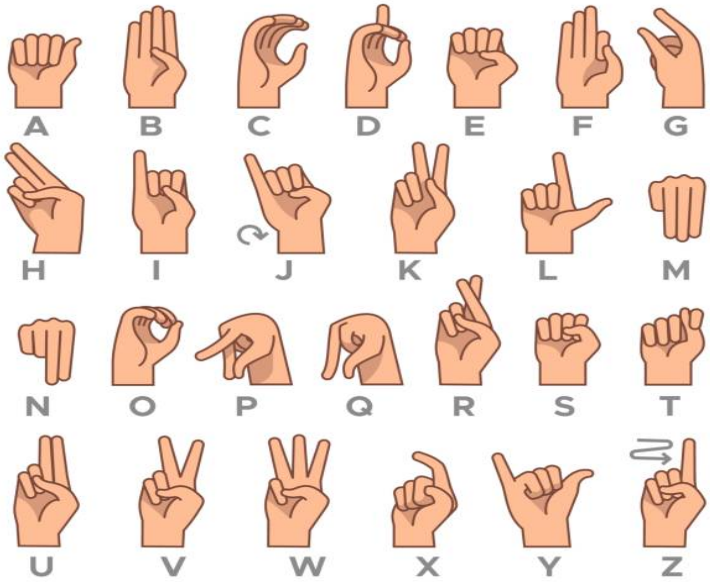
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

RECONOCIMIENTO DE LA LENGUA DE SIGNOS MEDIANTE APRENDIZAJE PROFUNDO



Estudiante: Durán Arce, Unai

Director/Directora: Nuñez Marcos, Adrián

Co-Director/Co-Directora: Labaka Intxauspe, Gorka

Curso: 2023-2024

Fecha: 27 de junio de 2024

Resumen

La comunicación efectiva es esencial en todas las interacciones y para acceder a muchos servicios. Para las personas con discapacidades auditivas, el lenguaje de signos es su principal medio de comunicación. No obstante, la barrera lingüística entre las lenguas de signos y las lenguas habladas, así como entre las diferentes lenguas de signos, puede dificultar la comunicación fluida entre personas con discapacidad auditiva y personas oyentes, así como entre las propias personas signantes.

Una aplicación para traducir puede desempeñar un papel crucial en la eliminación de esta barrera. Gracias a las tecnologías avanzadas actuales, como el aprendizaje profundo, es posible desarrollar sistemas capaces de reconocer la lengua de signos. Estos modelos pueden aprender patrones complejos en las secuencias de gestos de la lengua de signos y generar traducciones.

En este contexto, el presente proyecto se centra en el desarrollo y entrenamiento de un modelo de clasificación de secuencias basado en *transformers*. Este modelo se ha entrenado utilizando un conjunto de datos que incluye vídeos de personas realizando un solo signo en lenguaje de signos. De estos vídeos se ha extraído una secuencia de poses, formadas por puntos de interés del cuerpo humano, como la posición de las manos y los brazos, así como la expresión facial.

Para evaluar la eficacia del modelo, se ha utilizado un conjunto de datos de prueba denominado WLASL, que recoge dos mil ejemplos de diferentes poses en lengua de signos americana (ASL). Se ha calculado tanto la precisión del reconocimiento como la precisión top-5 para verificar la capacidad del modelo de clasificar correctamente. Aunque este trabajo no es exhaustivo debido al alcance del TFG, los resultados obtenidos no están lejos de los reportados en la literatura.

Laburpena

Komunikazio eraginkorra funtsezkoa da elkarrekintza guztietan. Entzumen-urritasunak dituzten pertsonentzat, zeinu-mintzaira da komunikazio-bide nagusia. Hala ere, zeinu-hizkuntzen eta hizkuntza mintzatuen arteko hizkuntza-oztopoak, bai eta zeinu-hizkuntzen artekoak ere, zaildu egin dezake entzumen-urritasuna duten pertsonen eta entzuleen arteko komunikazio arina, bai eta zeinu-mintzairaren beraren artekoa ere.

Itzultzeko aplikazio batek berebiziko garrantzia izan dezake hesi hori kentzeko. Gaur egungo teknologia aurreratuei esker, hala nola ikaskuntza sakonari esker, zeinu-hizkuntza ezagutzeko gai diren sistemak garatu daitezke. Eredu horiek eredu konplexuak ikas ditzakete zeinu-hizkuntzaren keinuen sekuentzietan, eta itzulpenak sor ditzakete.

Testuinguru horretan,*transformers* -en oinarritutako sekuentziak sailkatzeko eredu baten garapenean eta entrenamenduan zentratzen da proiektu hau. Eredu hau pertsonen bideoak barne hartzen dituen datu-multzo bat erabiliz entrenatu da, zeinu-hizkuntzan zeinu bakar bat eginez. Bideo horietatik pose-sekuentzia bat atera da, giza gorputzaren puntu interesgarriez osatuak, hala nola eskuen eta besoen posizioaz eta aurpegi-adierazpenaz.

Ereduaren eraginkortasuna ebaluatzeko, WLASL izeneko datu-multzoa erabili da, Amerikako zeinu-mintzairaren (ASL) bi mila adibide biltzen dituena. Ereduak behar bezala sailkatzeko duen gaitasuna egiaztatzeko, aintzatespenaren zehaztasuna eta top-5 zehaztasuna kalkulatu dira. Nahiz eta lan hau ez den zehatza GFAren irismena dela eta, lortutako emaitzak ez daude literaturan jasotakoetatik urrun.

Abstract

Effective communication is essential for all interactions and accessing many services. For people with hearing disabilities, sign language is their primary communication tool. However, the language barrier between sign and spoken languages, as well as between different sign languages, can hinder smooth communication between hearing and deaf people and among signers themselves.

A translation application can play a crucial role in eliminating this barrier. Thanks to advanced technologies like deep learning, it is now possible to develop systems capable of recognizing sign language. These models can learn complex patterns in sign language gesture sequences and generate translations.

In this context, the present project focuses on developing and training a sequence classification model based on transformers. This model has been trained using a dataset that includes videos of people performing individual signs in sign language. From these videos, a sequence of poses has been extracted, capturing key points of the human body, such as hand and arm positions, as well as facial expressions.

To evaluate the model's effectiveness, the WLASL test dataset, which includes two thousand examples of different poses in American Sign Language (ASL), has been used. Both recognition accuracy and top-5 accuracy have been calculated to verify the model's classification ability. Although this work is not exhaustive due to the project's scope, the results obtained are comparable to those reported in the literature.

Índice

Índice de figuras

Índice de tablas

1. Introducción

En esta sección se pondrá en contexto la base de este proyecto, para posteriormente presentar los objetivos a alcanzar en el proyecto y la estructura de la memoria.

1.1. Contexto

Las lenguas de signos [38] son un sistema de comunicación visual y gestual utilizado por millones de personas en todo el mundo [52]. A pesar de su importancia, la lengua de signos aún enfrenta barreras significativas en términos de reconocimiento y comprensión por parte de quienes no las utilizan regularmente. Estas barreras pueden dificultar la comunicación efectiva y la inclusión social de las personas con discapacidad auditiva, con dificultades de audición o mudas en una sociedad dominada por la lengua oral. [35]

Esta problemática no solo limita las oportunidades de interacción social, sino que también puede dificultar el acceso a servicios esenciales y la participación en la educación y el empleo. Un buen ejemplo sería la situación en la que una persona que solo pueda comunicarse usando la lengua de signos trate de buscar un trabajo en el cual sea fundamental estar de cara al público. Por ejemplo, el caso de un responsable de ventas en una tienda de ropa que no puede desarrollar su trabajo de forma óptima debido a una barrera de comunicación entre una persona sorda y una persona oyente (el cliente).

Debido a la consideración de estas lenguas como minoritarias y a la falta de atención por parte de organismos públicos o la comunidad científica, entre otros, no se han generado suficientes recursos como herramientas de comunicación específicas que permitan abordar adecuadamente estas barreras y promover la inclusión en diversos ámbitos sociales y laborales.

En cambio, para los idiomas hablados existe una gran variedad de herramientas como el traductor desarrollado por Google, los propios traductores desarrollados por compañías de diccionarios (los cuales están más afinados) o incluso herramientas tan potentes como chatGPT, Bard[1] o asistentes virtuales como Siri, Cortana, etc. Estas herramientas ayudan a aliviar la carga de trabajo y permiten el acceso rápido a ciertos contenidos. Mientras tanto, para el público no oyente y signante no existen herramientas parecidas que les permitan facilitar su vida.

En el contexto actual, este Trabajo de Fin de Grado (TFG) propone emplear la tecnología del aprendizaje profundo [13] (*deep learning*) como una herramienta prometedora para abordar la problemática planteada. El Aprendizaje Profundo ha destacado por su capacidad para descubrir patrones complejos, siendo particularmente eficaz en la resolución de tareas desafiantes como la clasificación de imágenes. Por ejemplo, distinguir entre imágenes de perros y gatos. A simple vista, estas imágenes pueden presentar similitudes notables, pero también poseen diferencias sutiles en características como la forma de las orejas, el contorno del hocico o la textura del pelaje. Estas diferencias, son captadas y procesadas por los modelos de Aprendizaje Profundo.

[1] **Bard:** Herramienta similar a chatGPT pero desarrollada por Google.

Un modelo de Aprendizaje Profundo examina una amplia variedad de imágenes de perros y gatos, aprendiendo automáticamente a identificar y distinguir las características distintivas. La experiencia adquirida examinando las imágenes permite al modelo clasificar correctamente aquellas imágenes que no ha visto antes.

Si se llevase la aplicación de esta tecnología a la lengua de signos. Mediante el uso de ejemplos de vídeos con una persona signante y su correspondiente traducción, es teóricamente posible que el modelo aprenda a reconocer y traducir la lengua de signos. Esto lleva a dos aplicaciones principales del aprendizaje profundo en este campo:

- **Traducción de lengua de signos a lengua hablada:** Proceso mediante el cual se convierten los signos y otros gestos utilizados en la comunicación con lengua de signos en palabras en una lengua oral. Esto implica capturar el significado de los signos y expresiones de la lengua de signos y transmitirlos de manera comprensible en la lengua oral.
- **Traducción automática de texto a la lengua de signos:** Dado un texto en un idioma hablado (p.e. castellano) este será traducido a lengua de signos. La traducción puede realizarse generando un vídeo (inteligencia artificial generativa) o utilizando *key points* (Puntos clave) para animar un avatar.

Figura 1: Avatar animados para presentar el signo *Again* en ASL

Otra alternativa consiste en utilizar audio como entrada, que luego será procesado por inteligencia artificial para transcribirlo en texto. Posteriormente, este texto se traducirá a lengua de signos. Además, es importante destacar que estas tareas se realizan con múltiples lenguas orales y de signos. Esta alternativa sería especialmente adecuada para aplicaciones móviles, ya que resulta mucho más cómoda y natural que escribir lo que se quiere expresar

En este TFG, nos limitaremos únicamente a la tarea de traducción de lengua de signos a lengua hablada. Debido a la complejidad de esta tarea, la literatura científica comúnmente la divide en las siguientes dos fases.

- **Reconocimiento de lengua de signos (Sign Language Recognition):** Es el proceso de identificar y comprender la información contenida en un vídeo que representa una oración en lengua de signos. El objetivo es traducir los gestos y expresiones presentes en el vídeo a una forma reconocible y comprensible, como texto o voz.

- **Traducción de lengua de signos (Sign Language Translation):** Es el proceso de convertir un texto o discurso de una lengua a otra, incluso convertir una representación intermedia basada en el reconocimiento de signos a texto o discurso.

Este trabajo se centrará en la tarea de reconocimiento de lengua de signos, dejando fuera de consideración la traducción directa de texto o voz a lengua de signos. Específicamente, se abordará el problema de identificar el signo que se está realizando en un vídeo dado (*Isolated Sign Language Recognition*), sin intentar segmentar ni traducir una secuencia completa de signos (*Continous Sign Language Recognition*).

Esta decisión se justifica por la limitación de tiempo inherente a un Trabajo de Fin de Grado. Si bien la traducción completa de oraciones en lengua de signos es un objetivo deseable, la complejidad de segmentar y traducir secuencias completas de signos, así como la implementación del *continuous sign language recognition*, podría extenderse más allá del alcance de este proyecto [22]. Sin embargo, se deja abierta la posibilidad de abordar el *continuous sign language recognition* y la traducción completa como una extensión natural de este trabajo en un futuro como Trabajo de Fin de Máster.

Durante el desarrollo de este TFG, se examinarán las características de la lengua de signos, se ofrecerá una introducción al aprendizaje profundo y se proporcionarán detalles sobre las herramientas y recursos disponibles para la implementación del sistema. Como resultado de este TFG:

- Se desarrollará un sistema capaz de reconocer un signo que una persona signante realiza en un vídeo.
- Se realizará una evaluación dicho sistema en un conjunto de datos público.

En última instancia, este TFG busca contribuir al desarrollo de tecnologías inclusivas que promuevan la comunicación efectiva entre personas con discapacidad auditiva y la sociedad en su conjunto, alineándose con el Objetivo de Desarrollo Sostenible (ODS) 10 de la Agenda 2030 (Reducción de las desigualdades)[2]. Al aplicar la capacidad del aprendizaje profundo a la lengua de signos, se esperan abrir nuevas posibilidades para la inclusión y la igualdad de oportunidades.

Esta iniciativa demuestra el potencial transformador de las herramientas asociadas al aprendizaje profundo en la resolución de problemas sociales y humanitarios, contribuyendo así a construir un mundo más justo y equitativo. Además, se ha tenido en cuenta el reglamento de convivencia de la UPV/EHU, siguiendo sus principios y valores democráticos[3].

1.2. Objetivos y alcance del proyecto

El objetivo principal de este proyecto es el desarrollo de un sistema capaz de realizar el reconocimiento de la lengua de signos a partir de un vídeo mediante técnicas de aprendizaje profundo. Para lograr este propósito, el TFG se desglosa en los siguientes objetivos específicos:

[2] **ODS 10:** https://www.pactomundial.org/ods/10-reduccion-de-las-desigualdades/

[3] **Reglamento de convivencia de la UPV/EHU:** https://www.ehu.eus/es/web/gardentasun-ataria/universidad-del-pais-vasco/euskal-herriko-unibertsitateko-elkarbizitza-araudia

1. **Investigación en lengua de signos y aprendizaje profundo:**
   - Revisión del artículo científico [35] proporcionado por los tutores del TFG al inicio del proyecto junto con el otro artículo científico [28] proporcionado sobre el conjunto de datos WLASL.
     - El primer artículo científico comentado (*A survey on Sign Language machine translation*) se ha utilizado para comprender los avances en la traducción automática de la lengua de signos, los métodos utilizados y los desafíos que enfrenta esta área.
     - El segundo artículo científico comentado (*Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparision*) se ha utilizado para comparar los resultados obtenidos con los publicados sobre el mismo conjunto de datos, y así comprender que otras herramientas pueden ser útiles en el reconocimiento de la lengua de signos.
   - Identificar y analizar los enfoques, metodologías y tecnologías más relevantes en el campo para comprender el estado del arte y tendencias actuales.
   - Investigar y comprender el funcionamiento de la lengua de signos.
2. **Recopilación y preprocesamiento de datos**
   - Recolectar un conjunto de datos para el entrenamiento y validación del sistema de reconocimiento de lengua de signos.
   - Desarrollo de software para el preprocesamiento de los vídeos, incluyendo la normalización y la extracción de características relevantes.
3. **Diseño de la arquitectura de aprendizaje profundo:**
   - Seleccionar y diseñar una arquitectura de red neuronal profunda que se adecúe a los objetivos del proyecto, los datos y la infraestructura disponible.
   - Se escogerán una selección inicial de hiperparámetros de la red, como el número de capas, neuronas, funciones de activación y tasa de aprendizaje, para comenzar con las pruebas y comprobar que el modelo comienza a aprender.
4. **Entrenamiento del modelo:**
   - Entrenar con una muestra pequeña de datos para ir visualizando el comportamiento del modelo como por ejemplo, comprobar la capacidad de aprendizaje del modelo mediante un *overfitting* inicial, búsqueda de posibles errores, etc.
   - Entrenar la red utilizando el conjunto de datos recopilado y preprocesado en búsqueda de los hiperparámetros más adecuados.
5. **Evaluación del Sistema:**
   - Evaluar el rendimiento del sistema mediante métricas como la precisión, el recall, la precisión ponderada y la matriz de confusión en el conjunto de datos de validación y si es posible, en un conjunto de datos de prueba.

Es importante reiterar que en el próximo proyecto no se llevará a cabo la tarea descrita en la sección 1.1 como "traducción automática de lengua de signos"debido a su complejidad. Además, caber destacar que se ha utilizado un conjunto de datos controlado que incluye vídeos con signos correspondientes a una única glosa. Este conjunto está controlado, ya que presenta a una sola persona que mira continuamente a la cámara, viste ropa adecuada y se encuentra frente a un fondo sin detalles. Estas medidas facilitan la reducción de ruido y de características no relevantes en los datos. Por lo tanto, es importante tener en cuenta que los resultados obtenidos pueden no ser extrapolables a un entorno real donde se logre una comunicación completa con frases.

1.3. Estructura del proyecto

Esta memoria se estructura de la siguiente manera:

- **1. Introducción**
  Introducción al contexto, la motivación del proyecto y descripción del desarrollo.
- **2. Lengua de signos**
  Explicación y trasfondo de la lengua de signos para comprender su funcionamiento general y las diferentes problemáticas existentes en la actualidad.
- **3. Aprendizaje profundo**
  Fundamentos teóricos del aprendizaje profundo junto con varios de los conceptos que la engloban.
- **4. Planificación**
  Se expone la planificación del proyecto: desde la planificación temporal, pasando por un repaso de las herramientas utilizadas, un análisis de gestión de riesgos y finalizando con una evaluación económica del coste del proyecto.
- **5. Desarrollo**
  Se presenta una explicación sobre el conjunto de datos utilizado, el preprocesado realizado sobre dicho conjunto, el entorno de experimentación que se ha utilizado y una descripción del modelo.
- **6. Pruebas y resultados**
  Teniendo en cuenta el desarrollo, se exponen las pruebas realizadas, los resultados y la valoración de los mismos.
- **7. Conclusiones**
  Se exponen conclusiones sobre el proyecto, conclusiones personales y trabajo a futuro a partir del TFG.
- **Referencias**
  Referencias utilizadas en el trabajo.
- **A. Anexo I**
  Anexo con las las tareas de los paquetes de trabajo.

2. Lengua de Signos

La lengua de signos es un tema de gran relevancia en el ámbito de la lingüística y la comunicación. A lo largo de la historia, ha sido objeto de estudios y análisis profundo que han permitido comprender su complejidad y riqueza como sistema de comunicación. [18]. De hecho, según la Federación Mundial de Sordos (WFD), se estima que hay aproximadamente 72 millones de personas en todo el mundo que utilizan la lengua de signos como su principal medio de comunicación. [15]

Además, cabe destacar, que no hay una única lengua de signos para todo el mundo, si no que existen una gran variedad de ellas en el mundo como pueden ser la lengua de signos española (LSE), la lengua de signos estadounidense (ASL), lengua de signos china (LSC), etc. Esto subraya la importancia de investigar y comprender este sistema lingüístico. Un ejemplo de la magnitud de su uso sería el presentado en la **Figura 2** donde se puede observar el numero de personas que utilizan la lengua de signos en algunos de los países Europeos.

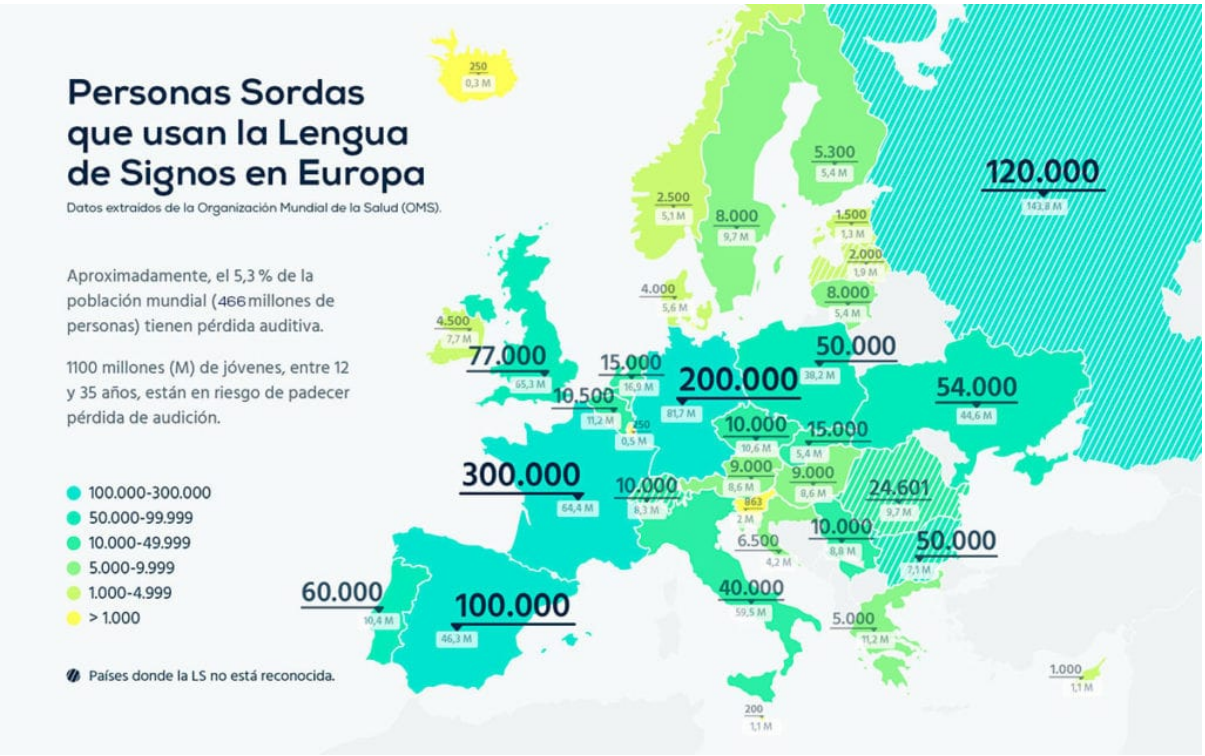


Figura 2: Personas sordas que usan la lengua de signos en Europa
Fuente: `https://www.visualfy.com`

La lengua de signos consiste en la realización de signos en el aire que representan conceptos, referencias, etc., uno tras otro, similar a como se concatenan palabras en las lenguas orales. Dentro de estos signos existen varias categorías, siendo las dos más relevantes para este TFG las siguientes.

Por una parte está el *fingerspelling* o deletreo manual. Es una técnica que permite representar letras individuales del alfabeto utilizando gestos y movimientos específicos de las manos. Según Stokoe (1960), [48] quien realizó un trabajo pionero en la investigación de la lengua de signos, el *fingerspelling* desempeña un papel esencial en la construcción de palabras y la comunicación precisa en este idioma visual-gestual.

