**CNN Multi-Class Image Classification for Sports Balls**

Convolutional Neural Networks (CNNs) are a type of neural network that is commonly used in image classification tasks. CNNs are designed to take in input data in the form of images and process the data through multiple layers, each of which applies a different set of filters to the data to extract different features. In this project, we aim to develop a CNN model to classify images of sports balls into their respective classes using a dataset containing various images of sports balls.

The dataset includes images of sports balls from 15 different sports, including basketball, soccer, tennis, and more. We will be training the model on this data and evaluating its performance in classifying sports balls accurately. The development of accurate sports ball classification models using CNNs has the potential to transform how we recognize and categorize images of sports balls, providing more accurate and personalized results than traditional methods. This project can have significant implications in various domains such as sports, gaming, and fitness.

**Technical Architecture:**

**Project Flow:**

• User interacts with the UI to upload an image.

• Uploaded image is analysed by the model which is integrated.

• Once the model analyses the image, the prediction is showcased on the UI To accomplish this, we have to complete all the activities listed below,

• Define Problem / Problem Understanding

o Specify the business problem

o Business requirements

o Literature Survey

o Social or Business Impact

• Data Collection & Preparation.

o Collect the dataset

o Data Exploration

o Data Preparation

• Prepare the layers of the neural network.

o Convoluted Layers

o MaxPooling

o Flatten

o Deep layers

• Performance Testing & Hyperparameter Tuning

o Testing model with multiple evaluation metrics

o Comparing model accuracy before & after applying hyperparameter tuning

• Model Deployment

o Save the best model

o Integrate with Web Framework

• Project Demonstration & Documentation

o Project Documentation-Step by step project development procedure

**Prior Knowledge:**

You must have prior knowledge of following topics to complete this project.

• ML Concepts

o Convolutional Neural Networks

• Flask Basics

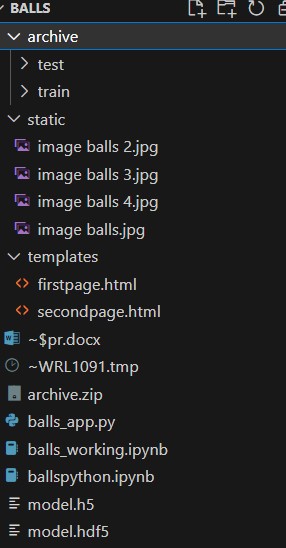
**Project Structure:**

• Create a Balls folder which contains files as shown below

• We are building a flask application which needs HTML pages stored in the templates folder and a python script main.py for scripting.

• model.hdf5 is where our machine learning model is saved. Further, we will use this model for flask integration.

• Static folder contains the images which will be used to style our web application.



**Milestone 1: Define Problem / Problem Understanding**

**Activity 1: Specify the business problem**

Refer Project Description

**Activity 2: Business requirements**

To ensure that the sports ball classification model meets business requirements and can be deployed for public use, it should follow the following rules and requirements:

• Accuracy: The model should have a high level of accuracy in classifying sports balls into their respective classes, with a low margin of error. This is crucial to ensure that the predictions are reliable and trustworthy.

• Privacy and security: The model should be developed in accordance with privacy and security regulations to protect user data. This includes ensuring that sensitive data is stored securely and implementing proper data access controls.

• Interpretability: The model should be interpretable, meaning that the predictions can be explained and understood by the end-users. This is important to build trust in the model and to allow users to make informed decisions based on the predictions.

• User interface: The model should have a user-friendly interface that is easy to use and understand. This is important to ensure that the model can be deployed for public use, even by individuals who may not have technical expertise.

**Activity 3: Literature Survey**

A literature survey for a CNN multi-class image classification for sports balls project would involve researching and reviewing existing studies, articles, and other publications related to image classification and CNNs. The survey would aim to gather information on current image classification techniques, their accuracy, and limitations. Additionally, the literature survey would analyse existing CNN algorithms and methods used for image classification, including different architectures, hyperparameter tuning, and transfer learning. The survey would also examine any gaps in knowledge and research opportunities in the field of image classification, including new CNN techniques, improved feature selection methods, and data quality issues.

**Activity 4: Social or Business Impact**

Social Impact: From a social perspective, the model can aid in the recognition and categorization of images of sports balls, which can have applications in various domains such as sports, gaming, and fitness.

Business Impact: The model can have significant implications in various industries, including sports equipment manufacturers, gaming companies, and fitness apps. For instance, in gaming, the model can improve the accuracy of object recognition in sports games, leading to more realistic gameplay.

