

# Sign Language Recognition

Carlos Berea Barcia

**Resumen–** Este proyecto se centra en el reconocimiento de lenguaje de signos para permitir que una persona sordomuda y una persona que no conoce el lenguaje de signos se puedan comunicar sin ningún tipo de problema. Esto se puede llegar a conseguir ,primero entrenando un modelo con una red neuronal que permita reconocer las diferentes letras del lenguaje de signos y después creando una aplicación para dispositivos móviles con el modelo previamente entrenado, que permita que las personas se puedan comunicar en cualquier lugar con el lenguaje de signos, o incluso puede llegar a servir para aprender lenguaje de signos uno mismo. En este caso la aplicación reconoce el ASL (American Sign Language), ya que fue el lenguaje de signos escogido para entrenar la red neuronal, está programada en Flutter por lo tanto es una aplicación multiplataforma, es decir puede utilizarse tanto en Android como en iOS. Esto permitirá que cualquier usuario podrá utilizar esta app independiente del sistema operativo del mismo.

**Paraules clau–** Reconocimiento de lenguaje de signos, red neuronal, American Sign Language, ASL, aplicación móvil, Flutter, Android, iOS.

**Abstract–** This project focuses on the sign language recognition to allow a deaf person and a person who doesn't know sign language to communicate without any problem. This can be achieved, first by training a model with a neural network that allows recognizing the different letters of sign language and then creating a mobile app with the previously established model, which allows people to communicate anywhere using sign language, or it can even be used to learn sign language on your own. In this case, the app recognizes ASL (American Sign Language), because it was the sign language chosen to train the neural network. It is programmed in Flutter, therefore it is a multiplatform app, that can be used on both Android and iOS . This will allow any user to use this application regardless of their operating system.

**Keywords–** Sign Language recognition, neural network, American Sign Language, ASL, mobile app, Flutter, Android, iOS.

## 1 INTRODUCCIÓN

ESTE es un proyecto que fue motivado por la idea de conseguir que las personas sordomudas puedan comunicarse con mayor facilidad, ya que no muchas personas conocen el lenguaje de signos, también fue una propuesta que fue ideada a raíz de la imposibilidad de poder seguir adelante con el otro proyecto de TFG, Speaker Recognition, debido a una problema al no tener la capacidad computacional requerida para llevar a cabo este proyecto y por lo tanto llegar a un punto en el que no se podía avanzar más.

El proyecto de Speaker Recognition tenía como objetivo conseguir un programa que pudiese reconocer la voz de la persona que estaba hablando en audios, se consiguió avanzar con el tema de búsqueda de la mejor red neuronal para poder llevar a cabo el proyecto, al igual que crear un código con esa red escogida al igual que las funciones de activación, de optimización y de loss correspondientes, también estaba escogido el dataset y se habían probado modelos pre entrenados con este, pero como se ha comentado antes por problemas computacionales y llegar a un punto en el proyecto en el que me veía bloqueado y con imposibilidad de avanzar decidí cambiar el tema del proyecto a poco más de un mes para terminar el plazo.

El proyecto podríamos decir que tiene dos partes una en la cual se preparan los datos, se entrena el modelo a través de la red neuronal y se sacan conclusiones de los datos. Y por la otra parte tendríamos la creación de la aplicación, en la cuál podemos reconocer las letras del ASL gracias al modelo entrenado y pasado a un formato especial para poder ser utilizado con el móvil.

• E-mail de contacto: carlos.berea@ub.cat  
• Mención realizada: Computación  
• Trabajo tutorizado por: Ramón Baldrich (Departamento de las Ciencias de la Computación, UAB)  
• Curso 2021/22

## 2 OBJETIVOS Y PLANIFICACIÓN

Viendo que esta podría ser una muy buena herramienta para ayudar en la comunicación de la gente sordomuda el objetivo principal de este proyecto sería conseguir una aplicación funcional que pudiese ayudarlos al tratar de comunicarse con otras personas. Esto pude lograrse gracias a las redes neuronal y el entrenamiento de modelos con ellas y estos modelos se pueden pasar a teléfonos móviles con la ayuda del Tflite que permite crear un modelo compatible con la tecnología de los teléfonos actuales.

Pasar este proyecto a teléfonos móviles en lugar de utilizar el ordenador directamente es debido a que este proyecto se pensó para poder ser una ayuda en cualquier lugar y situación. Se ha escogido el ASL ya que al ir por letras sería mucho más fácil para el usuario poder llegar a aprenderlo en el caso de querer utilizar la app como método de aprendizaje del mismo.

Este objetivo de que la app sea totalmente funcional entran varios apartados, debido a que hay diferentes funcionalidades que la app debe tener. Entre ellas tenemos:

- El reconocimiento de lenguaje de signos en tiempo que permitiría a la persona grabar los signos que esta realizando la persona con la que esta hablando y para que la app los pase a las letras correspondientes para poder entenderlo.
- La posibilidad de utilizar imágenes que tengamos en la galería para poder saber que signo es el que se está representando.
- Reconocimiento de signos a partir de foto con la cámara, en el caso de que no te funcione la opción de vídeo del móvil o que quieras sacarle una foto a algo estático para saber cual es el signo que está representando.
- Poder poner un texto en el teléfono móvil y que te cree una secuencia de imágenes con los diferentes signos a los que corresponde cada letra.

Estos serían los objetivos finales con respecto a la app, pero hay otro objetivo previo muy importante que es el de conseguir un modelo bien entrenado que pueda llegar a reconocer con facilidad todos estos signos para después poder llevar este modelo a formato app, que sin él todo lo que tiene que ver con la app no tendría el mismo valor y no se podría utilizar tan bien como se querría.

Para poder llegar a cumplir los objetivos correspondientes correctamente necesitamos una buena planificación. Como hemos comentado anteriormente el proyecto esta dividiendo en dos fases muy bien marcadas, en las cuales dentro de ellas hay muchas tareas para poder llegar al objetivo final de cada una de las dos fases. Estas dos fases son las siguientes:

- Fase de desarrollo del modelo: En esta fase se han realizado todas las tareas correspondientes a la creación

del modelo para poder predecir el signo que estaba haciendo una persona o que aparece en una imagen. Esta se divide en diferentes tareas como buscar el mejor dataset para este proyecto, escoger una buena red neuronal, hacer un data augmentation y hacer el código para crear el modelo.

