UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

Desarrollo de un traductor de lenguaje de signos mediante inteligencia artificial

Trabajo Fin de Máster

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

AUTOR/A: Aquerreta Montoro, Ander

Tutor/a: Julian Inglada, Vicente Javier

Cotutor/a: Marco Detchart, Cédric

Cotutor/a externo: Rincón Arango, Jaime Andrés

CURSO ACADÉMICO: 2023/2024

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Universitat Politècnica de València

Departamento de Sistemas Informáticos y Computación

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Trabajo Fin de Máster

Desarrollo de un traductor de lenguaje de signos mediante inteligencia artificial

Autor(a): Ander Aquerreta Montoro
Director(a): Julian Inglada, Vicente Javier
Cotutor(a): Marco Detchart, Cédric

Valencia, Septiembre - 2024

Este Trabajo Fin de Máster se ha depositado en el Departamento de Sistemas Informàticos y Computación de la Universitat Politècnica de València para su defensa.

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Título: Desarrollo de un traductor de lenguaje de signos mediante inteligencia artificial

Septiembre - 2024

Autor(a):	Ander Aquerreta Montoro
Director(a):	Julian Inglada, Vicente Javier Departamento de Sistemas Informáticos y Computación Universitat Politècnica de València
Co-director(a):	Marco Detchart, Cédric Departamento de Sistemas Informáticos y Computación Universitat Politècnica de València
Director(a) experimental:	RINCON ARANGO, JAIME ANDRES Departamento de Sistemas Informáticos y Computación Universitat Politècnica de València

Resum

El llenguatge de signes és la llengua natural de les persones sordes. És una llengua no parlada, de caràcter visual i gestual amb una gramàtica pròpia que permet expressar-se com qualsevol llengua parlada. Al contrari del que es podria pensar, existix una llengua de signes per cada idioma parlat, amb grans diferències entre ells. Les principals investigacions sobre reconeixement i traducció de llengua de signes se centren en la llengua de signes anglesa, alemanya i xinesa. Estes tracten, sobretot, de classificació de signes *mediantes xarxes neuronals convolucionals. Este treball final de màster té com a objectiu el desenrotllament d'un prototip de traductor de llenguatge de signes recolzat en intel·ligència artificial, amb la finalitat de facilitar la comunicació per a persones amb discapacitat auditiva. A través de la integració de tecnologies avançades i aprenentatge profund, es busca crear un model capaç d'interpretar gestos i expressions del llenguatge de signes amb la major precisió possible.

Resumen

El lenguaje de signos es la lengua natural de las personas sordas. Es una lengua no hablada, de carácter visual y gestual con una gramática propia que permite expresarse como cualquier lengua hablada. Al contrario de lo que se podría pensar, existe una lengua de signos por cada idioma hablado, con grandes diferencias entre ellos. Las principales investigaciones sobre reconocimiento y traducción de lengua de signos se centran en la lengua de signos inglesa, alemana y china. Estas tratan, sobre todo, de clasificación de signos mediantes redes neuronales convolucionales. Este trabajo final de máster tiene como objetivo el desarrollo de un prototipo de traductor de lenguaje de signos apoyado en inteligencia artificial, con el fin de facilitar la comunicación para personas con discapacidad auditiva. A través de la integración de tecnologías avanzadas y aprendizaje profundo, se busca crear un modelo capaz de interpretar gestos y expresiones del lenguaje de signos con la mayor precisión posible.

Abstract

Sign language is the natural language of deaf people. It is a non-spoken, visual and gestural language with its own grammar that allows it to express itself like any spoken language. Contrary to what one might think, there is one sign language for each spoken language, with great differences between them. The main research on sign language recognition and translation focuses on English, German and Chinese sign languages. These mainly deal with sign classification using convolutional neural networks. This master's thesis aims to develop a prototype sign language translator supported by artificial intelligence, in order to facilitate communication for people with hearing impairment. Through the integration of advanced technologies and deep learning, the aim is to create a model capable of interpreting gestures and expressions of sign language as accurately as possible.

Agradecimientos

En primer lugar, me gustaría agradecer a mis tutores por el apoyo que me han dado durante la beca de colaboración en el VRAIN, Vicente Javier Julian Inglada y Cédric Marco Detchart.

Por otro lado, quiero agradecer a ValgrAI – Valencian Graduate School and Research Network for Artificial Intelligence y a la Generalitat Valenciana por la ayuda proporcionada para la realización del máster.

Por último, y mas importante, me gustaría agradecer a toda mi familia y mi novia, por el apoyo durante mi estancia en Valencia y por todo en general.

Tabla de contenidos

Capítulo 1

Introducción

En España aproximadamente 21 de cada mil personas son sordas o con algún tipo de discapacidad auditiva (es decir, un 2,13 % de la población total) según los datos recogidos por la encuesta del Instituto Nacional de Estadística (INE) en su estudio "EDAD 2008"[1]. Esta cifra pone de manifiesto la necesidad de desarrollar tecnologías inclusivas que faciliten la comunicación entre las personas sordas y el resto de la sociedad. El lenguaje de signos es la principal forma de comunicación para muchas personas con discapacidad auditiva, pero su adopción y comprensión por parte de la población general sigue siendo limitada. Esto plantea importantes barreras para la integración plena de las personas sordas en diversos ámbitos de la vida cotidiana, como la educación, el trabajo y los servicios públicos. Por esta razón la construcción de un traductor neuronal de lengua de signos podría facilitar la vida de las personas con esta discapacidad.

1.1. Introducción a la traducción automática

La traducción automática de lengua de signos a lengua hablada es un campo de investigación interdisciplinar que une la visión por computador y la traducción automática, dos de los campos más importantes del aprendizaje profundo. Sin embargo, para poder abarcar este problema primero hay que entender que es la lengua de signos y como funciona.

1.2. Lengua de signos

Las lenguas de signos tienen orígenes inciertos, aunque se teoriza que surgieron antes que las lenguas orales, posiblemente en la prehistoria como una forma primaria de comunicación humana. La lengua de signos surge de manera natural cuando dos personas sordas interactúan, lo que sugiere que ha existido desde que personas sordas han podido comunicarse entre sí. En España, las primeras investigaciones lingüísticas sobre la Lengua de Signos Española (LSE) comenzaron en 1992, y en 2007, el parlamento reconoció oficialmente la LSE y la Lengua de Signos Catalana, garantizando su uso en educación, servicios, cultura y medios de comunicación[2].

En el marco más teórico o lingüístico de la lengua de signos hay varios estudios que remarcan la importancia de la existencia de un sistema de comunicación para las

personas sordas[3].

A lo largo de los años, la idea de una lengua de signos universal ha sido un tema recurrente en la investigación y en las comunidades sordas a nivel mundial. La diversidad de lenguas de signos, cada una con sus características lingüísticas y culturales propias, refleja la riqueza y la variedad de las comunidades sordas alrededor del mundo. Sin embargo, esta diversidad también plantea desafíos en términos de comunicación entre personas sordas de diferentes países y culturas. Estudios como el de Zeshan[4] han explorado la viabilidad de una lengua de signos universal, proponiendo que una lengua estándar podría facilitar la comunicación internacional y promover una mayor cohesión entre las comunidades sordas globales. La Lengua de Signos Internacional (LSI), desarrollada por la Federación Mundial de Sordos, es un ejemplo de este esfuerzo por crear un sistema de signos que pueda ser comprendido por personas sordas de diferentes orígenes. Desde un punto de vista práctico, una lengua de signos universal podría tener un impacto significativo en la accesibilidad y la inclusión de personas sordas en ámbitos globales.

No obstante, es importante reconocer que la implementación de una lengua de signos universal también enfrenta retos significativos. Hay estudios[5] que indican que cada lengua de signos está profundamente arraigada en la cultura y la identidad de su comunidad, lo que hace que la imposición de una lengua estándar pueda ser percibida como una amenaza a la diversidad cultural y lingüística. Además, la complejidad y la riqueza de las lenguas de signos individuales podrían no ser completamente capturadas en un sistema universal simplificado.

1.3. Lengua de signos española (LSE)

En nuestro trabajo vamos a centrarnos en la lengua de signos española (LSE). La LSE es un sistema de comunicación visual que utiliza una combinación de componentes manuales, como la configuración y orientación de las manos, el movimiento y la posición en el espacio, junto con componentes no manuales, como las expresiones faciales, la postura corporal y la dirección de la mirada. Estos elementos interactúan para formar un lenguaje complejo y altamente expresivo. Además, la LSE emplea el espacio de señas de manera estructurada, utilizando referencias espaciales que permiten establecer relaciones gramaticales y espaciales, como la posición relativa de sujetos y objetos, o la indicación temporal. En términos de morfología, la LSE es rica en la incorporación de significado dentro de los propios signos, permitiendo la modificación de estos para reflejar aspectos como número, tiempo o intensidad. La sintaxis en LSE es flexible, pero sigue patrones que difieren del castellano, como el orden Sujeto-Objeto-Verbo (SOV). Asimismo, la pragmática y el contexto juegan un papel crucial en la interpretación de los signos, influyendo en su significado en función de la situación comunicativa. Finalmente, la iconicidad en la LSE, donde muchos signos tienen una relación visual directa con su referente, junto con signos más arbitrarios, demuestra la riqueza y diversidad de esta lengua visual, que permite a las personas sordas expresar de manera precisa y matizada una amplia gama de significados[6].

1.4. Reto y complicaciones

A diferencia de lo que ocurre con otras lenguas de signos más populares y habladas, como la Lengua de Signos Americana (ASL) o la Lengua de Signos China (CSL), la Lengua de Signos Española (LSE) carece de datasets extensivos que puedan ser utilizados para el entrenamiento de modelos de aprendizaje automático. Esta carencia de recursos dificulta el desarrollo de tecnologías avanzadas, como sistemas automáticos de reconocimiento y traducción de la LSE, ya que los datasets son esenciales para entrenar algoritmos que puedan interpretar de manera precisa los signos y su correspondiente traducción al lenguaje oral.

Sin embargo, a pesar de la falta de datasets, la LSE cuenta con varios diccionarios que han sido desarrollados y utilizados tanto en la educación como en la difusión de esta lengua. Estos diccionarios son valiosas herramientas que registran una gran cantidad de signos, junto con sus significados y, en algunos casos, explicaciones visuales o videos que ilustran la ejecución de cada signo. Estos recursos, aunque no equivalentes a datasets estructurados, representan un punto de partida importante para la creación de corpus y bases de datos que puedan ser utilizados en la investigación y el desarrollo de modelos de traducción automática.

1.5. Objetivos

El objetivo principal de nuestro trabajo es desarrollar un traductor neuronal básico para la Lengua de Signos Española (LSE). Este traductor se basará en modelos de aprendizaje profundo que permitirán la traducción automática de signos en LSE al lenguaje escrito en castellano, contribuyendo así a la accesibilidad y comunicación efectiva con las personas sordas.

