**Summary of the chosen model:**

A type of deep neural network that is particularly useful for image processing and computer vision applications is the convolutional neural network (CNN). It is highly effective in tasks such as image classification, object detection, and image recognition. CNNs is also used in a number of other fields, such as medical image analysis and natural language processing.

Key features of CNNs include:

1. Convolutional Layers: Convolutional layers are used by CNNs to automatically and adaptively learn spatial feature hierarchies from input images. Convolutional filters are used by these layers to identify textures, edges, and patterns at various sizes.
2. Pooling Layers: By downsampling the input data's spatial dimensions, pooling layers lessen computing effort and produce a more abstract feature representation. Two popular pooling techniques are max pooling and average pooling.
3. Fully Connected Layers: These layers are in charge of generating predictions based on the learned features and are typically located at the end of the network. All of the neurons in the layers above and below are connected to all of the other neurons.
4. Activation Functions: To provide non-linearity to the model and help it understand complex relationships in the data, non-linear activation functions like Rectified Linear Unit (ReLU) are used.
5. Dropout: To avoid overfitting, dropout is a regularisation technique frequently employed in CNNs. During training, it arbitrarily removes a portion of neurons, pushing the network to pick up more resilient and broadly applicable properties.

CNN has the ability to automatically learn hierarchical features from raw data makes them well-suited for tasks where the input has a grid-like topology, such as images. Additionally, transfer learning, where pre-trained CNNs on large datasets are fine-tuned for specific tasks, has further enhanced their applicability in different domains with limited data.

**Training Process:**

1. Loading and Data Preprocessing:

* Importing the required libraries—TensorFlow, NumPy, Matplotlib, OpenCV, PIL, and others—is the first step in the code.
* The dataset is made up of pictures of different celebrities that are kept in subdirectories under the 'cropped' directory.
* After loading, the images are downsized to 128 by 128 pixels and added to the dataset with labels (label) that correspond to them.

1. Train-Test Split:

* The train\_test\_split function from scikit-learn is used to divide the dataset into training and testing sets. 20% is tested and 80% is trained.

1. Data Normalization:

* tf.keras.utils.normalize is used to normalize the image's pixel values so that they range from 0 to 1. During training, normalization facilitates faster convergence.

1. Model Definition:

* The CNN model is defined with the Keras API of TensorFlow. Convolutional layers, max-pooling layers, flatten layers, and densely connected layers make up this structure.
* The sparse\_categorical\_crossentropy loss function (suited for integer labels), accuracy as the evaluation metric, and the Adam optimizer are used to create the model.

1. Model Training:

* With a batch size of 128 and 200 epochs, training takes place. In addition, the model's performance on a subset of the training data that isn't utilized for training is tracked using a 10% validation split.
* For later visualization, the training history—which includes accuracy, validation accuracy, loss, and validation loss—is saved.

1. Training Visualization:

* Two plots—one for accuracy and the other for loss—are made with Matplotlib. PNG files are used to store these plots.

1. Model Evaluation:

* The accuracy of the model is printed for evaluating the model.

**Critical Findings:**

Despite reaching an impressive 87% accuracy after 200 epochs, the model's performance was probably limited by the tiny dataset, which only included 165 photos. The model may overfit due to difficulties in learning robust and generalizable features from the little sample. Extending and broadening the training dataset and experimenting with different regularization strategies to reduce overfitting would be necessary to improve the model's performance.