Por otra parte se encuentra la representación de palabras, otro pilar fundamental de esta lengua. A diferencia del *fingerspelling*, este se utiliza para conseguir comunicar palabras o expresiones más complejas mediante el uso de articulaciones, manos y gestos faciales.

A continuación, en la sección 2.1, se analizará y se pondrá en contexto el *fingerspelling*, continuando con la sección 2.2 en la que se hablará sobre que es la representación de palabras en la lengua de signos. Después, se realizará una explicación sobre que son las glosas en la lengua de signos en la sección 2.3 y finalmente se concluirá con la sección 2.4 en la que se comentará la problemática que se presenta en la comunicación entre una persona signante con discapacidad auditiva con una persona sin discapacidad y no signante y como esto supone una gran barrera en la inclusión de los afectados.

2.1. Fingerspelling

Dentro de la lengua de signos, existe una herramienta importante conocida como *fingerspelling* o "deduccionario", que desempeña un papel crucial en la comunicación.

2.1.1. ¿Qué es el *fingerspelling*?

El *fingerspelling*, también llamado "alfabeto dactilológico" [30], es una técnica de la lengua de signos que utiliza los dedos de la mano para representar letras individuales del alfabeto, por ejemplo en el caso de la Lengua de Signos Española representa el alfabeto del castellano. Cada letra se representa mediante una configuración específica de los dedos y la mano, y los signantes deletrean palabras o nombres usando este método. Es especialmente útil cuando no hay un signo especifico para una palabra en el vocabulario de la lengua de signos, como nombres propios, términos técnicos o palabras extranjeras.

El alfabeto dactilológico consta de un signo por cada letra del alfabeto, una para cada letra del alfabeto de la lengua oral que se use como referencia; en el caso del British Sign Language (BSL) con el inglés, dicho alfabeto consta de 26 letras. Sin embargo, en algunas variantes de la lengua de signos, se pueden utilizar modificaciones para representar letras adicionales o acomodar las necesidades específicas de los usuarios.

2.1.2. ¿Cómo se utiliza?

Cada letra del alfabeto se representa mediante una posición de mano única y distintiva. Aquí hay algunos ejemplos de representaciones de letras en el alfabeto dactilológico de la lengua de signos estadounidense (ver figura 3):

- La letra 'A' se forma cerrando el puño extendiendo el dedo pulgar.
- La letra 'B' se representa con la mano abierta y cerrando el pulgar.
- La letra 'C' se representa formando la letra 'C' con la mano.
- La letra 'D' se representa cerrando el puño y levantando el dedo índice hacia arriba.
- La letra 'E' se forma entrecerrando la mano y posicionando el dedo pulgar delante de los demás dedos.

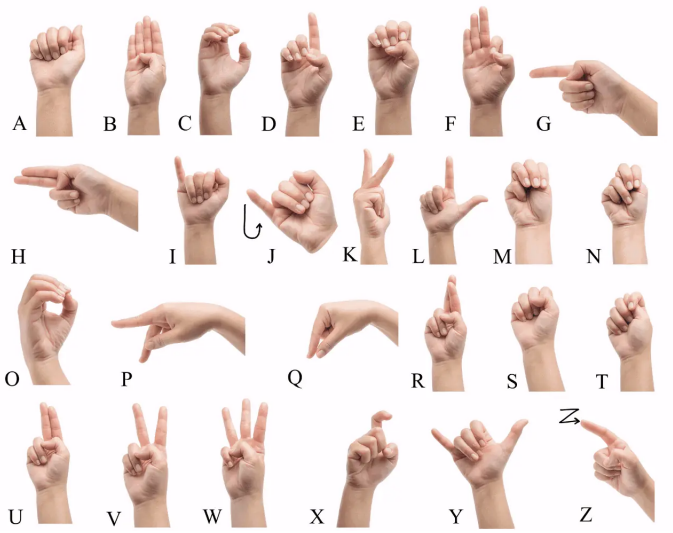


Figura 3: Alfabeto completo en lengua de signos estadounidense (ASL)

Estos son tan solo algunos ejemplos, y cada letra tiene su propia forma. Los signantes deletrean palabras juntando estas formas de manera fluida.

Esta rama de la lengua de signos se trata de una habilidad vital, ya que permite a las personas que lo utilizan comunicar conceptos que no tienen signos específicos atribuidos. Además, es esencial en situaciones donde se requiere precisión, como la ortografía de palabras o la identificación de nombres y lugares. También se utiliza comúnmente en situaciones educativas, como la enseñanza de vocabulario y la lectura de palabras escritas.

2.2. Representación de palabras

La representación de palabras es otra técnica fundamental en la lengua de signos. Esta se basa en la naturaleza visual y gestual de este lenguaje para expresar palabras y conceptos de manera eficiente y efectiva. A diferencia del alfabeto dactilológico o *fingerspelling*, donde se deletrean palabras letra por letra, la representación de palabras se centra en la expresión de significados completos mediante gestos o signos específicos.

En el artículo *Recognition of signed and spoken language: Different sensory inputs, the same segmentation procedure* [37], se destaca la importancia de la representación de palabras en la lengua de signos como una forma efectiva de comunicación y procesamiento del lenguaje en la comunidad sorda. El artículo subraya que la representación de palabras en la lengua de signos va más allá de la simple transliteración de palabras orales y contribuye significativamente a la expresión y comprensión de conceptos complejos.

2.2.1. Uso y aplicación de la representación de palabras

La manera más cotidiana de utilizar esta técnica es para representar conceptos tales como, por ejemplo, los medios de transporte (autobús, coche, avión, etc.) o expresar acciones cotidianas (conducir, empujar algo, trabajar, etc.), los cuales tienen ya signos asociados (ver figura 4). Aunque hay otros casos menos cotidianos y más especificos como:

Expresión de nombres propios

En la lengua de signos, representar palabras significa expresar nombres propios, incluyendo nombres de personas, lugares y entidades. En lugar de deletrear un nombre letra por letra, las personas signantes pueden utilizar un signo único que representa el nombre de manera más eficiente.
Los procedimientos más frecuentes para nombrar a alguien son los siguientes:

- Fijarse en una característica física visible en la cara; por ejemplo, tener un lunar o una cicatriz, la forma de la nariz, de las orejas o del pelo, el bigote o la barba, la utilización de gafas, etc.
- Utilizar el signo del nombre común, cuando el nombre propio coincide con alguno o es muy similar. Así, quien se apellide Rosario, puede ser nombrado con el mismo signo que representa un rosario.

Otras veces son las personas sordas quienes le adjudican una signo para representar a esa persona, como cuando a una persona la llaman por un mote. En general, son signos que no resultan ofensivos y que quien las recibe suele aceptarlas sin problema, puesto que tienen una gran utilidad comunicativa para las personas que utilizan la lengua de signos. [5]

Términos técnicos y jergas

Los términos técnicos y las jergas son esenciales en la lengua de signos porque permiten a los signantes comunicarse de manera precisa y efectiva en contextos especializados. Además, estas expresiones demuestran la flexibilidad y la adaptabilidad de la lengua de

signos para evolucionar y abordar nuevas áreas de conocimiento y diversidad cultural.

El desarrollo y la compresión de términos técnicos y jergas en la lengua de signos son fundamentales para la participación activa y efectiva en una variedad de campos, desde la ciencia y la medicina hasta la comunidad sorda LGBTQ+ [3] y más allá.

Estos elementos lingüísticos reflejan la riqueza y la diversidad de la comunidad sorda o con dificultad auditiva, y subrayan la importancia de considerar las particularidades culturales y lingüísticas al interactuar con esta comunidad.

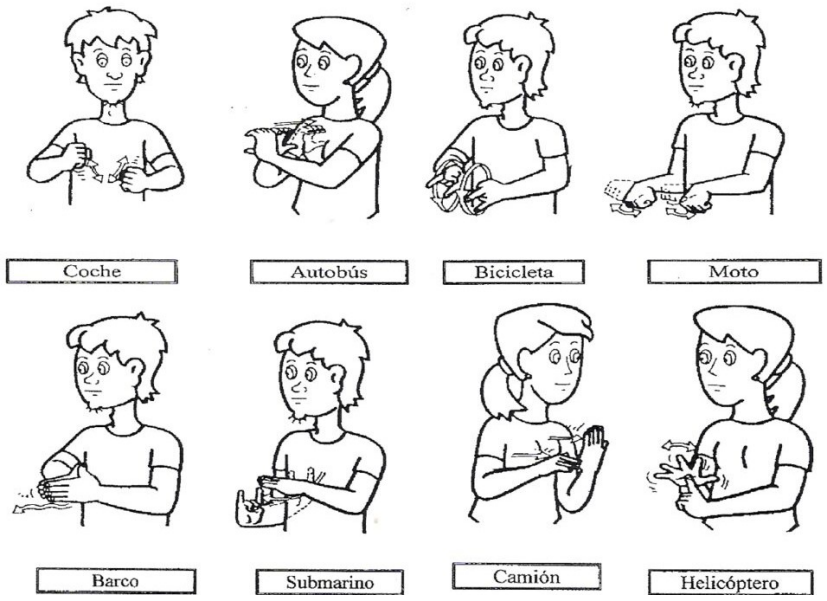


Figura 4: Algunos ejemplos cotidianos de representación de palabras en Lengua de Signos Española (LSE)

Fuente: `https://aprendelenguadesignos.com`

Creación de nuevos signos

La creación de nuevos signos es un proceso dinámico y colaborativo que refleja la capacidad de aceptación y evolución de la lengua visual y gestual.

La comunidad de personas signantes trabaja en conjunto para desarrollar signos que representen estos elementos de manera efectiva. A continuación se explicarán algunos de los pasos y consideraciones clave en la creación de nuevos signos en la lengua de signos [6].

Pasos en la creación de nuevos signos:

1. **Identificación de la necesidad:**
   El proceso comienza cuando surge la necesidad de expresar un concepto o término para el cual no existe un signo convencional. Esto puede deberse a la introducción de nuevas tecnologías, descubrimientos científicos, cambios culturales o cualquier otro

desarrollo que requiera una representación visual y gestual.

2. **Conceptualización:**
   Se comienza conceptualizando el término que se desea expresar. Esto implica comprender completamente el significado y los aspectos clave del término. La iconicidad, o la capacidad de un signo para representar visualmente la idea, puede variar dependiendo del contexto cultural y lingüístico, pero en general, busca crear una conexión clara entre el signo y el concepto que representa.

3. **Colaboración comunitaria:**
   La creación de nuevos signos es un proceso colaborativo en el que la comunidad signante trabaja en conjunto. Se reúnen y se discute el cómo podría representarse mejor el término o concepto en cuestión.

4. **Desarrollo del signo:**
   A partir de los pasos anteriores se desarrolla un nuevo signo. Esto puede implicar la combinación de movimientos de mano, gestos faciales y elementos corporales para representar visualmente el término. La simplicidad y la claridad son fundamentales para asegurar que el nuevo signo sea fácilmente reconocible y comprensible.

5. **Prueba y refinamiento:**
   Una vez creado el nuevo signo, se somete a pruebas en la comunidad para evaluar su efectividad. Como prueba, el signo se utiliza en contextos reales y se proporciona una retroalimentación sobre su comprensión y facilidad de uso. En función de esta retroalimentación, el signo se refina y ajusta.

6. **Adopción:**
   Finalmente, si el nuevo signo se considera exitoso y aceptado por la comunidad de signantes, se adopta como parte del vocabulario de la lengua de signos y se integra en la comunicación cotidiana.

Algunos ejemplos de ámbitos en los que se suele llevar a cabo la creación de nuevos signos son:

- **Tecnología:**
  Con la evolución de la tecnología, la lengua de signos ha incorporado nuevos signos para términos como “teléfono inteligente”, que involucra gestos que representan una pantalla táctil y la idea de inteligencia.

- **Términos científicos:**
  En campos científicos, se pueden crear signos para conceptos y términos científicos que no tienen una representación establecida.

2.3. Glosas en la lengua de signos

Las glosas consisten en la representación escrita de los signos y estructuras gramaticales utilizados en la lengua de signos. A través de las glosas, los lingüistas y estudiantes de la lengua de signos pueden analizar la estructura y la gramática de esta forma de comunicación.

En las glosas, cada signo se representa utilizando símbolos específicos que indican la forma de la mano, el movimiento, la ubicación y otros aspectos importantes del signo. Además, también pueden incluir información sobre la expresión facial y el contexto lingüístico en el que se utiliza el signo.

Por ejemplo, la glosa de la frase "Tengo cita para recoger mi título universitario" en lengua de signos puede ser representada de la siguiente manera:

Figura 5: Transición del signo a glosa
Fuente: `https://www.mdpi.com/1424-8220/24/5/1472`

El uso de las glosas ayuda a comprender la estructura y la gramática de la lengua de signos, así como para documentar y analizar las variaciones regionales y sociales en el uso de la lengua. Además, son una herramienta valiosa para la enseñanza y el aprendizaje de la lengua de signos, ya que permiten descomponer y comprender los signos de manera más detallada.

En la práctica, las glosas se utilizan en estudios lingüísticos, en la formación de intérpretes de lengua de signos y en la producción de materiales educativos para personas con discapacidad auditiva.

2.4. Problemática de la lengua de signos

Aunque la lengua de signos es una forma rica y expresiva de comunicación, enfrenta diversas problemáticas que pueden dificultar la interacción tanto entre personas signantes, como entre personas signantes y no signantes.

A continuación, se realiza un análisis de las problemáticas en la comunicación en la lengua de signos.

Diversidad de lenguas de signos

Problemática: Una de la problemáticas más prominentes en la comunicación en lengua de signos es la diversidad de lenguas de signos en todo el mundo. Cada país o región puede tener su propia lengua de signos con gramáticas, léxicos y estructuras únicas. Por ejemplo, la lengua de signos estadounidense es diferente de la lengua de signos británica o de la lengua de signos española. Esta diversidad lingüística crea barreras en la comunicación cuando signantes de diferentes lenguas de signos se encuentran. Sin embargo, es importante destacar que existen similitudes entre lenguas de signos de la misma familia o geográficamente cercanas, por lo que dos signantes podrían llegar a entenderse.

Impacto: Requiere que las personas sordas de diferentes regiones y los signantes aprendan múltiples lenguas de signos si desean comunicarse internacionalmente. Como referencia, se puede contemplar la variedad de familias de lengua de signos en el mundo en la **Figura 6**.

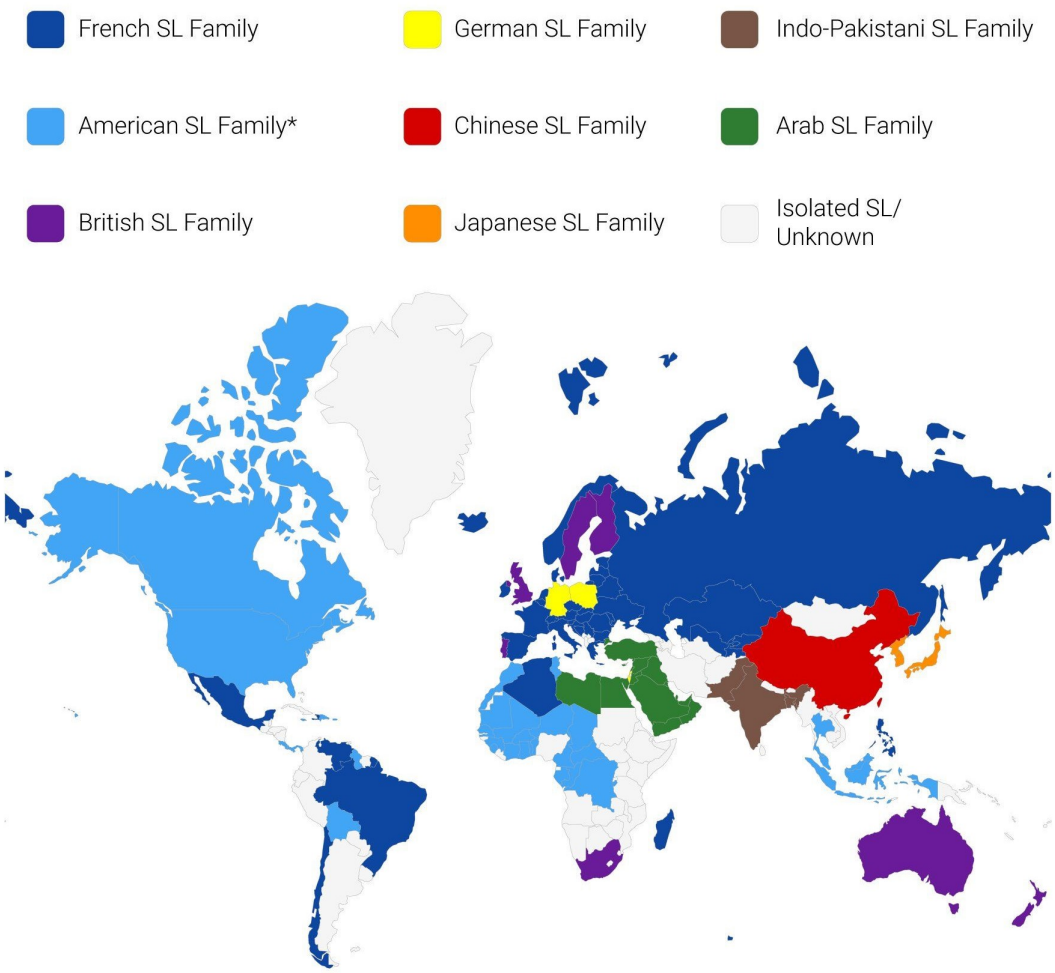


Figura 6: Principales familias del Lenguaje de Signos

Fuente: `https://www.facebook.com/mapas.milhaud/photos/a.145693298902369/1675093515962332/?type=3&locale=es_LA`

Variaciones regionales y sociales

Problemática: Incluso dentro de una misma lengua de signos, pueden existir variaciones regionales y sociales en los signos y gramática. Las diferencias dialectales pueden dificultar la comunicación entre signantes de diferentes áreas geográficas. Además, las variaciones sociales pueden estar relacionadas con la edad, el nivel de educación y otros factores, lo que puede llevar a malentendidos y confusiones en la comunicación. [54]

Impacto: Pueden crear desafíos en la comunicación, especialmente en situaciones donde se requiere precisión y claridad, como en contextos educativos o profesionales.

Acceso limitado a intérpretes

Problemática: La falta de acceso a intérpretes de lengua de signos cualificados es una problemática común. En situaciones importantes, como en entornos educativos, médicos o legales, la presencia de un intérprete es esencial para garantizar la comunicación efectiva.

Impacto: Limitar las oportunidades educativas y laborales de las personas sordas y dificultar su acceso a la información y los servicios. En la figura 7 se puede observar un gráfico que muestra la cantidad de personas con discapacidad auditiva en España en 2020.

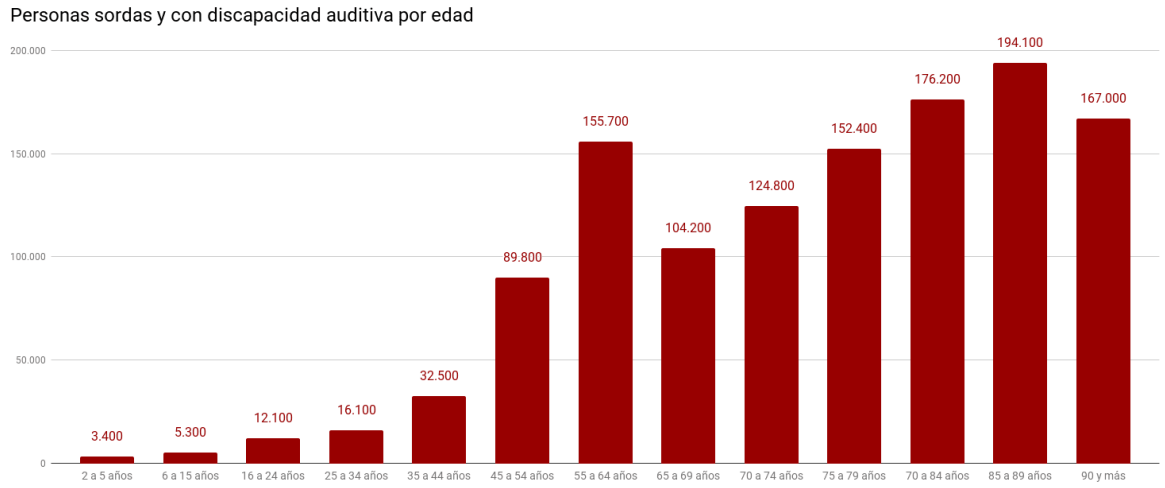


Figura 7: Número de personas con discapacidad auditiva por edad en España (2020)
Fuente: `https://www.ine.es`

Falta de conciencia y educación

Problemática: Muchas de las personas no signantes no están familiarizadas con la lengua de signos ni comprenden las necesidades y la cultura de la comunidad sorda.

Impacto: La falta de conciencia y educación puede hacer que las personas sordas se sientan marginadas y limitar sus oportunidades en la sociedad.

Barreras de comunicación

Problemática: La comunicación entre personas sordas y no signantes puede enfrentar diversas barreras. Estas barreras pueden incluir la falta de acceso a intérpretes, la falta de recursos de comunicación alternativos y la incomodidad en la interacción. Las personas sordas pueden sentirse frustradas cuando no pueden comunicarse de manera efectiva con quienes no comprenden la lengua de signos.

Impacto: Las barreras de comunicación pueden afectar las relaciones interpersonales, la educación, el empleo y la calidad de la vida en general.

Estigma y discriminación

Problemática: Las personas con discapacidad auditiva a menudo se ven afectadas por prejuicios y trato desigual en la sociedad.

Impacto: Actitudes negativas como la exclusión social y la falta de igualdad de oportunidades, que pueden llevar a afectar en la autoestima y el bienestar emocional de las personas sordas, así como en su participación en la sociedad.

Barreras tecnológicas

Problemática: A pesar de los avances tecnológicos, las barreras tecnológicas pueden dificultar la comunicación en la lengua de signos. La falta de accesibilidad en sitios web, aplicaciones y dispositivos electrónicos pueden limitar la participación en la sociedad digital.

Si que es verdad que en los últimos años se han hecho progresos en cuanto a herramientas que faciliten la comunicación o el entendimiento de la lengua de signos como es el caso de los guantes que recogen la lengua de signos y la traducen (ver figura 8). Pero hay un inconveniente, y es que se trata de una herramienta que resulta bastante incómoda para el usuario puesto que se encuentra en versiones muy prematuras y requiere que el usuario lo lleve puesto continuamente. [14]

Figura 8: Guante traductor de lengua de signos

Impacto: Las barreras tecnológicas pueden excluir a las personas sordas de las oportunidades en línea, la educación en línea y la comunicación en entornos digitales.

