Sistema de Reconocimiento de Lengua de Signos

Rubén García Viciana

Resumen—El objetivo de este trabajo es desarrollar un sistema capaz de reconocer signos del lenguaje de señas argentino (LSA) a partir de videos. Para ello, se utiliza la base de datos LSA64, compuesta por 64 señas realizadas por 10 sujetos, con un total de 3200 videos. El proceso incluye la segmentación de videos en frames, detección de manos y normalización. Se emplea una red neuronal que combina una CNN, para extraer características espaciales de las manos, y una LSTM, que modela la secuencia temporal del movimiento. El modelo se entrena con estos datos para clasificar correctamente los signos y así poder hacer una buena traducción de estos. Se clasifica correctamente el signo en cada video de manera offline. Los resultados tienen una precisión superior al 99% en los datos de test, y en el caso de los videos propios, se logra un rendimiento del 73% en condiciones diferentes a las de la base de datos.

Palabras clave—LSA (Lenguaje de Señas Argentina), Pidgin (Sistema que ha sido formado por una serie de signos que han sido pactados entre personas que no tienen una lengua en común), CNN (Convolutional neural network), Deep learning (Es un tipo de machine learning que entrena a un modelo para que realice tareas).

Abstract—The objective of this project is to develop a system capable of recognizing signs from the Argentine Sign Language (LSA) using video input. For this purpose, the LSA64 dataset is used, which consists of 64 different signs performed by 10 subjects, totaling 3200 videos. The process includes video segmentation into frames, hand detection, and normalization. A neural network is employed that combines a Convolutional Neural Network (CNN) to extract spatial features of the hands and a Long Short-Term Memory (LSTM) network to model the temporal sequence of movement. The model is trained on this data to correctly classify the signs, enabling accurate translation. Each video is classified offline, without requiring an internet connection. The results show over 99% accuracy on the test set, and in the case of custom-recorded videos, the system achieves a 73% accuracy under different conditions from those in the dataset.

Index Terms—LSA (Lenguaje de Señas Argentina), Pidgin (a system that has been formed by a series of signs that have been agreed upon by people who do not have a common language). CNN (Convolutional neural network), Deep learning (It is a type of machine learning that trains a model to perform tasks).

1 Introducción

La lengua de signos es un método de comunicación visual-gestual que permite a las personas sordas o con dificultades auditivas comunicarse a través de gestos, expresiones faciales. Como los idiomas hablados, la lengua de signos varía según el país o la región, de manera que han desarrollado su propia gramática, vocabulario y expresiones. A nivel internacional, hay múltiples lenguas de signos, como el Lenguaje de Señas Americano (ASL), Lengua de Signos Catalana (LSC), el Lenguaje de Señas Español (LSE) o el Lenguaje de Señas Argentino (LSA). Existen un movimiento para hacer un sistema universal de signos como puede ser el Sistema de Signos Internacional (SSI) [6] que aunque esta no sea una lengua, es un pidgin. Es decir, es un sistema que ha sido formado por una serie de signos que han sido pactados entre personas que no tienen una lengua en común.

Existen dos tipos principales de signos en estas lenguas: signos **estáticos** y signos **dinámicos**. Como se ve en la **Figura 1** los signos estáticos, como A, B y muchas letras del alfabeto, pueden reconocerse a partir de una imagen fija, ya que su significado depende únicamente de la forma de la mano y su posición. En cambio, los signos dinámicos como pueden ser J, Z o la mayoría de palabras requieren movimiento para transmitir información, ya que la trayectoria, la dirección y la velocidad del gesto son fundamentales para diferenciar significados.

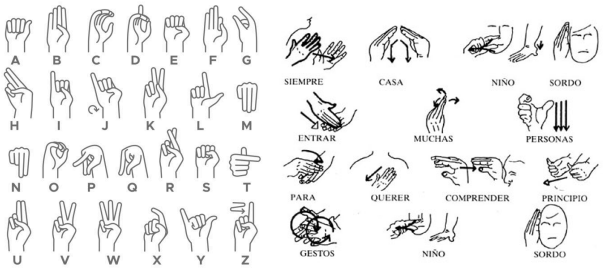

Fig 1. En la siguiente imagen se muestra la diferencia entre signos estáticos y signos dinámicos.

Aunque existen soluciones para traducir texto y voz como pueden ser Google Traductor u otros. No hay herramientas para traducir el lenguaje de señas, lo cual deja fuera a muchas personas sordas. Como los signos combinan gestos, movimientos y expresiones faciales, su reconocimiento automático es más complejo. Por eso, es necesario desarrollar sistemas que permitan traducir señas.

Según se puede observar en **Table 1** siguiente referencia

[15], la mayoría de los sistemas automáticos de reconocimiento de signos se centran en identificar letras o palabras individuales, sin llegar a traducir frases completas. Esto se debe, en gran parte, a la complejidad que implica procesar secuencias continuas de signos. Una de las principales dificultades radica en determinar cuándo empieza y cuándo termina un signo dentro de una secuencia continua de signos. Esta segmentación entre palabras es muy importante para poder agrupar correctamente los signos y darles un significado conjunto.

Como anécdota curiosa, este tipo de sistema también podría servir como **herramienta de validación** en eventos públicos. Un caso muy conocido ocurrió durante el funeral de Nelson Mandela en 2013, donde se descubrió que el supuesto intérprete de lenguaje de señas no estaba transmitiendo mensajes coherentes, y simplemente realizaba gestos sin sentido alguno. Esta situación generó indignación en la comunidad sorda.

El reconocimiento automático de lenguaje de señas tiene múltiples desafíos, ya que no solo implica identificar gestos con las manos, sino también comprender el contexto y las variaciones individuales en la ejecución de cada signo. Por este motivo, la mayoría de investigaciones actuales siguen centradas en unidades mínimas del lenguaje, como letras o palabras aisladas [15].

En este contexto, la motivación principal de este proyecto es desarrollar una **herramienta que permita a personas que no conocen el lenguaje de señas comunicarse de forma más fluida con usuarios que sí lo utilizan**. Así, se pretende evitar que una persona sorda tenga que recurrir a escribir o esforzarse por hacerse entender mediante gestos, facilitando una interacción más natural e inclusiva. Además, este tipo de herramienta también podría resultar útil para personas sordas que estén aprendiendo lengua de señas y necesiten apoyo para identificar signos desconocidos.

Por tanto, este proyecto se centrará en el reconocimiento de palabras en lengua de señas argentina (LSA) a partir de videos capturados mediante una webcam. El **objetivo es identificar y analizar el signo realizado y convertirlo automáticamente en texto**. En el apartado de procedimiento se explica la planificación que se llevara a cabo para la realización del proyecto y la metodología que se seguirá en cada una de las fases de este.

2 Objetivos

En este caso, el principal objetivo que queremos conseguir es que a partir de un video se haga la detección del signo siguiendo diferentes procedimientos y metodologías que explicaré posteriormente. Una vez hecha la detección del signo se mostrará por pantalla la palabra o frase relacionada con dicho signo. Es decir, la función de un traductor.

El objetivo principal que es la detección de signos tiene diferentes subdivisiones que llevarán al resultado esperado.

1. Buscar bases de datos
2. Procesamiento de videos
   a. Obtención de frames del video
3. Extracción de características
   a. Segmentación manos
   b. Coordenadas manos
4. Modelo
   a. Preparación de datos de aprendizaje
   b. Creación y entrenamiento del modelo
5. Testeo del modelo

Estas etapas son **críticas**, ya que sin su correcta ejecución no sería posible alcanzar el objetivo principal del proyecto.

