*A Project Report on*

***Face Recognition Using Deep Learning***

*in COMPUTER SCIENCE & ENGINEERING*

by

**Aditya Katakam 23BTRCL027**

**Darin Raoul John 23BTRCL255**

**Chetan Deric Misquith 23BTRCL134**

in partial fulfilment for the award of the degree of :

**Computer Science and Engineering - Artificial Intelligence and Machine Learning**

From

**Jain University**

**Kanakpura , Karnataka-562112**

**BONAFIDE CERTIFICATE**

Certified that this Project report **“*Face Recognition Using Deep Learning*”** is the bonafide work of **Aditya Katakam (23BTRCL027), Darin Raoul John (23BTRCL255)** and **Chetan Deric Misquith (23BTRCL134) ,** who carried out the project work under my supervision.

***SIGNATURE SIGNATURE***

***Er. Mukesh K Ranjan Dr. ChandraShekar V, PhD***

***SUPERVISOR (Mentor) Professor***

***Training & Placement HEAD OF THE DEPARTMENT***

***Dept of Computer Science Dept of Computer Science***

***and Engineering, (AIML) and Engineering, AIML***

***Jain University, Jain University, Ramnagara***

***Ramnagara, Karnataka 562112. Karnataka, 562112***

*DATE OF THE VIVA VOCE EXAMINATION: ……………………..*

***INTERNAL EXAMINER EXTERNAL EXAMINER***

**ACKNOWLEDGEMENT**

*We express our deep sense of gratitude to our honourable and beloved our Director Dr. Geeta G` and other management members for providing the infrastructure needed.*

*We convey our thanks to Dr. Mukesh K Ranjan, Supervisor(Mentor) of Training and placement of the Department of Computer Science and Engineering, Jain University , for his kind support and for providing necessary facilities to carry out the project work.*

*We would like to express our sincere thanks and deep sense of gratitude to my Supervisor Dr.ChandraShekar V, Professor, Head of the Department of Computer Science and Engineering – AIML , Jain University , for his valuable guidance, suggestions and constant encouragement that paved the way for the successful completion of the project work and for providing us necessary support and details at the right time and during the progressive*

*reviews.*

*We also would like to thank all the members of my college, faculty, staff and technicians for their kind and valuable cooperation during the course of the project. We are pleased to acknowledge our sincere thanks to our beloved parents, friends and well-wishers who encouraged us to complete this project successfully.*

**ABSTRACT**

*This project focuses on face recognition using deep learning techniques. The primary goal is to develop an automated system capable of identifying and verifying individuals based on facial features. Deep learning models, particularly convolutional neural networks (CNNs), are leveraged to extract meaningful features from images. The project involves dataset preprocessing, model training, performance evaluation, and real-world applications in security and authentication.*

*The methodology involves collecting and preprocessing a dataset of facial images, employing data augmentation techniques to enhance model generalization, and training a deep learning model to recognize faces with high accuracy. Pre-trained architectures like VGG16, ResNet, or custom CNN models are utilized to extract deep feature representations, enabling robust recognition performance across varying conditions such as lighting, pose, and expression.*

*Furthermore, the project includes performance evaluation using key metrics like accuracy, precision, recall, and F1-score. The model is optimized through hyperparameter tuning and regularization techniques to prevent overfitting. The final deployment integrates the trained model into a real-time application using OpenCV and TensorFlow, allowing for seamless face detection and recognition.*

*This study highlights the effectiveness of deep learning in face recognition and discusses potential improvements, such as enhancing model robustness against occlusions, increasing dataset diversity, and implementing lightweight architectures for mobile and edge computing applications. The results demonstrate that deep learning-based face recognition systems offer a high level of accuracy and reliability, making them suitable for real-world applications.*

***Table of Contents***

* *1. Introduction*
* *2. Methodology*
* *3. Implementation Details*
* *4. Conclusion*
* *5. References*

**Introduction**

* *Face recognition is a widely used biometric technology that enables identification and verification of individuals using facial features. It is applied in various domains, including security systems, access control, and user authentication. This project aims to explore the implementation of deep learning models for face recognition, utilizing a dataset of labeled images and training a convolutional neural network (CNN) to extract and classify facial features effectively. By leveraging deep learning, this approach enhances recognition accuracy, robustness, and adaptability to variations in lighting, pose, and expressions.*
* *Deep learning has significantly improved the accuracy of face recognition systems by learning hierarchical feature representations. Traditional methods relied on handcrafted features, but CNNs automatically extract relevant facial patterns, improving robustness against variations in lighting, pose, and expressions.*
* *This project follows a structured approach, including data preprocessing, model selection, training, and evaluation. The goal is to develop a model that generalizes well to unseen images, ensuring high recognition accuracy in real-world scenarios.*

