Machine Learning Engineer Nanodegree

Capstone Project

Marwa Younes
November 13th, 2019

I. Definition

Project Overview

Classification of fruits is traditionally done using manual resources due to which the time and economic involvements increase adversely with number of fruit types and items per class. In recent times computer based automated techniques have been used to alleviate this problem to a certain extent.

These techniques utilize image analysis and pattern recognition methodologies to automatically classify fruits based on their visual features like color, texture, and shape. However, challenges of such techniques include the fact that fruit appearances differ due to natural environments, geographical locations, stages of growth, size, orientations and imaging equipment

dataset used in this project consist of 10 different classes of fruits

Problem Statement

Fruits have certain categories that is hard to differentiate, so the artificial intelligence is used to complete this hard mission which takes a lot of time and work

the problem is to Classify number of fruits up to 10 classes and the output will be the accuracy of our classification model

the problem will be solved using CNN, Stacked layers of conv, maxpooling, dropout to be used for pre-processing or feature generation. Then fit the model and validate the results.

Evaluation Metrics

Accuracy score :

the number of correct predictions to all the number of predictions made by the model

loss function :

categorical loss function due to use multi-classification

II. Analysis

Data Exploration

Exploratory Visualization

The data I used is a subset of fruit_360 dataset found in kaggle which is resized to 100*100 pixels

The data consist of 6512 images belong to 10 classes of fruits :

train data :

```
[7] # Read Training data
    X_train,train_label = read_data(train_data_dir,"RGB",(100,100))
    print(X_train.shape)
    print(train_label.shape)
```

```
(4876, 100, 100, 3)
(4876,)
```

test data :

```
[24] # Read Testing data
     X_test,test_label = read_data(test_data_dir,"RGB",(100,100))
     print(X_test.shape)
     print(test_label.shape)
```

```
(1636, 100, 100, 3)
(1636,)
```

classes :

```
['Apple',
 'Banana',
 'Cherry',
 'Guava',
 'Kiwi',
 'Lemon',
 'Mango',
 'Orange',
 'Pear',
 'Raspberry']
```

Random sample grid visualization

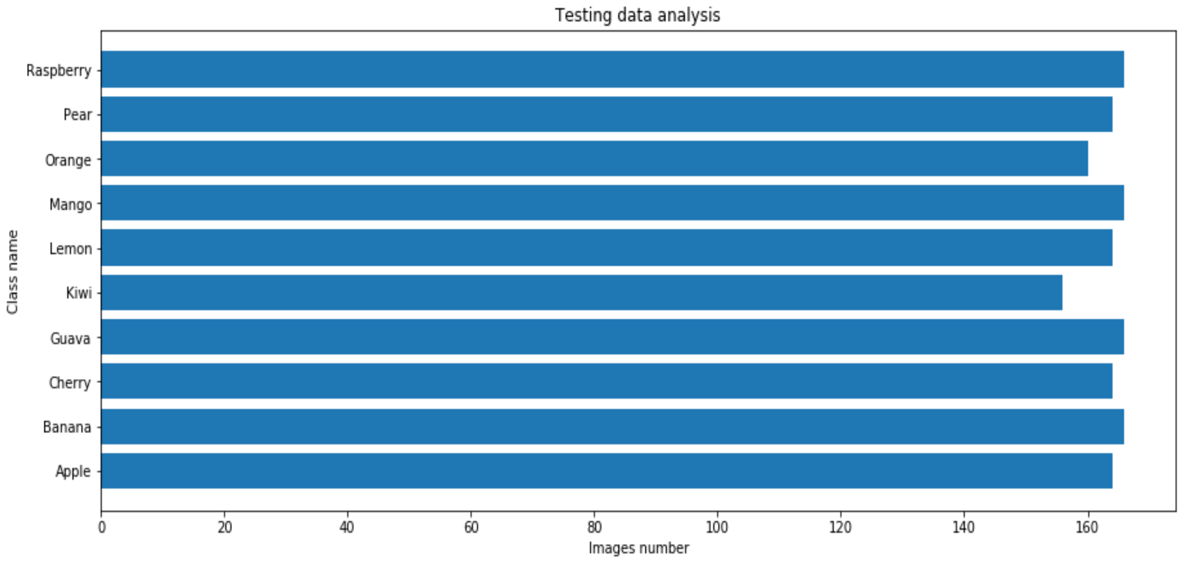
Data distribution bar plot :