3 Estado del arte

Para el desarrollo de este proyecto ha sido necesario seleccionar una base de datos adecuada. Se consideraron varias opciones. En primer lugar, la SIGNUM Database [4], una base de datos alemana que incluye grabaciones de lengua de signos realizadas por diferentes usuarios, centrada en palabras y frases en alemán. En segundo lugar, la RWTH-PHOENIX-Weather [5], también en alemán, compuesta principalmente por videos de noticias donde aparecen intérpretes de lengua de signos, incluyendo además una traducción al inglés. Sin embargo, ambas bases presentaban una dificultad importante: requerían realizar una distinción manual de las palabras y, al estar en alemán, suponían una barrera de idiomas adicional.

Finalmente, seleccione la base de datos argentina LSA64[1], que contiene un conjunto de 3200 videos. Cada video corresponde a una de las 64 palabras incluidas en el conjunto de datos, repetidas por 10 sujetos diferentes en 5 ocasiones cada una. Esta opción ha sido la seleccionada por su enfoque claro en palabras individuales, su lengua y la facilidad para trabajar con las etiquetas ya proporcionadas, lo que permite centrar el esfuerzo en el desarrollo del sistema de reconocimiento.

Existen diversas técnicas para el reconocimiento de signos en lengua de signos, entre las que destacan los métodos basados en **contornos, esqueleto** y **segmentación por color** se pueden ver en la **Figura 2**.

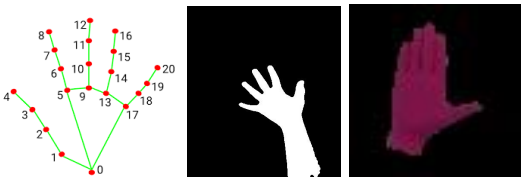

Fig 2. Imágenes de cada una de las técnicas de reconocimiento de signos.

- **Métodos basados en contornos**, que detectan la silueta de la mano mediante la comparación entre el fondo y la imagen en tiempo real. Aunque son simples, suelen ser muy sensibles a cambios en la iluminación o al movimiento en el entorno.
- **Métodos basados en landmarks o esqueleto**, utilizando herramientas como MediaPipe o OpenPose, que extraen puntos clave (joint positions) de las manos y el cuerpo. Suelen ofrecer buenos resultados, pero requieren modelos preentrenados y tienen un coste computacional mayor.
- **Segmentación por color**, como este proyecto, se basa en el uso de guantes de colores diferentes para

cada mano (en este caso, verde y rosa). Esta técnica permite una detección precisa de las manos incluso en condiciones visuales variadas, y facilita el análisis del movimiento en el tiempo.

En mi proyecto he optado por esta última estrategia, utilizando guantes verdes y rosas para detectar y segmentar las manos en los videos.

Gracias a los avances en **deep learning**, se ha popularizado la combinación de redes neuronales convolucionales (**CNN**) para extraer características espaciales de cada frame, y redes neuronales recurrentes (RNN), especialmente **LSTM**, para modelar la secuencia temporal del gesto. Estas arquitecturas han demostrado ser eficaces en la clasificación de signos dinámicos, ya que combinan información visual y temporal. Este proyecto se apoya en este último enfoque, combinando una **CNN** para analizar las imágenes segmentadas de las manos con una **LSTM** para capturar la evolución del movimiento a lo largo del tiempo.

4 PROCEDIMIENTO

En la siguiente sección se explicará la planificación y la metodología que se llevará a cabo para la realización del proyecto. Que estará dividida en diferentes fases. En este caso adoptaré un **método en cascada**, que explicaré posteriormente.

La **primera fase** consistirá en el procesamiento de los vídeos, donde se tratara cada uno de ellos para obtener los frames y la normalización de estos, asegurándonos que sean homogéneos. La **segunda fase** consistirá en la extracción de características donde se detectaran las manos y sus movimientos, es decir, coordenadas. La **tercera fase** consistirá en el diseño y entrenamiento del modelo, donde se hará la clasificación de cada uno de los signos y el movimiento realizado. Finalmente, la **cuarta fase** consistirá en la fase de testeo en la que haremos las pruebas con videos con el objetivo de que podamos hacer la traducción de estos.

4.1 Planificación y Metodología

En esta sección se explicará todo toda la planificación de tiempo que llevara realizar cada una de las tareas sin entrar mucho en detalle en cada una de estas, ya que pueden tener variaciones de tiempo y realización. En este caso se marca como finalización la propuesta de informe final, ya que para esa fecha ya debería estar acabado el proyecto, aunque todavía faltaría la presentación y defensa del TFG.

La siguiente planificación está relacionada con cada una de las subdivisiones realizadas anteriormente del objetivo principal del proyecto. Con la idea de así poder realizar una planificación que se adapte mejor a los resultados esperados del proyecto

En la **Figura 3** se muestra el tiempo que previsto que utilizaré para realizar cada una de las tareas y como se ha seguido la metodología en **cascada**, un modelo de **desarrollo secuencial** en el que cada fase debe **completarse antes de comenzar la siguiente**. Esta metodología resulta adecuada para proyectos donde solo trabaja **una persona** y se puede definir con claridad cada etapa del proceso y donde los **requerimientos no van a cambiar significativamente durante el desarrollo**.

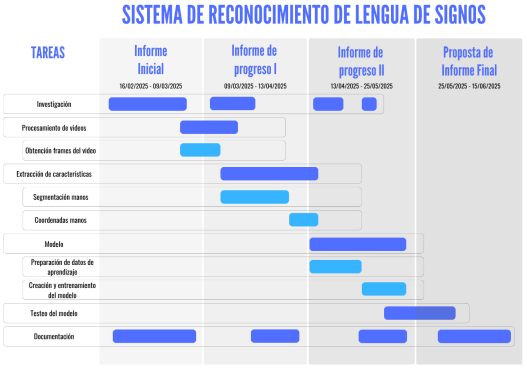

Fig 3. En la siguiente imagen se muestra que tiempo dedicaré a cada una de las tareas para realizar el proyecto según las entregas que tengo que realizar. (Si no ven bien la imagen está en el Apéndice)

El proyecto se ha estructurado en varias fases. En la fase inicial se realizó una investigación para definir la idea, seleccionar la base de datos y establecer el enfoque general. Los informes de progreso I y II se centraron en la codificación, dividida en tareas como procesamiento de videos, extracción de características, entrenamiento del modelo y testeo. Finalmente, en la fase del informe final se evaluará el sistema y se completará la documentación. A lo largo de todo el proceso, se ha llevado a cabo una documentación continua para reflejar el desarrollo y adaptar posibles cambios.

4.2 Explicación de las fases principales

Ahora se explicará de manera clara que procedimientos realizaré en cada una de las **fases principales**.

1. Procesamiento de videos

En la **primera fase**, que consiste en el procesamiento de videos, se tratará de hacer la separación por frames de cada uno de ellos, depende del signo, pero cada uno de los videos tiene una duración aproximada de 1 o 2 segundos.

Como la cámara utilizada en la base de datos para la grabación es una Sony HDR-CX240, el trípode se colocó a 2 m de la pared y a una altura de 1,5 m. Y además, la resolución de los videos es de 1920 x 1080, a 60 fotogramas por segundo.

2. Extracción de caracteristicas

Una vez tengamos los frames se pasará a la **segunda fase** a la extracción de características, donde se procederá a hacer la segmentación de cada uno de los guantes para así poder obtener cada una de las manos, posteriormente se hará una normalización con el objetivo de que cada imagen sea homogénea con las demás. Además, se extraerá el movimiento efectuado por cada una de las manos con el objetivo de diferenciar los signos y las diferentes palabras.

Para confirmar que lo hemos efectuado correctamente, la base de datos que hemos escogido [1] cuenta con un conjunto de trayectorias efectuadas por cada uno de los guantes en cada signo, que nos será muy útil a la hora de

monitorizar el movimiento. Esto se puede ver en la Figura 4. En nuestro caso, la trayectoria del signo la obtendremos a partir del orden de los frames obtenidos del video.