Para alcanzar este objetivo general, hemos establecido los siguientes objetivos específicos:

- **Recolección y Preparación de Datos:** Identificar y compilar los recursos disponibles, como diccionarios de LSE, y convertirlos en un dataset adecuado para el entrenamiento de modelos neuronales. Este proceso incluirá la digitalización y etiquetado de signos, asegurando que los datos estén estructurados y normalizados para su uso en modelos de aprendizaje automático.
- **Desarrollo del Modelo Neuronal:** Diseñar e implementar un modelo de traducción automática basado en redes neuronales que sea capaz de interpretar los signos en LSE y generar la correspondiente traducción al castellano. Este modelo se construirá utilizando técnicas avanzadas de procesamiento del lenguaje natural y visión por computadora.
- **Entrenamiento y Optimización del Modelo:** Entrenar el modelo neuronal utilizando el dataset recopilado, aplicando técnicas de optimización para mejorar la precisión y eficiencia del traductor. Esto incluirá la implementación de estrategias para manejar las limitaciones de los datos y ajustar los parámetros del modelo para obtener resultados óptimos.

Por falta de tiempo y de recursos solo vamos a poder realizar un reconocimiento de lengua de signos en vez de una traducción completa. Esto se debe a la carencia de un dataset etiquetado. Como solo tenemos diccionarios con conceptos (glosas), solo

podemos hacer una traducción de lengua de signos a glosas (SLR). Si tuviésemos un dataset bien estructurado y etiquetado podríamos hacer el proceso completo. Además, podríamos proponer otros objetivos específicos como la validación del traductor neuronal y comparación con otros datasets.

Capítulo 2

Estado del arte

2.1. Traducción automática de lengua de signos a texto

La traducción automática de lengua de signos a lengua hablada es uno de los campos más emergentes de los últimos años. Aunque es cierto que la mayoría de publicaciones se han publicado en el último lustro, la investigación en este campo lleva siendo un tema a tener en cuenta desde hace ya varios años.

La traducción automática de la lengua de signos a texto hablado ha experimentado un desarrollo significativo a lo largo de los años. Al principio, los esfuerzos se centraron en la creación de sistemas básicos de reconocimiento de signos utilizando técnicas de visión por computador. Con el avance de las tecnologías de aprendizaje automático y redes neuronales, los sistemas actuales han mejorado notablemente en precisión y eficiencia. Este progreso ha sido fundamental para mejorar la accesibilidad y la comunicación para las personas sordas y con problemas de audición.

Uno de los principales desafíos en la traducción de la lengua de signos es la alta variabilidad en la realización de los signos. Factores como las diferencias individuales en la ejecución de los signos, las variaciones regionales en las lenguas de signos y la influencia de las expresiones faciales y corporales complican la tarea de desarrollar modelos precisos. Además, la calidad y las condiciones de iluminación en los vídeos también pueden afectar la precisión del reconocimiento.

La traducción de lengua de signos a lengua hablada trata de fusionar dos campos de investigación, que son la traducción automática texto a texto[7] (en este caso la neural machine translation) y la visión por computador[8].

La traducción automática es una tarea sequence-to-sequence[9]. Esto significa que, recibe de entrada una secuencia de tokens de un lenguaje y el modelo genera otra secuencia de tokens. Los modelos más actuales de traducción automática de lengua de signos están basados en una estructura encoder-decoder, donde el encoder recibe los vídeos de lengua de signos y genera una secuencia de valores (embbedings) y el decoder recibe esta secuencia y genera la salida. Estas estructuras solían estar basadas en las redes neuronales recurrentes[10]. Las redes recurrentes más utilizadas eran las Long Short-Term Memory Networks (LSTM)[11] y las Gated Recurrent Units (GRUs)[12]. Sin embargo, las RNN tienen varias limitaciones. Una de las principales son el vanishing gradient y el exploding gradient[13]. Aquí se vio que, la principal

limitación de las RNN, son el problema que tienen para aprender dependencias temporales a largo plazo.

En los últimos años, la arquitectura más utilizada en esta línea de investigación es la arquitectura Transformer. La arquitectura Transformer fue pensada originalmente para el procesamiento del lenguaje natural. Fue introducida en el artículo .Attention is All You Need"por Vaswani et al. en 2017[14]. Los principales objetivos de esta arquitectura eran mejorar la eficiencia y el rendimiento en tareas de traducción automática, superando las limitaciones de las redes neuronales recurrentes (RNNs) y las redes neuronales convolucionales (CNNs), que eran las más comunes en ese momento.

En la traducción automática de texto a texto existe una base de datos de embeddings para cada palabra. Los embeddings son vectores que contienen la información de la palabra, que han sido entrenados con grandes corpus de entrenamiento. Alguno de los principales modelos de generación de embeddings son Word2Vec[15] y Global Vectors for Word Representation (GloVe)[16].

Estos métodos emplean redes neuronales artificiales para convertir palabras en vectores numéricos de dimensión fija que capturan contextos semánticos. Por ejemplo, Word2Vec ofrece dos arquitecturas principales: Continuous Bag of Words (CBOW) y Skip-Gram. CBOW[17] predice una palabra objetivo usando un contexto de palabras cercanas, mientras que Skip-Gram[18] hace lo contrario, utilizando una palabra objetivo para predecir palabras del contexto. Durante el entrenamiento, las palabras se representan inicialmente como vectores aleatorios que se ajustan a través del entrenamiento de los pesos de la red neuronal. A medida que el modelo se entrena con grandes corpus de texto, los vectores de palabras similares en contexto convergen en el espacio vectorial, capturando relaciones semánticas y sintácticas entre ellas. Este proceso permite que las palabras con significados similares tengan representaciones vectoriales cercanas, facilitando su uso en diversas aplicaciones de procesamiento del lenguaje natural.

En la figura2.1 podemos observar como es traducción entre texto escrito (en el ejemplo de inglés a español). Tenemos nuestra entrada, que son textos, los cuales tienen asignados un embedding ya entrenado, estos embeddings pasan por un encoder donde se aplican mecanismos de atención, positional encoding... Por último, la salida del encoder la recibe el decoder que produce nuestra traducción.

En lengua de signos no se utiliza texto, sino que se utiliza vídeo. Por tanto, ya no podemos hacer una base de datos de embeddings para los vídeos, debido a que cada vídeo puede diferir notoriamente de otros siendo el mismo signo. En los vídeos hay factores que intervienen como la calidad, la iluminación, los fps... Se ha estudiado que estos factores son diferenciales a la hora de conseguir resultados con vídeos para reconocimiento de acciones[19] y de caras[20].

Mientras que los textos de lenguas habladas pueden ser divididos en palabras, subpalabras, caracteres o tokens, que pueden ser representados como vectores, para la lengua de signos es mucho más complicado. No existe ninguna manera explícita de representar cada signo, ya que cada signo varía mucho según la persona que lo realiza y factores del vídeo como la iluminación, el fondo de la imagen... En la figura 2.2 se observa como es la arquitectura. Nuestro input es un vídeo (secuencia de frames) que, tras pasar por un encoder, genera un embedding que representa toda la información del vídeo. Este embedding es el que va a entrar al decoder para realizar la

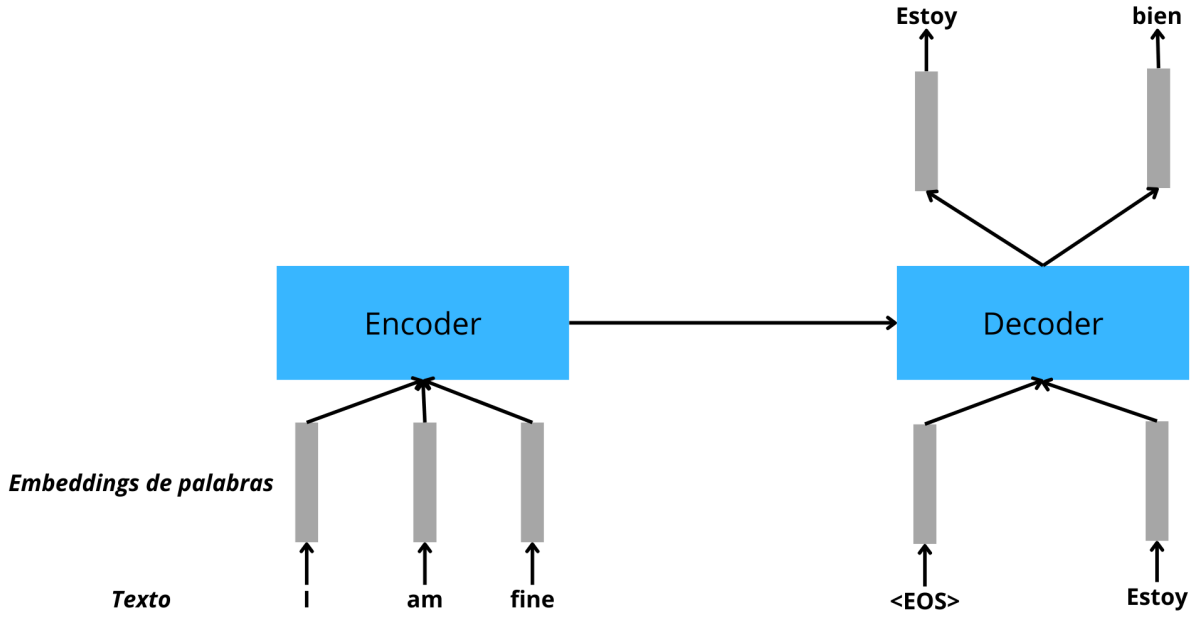

Figura 2.1: Traducción de texto a texto

traducción.

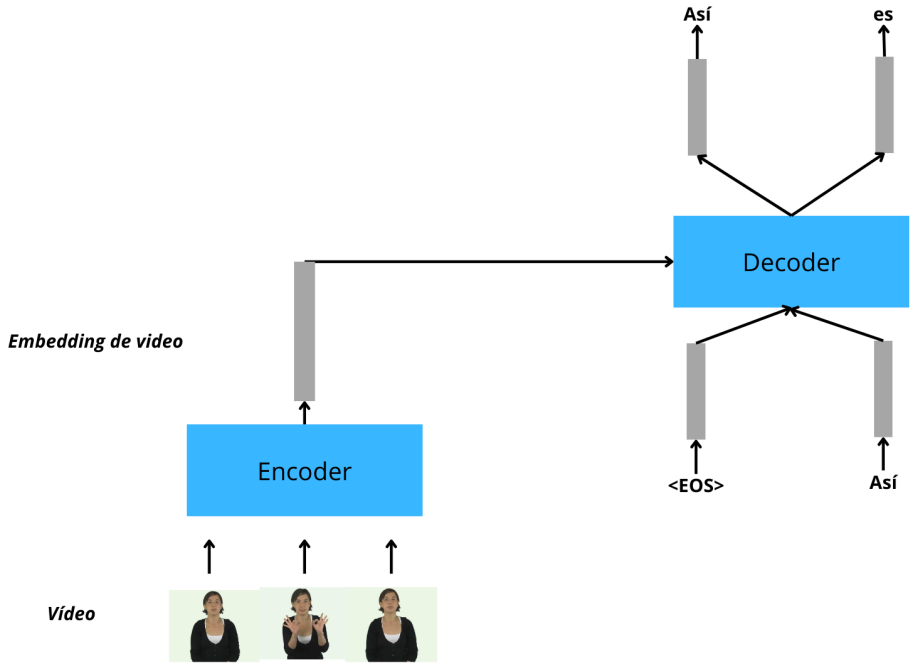

Figura 2.2: Traducción de lengua de signos a texto

Siguiendo las arquitecturas encoder-decoder para traducción neuronal el primer punto que hay que tener en cuenta es como procesar los vídeos. El procedimiento

habitual es dividir los vídeos en frames. Sin embargo, el principal problema del procesamiento con videos es la alta complejidad computacional, ya que ahora tenemos que procesar T imágenes dadas secuencialmente, donde T es el número de frames. Varios estudios[21] han determinado que tener en cuenta todos los frames de los vídeos es muy ineficiente computacionalmente y, en gran parte de las ocasiones, innecesario.