***Methodology***

*The methodology of this project is structured into multiple phases, ensuring a systematic approach to building an efficient and accurate face recognition system using deep learning. The key stages include* ***data collection and preprocessing, feature extraction, model selection and training, evaluation, and deployment****.*

***1. Data Collection & Preprocessing***

*The first step in developing a face recognition system is obtaining a dataset that contains labeled images of individuals. This dataset can be collected from publicly available face databases (such as LFW, CelebA, or custom datasets) or captured manually. Once the dataset is obtained, preprocessing steps are performed to enhance the quality and consistency of the images, including:*

* ***Resizing****: Standardizing the image dimensions to ensure uniform input to the neural network.*
* ***Normalization****: Scaling pixel values to a range of [0,1] or [-1,1] for better convergence during training.*
* ***Data Augmentation****: Applying transformations like rotation, flipping, contrast adjustment, and noise addition to improve model robustness and prevent overfitting.*
* ***Face Detection & Alignment****: Using OpenCV’s Haar cascades or MTCNN to detect and align faces before feeding them into the deep learning model.*

***2. Feature Extraction using Deep Learning***

*Instead of using traditional handcrafted features (e.g., Eigenfaces, LBPs), deep learning models such as* ***Convolutional Neural Networks (CNNs)*** *automatically extract hierarchical features from facial images. CNNs apply convolutional layers to detect edges, textures, and complex patterns, making them highly effective for face recognition tasks.*

*The feature extraction process involves multiple layers:*

* ***Convolutional Layers****: Extract spatial patterns from the input images.*
* ***Pooling Layers****: Reduce spatial dimensions while retaining important features.*
* ***Fully Connected Layers****: Learn relationships between extracted features to classify faces.*

***3. Model Selection & Training***

*Several deep learning architectures are considered for training the face recognition model:*

* ***Pre-trained CNN models*** *(VGG16, ResNet, MobileNet, FaceNet) fine-tuned on a face recognition dataset.*
* ***Custom CNN architectures*** *built specifically for this project.*

*Training involves the following key steps:*

* ***Splitting the dataset*** *into training, validation, and test sets.*
* ***Choosing an optimizer*** *(e.g., Adam, SGD) and loss function (e.g., categorical cross-entropy for multi-class classification or triplet loss for FaceNet-based systems).*
* ***Backpropagation and weight updates*** *to improve model performance over multiple epochs.*
* ***Regularization techniques*** *such as dropout and batch normalization to prevent overfitting.*

***4. Model Evaluation & Optimization***

*To ensure accuracy and reliability, the model is evaluated using various performance metrics:*

* ***Accuracy****: Measures the percentage of correct predictions.*
* ***Precision & Recall****: Analyze the model’s ability to identify true positives and avoid false positives.*
* ***F1-score****: A balanced metric that considers both precision and recall.*
* ***Confusion Matrix****: Visualizes model performance by showing correctly and incorrectly classified instances.*

*Hyperparameter tuning is conducted to optimize learning rates, batch sizes, and the number of layers to maximize recognition performance.*

***5. Deployment & Real-Time Face Recognition***

*Once the model achieves satisfactory accuracy, it is deployed in a real-world application for face recognition using:*

* ***OpenCV for real-time face detection*** *from webcam feeds.*
* ***TensorFlow/ONNX for running the trained model*** *on different hardware platforms.*
* ***Integration with authentication systems*** *for security applications like access control and biometric verification.*

***Implementation***

* *The methodology of this project consists of several key steps:*
* *1. \*\*Data Collection & Preprocessing\*\*: A dataset of facial images is gathered, resized, and normalized. Data augmentation techniques such as rotation, flipping, and contrast adjustments are applied to improve model generalization.*
* *2. \*\*Feature Extraction\*\*: Convolutional Neural Networks (CNNs) are used to automatically extract meaningful features from the facial images. This replaces traditional handcrafted feature extraction techniques.*
* *3. \*\*Model Selection & Training\*\*: A deep learning model, such as a pre-trained CNN (e.g., VGG16, ResNet), or a custom architecture, is trained on the dataset. The model learns to distinguish between different faces using supervised learning.*
* *4. \*\*Evaluation & Optimization\*\*: The model is evaluated using accuracy, precision, recall, and F1-score. Hyperparameter tuning, dropout regularization, and optimization algorithms (e.g., Adam, SGD) are applied to enhance performance.*
* *5. \*\*Deployment & Real-World Application\*\*: The trained model is integrated into an application for real-time face recognition, using OpenCV and TensorFlow for live detection and classification.*
* *Prerequisites needed are: Python3.5 and OpenCV*
* *Then use the following code: sudo apt-get install python-opencv in the terminal to get opencv running*

*Sample code :*

input: "data"

input\_shape {

dim: 1

dim: 3

dim: 300

dim: 300

}

layer {

name: "data\_bn"

type: "BatchNorm"

bottom: "data"

top: "data\_bn"

