Mini Project

Aman Tej Vidapu

I. Introduction

The aim of this Mini project is to use and understand a specific machine learning algorithm on fruits and vegetables image data set. which is taken from VICOS[1]. The algorithm that is used to classify our data in this document is CNN[2]. For this model we have to take three datasets: training, validation data to train the model and testing data to evaluate the model. In this document we will first generate labels using data generators for our data as we directly feed model with direct images. Then we train and test the model accordingly.

II. Setting up Environment

A. Programming Environment

Programming language used in this project is Python, because it is easy to understand and has many great advantages over other programming languages. Even new developers will find Python code to be compact and legible, which is advantageous for machine and deep learning projects[3]. Anaconda environment is selected to implement tasks of this assignment, which has Spyder IDE for code development. Anaconda is a free open-source environment well build environment to develop projects. We can get access to over 7,000+ open-source packages, we can be install them with “conda install ’package-name’” command [4].

B. System Specifications

• The operating system used for this assignment is Windows 11.
• System type is Windows- 64-bit operating system, x64 based processor.
• 8 GB of installed RAM

C. Installations

Anaconda environment was downloaded from the official Anaconda website[5]. Spyder IDE (Fig.2.) is integrated with the Anaconda environment which can be accessed from Anaconda by running “spyder” command,

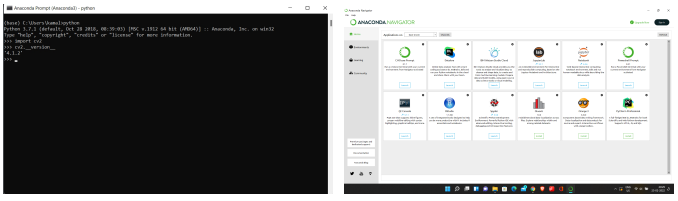

Fig. 1. Left: Anaconda Navigator. Right: Spyder IDE

III. Downloading Images and setting up directory

In this assignment we use 3 different fruits data images namely Orange, Banana and Strawberry. Images that are used in this assignment are downloaded from VICOS[1]. Now we will split the data into three datasets accordingly and will name them accordingly.

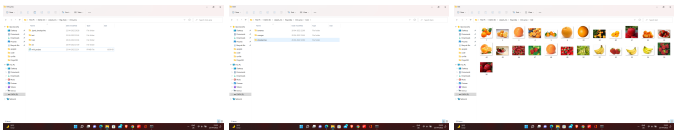

Fig. 2. Left: Main directory. Middle: Train/val data. Right: Test data

IV. Generating Datasets and Labels

First, we have to separate out images to three sub-folders as training, validation, testing data. We use ImageDataGenerator[6] function to generate data along with particular label. As the input images have different sizes, we will resize every data image to a particular default size.

```
train = ImageDataGenerator(rescale =
    1./255, shear_range = 0.3,
    horizontal_flip = True, zoom_range =
    0.3)
train_dataset = train.flow_from_directory
    (r"D:\classes_ms\Big_Data\mini_proj\
    train", target_size = (200, 200),
    batch_size = 3,color_mode = 'rgb',
    class_mode = 'categorical') #similar
    to validation and test datasets
```

ImageDataGenerator will assign label by folder name where we save data images using flow_from_directory function.

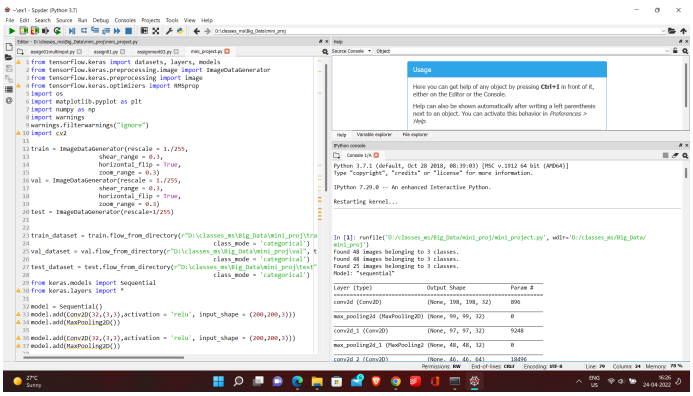

Fig. 3. Code for generating labels.