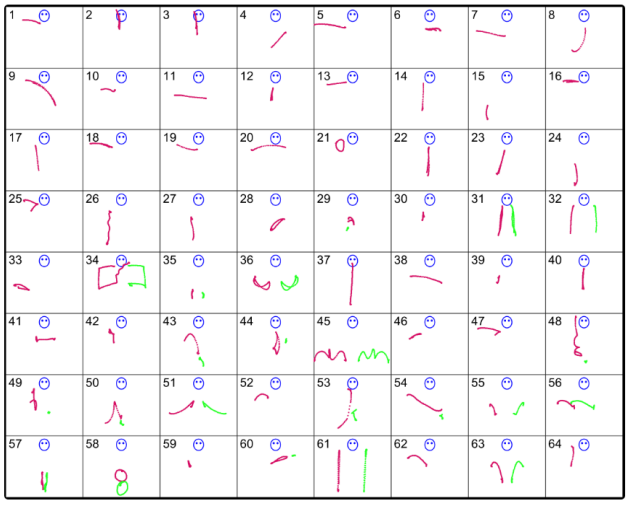

Fig 4. En la siguiente imagen se muestra que cada uno de los recorridos que hace la mano en cada signo. El color rosa pertenece a la mano derecha y el verde a la mano izquierda.

Para abordar la detección de signos dinámicos, he investigado distintos métodos utilizados en el reconocimiento de manos y movimientos en visión por computador. Algunas soluciones existentes incluyen:

- **Mediapipe Hands**[12]: Un framework desarrollado por Google que permite detectar y rastrear la posición de las manos mediante modelos de aprendizaje automático preentrenados.
- **OpenPose**[13]: Una herramienta que extrae puntos clave del cuerpo, incluyendo las manos, para analizar posturas y movimientos.
- **Optical Flow**[14]: Un método basado en el análisis de la variación del movimiento entre fotogramas consecutivos, lo que permite estimar la dirección y velocidad de los objetos en movimiento.

Sin embargo, en este proyecto no se ha utilizado ninguna de estas librerías, ya que en mi caso hemos priorizado la base de datos y hay problemas al detectar la mano con guantes utilizando estas librerías. En su lugar, una vez detectada la mano en el video, extraigo las coordenadas **x** e **y** de su posición en cada fotograma. A partir de esta información, analizo la trayectoria y los cambios en la posición de la mano para identificar el signo dinámico. Este enfoque me permite realizar la detección de movimientos sin depender de modelos preentrenados, lo que aporta mayor control sobre el proceso y evita posibles sesgos introducidos por bases de datos externas.

3. Modelo

En nuestro caso lo que necesitamos es extraer las características de las imágenes y manejar como cambian dichas características en el tiempo. Por lo tanto, en la **tercera fase** que consistirá en la creación y el entrenamiento del modelo.

Para ello, he analizado diferentes enfoques citados en la bibliografia como **Redes Neuronales Recurrentes (RNN)** y **Redes Neuronales Convolucionales (CNN)** [2][3]. Según he revisado las CNNs han demostrado proporcionar mejores resultados en la extracción de características de cada frame del video, como la forma y posición de las manos y dedos.

No obstante, para capturar el movimiento a lo largo del tiempo y las secuencias de imágenes necesitamos una red que pueda manejar información temporal. Para ello podemos utilizar, **LSTM (Long Short-Term Memory)** [7] una arquitectura de redes neuronales recurrentes (RNN) que está especializada en procesar datos secuenciales.

De esta forma, la **CNN** extraerá las características de cada frame y la **LSTM** se encargará de aprender las relaciones entre cada una de las imágenes en el tiempo.

4. Testeo del modelo

Una vez ya pasada la fase del modelo llegamos a la última de las fases. La **fase test** del modelo donde nos aseguraremos de que el modelo funcione tal y como se espera, con diferentes videos que serán separados para realizar esta prueba. Después de esto sé procederá a recoger y documentar los resultados obtenidos además del porcentaje, errores y aciertos.

5 Desarrollo

En la siguiente sección explicaré como he realizado cada uno de los procedimientos que he explicado anteriormente de una manera mucho más detallada, es decir, cada una de las sub-fases.

5.1 Obtención de frames del video

El primer paso en el procesamiento de los videos consiste en la **extracción de frames**. Esta etapa es fundamental para estandarizar el conjunto de datos y preparar las imágenes que serán posteriormente utilizadas para la segmentación de manos.

Para garantizar consistencia entre videos, he decidido extraer **30 frames por video**, distribuidos de manera uniforme a lo largo de la duración del mismo. Esto permite capturar el movimiento completo de la seña sin necesidad de procesar todos los frames del video. (Poner cuanto se reduce teniendo en cuenta 1 seg)

El script que he desarrollado para esta tarea utiliza la librería **OpenCV (cv2)** para la lectura y procesamiento de los videos. A cada frame extraído le aplico un redimensionamiento a una resolución de **512x512 píxeles**, con el objetivo de normalizar las dimensiones de entrada para las etapas posteriores.

El proceso comienza con la lectura de los videos que tenemos en la carpeta. Para cada uno, calculo el número total de frames y selecciono **30 frames equidistantes** a lo largo del video. A medida que leo los frames, compruebo si el índice del cuadro actual coincide con alguno de los seleccionados. Si es así, se redimensiona a **512x512 píxeles** y se guarda como una imagen con un nombre secuencial

(**frame_0000.jpg, frame_0001.jpg, etc.**) en una carpeta dedicada a ese video.

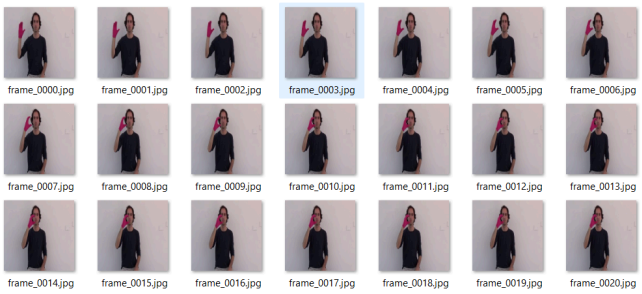

Fig 5. En la siguiente imagen se muestra un ejemplo de la extracción de frames que se hace para un video.

5.2 Procesamiento de frames: Segmentación de manos

El objetivo de esta etapa es procesar los frames extraídos de los videos para **identificar y registrar la posición de las manos** en cada uno de ellos. Esto es crucial, ya que la segmentación precisa de las manos es uno de los primeros pasos en la interpretación del lenguaje de señas. El proceso se lleva a cabo mediante varios pasos, que incluyen la segmentación basada en colores, el seguimiento de las posiciones de las manos a lo largo del tiempo y la detección de la cabeza para contextualizar el movimiento.

Para la segmentación de las manos en los videos de lenguaje de señas, se utiliza el color de los guantes, que en nuestro caso son violetas y verdes. Para mejorar la detección, los frames se convierten del espacio de color RGB al espacio **HSV (Hue, Saturation, Value)** [8]**,** que resulta más efectivo para separar colores bajo distintas condiciones de iluminación, se puede ver un ejemplo de como funciona en la **Figura 6**.

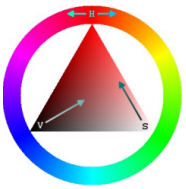

Fig 6. Imagen en la que se entiende de donde proceden los valores de HSV

Por lo tanto, definimos un rango específico para identificar los guantes de forma precisa, independientemente de la iluminación o variaciones ambientales.

Una vez que se calcula el rango de los colores, se comienza con el procesamiento de los frames. El sistema recorre todas las subcarpetas dentro de la carpeta de frames, correspondientes a cada video procesado previamente. Para cada video, se abre cada uno de los frames y se aplica el proceso de **detección de la cabeza** y **segmentación de las manos**.

