MINISTRY OF EDUCATION AND TRAINING
HCMc UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY FOR HIGH QUALITY TRAINING

------oOo-----

HCMUTE

REPORT

Topic: Classification of Fruits using Convolutional Neural Network

Intructor: Ph.D Nguyen Truong Thinh

Name: Hoang Doan Tien Phat
Student ID: 20146266
Subject: Artificial Intelligence
Group: AI Gr03CLC (sat, morning)

HCMc, May 2023

Table of contents

--- oOo ---

Github

Google Drive

I. Introduction

The rapid advancements in computer vision and deep learning techniques have revolutionized various industries, and one such area is the classification of objects in images. In the agricultural sector, the accurate and efficient identification of fruits is of paramount importance for numerous applications, including automated harvesting, quality control, and inventory management. The emergence of Convolutional Neural Networks (CNNs) has proven to be a powerful tool in achieving high accuracy and robust fruit classification.

The classification of fruits poses unique challenges due to the wide variations in shape, size, color, and texture among different species. Traditional image processing techniques often struggle to capture these intricate details, making it difficult to achieve accurate results consistently. However, CNNs have emerged as a breakthrough solution for fruit classification tasks, thanks to their ability to automatically learn complex hierarchical features directly from raw image data.

CNNs are a class of deep neural networks specifically designed to analyze visual data. They consist of multiple layers of interconnected neurons that extract and learn meaningful features from images. In the context of fruit classification, CNNs can effectively capture distinctive patterns such as the shape of leaves, the texture of the skin, and the color of fruits. By training a CNN model on a large dataset of labeled fruit images, it can learn to distinguish between different fruit classes with remarkable accuracy.

The classification process using CNNs typically involves two main stages: training and inference. During the training phase, the CNN model is presented with a labeled dataset, and it learns to adjust its internal parameters through a process known as backpropagation, minimizing the difference between predicted and actual labels. In the inference phase, the trained model is applied to new, unseen fruit images, and it predicts the most likely class for each input sample.

The benefits of using CNNs for fruit classification are extensive. They can handle a wide range of fruit varieties, even those with subtle visual differences, leading to highly accurate classification results. Additionally, CNN-based approaches offer scalability, enabling the system to handle large volumes of fruit images efficiently. Furthermore, the advancements in hardware acceleration technologies have facilitated real-time fruit classification on embedded devices, making it suitable for deployment in agricultural settings.

II. Methodology

The methodology for fruit classification using Convolutional Neural Networks (CNNs) involves several key steps, including data collection, preprocessing, model architecture selection, training, and evaluation. Each stage plays a crucial role in achieving accurate and robust fruit classification results.

Data Collection: A diverse and representative dataset of fruit images is essential for training a CNN model effectively. Fruit images can be obtained from various sources, including online repositories, agricultural databases, or by capturing images using cameras or sensors. It is crucial to ensure that the dataset covers a wide range of fruit classes, variations in lighting conditions, angles, and backgrounds to enhance the model's generalization capabilities. The dataset used, self-collected form google and istockphoto.com, contains a set of 1440 images for 6 different class of fruit we use 80% of them for training and 20% for validation and testing belonging to 6 species from different fruits:

Apple:	240 samples in total
Banana:	240 samples in total
Grapes:	240 samples in total
Golden pear:	240 samples in total
Tomato:	240 samples in total
Orange:	240 samples in total

Preprocessing: Before feeding the images into the CNN model, preprocessing steps are performed to standardize the data and enhance the model's performance. Common preprocessing techniques include resizing the images to a fixed size, normalizing pixel values, and augmenting the dataset by applying transformations such as rotation, scaling, and flipping. These steps help increase the diversity and robustness of the training data.

Model Architecture Selection: The choice of CNN architecture greatly influences the model's performance. The architecture selection depends on factors such as the complexity of the fruit classification task, available computational resources, and the size of the dataset.

