**Detailed Report on Image Classification of Segmented Signatures**

**1. Introduction**

This project aims to classify segmented signature images using a Convolutional Neural Network (CNN). The dataset consists of images organized by individuals, with each individual's signatures stored in separate folders. The goal is to train a model that can accurately identify the signatures based on the segmented images.

**2. Data Preparation**

**2.1 Image Preprocessing**

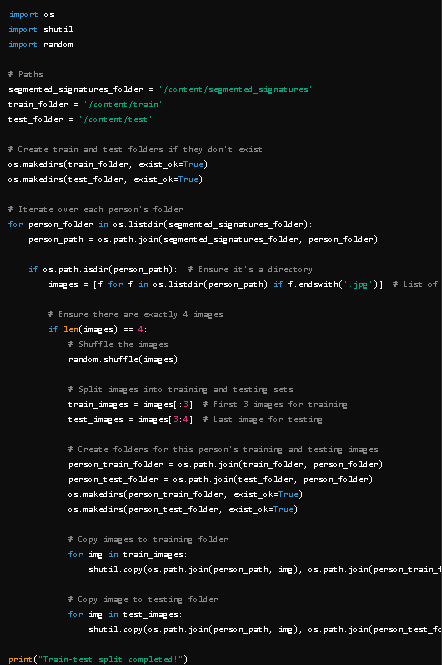
**The initial step involved preprocessing the images to prepare them for classification. The preprocessing steps included:**

1. **Removing Extra Spaces: This was crucial for ensuring that only the complete table content was included in the segmented images without any extra whitespace.**
2. **Dividing Images into Rows: Each segmented image containing signatures was processed to cut each row and store it as a separate image.**
3. **Extracting Columns: Each row was then divided into individual columns to extract the signatures of each person. This step ensured that each signature was isolated and ready for classification.**
4. **Organizing Signatures: Finally, the four signatures for each individual were stored in a separate folder under the segmented signatures directory for training and testing.**

**2.2 Data Organization**

The dataset was structured to facilitate training and testing. The signatures were segmented, and a Python script was implemented to split the dataset into training and testing folders.

* **Training Set**: Contains a specified number of images per individual for training the model.
* **Testing Set**: Contains images reserved for validating the model's performance.
* **2.2 Data Splitting Code**
* The following code snippet was used to organize the data into separate folders for training and testing:



**3. Model Architecture**

A CNN model was constructed using Keras with the following layers:

* **Convolutional Layers**: To extract features from the images.
* **MaxPooling Layers**: To reduce the size of the feature maps.
* **Flatten Layer**: To convert the 2D feature maps into 1D vectors.
* **Dense Layers**: For classification, ending with a softmax activation function.

A computer screen shot of a program

Description automatically generated

**4. Model Training**

The model was trained using the prepared training set. A defined number of epochs and batch size were set for the training process. Callbacks for model checkpointing and early stopping were used to improve model performance.

A screenshot of a computer program

Description automatically generated

**5. Results Visualization**

**5.1 Training History**

To assess the model's performance over the training epochs, plots of training and validation accuracy and loss were created.

A screenshot of a computer program

Description automatically generated