Fruit Classifier

Group 6

E/16/096 - Ekanayake J.E.M.D.Y.
E/16/290 - Prabodha U.A.K.
E/16/267 - Parackrama G.T.W.
E/16/126 - Hansika D.G.N.

Problem

- Recognizing different kinds of fruits is a recurring task in supermarkets.
- The cashier or the customer need to manually identify the class of product being bought and look for it in the system for payment process.

Solution

- Our focus is to design a model using neural networks that can be used to recognize different types of fruit and classification.
- The goal of this project is to speed up the checkout process in stores.

Why CNN ?

- CNN : convolutional neural network
- Very effective in reducing the number of parameters without losing on the quality of models.
- Shows high accuracy in image classification and recognition
- Specifically designed to process pixel data.
- Uses a system much like a multilayer perceptron that has been designed for reduced processing requirements.

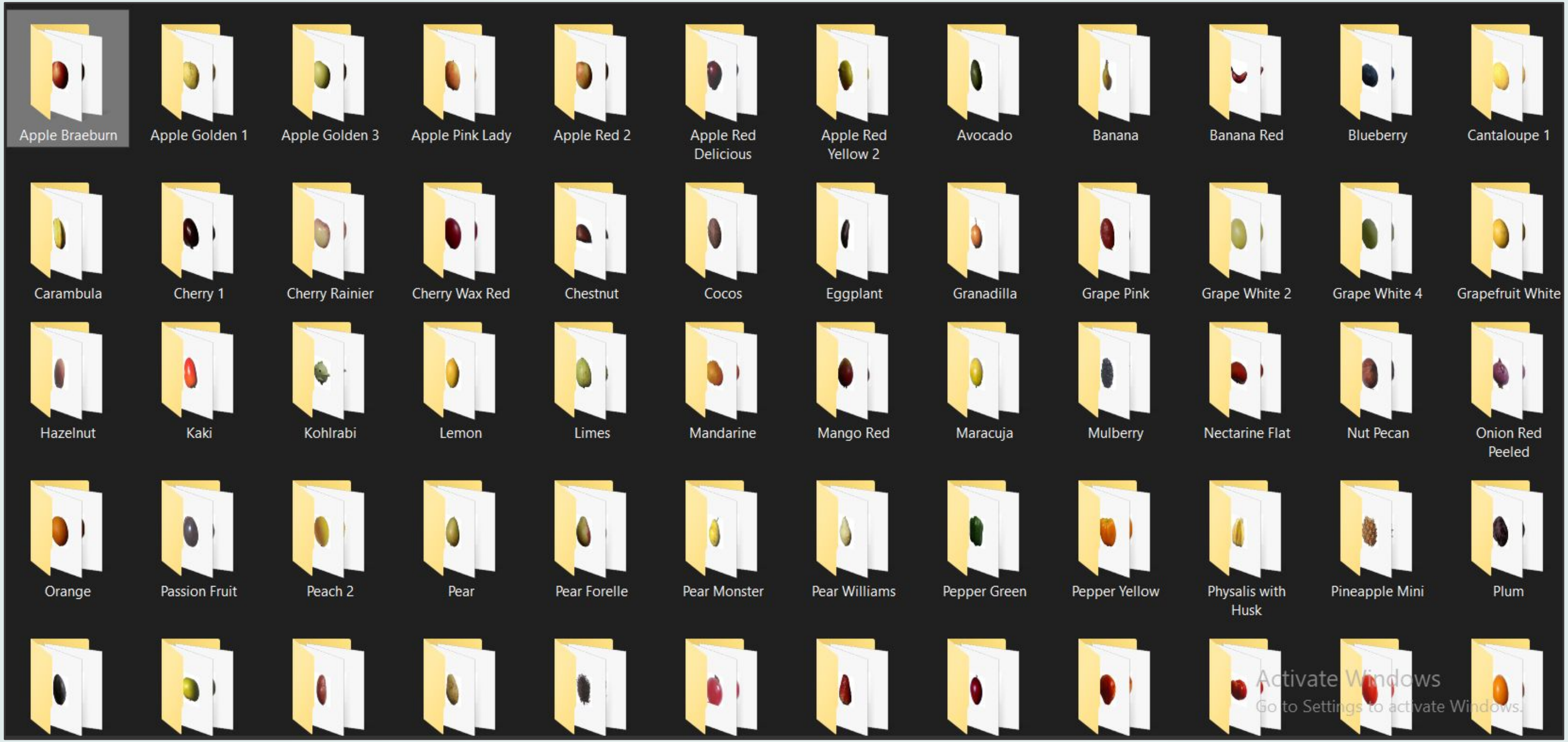
Dataset

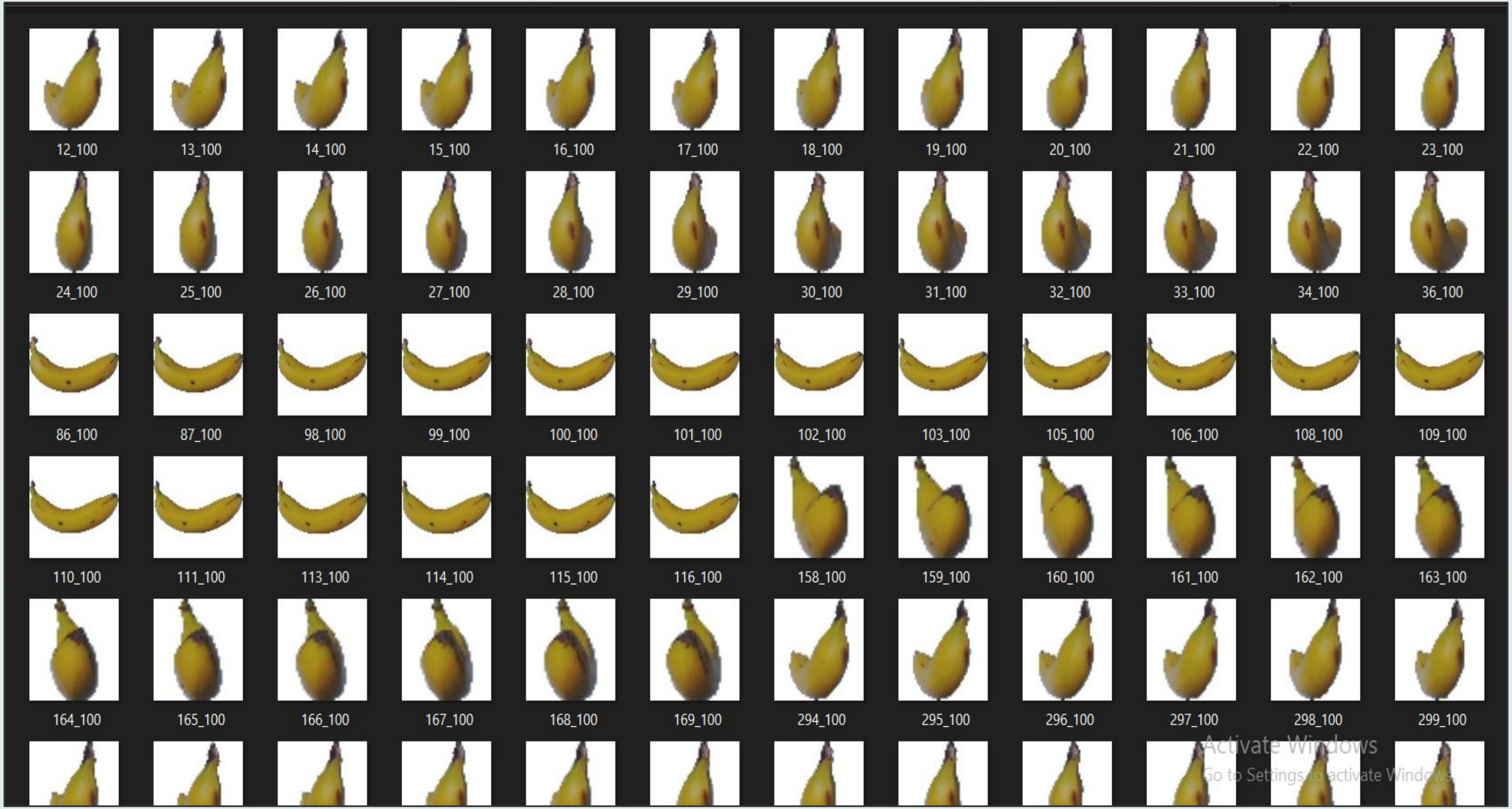
Dataset : Small fruit

Characteristics of the data:

- Total number of images: 40800 images
- Training set size: 30468 images
- Test set size: 10332 images
- Number of classes: 60 different fruits
- Image size: 100 x 100 pixels, RGB.

Link for the dataset: http://vegesm.web.elte.hu/


Apple Braeburn
Apple Golden 1
Apple Golden 3
