

# Image Classification Using CNN - Fruits

Yongjun Kim

AI20216801

# Introduction

- ▶ First examine the fruits
- ▶ Look at it as pictures and numbers of pictures
- ▶ Create CNN model
- ▶ Evaluate the results of the model created and examine them on the graph

# Dataset




































































































































- ▶ Total number of images: 90,483
- ▶ Training set size: 67,692 images (one fruit or vegetable per image)
- ▶ Test set size: 22,688 images (one fruit or vegetable per image)
- ▶ Number of classes: 131 (fruits and vegetables)
- ▶ Image size: 100 \* 100 pixels

# List of Training Dataset

	Fruits	Fruits Image
0	Apple Braeburn	Apple Braeburn/0_100.jpg
1	Apple Braeburn	Apple Braeburn/100_100.jpg
2	Apple Braeburn	Apple Braeburn/101_100.jpg
3	Apple Braeburn	Apple Braeburn/102_100.jpg
4	Apple Braeburn	Apple Braeburn/103_100.jpg
...	...	...
67687	Watermelon	Watermelon/r_6_100.jpg
67688	Watermelon	Watermelon/r_7_100.jpg
67689	Watermelon	Watermelon/r_81_100.jpg
67690	Watermelon	Watermelon/r_8_100.jpg
67691	Watermelon	Watermelon/r_9_100.jpg

