Instituto Tecnológico y de Estudios Superiores de Monterrey

Escuela de Ingeniería y Ciencias

Ingeniería en Ciencia de Datos y Matemáticas

Inteligencia artificial avanzada para la ciencia de datos II

Morales Ramón Michelle Yareni A01552627

Monterrey, Nuevo León. Fecha, 5 de noviembre de 2023

Keywords— Machine Learning, interfaz de usuario, desarrollo de aplicaciones, aplicación web, aplicación móvil

1. Introducción

La detección de poses se ha convertido en una tarea esencial en muchas aplicaciones de visión por computadora y análisis de imágenes, desde sistemas de seguimiento de gestos hasta aplicaciones de vigilancia y atención médica. Para este reporte escogí el proyecto que se está desarrollando para el reto y hablaré especificamente del modelo para detectar poses.

2. Detección de poses

Para la detección de poses encontré que mediapipe, la librería de Python, era util. Esta se trata de una extensión de la plataforma MediaPipe que permite a los desarrolladores de Python utilizar las capacidades de visión por computadora.

Hice algunas pruebas con esta librería pero como no era lo que quería, invetigué sobre Tensorflow, que es una biblioteca de código abierto utilizada para aprendizaje automático y para inteligencia artificial. Fue diseñada para facilitar la creación, el entrenamiento y la implementación de modelos que usan redes neuronales y otros algoritmos de procesamiento de datos.

La ventaja de usar TensorFlow es que cuenta con una gran cantidad de modelos preentrenados y es muy eficiente, flexible. Esto lo convierte en una herramienta muy util para el desarrollo de aplicaciones que requieran detección de poses precisas, como es el caso del reto.

Primero cargué el modelo preentrenado desde TensorFlow Hub y lo preparé para su uso en detección de poses. En este caso se usará para identificar las poses del video al habilitar la cámara.

Hice pruebas con dos modelos preentrenados que fueron "PoseNet" y "MoveNet". Ambos detectan poses pero tienen arquitecturas diferentes así como aplicaciones y rendimientos distintos. "PoseNet" usa una red neuronal convolucional profunda para estimar poses humanas en imágenes y se utiliza para la estimación de poses de una sola persona. Como se necesitaba que reconociera poses de multiples personas y además en tiempo real, se descartó este modelo y le eligió "MoveNet" el cual utiliza una arquitectura más ligera y eficiente que se ha diseñado específicamente para la detección de poses en tiempo real para multiples personas.

```
model = hub.load('https://tfhub.dev/google/movenet/multipose/lightning/1')
movenet = model.signatures['serving_default']
```

Figura 1: Carga del modelo

Después definí la función para dibujar líneas entre los keypoints detectados en cada frame de imagen. Para ello se establece un umbral de confianza y se dibujan líneas si la confianza de ambos puntos clave supera el umbral. Esto es útil para visualizar conexiones entre partes del cuerpo.

```
def draw_connections(frame, keypoints, edges, confidence_threshold):
    y, x, c = frame.shape
    shaped = np.squeeze(np.multiply(keypoints, [y,x,1]))

    for edge, color in edges.items():
        p1, p2 = edge
        y1, x1, c1 = shaped[p1]
        y2, x2, c2 = shaped[p2]

        if (c1 > confidence_threshold) & (c2 > confidence_threshold):
            cv2.line(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0,0,255), 4)
```

Figura 2: Dibujo de líneas

Posteriormente definí la función que dibuja los keypoints. De igual manera se establece un umbral de confianza y se dibujan círculos en la imagen en las ubicaciones de los puntos clave si la confianza supera el umbral.

```
def draw_keypoints(frame, keypoints, confidence_threshold):
    y, x, c = frame.shape
    shaped = np.squeeze(np.multiply(keypoints, [y,x,1]))

    for kp in shaped:
        ky, kx, kp_conf = kp
        if kp_conf > confidence_threshold:
            cv2.circle(frame, (int(kx), int(ky)), 6, (0,255,0), -1)
```

Figura 3: Dibujo de puntos

Después, la función loop_through_people se encarga de iterar a través de una lista de personas, donde cada persona tiene puntos clave detectados con sus respectivas confianzas. Luego, utiliza las funciones draw_connections y draw_keypoints para visualizar las conexiones y los puntos clave de cada persona en el frame de la imagen, lo que es útil para la detección de poses.

```
def loop_through_people(frame, keypoints_with_scores, edges, confidence_threshold):
    for person in keypoints_with_scores:
        draw_connections(frame, person, edges, confidence_threshold)
        draw_keypoints(frame, person, confidence_threshold)
```

Figura 4: Dibujo de lineas y puntos en cada frame

Además hice un diccionario con el propósito de especificar qué color se debe utilizar al dibujar las conexiones entre puntos clave en la imagen.

Finalmente se hace la captura de video desde la cámara en tiempo real, se redimensionan las imágenes y se usan las funciones definidas previamente para detectar poses en cada frame y muestrar el resultado en una ventana. El bucle continua hasta que se presione la tecla 'q'.

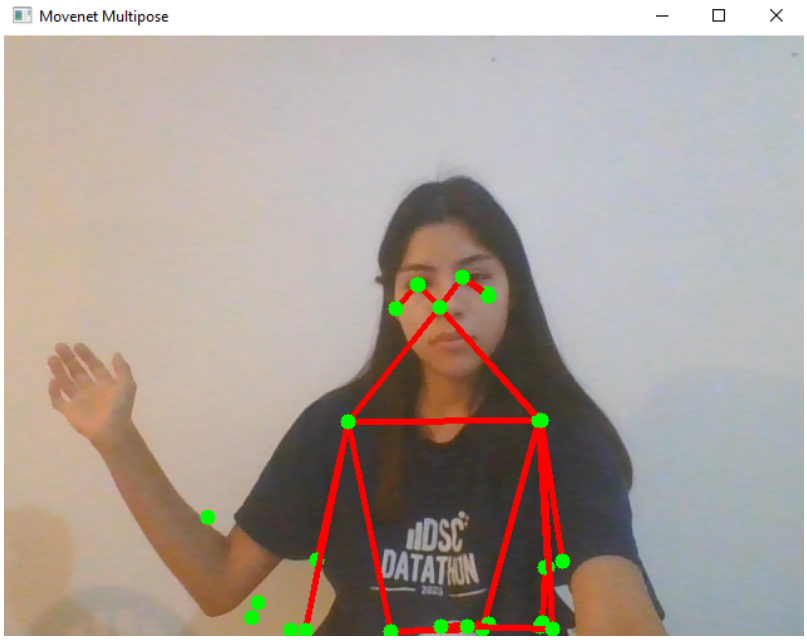

Figura 5: Modelo en funcionamiento

3. Conclusiones

El modelo funcionó tal como esparaba, sin embargo, hay que hacer mejoras ya que las poses no se detectan tan bien aunque en parte puede ser por la mala calidad del video, ya que en clase se probó en con otra cámara y los resultados fueron mejores.