En general, construir un buen encoder para los vídeos es un reto por sí mismo. Centrándonos en la lengua de signos, tenemos que lograr un encoder que sea capaz de captar y extraer la información relevante de la imagen. Este proceso se conoce como Sign Lenguage Recognition[22] (SLR) y se realiza previo a la traducción.

2.2. Keypoints

Los keypoints (puntos clave) son ubicaciones específicas en una imagen que representan características o puntos de interés. En el contexto de la visión por computador y el análisis de imágenes, los keypoints se utilizan para describir partes cruciales de una estructura, como articulaciones del cuerpo, características faciales, o puntos distintivos en objetos.

Por ejemplo, en el reconocimiento facial, los keypoints pueden ser los puntos donde se encuentran los ojos, la nariz y la boca. En el análisis de movimientos humanos, los keypoints suelen ser las articulaciones como los hombros, codos, muñecas, caderas, rodillas y tobillos. Existen diversos estudios que utilizan los keypoints para representar a las personas[23][24]. La lengua de signos utiliza movimientos de las manos, expresiones faciales y posturas corporales para comunicar significados. La identificación precisa de estos movimientos y gestos es crucial para el reconocimiento automático de la lengua de signos. Por lo tanto, puede resultar útil para la lengua de signos los keypoints en las manos, en el torso y en la cara, que son las partes del cuerpo que intervienen en el lenguaje.

Los puntos se eligen de manera estratégica. Los keypoints típicamente representan partes del cuerpo donde el movimiento es más articulado o donde la estructura es clave para entender la postura o la acción humana. La selección de keypoints en humanos generalmente sigue la anatomía y la biomecánica del cuerpo, enfocándose en lugares que proporcionan información crucial sobre la postura y el movimiento. Por ejemplo, normalmente para las manos se utilizan 21 keypoints. Por cada dedo tenemos 3 falanges (distal, medial y proximal), los puntos serán los extremos de cada falange. Es decir, por cada dedo tendremos como puntos la uña, la unión entre falange distal - medial, medial - proximal, y el nudillo. Por tanto, tenemos 20 puntos en los dedos, añadiendo un punto adicional a la muñeca de la mano, tal y como se observa en la figura 2.3. Para la cara de las personas se suelen utilizar 8 keypoints para los ojos, 1 para la nariz y dos para los labios. Y, por último, para lengua de signos puede ser útil también los keypoints de los hombros, codos y la cadera, como se observa en la figura 2.4.

Para poder aprender estos keypoints debemos entrenar un modelo con datos etiquetados[25] con las coordenadas de los puntos. Luego, estas imágenes reciben un preprocesado para pasar de puntos de coordenadas a mapas de calor[26]. Los mapas de calor transforman las coordenadas discretas de los keypoints en distribuciones continuas de probabilidad sobre la imagen. Esto ayuda a que el modelo sea más robusto a peque-

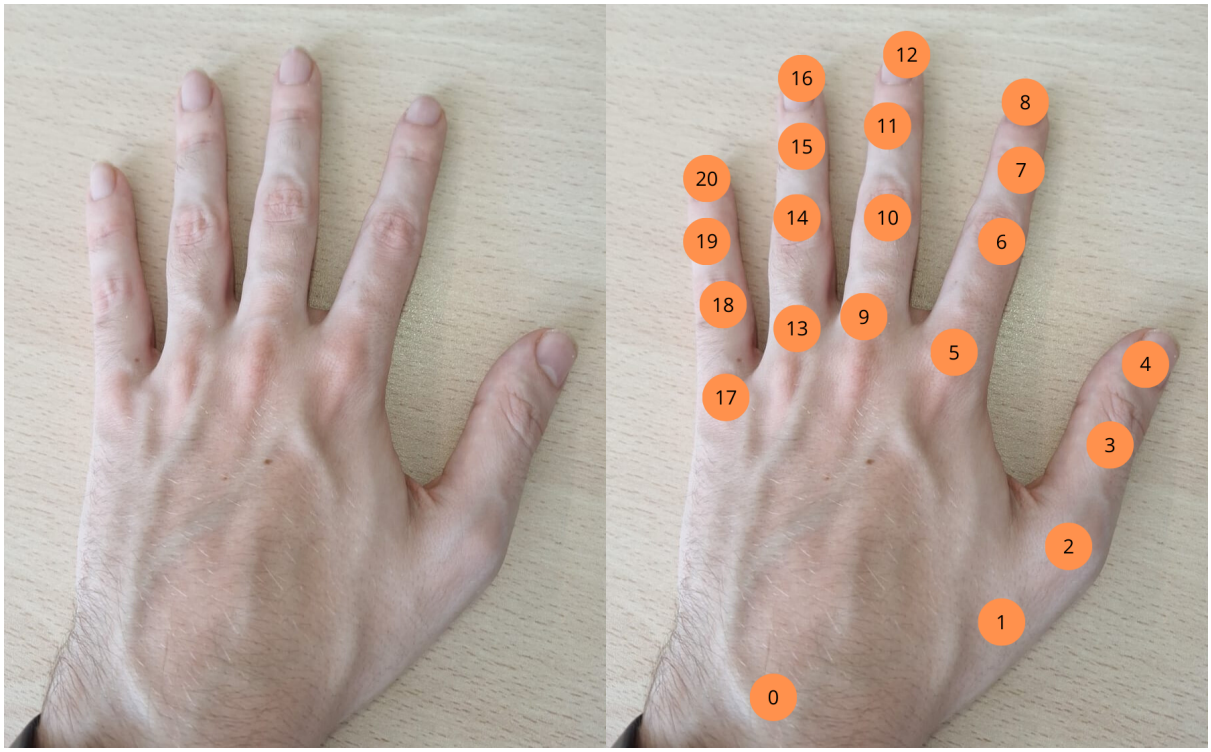

Figura 2.3: Keypoints en las manos

ñas variaciones y errores en la localización exacta de los keypoints. La representación en mapas de calor permite que el modelo aprenda a predecir la probabilidad de la posición de un keypoint en áreas cercanas al punto verdadero, en lugar de depender estrictamente de una posición precisa. Al no tener un punto específico no podemos utilizar métricas de distancias como el MSE2.1 o el MAE2.2, sino que se suele utilizar el IoU[27]. El IoU mide la superposición entre dos áreas: la predicción realizada por el modelo y el área verdadera o el objeto real. Se calcula como la relación entre la intersección y la unión de estas dos áreas2.3.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.1}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.2}$$

$$\text{IoU} = \frac{\text{Área Intersección}}{\text{Área Unión}} \tag{2.3}$$

Una de las arquitecturas más populares para segmentación de imágenes y segmentación de keypoints es la U-net. La red fue desarrollada inicialmente en el contexto del concurso ISBI Challenge 2015[28] para la segmentación de estructuras neuronales en microscopía electrónica[29]. Las técnicas de segmentación tradicionales no eran lo suficientemente efectivas para segmentar imágenes biomédicas, donde la precisión y la eficacia es esencial. La red fue diseñada para mejorar los modelos tradicionales

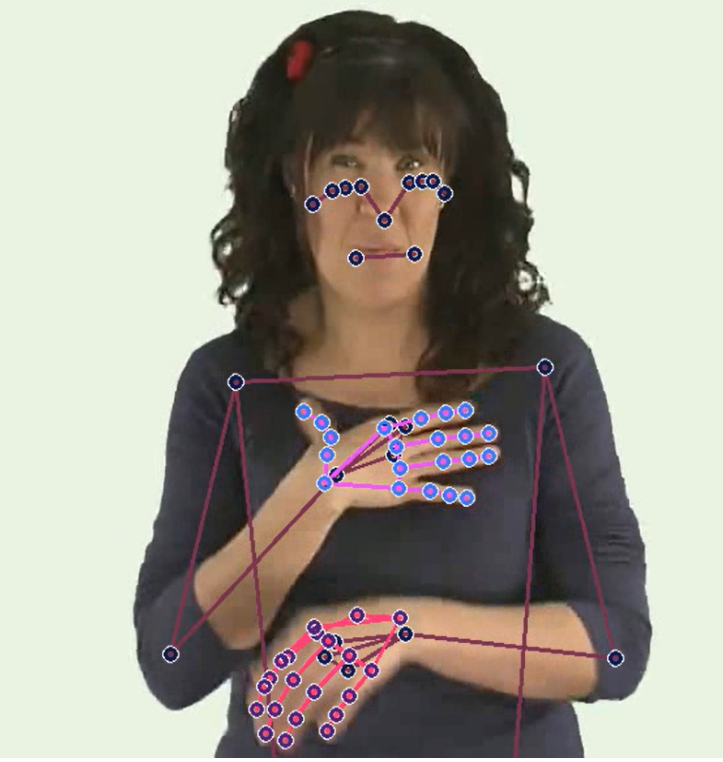

Figura 2.4: Keypoints en video de lengua de signos

mediante el uso de una estructura en forma de U, que se observa en la figura2.5. La U-Net tiene una estructura simétrica que consta de dos partes principales: contracción (parte izquierda) y expansión (parte derecha). En la fase de contracción, la imagen de entrada pasa a través de una serie de capas de convolución y pooling para reducir la resolución espacial y capturar características de alto nivel. En la fase de expansión, se realizan operaciones de upsampling y convoluciones para aumentar la resolución de las características extraídas y combinar la información de alto nivel con detalles locales mediante conexiones de "skip". Las conexiones "skip"permiten la transferencia de información relevante de las capas de alta resolución a las capas de baja resolución, mejorando la precisión de la segmentación.

Actualmente, los modelos que dan mejores resultados en la detección de keypoints suelen estar basados en redes residuales[30] o Vision Transformers[31]. Los Vision Transformers (ViTs) son una innovadora arquitectura de red neuronal que aplica el mecanismo de autoatención de los transformers. Los ViTs dividen cada frame en parches de tamaño fijo y linealizan estos parches para tratarlos como secuencias de tokens, similares a las palabras en un texto. Cada parche se proyecta a un vector de características y se añade un embebido posicional para mantener la información sobre la ubicación de los parches.

El framework más popular actualmente y que mejores resultados da es MediaPipe. MediaPipe[32] es un marco de trabajo de código abierto desarrollado por Google, diseñado para construir y desplegar tuberías de aprendizaje automático (ML) en tiempo real para el procesamiento de datos multimedia como texto, video y audio. Esta herramienta tiene modelos entrenados para abordar diferentes tareas como detección de objetos, clasificación de imágenes o, como nos interesa, detección de keypoints.