**Milestone 2: Data Collection & Preparation**

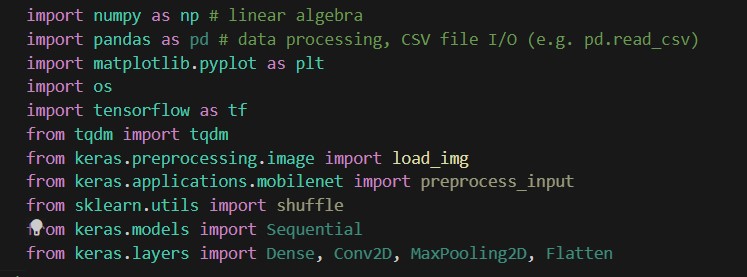
ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

**Activity 1:** Collect the dataset There are many popular open sources for collecting data. E.g., kaggle.com, UCI repository, etc. In this project we have used an image dataset. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: - https://www.kaggle.com/datasets/samuelcortinhas/sports-balls-multiclass-image-classification

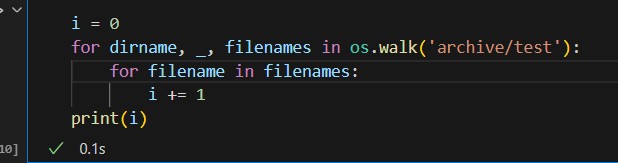
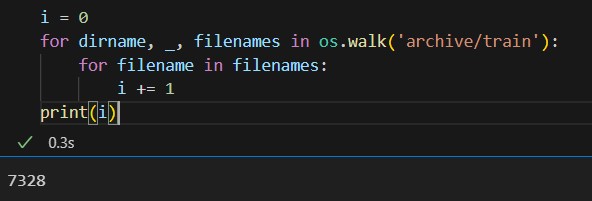
Download the dataset from the above link.

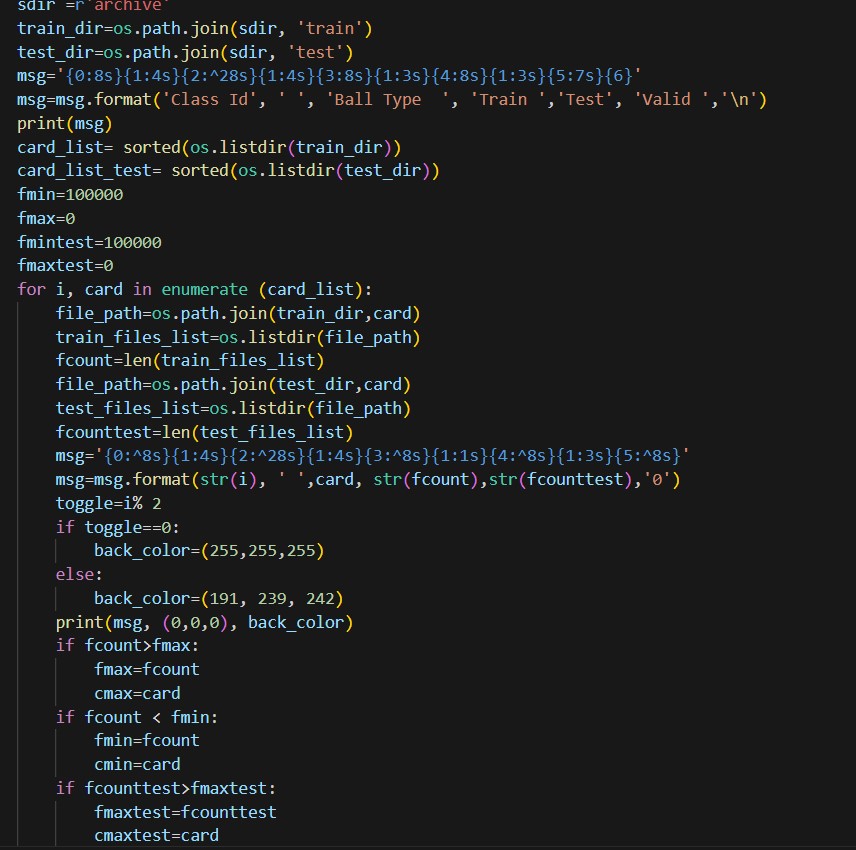
**Activity 1.1:** Importing the libraries as shown in the image below.