- Fase de desarrollo de la app: En esta fase se ha desarrollado la app con las diferentes funcionalidades que eran esenciales para que cumpliese con el objetivo de que fuese de ayuda tanto para saber que dice una persona al hacer lenguaje de signos como para predecirlo de una imagen haciendo un signo.

## 3 METODOLOGÍA

Para llevar a cabo este proyecto y poder realizar las tareas de la forma más eficiente posible, he decidido utilizar la metodología Agile. Esto me permite mayor flexibilidad a la hora de realizar el proyecto y llevar un buen control de las tareas.

Dentro de la metodología Agile se encuentran varias metodologías entre ellas esta la Scrum que es la cual he utilizado durante la primera etapa antes de cambiar al tema del proyecto actual en la cual hacia sprints semanales con unos objetivos previamente marcados y divididos en subtareas.

Cuando decidí cambiar el tema al proyecto actual de reconocimiento de lenguaje de signos, debido al poco tiempo de para poder desarrollar que tenía cambié la metodología de trabajo a Kanban. Esta me parece mucho más apropiada para la situación en la que estaba debido a que necesitaba hacer en un día tareas que de tener más tiempo las haría en sprints de una semana, por lo tanto funcionaría mucho mejor una metodología que funcione por tickets como es Kanban, porque me permite estructurar mejor las tareas y me da mayor flexibilidad ya que las entregas serían continuas, cada día tenía muchas tareas por hacer.

Para Kanban he utilizado Trello en el cual tenía un tablero dividido en cuatro columnas, las cuales eran TO DO, DO TODAY, DOING y DONE, esto me servía para poder tener una visión de como iba avanzando, las cosas que tenía que hacer ese día y las que me quedaban por hacer. En cada una de estas columnas tenía los tickets, los cuales eran pequeñas tareas a realizar para poder tener mayor flexibilidad a la hora de tomar decisiones.

Para validar que las tareas realmente estaban hechas tenía reuniones con mi tutor que al principio eran al finalizar cada sprint cuando hacía la metodología Scrum con el anterior proyecto pero esto cambio cuando pase a Kanban. En Kanban necesitaba reuniones más continuas para poder validar las tareas y organizar tareas nuevas, debido a que el proyecto avanzaba mucho más rápido y se hacían muchas tareas cada día, por lo tanto se realizaban varias reuniones a la semana. Estas reuniones se realizaban por Microsoft Teams, en ellas como he comentado anteriormente se validaban las tareas realizadas y se organizaban nuevas tareas para los próximos días y se hacía un estudio de hasta donde

se podía llegar con este proyecto viendo lo que quedaba por hacer y la estimación de tiempo que se le daba.

## 4 ESTADO DEL ARTE

En la actualidad según la Federación Mundial de sordos existen aproximadamente 70 millones de personas sordas en el mundo y los cuales utilizan más de 300 diferentes lenguas de señas [4], pero si sumamos las personas que tienen problemas de audición, según la ONU, la cifra asciende a más de 466 millones de personas. Por lo tanto cobra una importancia muy grande en la actualidad el lenguaje de signos. En la siguiente imagen podemos apreciar la cantidad de personas sordas que hay en Europa:

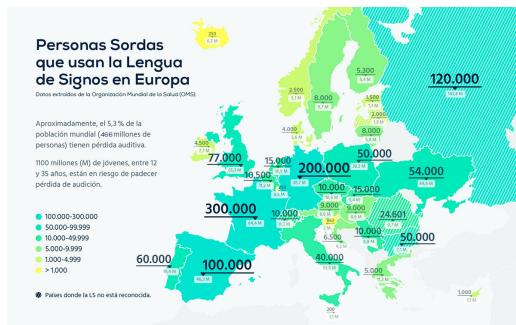


Fig. 1: Mapa representativo de la cantidad de sordos que hay en Europa.

Todo esto hablando únicamente de personas completamente sordas, también hay mucha gente con grandes problemas de audición por eso es tan importante el lenguaje de signos en nuestra sociedad. Un claro caso de que el lenguaje de señas ha cobrado una gran importancia en la actualidad es que la convención sobre los derechos de las personas con discapacidad reconoce y promueve el uso de las lenguas de señas y establece que tiene el mismo estatus que las lenguas habladas y obliga a los estados y obliga a los estados a que faciliten el aprendizaje de la lengua de señas. Incluso hay un día internacional el cual es el 23 de septiembre. De todos los lenguajes de signos cabría destacar el ASL (American Sign Language) ya que es uno de los más importantes a nivel mundial.

En la actualidad hay algunas apps que ayudan a aprender lenguaje de signos, como Hand Talk Tradutor para Libras el cuál permite pasar texto y audios a lenguaje de señas de Brasil y al ASL. Por otra parte empresas muy conocidas como Google están trabajando en que su famoso traductor de Google también sea capaz de traducir lenguaje de signos [7]. En el caso de la primera app se centra más en el aprendizaje, mientras que en el caso de google esta más enfocada a saber que es lo que está queriendo decir las señas realizadas por los individuos.

A parte de estos proyectos de aplicaciones hay muchos otros proyectos de código libre en internet los cuales utilizan diferentes datasets y redes neuronales. En la plataforma Kaggle hay muchos de estos ejemplos por ejemplo utilizando un dataset de ASL, lenguaje del cual hemos hablado anteriormente, hay muchos códigos ejemplo con diferen-

tes soluciones al problema de conseguir entrenar el mejor modelo posible para conseguir los mejores resultados posibles al pasarle imágenes de manos haciendo señas. Entre ellos hay proyectos muy interesantes de los cuales se pueden coger ideas para poder realizar un proyecto propio, ya que dentro de estos aparecen todos los resultados obtenidos y se pueden sacar conclusiones de cual es el más apropiado. Entre ellos podemos ver la utilización de redes neuronales entrenados utilizando tanto keras como pytorch y diferentes redes neuronales como CNN o InceptionResNetV2.

