Image Classification

The proposed neural network architecture for image classification is a Convolutional Neural Network (CNN). The model is trained on a dataset of cropped images, with data augmentation applied to the training set. The training process involves compiling the model, fitting it to the training data, and evaluating on the validation and test sets. The chosen evaluation metric is accuracy. The trained model and a class dictionary are saved for future use. A sample image is used to demonstrate making predictions using the trained model.

* Data Preparation:

Image files are collected from the specified directory (path\_to\_data).

A DataFrame (df) is created with file paths and corresponding labels (person's name).

The data is split into training, validation, and test sets.

* Data Augmentation:

ImageDataGenerator is used to perform data augmentation on the training set, which includes rescaling, shearing, zooming, and horizontal flipping. Validation and test sets are only 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 consists of several convolutional layers followed by 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 model is trained 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).

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.

The predicted person's name is printed.