer and has its own multiplier.

Processing one image using the sequential model can be seen as sending a signal forward once, and when training, we send one signal back toward the first layer to change the weights we use to calculate the probability.

Optimizing the model

Optimizing the model is a process of trial and error. We need to decide how many of each type we need to have. We need also to make decisions about the number of filters in each convoluted layer, the size of the pooling. When using a dropout layer we need to decide the threshold under which the signal is simply “noise”. All these are part of the

optimization process.

There are different ways we can use the wrong guesses to do the backstep and recalculate the weights. We choose one of the methods to optimize in the compile step. We connect the build model to an optimizer. Optimizer is an algorithm to sequentially change the weights in the model so it would better predict this kind of image. Each optimizer also has parameters we can change to improve learning.

The second thing we choose when we compile the model is the loss function. This function calculates the error we try to minimize. For multiclass models we use categorical crossentropy function. We also choose metrics to track the progress. We wrap these in a list. The model we use uses accuracy.

Description of the model

The input of our model is a tensor with size 258x320x3. The first convolutional layer applies 32 filters. The output is subjected to rectified linear function to take into account the nonlinearity of the model. Following the first convolution layer is a pooling layer that reduces the size of feature maps by half in both dimensions. We repeat the two layers once more - convolutional layer with 64 filters this time and max pooling layer once more reduces the size of the image by half.

Following these layers we add the flatten layer.

We have two dense fully connected layers. The first one has 48 cells, the second has 15 cells. The final layer calculates the probability the image belongs to each class. Between the two dense layers we have a dropout layer at level 0.5. This layer reduces the noise in the model. We use this layer only in the training process, and not when we make predictions.

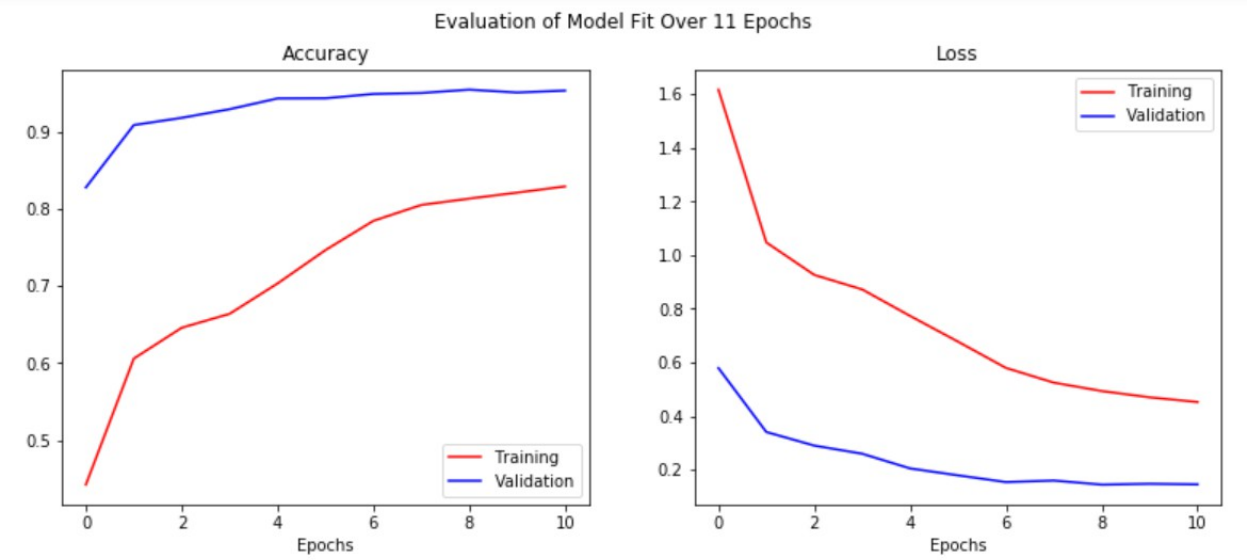
This structure of the model is the result of trial and error. We have had different number of layers and filters in each layer. We did not achieve better results introducing third convolutional layer and dropout layers at different places. For this reason, we stick to the simplest structure.

Our model has about 6,700,000 parameters. The bulk of these parameters is in the output of the dense layer. This is because of the size of the image and the number of filters in each convolution layer.