Utilizando un clasificador preentrenado de **Viola-Jones** [11] para la detección de cara, analizamos cada uno de los frames para identificar la presencia de la cabeza de la persona. Para realizar la detección realizamos una conversión de la imagen a escala de grises y posteriormente aplicamos el **clasificador de caras**. Si se detecta la cabeza, se registra la posición del centro de la cara, lo cual puede ser útil para obtener un contexto del movimiento de las manos en relación con la cara del usuario. En caso de no detectar una cara, asignamos una posición nula y vamos directamente a las manos. En la **figura 7** se puede ver que detectamos la cara de la persona que realiza el gesto correctamente:

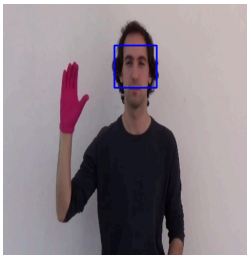

Fig 7. Imagen en la que se puede ver que la cara del usuario que realiza el gesto se detecta correctamente.

Una vez se ha detectado el rostro. A continuación, se realiza la **segmentación de las manos** en los frames utilizando el rango de color previamente calculado. La segmentación permite identificar las áreas correspondientes a las manos en el video, basándose en el color de los guantes (violeta y verde).

En lugar de trabajar con el fondo original de los frames, se guardan las **manos segmentadas con el fondo negro**. Para hacer esto se normalizarán las imágenes de las manos a un tamaño de 64x64. Esto ayuda a simplificar el análisis posterior y mejora la calidad del modelo de reconocimiento de signos, ya que se eliminan distracciones de otros elementos en la imagen. El conjunto de imágenes generadas se puede observar en la **figura 8**.

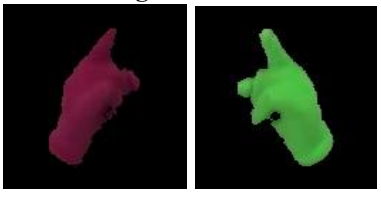

Fig 8. Imágenes con cada una de las manos segmentadas con fondo negro, para utilizarlas en el modelo

5.3 Coordenadas de movimiento

Para cada mano detectada, se calcula su posición en el frame. Específicamente, se identifican las coordenadas de la caja delimitadora (bounding box) de la mano, como se puede ver en la **figura 9**. Luego, se registra la posición central de la mano, lo que facilita el seguimiento de su movimiento a lo largo del video. Este paso es crucial para la posterior clasificación de los signos, ya que permite capturar el movimiento de las manos y guardarlo en un **fichero CSV** de **posiciones**.

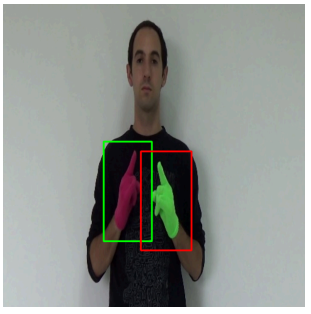

Fig 9. Imagen en la que se puede ver que las manos del usuario que realiza el gesto se detectan correctamente y genera una bounding box

alrededor.

Durante el procesamiento de los frames, se **registra la trayectoria** de cada mano guardando su posición en cada frame como coordenadas (x, y), permitiendo así un seguimiento continuo del movimiento a lo largo del video. Para aumentar la robustez ante cambios en la posición de la persona dentro del encuadre, se usa la detección del rostro como referencia.

Cuando se **detecta la cabeza**, se calcula la **posición relativa** de cada mano respecto a ella, normalizando las coordenadas según el tamaño y ubicación del rostro (diferencia entre centros dividida por ancho y alto de la cabeza). Esto asegura **consistencia** incluso si la persona se **acerca** o **aleja** de la cámara. Si **no se detecta la cabeza**, se usan las **coordenadas absolutas**.

Finalmente, se genera una imagen que muestra la trayectoria de ambas manos, como se ilustra en la **Figura 10**, ofreciendo una representación visual clara del movimiento.

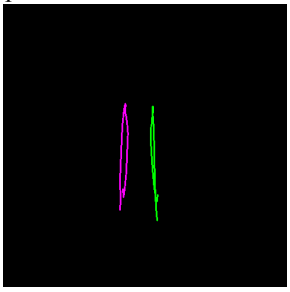

Fig 10. Imagen de la trayectoria de los signos, en este caso es el signo 31 (Desayuno)

5.4 Preparación de datos para aprendizaje

En esta sección, se describe el proceso de creación del conjunto de datos (DataSet) necesario para entrenar el modelo de reconocimiento de lenguaje de señas argentino (LSA) utilizando las posiciones y las imágenes segmentadas de las manos de los usuarios (violeta y verde) de cada frame.

El proceso de generación del DataSet se encarga de crear un conjunto de datos estructurado que pueda ser utilizado para entrenar un modelo de redes neuronales.

Se comienza leyendo las imágenes de las manos (violeta y verde) y las posiciones de estas a partir de los archivos CSV generados durante el procesamiento de los frames. Cada fila del CSV corresponde a las posiciones de las manos para un frame específico, incluyendo las coordenadas X e Y tanto de la mano violeta como de la mano verde.

En el caso de las **manos** se cargan de lo extraído anteriormente, se convierten a escala de grises y se normalizan para que sus valores estén en el rango de [0,1] con el objetivo de facilitar el procesamiento. Como se puede ver en la **Figura 11**.

Fig 11. Imagen de la mano derecha con la que el modelo se entrenara

En el caso del **posicionamiento de las manos**, se genera un mapa de calor (**heatmap**) para cada una utilizando una distribución **gaussiana 2D** centrada en la posición detectada de la mano (violeta o verde). Se crea una imagen donde el valor máximo está en el centro de la mano y disminuye suavemente hacia los bordes, simulando una campana gaussiana. Esto permite representar la posición de forma más suave y continua, lo que facilita al modelo aprender patrones espaciales más robustos.

Las coordenadas de las manos se normalizan previamente del tamaño original de 512x512 píxeles a 64x64, y a partir de esa posición se calcula el mapa de calor. Este tipo de representación aporta al modelo una información espacial más rica, ya que no solo le indica la ubicación, sino también un área de influencia alrededor del centro de la mano.

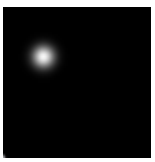

Fig 12. Posición de la mano derecha con gaussiana

Hay que tener en cuenta que en todos los signos no salen las dos manos. Por lo tanto, lo que se hace en caso de que no salga la mano izquierda (verde) se rellena el canal 2 y 4 de dicha mano con ruido suave, los valores se rellenan con una distribución normal centrada en 0 y con una desviación baja (0.01). Este ruido actúa como un “relleno neutral” para no romper la estructura del dataset y evitar que el modelo aprenda patrones erróneos. Es decir, que el modelo no ignore dicho canal.

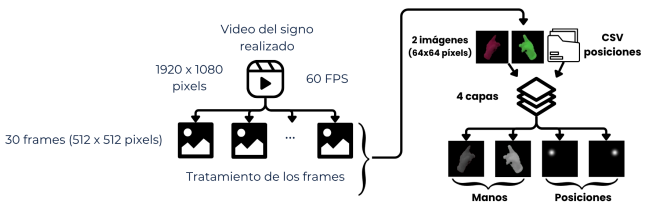

Fig 13. Esquema de creación de capas de imagen para entrenamiento

En cada secuencia de frames de un video, las imágenes se combinan en una sola matriz que pasa a tener 4 canales. Cada **imagen** final **tiene 4 canales** como se puede ver en la **Figura 13**:

- **2 canales** para las **manos** (violeta y verde) que contienen las imágenes de las manos.
- **2 canales** adicionales que representan las **posiciones** de las manos.

La imagen resultante es una combinación de las imágenes de las manos con las posiciones de estas.