Training data

<BarContainer object of 10 artists>

Test data :

<BarContainer object of 10 artists>



Algorithms and Techniques

- I will use CNN with typical structure that extract pattern and lower dimensionality
- CNN can detect patterns such as edges, shapes and particular characteristics
- by using filters to analyze images
- I will use image augmentation to increase accuracy
- then I will use pre-trained models that already trained over millions of image of different classes called xception model that would give higher accuracy

Benchmark

The benchmark model will be comparison between :

the first model and the next models with changes to improve accuracy

the final model with results from papers on the internet and other people on kaggle if it is available

III. Methodology

Data Preprocessing

- There is no need for cleaning data
- Data is loaded in RGB color
- Size of each image is 100*100 pixels
- Image is normalized to 0-1 range by dividing each pixel value by 255
- for data preprocessing I used ImageDataGenerator API from Keras

Data splitting

the training data contain 4876 images and split to 0.75 training and 0.25 validation

the testing data contain 1636 images

```
Found 3661 images belonging to 10 classes.
Found 1215 images belonging to 10 classes.
Found 1636 images belonging to 10 classes.
```

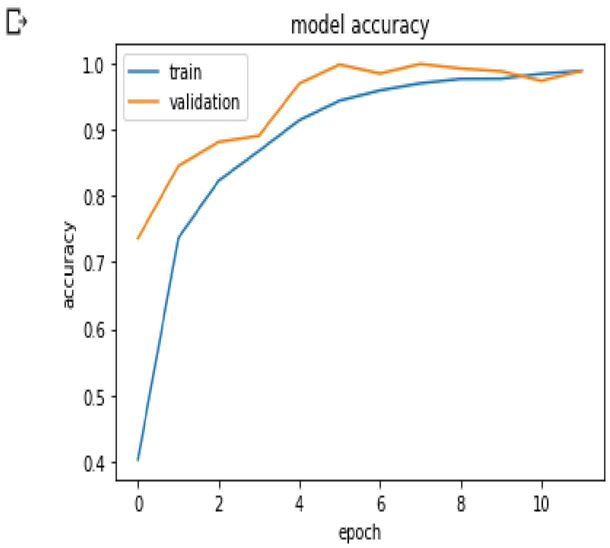
Implementation

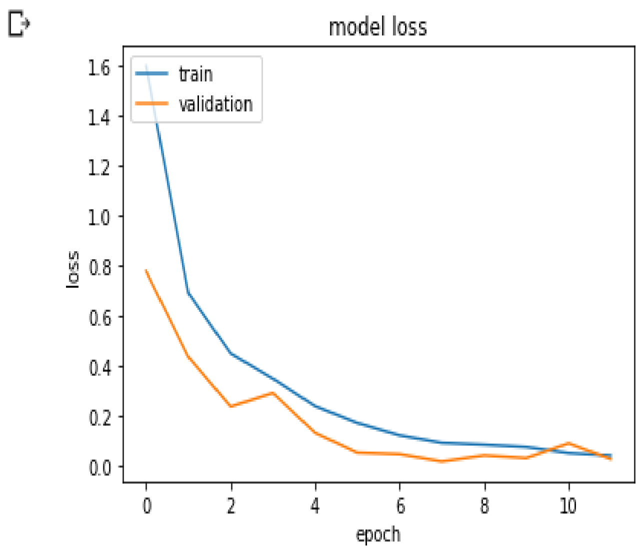
Basic CNN :