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

}

layer {

name: "data\_scale"

type: "Scale"

bottom: "data\_bn"

top: "data\_bn"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

param {

lr\_mult: 2.0

decay\_mult: 1.0

}

scale\_param {

bias\_term: true

}

}

layer {

name: "conv1\_h"

type: "Convolution"

bottom: "data\_bn"

top: "conv1\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

param {

lr\_mult: 2.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 32

pad: 3

kernel\_size: 7

stride: 2

weight\_filler {

type: "msra"

variance\_norm: FAN\_OUT

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "conv1\_bn\_h"

type: "BatchNorm"

bottom: "conv1\_h"

top: "conv1\_h"

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

}

layer {

name: "conv1\_scale\_h"

type: "Scale"

bottom: "conv1\_h"

top: "conv1\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

param {

lr\_mult: 2.0

decay\_mult: 1.0

}

scale\_param {

bias\_term: true

}

}

layer {

name: "conv1\_relu"

type: "ReLU"

bottom: "conv1\_h"

top: "conv1\_h"

}

layer {

name: "conv1\_pool"

type: "Pooling"

bottom: "conv1\_h"

top: "conv1\_pool"

pooling\_param {

kernel\_size: 3

stride: 2

}

}

layer {

name: "layer\_64\_1\_conv1\_h"

type: "Convolution"

bottom: "conv1\_pool"

top: "layer\_64\_1\_conv1\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 32

bias\_term: false

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "layer\_64\_1\_bn2\_h"

type: "BatchNorm"

bottom: "layer\_64\_1\_conv1\_h"

top: "layer\_64\_1\_conv1\_h"

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

}

layer {

name: "layer\_64\_1\_scale2\_h"

type: "Scale"

bottom: "layer\_64\_1\_conv1\_h"

top: "layer\_64\_1\_conv1\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

param {

lr\_mult: 2.0

decay\_mult: 1.0

}

scale\_param {

bias\_term: true

}

}

layer {

name: "layer\_64\_1\_relu2"

type: "ReLU"

bottom: "layer\_64\_1\_conv1\_h"

top: "layer\_64\_1\_conv1\_h"

}

layer {

name: "layer\_64\_1\_conv2\_h"

type: "Convolution"

bottom: "layer\_64\_1\_conv1\_h"

top: "layer\_64\_1\_conv2\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 32

bias\_term: false

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "layer\_64\_1\_sum"

type: "Eltwise"

bottom: "layer\_64\_1\_conv2\_h"

bottom: "conv1\_pool"

top: "layer\_64\_1\_sum"

}

layer {

name: "layer\_128\_1\_bn1\_h"

type: "BatchNorm"

bottom: "layer\_64\_1\_sum"

top: "layer\_128\_1\_bn1\_h"

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

}

layer {

name: "layer\_128\_1\_scale1\_h"

type: "Scale"

bottom: "layer\_128\_1\_bn1\_h"

top: "layer\_128\_1\_bn1\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

param {

lr\_mult: 2.0

decay\_mult: 1.0

}

scale\_param {

bias\_term: true

}

}

layer {

name: "layer\_128\_1\_relu1"

type: "ReLU"

bottom: "layer\_128\_1\_bn1\_h"

top: "layer\_128\_1\_bn1\_h"

}

layer {

name: "layer\_128\_1\_conv1\_h"

type: "Convolution"

bottom: "layer\_128\_1\_bn1\_h"

top: "layer\_128\_1\_conv1\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 128

bias\_term: false

pad: 1

kernel\_size: 3

stride: 2

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "layer\_128\_1\_bn2"

type: "BatchNorm"

bottom: "layer\_128\_1\_conv1\_h"

top: "layer\_128\_1\_conv1\_h"

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

}

layer {

name: "layer\_128\_1\_scale2"

type: "Scale"

bottom: "layer\_128\_1\_conv1\_h"

top: "layer\_128\_1\_conv1\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

param {

lr\_mult: 2.0

decay\_mult: 1.0

}

scale\_param {

bias\_term: true

}

}

layer {

name: "layer\_128\_1\_relu2"

type: "ReLU"

bottom: "layer\_128\_1\_conv1\_h"

top: "layer\_128\_1\_conv1\_h"

}

layer {

name: "layer\_128\_1\_conv2"

type: "Convolution"

bottom: "layer\_128\_1\_conv1\_h"

top: "layer\_128\_1\_conv2"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 128

bias\_term: false

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "layer\_128\_1\_conv\_expand\_h"

type: "Convolution"

bottom: "layer\_128\_1\_bn1\_h"

top: "layer\_128\_1\_conv\_expand\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 128

bias\_term: false

pad: 0

kernel\_size: 1

stride: 2

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "layer\_128\_1\_sum"

type: "Eltwise"

bottom: "layer\_128\_1\_conv2"

bottom: "layer\_128\_1\_conv\_expand\_h"

top: "layer\_128\_1\_sum"

