

**UAX**

**TRABAJO FIN DE MÁSTER**

**MUIA  
24-25**

**UNIVERSIDAD ALFONSO X EL SABIO**

**Business Tech**

**Máster Universitario en Inteligencia Artificial**



## **TRABAJO DE FIN DE MÁSTER**

**Traductor de Lengua de Signos Española mediante  
reconocimiento de imágenes y detección de  
objetos.**

**Adrián Guiral Mallart  
Beatriz Magán y Ángel Albarracín**

**Agosto 2025**

## RESUMEN

La lengua de signos es el principal método de comunicación para personas sordas y sordomudas y es esencial para garantizar la inclusión de estas personas en una sociedad en expansión y cada vez más diversificada. En España, la lengua de signos se remonta incluso hasta el siglo XVII, siglo en el cual está datado el considerado como primer tratado moderno para la enseñanza oral de sordos. Desde entonces, la lengua de signos española no ha hecho más que evolucionar y actualmente se encuentran variantes de la lengua por comunidades, imitando la propia historia lingüística de la península.

En este trabajo se presenta el desarrollo de herramientas para la detección del alfabeto de la lengua de signos española que permitan la traducción de estos signos a la población general y que permitan una introducción amigable a la lengua a personas no signantes o aprendiendo la lengua.

El conjunto de herramientas ha sido entrenado con una colección de imágenes del alfabeto de la lengua de signos española. En esta colección a cada letra le corresponden alrededor de 100 muestras que han sido aumentadas a través de la generación sintética de datos. Las herramientas desarrolladas son una CNN básica, un Transformer visual (ViT) y la arquitectura YOLO, todas han sido pre entrenadas con ‘datasets’ generales y en todas se ha realizado un ‘fine-tuning’.

Los resultados obtenidos junto con la comparación de los modelos implementados han resaltado la importancia del conjunto de datos elegido y la eficacia del ‘fine-tuning’ de modelos pre entrenados en la tarea del reconocimiento y detección de signos. Tras la comparación realizada, YOLO es el modelo que ha brillado sobre el resto destacando por su alto rendimiento y su enfoque en la detección de objetos en tiempo real. Sin embargo, el conjunto de datos ha sido un gran limitante y ha tenido un impacto negativo

en la calidad de la solución. Disponiendo de un conjunto de datos más amplio y robusto, es muy probable que la solución implementada pudiese detectar los signos de la Lengua de Signos Española de forma más precisa.

## Palabras clave

- Transformers
- CNN
- Reconocimiento de Imágenes
- Lengua de Signos
- Detección de Objetos

## ABSTRACT

Sign language is the primary method of communication for deaf and deaf-mute individuals, and it is essential to ensure the inclusion of these people in an expanding and increasingly diverse society. In Spain, sign language dates back as far as the 17th century, the century in which what is considered the first modern treatise for the oral education of deaf people was written. Since then, Spanish Sign Language has only continued to evolve, and today there are regional variants of the language across different communities, mirroring the linguistic history of the Iberian Peninsula itself.

This work presents the development of tools for the detection of the Spanish Sign Language alphabet, enabling the translation of these signs for the general population and offering a friendly introduction to the language for non-signers or those learning it.

The toolset has been trained using a collection of images of the Spanish Sign Language alphabet. In this collection, each letter corresponds to approximately 100 samples, which have been augmented through synthetic data generation. The developed tools include a basic CNN, a Vision

Transformer (ViT), and the YOLO architecture; all have been pre-trained on general datasets and fine-tuned for this specific task.

The results obtained, together with the comparison of the implemented models have highlighted the importance of the chosen dataset and the effectiveness of fine-tuning pre-trained models in the task of sign detection and recognition. After the comparison, YOLO stood out as the most outstanding model due to its high performance and its focus on real-time object detection. Nevertheless, the dataset has been a major obstacle and has impacted negatively on the quality of the solution given. With access to a broader and more robust dataset, it is highly likely that the implemented solution could detect Spanish Sign Language signs with greater accuracy.

## Keywords

- Transformers
- CNN
- Image Recognition
- Sign Language
- Object Detection

# ÍNDICE

<b>Capítulo 1: Introducción al Trabajo Fin de Máster .....</b>	<b>7</b>
1. Justificación del proyecto realizado.....	8
2. Presentación del proyecto y sus contenidos.....	9
<b>Capítulo 2: Objetivos del TFM.....</b>	<b>11</b>
1. Objetivo general .....	11
2. Objetivos específicos.....	11
<b>Capítulo 3: Marco Teórico.....</b>	<b>12</b>
1. LSE.....	12
2. CNN.....	13
3. Transformers.....	16
3.1. Embeddings .....	16
3.2. Arquitectura.....	17
3.3. Mecanismo de atención .....	19
4. Vision Transformer (ViT).....	20
5. YOLO.....	21
6. Estado del Arte.....	22
6.1. Análisis de Vanguardia .....	22
6.2. Estado del arte de la interpretación de signos .....	24
6.2.1. Detección de signos utilizando Leap Motion .....	25
6.2.2. Sistema traductor basado en FPGA.....	27
6.2.3. Reconocimiento Gestual con Kinect.....	28
6.2.4. Reconocimiento de la ASL mediante guante sensorial.....	29
6.2.5. Comparación del proyecto realizado con el estado del arte.....	31
<b>Capítulo 4: Marco Metodológico.....</b>	<b>32</b>
1. Origen de los datos.....	32
2. Arquitecturas .....	33
3. Desarrollo.....	34
3.1. EDA .....	34
3.2. Generación de datos sintéticos .....	36
3.3. Implementación CNN no pre entrenada.....	37

3.4. Implementación ViT .....	39
3.5. Implementación YOLO .....	41
<b>Capítulo 5: Presentación de análisis, resultados y discusión .....</b>	<b>43</b>
1. Resultados obtenidos .....	43
1.1. CNN no pre entrenada .....	43
1.2. ViT .....	45
1.3. YOLO .....	46
2. Discusión y análisis de los resultados .....	47
<b>Capítulo 6: Conclusiones .....</b>	<b>51</b>
1. Limitaciones .....	52
2. Prospectiva .....	53
3. Consideraciones finales .....	54
<b>Bibliografía .....</b>	<b>56</b>

## Capítulo 1: Introducción al Trabajo Fin de Máster

En un mundo cada vez más interconectado y en continua expansión, la comunicación constituye un pilar fundamental para el desarrollo tanto social como educativo y profesional de las personas. No obstante, existen barreras lingüísticas que afectan de forma directa a algunos colectivos, limitando su participación en la sociedad. En el contexto español, la Lengua de Signos Española (LSE) constituye un sistema lingüístico complejo, con gramática y léxico propios. A pesar de su difusión y la existencia de variantes por regiones de la península, la gran mayoría de personas oyentes no tiene conocimientos suficientes sobre la LSE.

Los avances tecnológicos en el campo del reconocimiento de imágenes y la detección de objetos resultan en que la implementación de herramientas y sistemas capaces de procesar información visual esté al alcance de todo aquel con conocimientos en el campo. Estos progresos abren las puertas al diseño de herramientas capaces de identificar los signos en tiempo real e imagen.

Por ello, este proyecto se centra en la investigación sobre la lengua de signos española y el reconocimiento de objetos en video para desarrollar un modelo de reconocimiento automático de signos y su traducción a texto de forma que se pueda dar más visibilidad a la lengua de signos, animar a su aprendizaje y uso y facilitar la comunicación entre personas signantes y no signantes.

Este capítulo consta de dos puntos: una justificación del proyecto y una presentación de este y sus contenidos. La justificación del proyecto contiene una contextualización del problema, una descripción de la situación actual y las ventajas que presenta el desarrollo del proyecto tanto desde el punto de vista social como tecnológico. La presentación consta de una descripción simple del problema a solucionar y la solución implementada seguido de un breve resumen de los contenidos que se tratan en este documento.

## 1. Justificación del proyecto realizado

Durante gran parte de la historia, los sordos han sido objeto de marginación, rechazo y olvido. Una persona sorda se consideraba completamente incapaz por no poseer la capacidad de la audición o del habla. El derecho romano negaba a las personas sordas de nacimiento el derecho a firmar un testamento porque se presumía que no entendían nada. A su vez, se les creía negados de fe como afirmaba San Agustín: "Aquel que no tiene oído no puede oír, y el que no puede oír jamás podrá entender, y la falta de oído desde el nacimiento impide la entrada de la fe". Aristóteles sostenía que los sordos de nacimiento carecen de ideas morales y capacidad de pensamiento abstracto; "pueden dar voces, mas no pueden hablar palabra alguna" (Dayas, 2020).

En la actualidad, a pesar de los avances logrados en inclusión como sociedad, siguen existiendo obstáculos y dificultades para personas sordas, sordomudas y mudas. La barrera lingüística es tangible y aunque cada vez más delgada, continúa siendo un muro que sortear. Las personas sordas siguen enfrentándose a espacios no adaptados y situaciones en las que hacerse oír es un reto. La necesidad de un traductor automático de la lengua de signos española radica en dos perspectivas: la social y la tecnológica. Desde la perspectiva social, la falta de accesibilidad comunicativa para la comunidad sorda actúa como una forma de exclusión que afecta a su calidad de vida. En España, miles de personas dependen de la LSE como principal medio de comunicación según datos de la Confederación Estatal de Personas Sordas (CNSE). El lento aprendizaje de la lengua de signos por parte de la población



oyente se traduce en limitaciones de acceso a servicios básicos, educación y participación social plena. Desde el punto de vista tecnológico, la implementación de un sistema de traducción de la LSE supone un desafío interesante, además el potencial de escalabilidad y adaptación junto con la variedad de integraciones posibles como entornos educativos, televisivos, administrativos contribuye al desarrollo del estado del arte en el reconocimiento automático de la lengua de signos.

Cuando la sociedad avanza, la tecnología avanza y viceversa, hasta se podría considerar una relación simbiótica entre las dos partes, a veces beneficiosa y en algunos casos perjudicial. Es nuestro deber como desarrolladores/as e investigadores/as promover el avance y asegurar un resultado beneficioso para la sociedad, no solo el total si no para cada miembro.

## 2. Presentación del proyecto y sus contenidos

Garantizar la inclusión de la comunidad de personas sordas en una sociedad en constante cambio y expansión supone un reto que merece ser abordado. La implementación de herramientas para la detección automática de signos puede suponer una ayuda para superar este reto y romper las barreras lingüísticas presentes.

En primer lugar, se explicará el objetivo que busca el desarrollo del proyecto desglosado en objetivos específicos que se completaran a lo largo del desarrollo.

En segundo lugar, se profundizará el marco teórico alrededor de las tecnologías que se van a utilizar para cumplir con el desarrollo. También se explicará el contexto del problema y la historia de la Lengua de Signos Española (LSE). En otras palabras, se contextualizará el problema usando la historia de la LSE y su situación actual y después se explicarán las tecnologías utilizadas: CNN, Transformers y la arquitectura YOLO.

Después, se ahondará en el marco metodológico, es decir, se explicará que diseño de solución se ha elegido, cual es el origen de los datos utilizados, que arquitecturas se han elegido y como se ha llevado a cabo la implementación.

A continuación, se estudiarán los resultados y se compararán a lo largo de las arquitecturas elegidas.

Por último, se expondrán las conclusiones sobre el proyecto desarrollado y sobre los resultados de las herramientas argumentando si contribuye a solucionar el problema.

## Capítulo 2: Objetivos del TFM

En este capítulo se va a exponer el objetivo general y un desglose en objetivos específicos que en su conjunto forman el objetivo general y final de este TFM.

### 1. Objetivo general

- Comparar las distintas tecnologías actuales de reconocimiento y detección de objetos y desarrollar un sistema automatizado de detección del alfabeto de la Lengua de Signos Española.

### 2. Objetivos específicos

1. Encontrar una base de datos de la Lengua de Signos Española.
2. Generar datos sintéticos para ampliar la base de datos.
3. Investigar qué modelos son la punta de lanza actual para la tarea de detección de objetos.
4. Diseñar una arquitectura propia de CNN.
5. Implementar un ViT para detección de objetos.
6. Hacer fine-tuning de la arquitectura YOLO para detectar signos en tiempo real.
7. Comparar las implementaciones y estudiar sus precisiones.