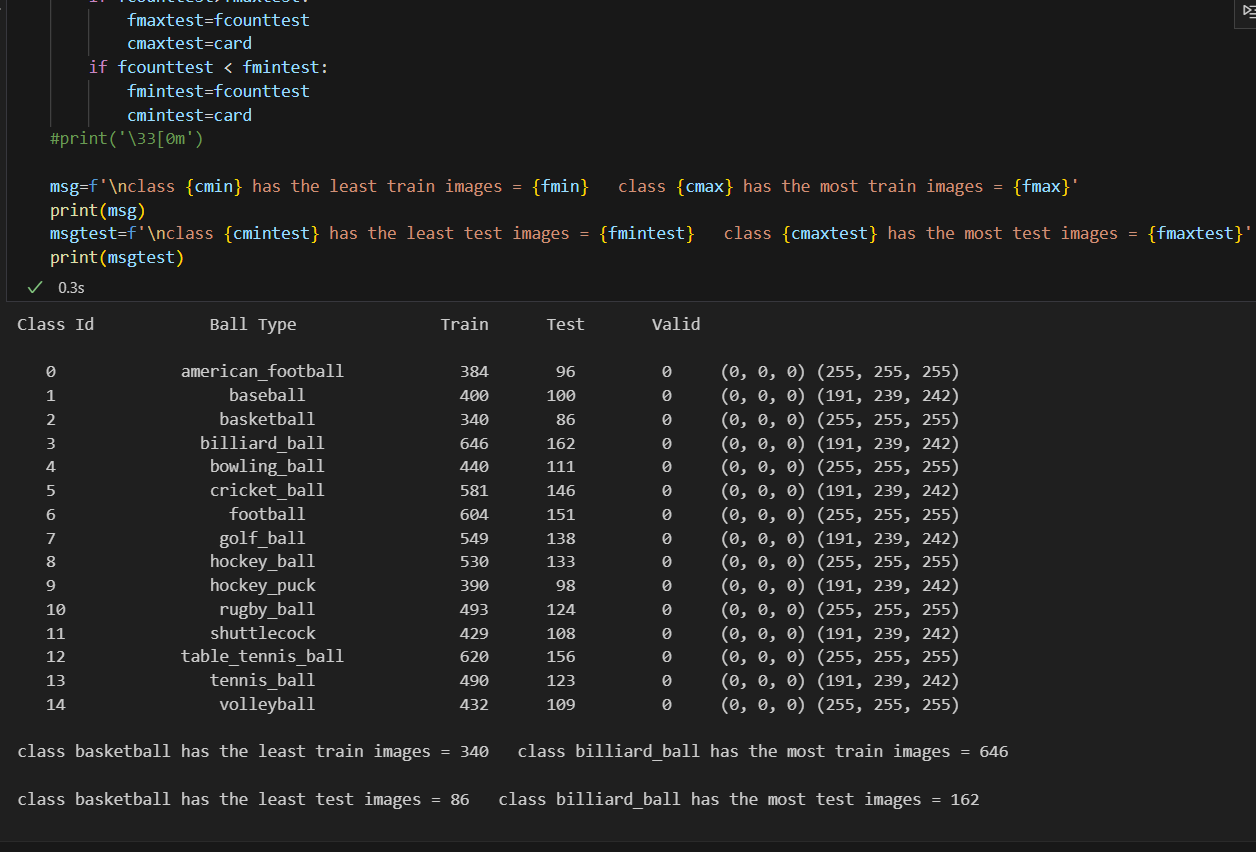


**Activity 1.2: Data Exploration**

The code here shows the number of images in the test and train dataset.

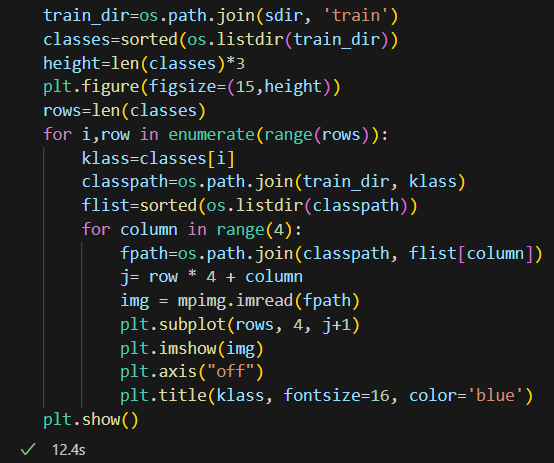


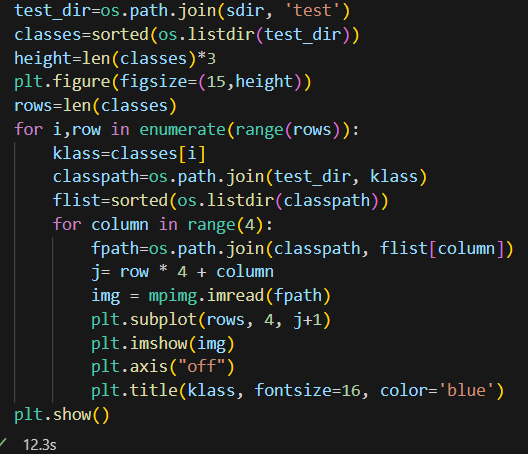
And also, the class with the most images and the least number of images in both test and train data.

****

**Activity 1.3: Read the Dataset**

Our dataset format could be in .csv, excel files,.txt, .json, and so on. With the help of pandas or other libraries we can read the dataset. Since our dataset is an image dataset, we use an image processing library such as OpenCV or PIL to read the images. We can use functions such as cv2.imread() or Image.open() to read individual images from our dataset. Here is the code and the output that shows the images for both test and train models.