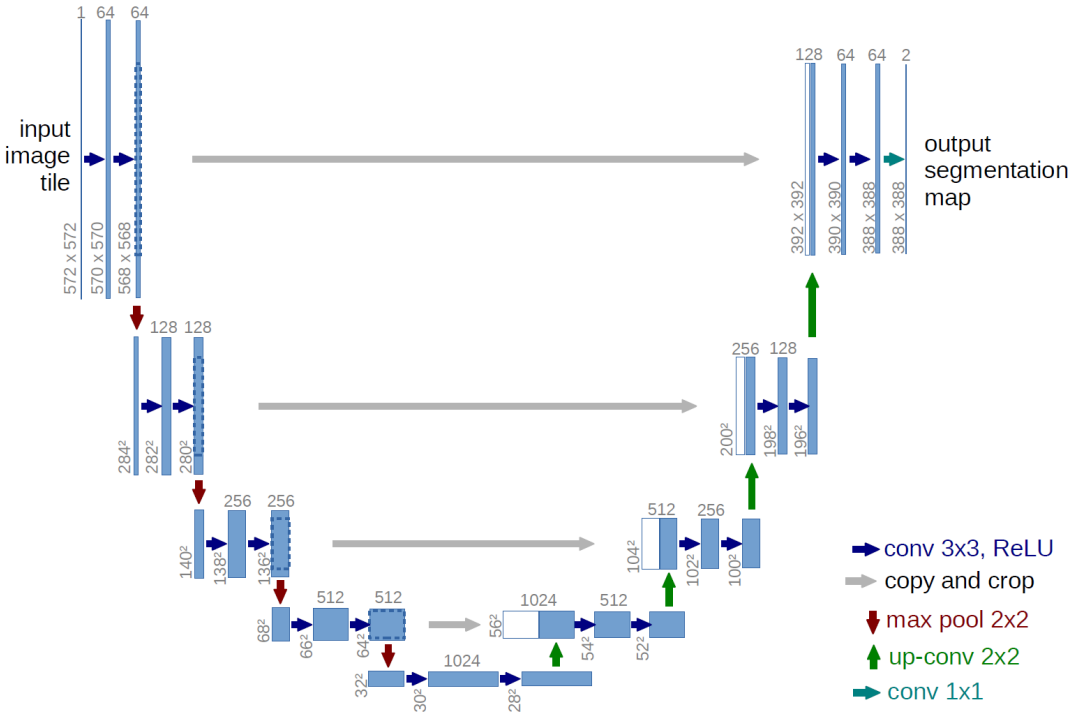

Figura 2.5: Diagrama de la Unet[29]

El modelo subyacente a la detección de keypoints en MediaPipe es conocido como BlazePose[33]. BlazePose consiste en dos modelos diferentes: un detector y un estimador. El detector recorta la figura humana de la imagen y el estimador divide la imagen de la figura humana en ventanas (normalmente de 256x256) como input y los diferentes keypoints como output. El Detector se basa en la arquitectura Single-Shot Detector (SSD)[34]. Toma una imagen de entrada (1,224,224,3) y produce una bounding box (1,2254,12) y un nivel de confianza (1,2254,1). La bounding box incluye coordenadas (x, y, w, h) y keypoints adicionales (kp1x, kp1y, ... kp4x, kp4y). Cada uno de los 2254 elementos tiene su propio ancla, escala y desplazamiento. Por último, el estimador utiliza mapas de calor para el entrenamiento, pero calcula los keypoints directamente para una inferencia más rápida. La primera salida del Estimador es un conjunto de 195 puntos de referencia (x, y, z, visibilidad, presencia) para cada uno de los keypoints.

2.3. Evaluación y métricas

Dentro de la línea de investigación en traducción de lengua de signos a castellano se suele utilizar métricas para comparar los resultados, normalmente respecto al mismo dataset de entrenamiento. Las métricas más utilizadas son el BLEU[35] y el ROUGE[36]. El BLEU (Bilingual Evaluation Understudy) es una métrica ampliamente utilizada para evaluar la calidad de traducciones automáticas comparando el texto predicho por el modelo con el texto real. Para calcularlo se divide el texto en n-gramas, se cuentan las coincidencias entre los n-gramas generados y los de referencia, y se calcula la precisión para cada tamaño de n-grama. Luego, se obtiene la media geométrica de estas precisiones, y se aplica una penalización por brevedad (ecuación 2.4) si

la traducción generada es significativamente más corta que las de referencia. Finalmente, el puntaje BLEU se calcula combinando este promedio con la penalización, reflejando la similitud entre la traducción generada y las referencias, como se observa en la ecuación 2.5. A diferencia de BLEU, que se centra en la precisión, ROUGE se orienta principalmente hacia el recall, lo que significa que mide cuántos de los n-gramas del texto de referencia se capturan en el texto predicho.

$$\text{BP} = \begin{cases} 1 & \text{si } c > r \\ e^{(1-r/c)} & \text{si } c \leq r \end{cases} \tag{2.4}$$

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{2.5}$$

2.4. Datasets

Para entrenar modelos de inteligencia artificial enfocados a la traducción de lengua de signos es esencial contar con métricas que permitan evaluar su rendimiento de manera objetiva y consistente. Sin embargo, estas métricas deben ser calculadas con relación a un mismo conjunto de datos, para asegurar la comparabilidad de los resultados obtenidos.

Un buen conjunto de datos para entrenar modelos de inteligencia artificial, especialmente en el ámbito de la lengua de signos, debe cumplir con varias características clave. En primer lugar, variedad, es decir, debe incluir una amplia gama de signos, estilos de ejecución, y ejemplos de diferentes personas para capturar la diversidad de expresiones y contextos en los que se utiliza la lengua de signos. En segundo lugar, volumen, ya que para construir un buen traductor neuronal es necesario contar con una cantidad suficiente de ejemplos para que el modelo pueda aprender de manera efectiva y generalizar correctamente a nuevos datos. Finalmente, velocidad, en el sentido de que los datos deben estar disponibles y ser procesables en tiempo real o casi real para garantizar que los modelos puedan ser entrenados y evaluados de manera eficiente. Estas "3 V"del Big Data[37] (figura 2.6) son fundamentales para construir un conjunto de datos robusto y representativo, capaz de soportar el desarrollo de modelos precisos y generalizables. En algunos artículos sobre Big Data se suele considerar dos Vs más (valor y veracidad). Sin embargo, las tres características principales son las anteriores, la variedad, el volumen y la velocidad.

En lengua de signos la mayoría de datasets son para la lengua de signos inglesa y alemana. Esto se debe a la mayor disponibilidad de recursos y apoyo en investigaciones para estas lenguas.

El dataset de referencia para la traducción de lengua de signos se llama RWTH-PHOENIX-Weather 2014[38]. Este conjunto de datos contiene grabaciones realizadas durante un período de tres años (2009 - 2011) de las emisiones diarias de noticias y pronósticos meteorológicos de la cadena de televisión pública alemana PHOENIX, que incluyen interpretación en lengua de signos. Los pronósticos meteorológicos de un subconjunto de 386 ediciones fueron transcritos utilizando notación de glosas. Este conjunto contiene más de 95 millones de cuadros, con 67 mil signos y 99 mil palabras en alemán. Además, se utilizó reconocimiento automático de voz con limpieza manual

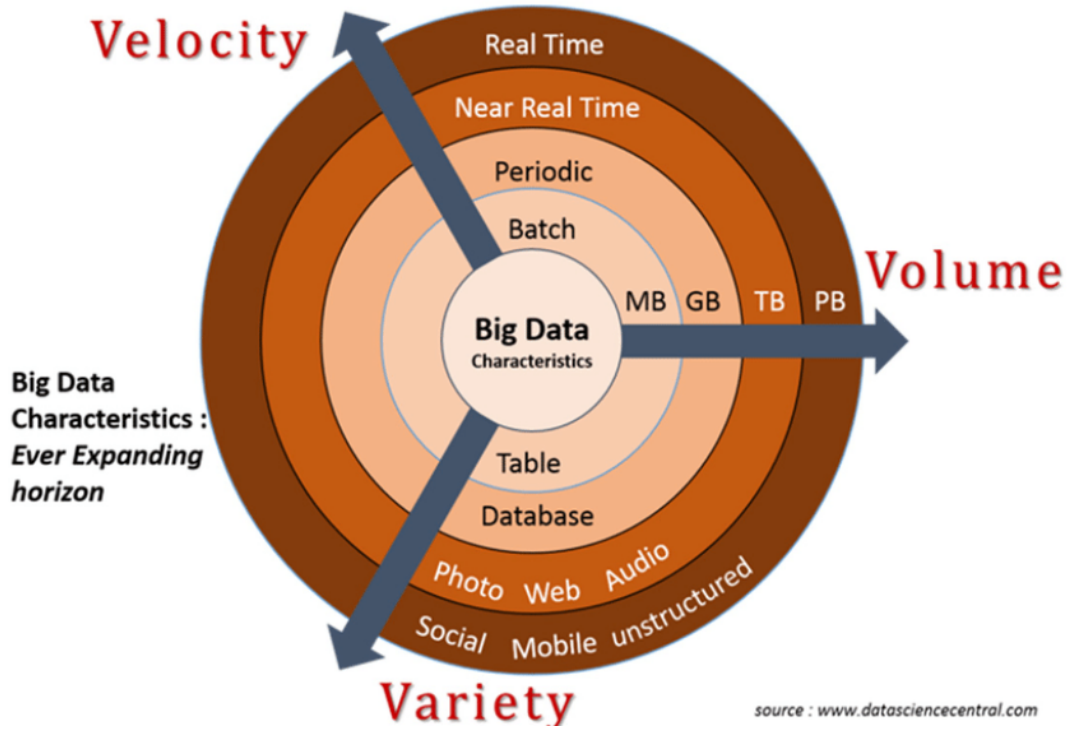

Figura 2.6: 3V del Big Data. Fuente: www.datasciencecentral.com

para transcribir el discurso original en alemán. Este corpus permite, por lo tanto, entrenar sistemas de traducción de lengua de signos de extremo a extremo, desde la entrada en video de la lengua de signos hasta el lenguaje hablado. Todos los videos grabados están a 25 fotogramas por segundo y el tamaño de los fotogramas es de 210 por 260 píxeles, mostrando únicamente el recuadro del intérprete. Por lo tanto, este conjunto de datos está compuesto por frames de vídeos a color.

Otro conjunto de datos de referencia es el CSL-Daily[39] (Chinese Sign Language Corpus), que contiene más de 10000 frases diarias comunes en la lengua de signos china. En este conjunto están anotadas tanto las traducciones reales como las glosas.

Teniendo en cuenta las herramientas como los keypoints, las métricas para evaluar los modelos y el conjunto de datos con el que se evalúa existen varios artículos científicos que tratan de resolver el problema de la traducción de lengua de signos a lenguaje hablado.

2.5. Trabajos pevios

Los primeros artículos[40] sobre este tema datan de 2018. Este artículo primero trata de mostrar la diferencia entre SLR (sign language translation) y CSLR (continous sign language recognition). Básicamente, la diferencia es que CSLR es un proceso de SLT, donde se trata de, a partir de los vídeos, obtener conceptos (glosas). Estos conceptos, luego pasarían por otro proceso para conjugarlos y obtener la traducción, como se observa en la figura 2.7.