Training: The training process involves feeding the preprocessed fruit images into the CNN model and iteratively adjusting the model's parameters to minimize the difference between predicted and actual labels. The optimization is achieved through backpropagation, where the gradients of the loss function with respect to the model parameters are computed and used to update the weights. The training is typically performed on high-performance GPUs to accelerate the computation.

In summary, the methodology for fruit classification using CNNs encompasses data collection, preprocessing, model architecture selection, training, evaluation. This iterative process enables the development of highly accurate and robust models capable of classifying fruits with a high degree of precision.

III. Model and Algorithm

1. Create train/test sets

1440 samples of 6 types of fruit are divided into train and test set that contain 80 and 20 percent of the total number of samples (1152 samples in train set and 288 samples in test set)

```
from os import listdir
from numpy import asarray,save
from keras.utils import load_img, img_to_array, to_categorical

folder = '/content/drive/MyDrive/AI_final_project/AI_images/'
photos, labels = list(), list()

for file in listdir(folder):
  output=0
  if file.startswith('apple'):
    output=1
  if file.startswith('banana'):
    output=2
  if file.startswith('grapes'):
    output=3
  if file.startswith('orange'):
    output=4
  if file.startswith('pear'):
    output=5
  if file.startswith('tomato'):
    output=6
  photo=load_img(folder+file,target_size=(60,60))
  photo=img_to_array(photo)
  photos.append(photo)
  labels.append(output)

photos = asarray(photos)
labels = asarray(labels)
labels = to_categorical(labels)
print('X size:', photos.shape,'\ny size:' , labels.shape)
save('imgs.npy',photos)
save('labels.npy',labels)
```

```
X size: (1440, 60, 60, 3)
y size: (1440, 7)
```

```
from sklearn.model_selection import train_test_split
import numpy as np

x = np.load('imgs.npy')
y = np.load('labels.npy')
x = x.reshape(1440, 60, 60, 3)
x = x.astype('float32')/255

x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.2,
                                                    train_size=0.8,
                                                    random_state=1)

print('x train:', x_train.shape, '\ny train:', y_train.shape)
print('x test:', x_test.shape, '\ny test:', y_test.shape)
```

```
x train: (1152, 60, 60, 3)
y train: (1152, 7)
x test: (288, 60, 60, 3)
y test: (288, 7)
```

2. Model architecting and training

The convolutional neural network model above has 15 layers:

Layer 01 - Conv2D
Layer 02 - LeakyReLU
Layer 03 - MaxPooling2D
Layer 04 - Dropout
Layer 05 - Conv2D
Layer 06 - LeakyReLU
Layer 07 - MaxPooling2D
Layer 08 - Dropout
Layer 09 - Conv2D
Layer 10 - LeakyReLU
Layer 11 - MaxPooling2D
Layer 12 - Dropout
Layer 13 - Conv2D
Layer 14 - LeakyReLU
Layer 15 - MaxPooling2D

```
from keras.models import Sequential
from keras.layers import Dense,Flatten,Dropout,Conv2D
from keras.layers import MaxPooling2D,Normalization,LeakyReLU

model=Sequential()
model.add(Conv2D(32,kernel_size=(3,3),activation='relu',
                 input_shape=(60,60,3),padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D((2,2),padding='same'))
model.add(Dropout(0.25))

model.add(Conv2D(64,kernel_size=(3,3),activation='relu',padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D((2,2),padding='same'))
model.add(Dropout(0.25))

model.add(Conv2D(128,kernel_size=(3,3),activation='relu',padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D((2,2),padding='same'))
model.add(Dropout(0.25))

model.add(Conv2D(256,kernel_size=(3,3),activation='relu',padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D((2,2),padding='same'))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256,activation='relu'))
model.add(Dense(7,activation='softmax'))

from keras.losses import categorical_crossentropy
from keras.optimizers import Adam
model.compile(loss=categorical_crossentropy,
              optimizer=Adam(),metrics=['accuracy'])
train = model.fit(x_train,y_train,batch_size=64,epochs=100,verbose=1)

model.summary()
```

The details of the layers of the CNN model are listed in order:

Conv2D layer (32 filters, kernel_size=(3,3), activation='relu', padding='same'): This layer performs convolutional operations on the input image. It has 32 filters with a size of (3,3) applied to the image. The input size of the image is (60, 60, 3), meaning the image has a size of 60x60 pixels and 3 color channels (RGB). The output of this layer is the feature maps.

