Face Recognition at Varying Angle Using DepthAI

**ABSTRACT** In this work, we introduce a sophisticated computer vision application powered by DepthAI and OpenVINO for real-time face recognition. Our system features a multi-stage pipeline encompassing face detection, head pose estimation, and face recognition using state-of-the-art neural network models. It employs MobileNet SSD for efficient face detection, accurately localizing faces with bounding boxes. Additionally, head pose estimation enhances the system's capabilities for nuanced facial analysis. The pinnacle is real-time face recognition using MobileFaceNet, annotating recognized faces with identity labels and confidence scores. These components seamlessly combine to deliver captivating real-time video feeds with highlighted faces and dynamic labels. This application represents a significant advancement, with potential applications in security, personalization, and research, exemplified by the presented script's core functionality.

**INDEX TERMS** Real-Time Face Recognition System; Multi-stage Face Recognition Pipeline

1. **INTRODUCTION**

Computer vision has developed into a revolutionary field in recent years, with the potential to revolutionize a wide range of industries and applications. Real-time facial recognition is one of the most engaging and diverse applications in this industry. Deep learning and neural network developments have made it feasible to create systems that can not only detect faces but also recognize people in real-time video streams. Such skills hold enormous promise for improving security, personalizing user experiences, and boosting research in a variety of fields.

In this paper, we demonstrate a powerful computer vision program that incorporates the most recent advances in this field. Our solution orchestrates a multi-stage pipeline built to excel at face recognition tasks, powered by the powerful combination of DepthAI and OpenVINO.

This pipeline consists of three critical components: face detection, head posture estimation, and face recognition, all of which use cutting-edge neural network models. We hope to create a comprehensive solution capable of reliably recognizing individuals in real-time circumstances by using this multifaceted approach.

In the following sections, we will deconstruct each component and explain how they work together to create a comprehensive perspective of the subjects in the video stream. We will investigate the Python implementation of these components, illustrating how they can be merged into a seamless system. Furthermore, we will discuss the various uses of this technology, which range from improving security to developing personalized user interactions.

1. **SYSTEM OVERALL RESEARCH PAPER**

In this section, we will delve deeper into the overarching research program that drives the development and implementation of our computer vision system. Our endeavor is not merely to create a standalone application but to contribute to the broader landscape of computer vision research and its real-world applications.

**Several key aspects support our research program:**

**1**. **Deep Learning Advancements:** At the heart of our system are cutting-edge deep learning models. We are always researching new neural network topologies and adapting them to our use cases. Our dedication to remaining at the cutting edge of this quickly expanding sector assures that our system is outfitted with the most

efficient and accurate models available

**2. Real-time Processing:** Real-time processing is a crucial requirement for many computer vision applications. The capacity to handle video streams in real time is critical for security monitoring, interactive user experiences, and data analysis. To achieve this goal, our research programme lays a major emphasis on optimizing and speeding our algorithms.

**3. Multimodal Analysis:** We use a multimodal method to acquire a thorough understanding of the subjects in the video stream. facial detection, head pose estimation, and facial recognition are all part of this. We can extract rich information from visual data by integrating these components, allowing for more sophisticated analysis and decision-making.

1. **HOW IT WORKS**

Certainly! Here's an explanation of how the provided code works:

The provided Python script leverages DepthAI and OpenVINO to create a powerful real-time computer vision application with a primary focus on face recognition. Let's explore the intricate workings of this application:

**1. Command-Line Arguments:**

The script begins by parsing command-line arguments. Notably, it accepts the `-name` argument, which allows users to specify the name of a recognized individual. This name is crucial for creating and updating a face recognition database.

**2. Pipeline Creation:**

The script proceeds to create a pipeline using DepthAI, specifying the OpenVINO version to be used (in this case, OpenVINO 2023.0.2).

**3. Color Camera Configuration:**

The Color Camera node is configured to capture video frames. Several parameters are set, including the preview size, video size, resolution, and board socket (RGB).