El conjunto de datos final X tiene la forma (n_videos,

n_frames, 64, 64, 4), donde n_videos = 3200 es el número de videos, n_frames = 30 es el número de frames por video, y 64, 64, 4 es el tamaño y número de canales de la imagen (2 canales de imágenes de las manos y 2 canales de posiciones).

Las etiquetas y son convertidas a formato one-hot usando to_categorical() de Keras, con un número de clases definido por n_classes (64 en este caso, que es el numero de signos).

5.5 Creación y Entrenamiento del modelo CNN + LSTM

Para la tarea de reconocimiento de señas dinámicas, como explique anteriormente opte por una arquitectura que combina redes convolucionales (CNN) y redes recurrentes tipo LSTM (Long Short-Term Memory), lo cual permite aprovechar tanto la información espacial contenida en cada frame como la información temporal a lo largo de la secuencia de video.

La arquitectura del modelo la he definido como una red secuencial con las siguientes capas (**Figura 15**):

- **Bloque convolucional aplicado a cada frame de forma independiente** mediante TimeDistributed, que permite aplicar la misma CNN a cada imagen de la secuencia. Tres capas Conv2D con 32, 64 y 128 filtros respectivamente.
- **Aplanamiento de características**: para convertir las características extraídas por la CNN de cada frame en un vector
- **Capa LSTM**: esta capa recibe la secuencia de vectores aplanados y aprende dependencias temporales a lo largo de los 30 frames.
- **Clasificación**: Se hace una clasificación multiclase con 64 clases diferentes que serían la se los signos

El modelo lo compilo con:

- **Optimizador**: Adam.
- **Función de pérdida**: categorical crossentropy (ya que las etiquetas están codificadas en one-hot).
- **Métrica**: accuracy.

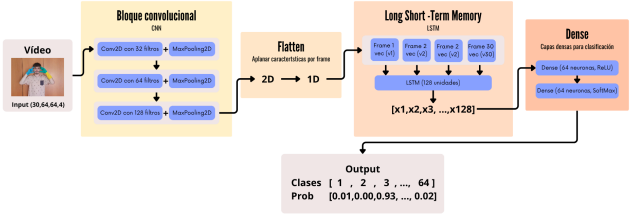

Fig 14. Gráfica del funcionamiento del modelo

Para **entrenar** el modelo, el dataset se divide en un 80% para entrenamiento y un 20% para validación, usando el parámetro **stratify** para mantener la misma proporción de clases en ambos conjuntos y evitar sesgos.

El entrenamiento se realiza durante 10 épocas, con un tamaño de lote de 4, lo que permite actualizar los pesos tras cada grupo pequeño de muestras. Al finalizar, el modelo se guarda en disco para reutilizarlo en futuras evaluaciones sin necesidad de entrenarlo de nuevo.

5.6 Predicción

Con el objetivo de realizar predicciones para comprobar el funcionamiento del modelo y así verificar que el modelo es capaz de generar una salida coherente, primero de todo necesitaremos el color de los guantes que va a utilizar el sujeto que realiza el signo. Para ello he codificado una web en la que el usuario puede subir una foto y seleccionar el color de guante que tiene, como se puede observar en la **figura 15**.

Fig 15. Selección de colores en la imagen (Desayuno)

Hecho esto, tendremos que ir a la sección para subir nuestro video, le daremos a confirmar y comenzará el procesamiento del video. Este se procesa con las mismas transformaciones aplicadas durante el entrenamiento:

- Redimensionamiento de los frames a 64x64.
- Extracción de 30 frames.
- Segmentación de las manos (canales violeta y verde).
- Generación de gaussianas de posición de las manos.

Este video se transforma así en una muestra con forma (1, 30, 64, 64, 4) y puede ser pasado al modelo previamente entrenado.

El modelo devuelve un vector de probabilidades, donde cada valor representa la probabilidad de que la muestra pertenezca a una de las 64 clases posibles. Con eso obtenemos:

- La clase con mayor probabilidad (argmax del vector).
- La probabilidad asociada a esa clase (max del vector).

Una vez tenemos la clase con mayor probabilidad nos disponemos a mostrar por pantalla la predicción hecha del video. En este caso, la **figura 16** es **Desayuno**.

6 RESULTADOS

En esta sección se analizan los resultados obtenidos tras el entrenamiento del modelo de clasificación con las 64 clases, diseñado para reconocer signos de la lengua de señas. En este apartado se podrán ver métricas de rendimiento, análisis visual mediante gráficas y una evaluación detallada de los errores.

6.1 Evolución del Entrenamiento

A continuación, se pueden ver las gráficas de la evolución de las métricas de precisión y pérdida durante las distintas épocas de entrenamiento.

En el caso de la curva de precisión podemos ver en la **Figura 18** que la curva muestra una tendencia creciente, indicando que el modelo mejora su capacidad de clasificación conforme avanza el entrenamiento. Se observa que tras la época 5, la precisión comienza a estabilizarse, lo cual sugiere un buen ajuste del modelo a los datos.

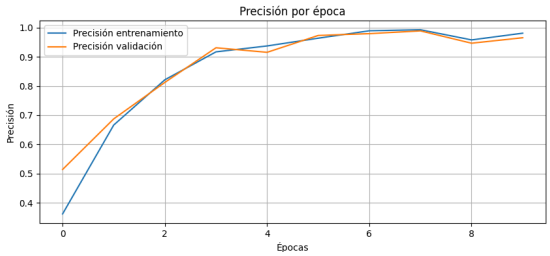

Fig 18. Gráfica de precisión por época

Por último, la curva de pérdida tal y como podemos ver en la **Figura 19** la disminución de la función de pérdida a lo largo del entrenamiento indica que el modelo está aprendiendo correctamente. Se observa una reducción constante hasta la época 5, tras la cual la pérdida se estabiliza, evitando así sobreajuste.

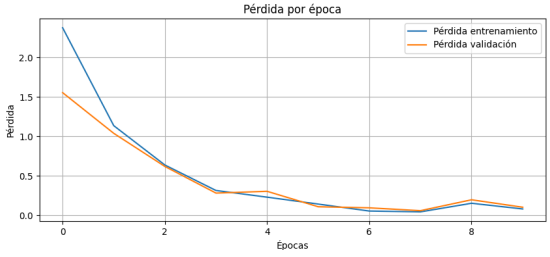

Fig 19. Gráfica de pérdida por época

6.2 Métricas de rendimiento

Ahora que hemos visto como han evolucionado las métricas durante el entrenamiento, tenemos las siguientes métricas.

Precisión (Accuracy): Durante el proceso de entrenamiento, el modelo alcanzó una precisión del 97,52% en la última época, dándonos un alto nivel de ajuste sobre los datos de entrenamiento.
Por otro lado, al evaluar el modelo entrenado sobre el **conjunto de validación**, se obtuvo una **precisión** final de **96,56%** que es la **métrica principal de interés**. Esta precisión indica el porcentaje de predicciones correctas respecto al total de ejemplos evaluados, proporcionando una medida clara de la capacidad del modelo para clasificar correctamente los signos en datos no vistos durante el entrenamiento.

Pérdida (Loss): Durante el proceso de entrenamiento, el modelo alcanzó una pérdida final del 9,75% en la última época, mostrando el nivel de error sobre los datos de entrenamiento.

La función de pérdida que he empleado ha sido la entropía cruzada categórica, ya que es adecuada para problemas de clasificación multiclase, porque mide la discrepancia entre las distribuciones de probabilidad predicha y la real.

Al evaluar el modelo sobre el **conjunto de validación**, he obtenido una pérdida final del **10,09%**, reflejando el nivel de error del modelo al enfrentarse a datos no vistos durante el entrenamiento. Esta métrica nos permite comprender cuánto se desvían las predicciones del modelo respecto a los valores reales.