LeakyReLU layer (alpha=0.1): The LeakyReLU activation layer is applied to the output of the Conv2D layer to introduce non-linearity to the model. Alpha has a value of 0.1, which means if the input value is negative, it will be multiplied by 0.1 before applying the activation function.

MaxPooling2D layer (pool_size=(2,2), padding='same'): This layer reduces the spatial dimensions of the feature maps by selecting the maximum value in each 2x2 region. The input and output size of the feature map remain the same due to the use of 'same' padding.

Dropout layer (rate=0.25): The Dropout layer randomly drops a portion of connections (25% in this case) between units in the previous layer during training. This helps prevent overfitting.

Steps 2-4 are repeated three times with an increasing number of filters in the Conv2D layers (64, 128, 256). The kernel_size, activation, padding, and pooling operations remain the same.

Flatten layer: This layer converts the 2D feature maps into a 1D vector, preparing it for the fully connected layers.

Two Dense layers: Two Dense layers are added after the Flatten layer. The first layer has 256 units with 'relu' activation. The second layer has 7 units (corresponding to the number of classes) with 'softmax' activation. The softmax layer produces probabilities for each output class.

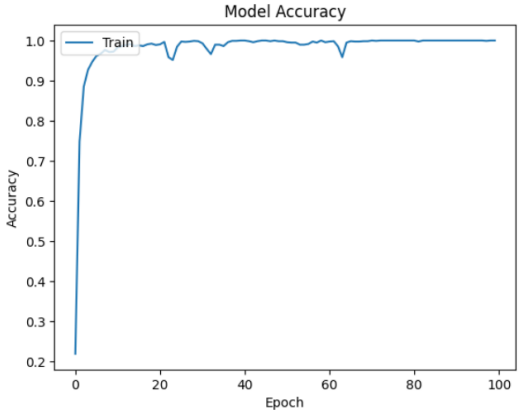
After that, the model is compiled with the categorical cross-entropy loss function, Adam optimizer, and accuracy as the metric.

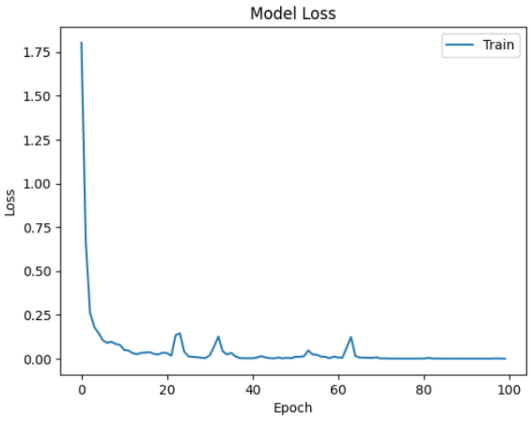
3. Calculating accuracy

The trained Convolutional Neural Network model reach accuracy of 98.6% with 288 samples in test set

```
1 loss, acc = model.evaluate(x_test, y_test)
2 print('accuracy:', acc, '\nloss:', loss)
```

```
9/9 [==============================] - 0s 5ms/step - loss: 0.0838 - accuracy: 0.9861
accuracy: 0.9861111044883728
loss: 0.08378544449806213
```





4. Example prediction

The trained model correctly predicted all four images (apple_on_tree.jpg, chinese_golden_pear.jpg, grapes_on_tree.jpg, and orange.jpg), which do not belong to either the train set or the test set.