Apple Pink Lady
Apple Red 2
Apple Red Delicious
Apple Red Yellow 2
Avocado
Banana
Banana Red
Blueberry
Cantaloupe 1
Carambula
Cherry 1
Cherry Rainier
Cherry Wax Red
Chestnut
Cocos
Eggplant
Granadilla
Grape Pink
Grape White 2
Grape White 4
Grapefruit White
Hazelnut
Kaki
Kohlrabi
Lemon
Limes
Mandarine
Mango Red
Maracuja
Mulberry
Nectarine Flat
Nut Pecan
Onion Red Peeled
Orange
Passion Fruit
Peach 2
Pear
Pear Forelle
Pear Monster
Pear Williams
Pepper Green
Pepper Yellow
Physalis with Husk
Pineapple Mini
Plum
Activate Windows
Go to Settings to activate Windows.


12_100
13_100
14_100
15_100
16_100
17_100
18_100
19_100
20_100
21_100
22_100
23_100
24_100
25_100
26_100
27_100
28_100
29_100
30_100
31_100
32_100
33_100
34_100
36_100
86_100
87_100
98_100
99_100
100_100
101_100
102_100
103_100
105_100
106_100
108_100
109_100
110_100
111_100
113_100
114_100
115_100
116_100
158_100
159_100
160_100
161_100
162_100
163_100
164_100
165_100
166_100
167_100
168_100
169_100
294_100
295_100
296_100
297_100
298_100
299_100
Activate Windows
Go to Settings to activate Windows.

Procedure

1. Dataset preparations
2. Test the dataset setup
3. Splitting the dataset
4. Preprocess the dataset
5. Training the network
6. Plot the accuracy and loss over time, both for training and validation data
7. Calculate the performance of our model on the test set

Procedure...