Training the model

The base model was fit with adam optimizer. We used a custom made generator to fit the images to the model. The size of a batch to feed the classifier was set to 95. The generator then calculates the number of batches for training and for validating. We fit the classifier using fit function. We did not change any of the default settings for this classifier. For full training I used 25 epochs and an early stopping monitor with patience 2. The training time was about 56 min, 14 epochs were run.

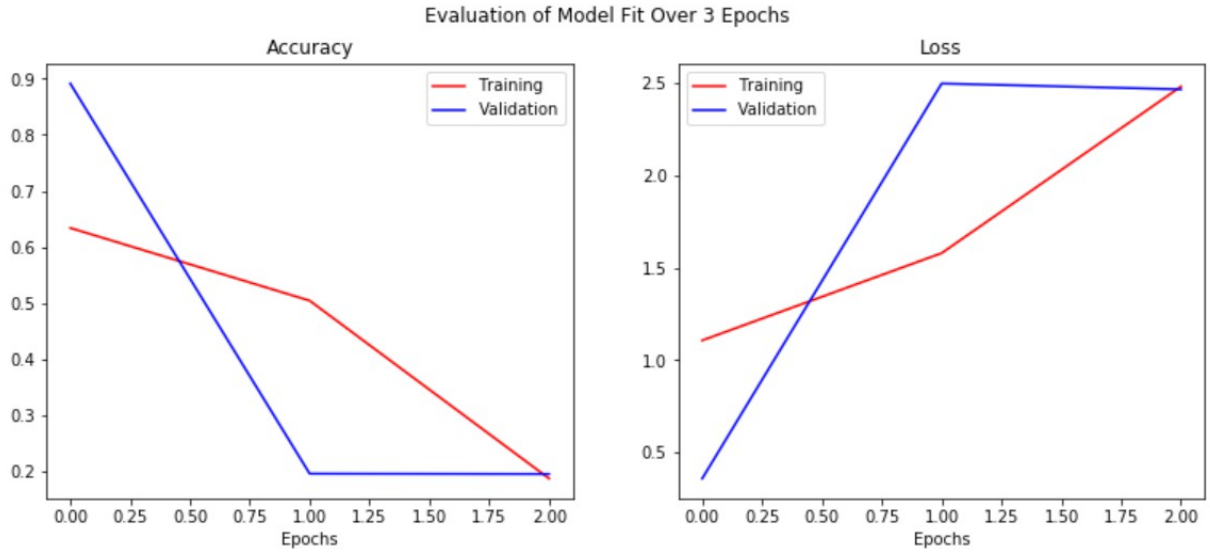
We compared the performance of all types of optimizers with the same data. We achieved it by using the same training and validation data and the size of the batches was the same. The highest performance optimizer with default setting was SGD with validation accuracy of 0.960. The base model without dropout validation accuracy was 0.953.




```
history_bsln.csv
Epochs: 11; Final accuracy: 0.829
Epochs: 11; Final val accuracy: 0.953
```

(accuracy and loss graphs)

One of the optimizers - nadam, did not perform well. The model was trained only three epochs. Possible reasons for that is too big learning rate, or need bigger or smaller batch sizes.