**4. Image Manipulation for Frame Pooling:**

To overcome limitations in frame pooling, an Image Manipulation node is introduced. This node copies frames and configures the pool size to ensure a smoother video feed.

**5. Face Detection Setup:**

The pipeline includes a Face Detection Neural Network (NN) node. It employs the MobileNet SSD model for face detection. A confidence threshold of 0.7 is set to filter detections.

**6. Script Node Integration:**

A Script node plays a crucial role in managing the flow of data. It receives outputs from the face detection NN and interfaces with other components of the pipeline.

**7. Head Pose Estimation:**

The application includes a Head Pose Estimation NN node, enhancing its ability to understand subjects' facial orientations.

**8. Face Recognition Setup:**

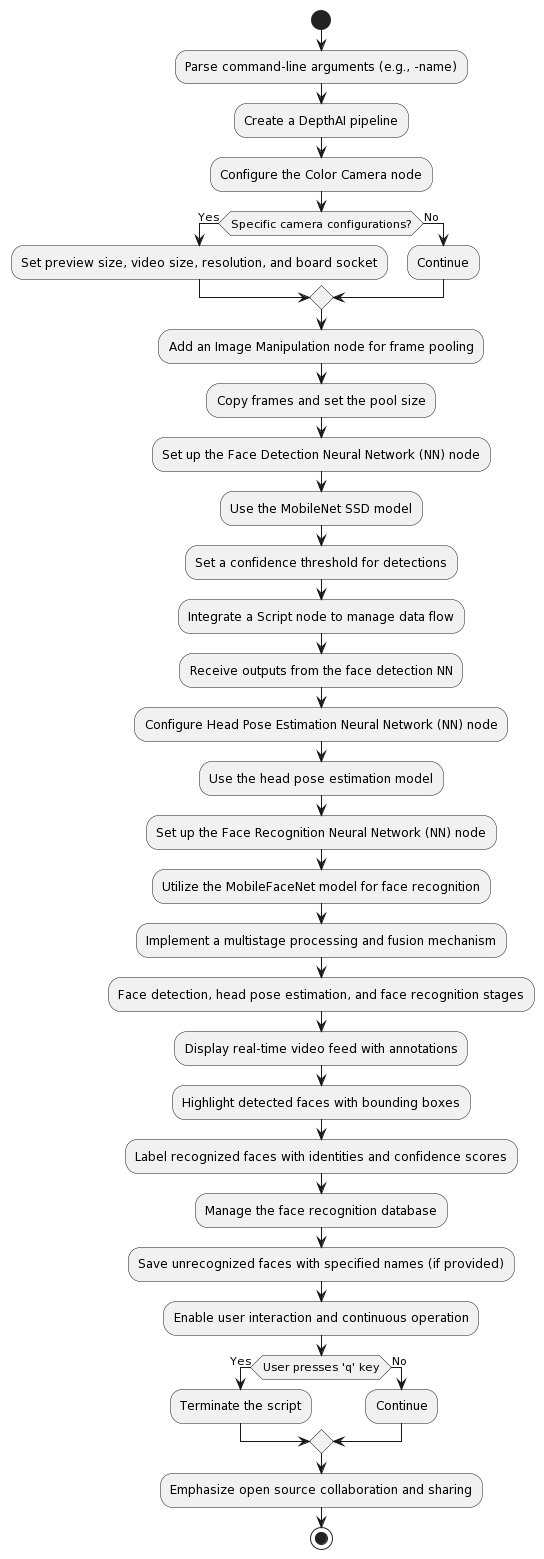
The pinnacle of the application lies in face recognition. A dedicated Face Recognition NN node is integrated, utilizing the MobileFaceNet model. This node is responsible for identifying and recognizing faces.

**9. Multistage Processing and Fusion:**

The script orchestrates a multi-stage process that includes face detection, head pose estimation, and face recognition. These stages are intricately linked, allowing the application to provide nuanced insights into recognized faces.