1. Dataset preprocessing
   a. Augment the data
      - flip images horizontally
      - rotates them
      - performs zooming
   b. Split dataset as train ,test and validation

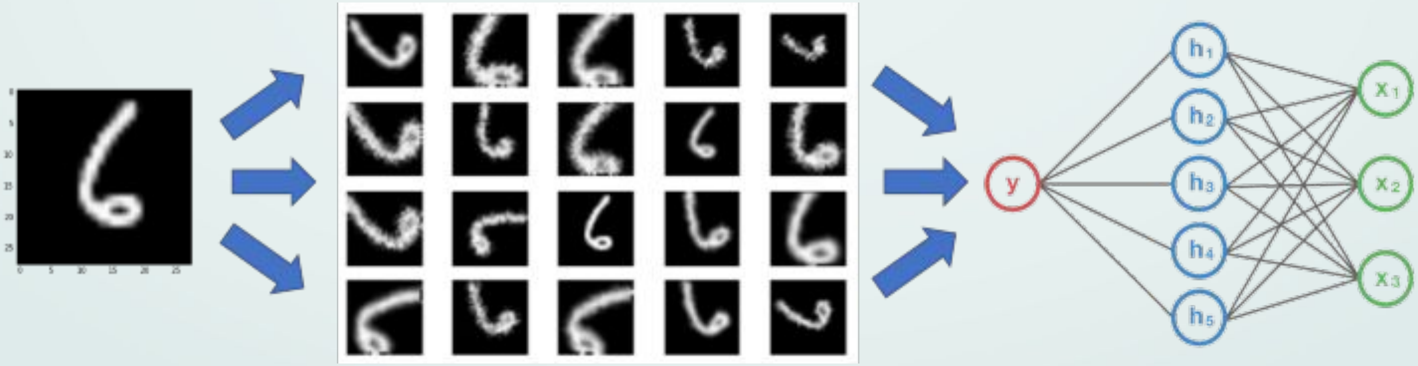
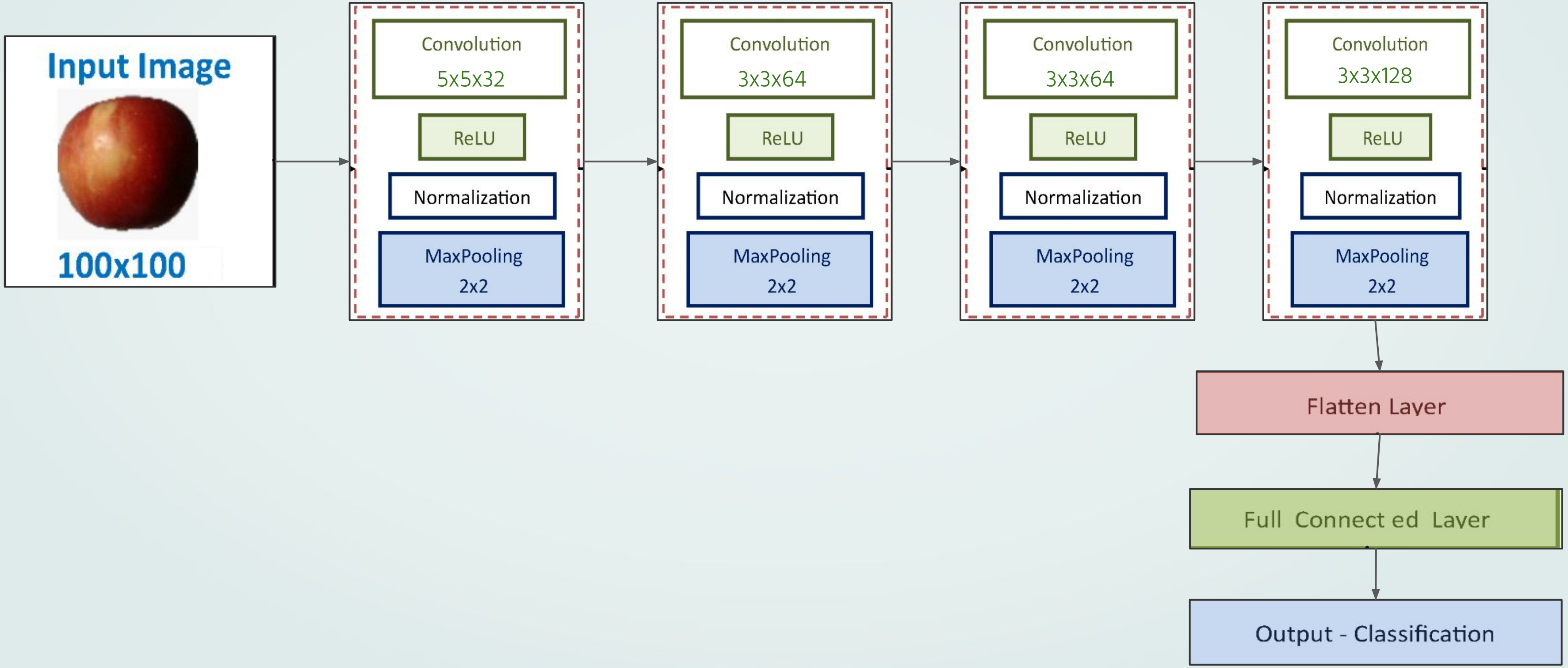


Image from: [Data Augmentation | How to use Deep Learning when you have Limited Data (nanonets.com)]


Input Image
100x100
Convolution
5x5x32
ReLU
Normalization
MaxPooling
2x2
Convolution
3x3x64
ReLU
Normalization
MaxPooling
2x2
Convolution
3x3x64
ReLU
Normalization
MaxPooling
2x2
Convolution
3x3x128
ReLU
Normalization
MaxPooling
2x2
Flatten Layer
Full Connect ed Layer
Output - Classification

Analysis

Activation Function

➔ **Used - ReLU function**

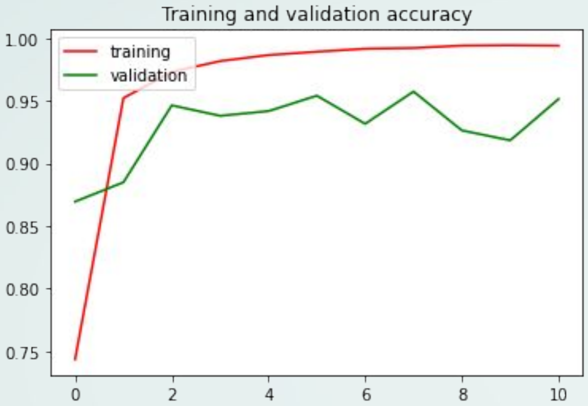
- Less computationally expensive
- Involves simpler mathematical operations.
- Does not have the vanishing gradient problem.

