Name: Xiyuan Liu
Class: MSML603-PCS2
Submission date: Dec 19th, 2022
Professor: Dr. Kemal Davaslioglu

Final project report

1 Research question:

Image classification problem has been one of the hottest topics in Artificial Intelligence. Since humanity entered the 21st century, the development of technology including the internet, improvement of physical cameras and data storage technology, caused the data size of images to grow rapidly. In order to understand the content of these images, humans need Artificial Intelligence to process the image and filter the most useful data instead of manual data processing. Because the need for Artificial Intelligence preprocessing increases, people need strong AI which can not only process the data in seconds, but also complete its mission accurately. In this project, I developed a convolutional neural network(CNN) to solve image classification problems. In the same time, I also developed other models to do the same problems and I compared their performance with CNN.

2 Dataset:

Dataset for the project is a folder with fruit photos. The fruit dataset contains 15 folders which correspond to 15 classes, including Apple, Banana, Carambola, Guava, Kiwi, Mango, Orange, Peach, Pear, Persimmon, Pitaya, Plum, Pomegranate, Tomatoes, and muskmelon. The whole dataset contains 12000 samples, 800 samples from each class and all the samples from one class is in one folder. In this case, the folder name is the label of samples. The dataset has been downloaded from the website and will be provided with a zip file. The total size of the dataset is 1.48 GB.

The size of samples are not all in the same shape. Most of the sample has 320*258 pixels and some samples has 480*320 pixels. Since the size of data varies between different samples, it is necessary to adjust the size of samples to a fixed value before fitting into the model. In this project, the size of sample has been set to 480*322 pixels. The dimension of each input will be 480*322*3.

The reason for using the fruit dataset to train the model is to observe the upper limit of the model. I am intentionally picking a fruit dataset with some pattern that the fruit is placed on a metal plate with different posture (figure 1). On one hand, I want to control the noise from the environment. With some level fixed noise in the input, the performance of the model can uncover the upper limit of the model. On the other hand, I want to see how the model deals with the images that have similarities with different labels. After I train the model, I will also use other training data to evaluate the generalization of the model.

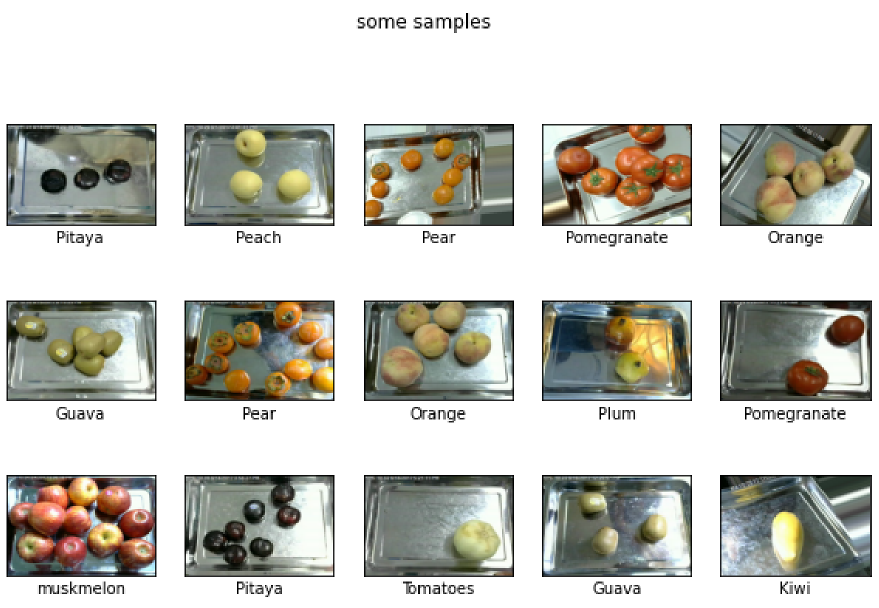


Figure 1

3 ML Methodology

3.1 Data preprocessing:

As mentioned above, since each input size is different, all samples have been adjusted to 480*322 pixels. The dataset has been split into 9600 training dataset and 2400 validation dataset. After I split the dataset, in order to save running time, I use the prefetching and caching function to load the data. With these functions, the training data can be cached into memory and the reading process can overlap with model execution of a training step.