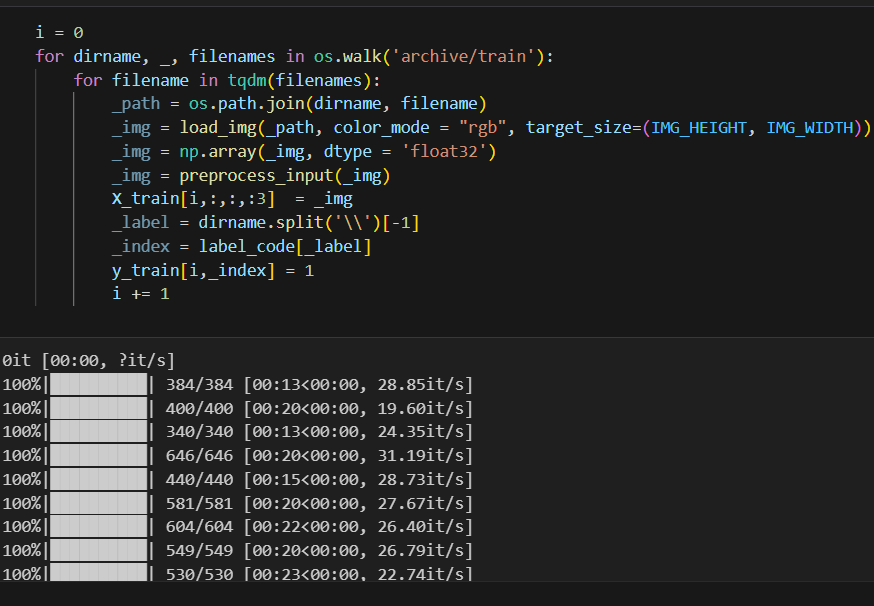


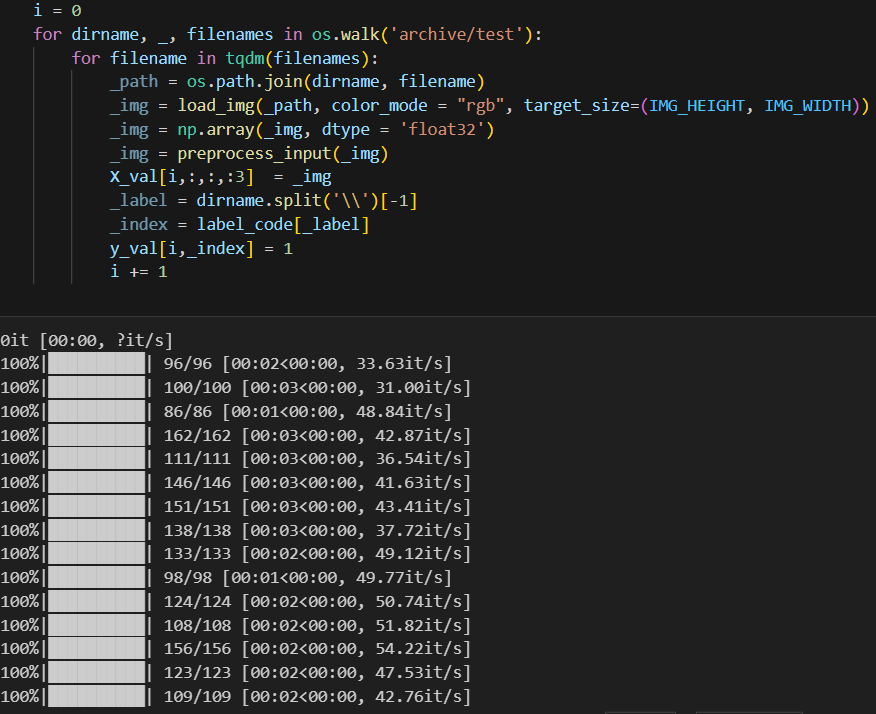
**Milestone 3: Data Preprocessing**

Before we can train our CNN model, we need to preprocess the image data. This involves several steps such as resizing the images and dividing them into training and testing sets.

**Activity 1:** Resizing the images.

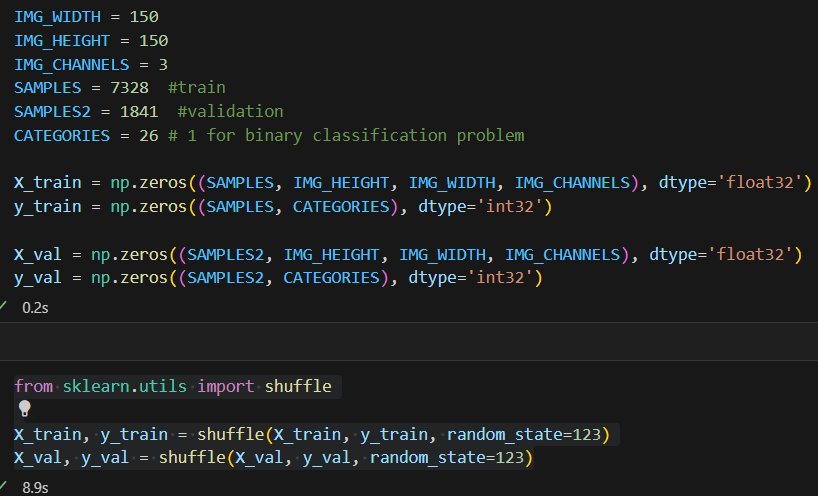
The first step in data preprocessing is to resize all the images to a common size. This is necessary because the input layer of our CNN model expects images of a fixed size. We can use image processing libraries such as OpenCV or PIL to resize the images.





**Activity 2:** Dividing the data into training and testing sets.

After resizing the images, we need to divide them into training and testing sets. The training set is used to train our CNN model, while the testing set is used to evaluate its performance. We can use a library such as scikit-learn to split the data into training and testing sets.



**Milestone 4: Prepare the layers of the neural network.**

In this project, we use the **Keras** library with a **TensorFlow** backend to build and train our CNN model.