V. Implementing a Deep Learning model

A. Implementing Convolution Neural Network

Implementing Convolution Neural Network model to predict the class of a given input. First Implemented a model and then trained with taring dataset of a specific categorical data images and added extra validation dataset to improve training accuracy. As we gave categorical input, every image will save its label as: Bananas class as [1,0,0], Oranges class as [0,1,0] and Strawberries class as [0,0,1]. Next, we take a sequential keras model and added layers to it. We use convolution layers, which are very good at finding hidden patterns in image based data. In convolution layers, we can see feature extraction, max pooling, activation (relu) and flattening feature vectors. For this model we use two convolution models with relu activation function and followed by max pooling with (2,2) matrix dimension. Then we use dense layers as ANN to classify the classes. I tested the data with multiple well-known Optimizers as Adam, SGD and rmsprop. Rmsprop has better results, so proceeded with that optimizer in this document. Also, at end we use softmax activation function for getting class probabilities.

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation = '
    relu', input_shape = (200,200,3)))
model.add(MaxPooling2D())
model.add(Conv2D(32,(3,3),activation = '
    relu', input_shape = (200,200,3)))
model.add(MaxPooling2D())
model.add(Conv2D(64,(3,3),activation = '
    relu', input_shape = (200,200,3)))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(1024,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3,activation = 'softmax')
    )
model.compile(loss = '
    categorical_crossentropy',optimizer =
    'rmsprop',metrics = ['accuracy'])
```

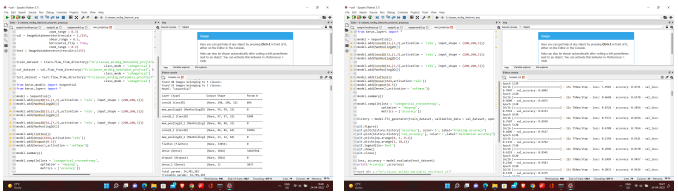

Fig. 4. CNN model summary and training.

B. Testing model

Now as we trained the model with specific data, we have to take same category dataset test data to predict labels. Now we got deep learning model so, we have to pass the test generator data to model to predict and evaluate performance. Also, In order to visualise the prediction for specific image, we used matplotlib. Refer Fig.5. for the code and can see the outputs in the same image with predictions as plot title with image.

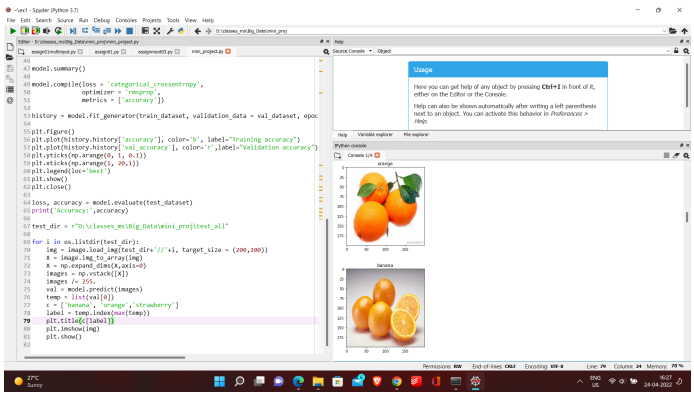

Fig. 5. Code for testing.

C. Performance of a machine learning model