En resumen, la comunicación en lengua de signos, a pesar de ser una herramienta poderosa, enfrenta diversas problemáticas que van desde la diversidad lingüística hasta la falta de conciencia y la discriminación. Abordar estas problemáticas requiere de un esfuerzo continuo de la sociedad para promover la inclusión, la educación y la igualdad de acceso a la comunicación para las personas sordas. La conciencia, la formación de intérpretes, la accesibilidad y el respeto por la lengua de signos son elementos clave para superar todos estos desafíos.

3. Aprendizaje Profundo

No se puede hablar de Aprendizaje Profundo o *Deep Learning* sin antes hablar de Aprendizaje Automático o *Machine Learning*.

Se basa en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender patrones a partir de datos y realizar tareas específicas sin una programación explícita. Estas máquinas aprenden de los datos proporcionados y mejoran su rendimiento para una tarea o conjunto de tareas. Esta capacidad de aprendizaje automático ha allanado el camino para el surgimiento del Aprendizaje Profundo, un subconjunto del *Machine Learning*, como se puede ver en la figura 9.

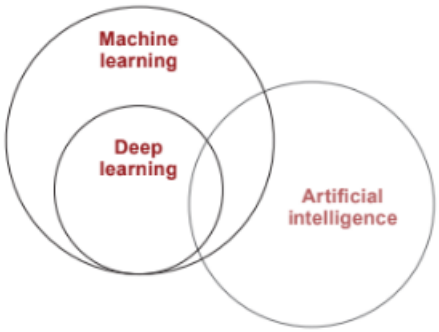


Figura 9: Diagrama de Venn entre *Machine Learning* (Aprendizaje Automático), *Deep Learning* (Aprendizaje Profundo), *Artificial Intelligence* (Inteligencia Artificial). *Deep Learning* es un subconjunto de *Machine Learning* y ambos tienen intersección con *Artificial Intelligence*

El Aprendizaje Profundo ha revolucionado diversos campos con su tecnología, convirtiéndose en un foco para la investigación y el desarrollo, debido a su capacidad para moldear y resolver tareas complejas que antes parecían inabordables.

En esencia, esta tecnología se basa en el concepto de redes neuronales [10], que han existido durante décadas [34]. Sin embargo, el avance del aprendizaje profundo radica en profundizar y diversificar estas redes, lo que antes no era viable debido a las limitaciones como la falta de datos, la ausencia de técnicas de normalización como la normalización por lotes (*batch normalization*) y la falta de potencia de cálculo, especialmente hasta la disponibilidad de GPUs. Esto permitió a los modelos de aprendizaje profundo realizar un aprendizaje jerárquico de conceptos más abstractos.

Para transformar una red neuronal simple en una red neuronal profunda, se añaden varias capas de procesamiento entre la entrada de información y la salida de resultados. Por definición, una red neuronal profunda debe tener más de tres capas [19], diferenciándose así de las redes neuronales de *perceptrón multicapa (MLP)*[4] convencionales. Estas capas adicionales, llamadas “capas ocultas”, siguen una estructura jerárquica: buscan patrones básicos a nivel de datos y, sobre esos patrones, identifican patrones más complejos en capas sucesivas. Esta profundización de la arquitectura permite a las redes neuronales

[4]**Perceptrón multicapa (MLP):** Un tipo de red neuronal artificial que consiste en múltiples capas de neuronas, donde cada capa está completamente conectada a la siguiente.

profundas capturar y representar características más abstractas y complejas de los datos.

Añadir estas capas ocultas permite que la red neuronal aprenda y comprenda la información de una manera más profunda, permitiendo la captura de patrones complejos como reconocer imágenes o traducir idiomas. En el caso de reconocer imágenes, estas capas permiten que la red identifique características y objetos específicos en una imagen, como reconocer rostros, identificar objetos, o clasificar escenas complejas. En cuanto a traducir idiomas, estas capas posibilitan que la red entienda la estructura y el significado de las palabras en diferentes idiomas, permitiendo así la traducción automática de texto entre idiomas distintos con mayor precisión y fluidez [19].

Sin embargo, este enfoque requiere grandes cantidades de datos para obtener resultados aceptables. Gracias al aumento de conjuntos de datos etiquetados y a la mejora en la capacidad de procesamiento, especialmente con el uso de Unidades de Procesamiento Gráfico (GPU)[5], se han generado modelos con trillones de variables a ajustar. Este avance ha supuesto una revolución en las tareas en las que han sido aplicados, permitiendo un rendimiento sin precedentes en áreas como el reconocimiento de imágenes, el procesamiento de lenguaje natural y muchas otras aplicaciones [27].

En los siguientes apartados, se analizarán y se explicarán algunos puntos de esta poderosa herramienta. Se abordará la explicación de las redes neuronales, la teoría de optimización, qué es el overfitting, la generalización y sus principales diferencias. Además, se realizará un análisis sobre las redes convolucionales, se explicará qué son y cómo se utilizan los *transformers*. Asimismo, en este apartado se destacarán ejemplos relevantes donde el uso del aprendizaje profundo ha tenido un impacto significativo, como en el caso de *ChatGPT*.

3.1. Redes neuronales

Las neuronas son unidades fundamentales que funcionan como modelos matemáticos. Cada neurona recibe información, la procesa y luego envía una señal de salida [31]. Estas tienen conexiones de entrada a través de las cuales se reciben estímulos externos (denominados valores de entrada). Con estos valores, la neurona realiza un cálculo interno y genera un valor de salida, funcionando de manera similar a una función matemática *F(x)* [25].

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

A continuación, se comentará cada uno de los términos reflejados en la formula anterior.

- **Entradas (x_i):** Cada neurona recibe una o más entradas, que generalmente se obtienen de las salidas de otras neuronas o de los datos de entrada de la red. Mate-

[5] **Unidades de GPU:** En el contexto del Aprendizaje Profundo son componentes hardware especializados que aceleran el procesamiento de datos para entrenar y ejecutar modelos de redes neuronales profundas.

máticamente, estas entradas pueden representarse como un vector $\mathbf{x} \in \mathbb{R}^n$, donde n es el tamaño del vector y $\mathbb{R}$ denota el conjunto de los números reales.

- **Pesos (w_i):** Son parámetros ajustables que determinan la importancia de cada entrada en una neurona. Estos pueden ser representados como una matriz $\mathbf{W}$. Sus dimensiones son $x_1 \times x_2$, donde x_1 es la misma dimensión que el vector de entradas $\mathbf{x}$, y x_2 es la dimensión de salida esperada.
- **Sesgo (bias) (b):** Permite a la neurona ajustar la función de activación, desplazándola para que la neurona pueda activarse (o no) en diferentes condiciones de entrada. $(x \cdot w + b)$ es una transformación lineal. Como consecuencia, el sesgo mejora el rendimiento y la precisión del modelo, ya que sin él la capacidad de ajuste de la red se vería significativamente limitada.
- **Función de activación (f):** Después de realizar la suma ponderada, se aplica una función de activación a ese valor. Esta función introduce la no linealidad en la operación y determina si la neurona se activa (produce una salida) o no. Las funciones más comunes son: [41]
  - *Función Sigmoide:* Transforma cualquier valor de entrada en un rango de 0 a 1. Es útil en problemas de clasificación binaria, donde la salida representa una probabilidad (Ver figura 10).

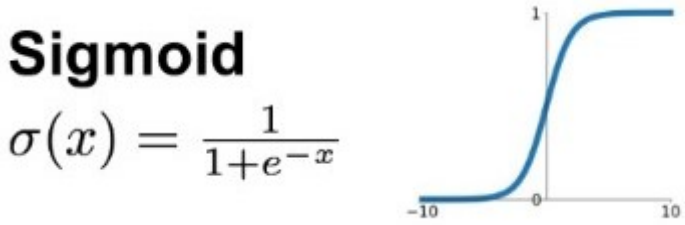


Figura 10: Función Sigmoide

  - *Función ReLU (Rectified Linear Unit):* Mantiene los valores positivos intactos y convierte los valores negativos en cero. Sin embargo, puede tener problemas con neuronas "muertas" si su entrada es siempre negativa (Ver figura 11).

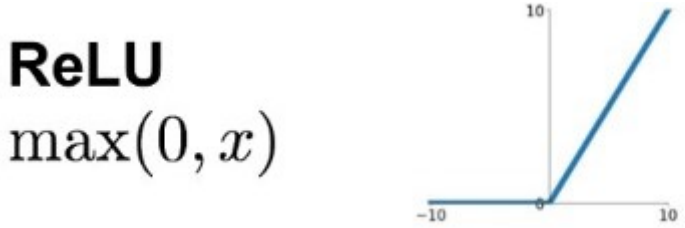


Figura 11: Función ReLU

- *Función tangente hiperbólica:* Es similar a la función Sigmoide (Ver figura 12).

Figura 12: Función tangente hiperbólica

- *Función Leaky ReLU:* Es una variante de ReLU que permite mitigar el problema de las neuronas "muertas" (Ver figura 13).

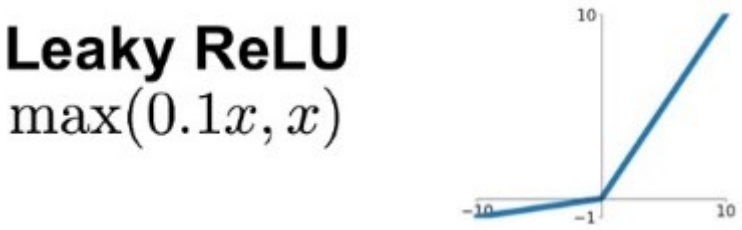


Figura 13: Función Leaky ReLU

- *Función Unidad Lineal Rectificada (ReLU Paramétrica - PReLU):* Similar a la Leaky ReLU, pero α se convierte en un parámetro que se puede ajustar, lo que lo hace más flexible (Ver figura 14).

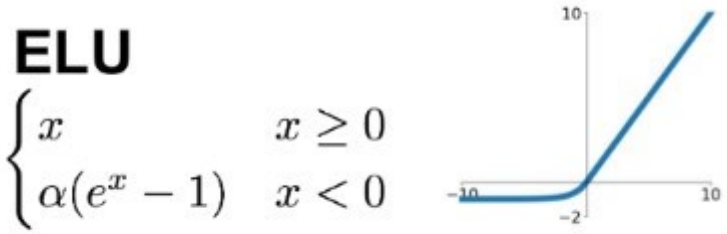


Figura 14: Función Unidad Lineal Rectificada

- **Salida (y):** Es el resultado de aplicar la función de activación a la suma ponderada. Esta salida puede utilizarse como entrada para las neuronas en las capas de la siguiente red.

Cuando el problema se vuelve más complejo de lo esperado, es posible utilizar más de una neurona. En estos casos, las salidas de varias neuronas se combinan para formar un vector de dimensión x_2, siguiendo el mismo principio que se aplica a los pesos (w_i). Estas neuronas interconectadas forman lo que se conoce como una capa de neuronas. Si una sola capa de neuronas no es suficiente para resolver el problema, se deben añadir más capas para mejorar la capacidad de la red. Estas capas adicionales constituyen un conjunto de neuronas que trabajan juntas para realizar una función específica. Cuando la red tiene tres capas o más, se la denomina *Multilayer Perceptron* (MLP), capaz de modelar relaciones complejas en los datos. Un ejemplo visual de este tipo de red se puede ver en la figura 15

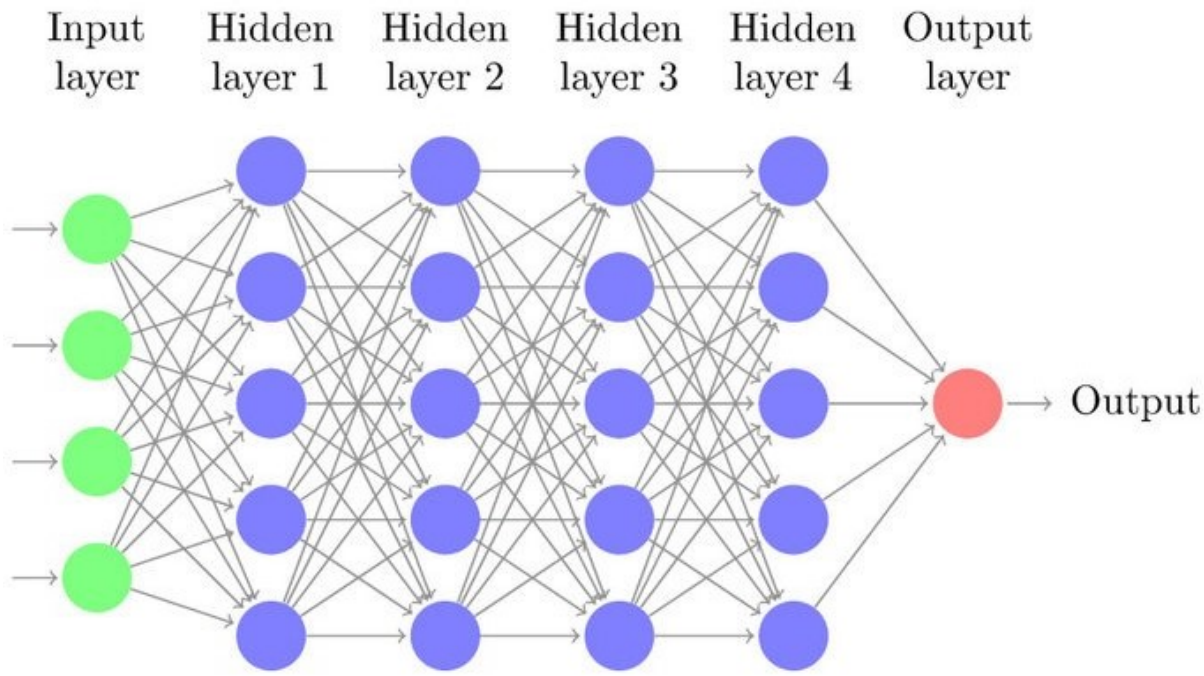


Figura 15: Diagrama de una red MLP formada por Representación de un MLP con cuatro entradas y cuatro capas ocultas

Una vez que se ha establecido la estructura de la red neuronal, se procede a la etapa de propagación hacia adelante [45]. En esta fase, el proceso de suma ponderada y activación se repite capa por capa. Este proceso permite que la red calcule una salida basada en los datos de entrada y los parámetros de la red, que incluyen tanto los pesos (w_i) como los sesgos (b).

Una vez obtenida la salida predicha, se compara con la salida deseada, lo que proporciona una medida de error. Esta medida se expresa comúnmente como una función de pérdida. Se encarga de evaluar qué tan bien la red está realizando la tarea.

La comparación entre la salida predicha y la salida deseada da lugar a la retropropagación del error. Este proceso, se explicará en detalle en la subsección 3.2 junto con la teoría de optimización.

3.2. Teoría de optimización en el Aprendizaje Profundo

La optimización en el aprendizaje profundo es una parte fundamental de la capacitación de modelos de redes neuronales. Se refiere al proceso de ajustar los parámetros (conjunto de pesos y sesgos) de una red neuronal de manera que minimice una función de pérdida.

Una vez se conoce el error mediante el proceso de propagación hacia adelante, este será utilizado para calibrar los parámetros para así modelar la red neuronal y conseguir mejorarla, es aquí donde entra el método más común, el *descenso del gradiente*. [46]

El descenso del gradiente se encarga de encontrar los valores óptimos de los parámetros de un modelo para minimizar la función de pérdida. Esto se hace ajustando los parámetros en pequeños pasos en la dirección en la que la función de pérdida disminuye rápidamente. Esta dirección se determina mediante cálculo del gradiente de la función de pérdida con respecto a los parámetros. La función del gradiente trata de indicar la dirección y la magnitud del cambio más pronunciado en la función de pérdida.

Por lo tanto, el algoritmo del descenso del gradiente, que se encarga de encontrar los valores óptimos de los parámetros para minimizar la función de pérdida, se basa en la siguiente fórmula de actualización de parámetros.

$$\theta = \theta - \alpha \cdot \nabla J(\theta) \tag{1}$$

- θ representa los parámetros del modelo.
- α es la tasa de aprendizaje, que determina el tamaño de los pasos que se dan por iteración.
- $\nabla J(\theta)$ es el gradiente de la función de pérdida $J(\theta)$ con respecto a los parámetros θ.

Teniendo en cuenta la función descrita, es importante explicar detalladamente qué es la tasa de aprendizaje, puesto que este elemento tiene un impacto significativo en la efectividad del algoritmo.

Como bien se ha descrito antes la *tasa de aprendizaje* [40] determina el tamaño de la magnitud de las actualizaciones de los parámetros que se dan por iteración; es decir, esta representa la velocidad con la que el algoritmo se mueve hacia los valores óptimos de los parámetros que minimizan la función de pérdida. Esto hace que la elección de este hiperparámetro sea importante.

Figura 16: Efecto de una tasa de aprendizaje grande (izquierda) y tasa de aprendizaje pequeña (derecha) en la optimización de una red neuronal a la hora de buscar el mínimo de la función de forma iterativa

La figura 16 muestra dos gráficos que ilustran diferentes tasas de aprendizaje. En el gráfico izquierdo, se observa una tasa de aprendizaje excesiva, lo que ocasiona que la función no alcance el punto óptimo. Debido a que la tasa es demasiado grande, los "saltos" dados por el algoritmo también lo son, dificultando así su capacidad para converger al punto óptimo, es decir, al mínimo de la curva. Sin embargo, es importante señalar que en una función no convexa como esta, es posible que el algoritmo se quede bloqueado en picos mínimos o *puntos de silla*[6]. En contraste, en el gráfico derecho, se aprecian pasos muy pequeños, lo que puede llevar al algoritmo a converger al mínimo de manera precisa, aunque a un ritmo más lento.

Ligado al descenso del gradiente se encuentra el *backpropagation* o retropropagación del error, un algoritmo fundamental en el entrenamiento de redes neuronales. Este algoritmo se utiliza para calcular cómo los pesos de la red deben ajustarse en función del error de predicción. La idea básica detrás de la retropropagación es calcular el gradiente de la función de pérdida con respecto a los parámetros de la red neuronal, utilizando la regla de la cadena para propagar el error desde la salida de la red hacia atrás a través de todas las capas.

Para una mejor comprensión de este proceso, consideremos un ejemplo ilustrativo. Supongamos que estamos entrenando una red neuronal para reconocer imágenes de gatos. Durante la fase de propagación hacia adelante, la red procesa una imagen de entrada a través de cada una de sus capas, realizando una serie de transformaciones que culminan en una predicción sobre si la imagen contiene o no un gato. Sin embargo, esta predicción inicial puede ser incorrecta, y es aquí donde entra en juego el proceso de retropropagación del error. Este proceso utiliza la regla de la cadena para ajustar los pesos de la red neuronal.

La regla de la cadena nos permite calcular cómo varía la función de pérdida en respuesta a cambios en los pesos de la red neuronal. Este cálculo se lleva a cabo descomponiendo dicha variación en las contribuciones individuales de cada neurona en cada capa. Posteriormente, esta información se utiliza para ajustar los pesos de la red mediante el descenso

[6]**Punto silla:** Se trata de una especie de lugar intermedio en una función donde no sube ni baja de forma clara, es decir, no es ni un punto máximo ni un punto mínimo.

del gradiente, moviéndolos en la dirección que minimiza la función de pérdida.

En resumen, la retropropagación, en conjunto con la regla de la cadena y el descenso del gradiente, conforma un ciclo fundamental que permite que la red aprenda de sus errores y mejore continuamente sus predicciones.

3.3. Overfitting vs Generalización

Ahora que se ha hablado de optimización es conveniente entender el concepto de *overfitting* [55] y *generalización* [26].

***Overfitting*:**

El *overfitting* [1] o sobreajuste es un fenómeno común en el desarrollo de redes neuronales, donde un modelo se ajusta demasiado bien a los datos de entrenamiento, lo que resulta en un bajo error en los datos de entrenamiento pero un error muy alto en los datos de prueba o validación. Esto ocurre cuando el modelo captura el ruido [7] en los datos de entrenamiento en lugar de aprender las relaciones que generalizan los nuevos datos.

Algunas de las causas que producen este sobreajuste son:

- El modelo es muy grande, con demasiados parámetros o capas; esto puede derivar en aprender detalles específicos de los datos de entrenamiento, incluido el ruido.
- Cuando la cantidad de datos de entrenamiento es limitada en comparación a la complejidad del modelo, esto puede llevar a que el modelo memorice en lugar de aprender patrones.
- Debido a un entrenamiento excesivo; es decir, la red ve los mismos datos muchas veces, es por ello que existe el riesgo de que el modelo de adapte en exceso a los datos de entrenamiento.

El *overfitting* acarrea varios efectos negativos como por ejemplo, la baja generalización. Esto provoca que el modelo sea incapaz de realizar predicciones válidas y precisas en el mundo real. Por lo tanto, se producirá un choque de realidades al comparar la eficacia del modelo durante el entrenamiento, donde se observará una alta precisión, en contraste con el bajo rendimiento que se reflejará en las pruebas finales como se puede comprobar en la figura 17.

[7] **Ruido:** Fluctuaciones impredecibles en los datos que dificultan identificar o reconocer las relaciones o patrones subyacentes.

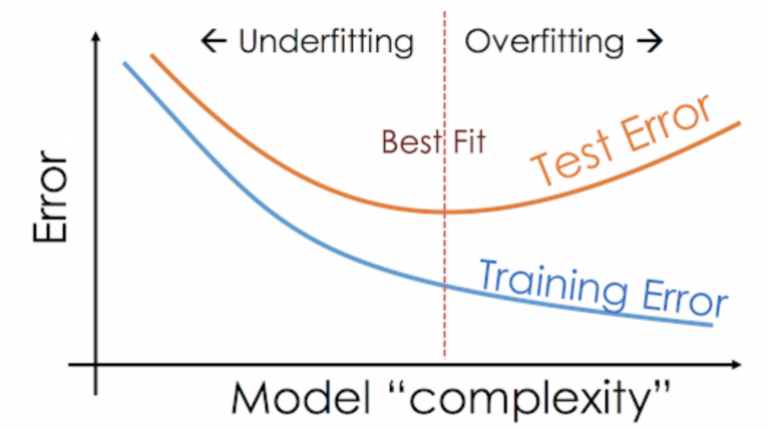


Figura 17: Representación gráfica del Overfitting

Por tanto, ¿cómo se podría mitigar el *overfitting*? A continuación se verán varias estrategias que ayudarán a disminuirlo.

- La recopilación de más datos y más variados puede ayudar a la red a aprender patrones más generales y así evitar la memorización.
- Utilizar técnicas de regularización, por ejemplo, **Dropout** [44], la cual impide que algunas neuronas se activen durante el entrenamiento, lo que reduce la dependencia entre las neuronas.
- Utilización del *early stopping* en el que se detiene el entrenamiento cuando el error de validación comienza a aumentar en lugar de disminuir, lo que indica el inicio del *overfitting*.

