**Image Classification**

Here, we proposed the Convolutional Neural Network (CNN) architecture for image classification. Data augmentation is applied to the training set of cropped photos, which is used to train the model. Compiling the model, fitting it to the training data, and evaluating on th3e validation and test set are involved in the training process. Here accuracy is chosen as the evaluation metrics. A class dictionary and the trained model are stored for later use. To show the trained model's prediction, an image is applied for demonstration.

* Data Preparation:  
  Image files are collected from the specified directory. Then a DataFrame is created with file paths and corresponding labels. After that the data is split into training, validation and test sets.
* Data Augmentation:

Data augmentation is performed using ImageDataGenerator on the training set, which includes rescaling, shearing, zooming and horizontal flipping. Validation and test sets are rescaled.

* Data Generators:

Data generators (train\_generator, val\_generator, and test\_generator) are set up using flow\_from\_dataframe to load and preprocess images in batches.

* Model architecture:

The architecture consist of several CNN layers such as max-pooling layers for feature extraction, a fully connected layer for classification, and a softmax activation layer for multi-class classification.

* A Sequential model is created using Keras.
* Convolutional layers with max-pooling are stacked to learn hierarchical features from images.
* Flatten layer is used to flatten the 2D output to a 1D vector.
* Dense layers with dropout are added for classification.
* The output layer has as many neurons as there are classes (persons) with a softmax activation function for multi-class classification.
* Model compilation:

The model is compiled with the Adam optimizer, categorical crossentropy loss (suitable for multi-class classification), and accuracy as the evaluation metric.

* Model Training

The training of the model is done using the training generator with 10 epochs and validation data from the validation generator.

* Model Evaluation

The trained model is evaluated on the test set to get the accuracy.

* Model Saving

The trained model is saved to a file (sports\_person\_model.h5). The class dictionary (mapping labels to indices) is saved to a JSON file (class\_dictionary.json).

* Making Predictions: A sample image path is provided (sample\_image\_path). Then the image is loaded, normalized, and expanded to add a batch dimension. The model is used to predict the class probabilities for the input image. The predicted class index is obtained by finding the index with the highest probability. The predicted class index is mapped to the person's name using the loaded class dictionary. Finally, the predicted person's name is printed.