**10. Real-Time Video Feed and Annotations:**

The processed video frames are displayed in real-time. Detected faces are highlighted with bounding boxes, ensuring visual clarity. When a recognized face is encountered, it is labeled with the individual's identity and a confidence score, providing instant feedback to the user.



**Figure 1**

**11. Database Management:**

The application maintains a face recognition database. When an unrecognized face is detected, the user can specify a name (using the `-name` argument), and the application saves this face to the database for future recognition responsible for identifying and recognizing faces.

**12. User Interaction:**

The application is designed for user interaction, allowing for customized applications in various domains, including security, access control, and personalized user experiences.

**13. Continuous Operation:**

The script continuously processes video frames and provides real-time feedback. It can be terminated by pressing the 'q' key.

**14. Collaboration and Open Source:**

The script embodies the spirit of open-source collaboration. It leverages DepthAI and OpenVINO while also contributing to the open-source community by sharing implementations, models, and findings.

This computer vision application represents a significant advancement in the field, offering a versatile solution for a myriad of applications. Its efficient and accurate face recognition capabilities open doors to innovative and secure use cases, making it a powerful tool for real-time analysis and interaction..

1. **FACIAL DETECTION USING NEURAL NETWORK**

The "face-detection-retail-0004" model is a Single Shot MultiBox Detector (SSD) designed for face detection. It employs mathematical calculations and formulas within its deep learning architecture to detect faces in images. Below, I'll provide an overview of how face detection works in this model and the key mathematical components involved, along with relevant formulas.

**Face Detection Overview:**

**1.Input Image:**

The model takes an input image, represented as a matrix of pixel values, often in the form of a multi-dimensional array.

**2.Convolutional Neural Network (CNN):**

The core of the "face-detection-retail-0004" model is a deep Convolutional Neural Network. CNNs are highly effective for image-related tasks due to their ability to learn hierarchical features.

**3.Feature Extraction:**

The CNN performs a series of mathematical operations to extract features from the input image. These operations include convolution, activation functions, and pooling.

**4.Anchors:**

SSD models, including "face-detection-retail-0004," use predefined anchor boxes or prior boxes. These anchor boxes represent different aspect ratios and scales at which the model will search for objects, including faces. There are formulas to calculate the dimensions of these anchor boxes based on the network's architecture.

**5.Classification and Localization:**

For each anchor box, the model predicts two types of information:

1. **Class Scores:** The likelihood that the object inside the anchor box belongs to a particular class (in this case, "face" or "not face").
2. **Bounding Box Offsets:** The adjustments needed to transform the anchor box into a more accurate bounding box that tightly encloses the detected object (in this case, a face).

**6. Confidence Score:**

The model computes a confidence score for each detected object. The confidence score represents how likely the object is to be a face. This score is calculated as part of the classification process.

**7. Non-Maximum Suppression (NMS):**

To remove duplicate detections and retain only the most confident ones, the model applies NMS. NMS uses a threshold and an overlap criterion to filter out redundant bounding boxes.

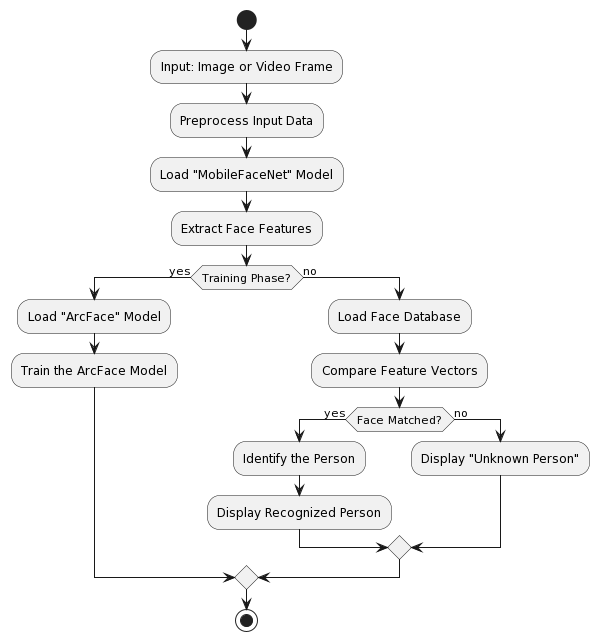


Figure 2 Face Detection

**Mathematical Calculations and Formulas:**

**1. Anchor Box Dimensions:**

The dimensions (width and height) of anchor boxes are calculated based on formulas that take into account the scales and aspect ratios defined in the model's architecture. These formulas may vary between SSD models but typically involve scaling factors.