Generalización:

La generalización es la meta a lograr en un modelo predictivo, puesto que como bien se ha comentado antes la finalidad del sistema no es la de memorizar ejemplos o datos sino la de extraer aquellos patrones que le ayuden a aprender para así lograr unas mejores predicciones.

Un ejemplo práctico para entender la diferencia de estos conceptos y cómo afectarían sería el siguiente. Suponiendo que se tiene un modelo de clasificación de imágenes, el cual se entrena con un conjunto de entrenamiento que incluye imágenes de gatos y perros en diversas poses, tamaños y fondos, la red se encargará de aprender a detectar características relevantes, como las orejas puntiagudas de los gatos y las orejas más redondeadas de los perros, las colas largas de algunos gatos y el pelaje corto de algunos perros.

Cuando se evalúa en el conjunto de prueba con nuevas imágenes, el modelo debe ser capaz de reconocer gatos y perros, incluso si las imágenes no son idénticas a las del conjunto de entrenamiento. Por ejemplo, debería generalizar lo que ha aprendido sobre las

características distintivas de los gatos y perros para clasificar una imagen de un gato en una nueva pose o fondo diferente.

La capacidad de la red para realizar clasificaciones de manera correcta en nuevas imágenes demuestra su capacidad de generalización. Si el modelo ha aprendido a extrapolar aquellos patrones será capaz de clasificar nuevas imágenes de esos animales en diversas situaciones del mundo real, lo que lo convierte en un modelo útil y robusto.

3.4. Redes convolucionales

Las *redes neuronales convolucionales* [29], conocidas como CNN (Convolutional Neural Networks) son una clase especializada de redes neuronales diseñadas para procesar datos como imágenes o vídeos. Las CNN han sido una revolución y han logrado un gran éxito en aplicaciones como el reconocimiento de objetos a partir de fotografías como es el caso del ejemplo de la sección 3.3 sobre la generalización.

Su denominación se deriva de un componente fundamental, las capas de convolución. Estas capas son esenciales en las CNN, ya que están diseñadas para colaborar entre sí en la identificación y el aprendizaje de características relevantes en los datos visuales.

Componentes clave de las redes convolucionales

1. ***Capa de convolución***

   Este componente es el corazón de las CNN. En lugar de conectar cada píxel de la imagen a una neurona, esta capa utiliza *filtros de convolución* [47] para explorar y detectar patrones en regiones locales de la imagen.

   Cada filtro es un cubo pequeño con dimensiones de altura, ancho y profundidad (H x W x C), que se desliza por la imagen para realizar una operación de convolución. Inicialmente, se puede considerar como una matriz para el caso de una entrada de un canal. Sin embargo, es importante tener en cuenta que puede tener una profundidad adicional para manejar múltiples canales. Después de esta aclaración, se procede a explicar cómo funciona la operación de convolución, lo cual será complementado con la figura 18.

   - Se coloca el filtro en la esquina superior izquierda de la imagen de entrada.
   - Se multiplican los valores de los píxeles de la imagen que están debajo del filtro por los valores correspondientes del filtro.
   - Se suman estos productos para obtener un único valor.
   - Este valor se coloca en la matriz de salida, que se denomina **mapa de activaciones**.

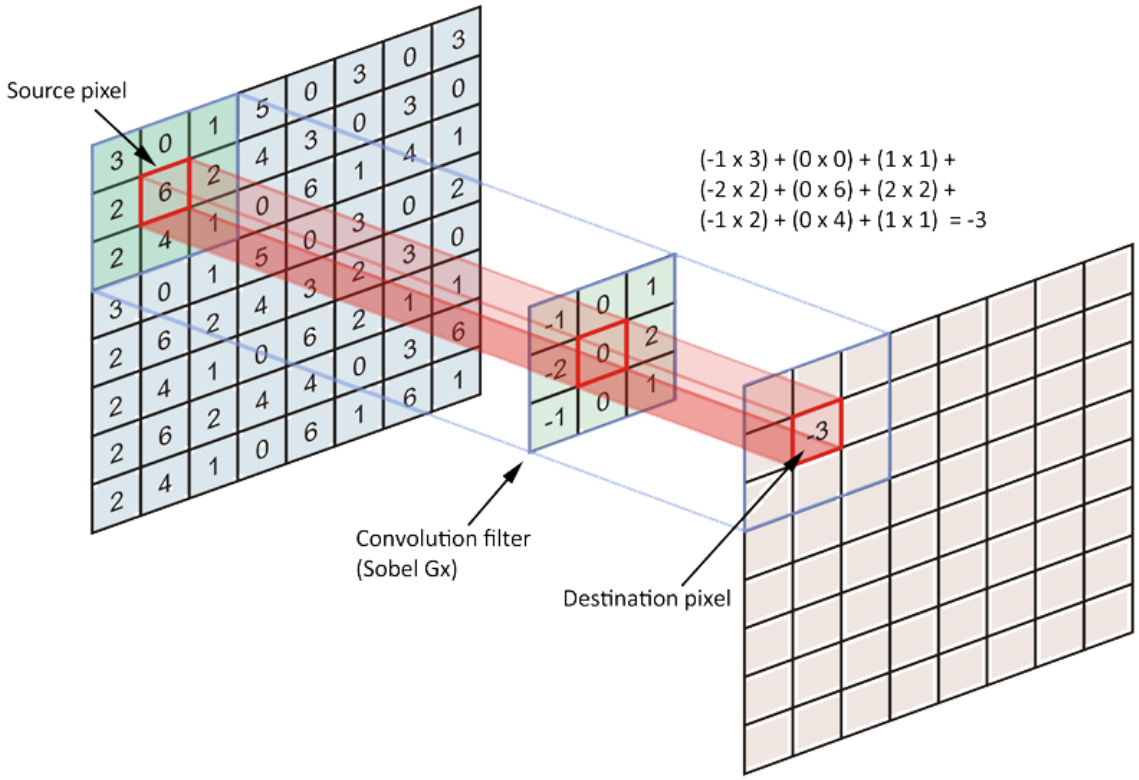


Figura 18: Ejemplo de operación de convolución

Cada filtro en una capa de convolución está diseñado para detectar características específicas en la imagen, y cada uno produce su propio mapa de características. Por ejemplo, uno de los filtros puede aprender a detectar bordes verticales, mientras que otro puede enfocarse en texturas. Estos mapas de características individuales se combinan mediante una operación llamada concatenación o superposición, donde se colocan uno al lado del otro o se superponen para formar un conjunto de características más rico y complejo. Esto permite que la red neuronal convolucional capture características complejas y abstraídas a medida que avanza en las capas.

2. ***Función de Activación***

   Como ya se ha explicado en el apartado de redes neuronales, estas redes utilizan una función de activación. En el caso de las CNN, la función que más comúnmente utilizada se trata de la ya comentada función ReLU. Esta permite que el sistema pueda discernir entre áreas de la imagen que contienen características relevantes y áreas que no. Además, permite que aprenda a representar patrones más complejos y no lineales en los datos.

3. ***Capa de Agrupación (Pooling)***

   Las capas de agrupación se utilizan para reducir la dimensión espacial de los mapas de características generados por la capa de convolución. El método más común es el

max-pooling [33]. Este reduce la cantidad de datos que se procesan a la vez que se conservan las características más importantes de la imagen.

El *max-pooling* se aplica comúnmente después de una o varias capas de convolución debido a que cada capa de convolución genera mapas de características. Estos mapas pueden ser muy grandes en términos de dimensiones espaciales, lo que aumenta significativamente la cantidad de parámetros y cálculos requeridos en la red. Por lo tanto, para reducir la dimensión de los mapas de características y disminuir la carga computacional, se aplica el *max-pooling*. Esta operación selecciona el valor máximo de un vecindario en cada región del mapa, conservando así las características más relevantes mientras reduce el tamaño de la entrada para las capas posteriores.

Este proceso se implementa utilizando una ventana deslizante, similar a la utilizada por los filtros convolucionales, con dimensiones como 2x2 o 3x3. A medida que esta ventana se desplaza por la matriz, se lleva a cabo la operación de *max-pooling*. Para un ejemplo visual de esta operación ver la figura 19.

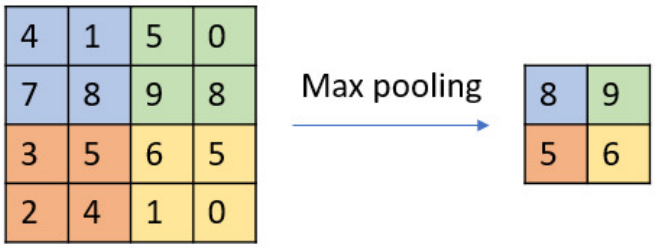


Figura 19: Ejemplo de max-pooling

4. ***Capas totalmente conectadas***

Después de varias capas de convolución y agrupación, la red suele terminar con capas completamente conectadas. Estas funcionan como en una red neuronal tradicional y utilizan la información de las características extraídas para realizar la clasificación final.

5. ***Capas de salida***

La capa de salida, situada al final de la red, es crucial para la tarea que la red neuronal está diseñada para realizar. En el caso de las tareas de clasificación, esta capa puede utilizar funciones de activación como el *softmax* para asignar probabilidades a las diferentes clases a las que pertenecen los datos de entrada. Por ejemplo, en el caso de clasificar imágenes de animales, el *softmax* podría asignar la probabilidad de que una imagen sea de un gato, perro u otro animal.

Por otro lado, en las tareas de regresión, el objetivo es predecir un valor numérico en lugar de clasificar entre diferentes categorías. En este caso, la capa de salida produce directamente un valor numérico que representa la predicción deseada. Por

ejemplo, en la predicción de ventas, la capa de salida podría generar un número que represente la cantidad esperada de ventas para un determinado período de tiempo.

3.5. Transformers

Los *transformers* son una arquitectura de red neuronal propuesta por *Vaswani et al.* en su artículo *Attention Is All You Need* [51]. Su innovación principal es el mecanismo de atención. Este mecanismo permite a la red enfocarse en partes específicas de la entrada durante el proceso de procesamiento o predicción. En esencia, asigna pesos a diferentes partes de la entrada en función de su relevancia para la tarea en cuestión, lo que mejora la capacidad de la red para modelar relaciones y dependencias entre elementos en secuencias de datos como texto o series temporales. Esto hace que los *transformers* sean altamente paralelizables y eficientes, ya que el mecanismo de atención permite que todas las unidades de atención procesen la información de manera independiente y simultánea.

La autoatención (*self-attention*) es una técnica que permite a la red enfocarse en partes relevantes de la entrada. Es un componente fundamental en las arquitecturas *transformer* y se utiliza para capturar relaciones y dependencias de datos secuenciales como texto. Es bastante eficaz para comprender el contexto y las relaciones en secuencias de datos.

La autoatención se basa en calcular un peso o puntuación para cada elemento en la entrada en función de su relación con otros elementos; estos pesos se utilizan para ponderar los valores correspondientes a cada elemento. A continuación, se calcula una combinación lineal de los valores de los elementos, utilizando los pesos calculados previamente, lo que genera una representación agregada de la secuencia. Esta representación captura la información relevante de toda la secuencia.

El cálculo de atención se repite varias veces por cada capa del modelo, permitiendo que la red capture relaciones y dependencias más complejas entre los elementos de la secuencia.

Dentro de cada capa, se utilizan conexiones residuales[8] y normalización de capa[9] para estabilizar el entrenamiento y facilitar el flujo de información. Un ejemplo visual de como es una aquitectura transformer se puede encontrar en la figura 20

[8]**Conexiones residuales:** Técnica utilizada para facilitar y estabilizar el entrenamiento de modelos. La idea es proporcionar una ruta directa para el flujo de información a través de las capas.

[9]**Normalización de capa:** Técnica utilizada para ajustar entradas o las activaciones en una red neuronal con el objetivo de mejorar el entrenamiento y la estabilidad del entrenamiento.

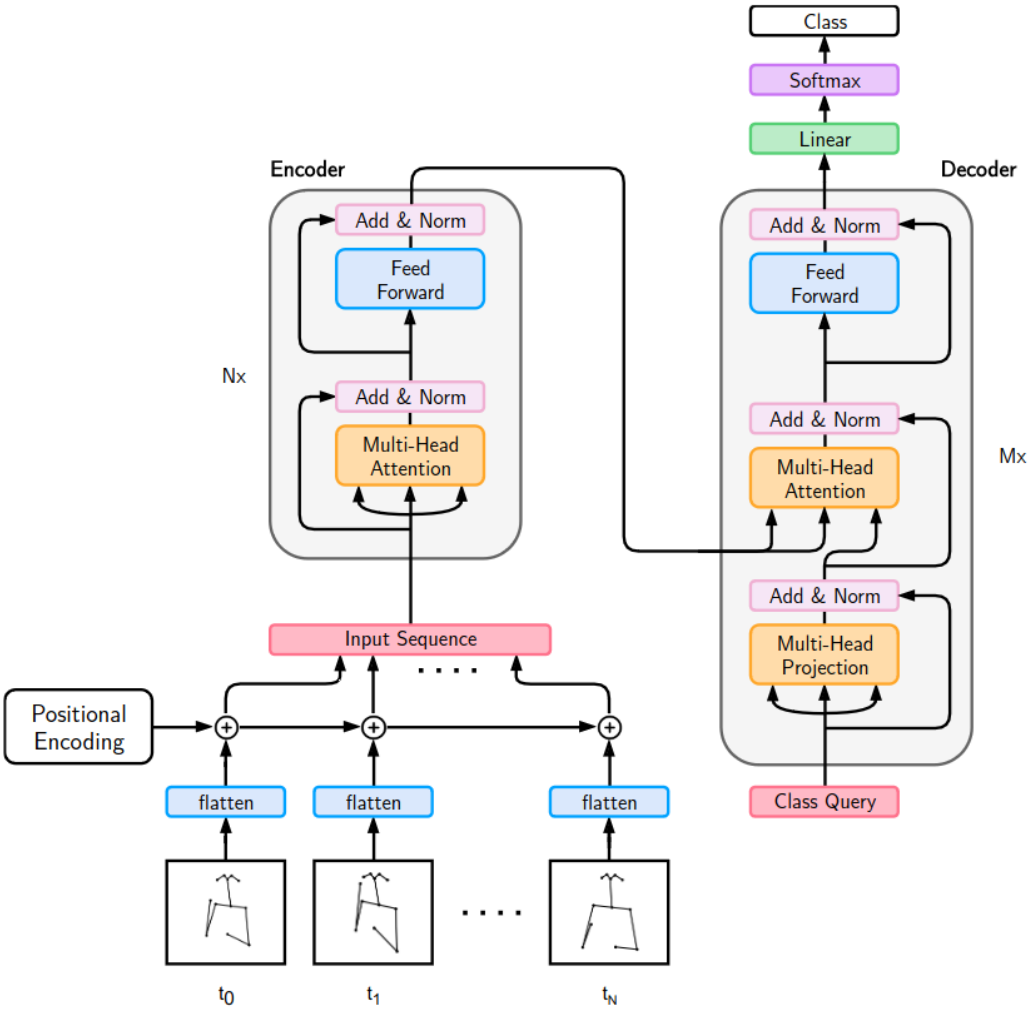


Figura 20: Ejemplo de arquitectura *transformer* encargada de procesar información secuencial como texto. Se compone de un *encoder* (Codificador) y un *decoder* (Descodificador), cada uno con capas que usan atención para capturar relaciones entre diferentes partes de la secuencia de entrada. El *encoder* procesa la información de entrada, mientras que el *decoder* genera la salida

Un ejemplo de aplicación de la arquitectura *transformer* sería por ejemplo *ChatGPT*[10], uno de los modelos transformer más grandes y avanzados.

Teniendo en cuenta esto, cuando se desea que *ChatGPT* genere un párrafo de texto sobre cualquier tema, el usuario proporciona una breve instrucción y el modelo generará el texto automáticamente como se puede ver en la figura 21. ¿Y cómo es eso posible? Esto es debido a que el modelo es capaz de captar el contexto y las relaciones semánticas en el texto, lo que permite generar respuestas significativas.

Además, se ha utilizado en una variedad de aplicaciones como traducción automática, generación de contenido, respuesta a preguntas, resumen de texto y más.

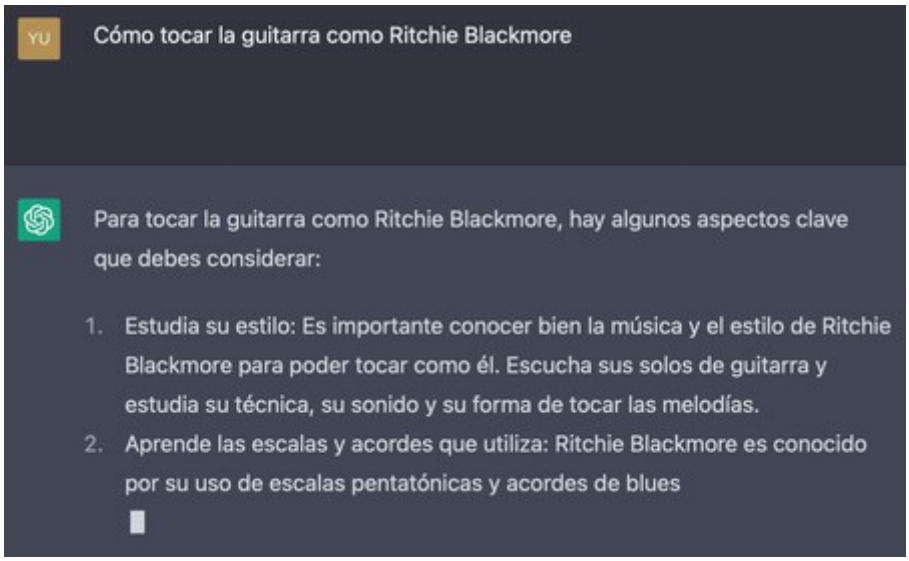


Figura 21: Ejemplo de una conversación pidiéndole a ChatGPT que le explique cómo tocar la guitarra como Ritchie Blackmore

[10]**ChatGPT:**[36] Producto comercial que tiene debajo un modelo preentrenado en una gran cantidad de texto de Internet y ha aprendido a entender y generar texto en varios idiomas y para diversas tareas de procesamiento de lenguaje natural.

4. Planificación

En esta sección, se mostrará cual ha sido la planificación de este proyecto. Comenzando con el alcance del proyecto, se continuará con los posibles riesgos durante la ejecución del proyecto y la gestión para solventarlos, y se finalizará con la evaluación económica del proyecto.

4.1. Alcance

El alcance del proyecto comprende el desarrollo de una herramienta de reconocimiento de lengua de signos mediante el uso de un modelo basado en *transformer* implementado con PyTorch. Esta herramienta permitirá a los usuarios cargar vídeos de lengua de signos y obtener las glosas correspondientes en texto.

El proyecto se divide en varias etapas, que abarcan desde la planificación inicial hasta la evaluación final del trabajo realizado. A lo largo de estas etapas, se llevarán a cabo las siguientes actividades:

- **Planificación del proyecto:**
  En este primer bloque de trabajo se realiza todo aquello que tiene que ver con la propia organización. En este bloque se tratan asuntos como las reuniones a realizar, establecimiento de objetivos y seguimiento del trabajo realizado, entre otros.
- **Formación:**
  En este segundo bloque se trabaja la formación como bien indica el nombre del apartado. La formación cuenta con diferentes pasos:
  - *Comprensión y familiarización con la lengua de signos*
  - *Formación sobre el aprendizaje automático (incluyendo redes neuronales)*
  - *Aprendizaje sobre aprendizaje profundo (incluyendo redes neuronales convolucionales y transformers)*
  - *Aprendizaje sobre el uso del software y sus librerías (Python, PyTorch, etc.)*

  Esta formación será de gran ayuda para comprender la problemática. Además, ayudará a establecer algunos de los objetivos del proyecto debido a la necesidad de entender el funcionamiento y el trasfondo de temas como la lengua de signos y el uso del aprendizaje profundo para abordar el problema.
- **Exploración del estado de arte:**
  En este tercer bloque de trabajo, se adquieren los conocimientos necesarios para dar perspectiva al desarrollo del modelo.
- **Desarrollo del proyecto:**
  Este bloque se inicia con el desarrollo del modelo. En él se recogerán las explicaciones pertinentes al desarrollo de la herramienta, al conjunto de datos utilizado, el entorno de experimentación en el que se ha trabajado y la descripción del modelo desarrollado.

- **Pruebas y resultados:**
  En este cuarto bloque, teniendo en cuenta el desarrollo realizado, se realizarán las pruebas necesarias para así demostrar el correcto funcionamiento del modelo realizado.
- **Memoria:**
  En este último bloque, se escribe y se revisa la memoria del proyecto.

4.1.1. EDT (Estructura de Desglose de Trabajo)

A continuación, se presenta con la figura 22 el diagrama EDT desarrollado, en el cual se encuentran los seis bloques principales con sus paquetes de trabajo correspondientes que dividen el proyecto en seis etapas, como bien se ha comentado en la subsección 4.1.

Para obtener una visión más detallada de los bloques de trabajo que conforman el EDT, se puede consultar en la sección A (Anexo I).

