SIGNTEACH

Antonio Aparicio González

Emilio Díaz Martel

Introducción

SignTeach es una innovadora aplicación móvil diseñada para facilitar el aprendizaje del lenguaje de signos de una manera interactiva y divertida. Esta aplicación nativa para dispositivos Android ofrece a los usuarios una experiencia educativa única que combina la tecnología de reconocimiento de signos con las manos, basada en modelos preentrenados de OpenCV y YOLOV8, con un juego interactivo y envolvente.

Motivación y Objetivos del Proyecto

La motivación detrás de SignTeach es abordar las necesidades y desafíos que enfrentan tanto las personas sordas como aquellas interesadas en aprender el lenguaje de signos. Algunas de las **principales motivaciones** incluyen:

- **Promover la Inclusión**: Creemos en la importancia de la inclusión y la igualdad de oportunidades para todas las personas, independientemente de su capacidad auditiva. El lenguaje de signos es una herramienta esencial para lograr esta inclusión.
- **Superar Barreras de Comunicación**: Las barreras de comunicación pueden ser una fuente de aislamiento para las personas sordas. SignTeach busca eliminar estas barreras y permitir una comunicación efectiva entre personas sordas y oyentes.
- **Falta de Recursos Accesibles**: A menudo, la falta de recursos accesibles y efectivos para aprender el lenguaje de signos es un obstáculo para aquellos que desean aprender. Además, no existen actualmente aplicaciones populares que incorporen este tipo de tecnologías aplicadas a la enseñanza de este lenguaje. SignTeach aborda esta carencia al proporcionar una plataforma accesible y efectiva.

Objetivos del Proyecto:

1. **Facilitar el Aprendizaje del Lenguaje de Signos**: El objetivo principal de SignTeach es proporcionar a los usuarios una herramienta efectiva y accesible para aprender el lenguaje de signos de manera eficiente y divertida.
2. **Ofrecer Retroalimentación en Tiempo Real:** SignTeach utiliza tecnología de reconocimiento de signos con las manos para proporcionar retroalimentación instantánea a los usuarios, permitiéndoles corregir y mejorar sus habilidades de manera inmediata.
3. **Hacer el Aprendizaje Divertido y Motivador**: Queremos convertir el aprendizaje del lenguaje de signos en una experiencia divertida y motivadora a través de un juego interactivo y envolvente.
4. **Desarrollar una Comunidad de Aprendizaje:** En versiones más avanzadas, se pretende que SignTeach incluya características sociales que permitan a los usuarios conectarse, compartir su progreso y aprender juntos, creando así una comunidad de aprendizaje sólida.

Descripción técnica

En este apartado, profundizaremos en la descripción técnica de la aplicación SignTeach, abarcando tanto su arquitectura general como los componentes clave que la componen. SignTeach se basa en una combinación de tecnologías que permiten la comunicación fluida entre el servidor y la aplicación nativa de Android. A continuación, se presentan detalles sobre su arquitectura e implementación.

Arquitectura general

La aplicación se basa en una arquitectura sólida que utiliza una API REST para comunicarse con un servidor backend desarrollado en Python con el framework Flask. Este servidor es el corazón de la aplicación, ya que procesa imágenes y videos para reconocer los signos realizados con las manos y proporciona retroalimentación en tiempo real a los usuarios.