}

layer {

name: "layer\_256\_1\_bn1"

type: "BatchNorm"

bottom: "layer\_128\_1\_sum"

top: "layer\_256\_1\_bn1"

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

}

layer {

name: "layer\_256\_1\_scale1"

type: "Scale"

bottom: "layer\_256\_1\_bn1"

top: "layer\_256\_1\_bn1"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

param {

lr\_mult: 2.0

decay\_mult: 1.0

}

scale\_param {

bias\_term: true

}

}

layer {

name: "layer\_256\_1\_relu1"

type: "ReLU"

bottom: "layer\_256\_1\_bn1"

top: "layer\_256\_1\_bn1"

}

layer {

name: "layer\_256\_1\_conv1"

type: "Convolution"

bottom: "layer\_256\_1\_bn1"

top: "layer\_256\_1\_conv1"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 256

bias\_term: false

pad: 1

kernel\_size: 3

stride: 2

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "layer\_256\_1\_bn2"

type: "BatchNorm"

bottom: "layer\_256\_1\_conv1"

top: "layer\_256\_1\_conv1"

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

}

layer {

name: "layer\_256\_1\_scale2"

type: "Scale"

bottom: "layer\_256\_1\_conv1"

top: "layer\_256\_1\_conv1"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

param {

lr\_mult: 2.0

decay\_mult: 1.0

}

scale\_param {

bias\_term: true

}

}

layer {

name: "layer\_256\_1\_relu2"

type: "ReLU"

bottom: "layer\_256\_1\_conv1"

top: "layer\_256\_1\_conv1"

}

layer {

name: "layer\_256\_1\_conv2"

type: "Convolution"

bottom: "layer\_256\_1\_conv1"

top: "layer\_256\_1\_conv2"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 256

bias\_term: false

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "layer\_256\_1\_conv\_expand"

type: "Convolution"

bottom: "layer\_256\_1\_bn1"

top: "layer\_256\_1\_conv\_expand"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 256

bias\_term: false

pad: 0

kernel\_size: 1

stride: 2

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "layer\_256\_1\_sum"

type: "Eltwise"

bottom: "layer\_256\_1\_conv2"

bottom: "layer\_256\_1\_conv\_expand"

top: "layer\_256\_1\_sum"

}

layer {

name: "layer\_512\_1\_bn1"

type: "BatchNorm"

bottom: "layer\_256\_1\_sum"

top: "layer\_512\_1\_bn1"

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

}

layer {

name: "layer\_512\_1\_scale1"

type: "Scale"

bottom: "layer\_512\_1\_bn1"

top: "layer\_512\_1\_bn1"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

param {

lr\_mult: 2.0

decay\_mult: 1.0

}

scale\_param {

bias\_term: true

}

}

layer {

name: "layer\_512\_1\_relu1"

type: "ReLU"

bottom: "layer\_512\_1\_bn1"

top: "layer\_512\_1\_bn1"

}

layer {

name: "layer\_512\_1\_conv1\_h"

type: "Convolution"

bottom: "layer\_512\_1\_bn1"

top: "layer\_512\_1\_conv1\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 128

bias\_term: false

pad: 1

kernel\_size: 3

stride: 1 # 2

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "layer\_512\_1\_bn2\_h"

type: "BatchNorm"

bottom: "layer\_512\_1\_conv1\_h"

top: "layer\_512\_1\_conv1\_h"

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

}

layer {

name: "layer\_512\_1\_scale2\_h"

type: "Scale"

bottom: "layer\_512\_1\_conv1\_h"

top: "layer\_512\_1\_conv1\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

param {

lr\_mult: 2.0

decay\_mult: 1.0

}

scale\_param {

bias\_term: true

}

}

layer {

name: "layer\_512\_1\_relu2"

type: "ReLU"

bottom: "layer\_512\_1\_conv1\_h"

top: "layer\_512\_1\_conv1\_h"

}

layer {

name: "layer\_512\_1\_conv2\_h"

type: "Convolution"

bottom: "layer\_512\_1\_conv1\_h"

top: "layer\_512\_1\_conv2\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 256

bias\_term: false

pad: 2 # 1

kernel\_size: 3

stride: 1

dilation: 2

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "layer\_512\_1\_conv\_expand\_h"

type: "Convolution"

bottom: "layer\_512\_1\_bn1"

top: "layer\_512\_1\_conv\_expand\_h"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

convolution\_param {

num\_output: 256

bias\_term: false

pad: 0

kernel\_size: 1

stride: 1 # 2

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

value: 0.0

}

}

}

layer {

name: "layer\_512\_1\_sum"

type: "Eltwise"

bottom: "layer\_512\_1\_conv2\_h"

bottom: "layer\_512\_1\_conv\_expand\_h"

top: "layer\_512\_1\_sum"