## 5 DESARROLLO

A continuación en este apartado se describirán todos los métodos utilizados en el proyecto y cómo se han llevado a cabo cada una de las tareas dentro del proyecto, dividido como vimos anteriormente en dos fases la primera de desarrollo del modelo y la segunda de desarrollo de la app.

### 5.1. Creación del modelo

Esta fase dentro del desarrollo del proyecto se dividirá en tres partes, la primera de ellas se hablará sobre el dataset escogido finalmente y las pruebas previas que se hicieron con otros dataset hasta llegar a este, en la segunda parte hablaremos sobre el data augmentation que se le ha realizado al mismo y porque se ha hecho así y por último sobre la red neuronal utilizada y la creación del modelo.

#### 5.1.1. Red neuronal

Una de las primeras cosas que hay que tener en cuenta a la hora de crear el modelo es escoger la mejor red neuronal posible para el proyecto. Como ya comenté anteriormente para este tipo de proyectos se utilizan diversas redes neuronales. He llegado a probar dos, una más sencilla la cual era una red neuronal secuencial sencilla con 4 capas Dense y que utiliza la función de activación relu ya que da buenos resultados, la otra que iba a ser la red neuronal que iba a utilizar para el proyecto es una CNN.

La CNN que he decidido utilizar en el proyecto es una red neuronal cuya estructura la he recogido de uno de los ejemplos de proyectos que os he comentado anteriormente, de los cuales se podían sacar ideas muy interesantes que se pueden aplicar al proyecto. Esta red neuronal cuenta con una arquitectura muy similar a la arquitectura VGG, la cual es una arquitectura de red neuronal que se asemeja al comportamiento del cerebro humano, pero un poco más sencilla. La arquitectura que utilizamos en el proyecto es muy parecida a la que se puede apreciar en la figura(2), la diferencia es que en el caso de la que se utiliza en este proyecto se realiza 2 convoluciones y max pooling 3 veces y se termina con 2 convoluciones más, flatten y dos dense uno con relu y el último softmax.

#### 5.1.2. ASL Dataset

Como se puede apreciar en el título de la sección y que ya se mencionó en la introducción, el ASL (American Sign Language) es el lenguaje de signos escogido para realizar este proyecto y por lo tanto el que se va a utilizar tanto

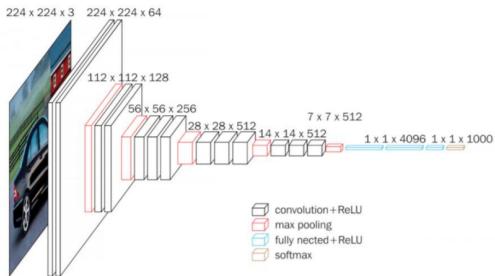


Fig. 2: Imagen Ejemplo de una arquitectura VGG, muy parecida a la estructura utilizada en la red neuronal del proyecto.

en el dataset como en las pruebas que se van a realizar. Se ha escogido por varias razones la primera es que es utilizado en muchas partes del mundo y aunque no hay una encuesta fiable se estima que es utilizado por entre 500.000 y 2.000.000 de personas en los estados unidos, la segunda razón es que es un lenguaje de signos fácil de aprender ya que son estáticos y con una sola imagen ya puedes saber el signo que está utilizando, mientras que en otros lenguajes de signos se necesitan videos para poder saber lo que se está queriendo decir y es mucho más complejo de aprender para una persona que es la primera vez que se propone aprender un lenguaje de signos.

El ASL consta de un total de 26 signos correspondientes a cada una de las letras del alfabeto, como podemos ver en la figura(3) esto nos permite formar palabras juntando los diferentes símbolos correspondientes a cada una de las letras que lleva la palabra. Aparte de esta existen otras lenguas de signos como la BSL (British Sign Language) que podemos ver en la figura(4), este a diferencia del ASL se realiza con dos manos, por lo tanto es mucho más sencillo el ASL y también más utilizado en todo el mundo.

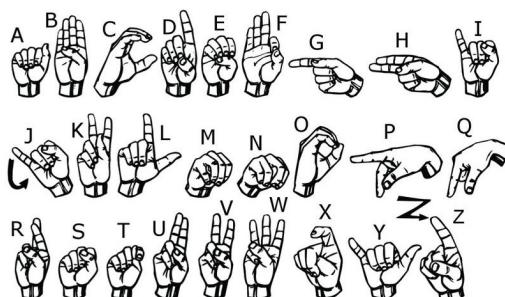


Fig. 3: Letras del alfabeto ASL (American Sign Language).

Una vez escogido el lenguaje de signos que se iba a utilizar pasamos a la fase de escoger el dataset más indicado para este proyecto. Para poder tener una idea aproximada de la magnitud a la que podía llegar este proyecto primero se decidió hacer pruebas con datasets más pequeños. El primer dataset que se escogió fue el Sign Language MNIST [1] debido a que era un dataset pequeño de un total de 27455 casos de train, el cual está compuesto por un csv en el que están guardados todos los datos de los pixeles de las imágenes, es decir este dataset no tiene las imágenes como tal si no un csv con columnas que correspondientes

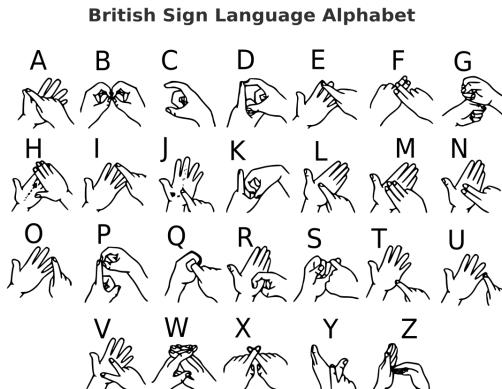


Fig. 4: Letras del alfabeto BSL (British Sign Language).

a los valores del label de cada imagen y a cada uno de los píxeles desde el píxel 1 hasta el 784, ya que son imágenes de 28x28 píxeles, estas imágenes las podemos apreciar en la figura(5). Cada una de las celdas de los labels tienen los valores de las letras correspondientes a cada una de las imágenes del dataset y las celdas de las columnas de los píxeles contienen valores del 0 al 255.



Fig. 5: Imágenes del dataset de Sign Language MNIST, este dataset está compuesto por imágenes de 28x28 píxeles.