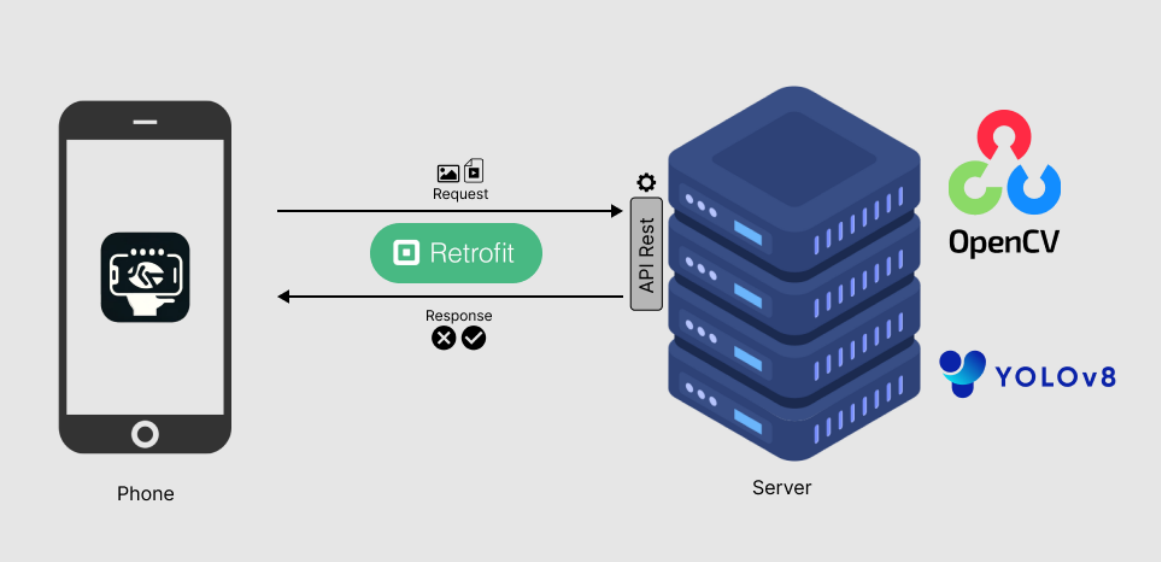

Imagen 1: Esquema de la arquitectura del proyecto

Backend Servidor Flask en Python

La aplicación backend para SignTeach procesa las imágenes y vídeos recibidos por el cliente (usuario) y trata de reconocer los signos realizados con las manos usando OpenCV y YOLOv8. Una vez procesados los datos, informa al cliente del resultado de la detección.

Estructura del Código

El código se divide en varias partes, cada una manejando una funcionalidad específica:

1. **Rutas del Servidor:**
    - Ruta raíz ('/') para mostrar una página HTML.
    - Ruta '/test-server' para procesar texto.
    - Ruta '/process-image' para procesar imágenes.
    - Ruta '/process-video' para procesar vídeos.
2. **Procesamiento de Datos:**
    - Procesamiento de texto, imágenes y vídeos.
    - Uso de OpenCV para manipular imágenes y vídeos.
    - Lógica para determinar si una imagen o un vídeo contiene la letra de signo solicitada.
3. **Ejecución del Servidor:**
    - Se realiza la configuración del host y el puerto, incluyendo una opción para HTTPS.

Detalles de las Rutas

Se implementa una API REST y se dispone de las siguientes rutas:

1. **Ruta Raíz ('/')**
    - Método: GET
    - Funcionalidad: Muestra la página principal del servidor.
2. **Ruta '/test-server'**
    - Método: POST
    - Funcionalidad: Recibe una cadena de texto en formato JSON, la procesa (ejemplo: convierte a mayúsculas) y devuelve el resultado.
3. **Ruta '/process-image'**
    - Método: POST

- Funcionalidad: Recibe una imagen, la procesa con OpenCV para detectar el signo y devuelve si el resultado coincide con la letra de signo solicitada.

4. **Ruta '/process-video'**
    - Método: POST
    - Funcionalidad: Recibe un archivo de vídeo, lo guarda temporalmente, procesa cada frame para detectar el signo y devuelve si el resultado coincide con la letra de signo solicitada

Procesamiento de Imágenes y Vídeos:

Se utiliza **secure_filename** para garantizar la seguridad al guardar archivos, los cuales se almacenan de forma temporal para su procesamiento. Se dispone de una opción para eliminar archivos temporales después del procesamiento. Se incluyen respuestas con códigos de estado HTTP y mensajes de error en caso de datos inválidos o ausentes para informar al cliente

Se realiza un preprocesamiento básico como rotar imágenes, y se utiliza OpenCV para decodificar y manipular las imágenes y vídeos.Se llama a funciones específicas **isLetterInImage** y **isLetterInVideo** cuya implementación utiliza la biblioteca YOLO (You Only Look Once) a través de **ultralytics** para el reconocimiento de signos de lenguaje de signos en imágenes y vídeos. A continuación, se detalla el funcionamiento de cada función:

Función isLetterInImage

Propósito: Detectar si una letra específica del lenguaje de signos está presente en una imagen.