**Activity 1: Prepare Convoluted Layers**

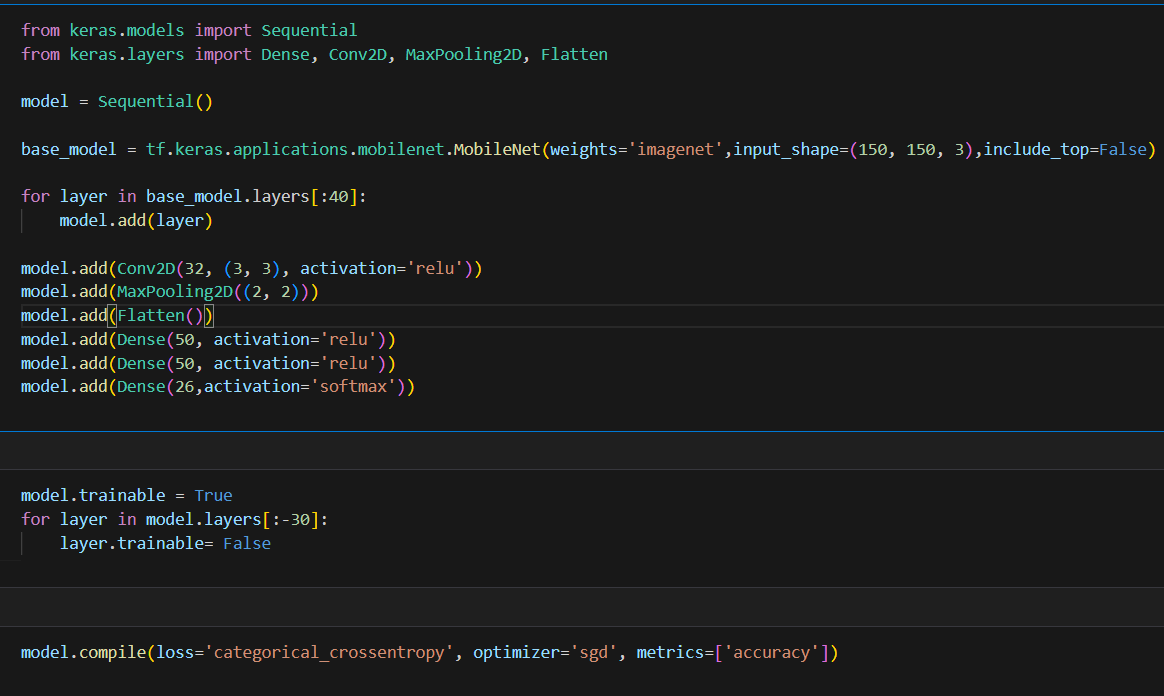
The first step in preparing the layers of the neural network is to add convolutional layers using the Conv2D class from the keras.layers module. These layers are responsible for extracting features from the input images by applying a set of filters to them. Each filter is designed to detect a specific type of feature, such as edges or corners, and produces a feature map that highlights the locations in the image where that feature is present.

**Activity 2: Prepare MaxPooling Layers**

After the convolutional layers, we add max pooling layers using the MaxPooling2D class from the keras.layers module. These layers are used to reduce the spatial dimensions of the feature maps produced by the convolutional layers. This is done by dividing the feature maps into non-overlapping regions and taking the maximum value within each region. This helps to reduce the number of parameters in the model and makes it more computationally efficient. o Flatten: After the max pooling layers, we add a flatten layer using the Flatten class from the keras.layers module. This layer is used to convert the multi-dimensional feature maps into a one-dimensional vector that can be fed into a fully connected neural network.

**Activity 3: Prepare Deep Layers**

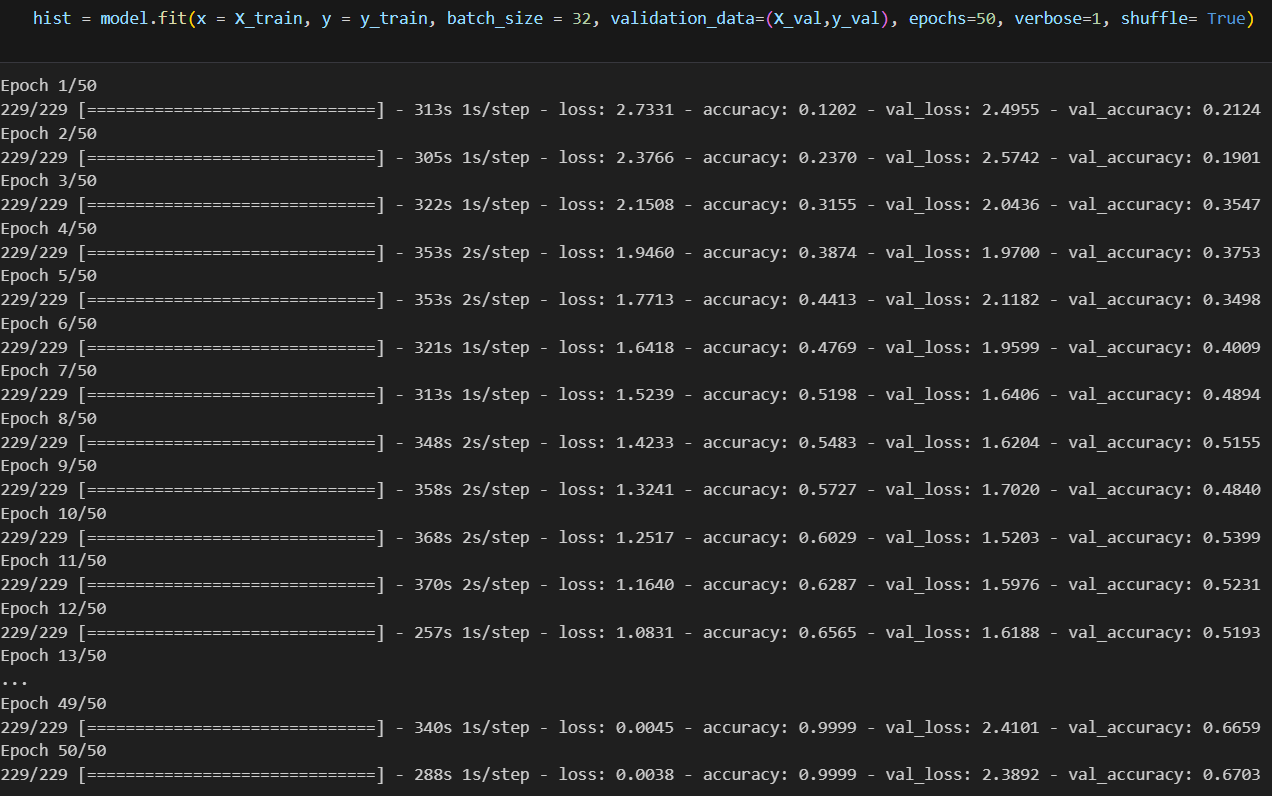
The final step in preparing the layers of the neural network is to add fully connected deep layers using the Dense class from the keras.layers module. These layers are responsible for combining the features extracted by the convolutional and max pooling layers to make predictions about which class each input image belongs to. The number and size of these layers can be adjusted based on the complexity of the classification problem.