## Capítulo 3: Marco Teórico

En este capítulo se busca realizar un estudio e investigación acerca de las redes neuronales convolucionales, los Transformers visuales y la arquitectura YOLO (You Only Look Once). Estos modelos son esenciales en el reconocimiento de imágenes y detección de objetos y van a ser útiles a la hora de conseguir el objetivo de este proyecto. A su vez, se busca dar contexto del tema del proyecto. Además, se realizará una comparación de los modelos más utilizados hoy en día y sus capacidades, es decir, un análisis del estado del arte actual.

### 1. LSE

En primer lugar, se va a contextualizar el tema, en este caso, la Lengua de Signos Española (LSE), lengua gestual que se utiliza a lo largo de la península ibérica y posee inteligibilidad mutua con el resto de las lenguas de signos de la península como por ejemplo la lengua gestual portuguesa. En España existen variedades como la catalana, valenciana, andaluza, canaria, gallega y vasca.

El origen de la lengua de signos española se remonta al 1851, año en el que se publicó según muchos el primer diccionario de la LSE llamado Diccionario Mímico y Dactilológico que en su mayor parte se trata de una traducción al castellano del diccionario francés con mismo nombre publicado en 1850 y escrito por Alejandro Blanchet, profesor del Instituto Nacional de Sordomudos de París. Por lo tanto, se trata de una lengua con historia y que nació con gran influencia de la lengua de señas francesa (LSF) y que actualmente está influenciada por la lengua de señas americana (ASL) (Wikipedia contributors, 2025).



*Ilustración 1: Alfabeto Dactilológico de la LSE.*

En la actualidad, se cree que acerca de 100.000 personas en la península son usuarios signantes, dato que refuerza la importancia de dar visibilidad e introducir al resto de la población a la lengua de signos, presente a lo largo de toda la península.

## 2. CNN

Las CNN (Convolutional Neural Network) o redes neuronales convolucionales son un tipo de red neuronal de aprendizaje profundo con un alto rendimiento en el reconocimiento de imágenes. Este rendimiento se debe a las capas ocultas que forman estas arquitecturas, entre ellas, las capas de convolución.

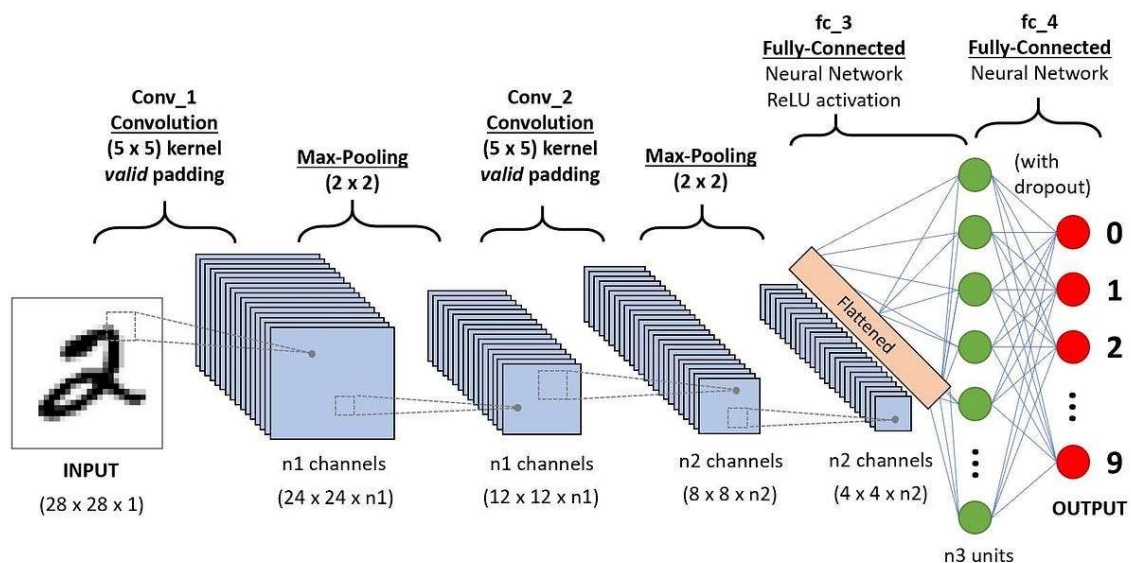
La convolución es una operación que consiste en combinar dos funciones y describir la superposición entre ambas.

$$(f * g)(t) \approx^{def} \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

En el procesamiento de imágenes, se utiliza para detección de bordes, aumento de nitidez, desenfoque de imágenes, etc. Se consigue aplicando una matriz de convolución sobre la imagen.

La principal diferencia entre una CNN y una ANN (Artificial Neural Network) convencional es que las capas de una CNN se componen de tres dimensiones basadas en la entrada (imágenes): altura, anchura y profundidad siendo esta última el número de canales de color de una imagen, por ejemplo, azul, rojo y verde (O'Shea & Nash, 2015).

Las CNNs están formadas por tres tipos de capas: capas convolucionales que son las encargadas de realizar la operación de convolución, capas de 'pooling' y capas densas (Jain, 2024).



*Ilustración 2: Ejemplo de Arquitectura CNN para MNIST Dataset*

En la ilustración se observan las distintas capas que componen la arquitectura:

1. La capa de entrada que contiene la información de los píxeles de la imagen.
2. La capa convolucional que se encarga de realizar la operación de convolución.
3. La capa 'pooling' que reduce el tamaño aplicando un filtro, en este caso, el máximo.
4. La capa densa que transforma la tridimensionalidad de los datos a una sola dimensión interpretable por el modelo y que coincide con el número de clases del problema, en este caso, una por dígito (10).

Las capas convolucionales usan 'kernels' con menor dimensionalidad que se desplazan por el input aplicando un input y resultando en un mapa 2D de activación. De esta forma, los 'kernels' aprenderán características en posiciones específicas del input.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 2 & 2 \\ 0 & 3 & 2 & 3 \\ 3 & 1 & 5 & 4 \\ 0 & 3 & 2 & 3 \end{pmatrix}$$

Input matrix of 6 × 6      Kernel of 3 × 3      Activation map

*Ilustración 3: Aplicación de Kernel*

En la ilustración se puede observar una aplicación de un 'kernel' 3x3 sobre un input 6x6 que resulta en una activación con valor 2.

Las capas de 'pooling' reducen el tamaño del input operando sobre el mapa de activación generado por la capa de convolución. Normalmente se aplican 'kernels' con dimensionalidad 2x2 con un desplazamiento de 2 que navegan por el mapa de activación realizando una operación como el máximo en el caso de las capas 'max-pooling' reduciendo de esta forma el tamaño del input un 25%. Existen otros tipos de capas 'pooling' dependiendo de la operación que realizan. Por un lado, 'overlapping pooling' que tiene un tamaño de 2x3, un

desplazamiento de 2 y recibe su nombre ya que al desplazar el ‘kernel’ la operación reutiliza información de la operación anterior. Por otro lado, ‘general pooling’ que realiza operaciones como la regularización L1 o L2 y ‘average pooling’ (media) (O’Shea & Nash, 2015).

Por último, las capas densas se conectan enteramente a la capa anterior pero no entre ellas, de esta forma reducen la dimensionalidad para que coincida con un output entendible y alineado por ejemplo con la cantidad de clases en los datos de entrada.

### 3. Transformers

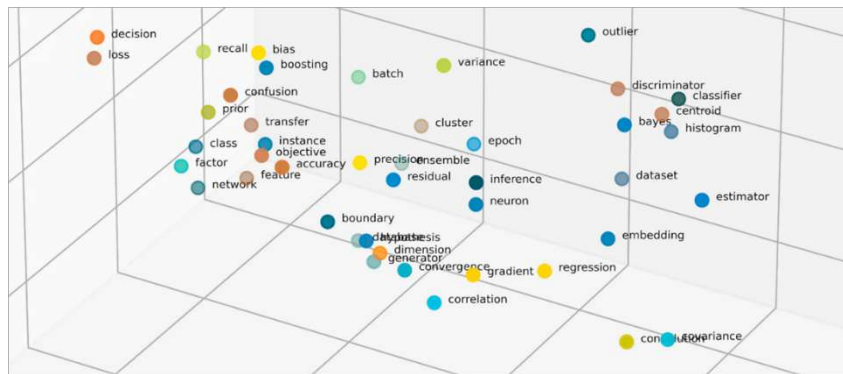
Antes de hablar de Transformers, se debe comprender que es un ‘embedding’ ya que se trata de una técnica fundamental para el funcionamiento de los Transformers.

#### 3.1. Embeddings

Los ‘embeddings’ son vectores numéricos que mapean información compleja en espacios vectoriales de menor dimensión como por ejemplo una palabra en el contexto de un lenguaje. Esta técnica reduce la complejidad y captura características de la información sin conocimiento previo del contexto. Estas representaciones capturan relaciones semánticas, estructurales o contextuales entre los datos en un espacio vectorial continuo.



Los ‘embeddings’ son útiles a la hora de representar datos como texto que no pueden utilizarse directamente en aprendizaje automático porque no tienen una estructura matemática explícita. Uno de los ejemplos más conocidos es el modelo de generación de ‘embeddings’ ‘Word2Vec’ desarrollado por Google que transforma texto en ‘embeddings’ por palabras. Se trata de un modelo entrenado para aprender las relaciones entre palabras. Los ‘Word Embeddings’ son esenciales para el entrenamiento de modelos como ‘Chat GPT’.

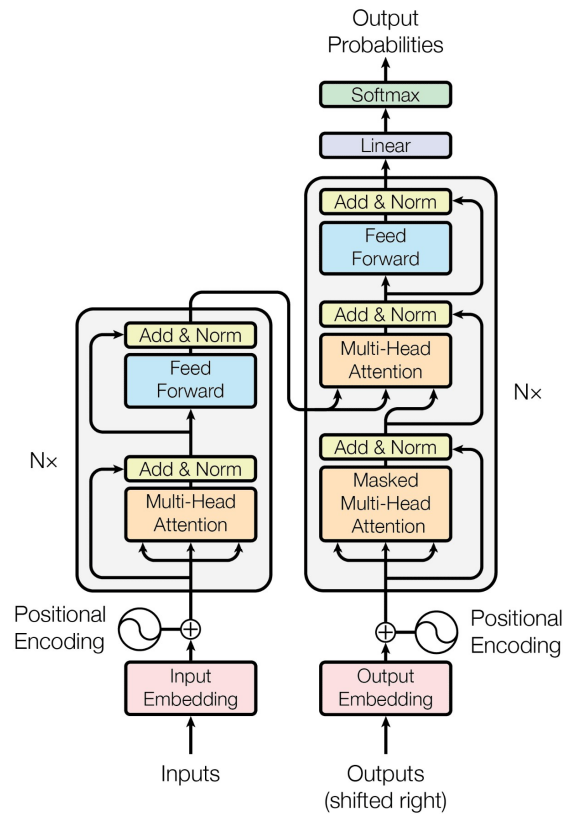


*Ilustración 4: Espacio Vectorial generado a partir de Word Embeddings.*

### 3.2. Arquitectura

En 2017, investigadores de Google publicaron un artículo llamado ‘Attention Is All You Need’ en el que describieron una arquitectura a la que dieron el nombre Transformer, se trataba de una arquitectura ‘encoder-decoder’ basada en un sistema de atención. Se propuso como una solución a la pérdida de información entre ‘inputs’ muy distantes en redes neuronales recurrentes y LSTM (Long Short-Term Memory), un problema inherente a estos modelos en secuencias muy largas. Estos modelos eran y siguen siendo considerados estado del arte para problemas como la traducción y el modelado del lenguaje, pero con la introducción de mecanismos de atención en arquitecturas como el Transformer se logra capturar dependencias globales entre inputs y outputs independientemente de su distancia.

La arquitectura propuesta por los investigadores fue la siguiente:



*Ilustración 5: Arquitectura Transformer*

- **Encoder:** El encoder está compuesto por 6 capas idénticas divididas en 2 subcapas. Estas dos capas son un mecanismo de auto atención multi-cabeza seguida de una red 'feed-forward' totalmente conectada. Ambas subcapas tienen conexiones residuales seguidas de una normalización. Las conexiones residuales permiten saltar a cualquiera de las subcapas en caso necesario (Vaswani et al., 2017).
- **Decoder:** Al igual que el 'encoder', el 'decoder' está compuesto por 6 capas idénticas con la diferencia de que el 'decoder' tiene una tercera subcapa de atención multi-cabeza que se realiza sobre el output del 'encoder'. Para asegurar que las predicciones para una posición

dependen únicamente de las posiciones anteriores, los ‘embeddings’ de salida se desplazan una posición (Vaswani et al., 2017).