3.2 Model selection:

Based on my study in the class, I am planning to develop 3 models.

- Model 1: Deep neural network model with CNN
- Model 2: Shallow neural network model with CNN
- Model 3: Support vector model (SVM)

With these models, I can observe how the depth of the model influences the neural network by comparing the performance of model 1 and model 2. At the same time, I can also observe the different performance of neural networks and SVM models by comparing the performance of model 2 and 3. More specific architecture of model will be in 3.3.

3.3 Model architecture:

Model 1:

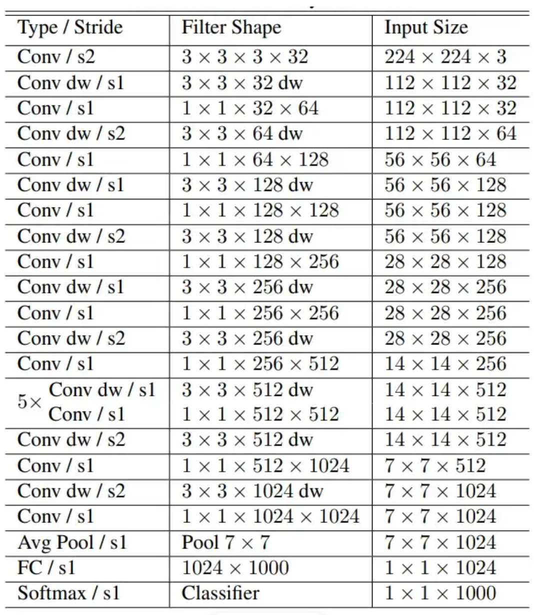
Since my lack of knowledge of how to construct a deep neural network model with CNN, I decided to use a pretrained model as my base model. In this case, the Resnet50 model is my best option. ResNet50 is a variant of the Residual Network model (Resnet) which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. Unlike usual CNN which learns different low, mid, high level features at the end of layers, Resnet tries to learn the residual which can be simplified as the difference of input layers and output layers. In Resnet, skip connection which connects activations of a layer to further layers by skipping some layers in between forms residual blocks which efficiently solve the problems from vanishing and exploding gradient. The full construction for Resnet50 is showing below:

1. 7×7, 64 kernels with a 2-sized stride = **1 layer**
2. A max pooling layer with a 2-sized stride.
3. 3×3,64 kernels + 1×1,64 kernels + 1×1,256 kernels, iterated 3 times = **9 layers**
4. 1×1,128 kernels + 3×3,128 kernels + 1×1,512 kernels, iterated 4 times. = **12 layers**
5. 1×1,256 kernels + 3×3,256 kernels + 1×1,1024 kernels, iterated 6 times. = **18 layers**
6. 1×1,512 kernels + 3×3,512 kernels + 1×1,2048 kernels iterated 3 times. = **9 layers**
7. Average pooling using the softmax activation function.= **1 layer**

Resnet gives me a 1 + 9 + 12 + 18 + 9 + 1 = 50 layers deep Convolutional network and I add 3 more layers after it including two dense layers with output shape (None, 512) and (None, 16) and final dense layer with output shape (None, 15). Therefore, 53 layers exist in this network.

Model 2:

For model 2, I want to construct a shallow CNN model which means I need to select a pretrained model that has less layers compared to Resnet50. In this case, the mobilenet model is a good choice. As the name suggests, the mobilenet model has been designed to use in mobile applications. Since the computation power of a phone is commonly less than a normal PC, mobilenet not only reduces its depth of neural network, but also reduces the numbers of parameters compared to the regular convolution network with same depth. MobileNet is a class of CNN that was open-sourced by Google and provides an extremely fast experience of training a classifier. The full construction for mobilenet is showing below:



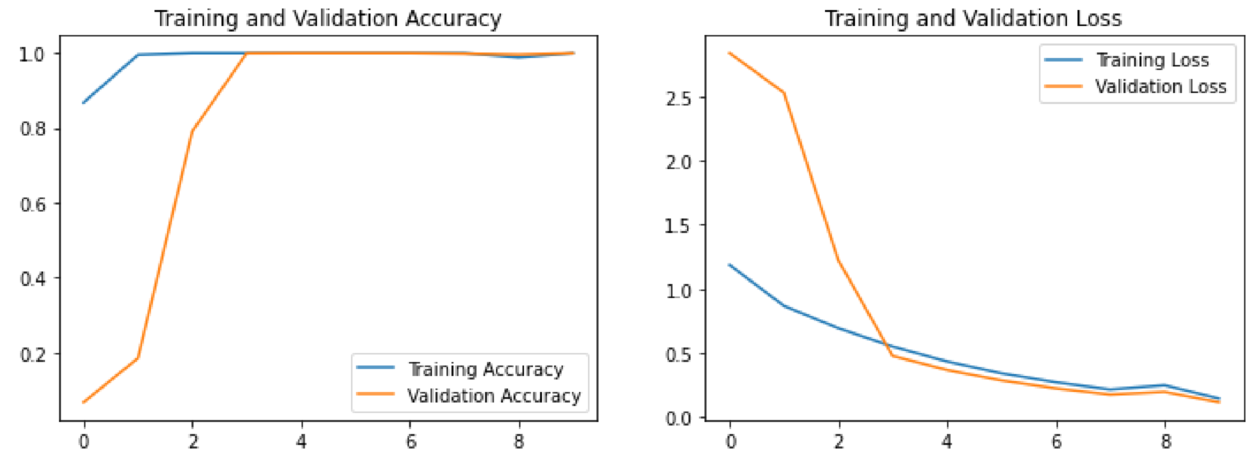
Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Architecture of mobilenet model [1]

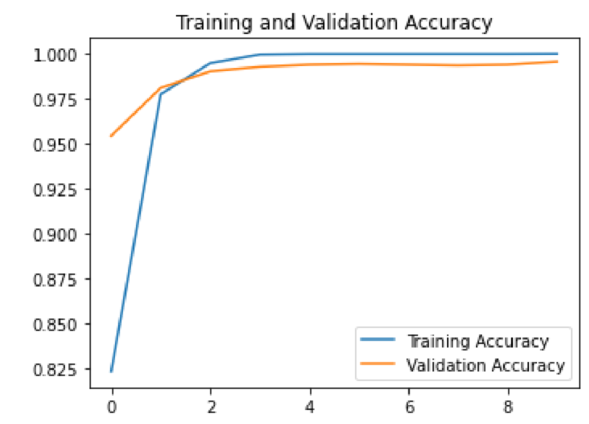
Model 3:

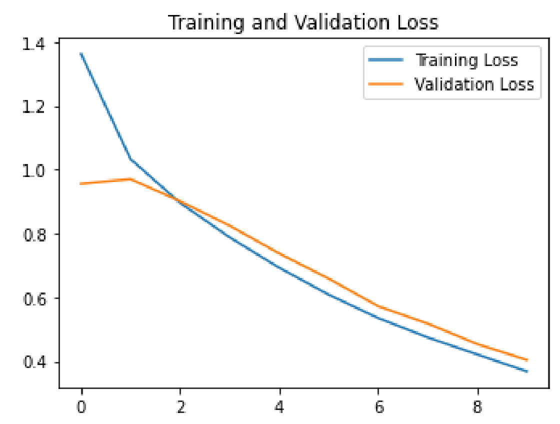
For model 3, I want to build a SVM classifier. In supervised machine learning models, svm is one of the most powerful machine methods I studied before. Using SVM still may solve the image classification problem. In order to train a SVM classifier, feature selection is a necessary step. Since the size of input sample has been set to 480*322 pixels, by flatten the feature to one dimension, the size of features will grow to 463680. SVM easily gets overfitting problems on processing high dimensional data. In this case, I use principal component analysis (PCA) to lower its dimension to 3. It is hard to determine how to set the value of dimension after PCA. However, with dimension 3, the data visualization may contribute to understanding the final performance.