- I used CNN consist of 3 hidden layers each to detect different kinds of features each one has a following max pooling for decreasing dimensionality
- followed by global average pooling layer to minimize overfitting by reducing the total number of parameter in the model
- ending with dense layer with softmax activation function to get classification results

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 100, 100, 16)      208
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 50, 50, 16)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 50, 50, 32)        2080
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 25, 25, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 25, 25, 64)        8256
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
global_average_pooling2d_2 ( (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 11,194
Trainable params: 11,194
Non-trainable params: 0
_________________________________________________________________
```

- I used 50 epochs of training and early stopping to lower training time if there is no improvements
- used checkpointer to monitor the validation set loss and save the weights of the model that gave best loss value
- RMSProp. Optimizer is used





for testing dataset :

```
[ ] print("Basic Model predicted {} images right,accuracy = {}%".format(right_pred,(right_pred*100./1636)))
```

```
Basic Model predicted 1567 images right,accuracy = 95.78239608801955%
```

for the data I gathered :

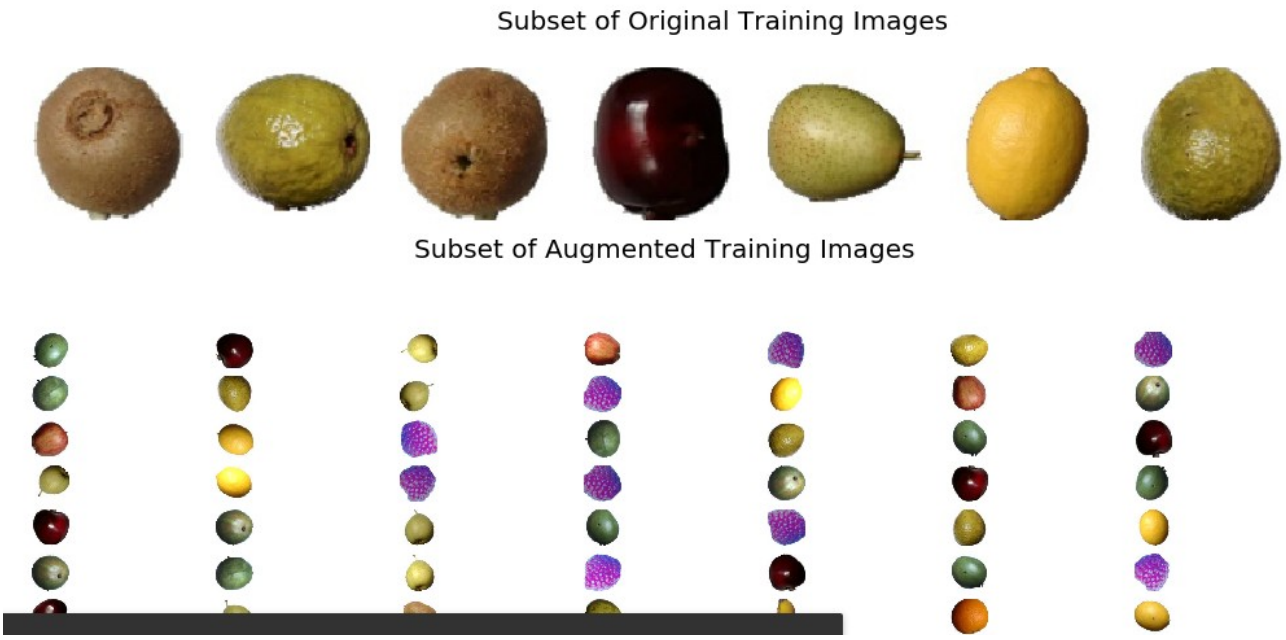
```
Basic Model predicted 17 images right,accuracy = 47.2222222222222%
```

Refinement

image augmentation

to overcome the problem of limited diversity of data, we can generate our own data with the existing data which we have

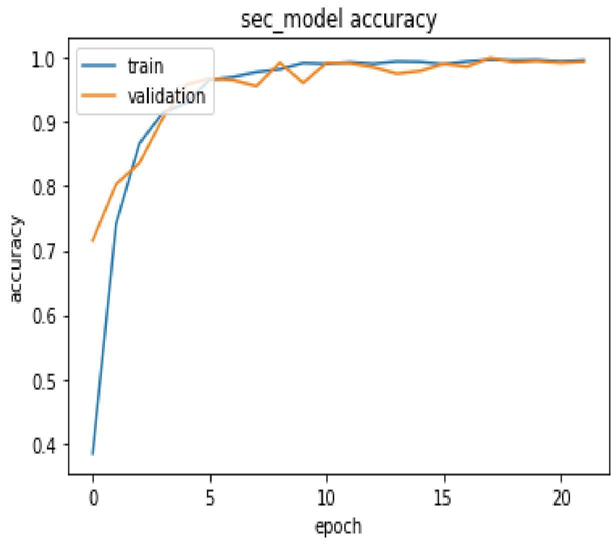
this methodology is called image augmentation visualization of augmented data

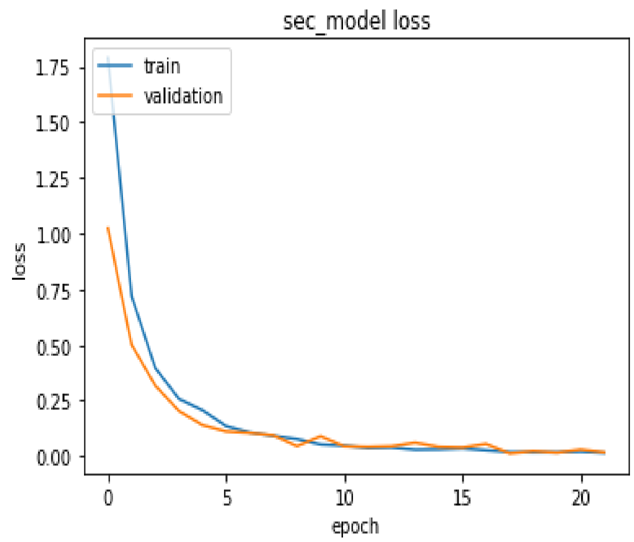


I used the same CNN structure