Como eran imágenes muy pequeñas y la primera red neuronal que probé era muy simple el ordenador no tardaba prácticamente nada en sacar los resultados, pero los resultados no eran todo lo buenos que cabía esperar para un sign language recognition ya que al ser tan pequeñas las imágenes se confundían entre muchas de las letras. Estos resultados se pueden apreciar a la hora de la validación del test con el modelo como podemos ver en la figura (6), en la cual el accuracy de la validación únicamente da un 0.81. Un resultado muy malo ya que posteriormente a pasarlo a vídeo o mandarle fotos con las cuales no ha entrenado le va a ser muy complicado poder predecir el resultado correctamente.

Con algunos cambios que se hicieron como cambiar la red neuronal a una convolutional neural network y hacer cambios a la hora de procesar cada uno de los frames de la cámara, cuando se hacía una predicción por video, se mejoraron los resultados pero no todo lo que se esperaba.

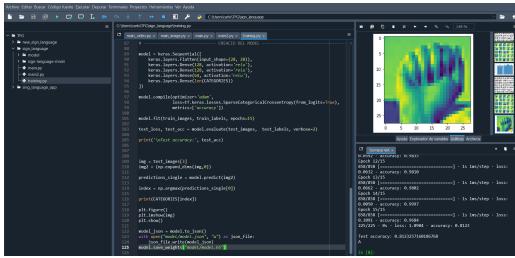


Fig. 6: Resultado de la validación del dataset Sign Language MNIST

Algunos de los procesos que se le hicieron a cada frame fueron por ejemplo dentro del frame detectar la mano por el color aproximado de la misma, una vez encontrada se recortaba la imagen a únicamente la parte en la que aparece la mano y se aplica una máscara para que únicamente se centre en la mano y no en el resto de elementos. En la figura(7) podemos ver una de las pruebas hechas para ver la capacidad de predicción en vídeo con la cámara del portátil del modelo creado con el dataset MNIST. Como se puede apreciar en al parte derecha de la imagen tenemos un frame pequeño de la mano recortado por color de piel, pero aún aplicando el recorte de la mano y su posterior cambio de tamaño a 28x28 el resultado de la predicción no es el correcto ya que se representa una A y predice que es una X. Una vez se llegó al punto en el cual no se podía avanzar

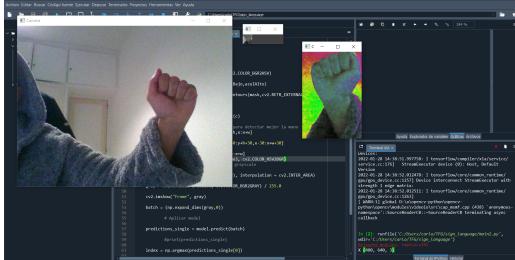


Fig. 7: Resultado de predicción de signos por vídeo con Sign Language MNIST.

más con este dataset se decidió cambiar al dataset actual el ASL alphabet [2] el cual es un dataset mucho mas grande y completo. Este dataset consta de un total de 87.000 imágenes con 29 clases, esto es debido a que aparte de las 26 letras del alfabeto también tiene tres clases que son space, delete y nothing. Del total de imágenes 3.000 corresponden a cada una de las clases, cada una de las imágenes de este dataset corresponden a frames de un vídeo de una persona haciendo cada una de las letras del alfabeto en lenguaje de signos y son imágenes de 200 x 200. En la figura (8) podemos apreciar 6 imágenes del dataset ASL alphabet, las cuales representan 6 de las 26 letras que contiene este alfabeto, como vemos son de mucha mejor calidad que las del dataset anterior MNIST.

El hecho de que sean imágenes tan parecidas entre ellas con casi el mismo fondo y la misma posición entre ellas hace que los resultados una vez entrenado el modelo no sean los mejores que se podrían esperar a la hora de comprobarlo en un vídeo ya que el fondo cambiaba y la mano era diferente. Se probaron diferentes opciones como recortar la mano de la imagen y utilizar una máscara para que se cen-



Fig. 8: Imágenes del ASL alphabet

tre únicamente en la mano pero no mejoraba los resultados. Viendo los resultados obtenidos se decidió hacer un data augmentation del cuál se hablará en la próxima sección en más profundidad.

### 5.1.3. Data Augmentation

Como se comentó en el apartado anterior se decidió hacer un data augmentation para así poder conseguir mejores resultados que los que se estaban consiguiendo actualmente. Para este data augmentation las dos partes más importantes fueron el código para detectar la mano y eliminar el fondo de la imagen y el código que se encargaba de juntar ambas imágenes y guardar las nuevas imágenes creadas a partir del fondo nuevo y la mano recortada en las carpetas correspondientes.

En este caso para las nuevas imágenes de fondos decidí hacer 4 fotos a diferentes partes de mi casa, como podemos apreciar en la figura(9) y coger también un fondo de un salón de internet, es decir en total 5 fotos para crear nuevas imágenes con diferentes fondos a los actuales, estos fondos al sacar la foto no tenían el tamaño apropiado por lo tanto había que cambiarles el tamaño a 200x200. De estos fondos solo 4 serán utilizados para entrenar el modelo, ya que uno de los fondos se dejará únicamente para el test para ver como se comporta la red neuronal con un fondo desconocido.

Estas imágenes las nombré como BG1 y así sucesivamente cambiando el número, cogía uno a uno los fondos nuevo y lo unía a la imagen de la mano con el fondo ya eliminado después de aplicar el detector de mano, estas imágenes nuevas las guardaba con el nombre de la imagen original más un número correspondiente al fondo que tenía cada una de ellas para así poder diferenciarlas fácilmente. Esto hizo que los resultados mejorasen considerablemente, de lo que hablaremos en la sección correspondiente a la implementación de la red neuronal. Todo este proceso que he explicado de creación de la nueva imagen a través de unir el fondo y la imagen de la mano recortada viene representado en la figura(10).



Fig. 9: Fondos utilizados en el data augmentation para crear las imágenes nuevas con fondos diferentes al original.