A diferencia de los estudios anteriores que trataban el reconocimiento de lenguaje de

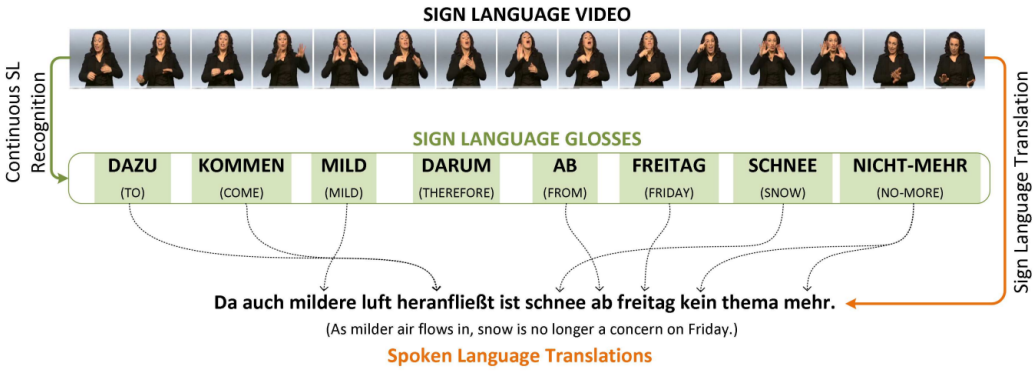

Figura 2.7: Diferencia entre SLT y CSLR[40]

señas (SLR) como un problema básico de reconocimiento de gestos, este trabajo introduce un problema más complejo: traducir directamente las señas en lenguaje hablado o escrito, teniendo en cuenta las diferencias estructurales entre estos lenguajes. Para ello proponen una arquitectura que combina Redes Neuronales Convolucionales (CNN) y mecanismos de atención dentro de una arquitectura encoder-decoder para aprender la representación espacio-temporal de los signos, el modelo de lenguaje subyacente y el mapeo entre el lenguaje de señas y el lenguaje hablado.

El problema de SLT se aborda como un problema de aprendizaje secuencia a secuencia (seq2seq). El objetivo es aprender la probabilidad condicional $p(y|x)$ de generar una oración en lenguaje hablado y dada una secuencia de video de lenguaje de signos x. Dado que los videos de signos tienen muchos más frames que las palabras en la traducción hablada, el modelo debe manejar secuencias de longitud desigual.

Dado que las señas son visuales, se utiliza una red neuronal convolucional (CNN) para aprender representaciones espaciales de cada frame del video. Cada frame se proyecta a un espacio de características no lineal mediante la CNN, con ello se obtiene embeddings de cada frame del video. Para las palabras del lenguaje hablado, se utiliza una capa completamente conectada que proyecta vectores one-hot (representaciones dispersas donde cada palabra es equidistante de las demás) a un espacio continuo más denso.

Tras la tokenización los datos entran a una arquitectura encoder decoder. En la red encoder procesa la secuencia de embeddings espaciales mediante una RNN y genera un vector de estado latente, que luego la recibirá la red decoder. La red decoder recibe el vector latente y la palabra anterior generada (si es la primera palabra de la frase toma el comienzo de frase). Para generar cada palabra se utiliza el embedding de la palabra y el estado oculto de la red anterior. Por último, se emplean mecanismos de atención. Con esta arquitectura se obtiene un BLEU de 19.26 en el dataset RWTH-PHOENIX-Weather 2014.

En 2020 se publicó un paper titulado Better Sign Language Translation with STMC-Transformer[41]. Los autores buscaban abordar las limitaciones de los sistemas actuales de traducción de lenguaje de señas, que suelen depender de una representación intermedia de glosas. En ese modelo, se intenta traducir directamente de videos de señas a texto hablado o escrito.

El paper destaca que las glosas no son una representación precisa ni eficiente del

lenguaje completo. Aunque las glosas ayudan a tokenizar el lenguaje de señas, este proceso introduce una pérdida de información debido a que las glosas simplifican las señales multidimensionales del lenguaje de señas a una forma más lineal y limitada. Los autores proponen que la traducción basada en glosas no necesariamente produce mejores resultados que un modelo de traducción directo de video a texto.

El trabajo introduce un modelo llamado STMC-Transformer para la traducción de video a texto, que mejora significativamente el rendimiento en comparación con modelos anteriores que se basan en glosas. Además de los Transformers, los autores aplican técnicas avanzadas como el weight tying, transfer learning, y el ensemble learning, las cuales mejoran aún más el rendimiento del modelo.

El STMC-Transformer (figura 2.8) es una combinación de dos redes principales:

- **STMC Network**: Se encarga de procesar las señales visuales multidimensionales del video de señas. El modelo se basa en múltiples módulos que descomponen las señales en características espaciales y temporales, analizando elementos visuales clave como la cara, las manos y la pose.
- **Transformer**: Para la parte de la traducción, se utiliza un Transformer que ha demostrado ser efectivo en tareas de traducción automática. Este modelo incluye mecanismos de atención, positional encoding y redes feed forward que permiten capturar las relaciones dependientes entre distintas partes de la señal del lenguaje de señas, facilitando una traducción más precisa.

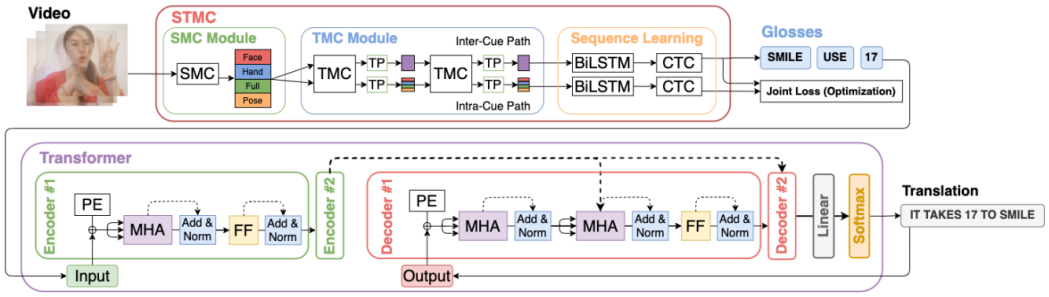

Figura 2.8: Modelo STMC-Transformer[41]

Esta red ha sido probada con el conjunto de datos de referencia y con otros respecto a dos tareas distintas, que son Gloss-to-Text y Sign-to-Gloss-to-Text. El trabajo sugiere que los sistemas de traducción automática de lenguaje de señas pueden mejorar eliminando el uso de glosas como representación intermedia, aumentando el BLEU a 25.4.

El siguiente estudio que superó el estado del arte fue Stocvhastic Transformer Networks with Linear Competing Units: Application to End-to-End SL Translation[42]. Este trabajo introduce un nuevo enfoque para la traducción de lenguaje de signos que elimina la necesidad de utilizar glosas como representación intermedia, diferenciándose de modelos anteriores. La principal innovación de este trabajo es la creación de una red Transformer estocástica con unidades lineales en competencia (Linear Competing Units) y pesos estocásticos ajustados mediante inferencia bayesiana variacional.

El uso de capas LWTA[43] (Local Winner-Takes-All) en vez de capas densas norma-

les con función de activación relu añadió un proceso estocástico a la arquitectura, lo que proporcionaba variabilidad y permitió mejorar la capacidad de representación del modelo. Estas capas introducen un mecanismo de competencia local entre varias unidades dentro de un bloque, de modo que solo una de estas unidades puede "ganarז pasar su activación a la siguiente capa de la red.

Además, los pesos de la red se tratan como variables aleatorias, y se ajustan distribuciones gaussianas (ecuación 2.6) a posteriori mediante inferencia bayesiana. Esta aproximación proporciona una ventaja en términos de rendimiento y permite la compresión eficiente del modelo sin una pérdida significativa de precisión.

$$\mathrm{q(w)} = \mathcal{N}(w|\mu, diag(\sigma^2)) \tag{2.6}$$

Por último, este trabajo introduce un proceso de compresión de la red, lo cual reduce considerablemente el tamaño del modelo (en más de un 70 %) sin sacrificar el rendimiento. Este enfoque se basa en la eliminación de bits menos significativos en los pesos de la red, según la varianza de las distribuciones gaussianas ajustadas a posteriori.

En resumen, este paper presenta un enfoque más avanzado y eficiente para la traducción de lenguaje de signos, eliminando las limitaciones asociadas con el uso de glosas y mejorando el rendimiento con técnicas estocásticas y compresión de red. Con estas innovaciones el BLEU respecto al dataset de referencia[38] aumentó a 25.59.

En 2022 se publicó un paper titulado Two-Stream Network for Sign Language Recognition and Translation[44], donde se introduce una nueva arquitectura de red neuronal diseñada para mejorar el reconocimiento y la traducción de lenguaje de señas, abordando los desafíos que presentan las señales visuales complejas y las limitaciones de los enfoques previos basados únicamente en videos RGB. Los autores proponen un modelo de dos flujos (Two-Stream Network), llamado TwoStream-SLR para el reconocimiento de lenguaje de señas (SLR) y TwoStream-SLT para la traducción de lenguaje de señas (SLT). La innovación de este paper es el uso de los keypoints para el reconocimiento de señas.

La arquitectura se basa en el uso de dos entradas distintas, las cuales son el vídeo en RGB (como se solía hacer hasta ahora) y los keypoints. En la figura 2.9 se puede observar el proceso que se sigue para el entrenamiento. Como entrada se recibe, antes de nada, el vídeo en RGB y se pasa por un proceso para extraer los keypoints. Los keypoints se pueden representar mediante un vector, en el que se guarden las componentes espaciales de los keypoints para todos los frames del vídeo o el vídeo (con el fondo negro) donde solamente esté en blanco los keypoints (o un mapa de calor de ellos). Los vídeos y los keypoints van a ser la entrada para la red de reconocimiento de signos.

La red, que es una arquitectura multistream, las cuales se han visto que obtienen muy buen resultado en el reconocimiento de acciones[45][46][47], está compuesta por lo siguiente (figura 2.10):

- **Video Encoder**: Se utiliza la red S3D[48], enfocada en la extracción de representaciones densas a lo largo de la dimensión temporal. El video de entrada es procesado para generar una representación comprimida en el tiempo que

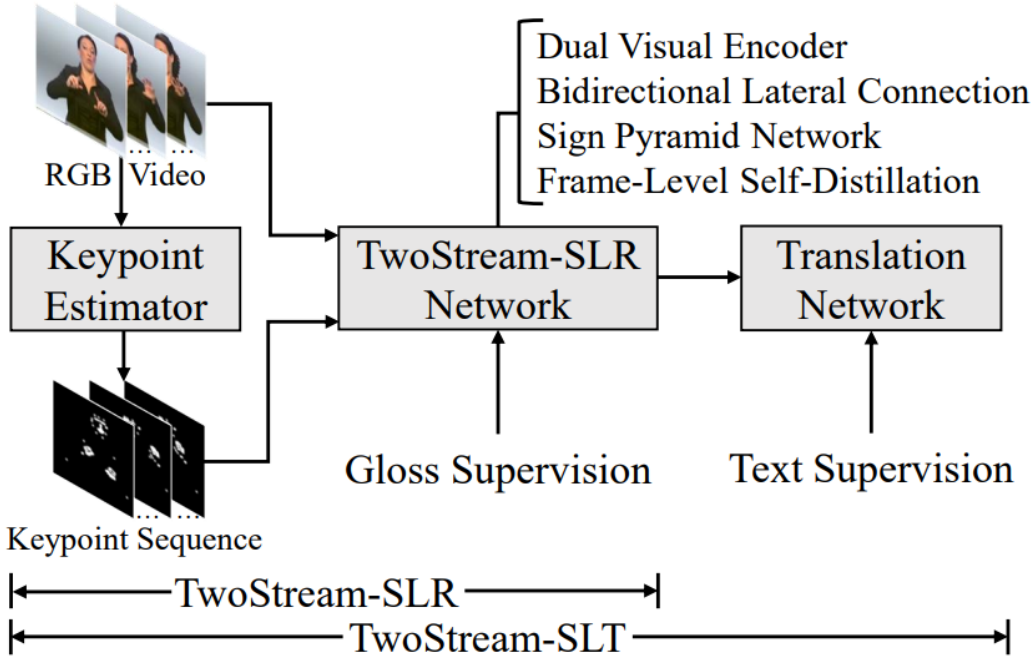

Figura 2.9: Resumen del proceso de la red Two Stream[44]

captura las características clave. El objetivo del "head network.[es] refinar las características temporales a través de capas lineales y convolucionales. Al final, se genera una probabilidad a nivel de cuadro para cada glosa, que es optimizada utilizando la pérdida CTC[49].