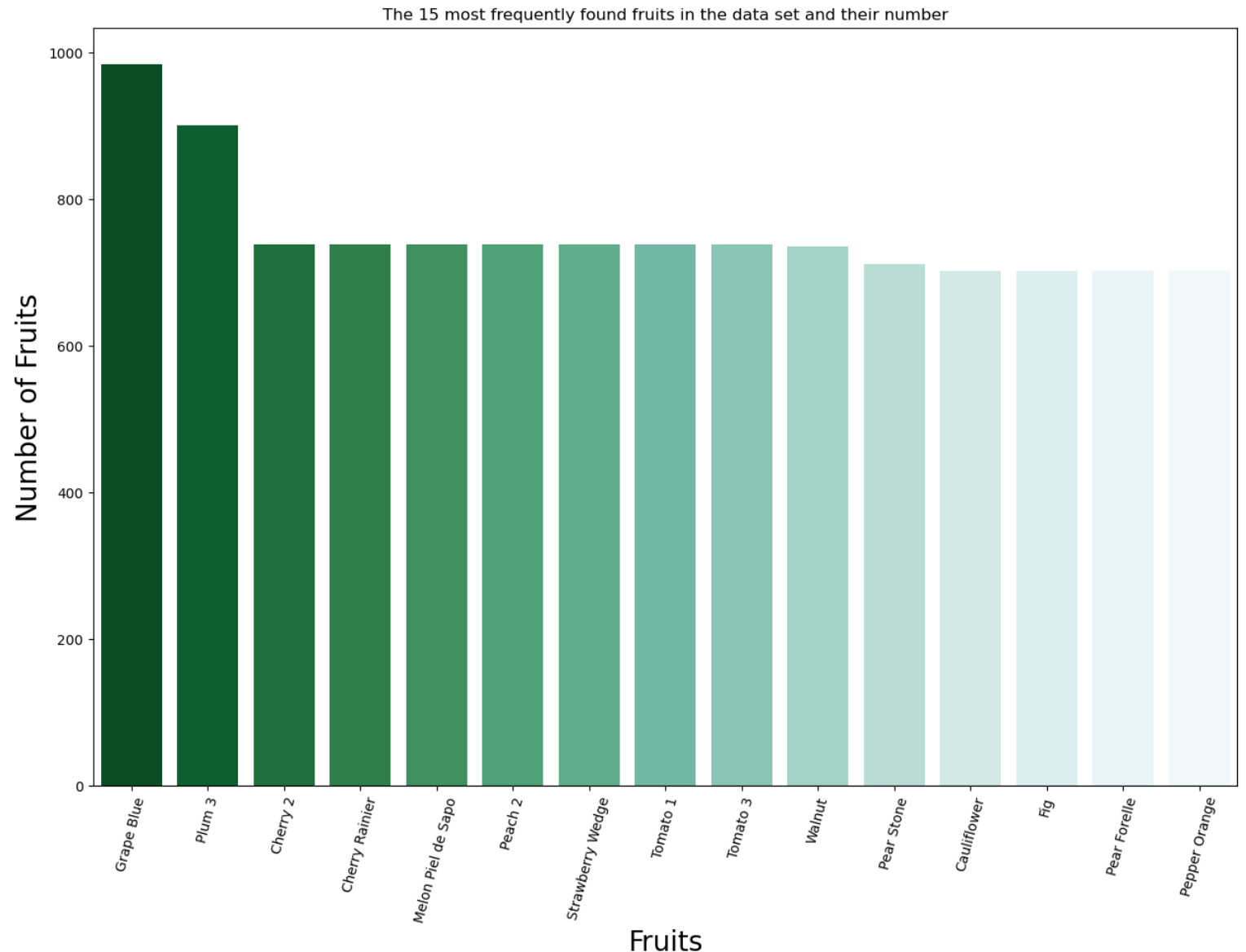
67692 rows × 2 columns

# List of Dataset

																
Apple Braeburn	Apple Crimson Snow	Apple Golden 1	Apple Golden 2	Apple Golden 3	Apple Granny Smith	Apple Pink Lady	Apple Red 1	Apple Red 2	Apple Red 3	Apple Red Delicious	Apple Red Yellow 1	Apple Red Yellow 2	Apricot	Avocado	Avocado ripe	Banana
																
Banana Lady Finger	Banana Red	Beetroot	Blueberry	Cactus fruit	Cantaloupe 1	Cantaloupe 2	Carambola	Cauliflower	Cherry 1	Cherry 2	Cherry Rainier	Cherry Wax Black	Cherry Wax Red	Cherry Wax Yellow	Chestnut	Clementine
																
Cocos	Corn	Corn Husk	Cucumber Ripe	Cucumber Ripe 2	Dates	Eggplant	Fig	Ginger Root	Granadilla	Grape Blue	Grape Pink	Grape White	Grape White 2	Grape White 3	Grape White 4	Grapefruit Pink
																
Grapefruit White	Guava	Hazelnut	Huckleberry	Kaki	Kiwi	Kohlrabi	Kumquats	Lemon	Lemon Meyer	Limes	Lychee	Mandarine	Mango	Mango Red	Mangostan	Maracuja
																
Melon Piel de Sapo	Mulberry	Nectarine	Nectarine Flat	Nut Forest	Nut Pecan	Onion Red	Onion Red Peeled	Onion White	Orange	Papaya	Passion Fruit	Peach	Peach 2	Peach Flat	Pear	Pear 2
																
Pear Abate	Pear Forelle	Pear Kaiser	Pear Monster	Pear Red	Pear Stone	Pear Williams	Pepino	Pepper Green	Pepper Orange	Pepper Red	Pepper Yellow	Physalis	Physalis with Husk	Pineapple	Pineapple Mini	Pitahaya Red
																
Plum	Plum 2	Plum 3	Pomegranate	Pomelo Sweetie	Potato Red	Potato Red Washed	Potato Sweet	Potato White	Quince	Rambutan	Raspberry	Redcurrant	Salak	Strawberry	Strawberry Wedge	Tamarillo
																
Tangelo	Tomato 1	Tomato 2	Tomato 3	Tomato 4	Tomato Cherry Red	Tomato Heart	Tomato Maroon	Tomato not Ripened	Tomato Yellow	Walnut	Watermelon					

# The 15 most frequently found fruits

```
[('Grape Blue', 984),  
 ('Plum 3', 900),  
 ('Cherry 2', 738),  
 ('Cherry Rainier', 738),  
 ('Melon Piel de Sapo', 738),  
 ('Peach 2', 738),  
 ('Strawberry Wedge', 738),  
 ('Tomato 1', 738),  
 ('Tomato 3', 738),  
 ('Walnut', 735),  
 ('Pear Stone', 711),  
 ('Cauliflower', 702),  
 ('Fig', 702),  
 ('Pear Forelle', 702),  
 ('Pepper Orange', 702)]
```