Como se puede observar, la pérdida en el conjunto de validación fue del 10,09% i en el conjunto de entrenamiento (9,75%). Esto puede ser por varios factores.

- **Sobreajuste (Overfitting):** El modelo ha aprendido a representar muy bien los datos de entrenamiento, logrando una pérdida mínima. Pero, al enfrentarse a ejemplos no vistos durante el entrenamiento (validación), su rendimiento desciende, indicando que ha memorizado patrones de los datos de entrenamiento en lugar de generalizar de forma óptima.
- **Complejidad del modelo**: Los modelos combinados de redes convolucionales (CNN) y redes recurrentes (LSTM) tienen muy alta capacidad para capturar patrones complejos. Esto, si no se regula adecuadamente, puede llevar a que el modelo se ajuste demasiado al conjunto de entrenamiento.
- **Variabilidad en los datos de validación:** El conjunto de validación podría contener ejemplos menos representados en el entrenamiento, lo que eleva la pérdida medida sobre este conjunto. Este no sería nuestro caso.

A pesar de esta diferencia en la pérdida, la precisión en validación sigue siendo elevada (96,56%), lo que quiere decir que, aunque el modelo tiene algo de incertidumbre en sus predicciones fuera del entrenamiento, sigue clasificando correctamente la mayoría de los casos.

Por tanto, la pérdida en esta validación no debe interpretarse de forma aislada, sino con métricas como la **precisión** y el **F1-score** con valores como **96,56%** y **97,00%** respectivamente, las cuales reflejan la capacidad del modelo para generalizar sobre datos no vistos.

6.3 Matriz de confusión

La matriz de confusión nos permite observar de forma detallada del desempeño del modelo para cada clase, mostrando qué signos tienden a clasificarse correctamente (diagonal principal) y cuáles son más propensos a ser confundidos con otras clases (valores fuera de la diagonal).

En nuestro caso, podemos ver que la mayoría de los signos son correctamente reconocidos, ya que la diagonal principal está intensamente marcada, lo que refleja un alto número de aciertos. Sin embargo, también se identifican algunos patrones de confusión entre clases específicas.

Por ejemplo, se aprecia en la **Figura 20** que las clases 6 (Azul claro) y 15 (Nacido) presentan errores recurrentes entre sí, indicando que el modelo tiende a confundirlas más

que con otras clases.

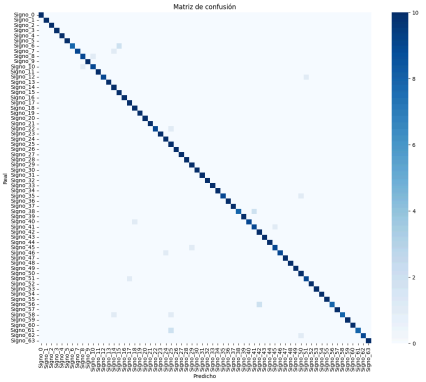

Fig 20. Matriz de confusión

De esta matriz es fundamental hacer su análisis para comprender las limitaciones del modelo y orientar hacia donde deberían ir las mejoras futuras, como ajustar el preprocesamiento de datos, aumentar las muestras de ciertas clases o aplicar técnicas de regularización que mejoren la capacidad de generalización.

6.4 Análisis de errores por clase

A continuación, podemos observar la **figura 21** que muestra las 5 clases (signos) con peor desempeño según la métrica F1-score, la cual combina tanto la precisión como la exhaustividad (recall) para ofrecer una visión equilibrada del rendimiento del modelo en cada clase.

Signo	Precisión	Recall	F1-score	Support
Signo 25	0.714	1.00	0.8333	10
Signo 41	0.818	0.90	0.8571	10
Signo 61	1.000	0.80	0.8888	10
Signo 58	1.000	0.80	0.8888	10
Signo 6	1.000	0.80	0.8888	10

Fig 21. Top 5 signos con peor F1-score (más errores)

Esto indica que los signos listados tienen mayor dificultad para el modelo en comparación con otras clases. Las razones detrás de esto pueden ser. Similitud entre los signos, lo que lleva al modelo a confundirlos durante la predicción. O variabilidad entre un mismo signo, es decir, diferentes formas de realizar el mismo signo que introduce ruido en el aprendizaje.

6.5 Validación realizando personalmente los signos

Ahora que el modelo ha sido entrenado y tenemos sus métricas de evaluación, he llevado a cabo pruebas prácticas en las que he ejecutado personalmente distintos signos pertenecientes al conjunto de entrenamiento. El objetivo de esta prueba es obtener una **primera aproximación del modelo en condiciones reales**, es decir, fuera del entorno controlado del dataset.

La validación ha consistido en reproducir los mismos signos que aparecen en el conjunto de datos, pero con algunas variaciones. En mi caso, la cámara se encontraba situada a una distancia de aproximadamente 1,60 m, mientras que en el dataset original estaba a 2,00 m. Además, utilicé un guante azul, aunque mi modelo permite la utilización de cualquier color de guante gracias al selector de color. Otro aspecto a destacar es que no contaba con un fondo uniforme o era azul (el mismo color que el guante), lo cual también introduce un grado mayor de variabilidad en las condiciones de prueba.

Para esta validación, he grabado un total de 5 videos por cada uno de los 64 signos, lo que da como resultado 320 videos de prueba como el que se puede observar en la **Figura 22**.

Fig 22. Condiciones de los videos realizados personalmente

Para evaluar el modelo con datos reales, he desarrollado una función que recorre carpetas con vídeos grabados por mí, identifica la clase esperada según el nombre de la carpeta, y predice la clase tras pre procesar los frames con el mismo pipeline usado en el entrenamiento.

Durante el entrenamiento, el modelo alcanzó una precisión del 97,52% sobre el conjunto de entrenamiento y del 96,56% sobre el de validación, lo que demuestra un alto rendimiento en un entorno controlado y homogéneo. Sin embargo, al evaluar el modelo con vídeos grabados por mí en condiciones distintas (distancia de cámara menor, guante derecho azul y fondo no uniforme), la **precisión** que obtuve fue del **73,48%**. No obstante, sí miramos en el estado del arte [15] se puede ver que oscilan entre 72% y 98%. Por lo tanto, el resultado es bastante aceptable.

Aunque este valor es inferior, sigue siendo un resultado positivo dadas las diferencias en las condiciones. El modelo muestra una capacidad de generalización razonable, y con técnicas adicionales como aumento de datos, su rendimiento en entornos reales podría mejorar significativamente.

7 CONCLUSION

El proyecto ha logrado construir un sistema completo para el reconocimiento de lengua de señas argentina (LSA), capaz de detectar y clasificar **64 signos dinámicos** a partir de videos. El código implementado abarca desde la captura y segmentación de las manos con guantes, hasta la generación

de un dataset estructurado y el entrenamiento de un modelo CNN para obtener las características de las manos + LSTM para capturar el movimiento en el tiempo.

Uno de los puntos fuertes del trabajo es la combinación de información espacial como son imágenes de las manos e información temporal como secuencia de posiciones a lo largo de los frames, lo cual permite al modelo aprender patrones complejos de movimiento. El modelo ha alcanzado un 99,16% de precisión en el conjunto de entrenamiento y un 93,75% en el conjunto de validación, mostrando un rendimiento muy bueno en datos no vistos.

Para validar su comportamiento en condiciones reales, introduciendo variaciones como una distancia de cámara menor (1,60 m), un guante azul en la mano derecha, y un fondo no uniforme. A pesar de estas diferencias con respecto al dataset original, el modelo ha logrado una precisión del 73,48%, lo cual es un resultado bueno y tiene una capacidad de generalización razonable. Esto demuestra que, incluso sin técnicas de aumento de datos adicionales, el sistema puede adaptarse a condiciones nuevas.