Parámetros:

- **frame**: Imagen en la que se realizará la detección.
- **target_letter**: Letra del lenguaje de signos que se busca detectar.
- **confidence_threshold**: Umbral de confianza para considerar válida una detección (por defecto 0.3).

Proceso:

1. Se utiliza el modelo YOLO para detectar objetos en la imagen (**frame**).
2. Se itera sobre las detecciones realizadas por el modelo.
3. Para cada detección, se verifica si la confianza es mayor o igual al umbral de confianza (**confidence_threshold**).
4. Se obtiene la clase detectada y se compara con la **target_letter**.
5. Si se encuentra una coincidencia, la función devuelve **True**. De lo contrario, devuelve **False**.

Uso de la Función:

- Se carga una imagen.

- Se especifica la letra objetivo.
- Se llama a **isLetterInImage** con la imagen y la letra objetivo como argumentos.

Función isLetterInVideo

Propósito: Determinar si una letra específica del lenguaje de signos se detecta en un porcentaje significativo de los frames de un vídeo.

Parámetros:

- **video_path**: Ruta del archivo de vídeo a procesar.
- **target_letter**: Letra del lenguaje de signos que se busca detectar en el vídeo.
- **model_path**: Ruta del modelo YOLO preentrenado (por defecto 'sign_language.pt').
- **confidence_threshold**: Umbral de confianza para considerar válida una detección (por defecto 0.3).
- **percentage_threshold**: Porcentaje mínimo de frames en los que debe detectarse la letra para considerar el vídeo como válido (por defecto 60%).

Proceso:

1. Se carga el modelo YOLO especificado en **model_path**.
2. Se abre el vídeo especificado en **video_path**.
3. Se procesa cada frame del vídeo con el modelo YOLO.
4. Se cuenta el número de frames en los que la **target_letter** es detectada con una confianza superior al **confidence_threshold**.
5. Se calcula el porcentaje de frames en los que se detectó la letra.
6. Si este porcentaje es mayor o igual a **percentage_threshold**, la función devuelve **True**. De lo contrario, devuelve **False**.

Características Adicionales:

- La función también dibuja rectángulos y etiquetas en los frames para visualizar las detecciones y se crea un vídeo de salida que muestra estas anotaciones para poder visualizar el resultado del procesamiento del video.

Uso de la Función:

- Se especifica la ruta del vídeo de entrada y la letra objetivo.
- Se llama a **isLetterInVideo** con estos argumentos.
- Se evalúa si la letra se detecta en un porcentaje significativo del vídeo.

Frontend: App Android Nativa con Kotlin

La aplicacion del cliente es una aplicación sencilla que utiliza la cámara del dispositivo para obtener imágenes del signo a reconocer. Tiene 2 modos posibles, foto y vídeo, con el objetivo de reducir posibles latencias que pueda producir la transmisión de video en algunas circunstancias y que se pueda seguir utilizando la aplicación únicamente con fotos, aunque la forma ideal de hacerlo es con vídeo debido a la cantidad de datos que YOLO podrá procesar sobre los cuales podrá detectar y devolver un resultado más preciso.