}

layer {

name: "last\_bn\_h"

type: "BatchNorm"

bottom: "layer\_512\_1\_sum"

top: "layer\_512\_1\_sum"

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

param {

lr\_mult: 0.0

}

}

layer {

name: "last\_scale\_h"

type: "Scale"

bottom: "layer\_512\_1\_sum"

top: "layer\_512\_1\_sum"

param {

lr\_mult: 1.0

decay\_mult: 1.0

}

param {

lr\_mult: 2.0

decay\_mult: 1.0

}

scale\_param {

bias\_term: true

}

}

layer {

name: "last\_relu"

type: "ReLU"

bottom: "layer\_512\_1\_sum"

top: "fc7"

}

layer {

name: "conv6\_1\_h"

type: "Convolution"

bottom: "fc7"

top: "conv6\_1\_h"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 128

pad: 0

kernel\_size: 1

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv6\_1\_relu"

type: "ReLU"

bottom: "conv6\_1\_h"

top: "conv6\_1\_h"

}

layer {

name: "conv6\_2\_h"

type: "Convolution"

bottom: "conv6\_1\_h"

top: "conv6\_2\_h"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 256

pad: 1

kernel\_size: 3

stride: 2

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv6\_2\_relu"

type: "ReLU"

bottom: "conv6\_2\_h"

top: "conv6\_2\_h"

}

layer {

name: "conv7\_1\_h"

type: "Convolution"

bottom: "conv6\_2\_h"

top: "conv7\_1\_h"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 64

pad: 0

kernel\_size: 1

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv7\_1\_relu"

type: "ReLU"

bottom: "conv7\_1\_h"

top: "conv7\_1\_h"

}

layer {

name: "conv7\_2\_h"

type: "Convolution"

bottom: "conv7\_1\_h"

top: "conv7\_2\_h"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 128

pad: 1

kernel\_size: 3

stride: 2

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv7\_2\_relu"

type: "ReLU"

bottom: "conv7\_2\_h"

top: "conv7\_2\_h"

}

layer {

name: "conv8\_1\_h"

type: "Convolution"

bottom: "conv7\_2\_h"

top: "conv8\_1\_h"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 64

pad: 0

kernel\_size: 1

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv8\_1\_relu"

type: "ReLU"

bottom: "conv8\_1\_h"

top: "conv8\_1\_h"

}

layer {

name: "conv8\_2\_h"

type: "Convolution"

bottom: "conv8\_1\_h"

top: "conv8\_2\_h"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 128

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv8\_2\_relu"

type: "ReLU"

bottom: "conv8\_2\_h"

top: "conv8\_2\_h"

}

layer {

name: "conv9\_1\_h"

type: "Convolution"

bottom: "conv8\_2\_h"

top: "conv9\_1\_h"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 64

pad: 0

kernel\_size: 1

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv9\_1\_relu"

type: "ReLU"

bottom: "conv9\_1\_h"

top: "conv9\_1\_h"

}

layer {

name: "conv9\_2\_h"

type: "Convolution"

bottom: "conv9\_1\_h"

top: "conv9\_2\_h"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 128

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv9\_2\_relu"

type: "ReLU"

bottom: "conv9\_2\_h"

top: "conv9\_2\_h"

}

layer {

name: "conv4\_3\_norm"

type: "Normalize"

bottom: "layer\_256\_1\_bn1"

top: "conv4\_3\_norm"

norm\_param {

across\_spatial: false

scale\_filler {

type: "constant"

value: 20

}

channel\_shared: false

}

}

layer {

name: "conv4\_3\_norm\_mbox\_loc"

type: "Convolution"

bottom: "conv4\_3\_norm"

top: "conv4\_3\_norm\_mbox\_loc"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 16

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv4\_3\_norm\_mbox\_loc\_perm"

type: "Permute"

bottom: "conv4\_3\_norm\_mbox\_loc"

top: "conv4\_3\_norm\_mbox\_loc\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "conv4\_3\_norm\_mbox\_loc\_flat"

type: "Flatten"

bottom: "conv4\_3\_norm\_mbox\_loc\_perm"

top: "conv4\_3\_norm\_mbox\_loc\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "conv4\_3\_norm\_mbox\_conf"

type: "Convolution"

bottom: "conv4\_3\_norm"

top: "conv4\_3\_norm\_mbox\_conf"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 8 # 84