**Milestone 5: Performance Testing & Hyperparameter Tuning**

**Activity 1: Testing model with multiple evaluation metrics.**

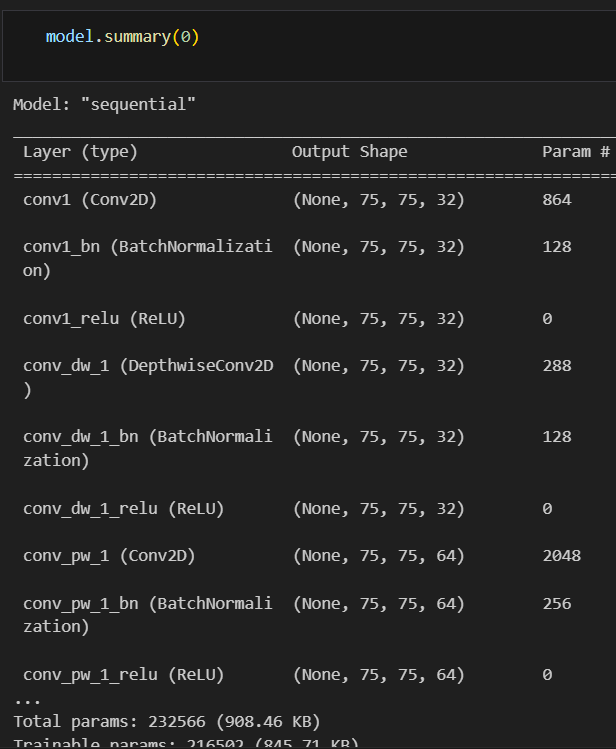
To evaluate the performance of our CNN model, we use multiple evaluation metrics such as accuracy, precision, recall, and F1-score. These metrics help us to assess how well the model is able to correctly classify the different types of sports balls and identify any areas for improvement.



**Activity 2: Comparing model accuracy before & after applying hyperparameter tuning.**

Hyperparameter tuning is the process of adjusting the values of the hyperparameters used in our CNN model to improve its performance. We compare the accuracy of our model before and after applying hyperparameter tuning to see if there is any improvement in its performance. This can be done by using techniques such as grid search or random search to find the optimal values for the hyperparameters.

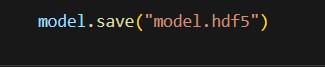




**Milestone 6: Model Deployment**

**Activity 1: Save the best model**

This code uses the “tensorflow” library in Python to save the trained CNN model named “model” as a file named “model.hdf5”. The “load” method from the tensorflow library is used to save the model object in a serialized form that can be used again later.



**Activity 2: Integrate with Web Framework**

In this section, we will be building a web application that would help us integrate the machine learning model we have built and trained. A user interface is provided for the users to upload an image of a sports ball for classification. The uploaded image is fed into the saved model, and the prediction is displayed on the UI.

The section has following tasks:

• Building HTML pages

• Building server-side script

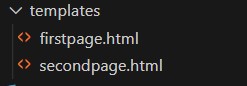
• Run the web application

**Activity 2.1: Building Html Pages:**

For this project, we create two HTML files:

• firstpage.html

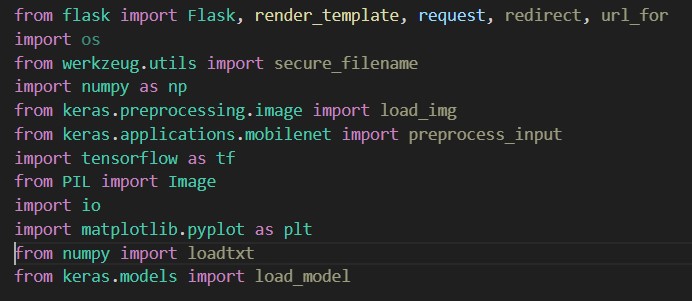
• secondpage.html



and save these HTML files in the templates folder.