### 3.3. Mecanismo de atención

Se trata de una función de mapeado entre una ‘query’ y una colección de ‘key-values’ (estructura de mapa) emparejados con una salida. Tanto la ‘query’ como las llaves, valores y la salida (output) son vectores. La salida es la suma ponderada de los valores, cada valor se calcula con una función de la ‘query’ y la llave correspondiente.

Existen varios mecanismos de atención presentes en los Transformers, entre los cuales se encuentran:

- **Scaled Dot-Product Attention:** Se realiza el producto escalar de la ‘query’ con todas las llaves, se divide por la raíz de la dimensión de las llaves y se aplica una función ‘softmax’ sobre el resultado (Vaswani et al., 2017).

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T \sqrt{d_k})V$$

- **Multi-Head Attention:** Se proyectan las ‘queries’, llaves y valores con proyecciones lineales aprendidas anteriormente donde se realiza la función de atención en paralelo. Finalmente se concatena y se vuelve a proyectar (Vaswani et al., 2017).

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W_O \text{ where } \text{head}_i = \text{Attention}(QW_{Q_i}, KW_{K_i}, VW_{V_i})$$

## 4. Vision Transformer (ViT)

Para utilizar los Transformers en problemas de visión por computadora, se sigue la misma arquitectura original propuesta por los investigadores de Google en 2017 pero se transforman los datos de entrada para adaptarlos a la restricción de input del Transformer. La arquitectura original recibe como input una secuencia de una dimensión de ‘token embeddings’ lo cual genera la necesidad de transformar las imágenes 2D en secuencias de parches 2D aplanados (con dimensión reducida):

$$x_p \in \mathbb{R}^{N \times (P^2 \times C)}$$

Donde:

- (H, W) es la resolución original de la imagen
- C es el número de canales
- (P, P) es la resolución de cada parche
- $N = HW / P^2$  siendo N el número de parches resultante (longitud de secuencia de entrada para el Transformer) (Dosovitskiy et al., 2020)

Tras esta primera transformación, cada parche se proyecta a una dimensión de tamaño fijo D, la cual se utiliza a lo largo de las capas del Transformer. A su vez, cada uno de los parches recibe un token especial al principio de su secuencia que representa a la imagen y se utiliza para su clasificación. Esto se complementa con la adición de ‘embeddings’ de posición que guardan información relativa a la posición (Dosovitskiy et al., 2020).

De esta forma se puede utilizar la arquitectura Transformer en problemas de visión de computadora, aprovechando todas sus fortalezas como el modelado

global de relaciones espaciales, la escalabilidad de los datos, la facilidad de transferencia entre tareas y la eliminación de operaciones como la convolución (alto coste computacional).

## 5. YOLO

‘You Only Look Once’ o abreviado YOLO es una arquitectura de red neuronal convolucional enfocada en la detección de objetos en tiempo real, una de las principales tareas de visión por computador. YOLO enfoca la detección de objetos como un problema de regresión lo cual permite una predicción directa de probabilidades de clase a partir de imágenes completas mediante una sola red neuronal.

YOLO está basada en una red neuronal convolucional profunda inspirada en ‘GoogLeNet’ pero simplificada para obtener mejor rendimiento a tiempo real. La arquitectura incluye 24 capas convolucionales seguidas de 2 capas densas y se suele pre entrenar con el dataset ‘ImageNet’ (colección de imágenes muy grande). El modelo utiliza el error cuadrático medio (MSE) como función de pérdida sobre las predicciones y utiliza técnicas como la regularización, ‘dropout’ y generación sintética de datos (transformaciones geométricas y de color) para evitar ‘overfitting’ (Redmon et al., 2015).

Las ventajas de YOLO frente a otros modelos como ‘R-CNN’ y ‘Fast R-CNN’ son las siguientes:

- **Velocidad:** Procesa imágenes a 45fps en tiempo real.
- **Simplicidad:** Se entrena en una única red neuronal.
- **Generalización:** Buena capacidad para generalizar en dominios distintos al de entrenamiento.

En resumen, YOLO representa un cambio de paradigma en la detección de objetos al ofrecer una arquitectura unificada y extremadamente rápida. Su simplicidad estructural, capacidad para ser entrenado ‘end-to-end’, y rendimiento en tiempo real lo convierten en una herramienta poderosa tanto para investigación como para aplicaciones industriales. Si bien tiene limitaciones en la localización precisa y detección de objetos pequeños, su eficiencia y capacidad de generalización abren la puerta a numerosos escenarios de uso como la robótica, la conducción autónoma y la realidad aumentada.

## 6. Estado del Arte

El apartado consiste en un análisis de los modelos a la vanguardia para tareas de visión por computador como el reconocimiento de imágenes y la detección de objetos seguido de un análisis de proyectos encontrados sobre la detección de signos tanto de la LSE como lenguas de signos extranjeras.

### 6.1. Análisis de Vanguardia

Para finalizar el marco teórico se van a exponer los principales modelos especializados en el reconocimiento de imágenes y detección de objetos que hoy en día componen el inventario de modelos a la vanguardia. Entre estos modelos encontramos desde CNNs como la propia arquitectura YOLO hasta Transformers visuales.

En la actualidad, el desarrollo de arquitecturas para tareas de visión por computadora ha crecido de forma exponencial impulsado por el descubrimiento de arquitecturas como los ya mencionados Transformers, los avances en arquitecturas ya existentes como las CNNs y el avance en técnicas de preentrenamiento. Este crecimiento viene acompañado de la amplia disponibilidad de grandes volúmenes de datos. Para analizar el estado del

arte, el artículo *Battle of the Backbones* (Goldblum et al., 2023) presenta un estudio comparativo de las distintas arquitecturas y va a referenciarse múltiples veces a lo largo de este apartado. Este estudio evalúa arquitecturas pre entrenadas bajo distintas técnicas como el aprendizaje supervisado, auto supervisado o modelos generativos. Las arquitecturas se han evaluado en múltiples tareas de visión por computador, desde la clasificación de imágenes hasta la detección y segmentación de objetos pasando por la recuperación de imágenes y la generalización ‘Out of Distribution’ (capacidad del modelo de mantener buen rendimiento en datos fuera de los datos de entrenamiento).

Las principales arquitecturas evaluadas son las siguientes (Goldblum et al., 2023):

Pretraining Style	Dataset	Architecture(s)
MoCo v3 [12]	ImageNet-1k [17]	ViT [18]
VICReg [3]	ImageNet-1k	ResNet [28]
VICRegL [4]	ImageNet-21k	ConvNeXt [58]
DINO [8]	ImageNet-1k	ResNet, ViT
MAE [30]	ImageNet-1k	ViT
Stable Diffusion [77]	LAION-2B [81]	Stable Diffusion encoder
CLIP [73]	LAION-2B, CLIP	ResNet, ViT
MiDaS [75]	12 × Depth Datasets	SwinV2 [57]
Image classification	ImageNet-21k, ImageNet-1k	Todas las anteriores
Random initialization	N/A	Todas las anteriores

El estudio destaca que las redes convolucionales modernas como ConvNext entrenadas de forma supervisada sobre conjuntos como ImageNet-1k, ImageNet-21k son altamente competitivas, superando a Transformers como ViT y SwinV2. Especialmente, ConvNext destaca por su robustez en tareas de clasificación, detección, segmentación y generalización OOD (Goldblum et al., 2023).

El estudio también destaca el papel creciente del preentrenamiento lengua-visión, una técnica que consiste en el entrenamiento simultáneo de imágenes con textos asociados en grandes cantidades para captar la relación entre el contenido visual y el lenguaje natural (Gan et al., 2022), representado por modelos como CLIP (Contrastive Language-Image Pretraining). CLIP y Stable Diffusion demuestran una notable capacidad de generalización semántica en escenarios fuera de distribución (OOD) y tareas de recuperación visual. CLIP, en particular es altamente competitivo gracias a su preentrenamiento contrastivo multimodal sobre un corpus inmenso de imágenes y texto (LAION-2B) aunque sufre limitaciones en tareas que requieren localización espacial precisa (Goldblum et al., 2023).

Asimismo, destaca la capacidad de escalado de los ViTs con más cantidad de datos y parámetros, lo cual puede indicar que a medida que la disponibilidad de datos aumente, tendencia creciente en el último siglo, los Transformers tomarán un papel dominante obteniendo mejores rendimientos en estas tareas (Goldblum et al., 2023).

En caso de que el coste computacional sea una barrera limitante, los modelos EfficientNet-B0, RegNetX-400MF y ResNet-18, modelos pequeños en comparación al resto, destacan como los más competitivos con bajo coste computacional.

Finalmente, el estudio enfatiza que no existe un único modelo dominante en todas las tareas. En cambio, se observa correlación entre tareas similares, lo que sugiere que la elección de la arquitectura debe estar guiada por el contexto de aplicación y las restricciones prácticas como por ejemplo el coste computacional o la disponibilidad de datos.

## 6.2. Estado del arte de la interpretación de signos

Tras el análisis de los modelos más potentes hoy en día y poniendo la vista en proyectos enfocados a la detección de signos en tiempo real, se han encontrado



proyectos enfocados a la interpretación de signos, algunos de ellos intentos para la LSE y otros para lenguas de signos extranjeras.

### 6.2.1. Detección de signos utilizando Leap Motion

El controlador de movimiento 'Leap Motion' es un módulo óptico de seguimiento encargado de capturar el movimiento de los manos y dedos de los usuarios para interactuar con contenido digital. El controlador es capaz de identificar 27 elementos distintos de la mano incluyendo huesos y articulaciones. Las principales aplicaciones del controlador 'Leap Motion' son las herramientas y juegos de realidad virtual pero también se puede utilizar como dispositivo de navegación del ordenador, imitando la funcionalidad de un ratón.



*Ilustración 6: Uso de Controlador Leap Motion para diseño gráfico 3D.*

El artículo *Arabic Static and Dynamic Gestures Recognition Using Leap Motion* publicado por Basma Hisham y Alaa Hamouda en la Universidad de Al-Azhar expone los resultados de la implementación de un sistema para el

reconocimiento de signos de la Lengua de Signos Árabe (ArSLR) utilizando el controlador ‘Leap Motion’ tanto para la colecta de datos como para la clasificación y reconocimiento.

En el proyecto mapearon en vectores las posiciones y direcciones de los dedos al realizar los signos y utilizaron dichos vectores para entrenamiento de los modelos utilizados. Se diferenciaron los signos estáticos de los dinámicos y se aplicó un modelo distinto para cada categoría.

Para los signos estáticos se utilizaron ‘Support Vector Machines’ (SVM), conjunto de algoritmos de aprendizaje supervisado cuyo objetivo es dividir el espacio de datos con el hiperplano más óptimo y están especializados en problemas de clasificación y regresión, ‘KNN’ o k-vecinos más cercanos, algoritmo de aprendizaje supervisado no parametrizado que funciona etiquetando los datos en base a los ‘k’ puntos de datos cercanos más similares y ANN, redes neuronales artificiales.

En cambio, para los signos dinámicos se utilizó ‘Dynamic Time Warping’ (DTW), algoritmo que mide la similitud entre dos secuencias temporal que pueden variar en velocidad y ha sido utilizado en tareas de reconocimiento de voz, minería de datos y visión por computador (Wikipedia contributors, 2025).

Los resultados arrojados por el proyecto destacan la eficacia de las máquinas de soporte vectorial y el algoritmo KNN, algoritmos con implementación sencilla y que alcanzan precisiones entorno al 97%-98% en signos estáticos de la Lengua de Signos Árabe. En el caso de los signos dinámicos, el DTW también aprueba como un algoritmo válido, alcanzando precisiones de entre el 89%-95% de acierto. A su vez, el artículo remarca la utilidad de herramientas como el controlador ‘Leap Motion’ para el mapeo de la estructura de la mano y dedos y la fiabilidad que otorga a la hora de entrenar modelos (Hisham & Hamouda, 2017).

### 6.2.2. Sistema traductor basado en FPGA

La FPGA (field-programmable gate array) es un conjunto de circuitos integrados programable compuesto por bloques de lógica cuya funcionalidad es configurable. El chip puede ser programado en cualquier momento y puede realizar funciones tales como la de una puerta lógica hasta sistemas complejos.