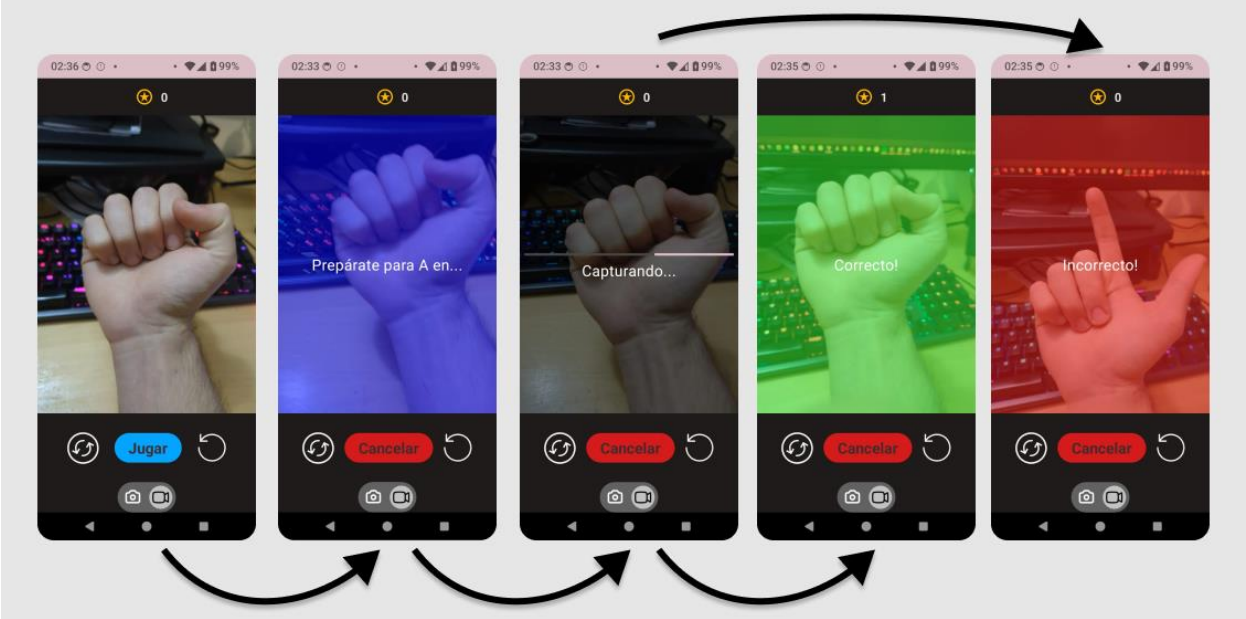

Imagen 2: Esquema de funcioinamiento de la app y transición entre estados. El us**uario pulsa en jugar cuando** este listo, y la **app le propone una letra aleatoria a realizar** (en este caso la A). La aplicación **toma un video de 5 segundos** y lo **envía al servidor para procesarlo.** En función de la respuesta del servidor, se detectará como **correcto o incorrecto** el signo realizado por el usuario. **En caso de acertar, se incrementará en 1 el contador de estrellas de la barra superior.**

A continuación, se documentan las diferentes partes del código de la aplicación:

1. **MainActivity**

Descripción: MainActivity es la actividad principal de la aplicación. Maneja la interfaz de usuario y la interacción inicial con el usuario, incluyendo la solicitud de permisos y la configuración de la vista dependiendo del estado del permiso de cámara.

Funciones Principales:

- **onCreate:** Inicializa la actividad. Configura **ServerInteraction**, solicita permisos de cámara y establece la interfaz de usuario.
- **allPermissionsGranted:** Verifica si todos los permisos requeridos han sido concedidos.
- **setupUI:** Configura la interfaz de usuario. Muestra la pantalla de cámara o una pantalla alternativa si no se otorga el acceso a la cámara.

Componentes Importantes:

- **serverInteraction:** Instancia de **ServerInteraction** para interactuar con el servidor.
- **requestPermissionLauncher:** Manejador para el proceso de solicitud de permisos.

2. **ApiService**

Descripción: ApiService es una interfaz que define los endpoints de la API REST para la comunicación con el servidor backend. Utiliza Retrofit para la gestión de llamadas de red.

Endpoints Definidos:

- **uploadImage:** Envía una imagen al servidor para su procesamiento.
- **uploadVideo:** Envía un video al servidor para su procesamiento.

3. **ResponseModel**

Descripción: ResponseModel es una clase de modelo de datos que representa la respuesta del servidor. Es utilizada por Retrofit para deserializar las respuestas JSON del servidor.

Atributos:

- **result:** Un booleano que indica el resultado del procesamiento del servidor (por ejemplo, si se detectó correctamente la letra objetivo en la imagen o el video).

4. **ServerInteraction**

Descripción: ServerInteraction es la clase responsable de interactuar con el servidor backend. Gestiona el envío de imágenes y vídeos al servidor y el manejo de las respuestas.

Funciones Principales:

- **sendImageResourceToServer:** Envía una imagen almacenada en recursos al servidor.
- **sendImageToServer:** Envía una imagen al servidor para su procesamiento y maneja la respuesta.
- **sendVideoToServer:** Envía un vídeo al servidor para su procesamiento y maneja la respuesta.

Detalles Técnicos:

- Utiliza Retrofit para las llamadas de red.
- Maneja la serialización y deserialización de datos.
- Procesa respuestas asincrónicas del servidor.