apple_on_tree.jpg

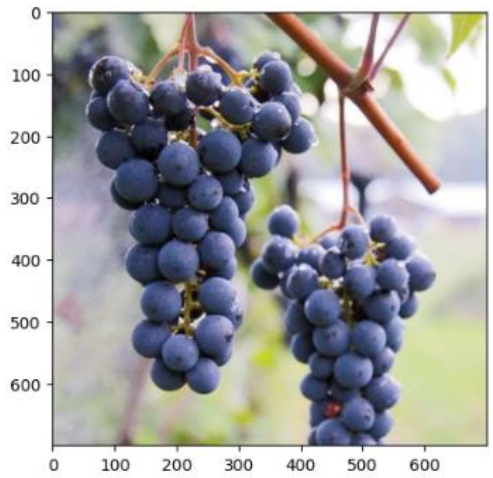
chinese_golden_pear.jpg

grapes_on_tree.jpg

orange.jpg

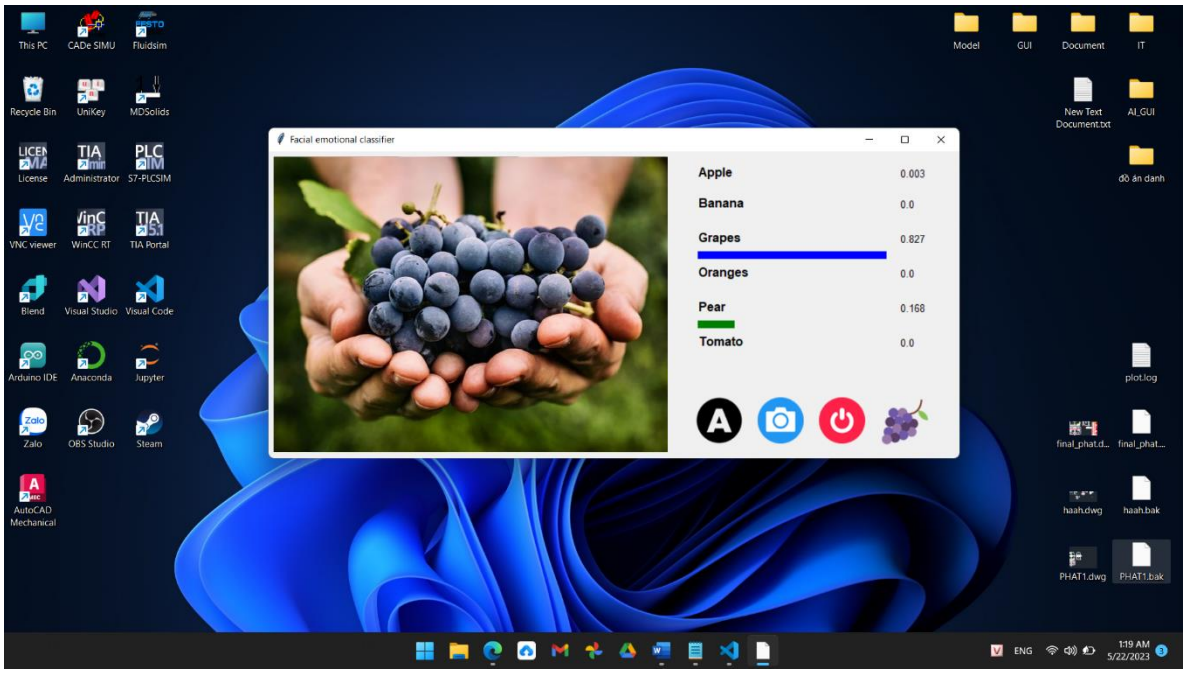
```python
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
from keras.models import load_model
my_dict = {1:'Apple', 2:'Banana', 3:'Grapes',
           4:'Orange', 5:'Pear', 6:'Tomato'}

uploaded_imgs = ['apple_on_tree.jpg',
                 'chinese_golden_pear.jpg',
                 'orange.jpg',
                 'grapes_on_tree.jpg']

for image in uploaded_imgs:
  # Load the image
  img = Image.open(image).convert('RGB')
  print('\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\nImage loaded: ', image)
  plt.imshow(img)
  # convert it to a numpy array
  img = img.resize((60,60))
  img_arr = np.array(img)
  img_arr = img_arr.reshape((1,) + img_arr.shape)
  img_arr = img_arr.astype('float32')/255
  # Perform prediction on the image
  prediction = model.predict(img_arr)
  print(prediction[0])
  # Get the label with the highest probability
  label = np.argmax(prediction)
  print('==> Best predicted label:', my_dict[label])
```