4 Results:



Accuracy and loss for model 1

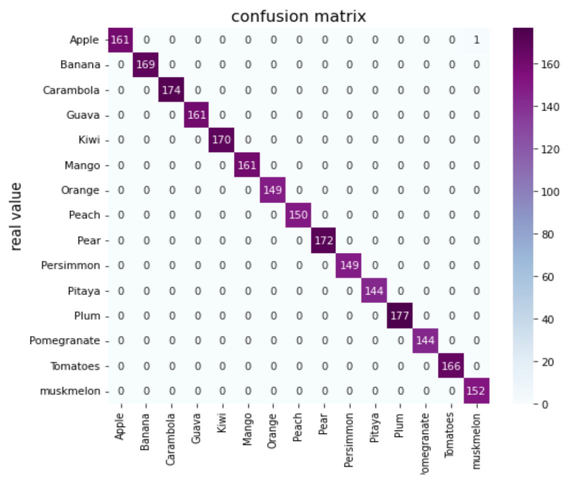




Accuracy and loss for model 1

To increase the computational speed, the code has been changed to use GPU installed in the computer allowing for a faster running of the tests. The fact is training a CNN image classifier requires a strong gpu to do computation. I cannot train the model locally and I used the NVIDIA Ampere A100 from Colab to train the network. The results show that for model 1 and model 2, the accuracy rate for both test dataset and validation dataset can reach about 99%. I also look into the confusion matrix. Model 1 almost gets the perfect prediction of validation data. At the same time, model 2 also gets a high score on prediction of validation data.

Model 1



	Apple	Banana	Carambola	Guava	Kiwi	Mango	Orange	Peach	Pear	Persimmon	Pitaya	Plum	Pomegranate	Tomatoes	muskmelon
Apple	161	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Banana	0	169	0	0	0	0	0	0	0	0	0	0	0	0	0
Carambola	0	0	174	0	0	0	0	0	0	0	0	0	0	0	0
Guava	0	0	0	161	0	0	0	0	0	0	0	0	0	0	0
Kiwi	0	0	0	0	170	0	0	0	0	0	0	0	0	0	0
Mango	0	0	0	0	0	161	0	0	0	0	0	0	0	0	0
Orange	0	0	0	0	0	0	149	0	0	0	0	0	0	0	0
Peach	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0
Pear	0	0	0	0	0	0	0	0	172	0	0	0	0	0	0
Persimmon	0	0	0	0	0	0	0	0	0	149	0	0	0	0	0
Pitaya	0	0	0	0	0	0	0	0	0	0	144	0	0	0	0
Plum	0	0	0	0	0	0	0	0	0	0	0	177	0	0	0
Pomegranate	0	0	0	0	0	0	0	0	0	0	0	0	144	0	0
Tomatoes	0	0	0	0	0	0	0	0	0	0	0	0	0	166	0
muskmelon	0	0	0	0	0	0	0	0	0	0	0	0	0	0	152

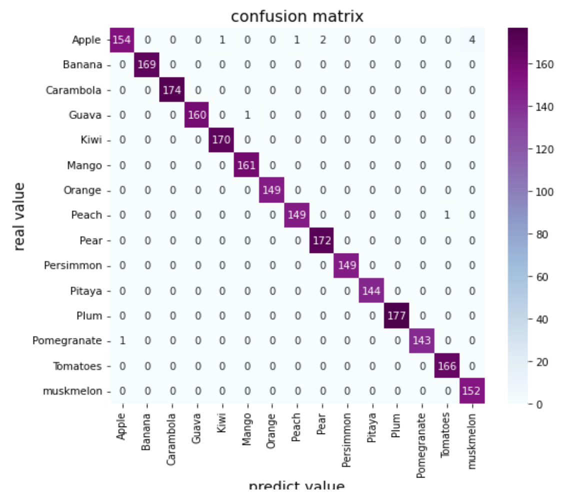
```
              precision    recall  f1-score   support

       Apple       1.00      0.99      1.00       162
      Banana       1.00      1.00      1.00       169
   Carambola       1.00      1.00      1.00       174
       Guava       1.00      1.00      1.00       161
        Kiwi       1.00      1.00      1.00       170
       Mango       1.00      1.00      1.00       161
      Orange       1.00      1.00      1.00       149
       Peach       1.00      1.00      1.00       150
        Pear       1.00      1.00      1.00       172
   Persimmon       1.00      1.00      1.00       149
      Pitaya       1.00      1.00      1.00       144
        Plum       1.00      1.00      1.00       177
 Pomegranate       1.00      1.00      1.00       144
    Tomatoes       1.00      1.00      1.00       166
   muskmelon       0.99      1.00      1.00       152

    accuracy                           1.00      2400
   macro avg       1.00      1.00      1.00      2400
weighted avg       1.00      1.00      1.00      2400

Loss function: 0.11607186496257782, accuracy: 0.9995833039283752
```