El artículo *Sistema traductor de la lengua de señas colombiana a texto basado en FPGA* publicado por Juan David Guerrero-Balaguera y Wilson Javier Pérez-Holguín en la Universidad Nacional de Colombia presenta el desarrollo de un sistema de reconocimiento de signos de la Lengua de Señas Colombiana (LSC) desde un enfoque de la implementación del hardware junto con un preprocesamiento de datos e implementación de un modelo de red neuronal perceptrón multicapa. El artículo expone los pasos realizados para la implementación del sistema, desde la segmentación de las imágenes mediante umbralización hasta el diseño de la red neuronal pasando por la extracción de características.

En el artículo se explica cómo se han capturado imágenes con contraste y buena iluminación para facilitar la elección de un umbral en una escala de grises que aisle la mano del fondo. Tras la umbralización, se comenta como se extrajeron las características transformando la imagen en una matriz binaria en la cual los unos representan píxeles de la mano y los ceros píxeles del fondo de la imagen. A partir de esta matriz calcularon vectores columna y fila resultantes de la suma de los valores de los píxeles (G. B. J. & P. H. W., 2015).

Después del preprocesamiento, se entrenó una red neuronal sencilla utilizando MATLAB. La red neuronal consistía únicamente de una capa oculta y realizaron pruebas variando el número de neuronas de la capa oculta.

Finalmente, obtuvieron buenos resultados, precisiones de entre el 97% y el 100% para este tipo de datos. Un dato resaltado en el estudio es el bajo uso de recursos utilizado por la FPGA, siempre inferiores al 50% de la capacidad.

### 6.2.3. Reconocimiento Gestual con Kinect

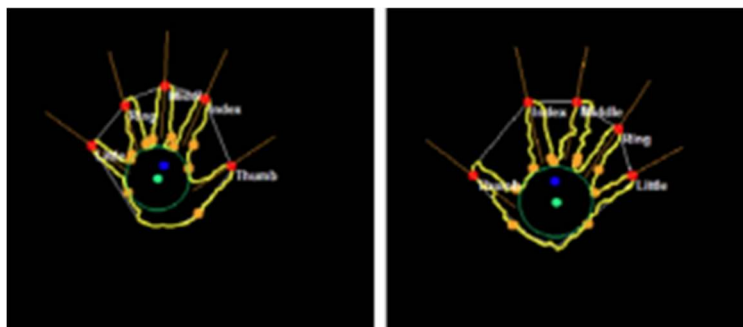
El controlador Kinect es un dispositivo que cuenta con una cámara RGB, un micrófono, un sensor de profundidad y un procesador. Kinect utiliza reconocimiento de voz y reconocimiento de movimiento, se lanzó en 2013 al mercado como un controlador que permitía a los usuarios interactuar con la interfaz y videojuegos de la consola Xbox 360 sin necesidad de contacto físico.



*Ilustración 7: Controlador Kinect para Xbox 360.*

El artículo *Hand Gesture Recognition Using Kinect* publicado por Yi Li en la Universidad de Louisville explora la capacidad del sensor para el reconocimiento de gestos. La investigadora Yi Li propuso un método para la detección de gestos que pasa primero por la detección de manos y dedos y sus direcciones. La detección de las manos propuesta consiste en indicar un umbral de profundidad donde los gestos deben ser detectados y proyectar los píxeles de las manos a un espacio bidimensional donde utilizando el algoritmo 'K-means' se agrupan los contornos de las manos, siendo cada mano una agrupación.

A su vez, en este proceso se detectaban las puntas de los dedos, utilizadas para identificar que dedos se utilizan en el gesto debido al ángulo que forman las puntas de los dedos respecto al centro de la palma de la mano (Li, 2012).



*Ilustración 8: Resultados de la identificación.*

Después de la identificación, el sistema propuesto utiliza tres capas de clasificación: una capa de conteo de dedos extendidos, otra capa que recoge los nombres de los dedos extendidos y por último una máquina vectorial que recoge las direcciones de los dedos extendidos junto con los ángulos que forman y predice el gesto realizado.

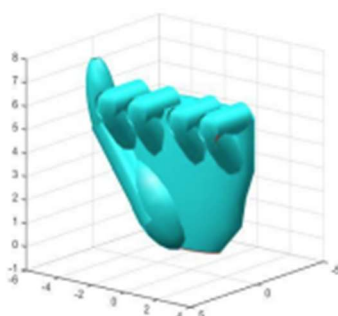
El artículo expone las pruebas realizadas para detectar 9 gestos distintos como el gesto de la victoria y el gesto de ‘Star Trek’ entre otros. Se destaca que el gesto con menor precisión de acierto es este último debido a la dificultad de realizar el propio gesto. De todas formas, el sistema desarrollado presenta precisiones de entre el 84% y 99% dependiendo del gesto.

#### 6.2.4. Reconocimiento de la ASL mediante guante sensorial

El artículo *American Sign Language Recognition Using Sensory Glove and Neural Network* publicado por Firas A. Raheem y Hadeer A. Raheem en la Universidad de la Tecnología de Bagdad presenta la implementación de un sistema de reconocimiento de la ASL (American Sign Language) basada en el

uso de redes neuronales profundas y de un guante sensorial compuesto por seis sensores: uno para cada dedo y otro para la muñeca.

En el artículo se expone como se procesaron los datos para entrenar una red neuronal profunda; se realizaron modelos 'kinematicos' de las manos realizando los signos estáticos en MATLAB.



*Ilustración 9: Representación de la letra A de la ASL en MATLAB.*

Los investigadores entrenaron una red neuronal con dos capas ocultas usando los datos preprocesados y utilizaron el guante sensorial para capturar datos y realizar pruebas. En el artículo se comenta que los resultados obtenidos por el sistema rondan el 96% de acierto (Abdulrazzaq & Raheem, 2019).



*Ilustración 10: Prueba de la letra D con guante sensorial.*

### 6.2.5. Comparación del proyecto realizado con el estado del arte

El proyecto realizado está enmarcado en un ámbito en el cual existen propuestas previas que abordan problemas similares o relacionados con el reconocimiento y la detección de signos. Resulta evidente contrastar y comparar los resultados obtenidos y el trabajo realizado con otros proyectos para identificar las contribuciones del estudio realizado.

Echando la vista a los proyectos explorados, se observa que la mayoría se enfocan en una implementación centrada o apoyada en algún tipo de ‘hardware’ mientras que el presente proyecto no requiere de ‘hardware’ específico. También destaca que todos los proyectos explorados realizaron un trabajo de construcción de datos de entrenamiento. En este sentido, el proyecto realizado introduce un enfoque centrado únicamente en el desarrollo de ‘software’, en este caso modelos, que permite una exploración más exhaustiva de este campo.

Al mismo tiempo, es necesario señalar como la mayoría de los proyectos explorados se centran en algoritmos que requieren entradas más estructuradas o preprocesadas y por ello realizan un paso previo de transformación de datos, umbralización y adaptación de los datos mientras que el presente proyecto se centra en técnicas más avanzadas propias de tareas de visión por computador, reconocimiento de imágenes y detección de objetos en tiempo real.

En conclusión, el presente proyecto destaca por el enfoque en la investigación exhaustiva para entender el problema y la comparativa realizada entre los métodos actuales en tareas de visión por computador, la gran escalabilidad de la solución y fácil integración en otros sistemas y la ausencia de la limitación o dependencia de ‘hardware’ en los proyectos explorados.

## Capítulo 4: Marco Metodológico

En este capítulo se describe la metodología seguida a la hora de desarrollar el proyecto desde el proceso de investigación hasta el diseño de la solución y su implementación.

El capítulo consta de cuatro apartados empezando por una descripción de la muestra utilizada en el estudio, entrenamiento y ajuste de las herramientas. Después se encuentra una explicación de las arquitecturas implementadas y el porqué. Seguido de las arquitecturas, se describe que métricas se han estudiado y elegido para el análisis de resultados. Por último, se describe el proceso completo para el desarrollo de las herramientas, desde el análisis de los datos y transformaciones realizadas hasta la implementación de las propias herramientas paso a paso; entrenamiento, ajustes, mejoras, pruebas, etc.

### 1. Origen de los datos

Al principio del desarrollo, se planteó usar como datos de entrenamiento el ‘dataset’ LSE\_Lex40\_UVIGO, un dataset para tareas de reconocimiento de la lengua de signos española y desarrollado por investigadores de la Universidad de Vigo. El ‘dataset’ consta de tres secciones: el alfabeto de la LSE compuesto por 30 letras signadas, 40 signos aislados tanto estáticos como dinámicos en los cuales una o dos manos intervienen con movimientos asimétricos o simétricos y diferentes orientaciones y 40 frases cortas de interacciones cotidianas (de uno a cinco signos por frase) (Docío-Fernández et al., 2020). Se descartó esta opción por la situación y disponibilidad de los datos, comentada con los propios investigadores a través de correo electrónico. Debido a la escasa disponibilidad de datos para tareas de detección de signos de la lengua de signos española y tras haber agotado cualquier otra vía, se decidió usar el único ‘dataset’ encontrado y disponible para uso público. Estos obstáculos se profundizan en el apartado de limitaciones.



Los datos provienen de un ‘dataset’ compuesto por alrededor de 100 muestras por clase (letra del alfabeto). Cada muestra es una imagen del signo. El ‘dataset’ se construyó con imágenes de tres manos distintas a la misma distancia de la cámara y diferentes ángulos. Al final del proceso, el ‘dataset’ fue revisado por una persona con conocimiento certificado en la lengua de signos española. Cabe destacar que el ‘dataset’ incluye únicamente las letras con signos estáticos, 19 letras del alfabeto, y todas las imágenes tienen un fondo blanco.

## 2. Arquitecturas

Las arquitecturas escogidas y entrenadas son las siguientes:

- Arquitectura de diseño propio de CNN
- ResNet50 con ‘fine-tuning’
- ViT pre entrenado y ‘fine-tuning’
- YOLOv11 con ‘fine-tuning’

El objetivo es comparar las distintas tecnologías de reconocimiento de imágenes y detección de objetos para encontrar la mejor solución al problema expuesto. Por este motivo, en primer lugar, se evaluaron distintas arquitecturas de CNN sin aplicar ‘fine-tuning’ con el fin de analizar el rendimiento del ‘dataset’ en combinación con una red convolucional. De esta forma, resultaron evidentes las debilidades del ‘dataset’ y la necesidad de realizar un preentrenamiento sobre un conjunto de datos más grande y después aplicar ‘fine-tuning’.

A continuación, tras observar las debilidades del ‘dataset’, se utilizó la arquitectura ResNet50, una red residual convolucional de 50 capas: 49 capas

convolucionales y 1 capa densa. ResNet50 incorpora conexiones de salto, denominadas conexiones residuales, que realizan mapeos de forma que la red aprende la diferencia entre la entrada y la salida de las capas en lugar de limitarse únicamente a la salida (He et al, 2015). Gracias a estas conexiones, cabe la posibilidad de entrenar redes más profundas sin sufrir desvanecimiento de gradiente y conseguir un mayor rendimiento.

Después, se analizó el rendimiento de la arquitectura de Vision Transformer pre entrenada en ImageNet-21k y ajustada con ‘fine-tuning’ usando el ‘dataset’.

Por último, se utilizó la arquitectura YOLO, exactamente la arquitectura pre entrenada YOLO11n y se ajustó al ‘dataset’. Esta última solución es la más cercana a un traductor de la Lengua de Signos Española ya que permite la detección de signos a tiempo real.

### 3. Desarrollo

En este apartado se explica los pasos seguidos para el desarrollo de las herramientas, desde el análisis exploratorio inicial seguido por la generación de datos sintéticos hasta la propia implementación de las soluciones.

#### 3.1. EDA

El análisis exploratorio de datos es crucial para analizar el ‘dataset’ y entender la naturaleza de los datos, su estado y calidad entre otras. En este caso, se ha analizado el balanceo de clases ya que nuestros datos son imágenes sin datos faltantes y ya etiquetadas.