Tecnologías empleadas

YOLOv8: Entrenamiento de un modelo

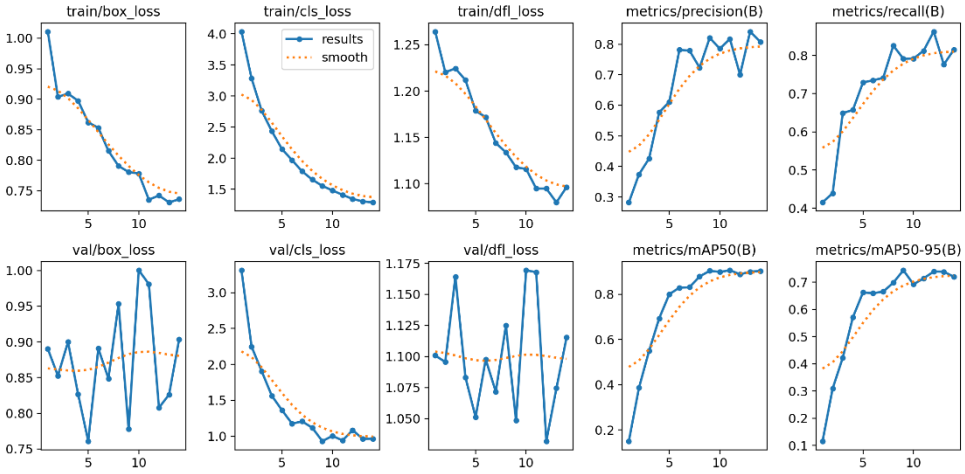

Imagen 3: Resultados del proceso de entrenamiento, como se puede ver son mejorables, posiblemente por la cantidad de datos del dataset, aunque ofrece una precisión razonable

Descripción General:

- **YOLO (You Only Look Once)** es un algoritmo de detección de objetos en tiempo real conocido por su eficiencia y precisión. La versión 8 de YOLO representa una mejora en esta serie de modelos, optimizada para un mejor rendimiento y precisión.

Implementación:

- Se carga el modelo YOLOv8 utilizando el archivo de pesos preentrenados yolov8n.pt. Este archivo contiene la arquitectura del modelo y los pesos aprendidos previamente, lo que facilita un punto de partida avanzado para el entrenamiento.

Configuración de Argumentos de Entrenamiento

Parámetros Específicos:

- **batch:** Tamaño del lote (batch size). Determina cuántas muestras se procesan antes de actualizar los pesos del modelo. En este caso, se establece en 16.
- **epochs:** Número total de épocas. Una época se completa cuando el modelo ha pasado por todas las muestras del conjunto de datos. Aquí se configura para 50 épocas.
- **imgsz:** Tamaño de la imagen para el entrenamiento. Las imágenes se redimensionarán a 416x416 píxeles.
- **patience:** Número de épocas para esperar una mejora en el rendimiento del modelo antes de detener el entrenamiento. Se configura en 5.

- **device:** Dispositivo utilizado para el entrenamiento. 'cuda' indica que se está utilizando una GPU, que es preferible para acelerar el entrenamiento.
- **cache:** Preprocesamiento y almacenamiento en caché de datos. Mejora la eficiencia del entrenamiento al almacenar en caché los datos preprocesados**.**
- **data:** Ruta al archivo YAML que define la configuración del conjunto de datos. Este archivo contiene información como la ruta a los datos, las clases, etc.

Proceso de Entrenamiento

- Se ejecuta el entrenamiento del modelo utilizando los argumentos de configuración definidos.
- results captura el resultado del entrenamiento, incluyendo métricas como la precisión, la pérdida, etc**.**

Otras herramientas que podrían usarse

Existen algunas tecnologías que o bien planteamos y posteriormente descartamos por no cumplir con los suficientes requisitos de calidad o por mantener el proyecto base lo mas simple posible:

1. **TensorFlow Lite y ML Kit para Android:**
    - **Descripción:** TensorFlow Lite es una versión ligera de TensorFlow para dispositivos móviles. ML Kit de Google ofrece una serie de APIs preentrenadas para tareas de Machine Learning en dispositivos móviles.
    - **Uso Potencial:** Se planteó la posibilidad utilizar TensorFlow Lite para ejecutar el modelo de reconocimiento de signos directamente en el dispositivo móvil, lo que podría mejorar la latencia y permitir la funcionalidad offline. No obstante, debido a los habituales problemas de compatibilidad de librerías y sus versiones, además de posibles limitaciones de rendimiento en algunos dispositivos, se opto por realizar el procesamiento en un servidor dedicado.
2. **WebRTC o WebSockets para Transmisión de Video en Tiempo Real:**
    - **Descripción:** WebRTC (Web Real-Time Communication) es una tecnología que permite la transmisión de video y audio en tiempo real a través de la web. **WebSockets**, por otro lado, es un protocolo de comunicación bidireccional sobre una sola conexión TCP, que permite una interacción en tiempo real entre el servidor y el cliente.
    - **Uso Potencial:** Nos hubiera gustado implementar funcionalidades en tiempo real, como detección de signos en tiempo real o interacción en tiempo real, para lo cual esto podrían ser herramientas útiles. En el backend se deja implementada una página web básica con el objetivo de poder añadir un protocolo similar en un futuro.

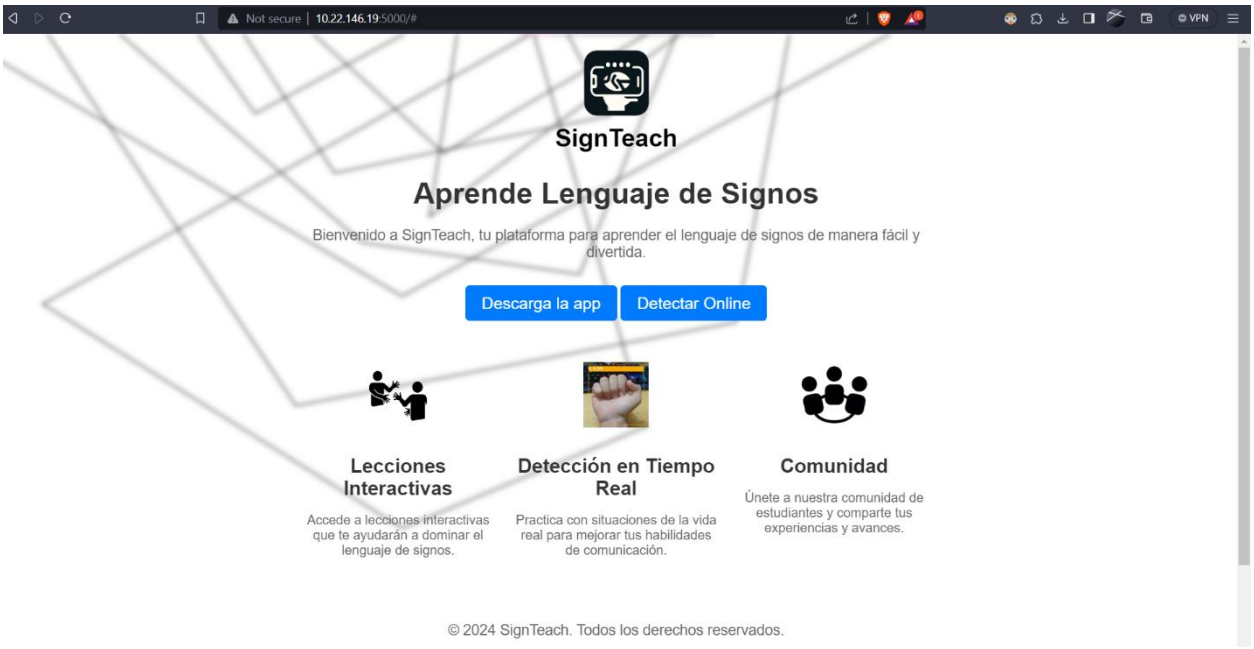

Imagen 4: Diseño de página web de la app, que podría incluir detección en directo con el uso de la cámara a través del navegador con WebSockets o WebRTC.

Fuentes y créditos

El dataset utilizado para entrenar el modelo YOLO 8 fue proporcionado por David Lee en octubre de 2020 y se puede encontrar en el siguiente enlace:

- https://public.roboflow.com/object-detection/american-sign-language-letters/1

La documentación y el desarrollo de la aplicación Android SignTeach se basó en recursos en el sitio web oficial de Android Developer:

- https://developer.android.com/