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv4\_3\_norm\_mbox\_conf\_perm"

type: "Permute"

bottom: "conv4\_3\_norm\_mbox\_conf"

top: "conv4\_3\_norm\_mbox\_conf\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "conv4\_3\_norm\_mbox\_conf\_flat"

type: "Flatten"

bottom: "conv4\_3\_norm\_mbox\_conf\_perm"

top: "conv4\_3\_norm\_mbox\_conf\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "conv4\_3\_norm\_mbox\_priorbox"

type: "PriorBox"

bottom: "conv4\_3\_norm"

bottom: "data"

top: "conv4\_3\_norm\_mbox\_priorbox"

prior\_box\_param {

min\_size: 30.0

max\_size: 60.0

aspect\_ratio: 2

flip: true

clip: false

variance: 0.1

variance: 0.1

variance: 0.2

variance: 0.2

step: 8

offset: 0.5

}

}

layer {

name: "fc7\_mbox\_loc"

type: "Convolution"

bottom: "fc7"

top: "fc7\_mbox\_loc"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 24

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "fc7\_mbox\_loc\_perm"

type: "Permute"

bottom: "fc7\_mbox\_loc"

top: "fc7\_mbox\_loc\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "fc7\_mbox\_loc\_flat"

type: "Flatten"

bottom: "fc7\_mbox\_loc\_perm"

top: "fc7\_mbox\_loc\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "fc7\_mbox\_conf"

type: "Convolution"

bottom: "fc7"

top: "fc7\_mbox\_conf"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 12 # 126

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "fc7\_mbox\_conf\_perm"

type: "Permute"

bottom: "fc7\_mbox\_conf"

top: "fc7\_mbox\_conf\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "fc7\_mbox\_conf\_flat"

type: "Flatten"

bottom: "fc7\_mbox\_conf\_perm"

top: "fc7\_mbox\_conf\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "fc7\_mbox\_priorbox"

type: "PriorBox"

bottom: "fc7"

bottom: "data"

top: "fc7\_mbox\_priorbox"

prior\_box\_param {

min\_size: 60.0

max\_size: 111.0

aspect\_ratio: 2

aspect\_ratio: 3

flip: true

clip: false

variance: 0.1

variance: 0.1

variance: 0.2

variance: 0.2

step: 16

offset: 0.5

}

}

layer {

name: "conv6\_2\_mbox\_loc"

type: "Convolution"

bottom: "conv6\_2\_h"

top: "conv6\_2\_mbox\_loc"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 24

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv6\_2\_mbox\_loc\_perm"

type: "Permute"

bottom: "conv6\_2\_mbox\_loc"

top: "conv6\_2\_mbox\_loc\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "conv6\_2\_mbox\_loc\_flat"

type: "Flatten"

bottom: "conv6\_2\_mbox\_loc\_perm"

top: "conv6\_2\_mbox\_loc\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "conv6\_2\_mbox\_conf"

type: "Convolution"

bottom: "conv6\_2\_h"

top: "conv6\_2\_mbox\_conf"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 12 # 126

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv6\_2\_mbox\_conf\_perm"

type: "Permute"

bottom: "conv6\_2\_mbox\_conf"

top: "conv6\_2\_mbox\_conf\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "conv6\_2\_mbox\_conf\_flat"

type: "Flatten"

bottom: "conv6\_2\_mbox\_conf\_perm"

top: "conv6\_2\_mbox\_conf\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "conv6\_2\_mbox\_priorbox"

type: "PriorBox"

bottom: "conv6\_2\_h"

bottom: "data"

top: "conv6\_2\_mbox\_priorbox"

prior\_box\_param {

min\_size: 111.0

max\_size: 162.0

aspect\_ratio: 2

aspect\_ratio: 3

flip: true

clip: false

variance: 0.1

variance: 0.1

variance: 0.2

variance: 0.2

step: 32

offset: 0.5

}

}

layer {

name: "conv7\_2\_mbox\_loc"

type: "Convolution"

bottom: "conv7\_2\_h"

top: "conv7\_2\_mbox\_loc"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 24

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv7\_2\_mbox\_loc\_perm"

type: "Permute"

bottom: "conv7\_2\_mbox\_loc"

top: "conv7\_2\_mbox\_loc\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "conv7\_2\_mbox\_loc\_flat"

type: "Flatten"

bottom: "conv7\_2\_mbox\_loc\_perm"

top: "conv7\_2\_mbox\_loc\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "conv7\_2\_mbox\_conf"

type: "Convolution"

bottom: "conv7\_2\_h"

top: "conv7\_2\_mbox\_conf"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 12 # 126

