Data Processing:

Here are the steps involved for data processing

1.Reading Video Frames:

OpenCV's VideoCapture class is used to read the video frames sequentially.

This class provides a convenient interface to capture video frames from various sources such as video files, cameras, or network streams.

The video file path is provided as input to VideoCapture, which then opens the video file and initializes the capturing process.

2.Resizing Video Frames:

Each video frame is resized while maintaining the aspect ratio.

Aspect ratio refers to the ratio of the width to the height of an image or frame.

When resizing the frames, it's essential to preserve this aspect ratio to prevent distortion and ensure consistency in processing.

The resizing process involves calculating new dimensions based on a target size while keeping the aspect ratio constant.

If the aspect ratio of the original frame differs from the target aspect ratio, padding may be applied to maintain the original aspect ratio.

The resize_with_aspect_ratio function in the code calculates the new dimensions for resizing while preserving the aspect ratio.

Resizing Gesture Image:

Similar to video frames, the input image for gesture representation is also resized.

This resizing ensures that the dimensions of the gesture image match those of the video frames, facilitating comparison and analysis.

The resize_with_aspect_ratio function is applied to the input image, ensuring consistency in size with the video frames.

By processing the video frames and the input gesture image in this manner, we ensure that both inputs are appropriately resized and ready for further analysis and comparison. This consistent preprocessing step is crucial for accurate gesture detection and reliable results.

Model Selection:

In the model selection/development phase, we leverage the MediaPipe Hands model provided by the mediapipe library for hand landmark detection. This model is a pre-trained neural network specifically designed for detecting and localizing hand landmarks in images or video frames. By utilizing this existing model, we benefit from its accuracy and efficiency in detecting hand landmarks, without the need to develop a new model from scratch. The mediapipe library provides a convenient and reliable implementation of this model, making it straightforward to integrate into our gesture detection system. By relying on a pre-trained model, we can achieve robust hand landmark detection capabilities without the complexities of training a new model, thereby streamlining the development process and ensuring high-quality results.

Detection Algorithm:

In the detection algorithm, the following steps are performed:

- Hand landmarks are extracted from both the video frames and the input image using the MediaPipe Hands model.

- Euclidean distance is calculated between the corresponding landmarks in the video frames and the image. This distance represents the spatial difference between the hand landmarks in the video frames and the reference image.

- A similarity score is computed based on the mean of the distances normalized by the maximum possible distance. Normalizing the distances allows for a consistent comparison across different hand configurations and scales.

- If the similarity score exceeds a predefined threshold, a gesture match is determined. This threshold serves as a criterion to determine whether the hand gesture in the video frame closely resembles the gesture represented by the reference image. If the similarity score is above the threshold, it indicates a match, and the gesture is considered detected.

By following these steps, the algorithm effectively compares the hand landmarks between the video frames and the reference image, enabling accurate detection of gestures in the video stream. Adjusting the threshold allows for flexibility in controlling the sensitivity of the gesture detection system to match specific application requirements or environmental conditions.

Annotion:

In the annotation step:

- An annotation method is implemented to overlay text on the video frames where a gesture is detected.

- The text "DETECTED" is displayed at the top right corner of the frame using the cv2.putText() function from the OpenCV library.

- By providing a visual indication on the video frames where a gesture is detected, users can easily identify the instances where the specified gesture occurs. This annotation adds interpretability to the output of the gesture detection system, making it more intuitive and user-friendly.

Assumptions Made:

The assumption is made that the hand gestures in the video frames closely resemble those in the input gesture image. Therefore, similarity in hand landmarks indicates a match.

Challenges Faced and Addressed:

1.Ensuring consistent resizing: Resizing frames and images while maintaining aspect ratio was critical to ensure accurate comparison. This was addressed by implementing a function to resize with aspect ratio.

Threshold determination: Determining an appropriate threshold for gesture detection required experimentation and tuning to balance between false positives and false negatives.

2.Resource optimization: Efficient resource utilization, especially memory and processing time, was a challenge. The use of existing models and optimizing the code helped address this challenge.