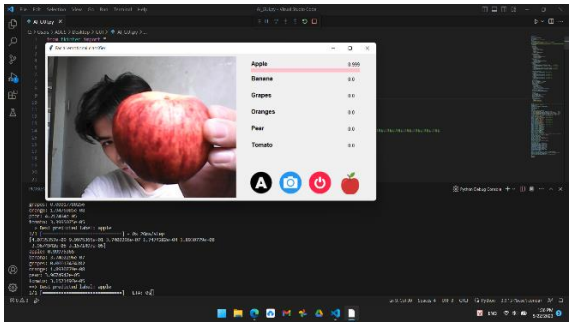
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Image loaded:  apple_on_tree.jpg
1/1 [==============================] - 0s 19ms/step
[1.2616734e-08 9.9786240e-01 4.6602174e-05 1.3602879e-03 1.0430360e-05
 7.1887072e-04 1.4609356e-06]
==> Best predicted label: Apple

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Image loaded:  chinese_golden_pear.jpg
1/1 [==============================] - 0s 19ms/step
[2.3234624e-12 8.7020025e-10 1.2313068e-07 9.4778774e-10 5.4708647e-07
 9.9999928e-01 3.7043097e-08]
==> Best predicted label: Pear

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Image loaded:  orange.jpg
1/1 [==============================] - 0s 20ms/step
[2.3119711e-12 5.2767169e-05 5.7018929e-06 2.4346580e-09 9.9820340e-01
 1.0668704e-03 6.7120971e-04]
==> Best predicted label: Orange

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Image loaded:  grapes_on_tree.jpg
1/1 [==============================] - 0s 21ms/step
[1.3607816e-16 6.6780709e-10 9.8177705e-13 1.0000000e+00 6.0462482e-16
 1.5100006e-10 4.4234120e-16]
==> Best predicted label: Grapes
```

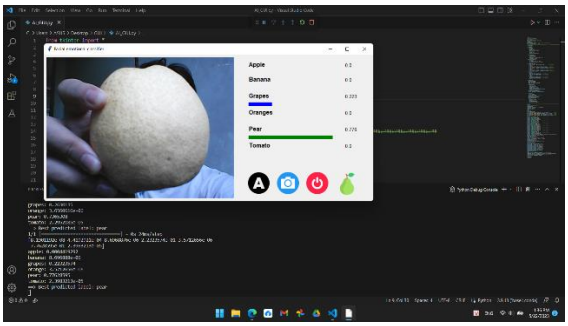


IV. Result and Discussions

The CNN model with an accuracy of over 90% used for classifying fruits showcases its impressive ability to accurately identify various types of fruits. This high level of accuracy underscores its potential for practical applications, such as automated sorting processes or aiding visually impaired individuals. With its reliable fruit classification capabilities, the CNN model holds promise for enhancing efficiency and accessibility in various domains.







V. Conclusion

In conclusion, the application of CNNs in fruit classification has opened up new possibilities for automating and improving various agricultural processes. With their ability to analyze complex visual information and provide accurate predictions, CNNs have become a vital tool for enhancing efficiency and precision in fruit-related industries. In the following sections, we will delve into the methodologies, datasets, and performance evaluation metrics employed in fruit classification using CNNs, shedding light on the exciting advancements and challenges in this field.