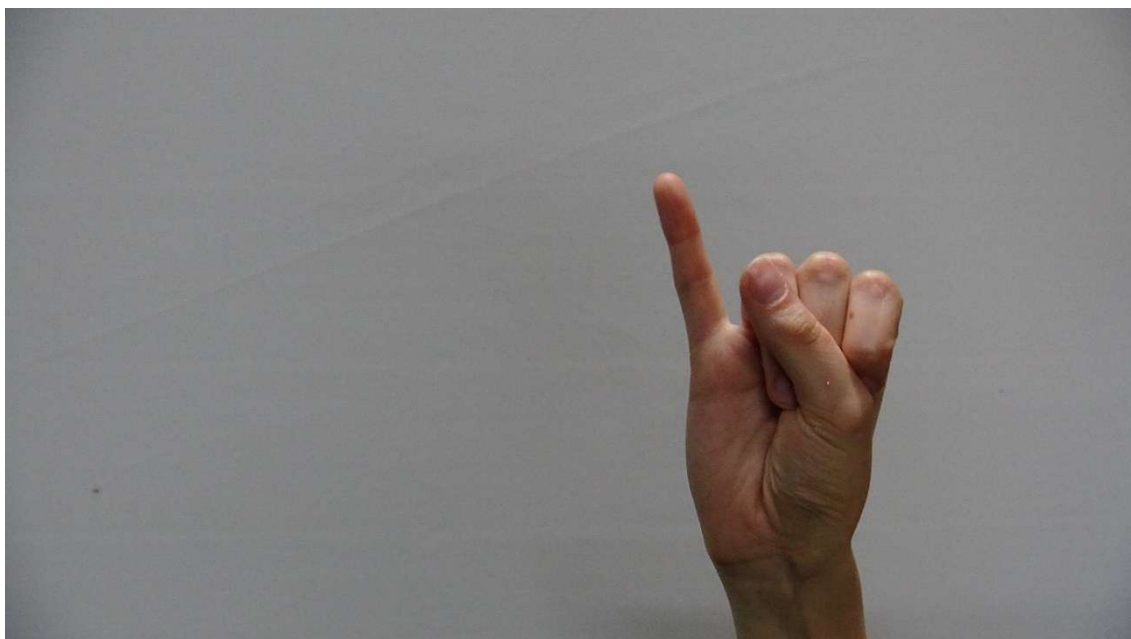


Ilustración 11: Imagen del 'dataset' del signo de la LSE para la letra i.

El 'dataset' contiene un total de 19 clases y cada una tiene entre 90 y 120 imágenes, todas en fondo blanco y con una resolución de 4288x2408, una resolución extremadamente grande para tratarlas en los modelos. Por esto, se realiza una normalización de las imágenes y se transforman a un tamaño mucho más manejable, de 512x512. Aquí se puede observar la distribución de muestras por clase:

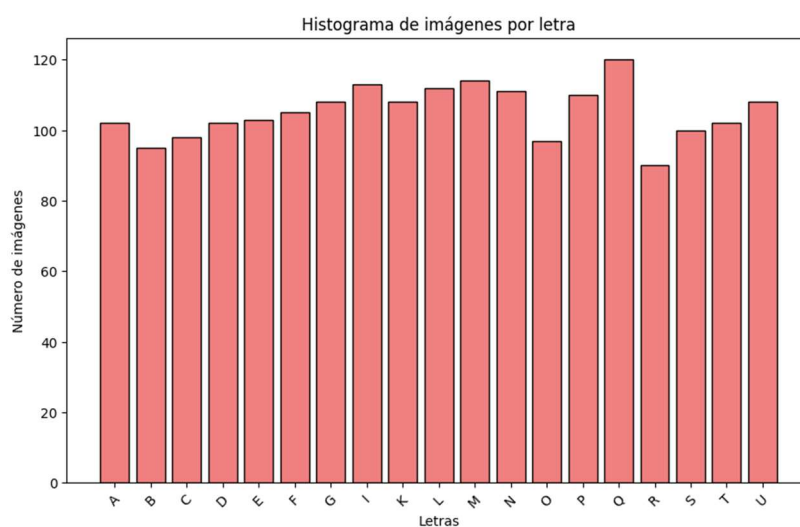


Ilustración 12: Histograma de imágenes por letra.

### 3.2. Generación de datos sintéticos

El segundo paso a la hora de conseguir un buen rendimiento en los modelos es la generación de datos sintéticos. Es un paso esencial debido a la escasa cantidad de datos disponibles. La generación de datos sintéticos ofrece la posibilidad de enriquecer el ‘dataset’ de forma sencilla.

Para generar datos sintéticos se ha realizado ‘data augmentation’, una técnica estadística que consiste en generar nuevas muestras a partir de perturbaciones en los datos manteniendo sus etiquetas. Entre las transformaciones más utilizadas se encuentran las geométricas, las transformaciones en las propiedades de color de la imagen y añadir ruido para introducir imperfecciones.

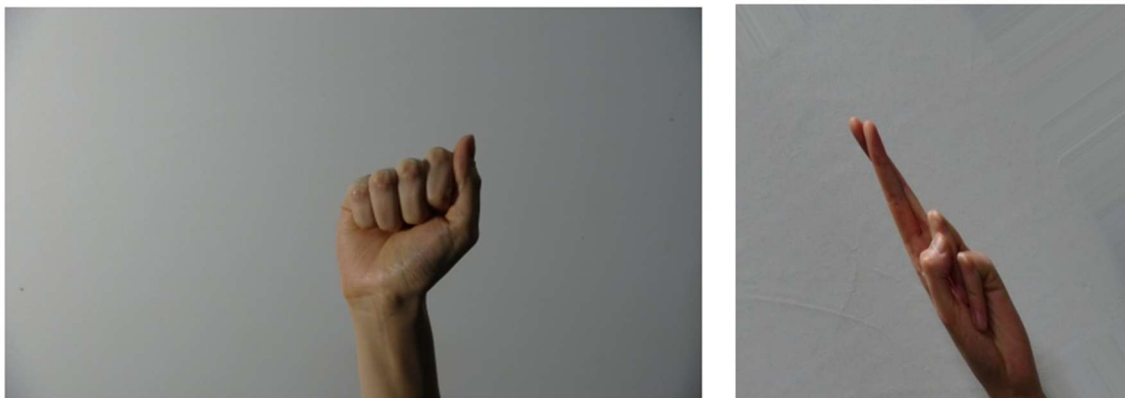
Asimismo, las transformaciones geométricas se subdividen en rotar, girar, recortar y trasladar entre otras. Por otro lado, las transformaciones de las propiedades de color de la imagen abarcan desde ajustes en el brillo, contraste y saturación hasta la introducción de vibración en el color. Por último, la adición de ruido se basa en añadir ruido Gaussiano a las imágenes o ‘Salt and Pepper Noise’ que se trata de añadir píxeles blancos y negros de forma aleatoria (Wikipedia contributors, 2025).

El ‘data augmentation’ se ha realizado con la librería de ‘keras’ ‘ImageDataGenerator’ y se han aplicado las siguientes transformaciones:

- Rotación -> 40
- Traslación horizontal y vertical-> 0.2
- ‘Shear’ -> 0.2
- Zoom -> 0.2

- Giro horizontal

Tras la generación de datos sintéticos, el ‘dataset’ se dividió en conjunto de entrenamiento y conjunto de validación.



*Ilustración 13: Imagen original frente a imagen aumentada.*

### 3.3. Implementación CNN no pre entrenada

La idea de usar una CNN sin un paso previo de preentrenamiento era mostrar la capacidad autónoma que se podía alcanzar usando únicamente el ‘dataset’ encontrado. Por desgracia, el dataset encontrado no es suficiente por si solo y queda reflejado en los resultados que se comentan en el próximo apartado.

Para el diseño e implementación de las arquitecturas de CNN se utilizó ‘models’ y ‘layers’ de la librería ‘keras’. Las capas utilizadas en las arquitecturas son las capas de convolución, las capas de normalización, ‘MaxPooling2D’, ‘Dropout’ y al final capas densas. Se probaron 4 arquitecturas distintas cada una con una configuración de capas, pero se va a explicar únicamente la arquitectura con la que se consiguieron los mejores resultados. Se analizarán los resultados en detalle en el próximo apartado.

La mejor arquitectura se divide en 5 grupos de capas con la siguiente distribución:

- 2 subgrupos uno detrás de otro que consisten en una capa convolucional y una capa de normalización seguidas de una capa de 'MaxPooling2D' y 'Dropout' del 20%.
- Un subgrupo que contiene una capa de convolución, una de normalización y una de 'MaxPooling2D'.
- Otro subgrupo que consiste en una capa de convolución, una de normalización, una de 'MaxPooling2D' seguidas de una capa de 'Dropout' del 20%, una capa de 'GlobalAveragePooling2D' y otra capa de 'Dropout'.
- Por último, un grupo de capas formado por una capa densa seguida de una capa de normalización, una capa de 'Dropout' y una última capa densa con un número de neuronas igual al número de clases, en este caso 19.

El entrenamiento se realizó usando los datos separados para entrenamiento y los datos para validación durante 20 épocas. Como optimizador se eligió AdamW frente a Adam por la mejora que ofrece en la generalización al aplicar la decaída de los pesos directamente en la actualización de los parámetros y como función de pérdida se utilizó la entropía cruzada al tratarse de un problema de clasificación multiclase.

Tras las pruebas realizadas en CNN sin preentrenamiento, se realizó una prueba con ResNet50 (previamente explicado) para reconocimiento de imágenes pre entrenado con 'ImageNet' y ajustado al 'dataset' de la LSE. Se bloquearon los pesos del modelo ResNet50 para mantener el aprendizaje con 'ImageNet' y se añadieron unas últimas capas con capacidad de aprendizaje:

‘AveragePooling2D’, ‘Dropout’ y dos capas densas: una de 128 neuronas y otra de 19 (número de clases). De esta forma podemos transferir el conocimiento aprendido por ResNet50 sobre ‘ImageNet’ a nuestro problema.

### 3.4. Implementación ViT

De cara a usar un Vision Transformer (ViT), se eligió la arquitectura original propuesta en *An Image is Worth 16x16 Words* (Dosovitskiy et al., 2020) pre entrenada con el ‘dataset’ ‘ImageNet21k’. Para utilizar un ViT en nuestro ‘dataset’, se debe seguir un proceso en el cual primero se preprocesan las imágenes para adaptarlas a los requisitos de entrada del ViT, después se deciden los parámetros de entrenamiento y por último se entrena y evalúa bajo las métricas elegidas.

A la hora de preprocesar las imágenes se utiliza un extractor de características que se encarga de normalizar, reescalar y cambiar de tamaño. Como extractor de características se utilizó el ‘ViTFeatureExtractor’ de la librería ‘Transformers’ de ‘Hugging Face’. Se dispone de un archivo ‘JSON’ con los valores de configuración utilizados para la extracción de características.


A su vez, se definieron dos funciones, una encargada de convertir las imágenes y sus etiquetas en tensores para su utilización en el ViT y otra encargada de calcular las métricas, en este caso, la precisión.

Poniendo el foco en los parámetros definidos para entrenar el ViT, estos son los parámetros más importantes a la hora del entrenamiento:

- Tamaño de lote (Batch Size) -> 16
- Épocas -> 5


- Pasos por época -> 100
- Tasa de aprendizaje = 0.0002

Una vez se definieron tanto los argumentos como las métricas y se preprocesaron las imágenes, se entrenó el ViT.



```
1  trainer = Trainer(  
2      model=model,  
3      args=training_args,  
4      data_collator=collate_fn,  
5      compute_metrics=compute_metrics,  
6      train_dataset=prepared_train,  
7      eval_dataset=prepared_val,  
8      tokenizer=feature_extractor,  
9  )
```

*Ilustración 15: Trainer con los argumentos de entrenamiento.*



```
1  train_results = trainer.train()  
2  
3  trainer.save_model()  
4  trainer.log_metrics("train", train_results.metrics)  
5  trainer.save_metrics("train", train_results.metrics)
```

*Ilustración 14: Entrenamiento ViT.*



Tras el entrenamiento se evaluaron las métricas que se comentaran en el próximo apartado y se guardó el mejor modelo encontrado durante las épocas de entrenamiento.

### 3.5. Implementación YOLO

Antes de utilizar el modelo YOLO, al igual que con el ViT se deben preparar los datos para poder ingestar al modelo como datos de entrenamiento. Este proceso ha sido más tedioso que el realizado en ViT ya que se debe indicar en cada imagen con un rectángulo el signo a detectar para que el modelo sea capaz de encapsular la mano y detectar correctamente el signo.