**Activity 2.2: Build Python code:**

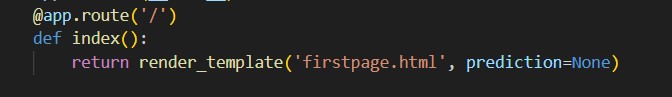
Importing the libraries.



This code first loads the saved CNN model from the “model.hdf5” file using the “tf.load()” method. After loading the model, the code creates a new Flask web application object named “app” using the Flask constructor. The “name” argument tells Flask to use the current module as the name for the application.

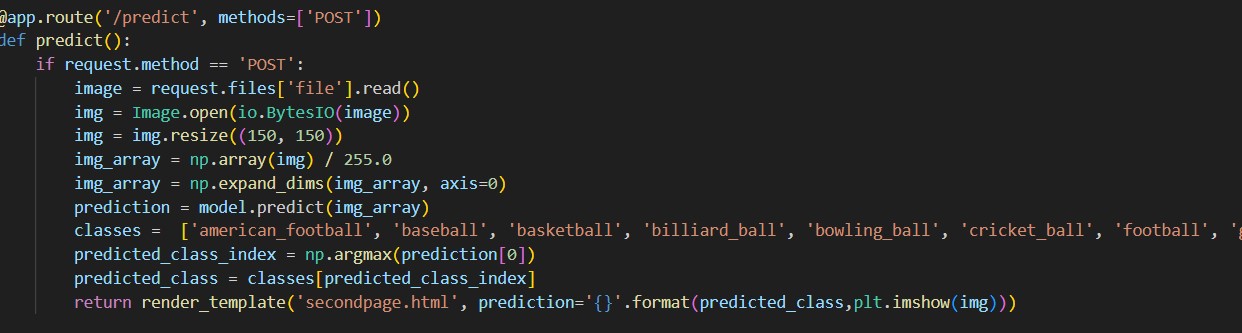


This code sets up a new route for the Flask web application using the “@app.route()” decorator. The route in this case is the root route “/”, which is the default route when the website is accessed. The function “home()” is then associated with this route. When a user accesses the root route of the website, this function is called. The “render\_template()” method is used to render an HTML template named “firstpage.html”. The “firstpage.html” is the home page.



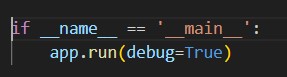
The route in this case is “/predict”. When a user accesses the “/predict” route of the website, this function is “firstpage()” called. The “render\_template()” method is used to render an HTML template named “secondpage.html”.

This code sets up another route for the Flask web application using the “@app.route()” decorator. The route in this case is “/predict”, and the method is set to GET and POST. The function “predict()” is then associated with this route. This function first loads the previously saved CNN model using “model = load(open(‘modelhdf5.pkl’))”. Then, the function receives an uploaded image from a user using “request.files[‘…’]”. The function then uses the loaded CNN model to predict which type of sports ball it is based on its image. Finally, the predicted sports ball type is passed to an HTML template, where it is displayed to the user.



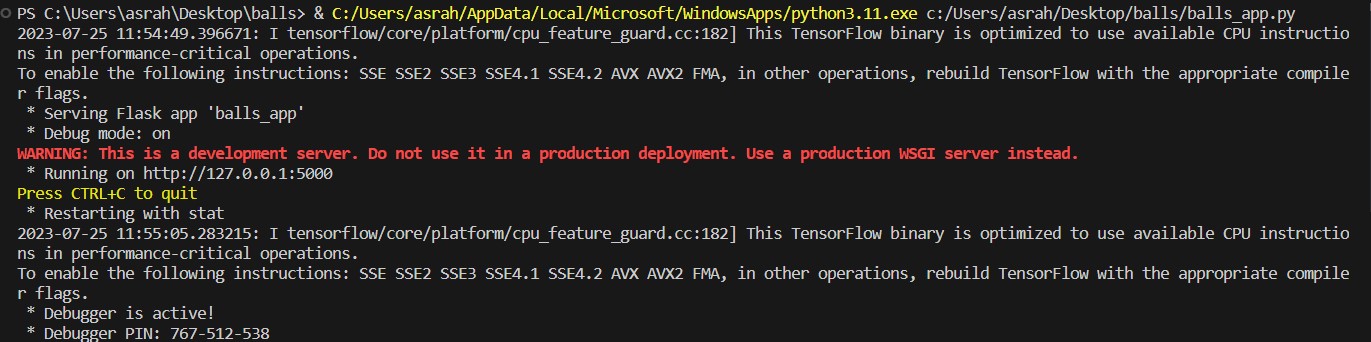
Main Function:

This code sets the entry point of the Flask application. The function “app.run()” is called, which starts the Flask development server.