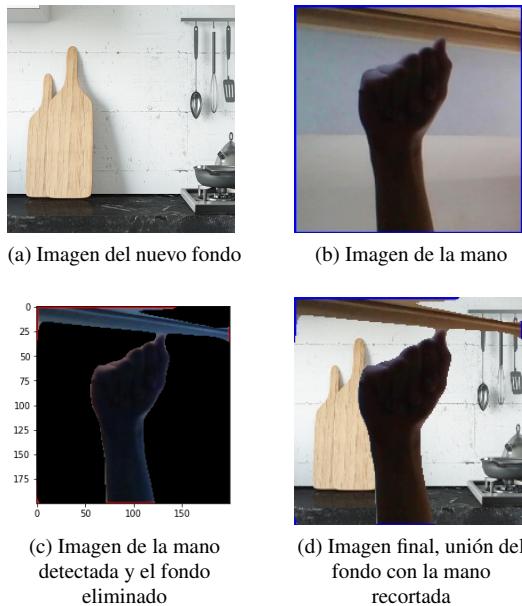


Fig. 10: Proceso de creación de las nuevas imágenes del dataset al realizar data augmentation

## 5.2. Aplicación móvil

Una vez terminada la parte de la creación del modelo para poder predecir correctamente los diferentes signos de ASL, pasamos a la creación de la parte más cercana al usuario que es la aplicación móvil, ya que es la que utilizará para poder utilizarlo en cualquier lugar.

Como se ha dicho anteriormente se decidió realizar la aplicación ya que realmente si lo que se quiere llegar a lograr es tener una herramienta que le pueda servir al usuario para poder comunicar se con otras personas con lenguaje de signos, en el caso de únicamente tener el modelo en el ordenador y solo poder utilizarlo ahí no serviría de mucho realmente porque solo lo podrías utilizar en tu casa. Para al parte de aprendizaje el hecho de tenerlo en el ordenador podría ser suficiente pero para comunicarse en la calle no, por eso se ha visto necesario el desarrollo de esta aplicación.

Lo primero que hay que decidir en estos casos es que tecnología utilizar, en mi caso decidí utilizar el framework Flutter que utiliza el lenguaje Dart. Esta elección fue por varias razones entre ellas esta el hecho más importante y es que Flutter te permite realizar con un solo código una aplicación tanto para Android como para iOS y esto te permite llegar a muchos más usuarios sin la necesidad de hacer dos aplicaciones nativas, lo cual sería mucho más costoso, aparte Flutter es muy nuevo pero evoluciona muy rápidamente ya que es de Google y están apostando mucho por el por lo tanto hay cada vez más facilidades para poder programar en este lenguaje y más documentación. Por otra parte también elegí Flutter porque estoy muy familiarizado con el Dart por lo que me sería mucho más fácil desarrollar la app.

Esta aplicación se decidió hacer en un principio en inglés pero posteriormente se quiere hacer también en español, esto es porque el ASL es un lenguaje de signos que se habla predominantemente en estados unidos donde el idioma es el inglés. En la figura(11) podemos apreciar un pantallazo de como es el diseño y distribución de la pantalla de elección en la aplicación móvil.

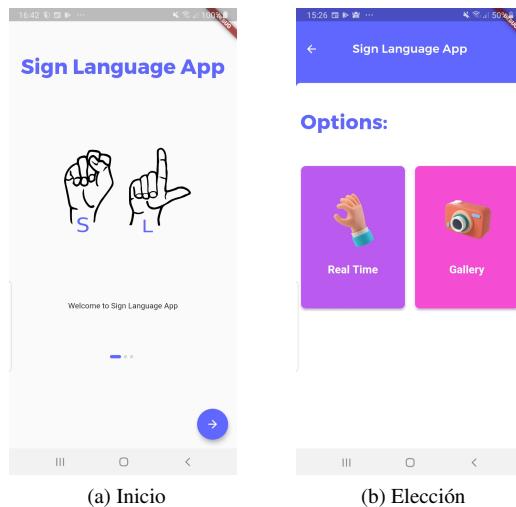


Fig. 11: Pantallas ejemplo de la app

Una vez terminado la elección del lenguaje se pasa a la estructuración de la aplicación, con un estudio de las diferentes partes que debe de tener la misma. Se llegó a la conclusión que deberían de haber dos partes diferenciadas una que sería enfocada a la parte más didáctica en la cual se aprendería lenguaje de signos de dos maneras diferentes y otra en la cual lo que prima es la comunicación entre las

personas que están dentro de la conversación y que utilizan la app para poder entenderse.

La parte de aprendizaje se divide en dos, por una parte la cual escoges una imagen de la galería y predices con el modelo previamente entrenado el signo que se ve en la imagen y otro apartado en el cual lo que puedes hacer es sacar una foto, para esas ocasiones en las cuales no puedes grabar un vídeo o quieras sacarle una foto a algún signo. En la parte de comunicación tenemos la parte en la cual grabando a la persona que hace los signos se predice lo que quieren decir cada uno de los signos que hace y por otro lado esta la sección en la que se escribe una palabra y aparecen por pantalla las imágenes con los signos que corresponden a cada una de las letras de esa palabra.

Para poder llegar a este punto en el cual todo esto pueda funcionar previamente se necesita pasar el modelo creado a un formato el cual sea legible en Dart y por lo tanto se pueda utilizar también en teléfonos móviles. Para esto investigue y encontré TFLite, el cual es una herramienta de tensorflow que te permite pasar tu modelo a un formato más óptimo para teléfonos móviles, por lo tanto modifique el código de la creación del modelo para poder crear el nuevo modelo para móviles.

Una vez terminada la creación del nuevo modelo se pasa a programar la nueva app ya que tengo todo lo necesario para poder llevar a cabo el mismo. La aplicación está dividida en un total de cinco pantallas una primera pantalla en la cual se enseñan todas las opciones que te brinda la app, sería algo así como una introducción a la aplicación, después pasariamos a la pantalla en la cual aparecen las diferentes opciones que tienes dentro de la app tanto la parte de aprendizaje como la de reconocimiento en tiempo real, la tercera pantalla es la cámara con la que haremos el reconocimiento en tiempo real, la cuarta pantalla es en la que escribes un texto y aparecen la secuencia de imágenes con las diferentes letras y la quinta pantalla es en la que se hace la parte de aprendizaje tanto en la que seleccionas una imagen de la galería como en la que sacas una foto.