➔ **Tested with - tanh function**

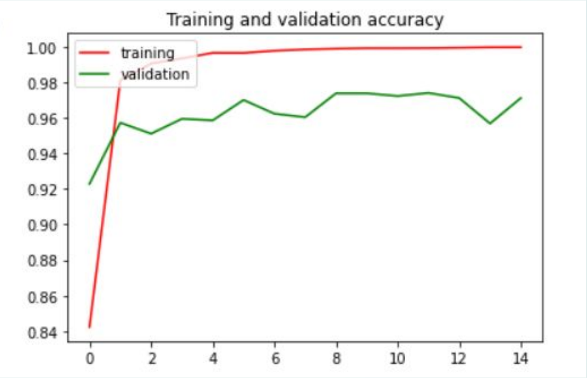
- Hyperbolic tangent activation function
- Vanishing gradient problem
- Results will be showed…

Tanh

ReLU

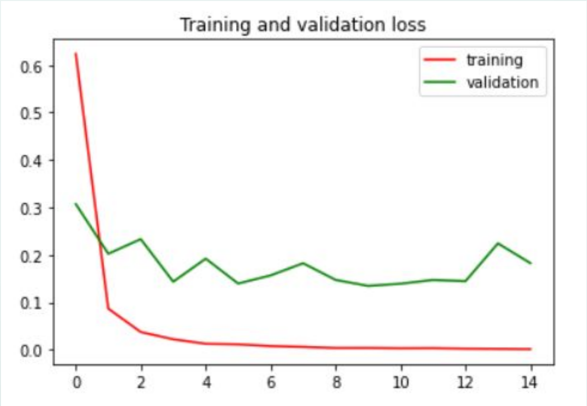


Training and validation accuracy
training
validation



Training and validation accuracy
training
validation

Training and validation loss
training
validation



Training and validation loss
training
validation

ReLu

```
loss,acc = model.evaluate_generator(generator=test_images_iter, steps=500, use_multiprocessing=False)
print('test loss: {}'.format(loss))
print('test accuracy: {:.2%}'.format(acc))
```

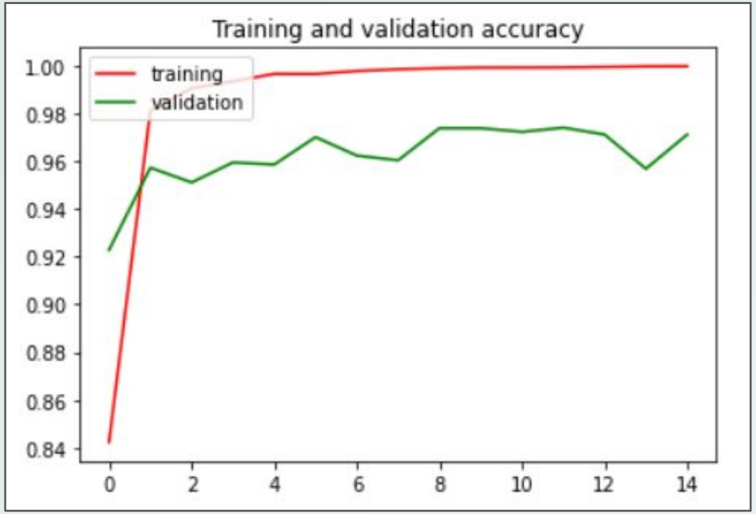
```
test loss: 0.09727661704928148
test accuracy: 98.09%
```

tanh

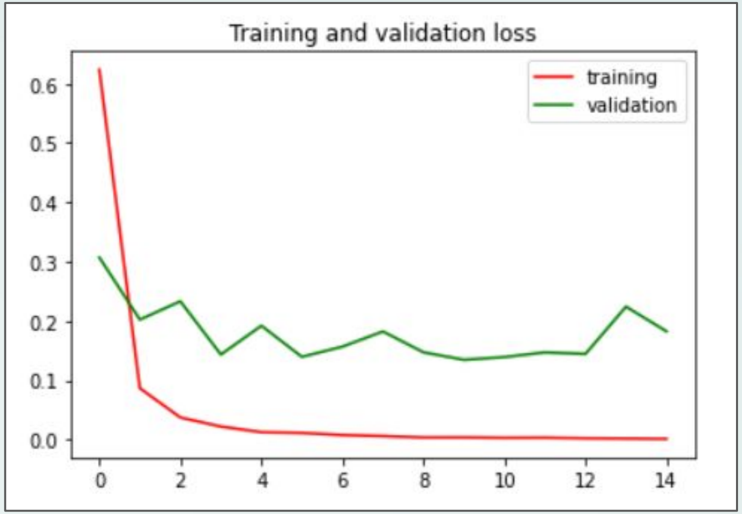
```
loss,acc = model.evaluate_generator(generator=test_images_iter, steps=500, use_multiprocessing=False)
print('test loss: {}'.format(loss))
print('test accuracy: {:.2%}'.format(acc))
```

```
test loss: 0.22262319467180006
test accuracy: 96.05%
```