Model 2



	Apple	Banana	Carambola	Guava	Kiwi	Mango	Orange	Peach	Pear	Persimmon	Pitaya	Plum	Pomegranate	Tomatoes	muskmelon
Apple	154	0	0	0	1	0	0	1	2	0	0	0	0	0	4
Banana	0	169	0	0	0	0	0	0	0	0	0	0	0	0	0
Carambola	0	0	174	0	0	0	0	0	0	0	0	0	0	0	0
Guava	0	0	0	160	0	1	0	0	0	0	0	0	0	0	0
Kiwi	0	0	0	0	170	0	0	0	0	0	0	0	0	0	0
Mango	0	0	0	0	0	161	0	0	0	0	0	0	0	0	0
Orange	0	0	0	0	0	0	149	0	0	0	0	0	0	0	0
Peach	0	0	0	0	0	0	0	149	0	0	0	0	0	1	0
Pear	0	0	0	0	0	0	0	0	172	0	0	0	0	0	0
Persimmon	0	0	0	0	0	0	0	0	0	149	0	0	0	0	0
Pitaya	0	0	0	0	0	0	0	0	0	0	144	0	0	0	0
Plum	0	0	0	0	0	0	0	0	0	0	0	177	0	0	0
Pomegranate	1	0	0	0	0	0	0	0	0	0	0	0	143	0	0
Tomatoes	0	0	0	0	0	0	0	0	0	0	0	0	0	166	0
muskmelon	0	0	0	0	0	0	0	0	0	0	0	0	0	0	152

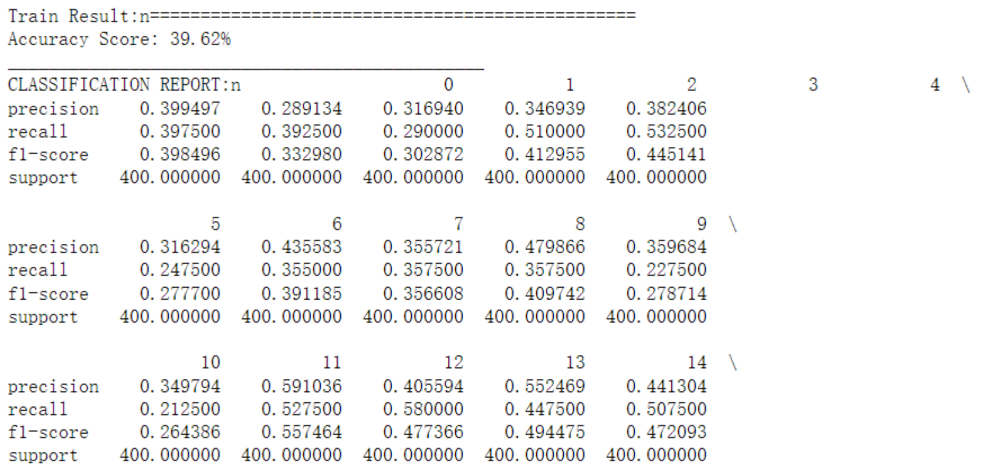
```
              precision    recall  f1-score   support

       Apple       0.99      0.95      0.97       162
      Banana       1.00      1.00      1.00       169
   Carambola       1.00      1.00      1.00       174
       Guava       1.00      0.99      1.00       161
        Kiwi       0.99      1.00      1.00       170
       Mango       0.99      1.00      1.00       161
      Orange       1.00      1.00      1.00       149
       Peach       0.99      0.99      0.99       150
        Pear       0.99      1.00      0.99       172
   Persimmon       1.00      1.00      1.00       149
      Pitaya       1.00      1.00      1.00       144
        Plum       1.00      1.00      1.00       177
 Pomegranate       1.00      0.99      1.00       144
    Tomatoes       0.99      1.00      1.00       166
   muskmelon       0.97      1.00      0.99       152

    accuracy                           1.00      2400
   macro avg       1.00      1.00      1.00      2400
weighted avg       1.00      1.00      1.00      2400

Loss function: 0.40400171279907227, accuracy: 0.9954166412353516
```