```
sec_model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_19 (Conv2D)           (None, 100, 100, 16)      208
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 50, 50, 16)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 49, 49, 32)        2080
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 23, 23, 64)        8256
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 11, 11, 64)        0
_________________________________________________________________
global_average_pooling2d_7 ( (None, 64)                0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                650
=================================================================
Total params: 11,194
Trainable params: 11,194
Non-trainable params: 0
_________________________________________________________________
```





notice that with image augmentation we have more number of epochs needed to gain stable value and that makes sense since we have bigger data than before so more learning time and steps is needed

for testing dataset :

```
print("Model after augmenting Images predicted {} images right,accuracy = {}%".format(right_pred,(right_pred*100./1636)))
Model after augmenting Images predicted 1636 images right,accuracy = 100.0%
```

for the data I gathered :

```
Model after augmenting Images predicted 19 images right,accuracy = 52.77777777777778%
```

Transfer learning

- we take pre-trained model which the weights and parameters of network that has been trained on large dataset before and fine-tune the model with our own dataset
- I used Xception as the pre-trained model since it achieved a good accuracy and it has good depth structure and parameters amount
- the top layer is removed and I added one global average layer followed with dense layer with softmax activation function for final classification
- Adam optimizer is used

IV. Results

Model Evaluation and Validation

My final model :

- I consider Xception the pre-trained model as my final model
- the top layer is removed and I added one global average layer followed with dense layer with softmax activation function for final classification
- Adam optimizer is used
- I used 50 epochs of training and early stopping to lower training time if there is no improvements
- used checkpointer to monitor the validation set loss and save the weights of the model that gave best loss value
- used image augmentation to overcome the problem of limited diversity of data

```
[ ] xception_transfer.evaluate_generator(test_generator, test_generator.samples)

[0.0006707814851110329, 1.0]
```

for the data I gathered :