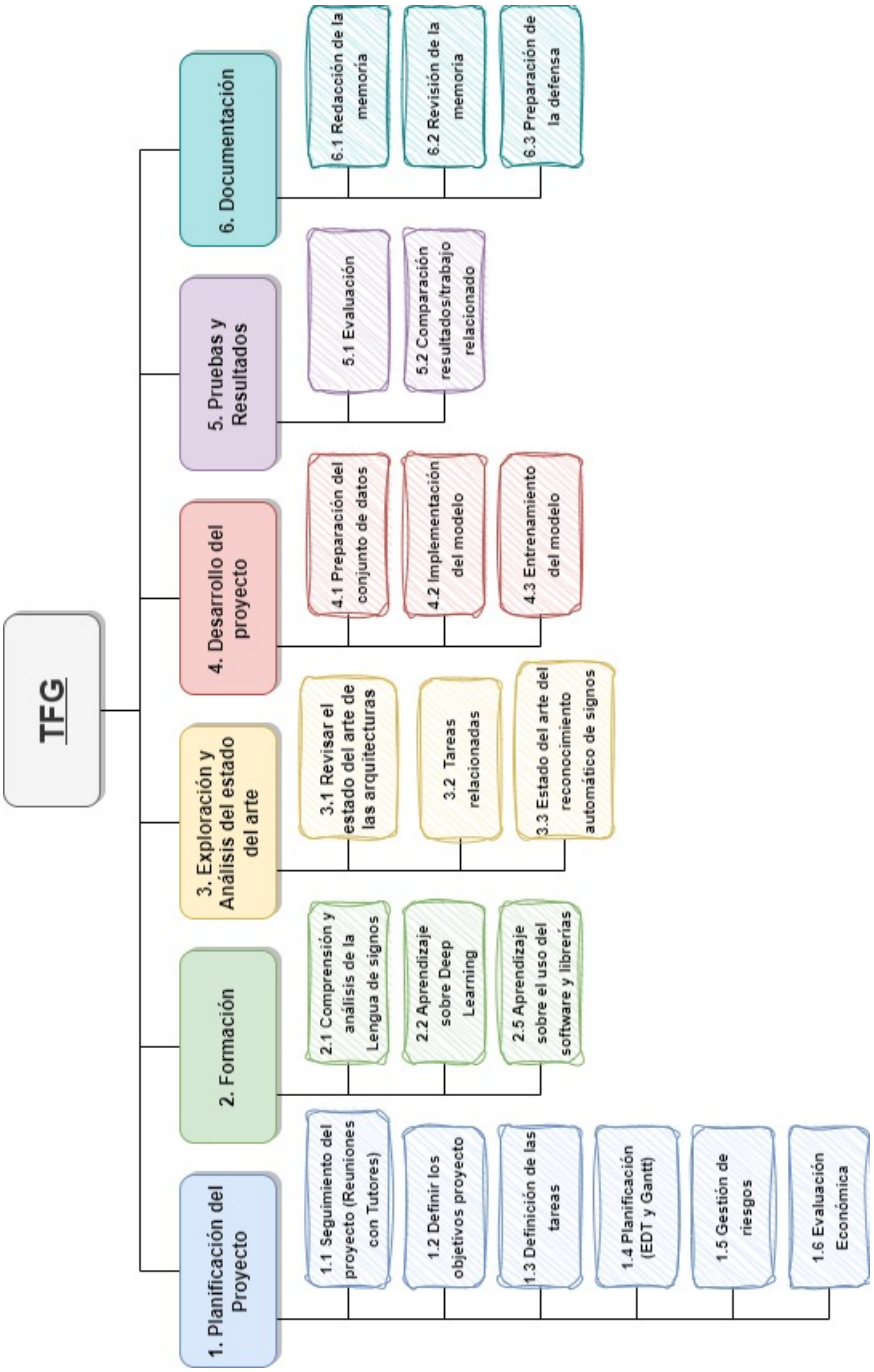


Figura 22: Estructura de Desglose de Trabajo del proyecto

4.1.2. Planificación temporal - Diagrama de Gantt

En esta subsección se encuentra el diagrama de Gantt (Figura 23 y 24), dividido en dos partes para mejorar la legibilidad, junto con la tabla 1 que resume la planificación temporal en horas y días.

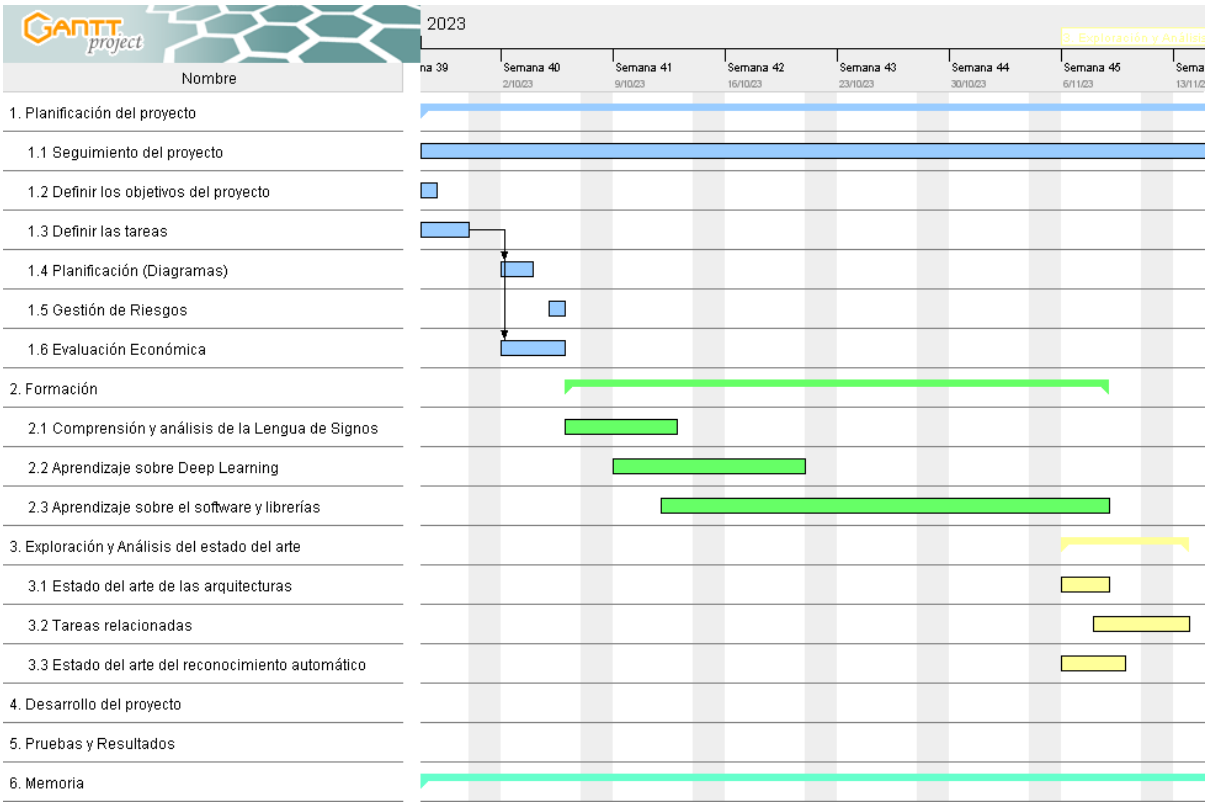


Figura 23: Primera parte del diagrama de Gantt (27/09/2023 - 13/11/2023)

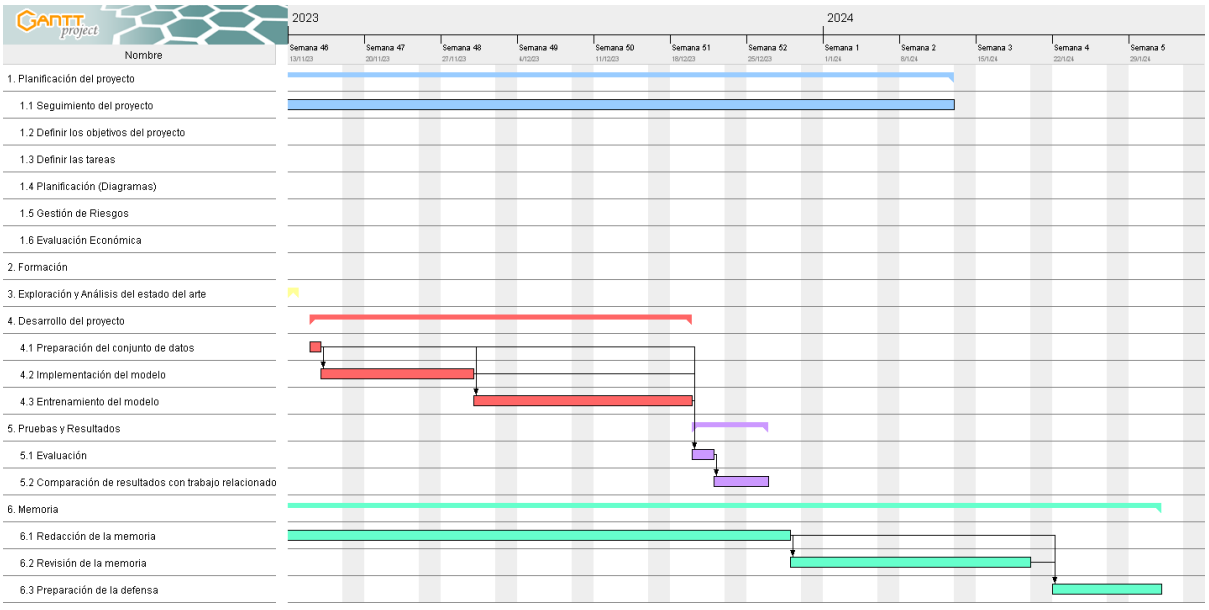


Figura 24: Segunda parte del diagrama de Gantt (14/11/2023 - 29/01/2024)

Fase/Tarea	Esfuerzo (Horas)	Periodo (Días)
1. Planificación del Proyecto		
1.1 Seguimiento del proyecto (Reuniones con tutores)	25	78
1.2 Definir los objetivos del proyecto	2	1
1.3 Definir las tareas	3	3
1.4 Planificación (Desarrollo EDT y Gantt)	2	2
1.5 Gestión de Riesgos	1	1
1.6 Evaluación Económica	3	4
2. Formación		
2.1 Comprensión y análisis de la Lengua de Signos	10	5
2.2 Aprendizaje sobre Aprendizaje Profundo	12	10
2.3 Aprendizaje sobre el uso de software y librerías	15	20
3. Exploración y Análisis del estado del arte		
3.1 Revisar el estado del arte de las arquitecturas	6	3
3.2 Tareas relacionadas	9	4
3.3 Estado del arte del reconociemiento automático de signos	8	4
4. Desarrollo del proyecto		
4.1 Preparación del conjunto de datos	4	1
4.2 Implementación del modelo	30	10
4.3 Entrenamiento del modelo	40	14
5. Pruebas y Resultados		
5.1 Evaluación	20	2
5.2 Comparación resultado con trabajo relacionado	6	3
6. Memoria		
6.1 Redacción de la memoria	70	67
6.2 Revisión de la memoria	20	16
6.3 Preparación de la defensa	14	8
Total	300	256

Tabla 1: Representación de la planificación temporal en días y horas del proyecto, dividida por los bloques presentados en el diagrama EDT desarrollado

4.2. Gestión de riesgos

La gestión de riesgos es una parte fundamental de cualquier tipo de proyecto. Por ello, en el siguiente apartado se mostrarán las tablas que contienen los diferentes riesgos identificados en el contexto de este proyecto. Las tablas están diferenciadas en tres bloques:

- Riesgos de Gestión: Representan los riesgos generados por fallos en la planificación o por una mala planificación establecida.
- Riesgos Internos:
  Son los riesgos generados por parte del desarrollador en este caso, como podría ser la salud del mismo o los conocimientos a cerca de la materia sobre la que se realizará el proyecto.
- Riesgos Externos:
  Representan los riesgos generados por fuentes externas.

Dentro de cada bloque cada tabla está dedicada a un riesgo específico. En ellas se obtiene una descripción detallada junto con la probabilidad de ocurrencia y su impacto potencial

sobre el proyecto. Además, se proponen planes de contingencia y estrategias de prevención adecuadas a cada riesgo identificado.

Cabe destacar que esta lista únicamente es un reflejo de los desafíos que se afrontarán a lo largo del desarrollo del proyecto y no quiere decir que sean los únicos riesgos que podrían manifestarse. Se deberá realizar una evaluación continua del proceso para poder prevenir riesgos todavía desconocidos.

4.2.1. Riesgos de Gestión

Exceder el tiempo de una tarea	
Descripción:	Tardar más tiempo de lo estimado en alguno de los paquetes de trabajo desarrollados.
Plan de Prevención:	Llevar a cabo un análisis de todo lo que se necesita preparar y saber antes del comienzo de la tarea para facilitar el desarrollo de la misma.
Plan de Contingencia:	Reajustar la planificación y realizar el trabajo en horas extra a las ya establecidas.
Probabilidad:	Alta.
Impacto:	Muy Alto.

Tabla 2: Riesgo de exceder el tiempo de una tarea

Falta de comunicación	
Descripción:	La comunicación en las reuniones de punto de control es insuficiente o poco eficaz, lo que dará resultado a futuras dudas y malentendidos.
Plan de Prevención:	Antes de las reuniones dejar apuntado qué puntos son los que se van a tratar o las dudas y durante las reuniones anotar los puntos que se desarrollan para dejar constancia. Aumentar la frecuencia de las reuniones o el contacto, por ejemplo, vía mail/chat.
Plan de Contingencia:	Se intentará solucionar lo antes posible ya sea por mail o por una reunión online express.
Probabilidad:	Baja.
Impacto:	Medio.

Tabla 3: Riesgo de falta de comunicación

Falta de tiempo en el desarrollo de la memoria	
Descripción:	Debido a que se ha dedicado demasiado tiempo al desarrollo práctico del proyecto, se deja de lado el desarrollo teórico (memoria).
Plan de Prevención:	Avanzar de manera "paralela" para no abandonar la redacción de la memoria.
Plan de Contingencia:	Se reservarán una serie de horas/días para avanzar con la redacción de la memoria.
Probabilidad:	Media-Baja
Impacto:	Alto.

Tabla 4: Riesgo de falta de tiempo en el desarrollo de la memoria

4.2.2. Riesgos Internos

Baja por enfermedad	
Descripción:	Indisponibilidad ya sea por enfermedad vírica o por dolencia, la cual queda examinada por un médico.
Plan de Prevención:	Plantear una planificación de reserva para evitar que el desarrollo del trabajo se retrase demasiado.
Plan de Contingencia:	Acudir al médico cuanto antes para poner freno a la enfermedad o dolor.
Probabilidad:	Baja.
Impacto:	Medio-Alto.

Tabla 5: Riesgo de baja por enfermedad

Afectación de la salud mental (sobrecarga de estrés)	
Descripción:	Debido a que la elaboración del proyecto se realiza de manera paralela a una jornada laboral de 8h, esta sobrecarga de trabajo puede generar un estrés acumulado.
Plan de Prevención:	Estimar y definir tiempos de descanso entre los desarrollos del trabajo y los del proyecto.
Plan de Contingencia:	Definir unos días de descanso que permitan no retrasar mucho el desarrollo del proyecto.
Probabilidad:	Media.
Impacto:	Medio.

Tabla 6: Riesgo de afectación de la salud mental

Exceso de carga de trabajo	
Descripción:	Teniendo en cuenta las fechas establecidas para la entrega del proyecto, esto puede derivar inconscientemente a un exceso de carga de trabajo en algún momento del desarrollo.
Plan de Prevención:	Realizar una buena planificación desde el primer día y consultarla con los tutores.
Plan de Contingencia:	Intentar dividir la carga en diferentes días para evitar una sobrecarga de trabajo acumulado.
Probabilidad:	Baja.
Impacto:	Bajo.

Tabla 7: Riesgo de exceso de carga de trabajo

Carencia de conocimientos de la materia/Herramientas	
Descripción:	Debido al no tratamiento de la materia durante el grado y el corto periodo de tiempo de realización del proyecto, a veces se pueden dar casos de carencia de los conocimientos en algún apartado del proyecto.
Plan de Prevención:	Establecer qué formaciones son interesantes y necesarias para poder entender las herramientas/materias.
Plan de Contingencia:	Establecer una reunión con los tutores para solventar el problema.
Probabilidad:	Media-Baja.
Impacto:	Bajo.

Tabla 8: Riesgo de carencia de conocimientos de la materia/Herramientas

4.2.3. Riesgos Externos

Pérdida de conexión a internet	
Descripción:	Debido a la pérdida de conexión a internet, pueden surgir diferentes problemas como la limitación de búsqueda de información o la ejecución y entrenamiento del modelo puesto que va ligado a servidores remotos.
Plan de Prevención:	Contratar una línea de internet de suficiente capacidad y estable.
Plan de Contingencia:	Utilizar los datos móviles durante el periodo de tiempo que se produzca la pérdida de conexión y contactar con la empresa de telecomunicaciones pertinentes.
Probabilidad:	Muy Baja.
Impacto:	Bajo.

Tabla 9: Riesgo de pérdida de conexión a internet

Problemas con el hardware/equipo informático	
Descripción:	Problemas surgidos por agentes externos como caída del equipo o la caída de líquidos sobre el mismo.
Plan de Prevención:	Almacenar el equipo portátil en su funda correspondiente y no guardarlo en un lugar alto, además de mantener un espacio de trabajo limpio y no trabajar a la vez que se toma un café, agua, etc.
Plan de Contingencia:	Reemplazo de equipo informático.
Probabilidad:	Muy Baja.
Impacto:	Muy Alto.

Tabla 10: Riesgo de sufrir problemas con el hardware/equipo informático

Pérdida de datos	
Descripción:	Perder la información del proyecto parcialmente o completamente.
Plan de Prevención:	Siempre tener tres copias de seguridad (una local y 2 en la nube).
Plan de Contingencia:	Restaurar la información perdida a través de las copias de seguridad realizadas.
Probabilidad:	Media.
Impacto:	Muy Alto.

Tabla 11: Riesgo de pérdida de datos

4.3. Evaluación Económica

En este apartado se pretende realizar una estimación del presupuesto de los costes del desarrollo derivados del proyecto. En particular, se evalúan los costes asociados a la adquisición del hardware y software y cualquier otro gasto relevante.

Para realizar una evaluación detallada, se lleva a cabo un análisis de los componentes económicos que se ven comprometidos en el proyecto. Estos componentes pueden tratarse tanto de elementos físicos tales como un equipo informático de alto rendimiento como la mano de obra o la adquisición de licencias de software necesarias para el desarrollo del modelo. Además, se considerarán los gastos de mantenimiento y actualización de los recursos mencionados.

Asimismo, se tendrán en cuenta los costes asociados al consumo energético durante el periodo de desarrollo junto con el gasto del uso de servidores remotos para un mayor rendimiento en las ejecuciones y pruebas del modelo.

Finalmente es importante destacar que al tratarse de un trabajo de fin de grado de la universidad todos los cálculos realizados son teóricos ya que no se dispone de presupuesto alguno.

4.3.1. Costes de mano de obra

Para realizar el cálculo de los gastos por mano de obra, es necesario comentar que un desarrollador en aprendizaje automático junior (recién graduado) tiene un sueldo anual promedio de 23.283€ brutos [23]. Además, la jornada máxima anual determinada por el ministerio de trabajo y economía social en la *Resolución de 13 de julio de 2023* [9] es de 1.800 horas anuales de trabajo efectivo. El salario por hora quedaría de la siguiente manera:

$$Salario/Hora = \frac{Salario\,anual}{Horas\,máximas\,anuales} = \frac{23283\,euros/anual}{1800\,horas/anual} = \mathbf{12{,}93€} \quad (2)$$

En este proyecto se ha estimado que se trabajará 570 días (teniendo en cuenta el cuadro de planificación), si se estiman un promedio de cinco horas y media diarias en total, serán unas 300 horas. Teniendo el cuenta el salario por hora obtenido del cálculo anterior:

$$Salario/Hora = Horas\,proyecto \cdot Salario\,por\,hora = 300\,horas \cdot \frac{12,93\,euros}{1\,hora} = \mathbf{3879€} \quad (3)$$

4.3.2. Costes en Hardware

En cuanto al hardware, a día de hoy estos son los precios de los dispositivos utilizados:

- **Ordenador portátil**: 579€ [2]
- **Servidor GPU remota**: Gratuito**[11].

El ordenador portátil que se utilizará es de uso personal. Por otro lado, el acceso a servidores será proporcionado por los directores del proyecto. El coste del hardware es de **579€**.

4.3.3. Costes en Software

En cuanto al software y las licencias de uso:

- **Pycharm [Versión pro]**: Gratuito, licencia de estudiante. [24]
- **Visual Studio Code**: Gratuito. [32]
- **Google colabs**: Gratuito. [7]
- **Overleaf**: Gratuito, se ha escogido el plan base gratuito. [39]
- **GanttProject**: Gratuito. [16]
- **Github**: Gratuito. [17]
- **Google Drive**: Gratuito. [12]

[11]** En servicios de nube como AWS, Google Cloud y Azure, el coste de entrenamiento podría oscilar entre 0.10€ y 10€ o más por hora de uso de recursos de GPU. Teniendo en cuenta que el proyecto tendrá un tiempo de desarrollo de unas 74 horas (Preparación del conjunto de datos, implementación del modelo y el entrenamiento del modelo), el coste aproximado sería de 373,17€si se pagase una media de 5,05€/hora.

- **Draw.io**: Gratuito. [4]
- **Librerías**: Gratuito, de uso libre. [43]

Al ser todo gratuito queda como resultado un coste de **0€** en software.

4.3.4. Costes de Formación

Los gastos en formación tanto de aprendizaje profundo como de la lengua de signos son los siguientes:

- **Libro "Deep Learning", Grooking**: 8€, precio de la impresión del libro en corsetería. [50]
- **Vídeos informativos sobre redes neuronales**: Gratuito. [11]
- **Curso online de la Universidad de Stanford**: Gratuito. Lista de vídeos subidos a su canal de Youtube. [8]
- **Lengua de signos (*fingerspelling* y representación de palabras)**: Gratuito.
- **Tutorial PyTorch**: Gratuito. [42]

Por lo tanto, suman un total de **8€** de gasto en formación.

4.3.5. Costes Indirectos

Los gastos indirectos son aquellos que no influyen directamente sobre el proyecto pero se deben tener en cuenta ya que son imprescindibles. Estos gastos representan el consumo de suministros como la electricidad y la conexión a internet.

Aproximadamente el gasto total en electricidad e internet ronda entre un 2 % y un 5 % de los gastos fijos del proyecto:

$$Gastos\,indirectos = (3879€ + 579€ + 8€) \cdot 0{,}02\,\% = 89,32€ \tag{4}$$

4.3.6. Costes Totales

Teniendo en cuenta todos los costes calculados en los apartados anteriores, se puede observar en la tabla 12 el total del coste del proyecto:

Gastos Totales Estimados	
Gastos por mano de obra:	3879€
Gastos en hardware:	579€
Gastos en software:	0€
Gastos en formación:	8€
Gastos indirectos:	89,32€
TOTAL:	4555,32€

Tabla 12: Tabla de resumen de costes del proyecto

5. Desarrollo de la herramienta

En esta sección, se abordarán las diferentes etapas del desarrollo de un modelo de reconocimiento de lengua de signos. La sección se divide en la subsección 5.1, donde se hablará sobre el conjunto de datos utilizado, siguiendo con la subsección 5.2 donde se explicará el tratamiento al que han sido sometidos los datos para posteriormente poder ser utilizados y, finalmente, las subsecciones 5.3 y 5.4 en las que se trata de explicar el entorno que se ha utilizado para poder llevar a cabo el proyecto y las herramientas utilizadas, además de una descripción del funcionamiento del modelo creado.

5.1. Conjunto de datos

El grupo de datos utilizado proviene del banco de datos WLASL (Wristband-less Action Detection) [49] que contiene 21.083 ejemplos [53]. Cada muestra esta compuesta por un par formado por un vídeo y un string. En este caso el vídeo presenta a una persona realizando un signo concreto en lengua de signos estadounidense (ASL) y el string corresponde a la glosa que hace referencia a dicho signo.

El repositorio WLASL permite obtener la información correspondiente a cada vídeo de manera muy sencilla y ordenada. El archivo que se puede descargar junto al conjunto de datos (WLASL_v0.3.json) se utiliza como índice para poder clasificar los vídeos en los diferentes grupos de tratamiento (vídeos de entrenamiento, vídeos de validación, vídeos de prueba); además de indicar a qué signo corresponde cada vídeo.


```
"gloss": "book",
"instances": [
    {
        "bbox": [
            385,
            37,
            885,
            720
        ],
        "fps": 25,
        "frame_end": -1,
        "frame_start": 1,
        "instance_id": 0,
        "signer_id": 118,
        "source": "aslbrick",
        "split": "train",
        "url": "http://aslbricks.org/New/ASL-Videos/book.mp4",
        "variation_id": 0,
        "video_id": "69241"
    },
```


Figura 25: Ejemplo de la entrada "*book*" (libro) que contiene varias etiquetas como el número de FPS del vídeo, resolución del mismo, etc

Echando un vistazo a la figura 25 se puede observar como cada vídeo del conjunto de datos incluye varios metadatos pertenecientes a un único ejemplo, en este caso de *book* (libro).