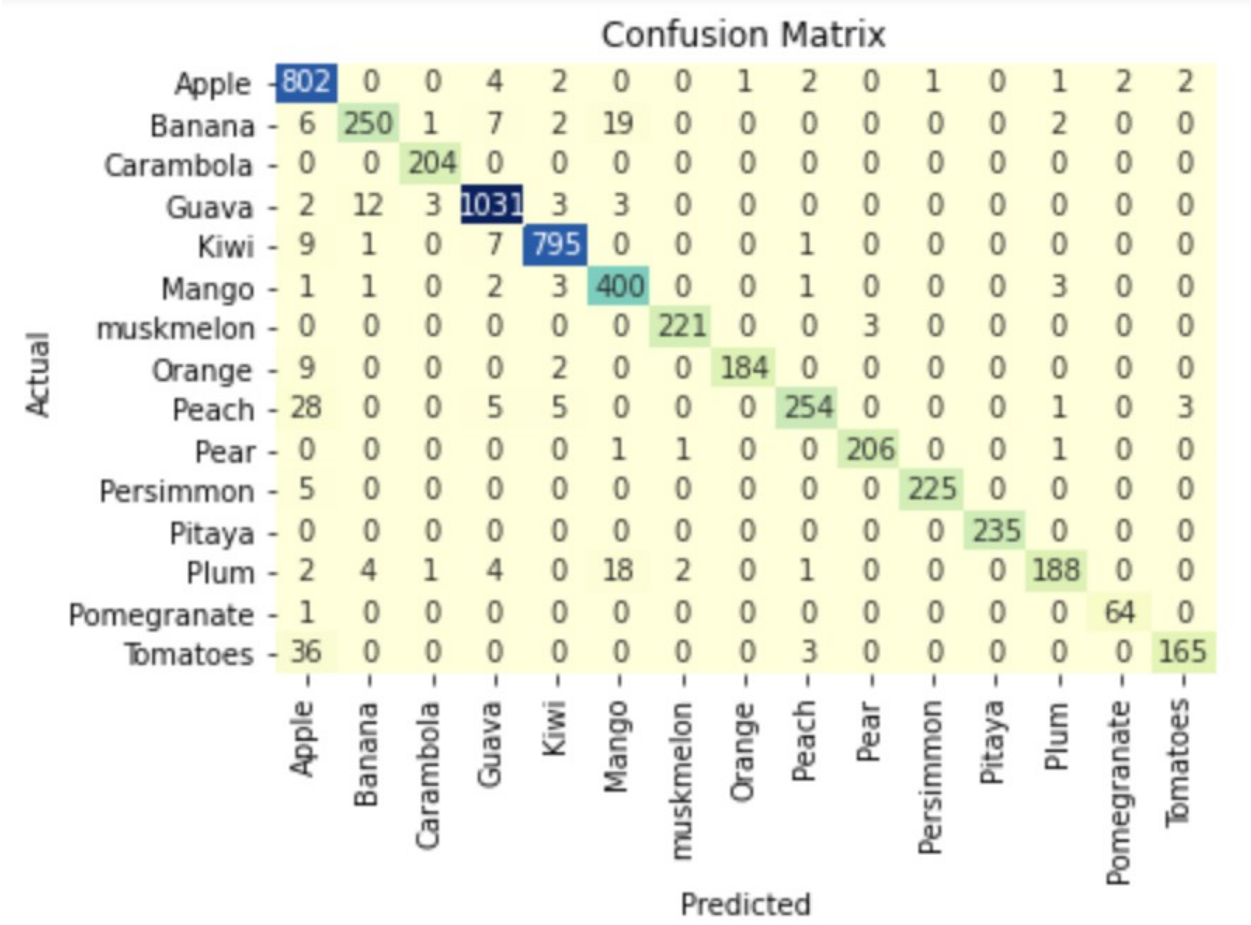



```
history_nadam.csv
Epochs: 3; Final accuracy: 0.187
Epochs: 3; Final val accuracy: 0.195
```

Evaluation on held out data

10% of the data was not used for training and validation. We used that data to evaluate the model.

The confusion matrix is shown below.



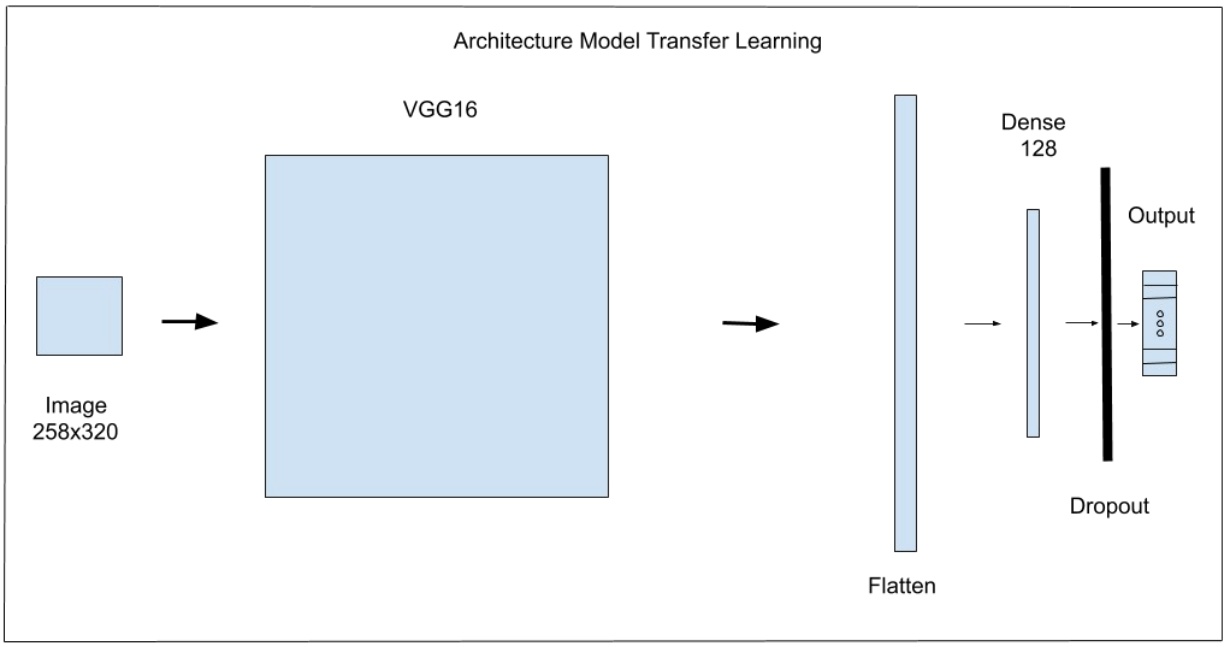
It shows pitaya, carambola and muskmelon are least missed with other fruits. Apple, banana, guava, peach and plum are much more misclassified. Pomegranate has only one misclassified. However, we have only 65 images. Considering the size of the sample, misclassification is similar to the misclassification of the apple. The most misclassified is tomato for apple - despite the small size of sample, 36 images were classified as apple class.