From model evaluate function we can see the accuracy, which is pre-defined function from keras library. We generate test data from directory same as training and validation data which is labelled data. So the generator will generate data and test the model. We given over 20 epochs, simply iterations to run the input data to train model. And, batch size is left as option to choose by model. We know, generator labels as:
Bananas class as [1,0,0]
Oranges class as [0,1,0] and
Strawberries class as [0,0,1].
The training accuracy and validation accuracy for each epoch

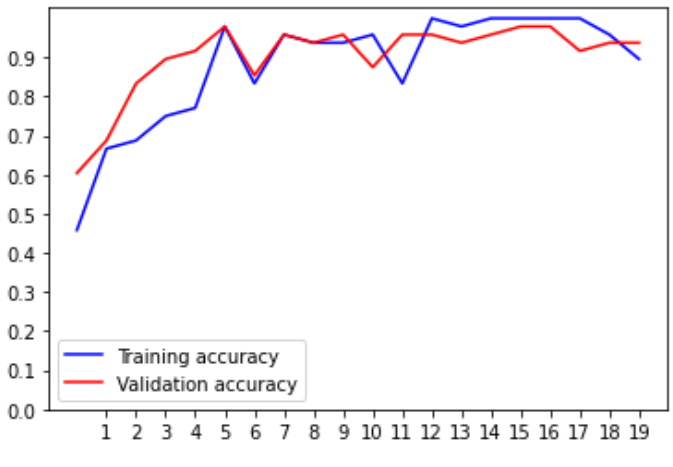

Fig. 6. Performance measure for both categorical datasets (2-class).

plots were shown in Fig.6. Finally, testing accuracy is around 80 percent.

D. misclassification

Now we have seen the predictions of model to our test data. Actual images class and predicted of a test data images miss

match for most of the orange class data, which predicts as bananas. This might be the reason of less input data and total iterations to train, we can over-come this issue by adding more epochs, filter layers to our model. But, the misclassification here is understandably low as we trained with less data. You can see the accurate predictions and misclassfied predictions in Fig. 7 and Fig.8.

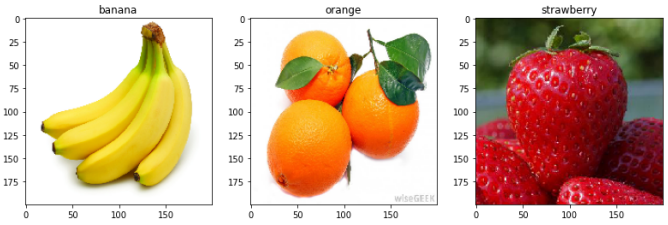

Fig. 7. Correct predictions of class: Banana, Orange and Strawberry

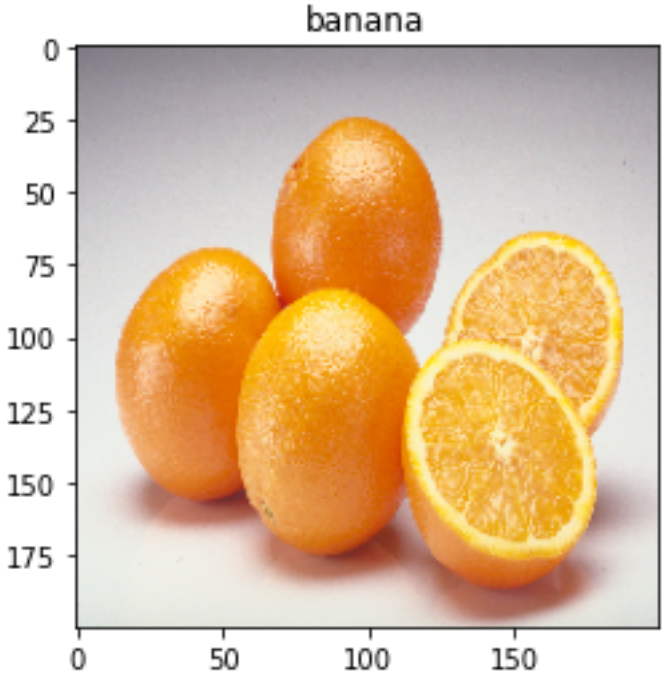

Fig. 8. Wrong prediction.

VI. CONCLUSION

We have seen implementation of a classification model based on deep learning technique to predict whether a given image is of class banana, orange or strawberry. Preprocessing of the input data is import for the convolution model as it has to predict hidden patterns in the data, every image should be of same size. Also, the number of input images, layers of the model and number of epochs to train the model will have impact on overall accuracy of the learning model. The more layers we add the more sophisticated the model become. Classification on new test data has pretty good predictions by this model mentioned in this document, we can improve the accuracy more by using data relative optimizer and activation functions.

REFERENCES

[1] https://www.vicos.si/Downloads/FIDS30
[2] https://keras.io/guides/sequential_model/
[3] https://www.section.io/engineering-education/why-python-is-good-for-machine-learning/
[4] https://docs.anaconda.com/anaconda/
[5] https://www.anaconda.com/products/individual
[6] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image
[7] https://datauab.github.io/catsvsdogs/
[8] Machine Learning Models and Algorithms for Big Data Classification – Shan Suthahara