Results



(a)



(b)

```
loss,acc = model.evaluate_generator(generator=test_images_iter, steps=500, use_multiprocessing=False)
print('test loss: {}'.format(loss))
print('test accuracy: {:.2%}'.format(acc))
```

```
test loss: 0.09727661704928148
test accuracy: 98.09%
```

(c)

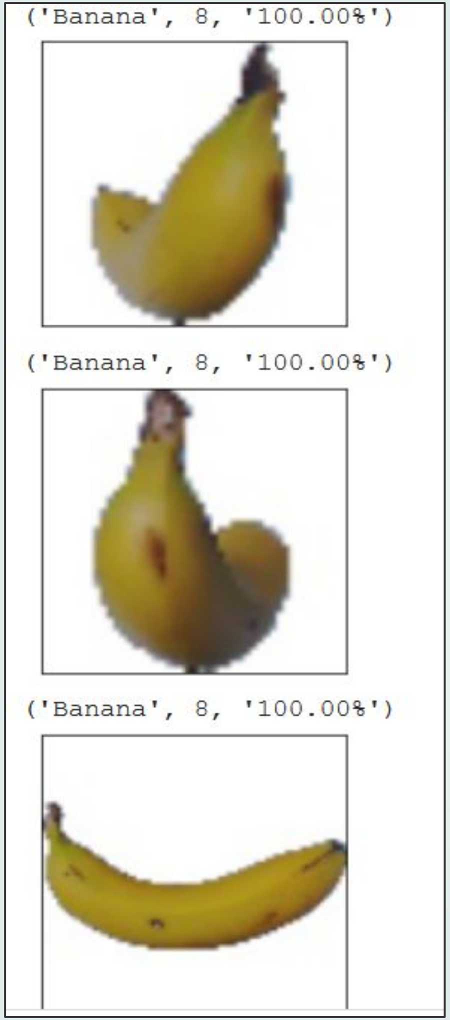
```python
#Checking the test accuracy
List=['/content/fruits-small/test/Tamarillo/22_100.jpg',
      '/content/fruits-small/test/Plum/38_100.jpg',
      '/content/fruits-small/test/Potato White/109_100.jpg',
      '/content/fruits-small/test/Pineapple Mini/122_100.jpg']

for i in List:

  loaded_image = keras.preprocessing.image.load_img(path=i, target_size=(100,100,3))
  img_array = keras.preprocessing.image.img_to_array(loaded_image) / 255
  imag_array = np.expand_dims(img_array, axis = 0)
  predictions = model.predict(imag_array)
  classidx = np.argmax(predictions[0])
  label = list(train_generator_iter.class_indices.keys())[classidx]
  pred= ["{:.2f}%".format(j * 100) for j in predictions[0] ]
  print((label, classidx, pred[classidx]))

  plt.figure(figsize=(3,3))
  plt.imshow(img_array)
  #title=a[i]+": "+b[ran]
  #plt.title(title)
  plt.xticks([])
  plt.yticks([])
  plt.show()
```