### 5.2.1. Aprendizaje de lenguaje de signos

En este apartado de la aplicación como ya expliqué anteriormente tenemos dos partes la de escoger imágenes de la galería y la de sacar una foto al signo que queremos reconocer, como este lenguaje es tan sencillo con esta metodología una persona que se inicia en el lenguaje de signos lo tendrá muy fácil para aprender. También a parte de para aprender se puede utilizar para hacer consultas sobre que signo es el que se está utilizando en la imagen que tienes guardada.

Se ha decidido poner en la misma pantalla para que sea más sencilla para el usuario, en esta pantalla únicamente tendríamos dos botones , un botón el cual abre la cámara para poder sacar la foto y el otro botón que lo que hace es abrir la galería para escoger la imagen con la cual quieras realizar la predicción, y una imagen en la cual al principio sale una imagen predeterminada y que cuando seleccionas una imagen o sacas una foto se cambiará y debajo de la

misma aparecerá la predicción que se ha realizado.

### 5.2.2. Predicción en tiempo real

Este es el apartado más importante de la aplicación, porque es la cual te permite comunicarte de una manera más fluida con las personas sordomudas. En esta parte de la app una vez en la pantalla principal seleccionas esta opción, aparece un dialogo para que puedas elegir entre pasar de texto a imágenes con los signos del texto o la predicción en tiempo real de los signos a letras. Todo de la forma más vistosa y sencilla para el usuario.

En el caso de texto a imágenes de los signos tendríamos una pantalla con un espacio para que pongas el texto que quieras que pase a lenguaje de signos, este texto preferiblemente mejor que sean palabras para que sea más fácil de comprender para el usuario que lo está viendo. Al principio aparecerá una imagen predeterminada la cual se cambiará una vez pongas la palabra a predecir y empezarán a aparecer imágenes con los signos correspondientes a cada letra por pantalla cada tres segundos para que le de tiempo al usuario de visualizarlas.

En el caso de la predicción en tiempo real lo único que aparecerá es la cámara abierta en la pantalla del usuario y mientras enfoca a la mano de la persona que está realizando los signos correspondientes saldrán arriba a la derecha las predicciones de lo que está queriendo representar con ese signo.

### 5.2.3. Organización interna de la app y package utilizados

Para conseguir una buena organización en la app y que todo funcione correctamente, se han dividido cada pantalla en carpetas una que sería la view y la otra logic, en la view tendríamos un archivo .dart que lo único que contienen son cosas referentes a la UI y a controlar los cambios de la misma en el caso que sea stateful gracias al bloc consumer. Después dentro de la carpeta logic tendríamos tres archivos uno freeze que es autogenerado, un cubit en el que esta todo lo referente a la lógica de la pantalla como puede ser en nuestro caso las funciones que realizan la predicción de la imagen o frame que se le pasa y su posterior gestión de los estados posteriores y el archivo state en el cual están todos los estados de la pantalla definidos. Gracias a esto podemos tener todo bien organizado y que funcione de una manera más fluida.

En cuanto a package importantes que se han utilizado en la app tenemos:

- Tflite: el cual es el encargado de utilizar el modelo que hemos entrenado para predecir el lenguaje de signos.
- Auto route: que permite gestionar los cambios entre pantallas de una forma mucho mas sencilla.
- Camera: Para poder hacer de una manera más sencilla toda la pantalla correspondiente a la predicción en directo con la cámara.
- Image Picker: Para poder escoger la imagen de la carpeta correspondiente de una forma más sencilla.

## 6 RESULTADOS

## 6.1. Resultados por ordenador

Como ya hemos comentado anteriormente todos los resultados que se van a ver a continuación se han sealizado con el dataset ASL alphabet y con una red neuronal CNN (Convolutional neural network), ya que esta fue la combinación que mejores resultados nos ha dado de las que hemos probado con anterioridad. Ahora se enseñarán los resultados obtenidos tanto en la versión en el ordenador como en la aplicación.

A continuación podremos ver los resultados de la creación del modelo con todas las configuraciones finales y el data augmentation realizado:

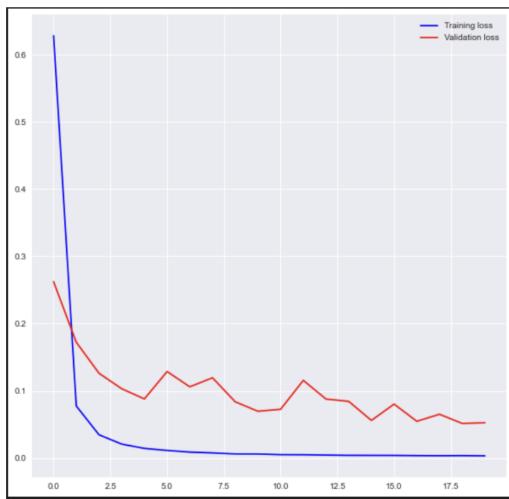


Fig. 12: Gráfica del loss al crear el modelo.

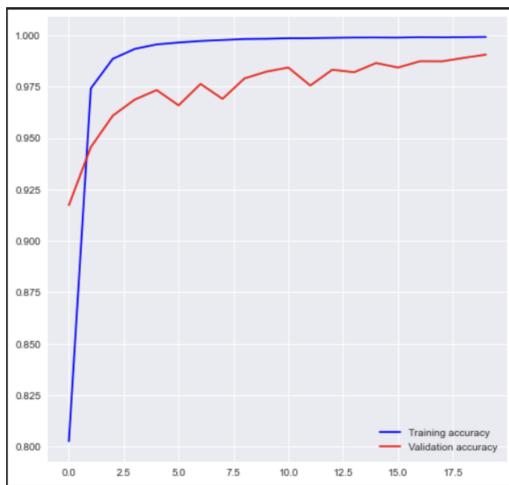


Fig. 13: Gráfica de accuracy al crear el modelo.