- **Keypoint Encoder**: Este componente utiliza la red HRNet[50] para extraer keypoints de las manos, rostro y parte superior del cuerpo, los cuales se representan mediante mapas de calor. La arquitectura del codificador de keypoints es similar a la del codificador de video, pero ajustada para manejar los mapas de calor generados por los puntos clave. Las predicciones de los puntos clave también se optimizan con la pérdida CTC.
- **Bidirectional Lateral Connection**: Este mecanismo permite la interacción entre los flujos de video y keypoints, combinando la información de ambos a través de una operación de suma entre mapas de características de las dos corrientes. Esto ayuda a que ambos flujos se complementen y mejoren el rendimiento.
- **Joint Head y Late Ensemble**: Además de los "head networksïndependientes para los flujos de video y keypoints, se introduce una cabeza conjunta que concatena las salidas de ambos flujos para generar una representación unificada. El late ensemble se realiza promediando las probabilidades generadas por los tres "heads", mejorando los resultados generales.
- **Sign Pyramid Network**: Esta red se agrega sobre el codificador visual dual para capturar glosas de diferentes duraciones temporales. La SPN ayuda a que las capas más superficiales aprendan representaciones significativas, proporcionando supervisión auxiliar a través de la pérdida CTC.
- **Frame-Level Self-Distillation**: Dado que los conjuntos de datos de lenguaje

de signos no proporcionan anotaciones detalladas de los límites temporales de las glosas, se utiliza la pérdida CTC para obtener una supervisión débil a nivel de secuencia. Las probabilidades de glosas a nivel de cuadro se utilizan como pseudo-etiquetas para guiar el aprendizaje de cada flujo, mejorando la precisión del modelo.

- **Función de Pérdida**: La función de pérdida total combina las pérdidas CTC aplicadas a los codificadores de video, puntos clave y la cabeza conjunta, junto con las pérdidas auxiliares y la pérdida de auto-destilación. Esta combinación permite al modelo aprender de manera efectiva y mejorar su capacidad de predicción.

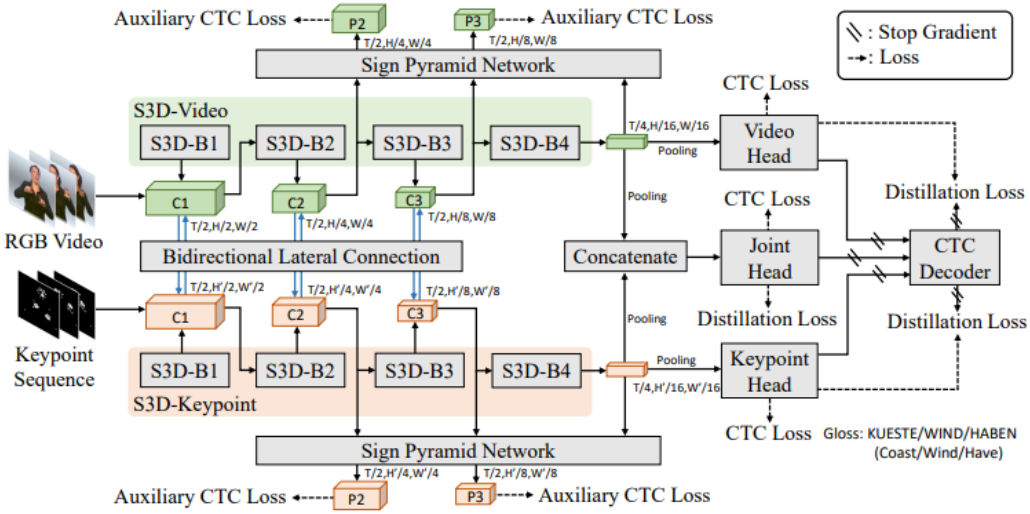

Figura 2.10: Arquitectura de la red Two Stream[44]

En conclusión, la introducción del modelo Two-Stream es un avance diferencial en la tarea de reconocimiento y traducción de lenguaje de señas, ya que permite combinar de manera efectiva la información proveniente de videos RGB y secuencias de keypoints. Este enfoque dual maximiza la capacidad del modelo para capturar tanto las características manuales como las no manuales del lenguaje de señas, lo que mejora significativamente la precisión en comparación con los modelos anteriores. Con este modelo se logra un BLEU de 28.95 respecto al dataset[38] de referencia.

Finalmente, el paper referencia del estado del arte, publicado en 2024, se titula Multi-Stream Keypoint Attention Network for Sign Language Recognition and Translation[51] e introduce un enfoque novedoso para el reconocimiento y la traducción de lenguaje de señas. Este enfoque utiliza secuencias de puntos clave en lugar de videos RGB tradicionales para mejorar la robustez frente a variaciones de fondo y reducir las demandas computacionales.

El modelo MSKA-SLR (Multi-Stream Keypoint Attention Sign Language Recognition) utiliza módulos de atención para procesar las secuencias de keypoints en paralelo y capturar características locales y globales de las articulaciones del cuerpo. Además, se introduce un mecanismo de fusión que permite combinar las salidas de los diferentes flujos, mejorando así la precisión del modelo. El sistema también incorpora un proceso de auto-destilación a nivel de cuadro, donde las predicciones de glosas a nivel

de cuadro se utilizan como pseudo-etiquetas para proporcionar una supervisión más detallada. El modelo se amplía al MSKA-SLT (Sign Language Translation) al agregar una red de traducción que convierte las características codificadas en secuencias de texto, utilizando un enfoque de traducción automática neuronal.

La principal innovación de este trabajo radica en el uso de un enfoque completamente basado en keypoints, sin necesidad de procesar directamente los videos RGB. Esto contrasta con métodos anteriores que combinaban videos y keypoints o que dependían exclusivamente de videos, ofreciendo un enfoque más robusto y eficiente.

La arquitectura de la figura 2.11 es un modelo para el reconocimiento de la lengua de signos a partir de keypoints del cuerpo, rostro, y manos. Se compone de módulos de atención que procesan estos puntos clave de forma independiente (cuerpo, manos izquierda y derecha, rostro) y luego aplican "pooling"para reducir la dimensionalidad. Cada parte del cuerpo tiene una cabeza de predicción (Body, Right, Face, Left Heads) que emplea una pérdida CTC para entrenar secuencias temporales. Además, hay una cabeza de fusión que combina la información de todas las partes para mejorar la predicción final, utilizando una pérdida de destilación para optimizar la generalización del modelo. Esta arquitectura modular permite un enfoque detallado y especializado en cada aspecto crítico de la lengua de signos.

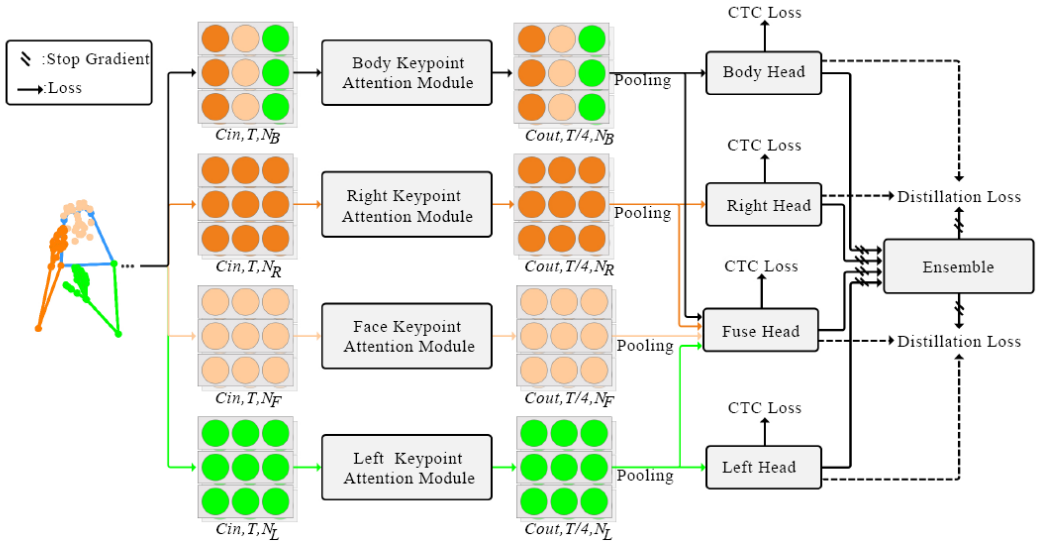

Figura 2.11: Arquitectura de la red MSKA-SLR[51]

El modelo propuesto, MSKA-SLR, y su extensión para traducción, MSKA-SLT, representan un avance significativo en la comprensión y traducción del lenguaje de signos. Al enfocarse en la atención multi-stream y la interacción entre flujos de keypoints, el modelo logra mejoras notables en precisión y eficiencia. El modelo entrenado con el dataset de referencia[38] obtiene un BLEU de 29.03, siendo el mejor hasta la fecha.

En resumen, hemos comprobado como ha ido evolucionando el campo de investigación de la traducción de lengua de signos a lengua hablada. Hemos visto como, año tras año, se mejoraba el resultado innovando cada vez más en la arquitectura de la red. El aumento más significativo concuerda con la introducción de los keypoints como herramienta para los videos. A continuación, teniendo todo lo anterior en cuenta, vamos a desarrollar un prototipo para la traducción de la LSE.

Capítulo 3

Desarrollo

En esta sección de desarrollo, se detalla el proceso completo llevado a cabo para la creación de un traductor neuronal básico de la Lengua de Signos Española (LSE). En primer lugar, se justifica la elección y construcción de nuestro dataset. Posteriormente, se describe el desarrollo del modelo de reconocimiento de signos, explicando las decisiones tomadas en cuanto a la arquitectura del modelo, los algoritmos de aprendizaje utilizados y los métodos de optimización aplicados. A lo largo del proceso, se han seguido criterios rigurosos para garantizar que las soluciones implementadas sean las más efectivas y adecuadas para el objetivo final de este trabajo.

