

RECONOCIMIENTO DE LENGUAJE DE SIGNOS USANDO IA

AUTORES: Alejandro Sorolla Martínez & Marcos Olmo López & Pablo Colomino Granell

1. FLUJO DEL PROYECTO

NOTA: lee **readme.md** para probar el programa.

Nuestro trabajo está compuesto de cuatro códigos con funciones propias.

Primero creamos el dataset del proyecto. Este se aloja en en una carpeta, que a su vez contiene otras carpetas con todas las clases. Cada clase está compuesta de 200 imágenes de cada letra más algunas funciones extra.

El siguiente paso es procesar las clases para extraer características, que nos servirán para entrenar nuestro modelo. Utilizamos la biblioteca MediaPipe para detectar los 21 puntos clave (landmarks) de cada mano en las imágenes. Luego, normalizamos estos puntos para que todas las manos estén alineadas en una posición común. Recorremos cada carpeta, procesamos sus imágenes y almacenamos los datos extraídos que se guardarán en un archivo .pickle.

Seguidamente entrenamos nuestro modelo de clasificación a partir del archivo .pickle, cargando las características numéricas extraídas de las imágenes junto con sus etiquetas. Estos datos se convierten en arrays de numpy y se dividen en un conjunto de entrenamiento (80%) y otro de prueba (20%). A continuación, se entrena y guarda el modelo basado en Random Forest, del que se evalúa la precisión, que es de un 100%, aunque tiene dificultades para diferenciar algunos gestos.

Finalmente nos ponemos en marcha con el sistema de reconocimiento en tiempo real. Accedemos a la cámara y usamos MediaPipe para detectar las manos y extraer sus puntos clave, generando un vector para que el modelo pueda predecir qué gesto es. El resultado se muestra en pantalla en tiempo real, y una vez que el usuario finaliza, el texto completo reconocido es mostrado y leído por un sistema TTS que podremos personalizar con nuestra propia voz.

2. DESCRIPCIÓN TÉCNICA

Reconocimiento de la mano (MediaPipe):

Para conseguir un *tracking* eficaz de la mano, se hace uso de 2 modelos de Deep Learning (CNNs) que trabajan juntos:

- Un modelo (ligero computacionalmente) ‘detector de palmas’ que recibe una imagen RGB completa que debe contener una mano y localiza a partir de esta la palma de la mano y crea una ‘caja delimitadora’ alrededor de la palma que discriminará la parte de interés de la imagen (la mano con los dedos) del resto de esta. En este proceso, se detecta únicamente la palma porque esta es más sencilla de diferenciar del fondo que con los dedos en conjunto. Finalmente, normalizará este recorte (se aplican rotaciones, escalados y traslaciones para que la imagen siempre sea estándar y facilitar el trabajo posterior) y lo enviará a la segunda CNN.
- Un modelo (más profundo) de ‘*landmarks*’, o marcas, que son una serie de puntos localizados en los dedos y en la palma de la mano. Este modelo recibe la imagen recortada, de manera que se ahorra una gran cantidad de trabajo, y establece los *landmarks*, siendo internamente coordenadas en 2.5D (x,y,z), ya que con una sola cámara común no se puede determinar a ciencia cierta la profundidad de un punto en la imagen y este es un valor estimado relativo a la mano.

El detector de palmas solo se aplica en el primer frame o cuando se detecta que la mano ya no está en cámara y vuelve a aparecer, ya que se usa información del frame anterior para estimar la posición de la mano en los posteriores, ahorrando así costes computacionales y consiguiendo un resultado a tiempo real más satisfactorio.

Para ser entrenado, se hizo uso de 3 datasets en conjunto:

- In-the-wild dataset: fotografías de manos de distintos tamaños, iluminación, localización geográfica, apariencias... Pero sin gestos.
- In-house collected gesture dataset: todo tipo de ángulos de todos los gestos físicamente posibles. Pero pocas variaciones de fondo y solo 30 personas. (Este dataset y el primero combinan muy bien)
- Synthetic dataset: este fue creado con un modelo 3D comercial de una mano, renderizada con iluminaciones complejas aleatorias, 3 cámaras, distintas texturas (tonos de piel), etc. para cubrir mejor los gestos y mejorar la profundidad.

Modelo clasificador (RandomForest):

Random Forest es un algoritmo de *aprendizaje supervisado* basado en conjuntos, que combina muchos árboles de decisión para mejorar la precisión del modelo. En nuestro caso, los puntos de entrada son los puntos clave de la mano, que detecta MediaPipe, y sus etiquetas. Para cada imagen (o frame), se extrae un vector de características. El modelo Random Forest entrena varios árboles de decisión, y cada uno de ellos utiliza un subconjunto aleatorio de las imágenes de entrenamiento y, en cada nodo, un subconjunto aleatorio de las características.

Al realizar una predicción, el vector de características de la imagen se evalúa recorriendo cada árbol del bosque. Cada árbol proporciona una predicción individual, y la clase final del modelo es aquella que recibe la mayoría de votos entre todos los árboles (votación por mayoría).

Text to speech (SPANISH-F5)

Este sistema de TTS está entrenado con un modelo de Diffusion Transformer (DiT). Es una variante del Transformer diseñada específicamente para tareas de generación usando procesos de difusión (diffusion), es decir, generar datos eliminando progresivamente ruido desde una distribución inicial hasta llegar a una señal limpia .

Se utiliza flow matching para aprender la trayectoria entre ruido y datos reales. Enseña al modelo a transformar ruido en voz real, siguiendo un camino directo y eficiente:

1. El modelo empieza con ruido.
2. Luego aprende cómo cambiar poco a poco ese ruido hasta que se convierte en un audio que diga lo que tú quieres.
3. Para eso, el modelo aprende la mejor manera de transformar ese ruido directamente hacia el audio correcto, sin pasos complicados.

3. POSIBLES MEJORAS

Aumento del dataset (precisión): se podría mejorar la precisión del modelo si este se entrenara con más manos de distintos tamaños, tonos de piel y formas, distintos fondos e iluminaciones y más cantidad de imágenes por mano (en este caso generamos 200 por clase (letra del abecedario)).

Aumento del dataset (funcionalidad): otra mejora considerable sería contar con la misma información del dataset para la mano izquierda. De esta manera, se podrían usar ambas.

Mejora del dataset: hay muchas mejoras a considerar a la hora de crear las imágenes que formarán el dataset. Rehacerlo con las mejores condiciones posibles (iluminación, fondo plano, movimientos precisos) mejoraría la precisión y captura de la mano en entornos menos favorables.

4. BIBLIOGRAFÍA

- *Instrucciones del reconocimiento de signos:*
<https://www.youtube.com/watch?v=MJCSjXepaAM>
- *Código base del reconocimiento de signos:*
<https://github.com/computervisioneng/sign-language-detector-python>
- *Estudio MediaPipe Hand Detector:* <https://arxiv.org/pdf/2006.10214>
- *Documentación RandomForestClassifier:*
<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
- *API SPANISH-F5 TTS:*
<https://huggingface.co/spaces/redradios/F5-TTS-Sp>
- *Documentación SPANISH-F5 TTS:*
<https://github.com/maxmcoding/Spanish-F5>
- *Vídeo explicativo Random Forest:* <https://www.youtube.com/watch?v=v6VJ2RO66Ag>
- *Documentación F5-TTS:* <https://arxiv.org/abs/2410.06885>
- *Información de MediaPipe Hand Detector:*
https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker?hl=es-419#models
- *Imagen readme ASL:* <https://es.ava.me/asl/alphabet>