Para realizar esta tarea de forma un poco más sencilla, se ha utilizado 'Roboflow', una web que facilita el etiquetado, entrenamiento y despliegue de soluciones de visión por computador. En este caso, solo se ha utilizado la parte de ayuda en etiquetado. El proceso es muy simple, se creó un proyecto y se cargó el 'dataset'. Al cargar el 'dataset', el propio 'Roboflow' ofrece opciones para realizar 'data augmentation' y separar en colecciones de entrenamiento, validación y prueba. Se indicó que se realizara un 'data augmentation' sencillo ya que ya se conocían las debilidades del 'dataset' expuestas tanto en el análisis exploratorio de datos como en la implementación de la CNN no pre entrenada. Por esta razón, era necesario aumentar el tamaño del 'dataset' para obtener mejor rendimiento. Después de aplicar 'data augmentation' se obtienen 4800 imágenes totales divididas en 4203 imágenes de entrenamiento, 400 imágenes de validación y 197 imágenes de prueba.

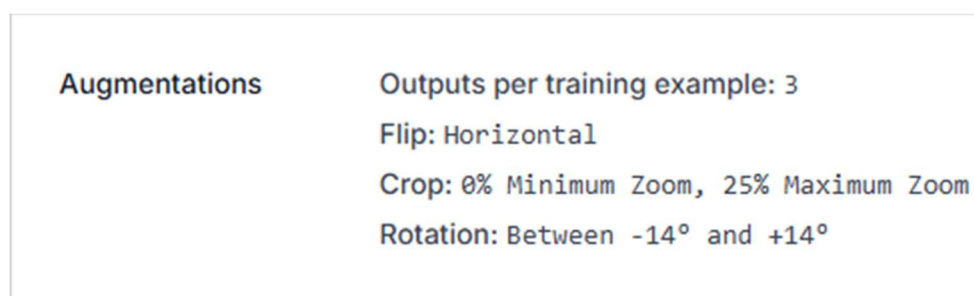


Ilustración 16: Data Augmentation en dataset para YOLO.

Tras cargar el ‘dataset’, aplicar ‘data augmentation’ y dividir en colecciones de entrenamiento, validación y prueba se realizó el tedioso proceso manual de etiquetado de manos para ajustar el ‘dataset’ a los requerimientos a la detección de objetos del modelo YOLO.

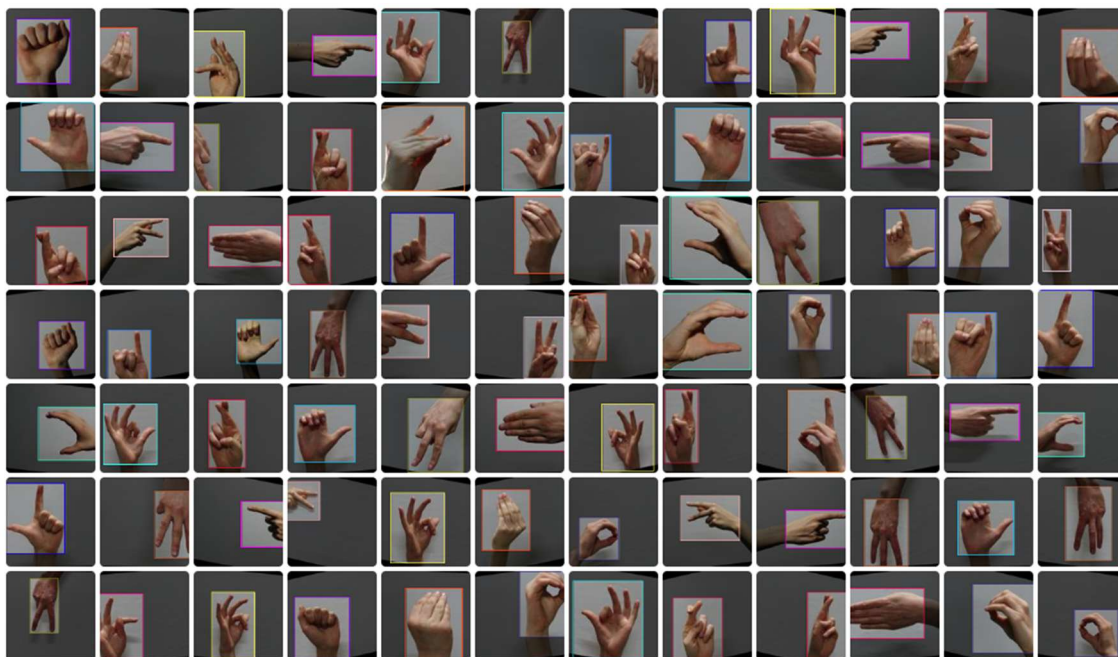


Ilustración 17: Etiquetado de manos para YOLO.

Una vez finalizado el etiquetado, se cargó el nuevo ‘dataset’ junto con el modelo pre entrenado ‘YOLO11n’ y se realizó un entrenamiento a lo largo de 20 épocas sobre el nuevo ‘dataset’. Para realizar la carga del modelo ‘YOLO11n’ se utilizó la librería ‘ultralytics’, librería que ofrece las distintas versiones de la arquitectura YOLO. Cabe destacar que YOLO nos ofrece la posibilidad de probar el rendimiento en tiempo real grabando a través de ‘webcam’ o también se pueden introducir videos al modelo.



Ilustración 18: Entrenamiento YOLO.

## Capítulo 5: Presentación de análisis, resultados y discusión

Este capítulo contiene los resultados obtenidos después del entrenamiento de los distintos modelos planteados en el capítulo anterior. La principal métrica comparativa es la precisión obtenida en validación y prueba.

El capítulo está dividido en dos apartados; los resultados obtenidos que exponen el rendimiento obtenido en los distintos modelos desde un tono descriptivo y la discusión y análisis de los resultados que contiene una interpretación de los resultados y una comparación entre modelos.

### 1. Resultados obtenidos

Este apartado tiene una división en subapartados por modelo, en cada subapartado se exponen los resultados obtenidos para dicho modelo y las pruebas realizadas.

#### 1.1. CNN no pre entrenada

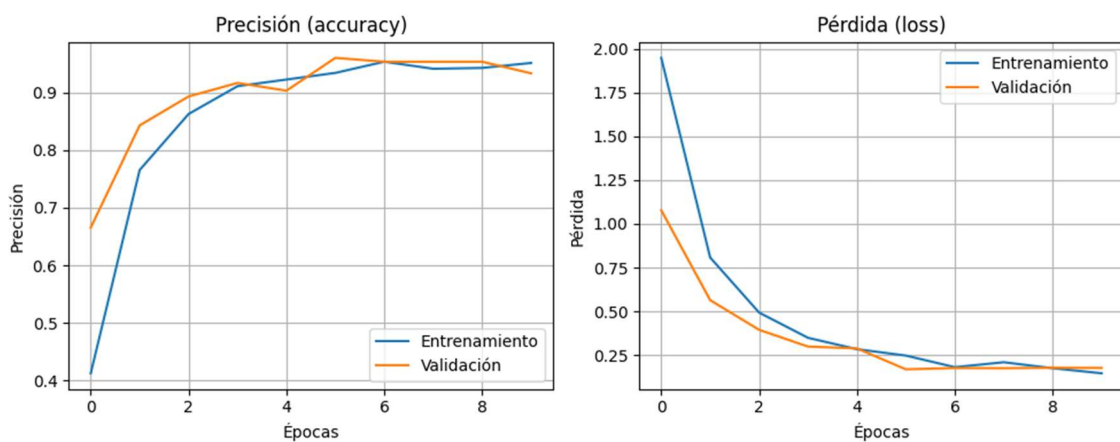
Se probaron cuatro arquitecturas de CNN que no recibieron preentrenamiento. Para guardar los resultados de cada arquitectura se creó un archivo 'JSON' que contiene la precisión en datos de prueba y la arquitectura junto con el nombre y una pequeña descripción. El código de las arquitecturas no se muestra en la memoria, pero está guardada en dicho archivo en el repositorio. En la primera CNN, las métricas obtenidas en entrenamiento eran tan bajas que no se tuvo en cuenta para evaluar en datos de prueba. La segunda CNN, la mejor de las cuatro, obtuvo una precisión de 0.19 en datos de prueba mientras que la tercera CNN obtuvo 0.05 y la cuarta 0.08. Son precisiones bastante bajas e insuficientes para el problema a

solucionar. Cabe destacar que, en uno de los entrenamientos sobre la arquitectura con mejores resultados, se obtuvo una precisión en prueba del 48%. Aun así, en las épocas de entrenamiento se observó que era anecdótico ya que la red sobre aprendía y entre épocas su rendimiento en validación bailaba entre el 15% y el 50% de acierto, destacando la sensibilidad de la red a la introducción de datos no vistos.

Nombre	Descripción	Accuracy (test)
CNN 1	Convolutional Neural Network para clasificación de imágenes.	N/A
CNN 2	Convolutional Neural Network para clasificación de imágenes.	0.19
CNN 3	Convolutional Neural Network para clasificación de imágenes.	0.05
CNN 4	Convolutional Neural Network para clasificación de imágenes.	0.08

*Ilustración 19: Resultados CNNs.*

Por el contrario, el ‘fine-tuning’ realizado en la arquitectura ‘ResNet50’ pre entrenada con ‘ImageNet’ presenta una buena adaptación a la tarea de reconocimiento de signos, postulándose como una solución válida.



*Ilustración 20: Resultados Entrenamiento y Validación ResNet50.*

## 1.2. ViT

En el caso del ‘Vision Transformer’ se obtuvieron las siguientes métricas en entrenamiento:

Step	Training Loss	Validation Loss	Accuracy
100	0.344500	0.322341	0.976589
200	0.096400	0.135441	0.983278
300	0.058200	0.073517	0.989967
400	0.046800	0.072267	0.993311

*Ilustración 21: Resultados entrenamiento ViT.*

Esta tabla muestra la pérdida en entrenamiento y validación y la precisión en la última época de entrenamiento. Después de evaluar el rendimiento de entrenamiento, se evaluó el modelo en el conjunto de prueba obteniendo los siguientes resultados:

```

***** eval metrics *****
epoch                        =          5.0
eval_accuracy                =         0.9871
eval_loss                    =         0.0817
eval_runtime                  =    0:00:04.10
eval_samples_per_second      =        75.216

eval_steps_per_second        =         9.493

```

### 1.3. YOLO

Tal y como se ha comentado en el apartado de implementación, se utilizó el ‘dataset’ modificado y adaptado para YOLO y se entrenó la versión ‘YOLO11n’ durante 20 épocas. Los resultados obtenidos en entrenamiento fueron los siguientes:

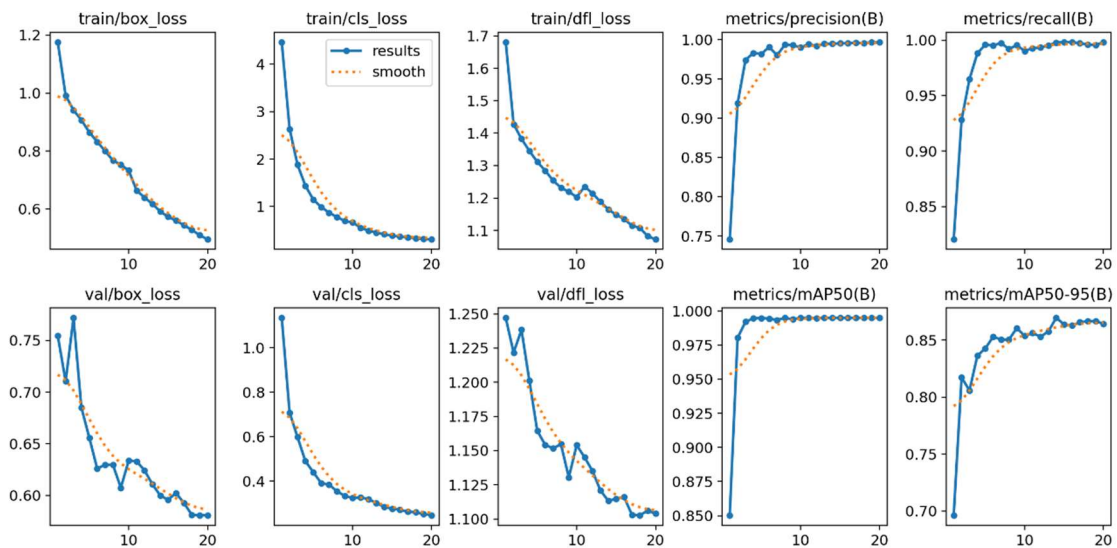


Ilustración 22: Métricas entrenamiento YOLO.