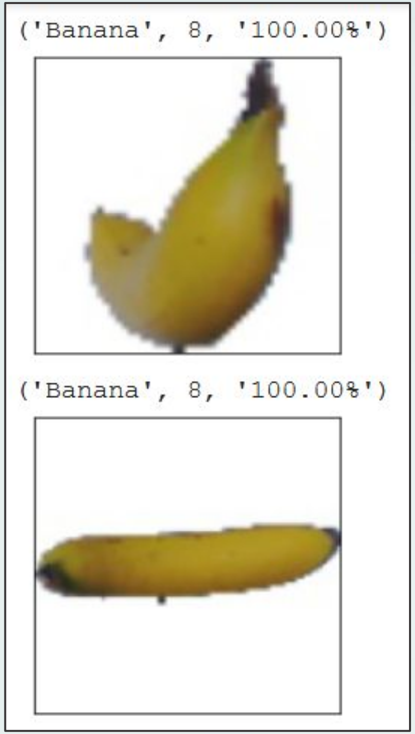
```
('Tamarillo', 55, '100.00%')
```

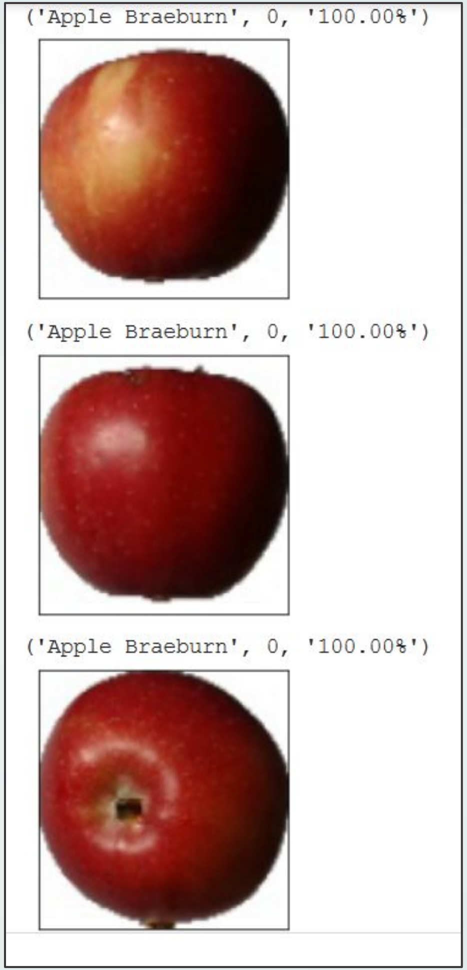
```
('Plum', 47, '100.00%')
```

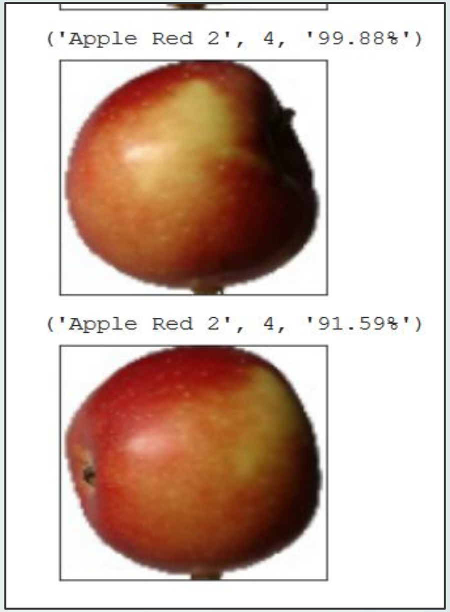
```
('Potato White', 51, '100.00%')
```

```
('Pineapple Mini', 46, '100.00%')
```


('Banana', 8, '100.00%')
('Banana', 8, '100.00%')
('Banana', 8, '100.00%')


('Banana', 8, '100.00%')
('Banana', 8, '100.00%')


('Apple Braeburn', 0, '100.00%')
('Apple Braeburn', 0, '100.00%')
('Apple Braeburn', 0, '100.00%')


('Apple Red 2', 4, '99.88%')
('Apple Red 2', 4, '91.59%')

('Grape Pink', 20, '99.99%')

('Grape White 2', 21, '100.00%')

('Grape White 4', 22, '100.00%')

('Grapefruit White', 23, '76.29%')

Challenges Faced

- Couldn’t get the expected Dataset, fruits 360 due to Server Error in Kaggle

- When importing fruit 360 for colab some error occurred due to large size of the data set.

  Because of that we change the dataset to fruit small dataset (which is in small size and have accurate dataset for our training)

- When using Jupyter Notebook in local PC,it took lot of execution time when training data there

  we used Colab

  (But sometime it also got crashed)

- Error when running codes

  Was able to handle after referring Some Docs

THANK YOU

Do you have any questions ?

```
1729/1729 [==============================] - 130s 75ms/step - loss: 0.6235 - acc: 0.8423 - val_loss: 0.3070 - val_acc: 0.9228
Epoch 2/15
1729/1729 [==============================] - 117s 68ms/step - loss: 0.0865 - acc: 0.9810 - val_loss: 0.2022 - val_acc: 0.9572
Epoch 3/15
1729/1729 [==============================] - 118s 68ms/step - loss: 0.0371 - acc: 0.9905 - val_loss: 0.2330 - val_acc: 0.9510
Epoch 4/15
1729/1729 [==============================] - 117s 68ms/step - loss: 0.0219 - acc: 0.9934 - val_loss: 0.1432 - val_acc: 0.9594
Epoch 5/15
1729/1729 [==============================] - 117s 68ms/step - loss: 0.0124 - acc: 0.9965 - val_loss: 0.1918 - val_acc: 0.9585
Epoch 6/15
1729/1729 [==============================] - 117s 68ms/step - loss: 0.0111 - acc: 0.9965 - val_loss: 0.1394 - val_acc: 0.9700
Epoch 7/15
1729/1729 [==============================] - 118s 68ms/step - loss: 0.0076 - acc: 0.9977 - val_loss: 0.1565 - val_acc: 0.9623
Epoch 8/15
1729/1729 [==============================] - 117s 68ms/step - loss: 0.0057 - acc: 0.9984 - val_loss: 0.1822 - val_acc: 0.9603
Epoch 9/15
1729/1729 [==============================] - 115s 66ms/step - loss: 0.0034 - acc: 0.9988 - val_loss: 0.1470 - val_acc: 0.9738
Epoch 10/15
1729/1729 [==============================] - 116s 67ms/step - loss: 0.0034 - acc: 0.9991 - val_loss: 0.1344 - val_acc: 0.9738
Epoch 11/15
1729/1729 [==============================] - 117s 68ms/step - loss: 0.0028 - acc: 0.9991 - val_loss: 0.1390 - val_acc: 0.9722
Epoch 12/15
1729/1729 [==============================] - 116s 67ms/step - loss: 0.0030 - acc: 0.9992 - val_loss: 0.1469 - val_acc: 0.9740
Epoch 13/15
1729/1729 [==============================] - 117s 68ms/step - loss: 0.0020 - acc: 0.9994 - val_loss: 0.1443 - val_acc: 0.9711
Epoch 14/15
1729/1729 [==============================] - 118s 68ms/step - loss: 0.0016 - acc: 0.9996 - val_loss: 0.2240 - val_acc: 0.9568
Epoch 15/15
1729/1729 [==============================] - 118s 68ms/step - loss: 0.0011 - acc: 0.9997 - val_loss: 0.1823 - val_acc: 0.9711
```

('Apple Braeburn', 0, '100.00%')