- *gloss:* Transcripción de la glosa en formato texto (ver Sección 2.2).
- *instances:* Contiene un array con todas las instancias etiquetadas con la glosa indicada en *gloss*. Cada instancia incluye los siguientes metadatos:
  - *split:* Indica el conjunto de datos al que pertenece la instancia. En el caso de la figura 25, el ejemplo indicado pertenece al conjunto de datos de entrenamiento, es decir, aquellos vídeos que se utilizarán para la etapa de entrenamiento del modelo.
  - *video_id:* Proporciona el identificador del vídeo al que corresponden estos metadatos.
  - *url:* Indica el hipervínculo que contiene la fuente de la cual se ha obtenido el vídeo.
  - *fps:* Representa el número de *frames* (fotogramas) por segundo; es decir, el número de imágenes que se utilizan en un segundo del vídeo.

Siguiendo con el ejemplo de la figura 25, si se accede a la url del índice se puede reproducir el vídeo en cuestión (figura 26).

Figura 26: Video 69241.mp4 correspondiente al ejemplo mostrado en la Figura 25

5.2. Preparación del conjunto de datos

El objetivo principal es extraer la pose de la persona en cada fotograma de los vídeos. La pose se representa mediante un esqueleto compuesto por puntos clave (llamados *key points* [12]) y conexiones entre ellos, que se utiliza para capturar la postura y la disposición del cuerpo.

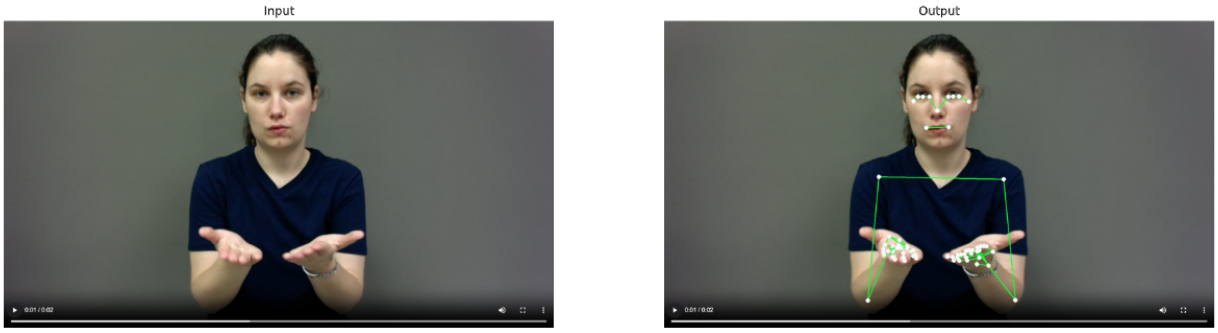


Figura 27: Ejemplo de obtención de *key points* mediante la herramienta Mediapipe. A la izquierda se puede ver un fotograma de uno de los vídeos en el que se realiza el signo que hace referencia a la glosa "*book*" (libro). A la derecha se encuentra el mismo fotograma pero en este caso se pueden observar una serie de puntos conectados por unas líneas que corresponden a los *key points* obtenidos (posición de los ojos, boca, manos y brazos)

Una vez establecido este objetivo, se procede a identificar y analizar los puntos clave específicos de interés, como la posición de las manos derecha e izquierda del sujeto. Estos puntos clave son puntos anatómicos detectados por la herramienta utilizada para el análisis de imágenes, y pueden incluir articulaciones del cuerpo humano, puntos faciales y manos, entre otros, como se puede ver en la figura 27.

Entre los puntos detectados se encuentran por ejemplo puntos faciales como el triángulo formado por los ojos y la boca, así como la estructura formada por los brazos y las falanges de las manos. Estos puntos son fundamentales para comprender y analizar la postura y los gestos realizados por el sujeto en cada fotograma de los vídeos del conjunto de datos.

Para poder obtener esos puntos se ha utilizado el software MediaPipe [20]. Se trata de un marco de trabajo desarrollado por Google que proporciona una serie de bibliotecas y herramientas que permiten aplicar rápidamente técnicas de inteligencia artificial y aprendizaje automático.

Para este proyecto en específico se ha utilizado la librería *Hollistic*, la cual permite la detección simultánea de caras, manos y poses corporales en tiempo real. Es decir, esta solución puede detectar no sólo dónde están ubicadas las manos y la cabeza de una persona en una imagen o vídeo, sino también cómo se están moviendo estas partes del cuerpo en relación con el resto del cuerpo.

[12] **Key points:** Puntos de interés detectados en una imagen y utilizados para identificar características específicas como las articulaciones del cuerpo humano o puntos clave en objetos.

Teniendo en cuenta esta breve introducción, para entender qué es MediaPipe se va a entrar un poco más en detalle en el procedimiento utilizado para la preparación del conjunto de datos. El proceso comienza extrayendo puntos clave del cuerpo humano en cada fotograma del vídeo utilizando la biblioteca MediaPipe previamente mencionada. Estos puntos clave se codifican mediante coordenadas en un sistema de referencia, con un origen definido en el centro del pecho del sujeto. Las coordenadas de los puntos clave representan las posiciones espaciales de diversas partes del cuerpo, como las articulaciones y las extremidades relativas a este origen. Por lo tanto, las coordenadas de los puntos clave no solo capturan la información de la posición en dos dimensiones (x, y), sino también la profundidad (z) en relación con el centro del pecho del sujeto. Las coordenadas se encuentran en un rango normalizado, donde las coordenadas x e y están en el rango [0, 1], representando la posición relativa dentro del marco de la imagen, mientras que la coordenada "z" puede variar y no está limitada a un rango específico.

Una vez se han extraído los puntos clave, se realiza un procesamiento detallado de los datos. Este procesamiento consta de varios pasos para la preparación de los datos de entrada al modelo de aprendizaje automático.

El primer paso consiste en adaptar los puntos clave para considerar la proporción y la escala del cuerpo humano en cada fotograma. Para ello, se reciben como entrada un conjunto de datos que contienen las coordenadas de los puntos clave y la *relación de aspecto inversa*[13] de la imagen. Se multiplican las coordenadas de los puntos clave por el inverso de la relación de aspecto para escalarlas de manera adecuada. Esto asegura que los puntos clave estén representados de manera precisa y proporcional. Se realiza debido a que diferentes personas pueden tener proporciones corporales diferentes. Es crucial para garantizar una buena capacidad de generalización en el entrenamiento del modelo. Es importante destacar que esta adaptación también puede incluir la corrección de posibles distorsiones introducidas por la perspectiva de la cámara o por la posición relativa del sujeto con respecto al sensor de imagen. Este proceso de adaptación se conoce como escalado de imagen y es esencial para garantizar la coherencia en la representación de los puntos de interés entre diferentes imágenes y sujetos.

Después del ajuste de escala, se implementa la asignación temporal para manejar situaciones donde algunos puntos clave pueden faltar o ser incorrectos en ciertos fotogramas. Esto implica la interpolación de datos entre fotogramas adyacentes para llenar los vacíos y garantizar una secuencia de datos continua y coherente; es decir, la asignación temporal "rellena" usando datos de los fotogramas cercanos aquellos puntos clave perdidos o incorrectos. Algo similar se da cuando se conectan puntos en una gráfica para encontrar el valor de un punto faltante.

Un paso clave en el análisis de movimientos humanos es la normalización de la postura. Esto significa ajustar los datos de los puntos clave del cuerpo para que sean consistentes y comparables entre diferentes personas y vídeos. Este proceso incluye varios pasos.

[13]**Relación de aspecto inversa:** Se refiere a la proporción entre la altura y la anchura de la imagen, pero expresada de manera inversa. Es decir, se calcula dividiendo la anchura entre la altura en lugar de al revés.

Ajustar las coordenadas de los puntos clave para corregir cualquier distorsión debida a la relación de altura y ancho del vídeo, rellenar los valores faltantes de los puntos clave en la secuencia de vídeo utilizando datos de los fotogramas anteriores y siguientes para interpolar o extrapolar los valores, y ajustar las posiciones de los puntos clave para que las posturas tengan proporciones y orientaciones consistentes. Este proceso asegura que los datos de los movimientos sean uniformes y fáciles de comparar y analizar.

Una vez que los datos de los puntos clave han sido ajustados para ser consistentes y comparables, se procede a la corrección de la orientación, que implica alinear los puntos clave con respecto a una referencia estándar. Posteriormente, se realiza la alineación espacial de los puntos clave para posicionarlos de manera precisa y coherente en relación con otras partes del cuerpo o elementos del entorno. Finalmente, se lleva a cabo la normalización de la escala para ajustar los puntos clave, garantizando así que las mediciones sean consistentes independientemente de la distancia a la que se encuentre el sujeto de la cámara.

Tras completar el procesamiento de los puntos clave, se puede realizar cualquier post-procesamiento adicional necesario, como el filtrado de *datos ruidosos*[14] o la conversión de los datos a un formato legible y utilizable para su entrenamiento posterior. Finalmente, la información resultante se almacena en un formato adecuado, en este caso en ficheros HDF5, los cuales permiten una mayor eficiencia de almacenamiento y un acceso rápido a los datos guardados.

5.3. Entorno de experimentación

El desarrollo del presente proyecto comprende una serie de herramientas y recursos que han sido seleccionados para garantizar la capacidad de realizar el tratamiento de datos, almacenamiento de información y entrenamiento de modelos de lengua de signos.

El software principal utilizado para la implementación y ejecución del código ha sido Visual Studio Code, un entorno de desarrollo integrado (IDE) reconocido por su versatilidad, facilidad de uso y amplia gama de extensiones disponibles. Además, ofrece una interfaz intuitiva y funcionalidades avanzadas que facilitan la escritura, depuración y ejecución de código en varios lenguajes de programación, incluido Python, que es el lenguaje principal utilizado en este proyecto.

El lenguaje de programación Python se ha convertido en el principal lenguaje utilizado en el campo del desarrollo de redes neuronales, aprendizaje automático y procesamiento de datos. Un buen ejemplo de ello son las principales librerías de aprendizaje profundo (TensorFlow y PyTorch), las cuales están desarrolladas en Python. Su popularidad radica en su amplia disponibilidad de bibliotecas especializadas y su facilidad de aprendizaje y uso. Ofrece una extensa gama de herramientas para el desarrollo de aplicaciones de inteligencia artificial, incluidas bibliotecas como TensorFlow, PyTorch y Scikit-learn, que son fundamentales para el entrenamiento y despliegue de modelos de aprendizaje automático y aprendizaje profundo. Además, Python es ampliamente utilizado en la implementación

[14] **Datos ruidosos:** Aquellos que contienen errores, variabilidad aleatoria o información irrelevante que puede interferir con el proceso de aprendizaje del modelo.

de modelos avanzados como *transformers*.

Para poder llevar a cabo el proyecto se ha hecho uso principalmente de las bibliotecas de Python: MediaPipe, OpenCV, PyTorch, Matplotlib y h5py.

Teniendo en cuenta que ya se ha realizado una introducción y explicación a MediaPipe anteriormente en la subsección 5.1, se comenzará con la explicación de las siguientes librerías mencionadas.

OpenCV es una biblioteca de visión y procesamiento de imágenes en tiempo real. Ofrece una amplia gama de funciones y algoritmos para el procesamiento y manipulación de imágenes. En este proyecto se utiliza en conjunto de MediaPipe para realizar tareas de preprocesamiento de imágenes y vídeos, como la conversión de formatos de imagen, el ajuste de tamaño y la conversión de colores.

h5py proporciona una interfaz para trabajar con el formato de archivo de datos *HDF5*. Este formato de archivo binario se diseñó para almacenar grandes conjuntos de datos de manera eficiente. En este proyecto, h5py se utiliza para almacenar y gestionar datos estructurados, como resultados de la obtención de los *key points* de cada vídeo.

PyTorch es una de las principales bibliotecas utilizadas en el aprendizaje automático que permite crear redes neuronales profundas. En este caso PyTorch se utiliza para implementar y entrenar modelos de aprendizaje profundo para el reconocimiento de patrones de movimiento humano.

Matplotlib permite crear visualizaciones y gráficos de alta calidad para representar datos de manera clara y compresible. Ofrece una amplia variedad de estilos gráficos y herramientas de personalización. En este proyecto, se utiliza para visualizar y presentar los resultados del aprendizaje del modelo y su evolución.

En cuanto al hardware utilizado, el proyecto se ejecuta en un entorno de servidor proporcionado por el centro HiTZ de la Universidad del País Vasco (UPV/EHU) [21] con acceso a diversos servidores. En este caso, este servidor ofrece recursos computacionales de alto rendimiento y está equipado con varias *GPU*[15] NVIDIA A100-SXM4-80GB, lo que permite realizar cálculos intensivos y procesamiento paralelo de manera eficiente.

En resumen, el entorno de experimentación empleado en este proyecto combina software y hardware de vanguardia para proporcionar un marco sólido para la investigación y el desarrollo en el campo del reconocimiento del movimiento humano enfocado a la lengua de signos en vídeos.

[15]**GPU:** (Unidad de Procesamiento Gráfico) es un tipo de procesador especializado diseñado para acelerar el procesamiento de gráficos y cálculos matemáticos en paralelo, comúnmente utilizado en tareas relacionadas con gráficos, inteligencia artificial y computación de alto rendimiento.

5.4. Descripción del modelo

El objetivo de este modelo de reconocimiento de lengua de signos es realizar la tarea de clasificación multiclase de signos individuales. Recibe como entrada los *key points* extraídos de los vídeos de lengua de signos y la salida será una distribución de probabilidad sobre todos los posibles signos previamente definidos.

Para comenzar, es importante comprender la arquitectura general del modelo de reconocimiento de lengua de signos y cómo se realiza la tarea de clasificación de vídeos. Se utiliza la clase *TransformerEncoder*, proveniente del modelo *transformer*, el cual es una clase de PyTorch, que se encarga de procesar la entrada y convertirla en una representación rica y contextualizada. Contiene todos los componentes necesarios para construir y ejecutar el modelo *transformer*, incluyendo capas de atención, capas totalmente conectadas y *embeddings* posicionales.

El modelo consta de una serie de bloques de codificador (ver sección 3.5). Cada bloque contiene múltiples capas de autoatención y redes neuronales totalmente conectadas que posibilitan al modelo aprender representaciones en diferentes niveles de generalización de las secuencias de datos. En este caso estas representaciones se utilizan para realizar predicciones y poder clasificar los vídeos.

Durante el proceso, se introduce un *token*[16] especial conocido como CLS. Este *token* se añade al principio de la secuencia de entrada y tiene un papel fundamental en la captura de información global de los datos. Al pasar la secuencia por el codificador, el modelo aprende representaciones que reflejan características generales de la entrada. Al final del procesamiento, el vector correspondiente al *token* CLS contiene una representación global de la secuencia, que luego se utiliza para tareas de clasificación mediante su paso por una capa completamente conectada, produciendo así las predicciones finales del modelo.

Además, se establecen los hiperparámetros del modelo, como dimensiones de entrada y ocultas, tamaño de lote, número de capas y número de cabezas de atención. Para optimizar la estructura y rendimiento del modelo se realizarán experimentos probando distintas combinaciones de estos parámetros. Los resultados de estos experimentos se detallan más adelante en la sección 6.1.

Los conjuntos de datos de entrenamiento, validación y prueba se cargan y preparan mediante un *DataLoader* personalizado con los datos del conjunto de datos WLASL. Los *DataLoader* facilitan la manipulación y el procesamiento de grandes conjuntos de datos. En este caso, se encargan de cargar los datos del conjunto de datos, aplicar operaciones de preprocesamiento como la *normalización* [17] y la *división en lotes* [18], para asegurar

[16] ***token:*** Representación vectorial de una porción de la imagen.

[17] **Normalización:** Proceso de ajustar los valores de las características de los datos a una escala común sin distorsionar las diferencias en los rangos de valores, lo que ayuda a mejorar la precisión y la eficiencia de los algoritmos de aprendizaje automático.

[18] **División en lotes:** Técnica donde los datos se dividen en grupos o "lotes" y el modelo se entrena iterativamente sobre cada lote.

que los datos estén listos para ser utilizados por el modelo.

En cuanto al entrenamiento consta de múltiples iteraciones denominadas *épocas* [19], donde se ha preestablecido un máximo a priori. En cada iteración se utiliza el conjunto de datos de entrenamiento para entrenar el modelo. El conjunto se divide en lotes y se realiza una pasada hacia adelante y hacia atrás por cada lote, así como una actualización de los parámetros tras acabar cada pasada hacia atrás (cálculo de la función de pérdida y actualización) utilizando el algoritmo de optimización Adam (Adaptive Moment Estimation). Se trata de un método de optimización para entrenar modelos que combina las ventajas de dos técnicas de optimización, *AdaGrad*[20] y *RMSProp*[21], para ajustar los parámetros del modelo de manera eficiente, mejorando la velocidad de convergencia y la estabilidad del entrenamiento.

Después de cada época, el modelo se evalúa en un conjunto de datos de validación para monitorear su rendimiento. Se calculan métricas como la precisión y la pérdida en este conjunto de datos para evaluar el desempeño del modelo. Es importante destacar que para este desarrollo se ha utilizado la técnica denominada *early stopping*. Esta estrategia consiste en detener el entrenamiento del modelo cuando el rendimiento en el conjunto de validación deja de mejorar durante un cierto número de comprobaciones definido por el parámetro de paciencia. Esto indica que el modelo ya no está generalizando bien frente a los datos no vistos, evitando así el sobreajuste.

Tras esta evaluación y una vez el entrenamiento haya terminado, se realiza una evaluación con un conjunto de datos no utilizado hasta el momento para medir su rendimiento final. En este proceso se calculan métricas tales como la precisión (top-1), la perdida y la precisión top-5. Esta última se refiere a la proporción de veces que la clase correcta aparece dentro de las cinco clases más probables predichas por el modelo. Esta métrica se considera especialmente relevante para este conjunto de datos debido a su amplia aplicación en la literatura científica, que comprende investigaciones y desarrollos validados.

Si se desea tener una idea más visual sobre como es la estructura y funcionamiento del modelo desarrollado se puede visualizar en la figura 28. Además, se puede acceder al código fuente del modelo de reconocimiento de la lengua de signos, el programa encargado de preparar los datos y el encargado de extraer los puntos clave de los vídeos mediante el siguiente hipervínculo.

https://github.com/duraan08/ASL_Transcriptor.git

[19] **Época:** Implica que el modelo ha pasado por todo el conjunto de datos una vez, actualizando sus parámetros para minimizar el error de predicción.

[20] **AdaGrad:** Algoritmo de optimización que se encarga de ajustar automáticamente la tasa de aprendizaje para cada parámetro del modelo basándose en la magnitud de los gradientes de las funciones de pérdida.

[21] **RMSProp:** Algoritmo de optimización que ajusta la tasa de aprendizaje de manera adaptativa para cada parámetro del modelo basándose en el cuadrado medio de los gradientes pasados.

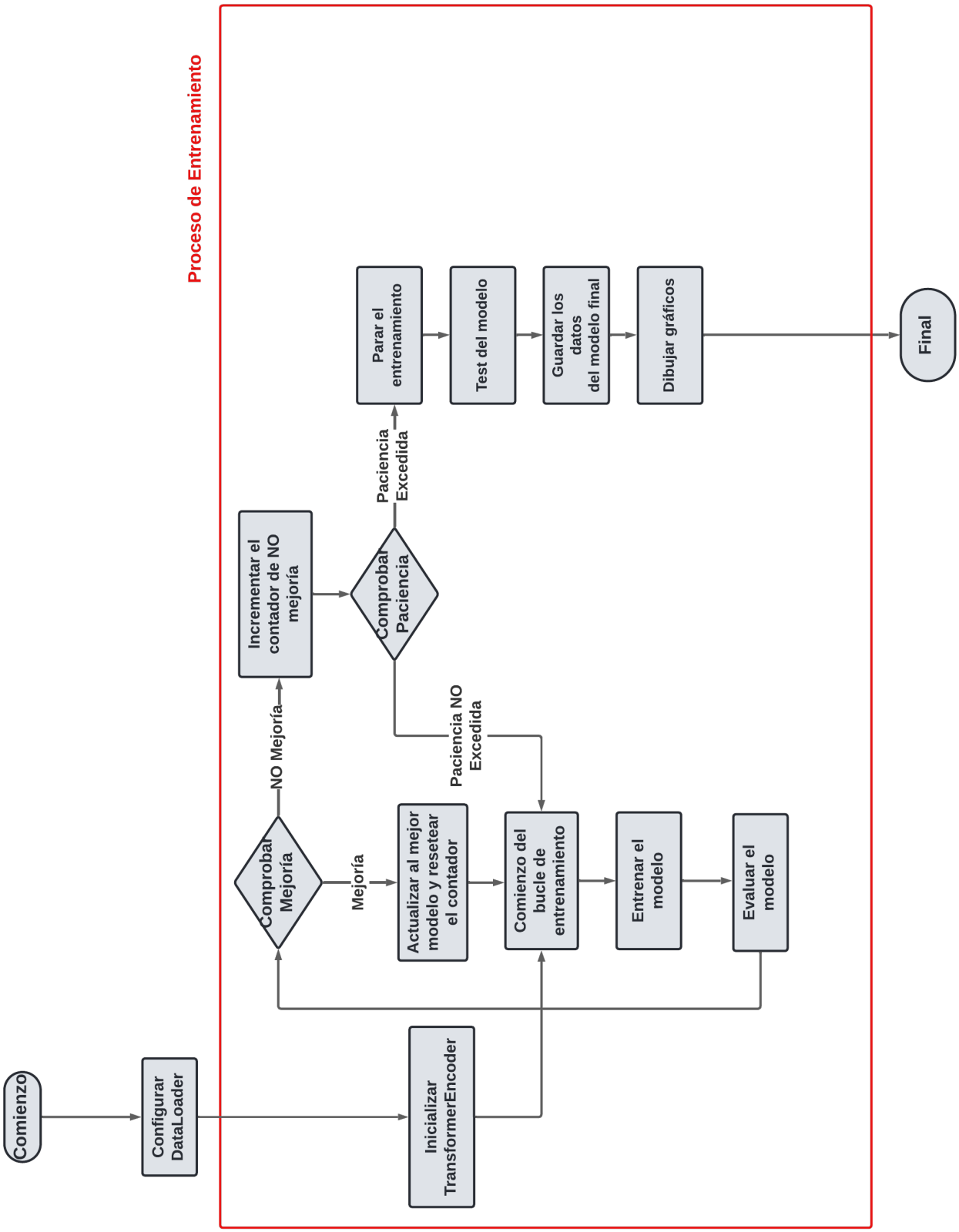


Figura 28: Esquema de instanciación del modelo, entrenamiento, etc

6. Pruebas y Resultados

En esta sección se explicarán los hiperparámetros seleccionados para los experimentos realizados junto con los resultados obtenidos, se presentará una comparación entre el mejor y peor resultado. Finalmente, se presentará un análisis de resultados comparando los resultados obtenidos en este proyecto junto con los presentados en el articulo científico "Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparision".