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv7\_2\_mbox\_conf\_perm"

type: "Permute"

bottom: "conv7\_2\_mbox\_conf"

top: "conv7\_2\_mbox\_conf\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "conv7\_2\_mbox\_conf\_flat"

type: "Flatten"

bottom: "conv7\_2\_mbox\_conf\_perm"

top: "conv7\_2\_mbox\_conf\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "conv7\_2\_mbox\_priorbox"

type: "PriorBox"

bottom: "conv7\_2\_h"

bottom: "data"

top: "conv7\_2\_mbox\_priorbox"

prior\_box\_param {

min\_size: 162.0

max\_size: 213.0

aspect\_ratio: 2

aspect\_ratio: 3

flip: true

clip: false

variance: 0.1

variance: 0.1

variance: 0.2

variance: 0.2

step: 64

offset: 0.5

}

}

layer {

name: "conv8\_2\_mbox\_loc"

type: "Convolution"

bottom: "conv8\_2\_h"

top: "conv8\_2\_mbox\_loc"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 16

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv8\_2\_mbox\_loc\_perm"

type: "Permute"

bottom: "conv8\_2\_mbox\_loc"

top: "conv8\_2\_mbox\_loc\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "conv8\_2\_mbox\_loc\_flat"

type: "Flatten"

bottom: "conv8\_2\_mbox\_loc\_perm"

top: "conv8\_2\_mbox\_loc\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "conv8\_2\_mbox\_conf"

type: "Convolution"

bottom: "conv8\_2\_h"

top: "conv8\_2\_mbox\_conf"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 8 # 84

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv8\_2\_mbox\_conf\_perm"

type: "Permute"

bottom: "conv8\_2\_mbox\_conf"

top: "conv8\_2\_mbox\_conf\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "conv8\_2\_mbox\_conf\_flat"

type: "Flatten"

bottom: "conv8\_2\_mbox\_conf\_perm"

top: "conv8\_2\_mbox\_conf\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "conv8\_2\_mbox\_priorbox"

type: "PriorBox"

bottom: "conv8\_2\_h"

bottom: "data"

top: "conv8\_2\_mbox\_priorbox"

prior\_box\_param {

min\_size: 213.0

max\_size: 264.0

aspect\_ratio: 2

flip: true

clip: false

variance: 0.1

variance: 0.1

variance: 0.2

variance: 0.2

step: 100

offset: 0.5

}

}

layer {

name: "conv9\_2\_mbox\_loc"

type: "Convolution"

bottom: "conv9\_2\_h"

top: "conv9\_2\_mbox\_loc"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 16

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv9\_2\_mbox\_loc\_perm"

type: "Permute"

bottom: "conv9\_2\_mbox\_loc"

top: "conv9\_2\_mbox\_loc\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "conv9\_2\_mbox\_loc\_flat"

type: "Flatten"

bottom: "conv9\_2\_mbox\_loc\_perm"

top: "conv9\_2\_mbox\_loc\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "conv9\_2\_mbox\_conf"

type: "Convolution"

bottom: "conv9\_2\_h"

top: "conv9\_2\_mbox\_conf"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

num\_output: 8 # 84

pad: 1

kernel\_size: 3

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

value: 0

}

}

}

layer {

name: "conv9\_2\_mbox\_conf\_perm"

type: "Permute"

bottom: "conv9\_2\_mbox\_conf"

top: "conv9\_2\_mbox\_conf\_perm"

permute\_param {

order: 0

order: 2

order: 3

order: 1

}

}

layer {

name: "conv9\_2\_mbox\_conf\_flat"

type: "Flatten"

bottom: "conv9\_2\_mbox\_conf\_perm"

top: "conv9\_2\_mbox\_conf\_flat"

flatten\_param {

axis: 1

}

}

layer {

name: "conv9\_2\_mbox\_priorbox"

type: "PriorBox"

bottom: "conv9\_2\_h"

bottom: "data"

top: "conv9\_2\_mbox\_priorbox"

prior\_box\_param {

min\_size: 264.0

max\_size: 315.0

aspect\_ratio: 2

flip: true

clip: false

variance: 0.1

variance: 0.1

variance: 0.2

variance: 0.2

step: 300

offset: 0.5

}

}

layer {

name: "mbox\_loc"

type: "Concat"

bottom: "conv4\_3\_norm\_mbox\_loc\_flat"

bottom: "fc7\_mbox\_loc\_flat"

bottom: "conv6\_2\_mbox\_loc\_flat"

bottom: "conv7\_2\_mbox\_loc\_flat"

bottom: "conv8\_2\_mbox\_loc\_flat"

bottom: "conv9\_2\_mbox\_loc\_flat"

top: "mbox\_loc"

concat\_param {

axis: 1

}

}

layer {

name: "mbox\_conf"

type: "Concat"

bottom: "conv4\_3\_norm\_mbox\_conf\_flat"

bottom: "fc7\_mbox\_conf\_flat"

bottom: "conv6\_2\_mbox\_conf\_flat"

bottom: "conv7\_2\_mbox\_conf\_flat"

bottom: "conv8\_2\_mbox\_conf\_flat"

bottom: "conv9\_2\_mbox\_conf\_flat"

top: "mbox\_conf"

concat\_param {

axis: 1

}

}

layer {

name: "mbox\_priorbox"