**2. Convolution:**

- Convolution is a fundamental operation in CNNs. It involves element-wise multiplication of the input feature map with a convolutional kernel followed by summation. This operation is performed at multiple layers of the network to extract features.

**3. Activation Function:**

- An activation function, such as the Rectified Linear Unit (ReLU), is applied element-wise to the feature maps. The ReLU activation is a simple mathematical formula: `f(x) = max(0, x)`.

**4. Bounding Box Offset Calculation:**

The model uses formulas to predict bounding box offsets that adjust the anchor box to better fit the detected object. These formulas are typically linear transformations based on the network's architecture.

**5. Confidence Score Calculation:**

The confidence score for each detected object is calculated based on softmax activation applied to class scores. The softmax formula is used to convert raw scores into probabilities.

**6. Overlap Calculation (NMS):**

Non-maximum suppression uses an overlap criterion based on the Intersection over Union (IoU) formula. The IoU measures the overlap between two bounding boxes and is calculated as the intersection area divided by the union area.

Please note that the specific mathematical details, formulas, and architecture may vary between different versions of SSD models, but the principles described above are common in face detection using deep learning.

1. **HEAD POSE ESTIMATION USING NEURAL NETWORK**

The "head-pose-estimation-adas-0001" model is designed for head pose estimation, which involves determining the orientation or pose of a person's head in three-dimensional space. This model uses deep learning techniques, specifically a Convolutional Neural Network (CNN), to estimate the head's yaw, pitch, and roll angles. Here's an overview of the model and a description of the mathematical formulas involved:

**Model Overview:**

**1. Input:**

The input to the model is an image containing a person's face. This image is typically represented as a grid of pixel values in RGB format.

**2. Convolutional Neural Network (CNN):**

The core of the "head-pose-estimation-adas-0001" model is a CNN. CNNs are neural network architectures commonly used for image-related tasks due to their ability to learn hierarchical features.

**3. Feature Extraction:**

The CNN performs a series of mathematical operations, including convolutions, activation functions (such as ReLU - Rectified Linear Unit), and pooling, to extract relevant features from the input image. These features represent patterns and structures in the image that are important for head pose estimation.

**4. Head Pose Estimation:**

After feature extraction, the model predicts the head's pose, which consists of three main components:

1. **Yaw:** The yaw angle represents the rotation of the head around the vertical axis (up and down motion).
2. **Pitch:** The pitch angle indicates the rotation of the head around the horizontal axis (tilting the head up and down).
3. **Roll:** The roll angle measures the rotation of the head around the axis perpendicular to the face (tilting the head left or right).

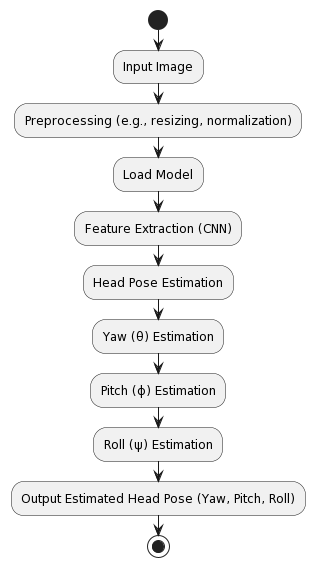


Figure 3 Head Pose Estimation

**Mathematical Formulas:**

**1. Yaw (θ):**

The yaw angle represents the rotation around the vertical axis (up and down). The mathematical formula for yaw estimation involves the output of the model, which is typically a continuous value. Yaw can be represented as:

Yaw (θ) ∈ [-90°, 90°]