Las gráficas de precisión y pérdida confirman que el modelo aprende de forma progresiva y evita caer en un sobreajuste extremo, aunque sí se detecta cierta diferencia entre entrenamiento y validación que podría mitigarse en un futuro con técnicas como regularización, dropout adicional o aumento del dataset. Además, he desarrollado una aplicación web en local que permite cargar y procesar nuevos videos, mejorando la accesibilidad.

En resumen, el proyecto representa una aportación en el campo del reconocimiento automático de signos, demostrando que es posible implementar un sistema que combine visión por computadora, aprendizaje profundo y herramientas prácticas para facilitar la comunicación inclusiva.

Como mejoras futuras, el proyecto podría incluir:

- Ampliar el dataset con nuevos usuarios y condiciones de grabación.
- Probar modelos sin guantes, usando segmentación avanzada de manos o librerías como mediapipe que construyan el esqueleto de esta.
- Desplegar el modelo en tiempo real para aplicaciones prácticas, como traductores en vivo.

Este trabajo constituye una buena base para investigaciones futuras y aplicaciones reales en el área de traducción automática de lengua de señas.

8 Información

En la siguiente sección se encontrará toda la información de contacto, curso realizado:

- **E-mail de contacto**:
  Rubén Garcia Viciana (ruben.garciavi@autonoma.cat)
- **Mención realizada**: Computación
- **Trabajo tutorizado por**:
  Gemma Sanchez Albaladejo (gemma.sanchez@uab.cat)
- **Curso 2024/25**

9 Bibliografía

[1] LSA64: A Dataset for Argentinian Sign Language · Facundo Quiroga. https://facundoq.github.io/datasets/lsa64/ (Acceso: 19 Febrero 2025).

[2] Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective. (Acceso: 17 Febrero 2025). In arXiv:1908.08597v1 [cs.CV] 22 Aug 2019 [Journal-article].

[3] Kumar, R., Singh, S. K., Galgotias College of Engineering and Technology, Bajpai, A., & Sinha, A. (n.d.). A comparative analysis of techniques and algorithms for recognising sign language [Journal-article]. (Acceso: 18 Febrero 2025)

[4] SIGNUM Database. (Acceso: 19 Febrero 2025). https://www.bas.uni-muenchen.de/Bas/SIGNUM/

[5] RWTH-PHOENIX-Weather 2014: Continuous Sign Language Recognition Dataset. (Acceso: 19 Febrero 2025). https://www-i6.informatik.rwth-aachen.de/~koller/RWTH-PHOENIX/

[6] Admin. (2024, June 10). ¿La Lengua de Signos es universal? IELSE. (Acceso: 02 Marzo 2025). https://ielse.es/la-lengua-de-signos-es-universal/#:~:text=El%20SSI%20no%20es%20una,de%20diferentes%20partes%20del%20mundo.

[7] ¿Qué son las Redes LSTM? | Codificando Bits. (2024, February 23). Codificando Bits. (Acceso: 02 Maro 2025). https://codificandobits.com/blog/redes-lstm/

[8] Benimeli, E. (2011). Los colores en Informática: modelos RGB y HSV. In Los colores en Informática: modelos RGB y HSV (pp. 1–6). (Acceso: 10 Abril 2025) https://www.esferatic.com/wp-content/uploads/rgb-hsb-gimp.pdf

[9] Imagen signos estáticos (Acceso: 09 Abril 2025) https://media.istockphoto.com/id/1220106111/es/vector/lenguaje-de-manos-sordomudos-alfabeto-de-aprendizaje-comunicaci%C3%B3n-sordo-silencio-no-verbal.jpg?s=612x612&w=0&k=20&c=LMChJEh2sCRN6PlGj0Sb_3Ces2C4D26TQx5YDuo63WQ=

[10] Imagen signos dinámicos (Acceso: 09 Abril 2025) https://www.cervantesvirtual.com/s3/BVMC_OBRAS/ffb/eaf/868/2b1/11d/fac/c70/021/85c/e60/64/mimes/imagenes/ffbeaf86-82b1-11df-acc7-002185ce6064_251.jpg

[11] Great Learning. (2024, September 2). Face Detection using Viola Jones Algorithm. Great Learning Blog: Free Resources What Matters to Shape Your Career! (Acceso: 08 Abril 2025). https://www.mygreatlearning.com/blog/viola-jones-algorithm/

[12] Hands parent: MediaPipe Legacy Solutions nav_order: 4 — MediaPipe v0.7.5 documentation. (Acceso: 22 Mayo 2025). https://mediapipe.readthedocs.io/en/latest/solutions/hands.html

[13] StrongRay.. GitHub - StrongRay/Openpose-Hand-Detection: CMU Perpetual Computing Lab Openpose modifications to show only Hand detection. GitHub. (Acceso: 22 Mayo 2025). https://github.com/StrongRay/Openpose-Hand-Detection

[14] OpenCV: Optical Flow. (Acceso: 22 Mayo 2025). https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html

[15] Sign Language Recognition: A Deep Survey (Acceso: 24 Mayo 2025). https://www.sciencedirect.com/science/article/abs/pii/S095741742030614X

APÉNDICE

Fig 3. Diagrama de gannt sobre la planificación de tareas

En la siguiente imagen se muestra que tiempo dedicaré a cada una de las tareas para realizar el proyecto según las entregas que tengo que realizar.