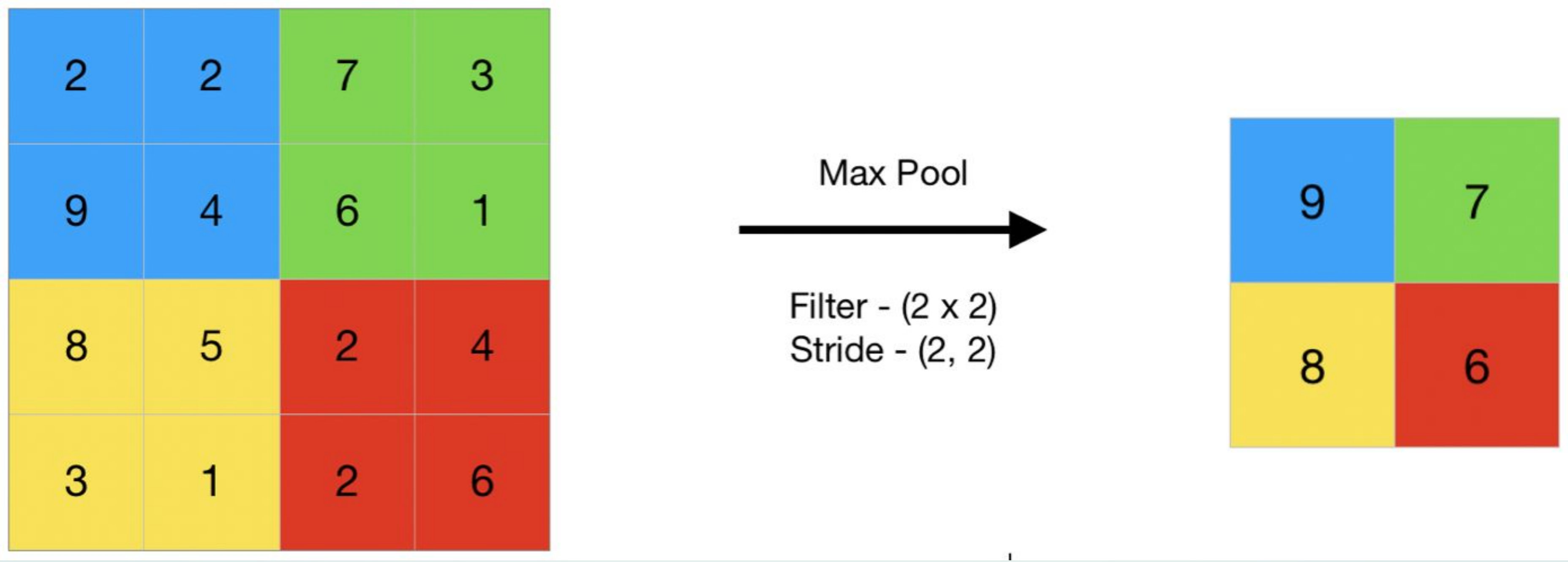
('Apple Braeburn', 0, '100.00%')

('Apple Braeburn', 0, '100.00%')

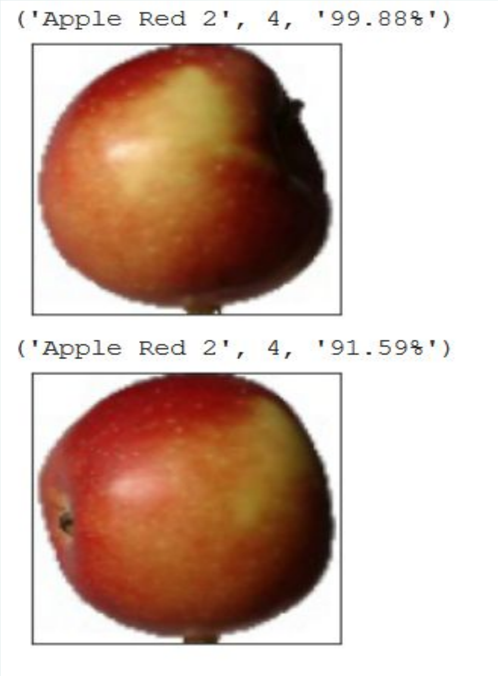
('Apple Red 2', 4, '99.88%')

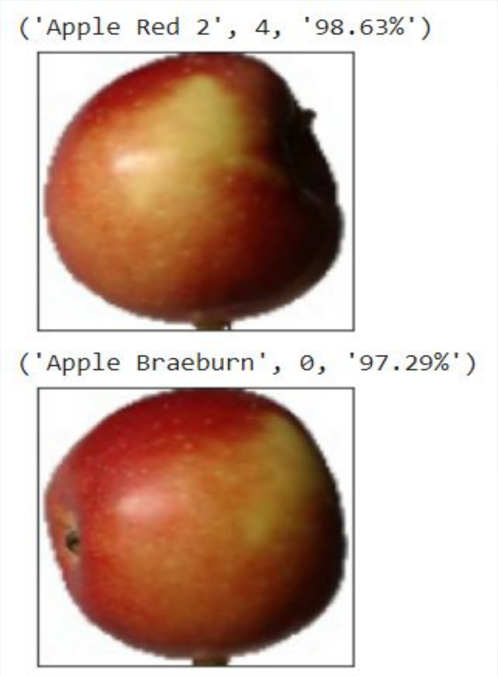
('Apple Red 2', 4, '91.59%')

Q&A


2 2 7 3
9 4 6 1
8 5 2 4
3 1 2 6
Max Pool
Filter - (2 x 2)
Stride - (2, 2)
9 7
8 6

ReLU vs Tanh...