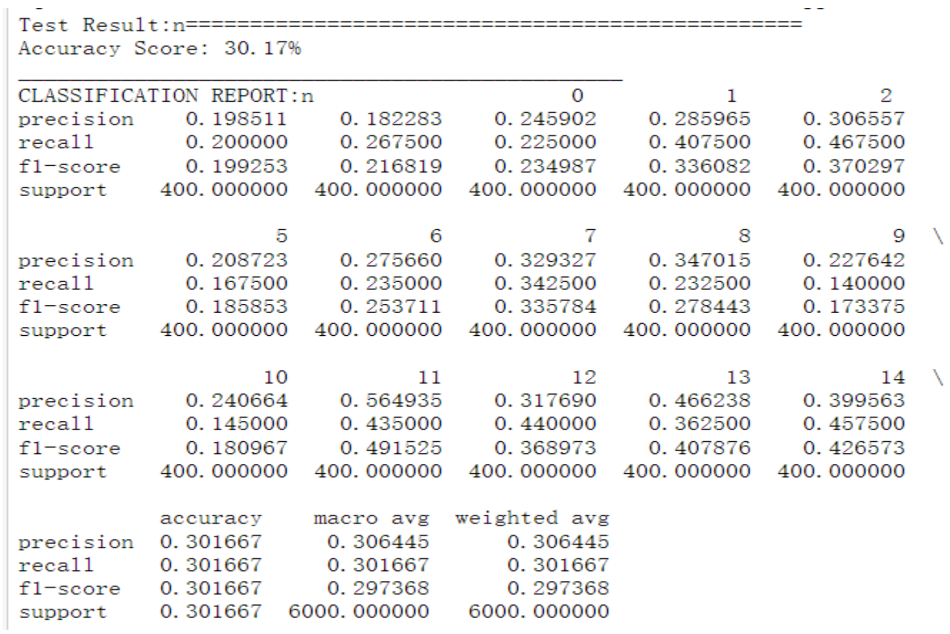
However, model 3 the accuracy rate for both test dataset and validation dataset can only reach 35-40%. In this case, SVM is a weak classifier on image classification.



```
Train Result:n============================================
Accuracy Score: 39. 62%
_______________________________________________
CLASSIFICATION REPORT:n                0           1           2           3           4  \
precision     0. 399497    0. 289134    0. 316940    0. 346939    0. 382406
recall        0. 397500    0. 392500    0. 290000    0. 510000    0. 532500
f1-score      0. 398496    0. 332980    0. 302872    0. 412955    0. 445141
support     400. 000000  400. 000000  400. 000000  400. 000000  400. 000000

                      5            6            7            8            9  \
precision     0. 316294    0. 435583    0. 355721    0. 479866    0. 359684
recall        0. 247500    0. 355000    0. 357500    0. 357500    0. 227500
f1-score      0. 277700    0. 391185    0. 356608    0. 409742    0. 278714
support     400. 000000  400. 000000  400. 000000  400. 000000  400. 000000

                     10           11           12           13           14  \
precision     0. 349794    0. 591036    0. 405594    0. 552469    0. 441304
recall        0. 212500    0. 527500    0. 580000    0. 447500    0. 507500
f1-score      0. 264386    0. 557464    0. 477366    0. 494475    0. 472093
support     400. 000000  400. 000000  400. 000000  400. 000000  400. 000000
```



```
Test Result:n================================================
Accuracy Score: 30. 17%
_______________________________________________
CLASSIFICATION REPORT:n                0           1           2
precision     0. 198511    0. 182283    0. 245902    0. 285965    0. 306557
recall        0. 200000    0. 267500    0. 225000    0. 407500    0. 467500
f1-score      0. 199253    0. 216819    0. 234987    0. 336082    0. 370297
support     400. 000000  400. 000000  400. 000000  400. 000000  400. 000000

                      5            6            7            8            9  \
precision     0. 208723    0. 275660    0. 329327    0. 347015    0. 227642
recall        0. 167500    0. 235000    0. 342500    0. 232500    0. 140000
f1-score      0. 185853    0. 253711    0. 335784    0. 278443    0. 173375
support     400. 000000  400. 000000  400. 000000  400. 000000  400. 000000

                     10           11           12           13           14  \
precision     0. 240664    0. 564935    0. 317690    0. 466238    0. 399563
recall        0. 145000    0. 435000    0. 440000    0. 362500    0. 457500
f1-score      0. 180967    0. 491525    0. 368973    0. 407876    0. 426573
support     400. 000000  400. 000000  400. 000000  400. 000000  400. 000000

             accuracy     macro avg  weighted avg
precision   0. 301667    0. 306445     0. 306445
recall      0. 301667    0. 301667     0. 301667
f1-score    0. 301667    0. 297368     0. 297368
support     0. 301667 6000. 000000  6000. 000000
```