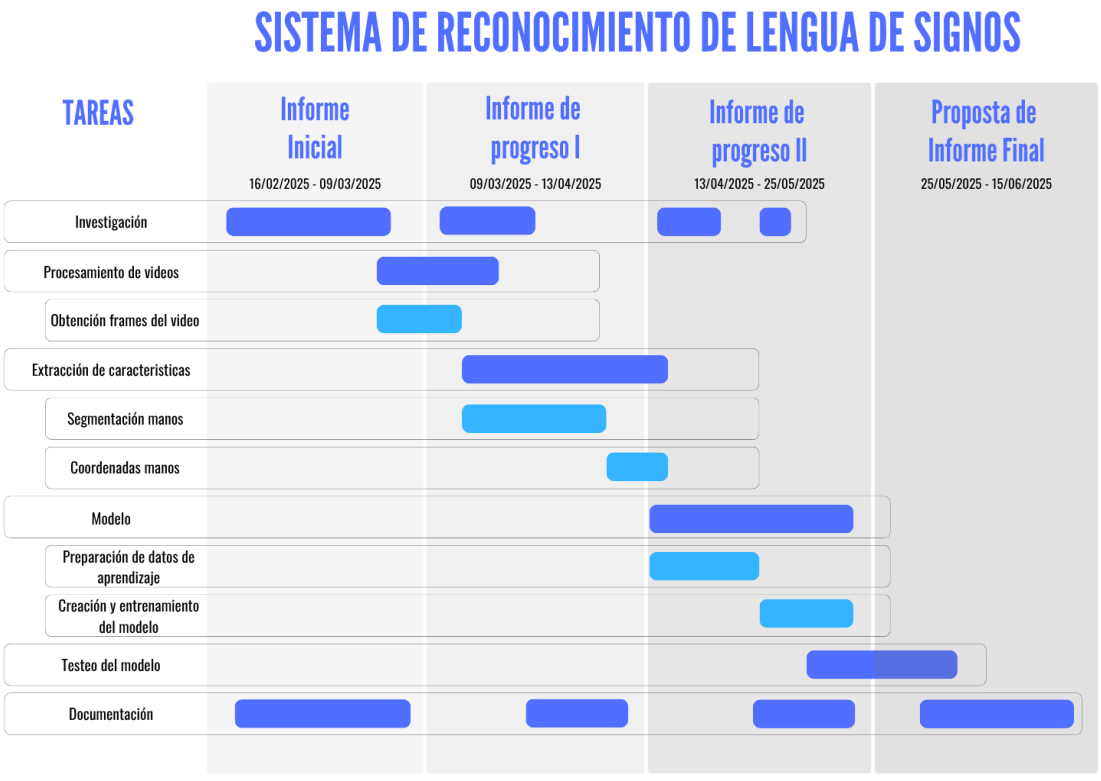

Fig 13. Esquema de creación de capas de imagen para entrenamiento

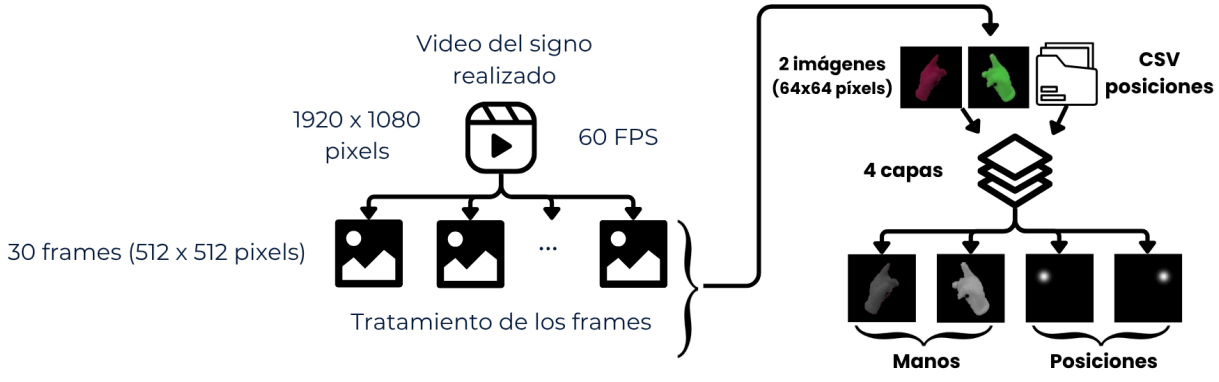

Fig 14. Gráfica del funcionamiento del modelo

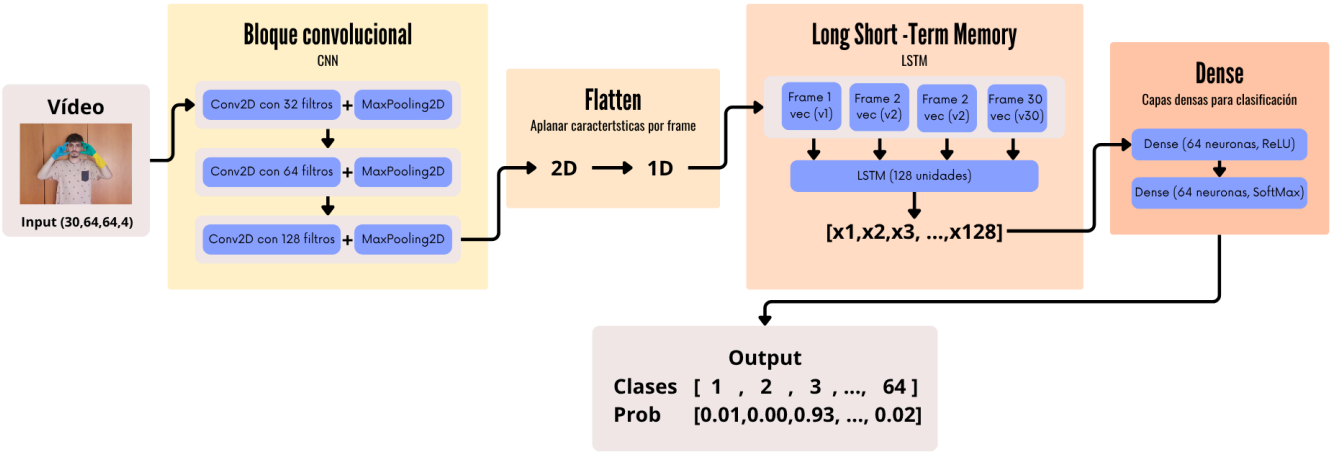

Fig 18. Gráfica de precisión por época

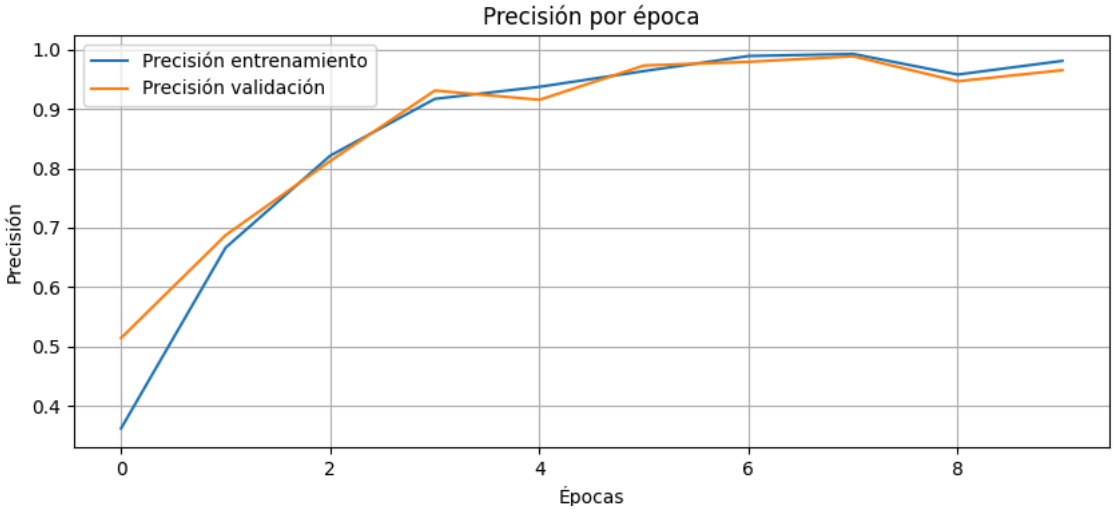

Fig 19. Gráfica de pérdida por época

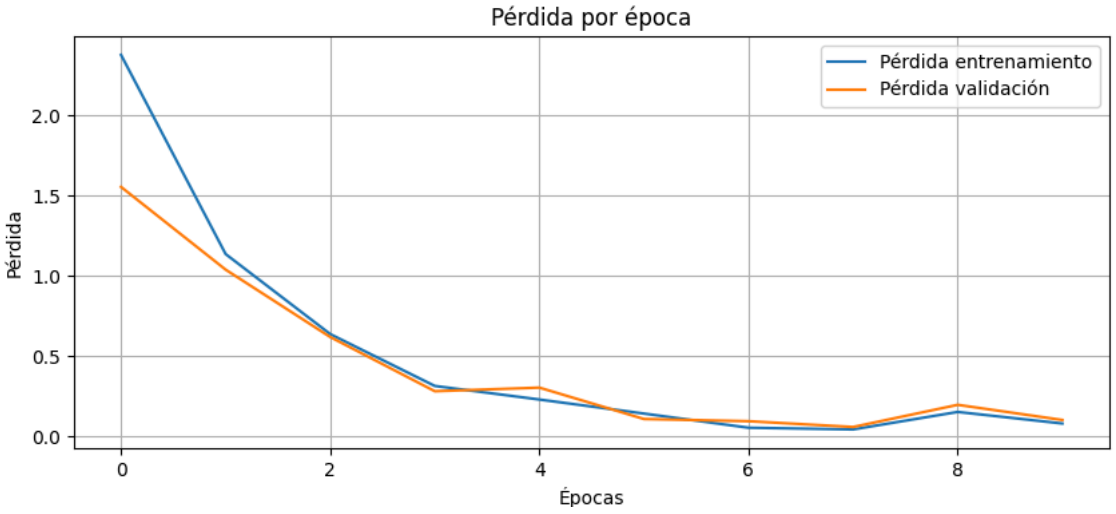

Fig 20. Matriz de confusión