```
Model predicted 13 images right,accuracy = 36.11111111111114%
```

Justification

My final result on the testset :

basic CNN model : 95.27%

basic CNN model with image augmentation : 100 %

transfer learning (pre-trained model) with image augmentation : 100 %

the results of benchmark is specified according the whole data (120 classes) and its accuracy was 96.13%

on this paper :

https://www.researchgate.net/publication321475443_Fruit_recognition_from_images_using_deep_learning https://www.researchgate.net/figure/Results-of-training-the-neural-network-on-the-fruits-360-dataset_tbl5_321475443

My final result on the data I gathered :

basic CNN model : 47.2%

basic CNN model with image augmentation : 52.7 %

transfer learning (pre-trained model) with image augmentation : 36.1 %

the accuracy on the data I gathered is lower than that of test set due to the difference between them

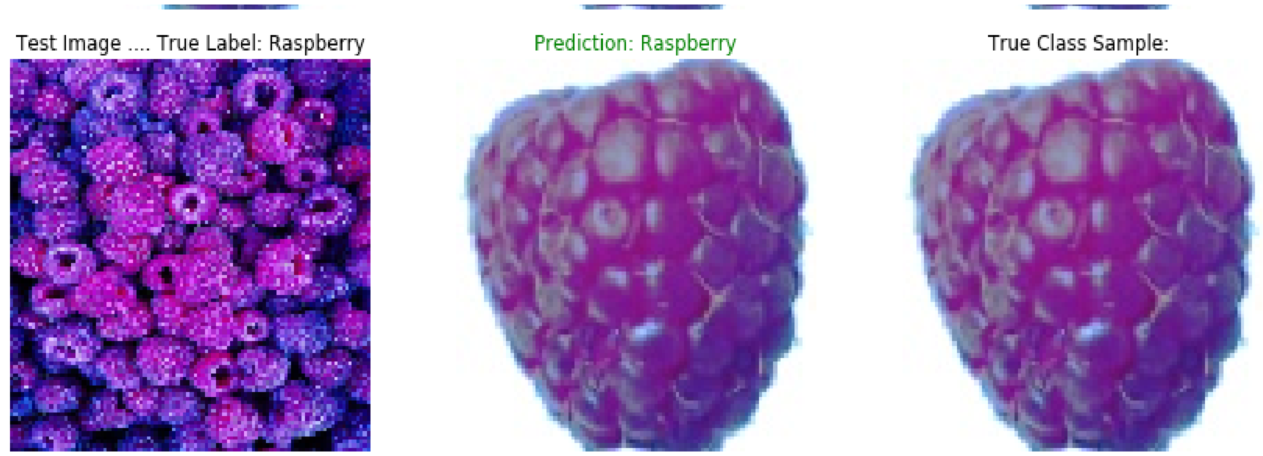
the real data have one or more from the same fruit which is different from the test date that have only one fruit per image

I think the model is significant enough for this problem's solution according to this dataset

V. Conclusion

Free-Form Visualization

Let's check some images and its actual label and predicted probability as below :



Reflection

- The process used for this project can be summarized using the following steps:
- Get data from Kaggle and the dataset was good in size and quality.
- Read data and preprocess it to RGB and normalize it to 0-1 range
- split the data set to train, validation and test sets
- use deep learning and made my basic CNN model
- Use image augmentation to transform the image and prepare for training.
- Use transfer learning and used Xception model
- test the data and get the accuracy and loss vector at the last three steps
- Finally predict for real test data I gathered from google and get the accuracy

the difficult aspect was the weakness of data set :

there is a huge difference between the data and the real fruits in color and shape since the data consist of classes each class is a fruit and rotated on a motor so the model memorize its shape and color and could not be able to recognize any other shapes or color

Improvement

- try to improve the data set by adding diverse images of the class and use data with more good pixel size
- try more than one architecture of CNN to improve accuracy
- use other transfer learning models like Inception V2, Inception V3 , VGG16 or VGG19