Como podemos apreciar en las figuras (13 y 12)en la parte de validation en loss y accuracy tenemos 0.0524 y 0.9906 en la última epoch y en el training un 0.0031 y 0.9992 respectivamente lo que son muy buenos resultados. Antes de llegar a estos resultados se hicieron muchas otras pruebas con cambios en el dataset las cuales antes de data augmentation daban mejores resultados pero estos no

## TAULA 1: RESULTADOS DE LOS MODELOS

Resultados validación		
Modelos	Accuracy	Loss
Con Data Augmentation	0.9906	0.0524
Sin Data Augmentation	0.2623	22.6503

eran del todo reales ya que eran imágenes prácticamente iguales.

Después de ver estos resultados podemos concluir que el hecho de hacer un data augmentation era muy necesario ya que como podemos ver los datos son muy buenos y más aún si los comparamos con los datos de validación de antes del data augmentation como podemos apreciar en la tabla(1), para sacar estos resultados hemos utilizado en ambos casos el mismo test de validación para que los resultados fuesen reales, ya que con su propio test el modelo antes del data augmentation daba unos resultados de validación muy buenos pero no era real ya que las imágenes eran muy parecidas y todas con los mismos fondos, mientras que en el nuevo dataset de imágenes de test habíai imágenes con diferentes fondos y por lo tanto era más realista y veríamos realmente como se comportaría el modelo.

Una vez creado el modelo y hecha la validación con sus gráficos correspondientes pasamos a la parte de predicción del modelo con imágenes y con vídeo. Primeo de todo fue crear un código para validar como se comporta al pasarle una imagen cualquiera del dataset que obviamente hacerto sin problemas como podemos ver en la figura(14), en el cual da un 81 % de probabilidad de que es esa la letra del signo.

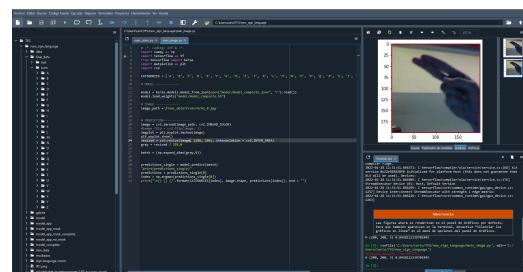


Fig. 14: Predicción de una foto del signo de la letra H.

Posteriormente después de realizar las pruebas de imágenes con varias fotos de signos diferentes se pasó al proceso de predicción por vídeo con la cámara del ordenador para ver si era capaz de predecir bien la letra a la que correspondía cada uno de los signos, para así posteriormente poder pasar el modelo a la app. En la figura(15) podemos apreciar que acierta la letra W con un buen resultado pero a veces hay problemas con algunas letras que se parecen entre ellas y le cuesta un poco más de predecir causando resultados peores en la probabilidad de que sea una letra u otra hasta en algunos casos en los que la mano no este del todo bien puesta fallar como podemos ver en la figura(15).

Que falle en un momento puntual no quiere decir que

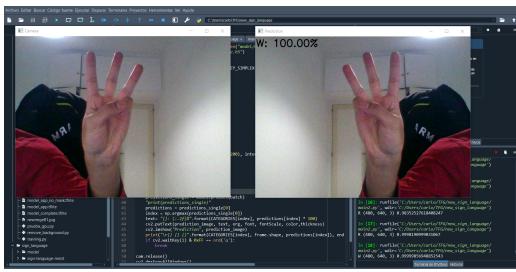


Fig. 15: Predicción en tiempo real del signo de la letra W con el ordenador.

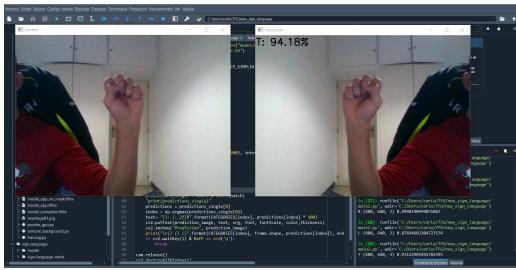


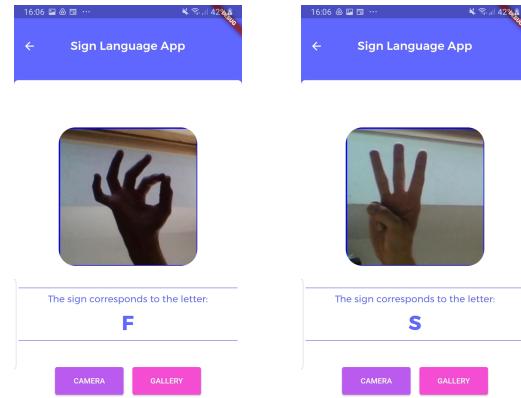
Fig. 16: Predicción fallida en tiempo real del signo de la letra S con el ordenador.

no pueda predecir ese signo ya que como se puede ver en mi github del proyecto en el apartado de resultados se ha llegado a acertar todos los signos.

## 6.2. Resultados por la aplicación

Después de ver que conseguía predecir todas las letras por vídeo viene la parte de revisar los resultados de la app, es decir ver como se comporta en los diferentes apartados que hemos creado dentro de la aplicación móvil, tanto la parte de predicción de imágenes como la de predicción por vídeo. En la figura(17) podemos ver un ejemplo de una foto escogida de la galería, en este caso es una imagen que ya ha utilizado el modelo como train por lo tanto debería predecirla correctamente sin ningún problema como se aprecia, pero eso no es del todo cierto, ya que como podemos ver en la otra imagen de la figura no se acierta la predicción ya que es una W y dice que es una S, todo esto teniendo en cuenta que es una imagen con la que ya ha entrenado.

En la sección de vídeo con la cámara del móvil para hacer predicción en tiempo real los resultados tampoco son los esperados ya que como podemos comprobar en la figura(18) no consigue acertar el signo que se está queriendo representar, esto lo podemos apreciar en la parte de arriba a la izquierda de la pantalla. Esto es un problema ya que no consigue realizar una buena predicción al pasar el modelo a versión tflite para móviles, por lo tanto las funcionalidades de predicción no se van a poder utilizar correctamente. A pesar de que el modelo en .h5 da muy buenos resultados en todos los tipos de pruebas que he realizado, al pasar el modelo a .tflite el accuracy baja muchísimo y no consigue predecir por lo tanto correctamente los signos.