Transfer learning

The second approach that can be taken to classify images is using a trained classifier. The trained classifier can be used as a feature extractor and we can add a dense layer to train it for new computer vision problem. We also can re-engineer the classifier and re-train it. Our approach was the first one. We attached a dense layer, which we trained to recognize images. We used VGG16 as a convolutional base to train.

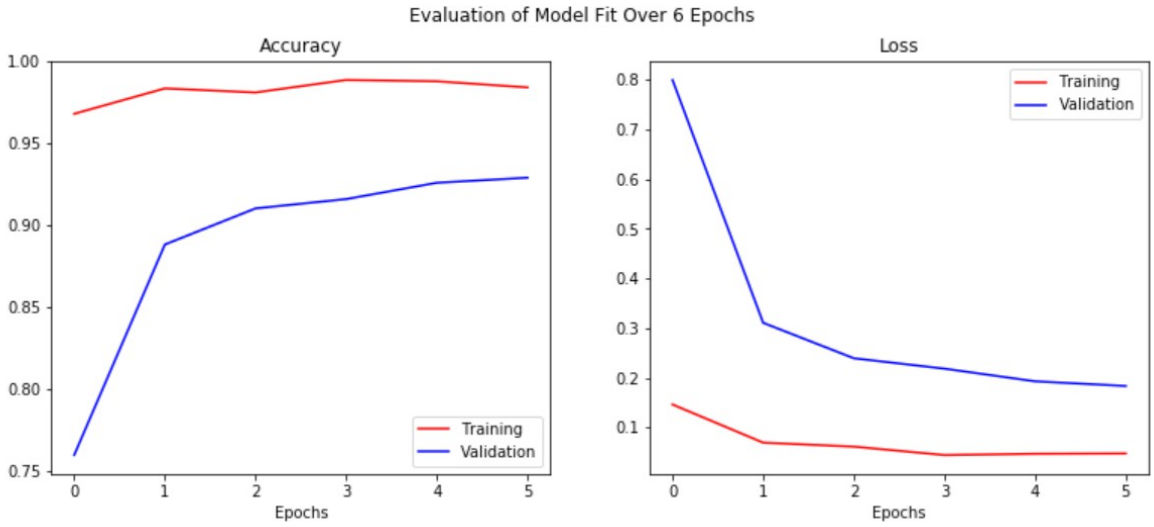
The VGG16 base did not require additional steps to prepare images. We had to change

the input size to fit our images (258 x 320 x 3). We experimented with different sizes dense layer. We got good results using a layer with 128 cells. The dropout level was set to 0.25.