# The 15 most frequently found fruits

## The 15 Most Abundant Fruits

GRAPE BLUE



PLUM 3



CHERRY 2



CHERRY RAINIER



MELON PIEL DE SAPO



PEACH 2



STRAWBERRY WEDGE



TOMATO 1



TOMATO 3



WALNUT



PEAR STONE



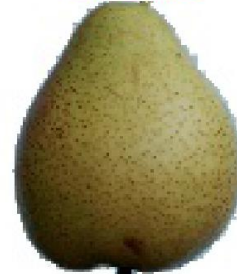
CAULIFLOWER



FIG



PEAR FORELLE



PEPPER ORANGE



# Libraries

- ▶ Numpy for Fundamental scientific computations
- ▶ Pandas for use open source data analysis and manipulation
- ▶ Matplotlib for data visualization
- ▶ Keras for High level neural network library

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
from glob import glob
import cv2

# import model
from keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense

# import warnings
import warnings
warnings.filterwarnings('ignore')

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all

import os
for dirname, _, filenames in os.walk('.\\#fruit#fruits-360_dataset#fruits-360'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```



# Image Shape

- ▶ Shape of the pictures
- ▶ This is important because this will be the input value of model
- ▶ Height is 100, width is 100, channel is 3

```
array_image = img_to_array(img)  
print("Image Shape --> ", array_image.shape)
```

```
Image Shape --> (100, 100, 3)
```

# Model

- ▶ Used Conv2D to create convolution filter
- ▶ Filter Size is 32, 32, 64
- ▶ Kernel Size is 3 \* 3
- ▶ RELU(Activation Function) is used as an activation function to add non-linearity
- ▶ MaxPooling2D is reducing the amount of parameters and computation in the network

```
model = Sequential()  
model.add(Conv2D(32, (3, 3), input_shape = array_image.shape))  
model.add(Activation("relu"))  
model.add(MaxPooling2D())  
  
model.add(Conv2D(32, (3, 3)))  
model.add(Activation("relu"))  
model.add(MaxPooling2D())  
  
model.add(Conv2D(64, (3, 3)))  
model.add(Activation("relu"))  
model.add(MaxPooling2D())  
  
model.add(Flatten())  
model.add(Dense(1024))  
model.add(Activation("relu"))  
model.add(Dropout(0.5))  
model.add(Dense(numberOfClass)) # output  
model.add(Activation("softmax"))
```

# Compile Model, Epochs and Batch Size

- ▶ For a binary classification like this example, the typical loss function is the binary cross-entropy/loss
- ▶ Epoch is when an entire dataset is passed forward and backward through the neural network only once
- ▶ Since one epoch is too big to feed to the computer at once, divide it in several smaller batches

```
model.compile(loss = "categorical_crossentropy",  
              optimizer = "rmsprop",  
              metrics = ["accuracy"])
```

```
epochs = 100  
batch_size = 32
```

# Data Augmentation

- ▶ Target size is 100 \* 100
- ▶ Use ImageDataGenerator to create train\_datagen and test\_datagen from image data augmentation
- ▶ Use as training data and test data

```
print("Target Size --> ", array_image.shape[:2])
```

Target Size --> (100, 100)

```
train_datagen = ImageDataGenerator(rescale= 1./255,  
                                   shear_range = 0.3,  
                                   horizontal_flip=True,  
                                   zoom_range = 0.3)
```

```
test_datagen = ImageDataGenerator(rescale= 1./255)
```

```
train_generator = train_datagen.flow_from_directory(  
    train_dir,  
    target_size= array_image.shape[:2],  
    batch_size = batch_size,  
    color_mode= "rgb",  
    class_mode= "categorical")
```

Found 67692 images belonging to 131 classes.

