```
import cv2
import mediapipe as mp


class SignLanguageInterpreter:
    def __init__(self):
        self.signs = {
            'hi': 'Hi',
            'ok': 'OK',
            'vanakkam': 'Vanakkam',
            'salute': 'Salutation',
        }

    def interpret_sign(self, sign):
        return self.signs.get(sign, 'Unknown sign')


def detect_gesture(hand_landmarks):
    if hand_landmarks is not None:
        # Check if all five fingers are open
        all_fingers_open = all(
            landmark.y < hand_landmarks.landmark[mp.solutions.hands.HandLandmark.INDEX_FINGER_MCP].y
            for landmark in hand_landmarks.landmark[mp.solutions.hands.HandLandmark.INDEX_FINGER_TIP.value:mp.solutions.hands.HandLandmark.PINKY_TIP.value + 1]
        )

        # Check for thumbs-up gesture (thumb tip lower than thumb base)
        is_thumbs_up = hand_landmarks.landmark[mp.solutions.hands.HandLandmark.THUMB_TIP].y < hand_landmarks.landmark[mp.solutions.hands.HandLandmark.THUMB_IP].y

        # Check if the wrist is positioned on or above the nose (representing a salute)
        is_salute = hand_landmarks.landmark[mp.solutions.hands.HandLandmark.WRIST].y <= hand_landmarks.landmark[mp.solutions.holistic.PoseLandmark.NOSE].y
```

```
        if all_fingers_open:
            return 'hi'
        elif is_thumbs_up:
            return 'ok'
        elif is_salute:
            return 'salute'

    return None

def main():
    interpreter = SignLanguageInterpreter()

    # Initialize Mediapipe Hands module
    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands()

    # Open the webcam
    cap = cv2.VideoCapture(0)

    while True:
        # Capture frame-by-frame
        ret, frame = cap.read()

        # Convert the BGR image to RGB
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Process the frame with Mediapipe Hands
        results = hands.process(rgb_frame)

        # Get hand landmarks if detected
```

```
        hand_landmarks = results.multi_hand_landmarks[0] if results.multi_hand_landmarks else None

        # Display the resulting frame
        cv2.imshow('Webcam Feed', frame)

        # Perform gesture detection
        detected_gesture = detect_gesture(hand_landmarks)

        if detected_gesture:
            meaning = interpreter.interpret_sign(detected_gesture)
            print(f"Detected Gesture: {detected_gesture}, Meaning: {meaning}")

        # Break the loop if 'q' key is pressed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Release the capture
    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```