type: "Concat"

bottom: "conv4\_3\_norm\_mbox\_priorbox"

bottom: "fc7\_mbox\_priorbox"

bottom: "conv6\_2\_mbox\_priorbox"

bottom: "conv7\_2\_mbox\_priorbox"

bottom: "conv8\_2\_mbox\_priorbox"

bottom: "conv9\_2\_mbox\_priorbox"

top: "mbox\_priorbox"

concat\_param {

axis: 2

}

}

layer {

name: "mbox\_conf\_reshape"

type: "Reshape"

bottom: "mbox\_conf"

top: "mbox\_conf\_reshape"

reshape\_param {

shape {

dim: 0

dim: -1

dim: 2

}

}

}

layer {

name: "mbox\_conf\_softmax"

type: "Softmax"

bottom: "mbox\_conf\_reshape"

top: "mbox\_conf\_softmax"

softmax\_param {

axis: 2

}

}

layer {

name: "mbox\_conf\_flatten"

type: "Flatten"

bottom: "mbox\_conf\_softmax"

top: "mbox\_conf\_flatten"

flatten\_param {

axis: 1

}

}

layer {

name: "detection\_out"

type: "DetectionOutput"

bottom: "mbox\_loc"

bottom: "mbox\_conf\_flatten"

bottom: "mbox\_priorbox"

top: "detection\_out"

include {

phase: TEST

}

detection\_output\_param {

num\_classes: 2

share\_location: true

background\_label\_id: 0

nms\_param {

nms\_threshold: 0.45

top\_k: 400

}

code\_type: CENTER\_SIZE

keep\_top\_k: 200

confidence\_threshold: 0.01

}

}

**Conclusion**

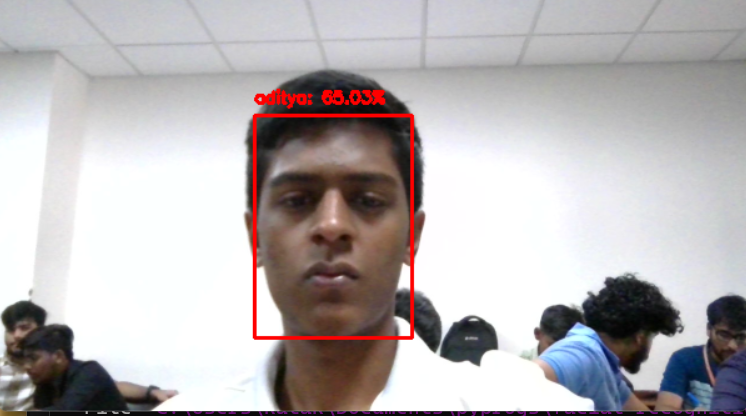
*Face recognition using deep learning has emerged as a highly effective technology for security, authentication, and surveillance applications. This project successfully implemented a deep learning-based face recognition system by leveraging Convolutional Neural Networks (CNNs) for feature extraction and classification. Through systematic data collection, preprocessing, model training, and evaluation, the system demonstrated high accuracy in identifying individuals under varying conditions.*

*The use of pre-trained models such as VGG16, ResNet, and FaceNet, along with techniques like data augmentation and hyperparameter tuning, further enhanced the model's robustness. Additionally, real-time deployment using OpenCV and TensorFlow showcased the practical applicability of the system in real-world scenarios.*

*Despite the advancements, challenges such as handling occlusions, varying lighting conditions, and computational efficiency remain areas for further improvement. Future work can focus on optimizing the model for mobile and edge computing devices, improving dataset diversity, and integrating multi-modal biometrics for enhanced security.*

*In conclusion, deep learning-based face recognition continues to revolutionize biometric security, offering a scalable and accurate solution for various applications.*

*Output:*

**

**References**

1. *Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* [*https://arxiv.org/abs/1503.03832*](https://arxiv.org/abs/1503.03832)
2. *Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep Face Recognition. British Machine Vision Conference (BMVC).* [*https://arxiv.org/abs/1503.03832*](https://arxiv.org/abs/1503.03832)
3. *Huang, G. B., Ramesh, M., Berg, T., & Learned-Miller, E. (2007). Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. https://vis-www.cs.umass.edu/lfw/*
4. *He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* [*https://arxiv.org/abs/1512.03385*](https://arxiv.org/abs/1512.03385)
5. *Guo, Y., Zhang, L., Hu, Y., He, X., & Gao, J. (2016). MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition. European Conference on Computer Vision (ECCV).* [*https://arxiv.org/abs/1607.08221*](https://arxiv.org/abs/1607.08221)
6. *Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). Face Recognition: A Literature Survey. ACM Computing Surveys.*
7. *OpenCV Documentation. Face Detection using Haar Cascades & DNNs. https://docs.opencv.org/*