3.1. Dataset

Como hemos comentado anteriormente, para lenguas de signos más habladas y populares como la americana o la china existen datasets ya preparados y etiquetados para su uso en investigación, como lo son RWTH-PHOENIX-Weather 2014[38] y CSL-Daily[39]. Para el castellano existen datasets de reconocimiento de signos, pero con imágenes en vez de vídeos. Por ejemplo, existe un dataset llamado Dactilología del Lenguaje de Signos Español[52], el cual contiene imágenes de los signos que representan las 26 letras del abecedario español. En total hay 2600 imágenes (100 por cada letra). Esto para nuestro trabajo no nos sirve, ya que nosotros vamos a trabajar con vídeos. Por lo tanto, nuestra única opción es construir nuestro propio dataset de lengua de signos española.

Para la lengua de signos española existen varias páginas que contienen diccionarios de conceptos (o glosas). La más común es el diccionario del Centro de Normalización Lingüística de la Lengua de Signos Española llamado Dilse[53] (Diccionario de la lengua de signos española). Este diccionario bilingüe de lengua de signos y castellano recoge más de 3.500 signos. La web permite consultar, descargar y compartir tanto las fotos como los vídeos de los signos. Sin embargo, por la forma en la que está construida la página web no permite la descarga del diccionario completo ni la descarga con web scrapping.

Finalmente, el diccionario que hemos escogido como base para construir nuestro dataset es el de ARASAAC[54] (Centro Aragonés para la Comunicación Aumentativa y Alternativa). Este diccionario contiene más de 3800 vídeos de lengua de signos. Esta página tampoco permite descargar una carpeta comprimida con todos los vídeos, pero

sí que se puede hacer mediante web scrapping.

Web scraping es el proceso de extraer información de sitios web de manera automatizada, generalmente utilizando programas o scripts que navegan por las páginas, extraen datos específicos y los almacenan para su análisis o uso posterior. Para ello utilizaremos la librería de python Selenium[55]. Selenium es una herramienta para automatizar navegadores web y es muy útil para descargar archivos de sitios web, especialmente cuando estos archivos no están directamente accesibles a través de un enlace estático (por ejemplo, cuando los enlaces se generan dinámicamente o requieren interacción del usuario).

Mediante un script de python (ver Anexo .1) donde hagamos uso de esta librería recorriendo todas las páginas del sitio web, seleccionando todos los archivos y los añadiéndolos a una lista de descargas de forma automática. Por último, se descarga todo en una carpeta comprimida y se descomprime manualmente o mediante otro script de python. Ahora hemos obtenido un conjunto de 3800 vídeos de conceptos de lengua de signos, pero lo que nos interesa son las frases.

Antes de formar las frases vamos a crear otro conjunto de datos con los videos con los keypoints marcados. Crearemos un dataset con los vídeos RGB más los keypoints y otro con todo el fondo negro menos los keypoints. Para ello hacemos uso de la librería mediapipe[32] para extraer los keypoints en cada frame, dibujarlos en una imagen y construir un nuevo video con los keypoints frame a frame (ver Anexo .2). Cabe destacar que esto es un proceso largo debido a la cantidad de frames que hay en cada video y la cantidad de vídeos (más de 100 horas de ejecución).

Con los keypoints creados ahora sí que podemos crear las frases. Como nosotros lo que buscamos es un modelo que sea capaz de reconocer frases, no nos importa la coherencia en las frases, por lo que vamos a hacer frases aleatorias. Para no hacer videos demasiado largos haremos las frases de 3 palabras. Para hacer comparaciones vamos a hacer frases teniendo en cuenta todas las palabras y frases donde solo tendremos en cuenta 50 de las palabras más frecuentes en la LSE, que se pueden observar en la figura 3.1. Generaremos un total de 5000 videos, teniendo así un conjunto de datos bastante considerable.

3.2. Desarrollo del modelo

El desarrollo del modelo de traducción para la Lengua de Signos Española (LSE) siguió un proceso iterativo en el que se evaluaron varias herramientas y enfoques antes de llegar a la solución final. Inicialmente, probamos herramientas de traducción automática, como OpenNMT[56], para evaluar si podían adaptarse para procesar secuencias de vídeo en lugar de texto. OpenNMT es una plataforma de código abierto para la traducción automática basada en modelos de redes neuronales, generalmente utilizada para tareas de traducción de texto a texto. El modelo subyacente a OpenNMT es un transformer, una arquitectura que ha demostrado ser muy efectiva en tareas de procesamiento de lenguaje natural.

Sin embargo, adaptar OpenNMT a la entrada de vídeo requeriría extraer embeddings de los fotogramas de los vídeos, una representación numérica compacta que encapsula las características relevantes de cada fotograma. A pesar de que esta aproximación podría parecer viable, la variabilidad en los vídeos de lengua de signos, debido

Figura 3.1: Nube de palabras con las palabras de nuestro dataset

a diferencias en el estilo de señas, ángulos de cámara y otros factores, haría que los embeddings no fueran consistentes entre diferentes vídeos que representan la misma señal. Esto podría llevar a una considerable pérdida de precisión en la traducción.

Posteriormente, intentamos utilizar un código basado en el modelo de un artículo científico conocido como Modelo TwoStream[44]. Este modelo emplea dos flujos paralelos para capturar información tanto del movimiento como de la apariencia visual en los vídeos. Aunque esta aproximación parecía prometedora, nos encontramos con que las entradas esperadas por el modelo no se alineaban exactamente con nuestras secuencias de vídeo. Además, la modularización excesiva del código dificultaba realizar las adaptaciones necesarias para nuestro caso específico, lo que llevó a la conclusión de que esta no era una solución práctica.

Finalmente, decidimos desarrollar un modelo propio, combinando componentes del TwoStream con otros elementos personalizados. Utilizamos el encoder del modelo TwoStream, que es responsable de capturar las características clave de las secuencias de vídeo, y lo combinamos con un decoder basado en una LSTM (Long Short-Term Memory), una arquitectura recurrente que es particularmente efectiva en la gestión de secuencias temporales. Este diseño nos permitió manejar la variabilidad en los vídeos de lengua de signos y generar traducciones más precisas y consistentes.

Este enfoque, aunque más laborioso, nos proporcionó un mayor control sobre las especificidades de nuestro problema y permitió construir un modelo que se adaptara mejor a las características únicas de la LSE.

3.3. Modelo Final

3.3.1. Preparación de datos

Para nuestro modelo lo hemos programado en python. Utilizaremos varias bibliotecas clave como PyTorch para el manejo del modelo y los datos, transformers para el uso de BERT en la tokenización, cv2 para la manipulación de videos, y nltk para la

evaluación de traducciones con la métrica BLEU. El uso de CUDA también es considerado, lo que permite utilizar una GPU para acelerar los cálculos si está disponible.

En primer lugar, se cargan listas de palabras personalizadas (se observan en la figura 3.1) desde un archivo de texto, que junto con tokens especiales como [SOS] (start of sequence), [EOS] (end of sequence), [PAD] (espacios en blanco), y [UNK] (desconocido), se utilizan para crear un vocabulario. Este vocabulario se mapea a índices, permitiendo que las palabras se representen como secuencias numéricas, necesarias para entrenar nuestro modelo.

En segundo lugar, creamos la clase SignLanguageDataset, que es una subclase de Dataset de PyTorch, diseñada para manejar los datos de los videos de lenguaje de señas. Los métodos de esta clase cargan los videos y los ajustan a un número fijo de frames (debido a la alta complejidad computacional que supone procesar todos los frames del video), convirtiéndolos en tensores para que puedan ser procesados por la red neuronal. En nuestro caso, para poder ejecutar correctamente el modelo y en un tiempo razonable hemos establecido 50 frames. Además, las etiquetas textuales se tokenizan utilizando el vocabulario previamente definido, y se preparan como secuencias de tokens para que puedan ser alimentadas al modelo como objetivos de entrenamiento.

Para terminar con la preparación de los datos dividimos el dataset (tanto videos como etiquetas) dos nuevos datasets, que uno va dirigido al entrenamiento del modelo (80 % de los vídeos) y el otro a la validación del mismo (20 % restante).

3.3.2. Encoder

El encoder de nuestro modelo está basado en el encoder del modelo TwoStream[44], pero una versión más simple. El diseño del encoder sigue un enfoque de red neuronal profunda, utilizando bloques convolucionales 3D (convoluciones tridimensionales) y una red de cabeza (HeadNetwork) que refina estas características y produce las salidas finales. A continuación, se detallan los elementos del Encoder:

- **S3D:** Este es básicamente un bloque, donde se realiza una convolución3D 3x3 con stride 1 y padding 1, seguido de una normalización por lotes (Batch Normalization) y, por último, una función de activación Relu.
- **HeadNetwork:** La red consiste en dos capas de convoluciones tridimensionales adicionales enfocadas principalmente en capturar patrones temporales, cada una seguida de normalización por lotes y activación ReLU para estabilizar el entrenamiento y añadir no linealidad. Tras la extracción de estas características temporales, se realiza un promedio sobre las dimensiones espaciales, reduciendo la dimensionalidad del tensor. Finalmente, la salida se aplana y se pasa por una capa fully connected que convierte estas características en probabilidades logarítmicas para cada clase objetivo, utilizando la función logsoftmax para asegurar que las salidas sumen uno.
- **Encoder:** Esta clase agrupa las dos anteriores en una arquitectura coherente que procesa secuencias de video desde una entrada en bruto hasta la extracción de características útiles. Está compuesta de 4 bloques S3D (cada vez mayor la salida) y de una HeadNetwork.

En la figura 3.2 se puede observar la arquitectura del Encoder resumida. La entrada

son vídeos de dimensión TxHxWx3, siendo T el número de frames, 3 el número de canales (RGB) y, H y W, el ancho y alto de cada fotograma. Tras pasar por toda la arquitectura la salida tiene una dimensión de T/4 x 512.

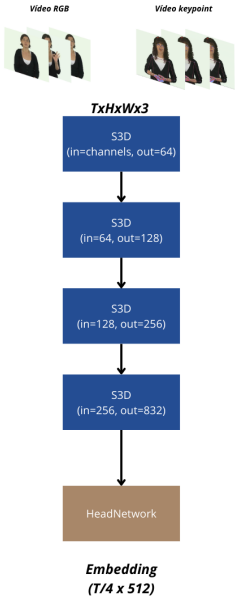

Figura 3.2: Encoder de vídeo

3.3.3. Decoder

El Decoder es una parte crucial de un modelo secuencial, diseñado para generar una secuencia de salida basada en la información codificada por el Encoder. En este caso, está compuesto por varias capas: una capa de embeddings, una red LSTM (añadiremos la GRU en las pruebas), y una capa fully connected con softmax para la salida final.

La Long Short-Term Memory (LSTM) es un tipo de red neuronal recurrente (RNN) diseñada para modelar secuencias de datos, como texto o vídeos, y es particularmente útil cuando es necesario capturar dependencias temporales o relaciones a largo plazo entre los elementos de una secuencia. Una red LSTM está compuesta por unidades llamadas celdas LSTM, y cada celda tiene una estructura interna que incluye tres puertas (como se observa en la figura 3.3) que controlan el flujo de información:

- **Puerta de Olvido (Forget Gate):** Esta puerta decide cuánta información de la celda anterior debe ser olvidada. La decisión se basa en la entrada actual y en el estado oculto anterior. Matemáticamente, esto se expresa como una multiplicación entre el estado de la celda y un valor entre 0 y 1 (salida de una función sigmoide), donde 0 significa olvidar completamente y 1 significa retener toda la información.
- **Puerta de Entrada (Input Gate):** Esta puerta decide cuánta información nueva será agregada al estado de la celda. Se calcula de dos maneras: una parte decide qué valores actualizar, y otra crea un nuevo vector de posibles valores candidatos para ser añadidos al estado de la celda.