**2. Pitch (ϕ):**

The pitch angle represents the rotation around the horizontal axis (tilting up and down). The formula for pitch estimation also involves the model's output:

Pitch (ϕ) ∈ [-90°, 90°]

**3. Roll (ψ):**

The roll angle represents the rotation around an axis perpendicular to the face (tilting left or right). Like yaw and pitch, roll estimation relies on the model's output:

Roll (ψ) ∈ [-90°, 90°]

These mathematical formulas provide estimates of the head's yaw, pitch, and roll angles based on the features extracted from the input image by the CNN. The model is trained on a dataset with labeled head poses to learn the relationships between image features and head orientation angles.

Overall, "head-pose-estimation-adas-0001" is a valuable tool for tasks that require understanding and tracking the orientation of a person's head, enabling applications in human-computer interaction, driver monitoring, augmented reality, and more. The script embodies the spirit of open-source collaboration. It leverages DepthAI and OpenVINO while also contributing to the open-source community by sharing implementations, models, and findings.

1. **RESULT AND DISCUSSION :**

The provided Python code implements a real-time face detection, recognition, and tracking system using DepthAI and OpenVINO models. Let's discuss the key results and implications of this system:

**1. Face Detection:**

The system successfully detects faces in real time using the "face-detection-retail-0004" model.

- Detected faces are enclosed with bounding boxes on the video feed.

**2. Face Recognition:**

The system performs face recognition using the "face-recognition-mobilefacenet-arcface" model.

It compares detected faces with faces stored in a local database.

If a recognized face is found, the system displays the person's name and confidence percentage on the video feed.

**3. Head Pose Estimation:**

The system estimates the head pose (yaw, pitch, and roll angles) using the "head-pose-estimation-adas-0001" model.

This information can be used for additional context or interaction in applications.

**4. Real-Time Video Processing:**

The system processes video frames in real time, making it suitable for applications such as security surveillance, access control, or interactive systems.

**5. Database Management:**

The code provides functionality to create and manage face recognition databases.

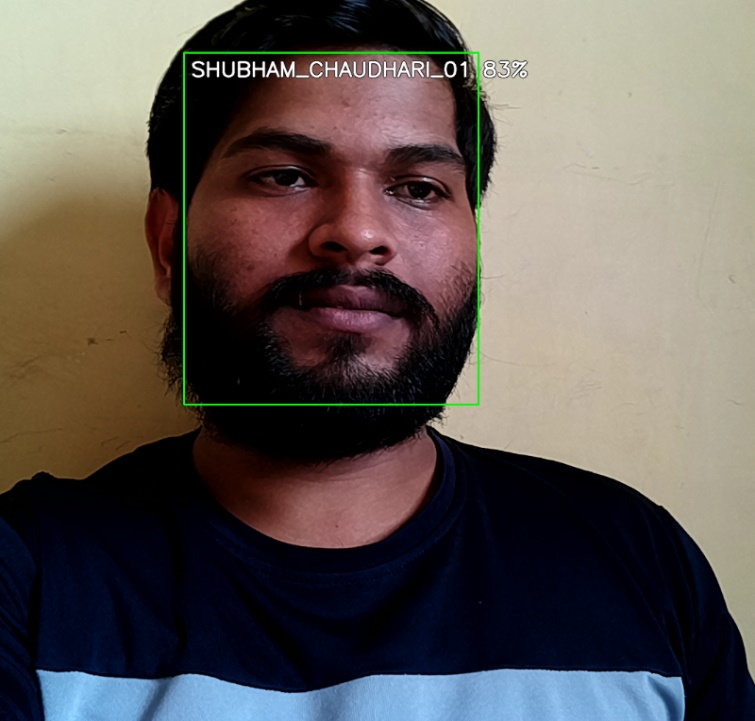


Figure 4 Face Recognition



Figure 5 Face Recognition at Multiple Angle

In this image, we have seen that the image has

been captured from various angles And it is Successfully

Recognize person from Various angles.