6.1. Experimentos realizados

En el presente apartado, se profundizará en la definición de los experimentos que se han realizado y sus resultados, junto con la explicación de la elección de los hiperparámetros que configuran la arquitectura y el entrenamiento del modelo encargado del reconocimiento de signos. La elección y ajuste de estos hiperparámetros son importantes para optimizar el rendimiento, minimizar el sobreajuste (el denominado *overfitting*) y garantizar una eficiencia adecuada. A continuación, se detallará la importancia y el impacto de algunos hiperparámetros que se utilizan en el entrenamiento, los cuales se han mantenido fijos desde el principio.

- **input_dim:** Este parámetro define la dimensionalidad de la entrada del modelo. En el caso del modelo utilizado para el proyecto (*TransformerEncoder*), corresponde al número de características o dimensiones de los datos de entrada. La elección de este hiperparámetro determina la capacidad del modelo para representar la complejidad de los datos de entrada. Un valor más alto permite al modelo capturar más características, pero también aumenta la *complejidad computacional*[22] y el riesgo de sobreajuste.
- **num_heads:** Este parámetro especifica el número de "cabezas" en el mecanismo de atención del *transformer*. Cada "cabeza" permite al modelo aprender diferentes características de los datos de entrada. Un valor elevado de cabezas permite al modelo aprender representaciones más ricas, pero como en anteriores hiperparámetros, esto aumenta la complejidad computacional.
- **dim_feedforward:** Este parámetro define la dimensionalidad de la capa totalmente conectada dentro de cada capa del *transformer*. La capa totalmente conectada permite al modelo aprender transformaciones no lineales de los datos de entrada. Un valor más alto para *dim_feedforward* permite al modelo aprender transformaciones más complejas, pero aumenta la complejidad computacional.
- **output_dim:** Este parámetro se identifica como el tamaño de la dimensión de salida del modelo que corresponde al número de clases en el problema de clasificación. La elección de este hiperparámetro se utiliza para asegurar que el modelo pueda clasificar correctamente los datos de entrada.

[22] **Complejidad computacional:** Cantidad de recursos computacionales necesarios para resolver un problema o ejecutar un algoritmo.

- **max_seq_len:** Este parámetro se identifica como el número máximo de *tokens*[23]en una secuencia de entrada. Es importante para definir la longitud de estas secuencias para el *transformer*, permitiendo al modelo manejar eficientemente secuencias de diferentes longitudes.

Además, cabe destacar que los hiperparámetros que se explicarán a continuación son aquellos que se han utilizado para ajustar y obtener los resultados que posteriormente se representarán.

- **hidden_dim:** Este parámetro establece la dimensión de los espacios latentes del modelo, que son las representaciones vectoriales internas utilizadas en el proceso de codificación y decodificación. *hidden_dim* no solo influye en la complejidad y capacidad de representación del modelo, sino que también determina la dimensionalidad de las salidas tanto de la capa de *embedding* como de las capas del *transformer*. Es decir, esta dimensión impacta en cómo se estructuran y dimensionan las representaciones vectoriales de las entradas y las transformaciones aplicadas a lo largo del modelo, lo que a su vez afecta la capacidad del modelo para capturar información y realizar tareas específicas. Un valor más alto para *hidden_dim* permite al modelo aprender representaciones más ricas y complejas, pero también aumenta la complejidad computacional y el riesgo de overfitting.
- **num_layers:** Este parámetro determina el número de bloques *transformer* (un bloque incluye varias capas). Cada bloque *transformer* adicional permite al modelo aprender representaciones más profundas de los datos, pero también aumenta la complejidad computacional y el riesgo de overfitting. Por lo tanto, la elección de *num_layers* es un equilibrio entre la capacidad del modelo para aprender representaciones complejas y la eficiencia computacional.
- **learning_rate:** Este parámetro es la tasa de aprendizaje utilizada por los optimizadores. Una tasa de aprendizaje más pequeña puede hacer que el entrenamiento sea más estable, pero también puede provocar que el entrenamiento sea más lento o, peor aún, que no converja. Además, la elección de la tasa de aprendizaje está relacionada con el tamaño del lote. Esta relación se debe a que un tamaño de lote más grande puede proporcionar una estimación más precisa del gradiente, permitiendo el uso de tasas de aprendizaje mayores, mientras que un tamaño de lote más pequeño puede requerir una tasa de aprendizaje menor para evitar saltos bruscos y asegurar una convergencia más suave.
- **batch_size:** Este parámetro determina el número de muestras que pasan por una red neuronal simultáneamente durante el entrenamiento y cuya pérdida conjunta se utiliza para la actualización de parámetros. Un valor alto requiere más memoria y puede afectar a la calidad de las actualizaciones de los parámetros. En particular, un valor muy grande puede llevar a actualizaciones de parámetros menos frecuentes pero más precisas, mientras que un valor muy pequeño puede hacer actualizaciones más frecuentes pero con mayor variabilidad. La elección del tamaño del lote está relacionada tanto con la capacidad de memoria disponible en la GPU (si las

[23]**Tokens:** Unidad básica de texto, como una palabra o un símbolo, que se utiliza como entrada para modelos de aprendizaje automático.

ejecuciones se realizan con una GPU) como con los hiperparámetros *num_layers* y *hidden_dim*, ya que estos también influyen en el consumo de memoria del modelo.

- **weight_decay:** Este parámetro se utiliza para aplicar la "regularización L2" en la función de pérdida del modelo. La "regularización L2" penaliza los pesos grandes al agregar un término a la función de pérdida que es proporcional al cuadrado de la magnitud de los pesos. Esto ayuda a prevenir el sobreajuste al forzar al modelo a mantener los parámetros en valores más bajos. Sin embargo, es importante elegir un valor adecuado para "weight_decay", ya que si es demasiado alto, existe el riesgo de que el modelo sufra de *underfitting* al no poder representar adecuadamente los datos.
- **transformer_dropout:** Tasa de *dropout* aplicada a las salidas de las capas del modelo. El *dropout* es una técnica de regularización que ayuda a prevenir el sobreajuste al "apagar" aleatoriamente algunas neuronas durante el entrenamiento. (En nuestros experimentos el valor del *dropout* será fijo).
- **num_epochs:** Número de veces que el algoritmo de aprendizaje recorre el conjunto completo de datos durante el entrenamiento. Cada pasada completa se conoce como una "época". Además, está relacionado con otros parámetros como el tamaño del lote, la tasa de aprendizaje y la dificultad de la tarea, entre otras.

  En este caso al utilizar la técnica *early stopping* no es necesario configurarlo, por lo que se representarán como el número de épocas completas antes de darse el *early stopping*.

	HIPERPARÁMETROS				
#Prueba	Hidden Dimension	Num Layers	Learning Rate	Batch Size	Weight Decay
Prueba 1	225	2	10^{-4}	100	10^{-5}
Prueba 2	225	2	10^{-4}	100	10^{-6}
Prueba 3	225	2	10^{-4}	100	10^{-7}
Prueba 4	225	2	$5 \cdot 10^{-5}$	100	10^{-7}
Prueba 5	225	2	$5 \cdot 10^{-5}$	100	10^{-6}
Prueba 7	225	4	$5 \cdot 10^{-5}$	100	10^{-5}
Prueba 8	225	4	$5 \cdot 10^{-5}$	100	10^{-6}
Prueba 9	225	4	$5 \cdot 10^{-5}$	100	10^{-7}
Prueba 10	225	4	10^{-4}	100	10^{-7}
Prueba 11	225	4	10^{-4}	100	10^{-6}
Prueba 12	225	4	10^{-4}	100	10^{-5}
Prueba 13	112	4	10^{-4}	100	10^{-5}
Prueba 14	112	4	10^{-4}	100	10^{-6}
Prueba 15	112	4	10^{-4}	100	10^{-7}
Prueba 16	112	4	$5 \cdot 10^{-5}$	100	10^{-7}
Prueba 17	112	4	$5 \cdot 10^{-5}$	100	10^{-6}
Prueba 18	112	4	$5 \cdot 10^{-5}$	100	10^{-5}
Prueba 19	112	2	$5 \cdot 10^{-5}$	100	10^{-5}
Prueba 20	112	2	$5 \cdot 10^{-5}$	100	10^{-6}
Prueba 21	112	2	$5 \cdot 10^{-5}$	100	10^{-7}
Prueba 22	112	2	10^{-4}	100	10^{-7}
Prueba 23	112	2	10^{-4}	100	10^{-6}
Prueba 24	112	2	10^{-4}	100	10^{-5}

Tabla 13: Hiperparámetros y sus valores seleccionados. El color amarillo indica indica una modificación respecto al experimento anterior (Siendo la prueba 1 la base, sin amarillo)

	Resultados					
	Entrenamiento	Evaluación		Test		Épocas
#PRUEBA	ACC	ACC (Top - 1)	ACC (Top - 5)	ACC (Top - 1)	ACC (Top - 5)	
Prueba 1	91,20	15,68	43,31	16,57	42,40	181
Prueba 2	90,70	17,80	44,93	16,50	44,60	169
Prueba 3	91,11	16,28	42,67	16,20	42,26	187
Prueba 4	90,65	16,51	44,83	17,01	42,41	323
Prueba 5	91,61	19,97	44,28	17,08	42,12	328
Prueba 6	90,93	15,26	43,40	16,42	40,58	323
Prueba 7	90,65	15,04	40,45	13,43	37,59	212
Prueba 8	91,14	12,04	36,12	11,97	34,45	232
Prueba 9	91,38	14,44	40,91	14,53	39,42	259
Prueba 10	90,81	13,70	38,15	13,94	38,61	132
Prueba 11	90,87	14,02	38,42	12,92	35,91	157
Prueba 12	91,08	13,67	39,62	14,89	38,61	135
Prueba 13	89,97	13,01	38,51	13,58	35,84	294
Prueba 14	88,64	13,98	38,38	14,53	36,86	287
Prueba 15	89,44	13,91	39,07	14,38	40,15	289
Prueba 16	90,00	13,84	38,47	13,28	36,13	534
Prueba 17	88,54	13,70	39,99	14,09	38,10	473
Prueba 18	88,27	13,79	38,84	13,14	37,74	523
Prueba 19	0,14	0,18	0,55	0,15	0,51	22
Prueba 20	67,76	14,53	41,24	16,20	40,29	443
Prueba 21	0,11	0,05	0,46	0,15	0,44	24
Prueba 22	89,37	16,97	46,70	18,25	44,23	389
Prueba 23	87,83	16,65	45,76	18,91	44,45	348
Prueba 24	89,25	17,11	46,08	17,01	43,87	378

Tabla 14: Resultados finales (verde: los mejores resultados teniendo en cuenta las métricas ACC top-1 y top-5 del test, rojo: el peor resultado teniendo en cuenta que su valor es muy bajo en la métrica ACC en el entrenamiento, evaluación y test). El número de épocas se refiere a la cantidad de ciclos de entrenamiento completados hasta que el modelo deja de mejorar

Los resultados presentados en la Tabla 14 muestran el desempeño del modelo de reconocimiento de lengua de signos en varios experimentos realizados. Estos experimentos se llevaron a cabo para evaluar la capacidad del modelo.

En cuanto a métricas de evaluación, se observan valores de precisión (ACC) para el conjunto de entrenamiento, evaluación y test en cada una de las pruebas realizadas. La precisión en el conjunto de entrenamiento indica la capacidad del modelo para ajustarse a los datos utilizados durante el entrenamiento, lo cual se espera que sea alta, ya que el modelo se ha entrenado específicamente con estos datos. Por otro lado, la precisión en los conjuntos de evaluación y test proporciona información sobre la capacidad de generalización del modelo a datos no vistos previamente. Aunque es importante tener en cuenta, que en la evaluación se ha usado levemente el conjunto de datos para seleccionar el modelo con mejores resultados. Esto puede afectar la verdadera capacidad de generalización del modelo.

Además, se calculan métricas más específicas como la precisión top-1 (ACC TOP-1) y la precisión top-5 (ACC (TOP-5)). La precisión top-1 se refiere a la proporción de predicciones correctas en la clase más probable, mientras que la precisión top-5 evalúa la proporción de veces que la clase correcta se encuentra entre las cinco clases más probables predichas por el modelo.

En los resultados se puede observar la variabilidad en el rendimiento del modelo en los diferentes experimentos. Si se tienen en cuenta los resultados resaltados, en el resultado resaltado en color rojo (prueba 21) se puede ver como los valores son ínfimos teniendo en cuenta que los valores que están representados en la tabla son sobre cien. Esto se debe a la elección de los valores de los hiperparámetros utilizados (Tabla 13), lo que sugiere la necesidad de ajustarlos con otras combinaciones diferentes para mejorar el rendimiento del modelo.

En cuanto a los resultados resaltados en color verde (prueba 22 y prueba 23) se puede ver como en comparación a los demás en las columnas de ACC de evaluación y ACC de test son las que mayor porcentaje de precisión han tenido, siendo en el caso del *test* casi un 20 % en la columna top-1 en ambos experimentos.

A continuación, se puede ver una representación gráfica de la precisión de la prueba 21 (Figura 29 Izquierda) y de la prueba 22 (Figura 29 Derecha).

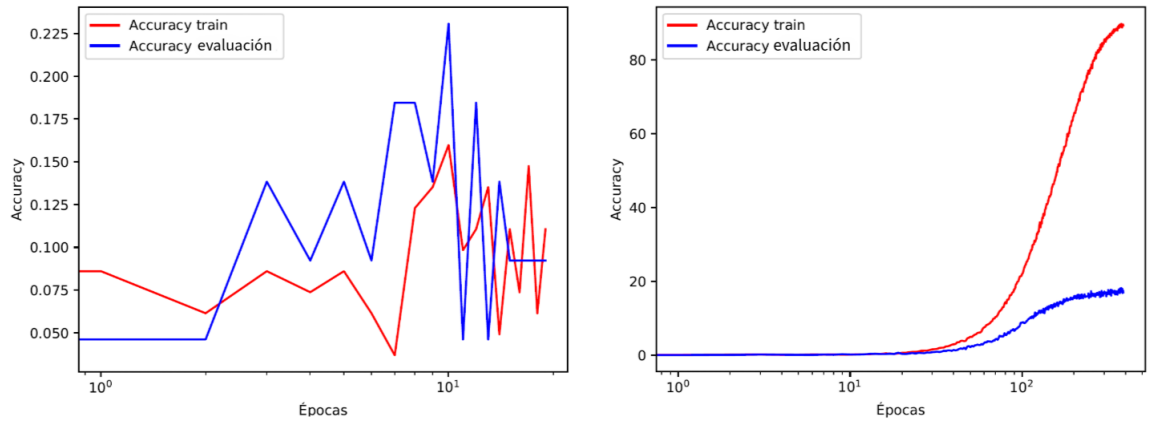


Figura 29: Gráficos de los resultados de las prueba 21 (izquierda) y 22 (derecha)

Por un lado, se puede ver como la prueba con peores resultados (Prueba 21) la precisión se muestra inestable, además de situarse en valores muy pequeños (entre 0.050 y 0.225 sobre cien en el conjunto de evaluación y para el conjunto de entrenamiento entre 0.050 y 0.160 aproximadamente), esto es un claro indicio de que la combinación de hiperparámetros seleccionada no es la adecuada.

Por otro lado, para una de las pruebas con los mejores resultados, como por ejemplo la prueba 22, se puede observar cómo a lo largo de las épocas la precisión va en aumento, lo cual es un buen indicador de que el modelo está aprendiendo. Sin embargo, es importante destacar que durante las primeras épocas la función permanece completamente recta, lo que indica que el modelo no está aprendiendo en ese período inicial.

6.2. Análisis de resultados

En este apartado, se evalúa el modelo de reconocimiento de lengua de signos mediante la comparación de resultados y métodos presentados en el documento "Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparision" de la WACV 2020 (Winter Conference on Applications of Computer Vision) [28]. Este estudio destaca por su presentación del conjunto de datos WASL2000, siendo este el mismo conjunto de datos utilizado para el desarrollo del proyecto presentado en esta memoria.

En el documento mencionado, se realizan experimentos utilizando varios modelos de reconocimiento de lengua de signos. Entre estos modelos se encuentran:

- **VGG-GRU:** Este modelo combina la arquitectura de red neuronal convolucional VGG (Visual Geometry Group) con una unidad de red neuronal recurrente (GRU)[24] para el reconocimiento de lengua de signos. La VGG se encarga de extraer características visuales de los fotogramas del vídeo, mientras que la GRU modela la secuencia temporal del gestos para realizar la clasificación a nivel de glosas.
- **Pose-GRU:** En este modelo, se utiliza la información de la pose corporal o gestual de las personas en los fotogramas del vídeo junto con una unidad de red neuronal recurrente para el reconocimiento. La información de la pose se combina con la información visual de una red neuronal convolucional para la clasificación.
- **Pose-TGCN (Temporal Graph Convolutional Network):** Este modelo utiliza la información de pose junto con una red neuronal convolucional TGCN para el reconocimiento. La TGCN modela las relaciones temporales entre los gestos para realizar la clasificación a nivel de glosas.
- **I3D (Inflated 3D ConvNet):** El modelo I3D procesa directamente los vídeos en 3D utilizando una arquitectura de red neuronal convolucional. La red I3D extrae características espaciales y temporales de los vídeos para realizar la clasificación.

En la tabla 15, se pueden observar los resultados de los modelos comentados frente al conjunto de datos WSASL2000.

WLASL2000		
MÉTODO	ACC TOP-1	ACC TOP-5
Pose-GRU	22.54	49.81
Pose-TGCN	23.65	51.75
VGG-GRU	8.44	23.58
I3D	32.48	57.31
Prueba 22	18.25	44.23
Prueba 23	18.91	44.45

Tabla 15: ACC top-1 y ACC top-5 para los diferentes modelos frente al conjunto de datos WSASL2000

[24]**Red neuronal recurrente (GRU):** Tipo de arquitectura de red neuronal recurrente (RNN) que se utiliza en el campo del aprendizaje profundo para modelar secuencias de datos temporales, como texto, audio o series temporales

Teniendo en cuenta los resultados de la Tabla 15 si lo comparamos con los mejores resultados obtenidos en el desarrollo realizado en este TFG recogidos en color verde en la Tabla 14 (Prueba 22 y Prueba 23) se puede observar como los resultados obtenidos en este TFG no están muy lejos de los resultados recogidos en el documento de la WACV 2020 (Pose-GRU, Pose-TGCN, VGG-GRU, i3D).

Esto surgiere que el modelo desarrollado en este proyecto ha logrado resultados competitivos en términos de precisión en la clasificación de la lengua de signos. Por lo tanto, esto indica la efectividad y relevancia del modelo propuesto en este proyecto.

7. Conclusiones

7.1. Conclusiones del TFG

En este Trabajo de Fin de Grado se ha explorado el desarrollo de un sistema de reconocimiento de la lengua de signos americana a partir de vídeos, utilizando técnicas de aprendizaje profundo, específicamente la arquitectura *transformer*. A lo largo del proyecto, se han abordado diferentes etapas, desde la investigación inicial hasta la evaluación del modelo.

7.1.1. Eficacia del modelo en el reconocimiento de lengua de signos

Los resultados obtenidos en las pruebas realizadas con el modelo propuesto muestran un rendimiento prometedor en la tarea del reconocimiento de signos a nivel de glosas individuales. Las métricas de evaluación, precisión (ACC) y la precisión top-5 (ACC Top-5), indican que el modelo es capaz de clasificar correctamente los signos a partir de las secuencias de *key points* extraídas de los vídeos. Si bien los resultados no alcanzan las cotas de precisión de modelos más avanzados, se observa una capacidad notable para generalizar a datos no vistos previamente (Tabla 16).

WLASL2000		
MÉTODO	ACC TOP-1	ACC TOP-5
Pose-GRU	22.54	49.81
Pose-TGCN	23.65	51.75
VGG-GRU	8.44	23.58
I3D	32.48	57.31
Modelo del TFG	**18.91**	**44.45**

Tabla 16: Comparación de resultados ACC top-1 y ACC top-5 entre los métodos más avanzados y el desarrollado en el TFG

Es importante destacar que los resultados obtenidos se centran en la clasificación de glosas individuales. Esto significa que el modelo es capaz de identificar correctamente un signo aislado dentro de un vídeo, pero no se ha evaluado su desempeño en la clasificación de secuencias de glosas.

7.1.2. Representación espacio-temporal de los signos

La representación de los datos de entrada ha sido un factor determinante en el desempeño del modelo. La elección de *key points* como base para representar la información espacio-temporal de los signos ha permitido capturar las características esenciales de los signos y su evolución temporal. El preprocesamiento realizado, incluyendo la normalización de la postura y la asignación temporal, han permitido una buena representación de los datos, mejorando la capacidad del modelo para aprender patrones relevantes y generalizar a nuevos ejemplos.

Sin embargo, la utilización de *key points* como única fuente de información presenta ciertas limitaciones. La precisión del modelo depende en gran medida de la capacidad del algoritmo de detección de puntos clave para capturar con precisión la posición y el

movimiento de las manos y el cuerpo. En situaciones donde la calidad de los *key points* se ve afectada por sus factores como la oclusión, la iluminación o la baja resolución del vídeo, el rendimiento del modelo puede verse comprometido.

7.1.3. Aprendizaje profundo como herramienta para la inclusión

Este proyecto no solo ha explorado las posibilidades del aprendizaje profundo en el reconocimiento de lengua de signos, sino que también pone en manifiesto el potencial de esta tecnología para la creación de herramientas inclusivas que puedan mejorar la comunicación y accesibilidad para la comunidad con discapacidad auditiva.

El desarrollo de sistemas de reconocimiento de lengua de signos precisos y robustos puede tener un impacto significativo en la vida de las personas con discapacidad auditiva, facilitando su acceso a la participación plena en la sociedad. La capacidad de traducir automáticamente la lengua de signos a lengua hablada podría romper barreras de comunicación y promover la inclusión social.

7.1.4. Limitaciones y desafíos del proyecto

El desarrollo de este proyecto se ha visto afectado por algunas limitaciones, siendo algunas ya previstas en la gestión de riesgos.

En particular, se ha producido el riesgo de exceder el tiempo estimado para las diferentes tareas (Tabla 2 de la Sección 4.2.1), especialmente en la etapa del desarrollo del modelo, entrenamiento y evaluación. En estas etapas se han encontrado diversos problemas a la hora de conseguir que el modelo comenzase a aprender de manera correcta, ya que el ajuste de algún método en el aprendizaje junto con un primer algoritmo de obtención de *key points* erróneo atrasó por completo el desarrollo al inicio.