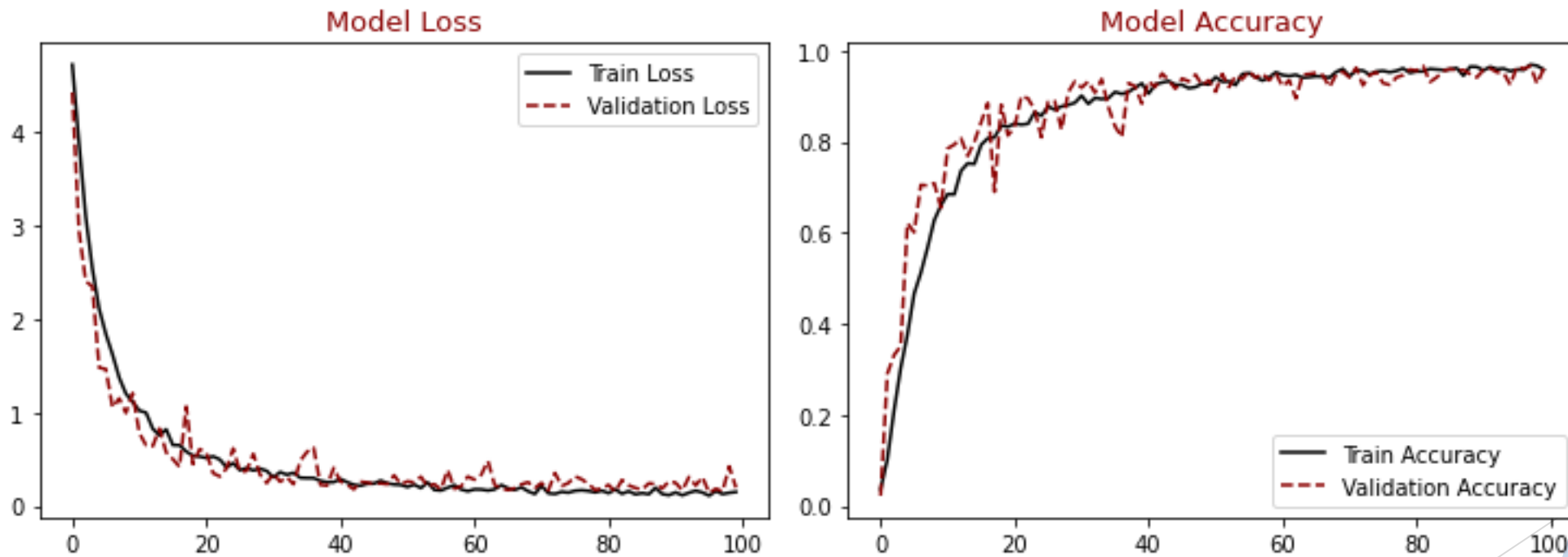
```
test_generator = test_datagen.flow_from_directory(  
    test_dir,  
    target_size= array_image.shape[:2],  
    batch_size = batch_size,  
    color_mode= "rgb",  
    class_mode= "categorical")
```

Found 22688 images belonging to 131 classes.

# Fit model

```
Epoch 1/100  
50/50 [=====] - 16s 194ms/step - loss: 4.6894 - accuracy: 0.0300 - val_loss: 4.1474 - val_accuracy: 0.0850  
Epoch 2/100  
50/50 [=====] - 10s 189ms/step - loss: 3.8770 - accuracy: 0.0969 - val_loss: 3.1949 - val_accuracy: 0.1937  
Epoch 3/100  
50/50 [=====] - 9s 187ms/step - loss: 3.1221 - accuracy: 0.1931 - val_loss: 2.6465 - val_accuracy: 0.3350  
Epoch 100/100  
50/50 [=====] - 4s 82ms/step - loss: 0.1741 - accuracy: 0.9538 - val_loss: 0.3945 - val_accuracy: 0.9225
```

# Result - Model Loss, Model Accuracy



# Conclusion

- ▶ Solved fruits classification problem using Machine Learning method - CNN approach with 0.1741 loss and 0.9538 accuracy at 100 epoch

# References

- ▶ <https://www.kaggle.com/rafetcan/image-classification-using-cnn-fruits>
- ▶ <https://www.kaggle.com/etatbak/cnn-fruit-classification>
- ▶ [https://github.com/yungbyun/pr/blob/main/02.Subhajit\\_Flowers%20recognition.pdf](https://github.com/yungbyun/pr/blob/main/02.Subhajit_Flowers%20recognition.pdf)
- ▶ [https://github.com/yungbyun/pr/blob/main/02\\_Faiza\\_Qayyum\\_covid%20detection.pdf](https://github.com/yungbyun/pr/blob/main/02_Faiza_Qayyum_covid%20detection.pdf)



Thank you