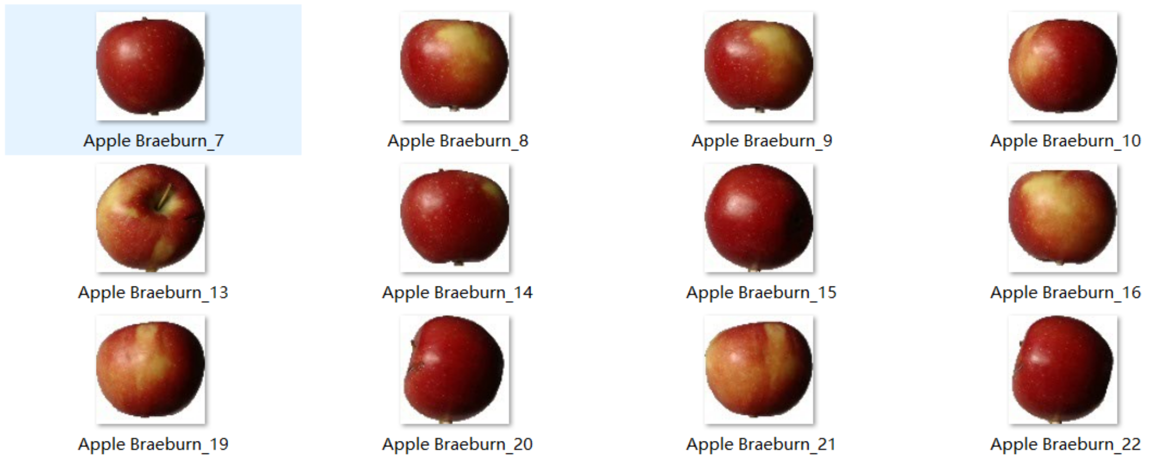
After comparing all models' performance, since model 1 gets the highest accuracy score and the validation set is unbiased, model 1 is the best model.