Se observa claramente un descenso progresivo en las funciones de pérdida y un aumento progresivo de la precisión y recall llegando a una precisión media promedio de 0.99. Otra forma de observar el rendimiento del modelo es la matriz de confusión que presenta visualmente los casos identificados correctamente en la diagonal de la matriz e indica que clase se ha asignado a los casos clasificados de forma incorrecta. En la matriz de confusión se observa claramente que el modelo clasifica prácticamente de forma perfecta las clases. En el apartado de análisis de resultados se describe porque esto puede ser engañoso a la hora de utilizar este modelo y otros.

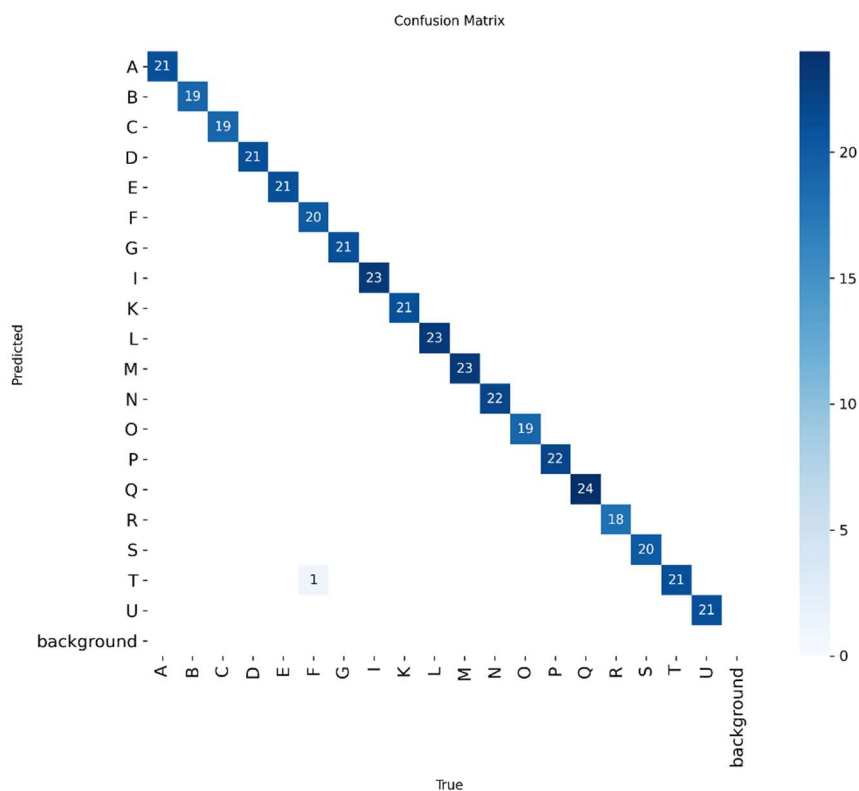


Ilustración 23: Matriz de confusión YOLO.

## 2. Discusión y análisis de los resultados

Este apartado contiene una interpretación exhaustiva y una comparación entre las arquitecturas probadas y los rendimientos obtenidos. Se repasan las métricas obtenidas y se explica el porqué.

En primer lugar, los rendimientos obtenidos en las arquitecturas CNN son pobres e insuficientes para plantear una solución de calidad para la detección de signos. Destaca la diferencia de rendimiento entre las arquitecturas no pre entrenadas y la arquitectura ‘ResNet50’ pre entrenada con ‘ImageNet’. La diferencia denota la importancia de la calidad del ‘dataset’ a la hora de entrenar una red neuronal convolucional. El ‘dataset’ utilizado es pequeño y



no tiene suficiente variabilidad en las imágenes por lo cual la calidad de las redes entrenadas se ve gravemente afectada. En cambio, la arquitectura ‘ResNet50’ pre entrenada en ‘ImageNet’ y reentrenada con el ‘dataset’ de la LSE es capaz de utilizar el aprendizaje del entrenamiento con ‘ImageNet’ y transferirlo a la tarea de reconocimiento de signos. La transferencia de conocimiento también resalta la importancia del conjunto de datos y la efectividad de la técnica a la hora de adaptar redes neuronales convolucionales generalizadas a tareas específicas.

En segundo lugar, el ViT ha obtenido unas métricas decentes para considerarlo como una solución al reconocimiento de signos de la LSE y postulándose también como una solución válida. Los resultados obtenidos son en gran parte gracias a que la arquitectura elegida estaba pre entrenada en el conjunto de datos ‘ImageNet21k’. La potencia de generalización obtenida en el preentrenamiento fue clave para obtener buenos resultados sobre el conjunto de datos de signos de la LSE.

Por último, los resultados obtenidos por la arquitectura YOLO también se explican debido al preentrenamiento con el conjunto de datos ‘COCO’, un ‘dataset’ a gran escala de imágenes etiquetadas para la detección de objetos. La facilidad que ofrece YOLO para probar el modelo en tiempo real ha sido clave a la hora de detectar debilidades en el modelo. La primera observación realizada al utilizar la detección en tiempo real es que el modelo no detecta bien prácticamente ningún signo lo cual puede ser sorprendente viendo la alta precisión obtenida tanto en entrenamiento como validación como prueba.

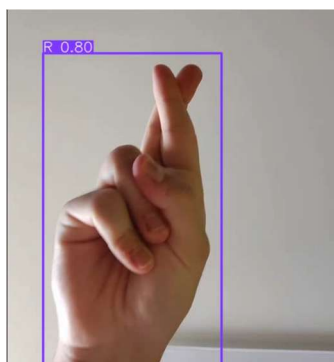
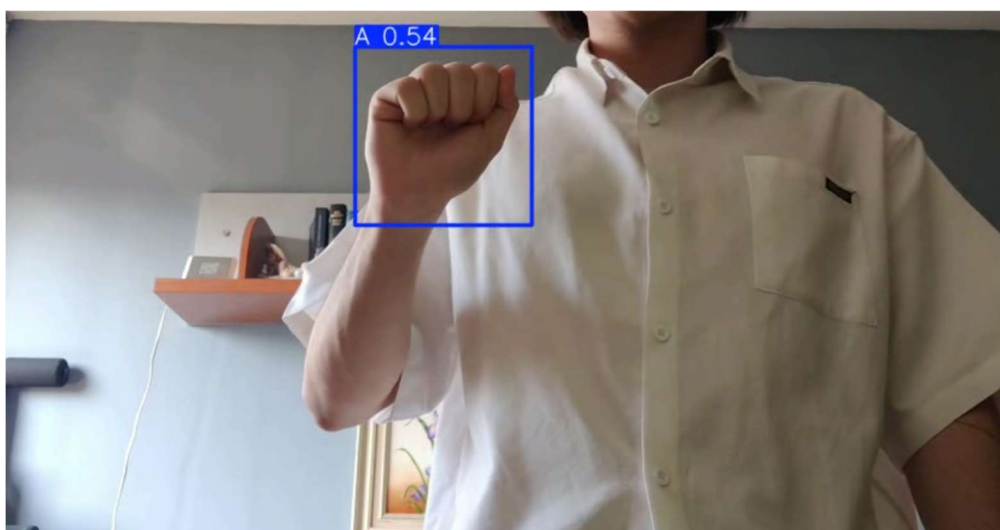


Ilustración 24: Detección  
Letra R.



Tras esta primera prueba en directo se analizó el conjunto de imágenes de signos de la LSE y contando que cada 'frame' del video en tiempo real es un nuevo dato que el modelo no ha visto en entrenamiento, resulta evidente que el modelo ha aprendido a detectar los signos de forma precisa sobre un fondo blanco ya que todas las imágenes del 'dataset' son sobre un fondo blanco. Esto resulta un problema ya que una mano acompaña a un brazo y este otro al hombro y así sucesivamente hasta formar una persona y el modelo es incapaz de detectar signos con una persona de fondo o fondos con tonalidades varias.



*Ilustración 26: Detecta la letra A de la LSE por la camisa blanca de fondo.*



*Ilustración 25: No detecta la Letra A de la LSE al separar la mano del fondo blanco de la camisa.*

En definitiva, este problema afecta tanto a YOLO como al ViT y las CNNs, las cuales si recibiesen nuevas imágenes sin fondo blanco como entrada verían su rendimiento reducido de forma considerable.

Teniendo en cuenta el problema existente con los modelos entrenados y el conjunto de datos de la LSE y con los resultados obtenidos, resulta evidente que las CNNs no pre entrenadas no son una opción válida para solucionar el problema de la detección de signos de la LSE y se quedan rezagadas frente al resto de modelos propuestos. Por tanto, no se van a utilizar en la comparativa general de modelos. En cambio, por los resultados obtenidos tanto por ‘ResNet50’ como por el ViT y YOLO, los tres son opciones válidas para usar como solución. Los tres modelos se ajustan a la tarea con alto rendimiento sin tener en cuenta los problemas con los fondos de las imágenes, pero existen diferencias a la hora de la aplicación de los modelos. ‘ResNet50’ y ViT son modelos enfocados al reconocimiento de imágenes y aunque ofrecen una vista y una posible solución al reconocimiento y detección de signos, su implementación en un sistema de detección de signos de la LSE en tiempo real requiere de pasos extra y no ofrecen la transparencia que ofrece YOLO al detectar un signo. Además, YOLO si está enfocado en la detección de objetos y permite el procesamiento en video en tiempo real lo cual abre puertas a un gran número de aplicaciones. YOLO también ofrece una vista más intuitiva de la detección del signo al especificar con una “caja” la mano en el momento de signar y asignar una etiqueta sobre esta “caja”, visualización que puede ayudar en aplicaciones de aprendizaje de la LSE.

Dado lo expuesto, YOLO ofrece un rendimiento igual o superior a arquitecturas como ‘ResNet50’ y ViT, resulta más intuitivo y ofrece más posibilidades para la detección de los signos de la Lengua de Signos Española, por tanto, es el modelo más adecuado como solución a la tarea expuesta.

## Capítulo 6: Conclusiones

Este capítulo consta de varios apartados, primero se exponen las limitaciones encontradas y los obstáculos observados. Después, se analizan las posibles ampliaciones y continuaciones del proyecto, mejoras, etc. Por último, se hace una reflexión sobre el trabajo realizado y lo aprendido. Este capítulo pone punto final al proyecto, cierra un ciclo para dar paso a los que vendrán.

Antes de pasar a las limitaciones, es necesario echar la vista atrás a los objetivos planteados al inicio del proyecto y valorar su estado y si se han cumplido. El primer objetivo planteado fue encontrar una base de datos o dataset de la LSE, en este caso, el objetivo está a medio completar porque sí que se han encontrado datos para realizar el proyecto, pero su calidad es baja, no es el dataset óptimo. Por el contrario, se han generado datos sintéticos para ampliar dicho dataset y por tanto el segundo objetivo específico se ha cumplido. Por otro lado, se ha realizado una investigación sobre modelos a la vanguardia y el estado del arte actual de la detección y reconocimiento de objetos y se ha diseñado una arquitectura propia de CNN, aunque no con buenos resultados. Asimismo, se ha implementado un Transformer visual para reconocimiento de imágenes. Otro de los objetivos cumplidos es aprovechar la arquitectura YOLO, especializada en detección de objetos, pre entrenar la arquitectura y ajustarla al problema con el dataset disponible, realizando así una tarea de ‘fine-tuning’. El último objetivo específico se planteó con la idea de realizar un estudio de las soluciones propuestas, el trabajo comparativo realizado entre CNN, ViT y YOLO marca el objetivo como cumplido.

Para finalizar con el objetivo general del proyecto, se ha realizado un estudio del estado del arte y se ha desarrollado un sistema automatizado de detección del alfabeto de la LSE, siendo este la arquitectura YOLO con ‘fine-tuning’ y se ha comparado con otras herramientas de reconocimiento de imágenes. Por tanto, se da por cumplido el objetivo general.

## 1. Limitaciones

En este apartado se exponen las distintas limitaciones encontradas a lo largo del desarrollo del proyecto, tanto las limitaciones técnicas como sociales y circunstanciales. Obstáculos que se hayan sorteado o no merecen comentario.