Además, en la etapa de entrenamiento se han dado situaciones en la que la cola de ejecución del servidor en la que se realizaba el entrenamiento se encontraba completa y eso provocaba largas esperas para poder probar diferentes combinaciones de hiperparámetros o cambios en el código. A esto hay que añadirle la necesidad de compaginar el desarrollo del proyecto con un trabajo de jornada completa, lo cual ha supuesto un reto adicional en el desarrollo del proyecto.

Sin embargo, a pesar de estas limitaciones, el proyecto ha logrado alcanzar sus objetivos principales dejando una base a futuros avances.

7.2. Reflexión personal

En lo referente a la reflexión personal, al inicio, cuando trataba de encontrar el tema de mi trabajo de fin de grado, mi primera idea fue relacionarla con la gestión de las finanzas y la economía personal, ya que en aquel momento me encontraba realizando las prácticas de grado para una empresa proveedora de software para el banco BBVA. Sin embargo, finalmente la idea tomó un rumbo distinto.

Después de reunirme con mi director de TFG, Adrián, su orientación y la propuesta del tema que finalmente seleccionaría resultaron ser cruciales por dos razones. En primer lugar, representaba un desafío personal aprender y desarrollar con tecnologías que apenas conocía y que hasta entonces no había utilizado, como el aprendizaje profundo.

El segundo motivo que me hizo decidirme por esta propuesta está ligado a la problemática tratada en el proyecto, el desarrollo de una pequeña herramienta que sería el granito de arena hacia una aplicación capaz de mejorar la comunicación entre usuarios que usan la lengua de signos y los que no. Además, este problema me ha tocado vivirlo en primera persona. Durante el curso trabajaba de cara al público en una tienda de zapatillas y han sido pocas las situaciones en las que me habré visto envuelto en la incapacidad de poder comunicarme con un cliente por no saber lengua de signos, pero las suficientes para haber sentido esa frustración de no poder entender completamente a otra persona y no poder ayudarla con sus dudas o simplemente atenderla de manera óptima.

Además en ese momento se echaba en falta alguna herramienta igual a las que disponemos para cuando no conocemos otros idiomas como pueden ser los traductores en línea tales como Google traductor, lo cual me habría facilitado la comunicación con clientes que utilizaban la lengua de signos y mejorando así mi capacidad de atención al cliente.

Habiendo aceptado la propuesta de TFG, adentrarme en el mundo del aprendizaje profundo no ha sido fácil para mí. Esto se debe a las dificultades que he tenido en diversos momentos para entender ciertos conceptos completamente nuevos para mí los cuales me hacían acarrear errores en el desarrollo. Es por ello que agradezco la tremenda paciencia de los directores de este proyecto para ayudarme y la atención que me han ofrecido para solventar los errores más críticos que surgieron durante el desarrollo de la herramienta.

Mirando hacia atrás, también estoy agradecido por haber tomado este camino. Ha ampliado mis conocimientos al utilizar nuevas tecnologías para mí, y además me ha ayudado a desarrollar una mejor planificación en mi día día ya que he tenido que compaginar el desarrollo de este trabajo junto con mi trabajo a tiempo completo.

7.3. Trabajo a futuro

En cuanto al trabajo a futuro, hay varios puntos que podrían continuarse desde donde termina este trabajo:

- **Realizar pruebas con diferentes conjuntos de datos:** En este trabajo se han ensayado con vídeos de lengua de signos americana extraídos del repositorio WLASL. Sin embargo, sería interesante probar con otros conjuntos de vídeos de lengua de signos americana o de otras lenguas para ver como la herramienta lo afrontaría. Unos cuantos ejemplos pueden ser los enumerados en la figura 30.

id	Name	Country	Classes	Subjects	Samples	Data	Language level	Type	Annotations	Availability
1	DGS Kinect 40	Germany	40	15	3000		Word	Videos, multiple angles		Contact Author
2	RWTH-PHOENIX-Weather	Germany	1200	9	45760	53gb	Sentence	Videos	Face, hand, end/start (unfinished)	**Publicly Available**
3	SIGNUM	Germany	450	25	33210	920gb	Sentence	Videos		1TB **contact author**
4	GSL 20	Greek	20	6	~840		Word			Contact Author
5	Boston ASL LVD	USA	3300+	6	9800		Word	Videos, multiple angles	hand,end/start	**Publicly Available**
6	PSL Kinect 30	Poland	30	1	30×10=300	~1.2gb	Word	Videos, depth from Kinect camera		Publicly Available

Figura 30: Varios ejemplos de diferentes conjuntos de datos de lengua de signos
Fuente: `http://facundoq.github.io/guides/sign_language_datasets/slr`

- **Realizar ensayos cambiando las características de entrada:** En este trabajo se han utilizado como entrada características basadas en *key points*, por lo que una buena practica sería probar otras características extraídas de modelos como CNNs o Vision Transformers (ViT) para evaluar el rendimiento frente a los resultados obtenidos en este proyecto.
- **Exploración de modelado con secuencia de grafos en tres dimensiones:** Utilizar una secuencia de grafos en lugar de una secuencia de vectores para modelar la entrada. Haría falta trabajar con *3D Graph neural networks* para procesar el "grafo 3D". Estas redes neuronales están diseñadas para trabajar con datos de grafos en tres dimensiones, permitiendo la captura de relaciones espaciales y temporales más complejas en los datos de entrada.
- **Continuous Sign Language Recognition (CSLR):** Es necesario incorporar técnicas de segmentación de vídeo para identificar regiones candidatas a ser signos válidos. Además, se requeriría establecer la confianza de cada región para determinar si efectivamente representa un signo y, además, filtrar en tiempo real cuáles son las propuestas más aceptables, evitando la duplicación de signos. Por ejemplo, podríamos tener dos regiones que representan el mismo signo, pero una de ellas captura mejor el inicio y el final del signo que la otra.

Referencias

[1] Grupo Atico 34. Overfitting. qué es, causas, consecuencias y cómo solucionarlo. https://protecciondatos-lopd.com/empresas/overfitting/#:~:text=Este%20fenÃşmeno%20se%20llama%20overfitting,en%20nuestro%20conjunto%20de%20datos.

[2] Amazon.es. Lenovo ideapad 3 gen 6. https://www.amazon.es/Lenovo-IdeaPad-Gen-Ordenador-i5-1155G7/dp/B0BQRK5T8H/ref=sr_1_5?adgrpid=68090007775&hvadid=338576367275&hvdev=c&hvlocphy=1005509&hvnetw=g&hvqmt=e&hvrand=3998976824586750259&hvtargid=kwd-340730546081&hydadcr=3221_1809154&keywords=lenovo+ideapad+3&qid=1695987354&sr=8-5&ufe=app_do%3Aamzn1.fos.5e544547-1f8e-4072-8c08-ed563e39fc7d.

[3] Jose M . Ambrocio. El lÉxico de la comunidad lgtb. https://www.academia.edu/37761461/EL_L%C3%89XICO_DE_LA_COMUNIDAD_LGTB.

[4] app.diagrams.net. Draw.io. https://app.diagrams.net/.

[5] blogolengua.com. Nombre propios en la lengua de signos. http://www.blogolengua.com/2010/12/nombres-lengua-signos-sordos.html.

[6] Diane Brentari. Sign languages. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjpoLLj1d6BAxW0U6QEHaJ3CBQQFnoECAcQAQ&url=https%3A%2F%2Fedisciplinas.usp.br%2Fpluginfile.php%2F4415432%2Fmod_folder%2Fcontent%2F0%2FCambridge%2520Language%2520Surveys%2FBrentari%25202010.%2520Sign%2520Languages.pdf%3Fforcedownload%3D1&usg=AOvVaw2l8DiQrQinC3mttiJoK4pS&opi=89978449.

[7] colab.google. Google colab. https://colab.google/.

[8] Universidad de Stanford. Convolutional neural networks for visual recognition. https://www.youtube.com/watch?v=vT1JzLTH4G4&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv.

[9] Ministerio de trabajo y economía social. Resolución de 13 de julio de 2023, de la dirección general de trabajo, por la que se registra y publica el xviii convenio colectivo estatal de empresas de consultoría, tecnologías de la información y estudios de mercado y de la opinión pública. *Boletín Ofical del Estado*, 2023.

[10] AD Dongare, RR Kharde, Amit D Kachare, et al. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, 2(1):189–194, 2012.

[11] DotCSV. Lista de vídeos sobre redes neuronales. https://www.youtube.com/watch?v=MRIv2IwFTPg&list=PL-Ogd76BhmcB9OjPucsnc2-piEE96jJDQ.

[12] drive.google.com. Google drive. drive.google.com/.

[13] Issam El Naqa and Martin J Murphy. *What is machine learning?* Springer, 2015.

[14] Europapress. Desarrollan un guante que traduce el lenguaje de signos al habla en tiempo real. https://www.europapress.es/ciencia/laboratorio/noticia-desarrollan-guante-traduce-lenguaje-signos-habla-tiempo-real-20200629174742.html.

[15] EuropaPress. Más de 70 millones de personas en el mundo son sordas y utilizan 300 lenguas de signos diferentes. https://www.europapress.es/epsocial/igualdad/noticia-mas-70-millones-personas-mundo-son-sordas-utilizan-300-lenguas-signos-diferentes-20190923121719.html.

[16] ganttproject.biz. Gantt project. https://www.ganttproject.biz/.

[17] Github.com. Github. https://github.com/.

[18] María Ángeles Rodríguez González and M ª Ángeles. *Lenguaje de signos*. Confederación Nacional de Sordos de España, 1992.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[20] Google. Mediapipe. `https://developers.google.com/mediapipe/solutions/vision/holistic_landmarker`.

[21] HITZ. Servidor upv. `https://www.hitz.eus/`.

[22] A Huerta, E Ibáñez, R San-Segundo, F Fernández, R Barra, and LF D'Haro. Primera experiencia de traducción automática de voz en lengua de signos. 2015.

[23] Indeed. Sueldos por año de machine learning engineer/a en españa. `https://es.indeed.com/cmp/Quental/salaries/Machine-learning-engineer-a`.

[24] JetBrains.com. Pycharm. `https://www.jetbrains.com/es-es/pycharm/`.

[25] julen Pastor Rodríguez. Machine learning y redes neuronales artificiales (rna). `https://microbacterium.es/machine-learning-y-redes-neuronales-artificiales-rna`.

[26] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 1(8), 2017.

[27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[28] DONGXU LI, Cristian Rodriguez, Xin Yu, and HONGDONG LI. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[29] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021.

[30] Lorenzo López Salcedo et al. Alfabeto dactilológico. 2001.

[31] Damián Jorge Matich. Redes neuronales: Conceptos básicos y aplicaciones. `https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf`.

[32] Microsoft. Visual studio code. `https://code.visualstudio.com/`.

[33] Naila Murray and Florent Perronnin. Generalized max pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2473–2480, 2014.

[34] Na8. Historia de las redes neuronales desde 1950 a 2018.

[35] Adrián Núñez-Marcos, Olatz Perez-de Viñaspre, and Gorka Labaka. A survey on sign language machine translation. *Expert Systems with Applications*, page 118993, 2022.

[36] OpenAI. Chatgpt. `https://chat.openai.com`.

[37] Eleni Orfanidou, Robert Adam, Gary Morgan, and James M McQueen. Recognition of signed and spoken language: Different sensory inputs, the same segmentation procedure. *Journal of Memory and Language*, 62(3):272–283, 2010.

[38] Isabel de los Reyes Rodríguez Ortiz. *Comunicar a través del silencio: las posibilidades de la lengua de signos española*, volume 5. Universidad de Sevilla, 2005.

[39] Overleaf.com. Overleaf. `https://www.overleaf.com/`.

[40] Salim Oyinlola. Descenso de gradiente: ejemplo de algoritmo de aprendizaje automático. `https://www.freecodecamp.org/espanol/news/descenso-de-gradiente-ejemplo-de-algoritmo-de-aprendizaje-automaticod/`.

[41] JAHAZIEL PONCE. Conoce quÉ son las funciones de activaciÓn y cÓmo puedes crear tu funciÓn de activaciÓn usando python, r y tensorflow. https://jahazielponce.com/funciones-de-activacion-y-como-puedes-crear-la-tuya-usando-python-r-y-tensorflow/.

[42] pytorch.org. Deep learning with pytorch: A 60 minute blitz. https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html.

[43] pytorch.org. Pytorch. https://pytorch.org/.

[44] Vicente Rodriguez. Dropout y batch normalization. https://vincentblog.xyz/posts/dropout-y-batch-normalization.

[45] Alberto Rubiales. Redes neuronales: Propagación hacia adelante y propagación hacia atrás. https://rubialesalberto.medium.com/redes-neuronales-propagaciÃşn-hacia-adelante-y-propagaciÃşn-hacia-atrÃąs-4745c0fb6286.

[46] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[47] Dr. Lluís Solí. Filtros de convolución. http://www.dimages.es/Tutorial%20A.I/enhancement/filtros.htm.

[48] Stokoe. *the visual communication systems of the American deaf.* 1960.

[49] Federico Tavella, Viktor Schlegel, Marta Romeo, Aphrodite Galata, and Angelo Cangelosi. Wlasl-lex: a dataset for recognising phonological properties in american sign language. *arXiv preprint arXiv:2203.06096*, 2022.

[50] Andrew W. Trask. Grokking deep learning. https://www.perlego.com/es/book/2682860/grokking-deep-learning-pdf.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[52] WFD. 70 million deaf people.

[53] WLASL. github repository. https://github.com/dxli94/WLASL.

[54] Dianne y Stephen Parkhurst. La variación en las lenguas de signos: un estudio de causas y una metodología analítica. https://cnlse.es/es/recursos/biblioteca/la-variaci%25C3%25B3n-en-las-lenguas-de-signos-un-estudio-de-causas-y-una-metodolog%25C3%25ADa.

[55] Xue Ying. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, volume 1168, page 022022. IOP Publishing, 2019.

A. Anexo I: Tareas de los paquetes de trabajo

A.1. Tareas de Planificación

1.1 Seguimiento del proyecto	
Paquete de Trabajo:	1. Planificación del proyecto.
Periodo estimado:	2 meses y 18 días (78 días - 25 horas)
Descripción:	Realizar seguimiento del proyecto y de las tareas mediante reuniones con los tutores.
Precedentes:	Ninguno.
Salidas:	Seguimiento del desarrollo realizado.
Recursos necesarios:	Ninguno.

Tabla 17: Descripción de la tarea 1.1

1.2 Definición de objetivos	
Paquete de Trabajo:	1. Planificación del proyecto.
Periodo estimado:	1 día (2 horas)
Descripción:	Describir los objetivos a completar durante el proceso.
Precedentes:	Ninguno.
Salidas:	Objetivos fijados
Recursos necesarios:	Ninguno.

Tabla 18: Descripción de la tarea 1.2

1.3 Definición de tareas	
Paquete de Trabajo:	1. Planificación del proyecto.
Periodo estimado:	3 día (3 horas)
Descripción:	Describir las tareas a realizar en el proyecto.
Precedentes:	Ninguno.
Salidas:	Tareas establecidas.
Recursos necesarios:	Ninguno.

Tabla 19: Descripción de la tarea 1.3

1.4 Planificación (EDT y Gantt)	
Paquete de Trabajo:	1. Planificación del proyecto.
Periodo estimado:	2 día (2 horas)
Descripción:	Realizar los diagramas que mostraran la planificación del proyecto (Diagrama EDT y Diagrama Gantt).
Precedentes:	Tarea 1.3
Salidas:	Tareas descritas y visualizadas.
Recursos necesarios:	Ninguno.

Tabla 20: Descripción de la tarea 1.4

1.5 Gestión de riesgos	
Paquete de Trabajo:	1. Planificación del proyecto.
Periodo estimado:	1 día (1 hora)
Descripción:	Determinar los riesgos del proyectos y realizar una evaluación con la finalidad de evitarlos.
Precedentes:	Ninguno.
Salidas:	Riesgos determinadsos y gestionados.
Recursos necesarios:	Ninguno.

Tabla 21: Descripción de la tarea 1.5

1.6 Evaluación económica	
Paquete de Trabajo:	1. Planificación del proyecto.
Periodo estimado:	4 días (3 horas)
Descripción:	Analizar y calcular los costes del proyecto.
Precedentes:	Tarea 1.3
Salidas:	Evaluación económica completada.
Recursos necesarios:	Ninguno.

Tabla 22: Descripción de la tarea 1.6

A.2. Tareas de Formación

2.1 Comprensión y análisis de la lengua de signos	
Paquete de Trabajo:	2. Formación
Periodo estimado:	5 días (10 horas)
Descripción:	Obtener conocimientos básicos sobre la lengua de signos y su trasfondo.
Precedentes:	Ninguno.
Salidas:	Conocimientos adquiridos sobre como funciona la lengua de signos.
Recursos necesarios:	Internet.

Tabla 23: Descripción de la tarea 2.1

2.2 Formación sobre aprendizaje Profundo	
Paquete de Trabajo:	2. Formación
Periodo estimado:	10 días (12 horas)
Descripción:	Obtener conocimientos básicos sobre Aprendizaje Profundo
Precedentes:	Ninguno.
Salidas:	Conocimientos adquiridos de aprendizaje profundo.
Recursos necesarios:	Libro Grooking, tutoriales online, blogs.

Tabla 24: Descripción de la tarea 2.2

2.3 Aprendizaje sobre el uso del software y librerías	
Paquete de Trabajo:	2. Formación
Periodo estimado:	20 días (15 horas)
Descripción:	Obtener conocimientos sobre el uso del software y librerías como pyTorch, Numpy, etc.
Precedentes:	Ninguno.
Salidas:	Conocimientos acerca del uso de las diferentes herramientas.
Recursos necesarios:	Libro Grooking, tutoriales online, blogs.

Tabla 25: Descripción de la tarea 2.3

A.3. Tareas de Exploración y Análisis del Estado del Arte

3.1 Revisar el estado del arte de las arquitecturas	
Paquete de Trabajo:	3. Exploración y Análisis del estado del arte
Periodo estimado:	3 días (6 horas)
Descripción:	Estudiar el estado del arte de las redes neuronales, etc.
Precedentes:	Ninguno.
Salidas:	Estado del arte de las arquitecturas.
Recursos necesarios:	Internet.

Tabla 26: Descripción de la tarea 3.1

3.2 Tareas relacionadas	
Paquete de Trabajo:	3. Exploración y Análisis del estado del arte
Periodo estimado:	4 días (9 horas)
Descripción:	Estudiar el uso actual de la clasificación de imágenes
Precedentes:	Ninguno.
Salidas:	Estudio de tareas relacionadas como clasificación de imágenes.
Recursos necesarios:	Internet.

Tabla 27: Descripción de la tarea 3.2

3.3 Estado del arte del reconocimiento automático de signos	
Paquete de Trabajo:	3. Exploración y Análisis del estado del arte
Periodo estimado:	4 días (8 horas)
Descripción:	Estudiar el estado del arte del procesamiento de la lengua de signos hoy en día.
Precedentes:	Ninguno.
Salidas:	Estado del arte del procesamiento de lenguaje natural.
Recursos necesarios:	Internet.

Tabla 28: Descripción de la tarea 3.3

A.4. Tareas de Desarrollo del proyecto

4.1 Preparación del conjunto de datos	
Paquete de Trabajo:	4. Desarrollo del proyecto
Periodo estimado:	1 día (4 horas)
Descripción:	Adecuar el conjunto de datos que se utilizará para el desarrollo, para agilizar y acomodar su utilización.
Precedentes:	Ninguno.
Salidas:	Dataset adecuado.
Recursos necesarios:	Internet, Github.

Tabla 29: Descripción de la tarea 4.1

4.2 Implementación del modelo	
Paquete de Trabajo:	4. Desarrollo del proyecto
Periodo estimado:	10 días (30 horas)
Descripción:	Implementar el modelo.
Precedentes:	Paquete de trabajo 4.1.
Salidas:	Modelo implementado.
Recursos necesarios:	Internet, Github, Pycharm, PyTorch, Librerías.

Tabla 30: Descripción de la tarea 4.2

4.3 Entrenamiento del modelo	
Paquete de Trabajo:	4. Desarrollo del proyecto
Periodo estimado:	14 días (40 horas)
Descripción:	Adecuar el conjunto de datos que se utilizará para el desarrollo, para agilizar y acomodar su utilización.
Precedentes:	Paquete de trabajo 4.1 y 4.2.
Salidas:	Modelo entrenado.
Recursos necesarios:	Pycharm, librerías, Github.

Tabla 31: Descripción de la tarea 4.3

A.5. Tareas de Pruebas y resultados

5.1 Evaluación	
Paquete de Trabajo:	5. Pruebas y Resultados
Periodo estimado:	2 días (20 horas)
Descripción:	Evaluar los resultados del entrenamiento realizado.
Precedentes:	Paquetes de trabajo 4.1, 4.2, 4.3
Salidas:	Evaluación realizada
Recursos necesarios:	Conjunto de datos de pruebas, internet, Github, Pycharm.

Tabla 32: Descripción de la tarea 5.1

5.2 Comparación del resultado con trabajo relacionado	
Paquete de Trabajo:	5. Pruebas y Resultados
Periodo estimado:	3 días (6 horas)
Descripción:	Realizar una comparación con trabajos similares realizados.
Precedentes:	Paquete de trabajo 5.1
Salidas:	Conclusiones sacadas de la comparación.
Recursos necesarios:	Internet.

Tabla 33: Descripción de la tarea 5.2

A.6. Tareas de Memoria

6.1 Redacción de la Memoria	
Paquete de Trabajo:	6. Memoria
Periodo estimado:	2 meses y 7 días (67 días - 70 horas)
Descripción:	Escribir la memoria durante el desarrollo de manera paralela.
Precedentes:	Paquetes de trabajo 2, 3, 4, 5
Salidas:	Memoria escrita.
Recursos necesarios:	Overleaf (LaTeX), draw.io, tablesgenerator.com

Tabla 34: Descripción de la tarea 6.1

6.2 Revisón de la Memoria	
Paquete de Trabajo:	6. Memoria
Periodo estimado:	16 días (20 horas)
Descripción:	Revisión de la memoria durante su redacción.
Precedentes:	Tarea 6.1
Salidas:	Memoria revisada.
Recursos necesarios:	Overleaf (LaTex)

Tabla 35: Descripción de la tarea 6.2

6.3 Preparación de la defensa	
Paquete de Trabajo:	6. Memoria
Periodo estimado:	8 días (14 horas)
Descripción:	Realización de la presentación y preparación de la defensa.
Precedentes:	Paquetes de trabajo 6.1 y 6.2
Salidas:	Defensa preparada.
Recursos necesarios:	Presentaciones de Google, Documentos de Google.

Tabla 36: Descripción de la tarea 6.3