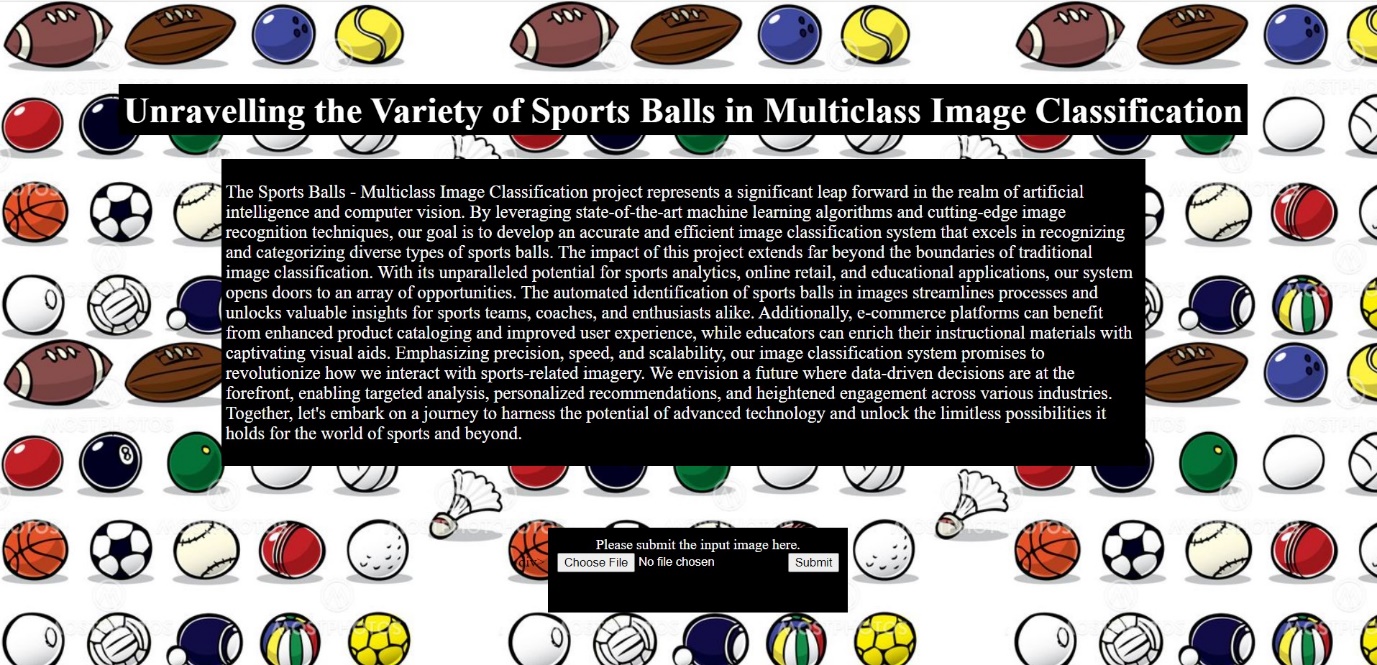


**Activity 2.3: Run the web application**

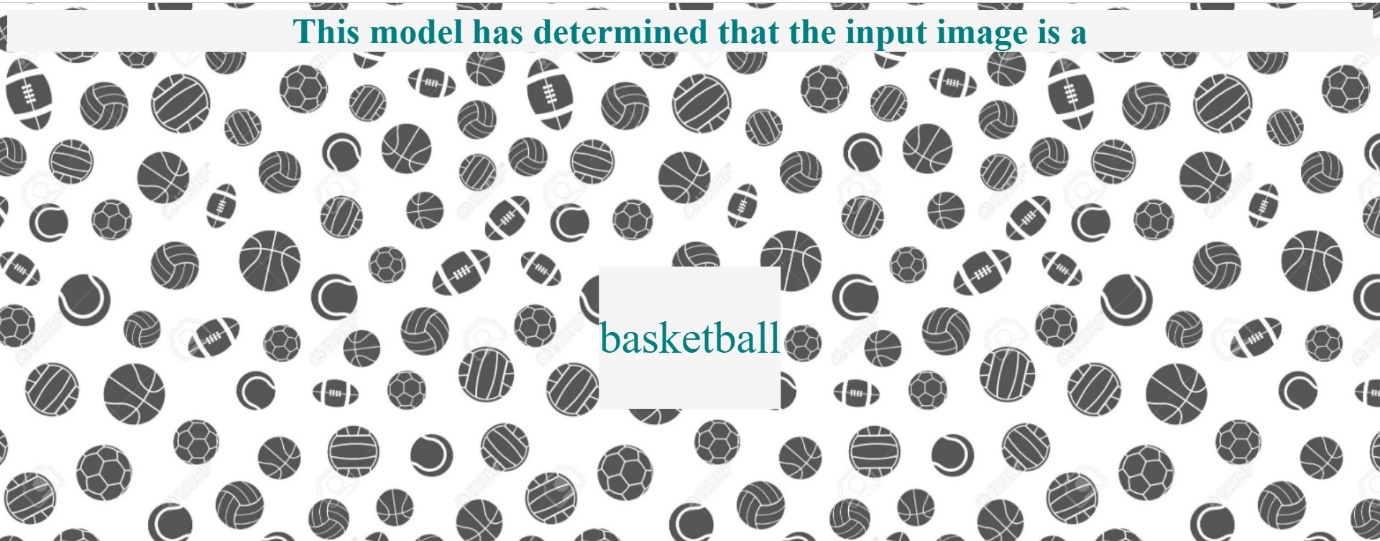
When you run the “main.py” file this window will open in output terminal. Copy http://127.0.0.1:5000 and paste this link in your browser.



This is “firstpage.html” file that appears when we paste URL into browser. To proceed next page, click on ‘Submit’ button.



On this page, a user will see the predicted classification for the input image of a sports ball.



**Milestone 7: Project Demonstration & Documentation**

Below mentioned deliverables are to be submitted along with other deliverables: Activity 1: - Record explanation Video for project end-to-end solution.