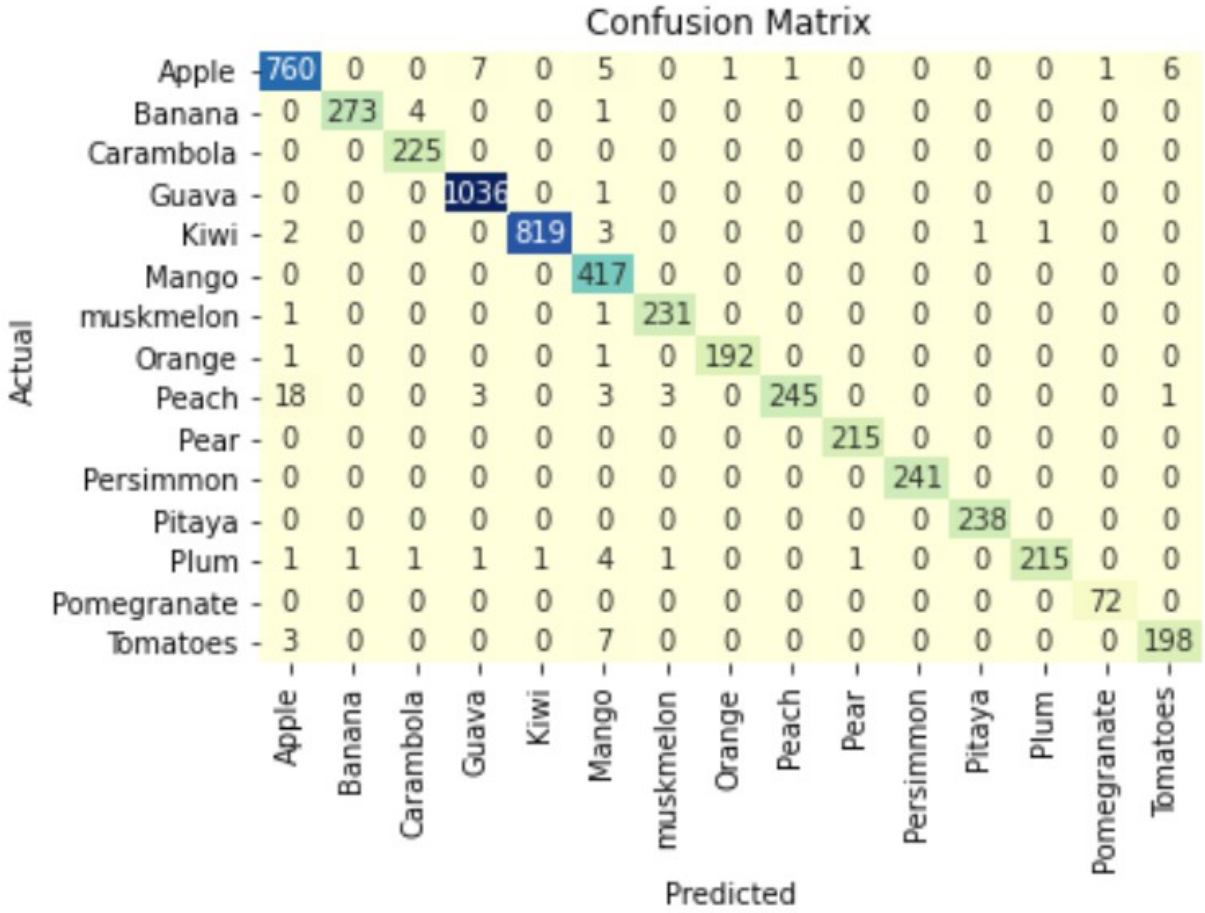
We used the same data and same sequence for the pre-trained model. With patience 2, our model finished training with 6 epochs.

The final accuracy for this model is 0.984, and it is trained for only 6 epochs.
The transfer learning confusion matrix shows a different distribution of the misclassified images.




```
history_transfer.csv
Epochs: 6; Final accuracy: 0.984
Epochs: 6; Final val accuracy: 0.929
```

Here is the confusion matrix for the transfer learning model.



The transfer classifier did better on tomato class. Much smaller number of images were misclassified as apples. Peach class also got some improvement, but not by much.The other improvement was the plum images. Plum was misclassified for all classes, but in much smaller quantities.

Discussion and conclusion

Two main approaches were presented in this project. Building computer classifier from scratch and using trained classifier to fit new image classes. The pre-trained classifier performed better than the classifier from scratch, in two different ways. One, it had higher validation accuracy and two, it trained for a lot shorter time.

There are some limitations. The classifier recognizes images of fruits in particular setting. The tray used in the images is the same. There may be a variation of the color of the tray in stores, or lightning quality and intensity of light may vary significantly from the variety presented in the set.

The classifier presented here is trained only on the original images with size of images 258x320. In the future, we can improve it by allowing augmentation of images and collecting more images. Also, we can fine tune using more images. This model is geared toward grocery stores. We can also include other types of fruits to be classified.