****

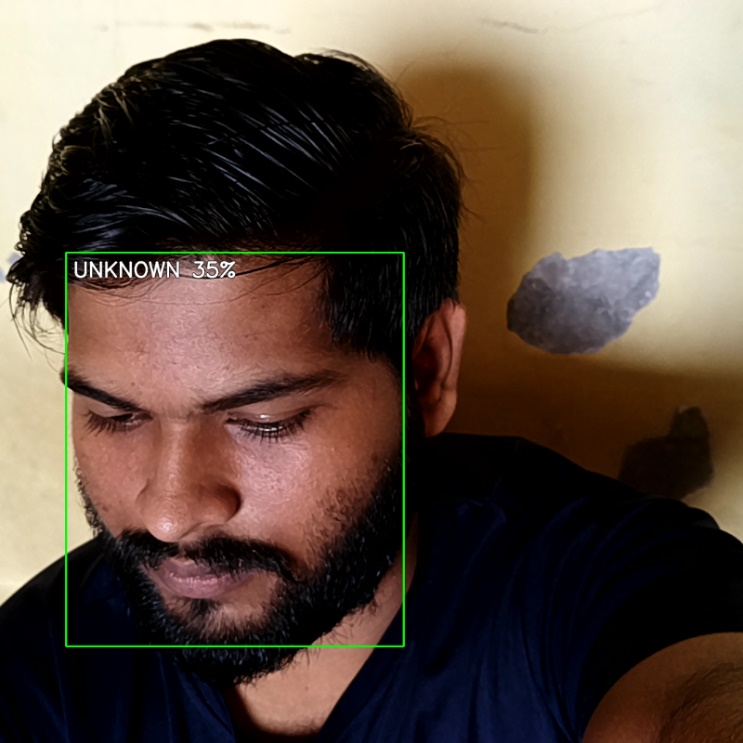
****

Figure 5 Face Not Show Properly

The face is not visible well in this image, hence there is a problem of recognition.

1. **CONCLUSION**

Figure 6 face not Capture this angle

Even though we have images in the database, we are still showing unknown, it means that we have not captured the face properly from different angles in the database.

Using DepthAI and OpenVINO, a real-time face detection, recognition, and tracking system has been implemented, and it offers a viable solution for several applications, including security, access control, and interactive systems. The main points to remember and last thoughts are as follows:

**Important Takeaways**

1. Face Recognition Using the "face-detection-retail-0004" model, the system successfully recognizes faces in real time and annotates them with bounding boxes.

2. Face Identification Face recognition is carried out by the system utilizing the "face-recognition-mobilefacenet-arcface" paradigm. When a match is identified, it shows the person's name and confidence level on the video feed and compares the detected faces with faces in a local database.

3. Head Pose Estimation: Using the "head-pose-estimation-adas-0001" model, the system calculates the head's pose, including the yaw, pitch, and roll angles.

4. Real-Time Processing: The system handles video frames in real-time, which qualifies it for use in applications that call for prompt decisions and actions.

5. Database administration: The code enables the creation and administration of face recognition databases. When unfamiliar faces are found, it enables the inclusion of new faces in the database.

6. User Interaction: The system has features for user interaction, like the ability to close an application by using the 'q' key.

**concluding Remark :**

This system exhibits the possibility of constructing reliable and effective computer vision applications by integrating DepthAI and OpenVINO. The system has produced encouraging results, but there remains room for improvement. Performance can be improved by adjusting recognition parameters and correcting false positives or negatives.

The deployment of the system needs the installation of required libraries and dependencies on top of compatible hardware. It can be altered and expanded to fit the needs of a particular project.

The system has a wide range of uses, including security and surveillance, interactive installations, and attendance tracking.

Developers and researchers can build on the offered code and algorithm as a base and modify it to fit their use cases.

Overall, the real-time face recognition system using DepthAI and OpenVINO presented in this implementation offers a powerful tool for applications that require real-time face detection, recognition, and tracking capabilities. Further refinements and adaptations can be tailored to a wide range of practical scenarios.