5 Discussion

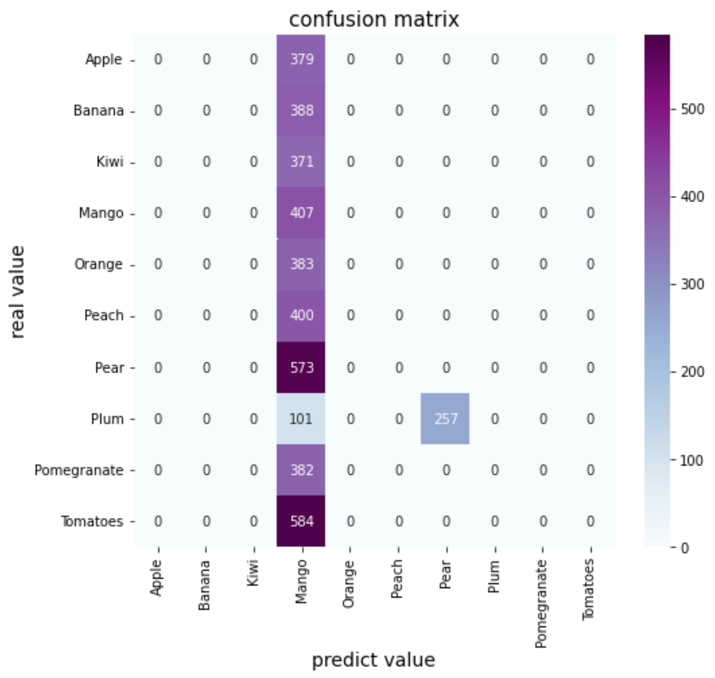
As mentioned above, one of the objectives of this project is to evaluate the upper limit of neural networks on image classification tasks and compare it with the performance of SVM. It is not surprising that the Resnet50 and mobilenet will have higher performance than SMV, however, the huge gap of performance between neural network and svm is unexpected. It might prove how superior the neural network is compared with SVM during complex missions.

Another objective of this project is to observe the upper limit of neural networks. As mentioned before, I intentionally take images of fruits with a pattern that a fruit is placed on a metal plate in order to create a relatively stable noise to my model. The result of model 1 and model 2 indicated that the neural network is an extremely strong classifier in image classification tasks when noise is stable and regular. Since my model is performing perfectly on validation data, the generalization of the model became another standard to evaluate my model.

Unfortunately, model 1 is overfitting based on its performance on the test dataset. After I trained model 1 which got 99% accuracy on validation dataset, I used another fruit data which has a completely different pattern of noise included to test how the generalization of the model performed. Here is some sample I used for testing:



As the sample performance, the noise has a different pattern of noise inside the test dataset. Here is the confusion matrix of model 1 predicting with test dataset:



	Apple	Banana	Kiwi	Mango	Orange	Peach	Pear	Plum	Pomegranate	Tomatoes
Apple	0	0	0	379	0	0	0	0	0	0
Banana	0	0	0	388	0	0	0	0	0	0
Kiwi	0	0	0	371	0	0	0	0	0	0
Mango	0	0	0	407	0	0	0	0	0	0
Orange	0	0	0	383	0	0	0	0	0	0
Peach	0	0	0	400	0	0	0	0	0	0
Pear	0	0	0	573	0	0	0	0	0	0
Plum	0	0	0	101	0	0	257	0	0	0
Pomegranate	0	0	0	382	0	0	0	0	0	0
Tomatoes	0	0	0	584	0	0	0	0	0	0

It is not surprising that the model has an overfitting problem. When I only feed the model with data which has some kind of pattern, I created human bias on the dataset which made the model trained with biased data. Such errors are lethal to generalization but there are plenty of ways to fix them such as data augmentation. For example, I can add preprocessing layers which perform operations like resizing, rescaling, random flip and random rotation can significantly mitigate the overfitting problem of human bias on dataset. Data augmentation is cheap and efficient in solving such problems.

From another perspective, data augmentation is a solution that balances the bias and variance of the model. It is a trade off that gives up some bias of the model in order to get a boost in variance of the model. However, there might be a different approach that does not need to do the trade off. For example, use different models to process different tasks and combine their results in some way. For example, I can split the task into two sub-task which identify where the fruit is and what the fruit is. And I can process these tasks with different models such as model for segmentation and model 1. In this case, it is possible that the segmentation model can solve the problem of generalization and model 1 can perform its best on classification missions.