(a) Resultado correcto de la predicción del signo F con la app.  
(b) Resultado fallido de la predicción del signo W con la app.

Fig. 17: Resultados de la app



Fig. 18: Predicción fallido en tiempo real de signo B.

## 7 AMPLIACIONES

En un futuro como posibles ampliaciones a este proyecto hay un abanico de posibilidades ya que se puede mejorar desde el punto de vista de entrenamiento del modelo hasta el apartado de la aplicación.

Si hubiese más tiempo para realizar este proyecto contemplo diferentes posibles mejoras dentro del apartado del dataset en primer lugar intentaría hacer un data augmentation aún mayor de mi dataset intentando poner imágenes propias más, posiblemente frames de un vídeo mío haciendo las diferentes letras dentro del American Sign Language, así se le añadiría mayor robustez al dataset ya que habría escenarios con manos de otras personas y no de la misma persona como hay actualmente. También en este data augmentation se le podrían aplicar a las imágenes con un poco de rotación, la rotación justa ya que si no habría signos que se podrían confundir con otros y eso sería contraproducente y también se les podría poner un poco de brillo a las imágenes, en resumen crear más

imágenes con diferencias entre ellas para hacer un dataset más robusto. Dentro del apartado del dataset también me gustaría probar a entrenar en lugar de imágenes del ASL poder entrenar vídeos de lenguaje de signos ya que es más común entre los sordomudos que el ASL, aunque mi capacidad computacional actual no se si me lo permitiría.

Dentro del apartado de la app mejoraría una sección que es la de la predicción en tiempo real, en esta sección en vez de mostrar simplemente la letra, como se está prediciendo actualmente, se podrían ir guardando los valores de cada letra y mostrar la palabra por pantalla y en caso de no se una palabra contemplada en el diccionario se darían diferentes opciones de posibles palabras a las que te quieras referir. En este caso se debería de restringir las palabras a un único idioma.

También se haría una investigación más a fondo y una mejora del modelo de predicción para la app móviles en formato tflite, para que este pueda predecir correctamente las letras de los signos que se le envían tanto en los frames de la predicción por vídeo en tiempo real como en las imágenes de la galería.

## 8 CONCLUSIONES

En conclusión en el caso de la parte del modelo en el ordenador se llegó a los resultados deseados ya que se consiguió una muy buena predicción de las diferentes letras dentro de los signos del ASL, todo esto consiguiendo que los predijese bien sin que hubiese un fondo completamente blanco detrás, por lo tanto por esa parte el objetivo se cumplió. También se ha conseguido hacer un buen dataset con el data augmentation como se ha podido comprobar en los resultados obtenidos antes y después de este data augmentation en la parte de validación del modelo.

Por contra en el caso de la aplicación en la parte de predicción en tiempo real no se ha conseguido lo deseado ya que el modelo pasado a TfLite no consigue los resultados esperados, al contrario que en la versión del modelo de ordenador este no predice bien las letras de los signos representados por pantalla. En este caso se necesitaría una mejora en la creación del modelo para conseguir un modelo totalmente funcional para la aplicación que ayude realmente en la comunicación con los sordomudos.

Por lo tanto hemos conseguido un código que puede ayudar a los sordomudos pero su paso a dispositivos móviles se ha visto truncado por la falta de precisión de los modelos en formato de Tensorflow Lite el cual da unos resultados que no pueden ser aprovechados para hacer una buena predicción en tiempo real con la cámara.

## AGRADECIMIENTOS

En primer lugar me gustaría agradecer a mi tutor de TFG Ramón Baldrich, por el apoyo mostrado cuando decidí cambiar de proyecto, su constante preocupación día a día y sus consejos. Quería agradecer también a mi familia y amigos

por el apoyo constante durante todo el proceso de desarrollar este proyecto.

## REFERENCIAS

- [1] Sign Language MNIST - tecperson, acceso 20 de Diciembre de 2021, <https://www.kaggle.com/datamunge/sign-language-mnist>
- [2] ASL alphabet - Akash, acceso 21 de Diciembre de 2021, <https://www.kaggle.com/grassknotted/asl-alphabet>
- [3] Sign language recognition datasets - Facundo Quiroga, acceso 21 de Diciembre de 2021, [http://facundoq.github.io/guides/sign\\_language\\_datasets/slrs](http://facundoq.github.io/guides/sign_language_datasets/slrs)
- [4] Día internacional de las lenguas de señas - Naciones Unidas, acceso 20 de Enero de 2022, <https://www.un.org/es/observances/sign-languages-day>
- [5] Deaf Employment Reports - Gallaudet University, acceso 25 de Enero de 2022, <https://www.gallaudet.edu/office-of-international-affairs/demographics/deaf-employment-reports>
- [6] Traductor de Google traducirá también lenguaje de señas - Juan Diego Polo, acceso 25 de Enero de 2022, <https://www.whatstnew.com/2020/01/29/traductor-de-google-traducira-tambien-lenguaje-de-senas/>
- [7] Sign Language Recognition for Computer Vision Enthusiasts - Vaishnavi Patil , acceso 22 de Diciembre de 2021, <https://www.analyticsvidhya.com/blog/2021/06/sign-language-recognition-for-computer-vision-enthusiasts/>
- [8] ASL recognition keras CNN - Yasin Soylu, acceso 24 de Diciembre de 2021, <https://www.kaggle.com/yasinsoylu123/asl-recognition-keras-cnn>
- [9] ASL alphabet — InceptionResNetV2 - Tuhin Kumar Dutta, acceso 24 de Diciembre de 2021, <https://www.kaggle.com/tuhinkumardutta/asl-alphabet-inceptionresnetv2-95-accuracy>
- [10] Hand Detection - Rishabh Chauhan, acceso 15 de Enero de 2022, <https://github.com/rishabh32/hand-detection>
- [11] Adding (blending) two images using OpenCV - OpenCV, acceso 16 de Enero de 2022, [https://docs.opencv.org/3.4/d5/dc4/tutorial\\_l\\_adding\\_images.html](https://docs.opencv.org/3.4/d5/dc4/tutorial_l_adding_images.html)
- [12] Flutter plugin for TensorFlow Lite - Shaqian, acceso 10 de Enero de 2022, <https://pub.dev/packages/tflite>