La primera limitación encontrada para el desarrollo del proyecto ha sido la disponibilidad de datos. El primer paso de la investigación fue la búsqueda de un ‘dataset’ de los signos de la lengua de signos española. Durante esta búsqueda se descubrieron las carencias en la disponibilidad de estos datos. El único ‘dataset’ encontrado con datos suficientes y de calidad fue el ‘dataset’ LSE\_eSaude\_UVIGO, una colección de videos de personas signantes y expertos en la lengua de signos española que mostraban como signar. Se contactó con los investigadores y autores de la construcción del ‘dataset’, en concreto con José Luis Alba-Castro para averiguar la disponibilidad del ‘dataset’ y comentar el contexto del proyecto. Tras este contacto, se conoció que el ‘dataset’ es privado y no es de distribución pública debido a la naturaleza propia (videos de personas) de los datos que están sujetos a la ley de protección de datos. A su vez, como consejo de José Luis, se llegó a la conclusión que la detección de todos los signos de la lengua de signos española era un objetivo demasiado ambicioso para este proyecto y algo que todavía no se había realizado en España. Debido a la propia limitación de datos y siguiendo el consejo de los investigadores de la Universidad de Vigo, se decidió reducir el alcance para cubrir únicamente el alfabeto de la Lengua de Signos Española.

A su vez, el ‘hardware’ ha sido otro aspecto limitante. El entrenamiento de modelos de estas capacidades necesita de equipos con gran capacidad de procesamiento. Este aspecto no ha evitado el cumplimiento, pero ha tenido un impacto negativo en los tiempos de entrenamiento de los modelos retrasando la finalización del desarrollo.

Por otro lado, no todo son limitaciones técnicas, el factor humano es una parte de la investigación. Nuestra situación, tanto económica como social afecta a nuestro trabajo de investigación. Aquí voy a exponer los obstáculos que he encontrado en la realización del proyecto desde el lado social. En primer lugar, el tiempo ha sido el principal limitante, compaginar el trabajo, el estudio, la realización del proyecto y conciliar con el resto de las responsabilidades del día a día ha supuesto un reto. En segundo lugar, y casi más importante, la situación familiar, cada uno de nosotros como individuo carga con una responsabilidad inherente que influye de manera significativa a la disponibilidad y el rendimiento. En el marco de la realización del TFM tanto como el marco del propio Máster, esta circunstancia puede suponer un obstáculo o reto adicional a gestionar.

## 2. Prospectiva

El desarrollo del traductor de la LSE ha permitido demostrar la viabilidad de aplicar técnicas de visión por computador y reconocimiento de objetos para facilitar la comunicación entre personas signantes y oyentes. No obstante, por la naturaleza del desarrollo ya que se trata de una primera fase de una solución tecnológica, el sistema presenta oportunidades de mejora y de escalabilidad. En este apartado se van a comentar las principales mejoras y como continuar el proyecto.

Una de las principales mejoras para la continuación del proyecto es el uso de equipos con mayor capacidad de procesamiento para acortar tiempos de entrenamiento y acceder a arquitecturas más potentes. El acceso a equipos con mayor capacidad permitiría utilizar arquitecturas más grandes y al mismo tiempo reducir el tiempo general de entrenamiento. A su vez, permitiría realizar más pruebas en menor tiempo.

Continuando con las pruebas, otro aspecto de mejora es el diseño de un conjunto de arquitecturas de prueba para encontrar la mejor solución y ampliar y continuar con las pruebas realizadas para abarcar un mayor

número de posibilidades y evaluar sus rendimientos. De nuevo, este aspecto ha estado limitado por el tiempo.

Por otro lado, una de las grandes mejoras para el desarrollo de la detección de signos es el ‘dataset’, el conjunto de datos de entrenamiento. Dada la escasa, por no decir nula, disponibilidad de datos ya sean fotos o videos de carácter público para uso en entrenamiento de modelos y considerando las debilidades del ‘dataset’ utilizado: datos insuficientes, poca variabilidad y falta de signos resulta evidente que una mejora muy importante es el desarrollo de un ‘dataset’ propio y amplio que contenga datos para todos los signos de la LSE. Cabe destacar que el desarrollo de un ‘dataset’ de este tamaño y características podría considerarse como un proyecto propio, como el realizado en la Universidad de Vigo (LSE\_UVIGO).

Continuando con el punto anterior, resulta evidente indicar que una continuación del proyecto es la ampliación a más signos, tanto estáticos como dinámicos. Se podría plantear un diseño de un desarrollo iterativo para añadir signos de forma constante al modelo y evaluar los cambios de rendimiento con el objetivo final de abarcar todos los signos de la Lengua de Signos Española. Esta ampliación va de la mano con el desarrollo del ‘dataset’ completo ya que daría acceso a la ampliación comentada.

### 3. Consideraciones finales

En definitiva, con este proyecto he conocido en profundidad las dificultades que enfrentan día a día las personas signantes y la importancia de investigaciones como la realizada. A su vez, he entendido con más detalle el proceso para ofrecer soluciones en tareas de reconocimiento y detección de signos y los obstáculos que acompañan estas tareas. En mi opinión, los resultados obtenidos iluminan un camino a seguir en futuras investigaciones, incluida la extensión de este mismo proyecto, mostrando el potencial de la solución y su escalabilidad. Además, los resultados obtenidos han resaltado las debilidades de los datos elegidos y los límites que han supuesto en la

solución propuesta acotando la calidad a ciertos escenarios. A lo largo del proyecto he aprendido acerca de los requisitos de los modelos para reconocimiento de imágenes, sus fortalezas y debilidades y la importancia del conjunto de datos para ofrecer una solución de calidad.

Asimismo, a lo largo del máster y este proyecto he asimilado la importancia de conocer la tarea propuesta, de entender el contexto que la rodea y con ello realizar una correcta elección de los datos y adaptación. Al final, la calidad de estos modelos depende en gran medida de la robustez de los datos: por muy bueno que sea el diseño, si el material de construcción es débil, el edificio está destinado a colapsar.

Finalmente, quisiera dar las gracias a aquellas personas que me han apoyado a lo largo de estos meses, quienes con su apoyo han hecho posible la finalización de este proyecto. A mi abuela Raquel, por ser una de las personas más valientes que he conocido, por su humanidad y cariño y por las fuerzas que me ha infundido con su ejemplo. A mi abuela Esther, por su cariño, su cercanía, su preocupación y por estar pendiente de mí. A mis amigos Fermín y Navas que no han dudado de mí en momentos en los que yo dudaba, quienes me han apoyado incansablemente desde que los conocí, quienes representan un sitio seguro para mí y quienes han marcado la diferencia estos últimos meses y a los que estoy eternamente agradecido. A mi amigo Almendra, que me ha ayudado a seguir adelante semana tras semana, que me ha apoyado y ha tenido fe en mí sin pedir nada a cambio y al cual considero un gran amigo. A mi amiga Noemí, que a pesar de la distancia me ha apoyado desde Alemania, me ha escuchado de forma paciente, atenta y sincera cuando lo necesitaba, me ha hecho reflexionar y empatizar, me ha sacado una sonrisa en momentos tristes y me ha hecho seguir adelante. A mi amiga Leire, quien me empujó a seguir en todo momento, quien me tuvo en consideración sin dudar, quien me hizo crecer como persona, en quien pude confiar y cuya sola presencia era suficiente para animarme, con quien he compartido buenos momentos y a quien admiro. Gracias.



## Bibliografía

- Dayas, I. A. (2020, 10 agosto). El Renacimiento y la invención de la lengua de signos. *Historia National Geographic*. [https://historia.nationalgeographic.com.es/a/renacimiento-y-invencion-lengua-signos\\_13360](https://historia.nationalgeographic.com.es/a/renacimiento-y-invencion-lengua-signos_13360)
- Wikipedia contributors. (2025, 17 julio). *Convolutional neural network*. Wikipedia. [https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)
- Convolución*. (s. f.). MATLAB & Simulink. <https://es.mathworks.com/discovery/convolution.html>
- Jain, A. (2024, 12 febrero). Understanding Convolutional Neural Networks (CNNs) with an Example on the MNIST Dataset. *Medium*. <https://medium.com/@abhishekjainindore24/understanding-convolutional-neural-networks-cnns-with-an-example-on-the-mnist-dataset-a64815843685>
- Emmert-Streib, Frank & Yang, Zhen & Feng, Han & Tripathi, Shailesh & Dehmer, Matthias. (2020). An Introductory Review of Deep Learning for Prediction Models With Big Data. *Frontiers in Artificial Intelligence*. 3. 4. 10.3389/frai.2020.00004.
- O'Shea, K., & Nash, R. (2015, 26 noviembre). *An Introduction to Convolutional Neural Networks*. arXiv.org. <https://arxiv.org/abs/1511.08458>



- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, 12 junio). *Attention is all you need*. arXiv.org. <https://arxiv.org/abs/1706.03762>
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020, 22 octubre). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv.org. <https://arxiv.org/abs/2010.11929>
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015, 8 junio). *You only look once: Unified, Real-Time Object Detection*. arXiv.org. <https://arxiv.org/abs/1506.02640>
- Goldblum, M., Souri, H., Ni, R., Shu, M., Prabhu, V., Somepalli, G., Chattopadhyay, P., Ibrahim, M., Bardes, A., Hoffman, J., Chellappa, R., Wilson, A. G., & Goldstein, T. (2023, 30 octubre). *Battle of the Backbones: A Large-Scale Comparison of Pretrained Models across Computer Vision Tasks*. arXiv.org. <https://arxiv.org/abs/2310.19909>
- Gan, Z., Li, L., Li, C., Wang, L., Liu, Z., & Gao, J. (2022, 17 octubre). *Vision-Language Pre-training: Basics, Recent Advances, and Future Trends*. arXiv.org. <https://arxiv.org/abs/2210.09263>
- Laura Docío-Fernández, José Luis Alba-Castro, Soledad Torres-Guijarro, Eduardo Rodríguez-Banga, Manuel Rey-Area, Ania Pérez-Pérez, Sonia Rico-Alonso, Carmen García-Mateo. *LSE\_UVIGO: A Multi-source Database for Spanish Sign Language Recognition*. [http://www.lrec-conf.org/proceedings/lrec2020/workshops/SIGN2020/pdf/2020.signlang\\_lrec-1.8.pdf](http://www.lrec-conf.org/proceedings/lrec2020/workshops/SIGN2020/pdf/2020.signlang_lrec-1.8.pdf)

He, K., Zhang, X., Ren, S., & Sun, J. (2015, 10 diciembre). *Deep Residual Learning for Image Recognition*. arXiv.org. <https://arxiv.org/abs/1512.03385>

Wikipedia contributors. (2025, julio 19). *Data augmentation*. Wikipedia. [https://en.wikipedia.org/wiki/Data\\_augmentation](https://en.wikipedia.org/wiki/Data_augmentation)

Wikipedia contributors. (2025c, agosto 11). *Dynamic time warping*. Wikipedia. [https://en.wikipedia.org/wiki/Dynamic\\_time\\_warping](https://en.wikipedia.org/wiki/Dynamic_time_warping)

Hisham, B., & Hamouda, A. (2017). *Arabic Static and Dynamic Gestures Recognition Using Leap Motion*. *Journal Of Computer Science*, 13(8), 337-354. <https://doi.org/10.3844/jcssp.2017.337.354>

David, G. B. J., & Javier, P. H. W. (2015). *Sistema traductor de la lengua de señas colombiana a texto basado en FPGA*. Biblat. <https://biblat.unam.mx/es/revista/dyna-medellin/articulo/sistema-traductor-de-la-lengua-de-senas-colombiana-a-texto-basado-en-fpga>

Yi Li (2012, 1 junio). *Hand gesture recognition using Kinect*. <https://ieeexplore.ieee.org/document/6269439>

Firas Abdulrazzaq Raheem & Hadeer Raheem (2019, diciembre). *American Sign Language Recognition Using Sensory Glove and Neural Network*. [https://www.researchgate.net/publication/340721626\\_American\\_Sign\\_Language\\_Recognition\\_Using\\_Sensory\\_Glove\\_and\\_Neural\\_Network](https://www.researchgate.net/publication/340721626_American_Sign_Language_Recognition_Using_Sensory_Glove_and_Neural_Network)



# Anexos

Repositorio de código GITHUB: <https://github.com/Tappedz/TFM-Traductor-LSE>