('Apple Red 2', 4, '99.88%')
('Apple Red 2', 4, '91.59%')


('Apple Red 2', 4, '98.63%')
('Apple Braeburn', 0, '97.29%')

Tanh vs ReLU

('Grape White 4', 22, '100.00%')

('Grapefruit White', 23, '91.70%')

('Grape White 4', 22, '100.00%')

('Grapefruit White', 23, '76.29%')

- **Training data.** This type of data builds up the machine learning algorithm. The data scientist feeds the algorithm input data, which corresponds to an expected output. The model evaluates the data repeatedly to learn more about the data’s behavior and then adjusts itself to serve its intended purpose.
- **Validation data.** During training, validation data infuses new data into the model that it hasn’t evaluated before. Validation data provides the first test against unseen data, allowing data scientists to evaluate how well the model makes predictions based on the new data. Not all data scientists use validation data, but it can provide some helpful information to optimize hyperparameters, which influence how the model assesses data.
- **Test data.** After the model is built, testing data once again validates that it can make accurate predictions. If training and validation data include labels to monitor performance metrics of the model, the testing data should be unlabeled. Test data provides a final, real-world check of an unseen dataset to confirm that the ML algorithm was trained effectively

Procedure...

The training network has the following architecture:

- A convolutional layer with 5x5 kernel and 32 filters
- A 2x2 MaxPooling layer
- Two convolutional layers with 3x3 kernels and 64 filters each
- A MaxPooling layer
- Another 3x3 convolutional layer with 128 filters, followed by a MaxPooling layer
- A fully connected layer of 512 units
- A final softmax layer

All layers have ReLU activations. Train the network for 15 epochs.

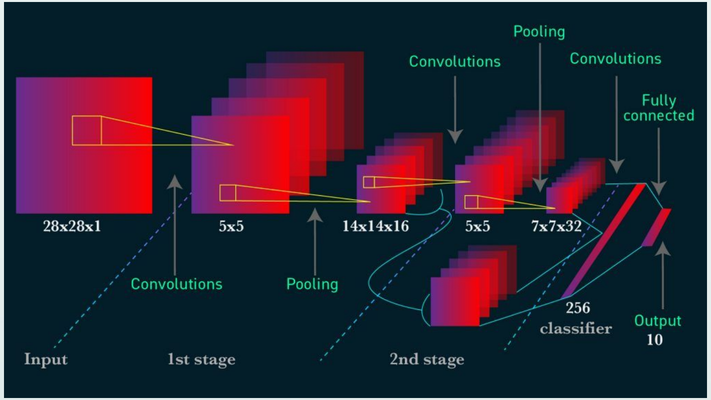
Extra-analysis ekata

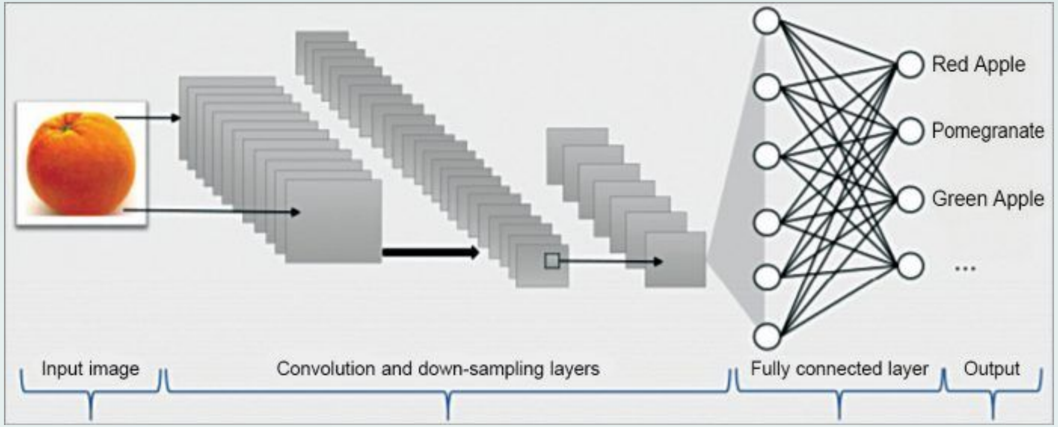
-Training the CNN by various combinations of hidden layers and increasing the number of epochs resulted in the highest test accuracy than the training accuracy.
-The difference between training and test recognition accuracies

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. ... A CNN uses a system much like a multilayer perceptron that has been designed for reduced processing requirements.

Convolutional Neural Networks (CNN or ConvNet) are complex feed forward neural networks.

CNNs are used for image classification and recognition because of its high accuracy





CNN places a convolutional layer and a pooling layer in the concealed layer between the info and yield layers .
The convolution layer applies a weighted channel to a piece of the information picture that might be useful for grouping, making an element map
The pooling layer decreases the element map by sub-testing the most significant piece of the component map received from the convolutional layer.

Algorithm

Training a CNN model
Uses _Keras model for fruit classification

- Model imports
- Data & model configuration
- Loading & preparing the data
- Specifying the model architecture
- Model compilation & starting the training process
- Full model code

Evaluation and results
Accuracy and FI score for model for model evaluation

Contribution

- Survey : Wasana , Kalani
- Data processing : Dulmini, Naduni
- Model design : Dulmini, Naduni,Wasana , Kalani
- Model Implementation : Wasana , Kalani, Dulmini, Naduni
- Hyperparameter tuning : Wasana, Naduni
- Model testing : Dulmini, Kalani
- Model Evaluation : Wasana, Kalani, Dulmini, Nanduni

Background including a review of existing work

Agriculture and fruit harvesting :

- *DeepFruits* : autonomous agricultural robotic platform for fruit yield estimation and automated harvesting
  - Fruit detection using imagery obtained from two modalities: colour (RGB) and Near-Infrared (NIR). (https://www.researchgate.net/publication/305824563_DeepFruits_A_Fruit_Detection_System_Using_Deep_Neural_Networks)

Cashiers in retail stores, supermarkets and factories