- **Puerta de Salida (Output Gate):** Esta puerta decide qué parte del estado interno de la celda será usada como la salida de la LSTM en este paso temporal. La salida es controlada tanto por la puerta de salida como por el estado de la celda actual, lo que ayuda a generar el siguiente estado oculto.

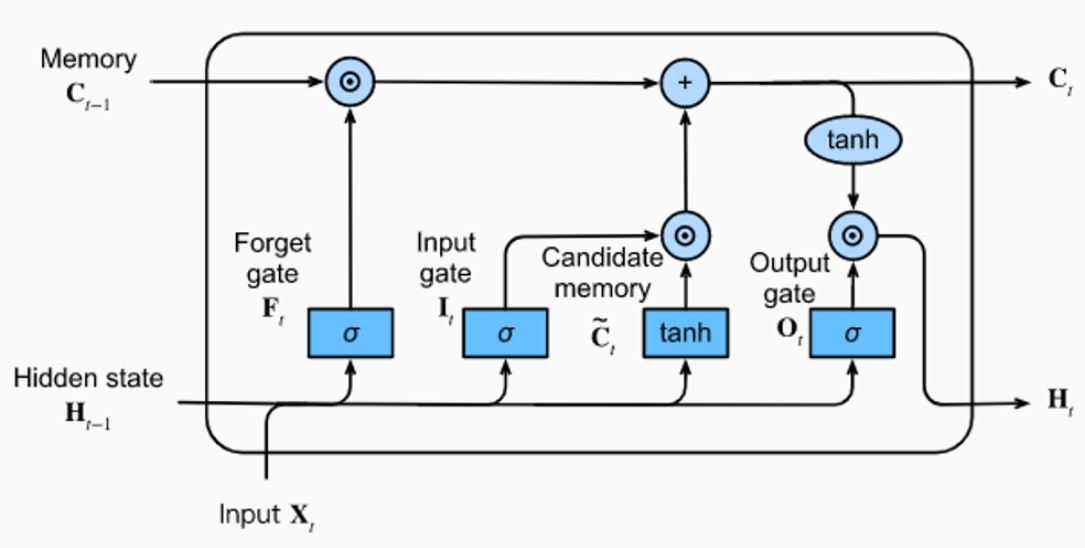

Figura 3.3: Ilustración de la red LSTM. Fuente: `https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb`

En el traductor neuronal para la Lengua de Signos Española (LSE), el decoder basado en LSTM juega un papel clave en la generación de la secuencia de palabras que corresponden a los signos de los vídeos. El decoder LSTM funciona de manera secuencial, procesando paso a paso la información que recibe. Para cada palabra que genera el decoder, las entradas que recibe la LSTM son:

- **Salida del encoder:** Después de procesar el vídeo con el encoder (basado en convoluciones 3D), se obtiene una representación comprimida del vídeo (embedding) que encapsula las características relevantes de los signos. Esta salida se combina en cada paso temporal con la palabra anterior generada por la LSTM.
- **Palabra anterior:** En la fase de entrenamiento, la palabra anterior es la palabra real de la secuencia de referencia , pero durante la traducción real, la palabra anterior es la palabra que la LSTM ha generado en el paso anterior.

Durante la fase de entrenamiento, esta palabra se toma de la secuencia de palabras correctas como referencia. Sin embargo, durante la inferencia, la palabra que se utiliza es la que la LSTM ha generado en el paso anterior. Para procesar la palabra, primero se convierte en un vector denso utilizando una capa de embeddings (disponible para texto escrito). Luego, este vector de palabra se concatena con la representación del vídeo, lo que permite que la LSTM procese tanto el contexto visual como la información lingüística previa al tomar decisiones sobre qué palabra generar a continuación.

Una vez que la LSTM recibe la concatenación de la palabra anterior y las características del vídeo, utiliza sus mecanismos internos para procesar esta información y actualizar sus estados ocultos. La puerta de olvido decide cuánta información de pasos anteriores debe descartar, mientras que la puerta de entrada determina qué

parte de la nueva información debe añadirse al estado de la celda. Finalmente, la puerta de salida controla qué información se debe utilizar para generar la salida en el paso actual. Estas puertas permiten que la LSTM gestione de manera efectiva la información a lo largo de la secuencia temporal, reteniendo solo lo más relevante para generar la siguiente palabra en la oración.

Después de actualizar sus estados, la LSTM produce una nueva salida. Esta salida pasa por una capa fully connected, que proyecta el estado oculto de la LSTM en un espacio de dimensionalidad igual al del vocabulario. A continuación, se aplica una función de activación softmax para convertir esta salida en una distribución de probabilidades sobre todas las palabras posibles del vocabulario. El modelo selecciona la palabra con la mayor probabilidad como la más adecuada para continuar la secuencia.

La palabra generada en este paso se convierte en la entrada para el siguiente paso temporal, junto con la representación del vídeo que permanece constante durante toda la secuencia. Este ciclo de concatenación, actualización de la LSTM y generación de palabras se repite en cada paso temporal, hasta que el modelo alcanza un token de fin de secuencia (EOS). De esta manera, la LSTM construye una oración completa, traduciendo de forma secuencial el contenido visual del vídeo en Lengua de Signos Española a una secuencia de palabras en texto.

En resumen, a lo largo de este capítulo se ha descrito el proceso de desarrollo de un traductor neuronal básico de la Lengua de Signos Española (LSE). Se comenzó con la necesidad de construir un dataset propio, dado que no existen bases de datos adecuadas en español que contengan vídeos etiquetados para tareas de traducción de la LSE. Sin embargo, debido a los recursos limitados solo pudimos contruir un dataset a partir de un diccionario, por lo que solo tenemos una muestra por signo.

Posteriormente, se detalló el proceso de desarrollo del modelo, desde las primeras aproximaciones con herramientas preexistentes como OpenNMT y el modelo TwoStream, hasta la construcción de un modelo propio adaptado a las necesidades específicas de este trabajo. Se integraron distintas arquitecturas y componentes, como el encoder basado en el TwoStream y un decoder LSTM personalizado, lo que permitió gestionar la complejidad y variabilidad de los vídeos en lengua de signos.

El modelo final resultante emplea una arquitectura robusta que combina convoluciones 3D para la extracción de características de los vídeos, junto con redes recurrentes para manejar las secuencias temporales de los signos. A través de un proceso iterativo de desarrollo, se logró una solución que es capaz de traducir videos de lengua de signos española (dentro de las limitaciones de nuestro dataset).

Capítulo 4

Resultados

En este capítulo se muestran los resultados obtenidos en el TFM con nuestro modelo. La métrica que vamos a utilizar para medir los resultados es el BLEU, que la hemos explicado anteriormente. Además, vamos a explicar los resultados de manera temporal, donde se verá la importancia que ha tenido en nuestros resultados la elección del dataset. Cabe destacar que todas las pruebas realizadas han sido a 150 épocas, por lo que las ejecuciones pueden tardar semanas.

Nuestros primeros intentos fueron utilizando el diccionario completo de palabras (más de 3800 signos distintos). Con este dataset y nuestro modelo obtenemos resultados muy incoherentes. Como se observa en la figura 4.2, las predicciones no tienen nada que ver con las frases originales (figura 4.1). Es más, el BLEU total (suma de los bleu de cada frase de validación) fue de 4.39 y tan solo acertó cuando las palabras eran desconocidas. Esto se debe a la cantidad de posibles signos que tenemos y tan solo tenemos un video por cada uno, eso aporta muy poca información al modelo y hace que no pueda aprender correctamente. Debido a esto es por lo que reducimos en número de signos posibles a 50.

- Texto1: paisaje huella parche
- Texto2: menos cuna minero
- Texto3: pantalones rabanitos lumbalgia
- Texto4: ojos camino cuidadora
- Texto5: lleno mortero carnaval
- Texto6: trapo banco_sueco u

Figura 4.1: Frases de validación

- Predicción1: pelar viento viento
- Predicción2: sorprendido [UNK] chanclas
- Predicción3: 2_dado transportes historia
- Predicción4: sal cartera jinete
- Predicción5: zanahoria [UNK] sable
- Predicción6: prohibido_fumar aquellos zumbar

Figura 4.2: Predicción de las frases de validación

Con nuestro dataset final los resultados mejoran considerablemente. Al tener menos caracteres posibles es mucho más simple la clasificación, ya que nuestra softmax final del modelo pasa de tener 3800 posibilidades a solo 50. Con la LSTM como decoder mejoramos significativamente los resultados, vemos como las predicciones (figura 4.4) ahora sí que son mucho más acertadas a los textos de validación (figura 4.3).

- Texto1: flamenco radiador nunca
- Texto2: camilla desnivel numerador
- Texto3: flamenco salamandra croquetas
- Texto4: bloques entre colocar
- Texto5: pez_espada derecha novios
- Texto6: berenjena patines grifo_monomando

Figura 4.3: Frases de validación con el dataset final

- Predicción1: flamenco radiador salamandra
- Predicción2: camilla anochecer numerador
- Predicción3: flamenco croquetas croquetas
- Predicción4: bloques entre flamenco
- Predicción5: cosechadora derecha entre
- Predicción6: berenjena hermano chelo

Figura 4.4: Predicción de las frases de validación con el dataset final

La tabla 4.1 muestra los resultados obtenidos en términos de la puntuación BLEU para dos tipos de vídeos: Videos RGB y Videos con keypoints, utilizando dos tipos de decoders, LSTM y GRU.

El modelo basado en LSTM obtiene una puntuación BLEU de 75.64 con los vídeos en formato RGB. Esto sugiere un rendimiento moderado, indicando que el modelo logró capturar de manera adecuada la información visual de los vídeos y generar secuencias traducidas con una calidad razonable. El decoder GRU, en comparación con el LSTM, parece tener un comportamiento muy diferente con una puntuación BLEU significativamente más alta (366.45).Con vídeos basados en keypoints, el LSTM obtiene una puntuación BLEU de 183.08. Este valor es considerablemente mayor

que en el caso de los vídeos RGB, lo que sugiere que trabajar con keypoints podría proporcionar información más relevante o más fácil de procesar para el modelo.

Tipo de Video	**LSTM**	**GRU**
Videos RGB	75.64	366.45
Videos con keypoints	183.08	-

Cuadro 4.1: BLEU total de los videos RGB y con keypoints con LSTM o GRU como Decoders

Los resultados muestran que, en general, los keypoints ofrecen una ventaja para el modelo basado en LSTM al proporcionar una mejor puntuación BLEU. Por último, parece que el decoder con GRU funciona mejor que el LSTM y, por regla de tres, parecería que el mejor resultado se obtendría con el decoder GRU y los vídeos con keypoints. Sin embargo, eso no podemos saberlo con seguridad, ya que la ejecución no ha acabado a tiempo.

Capítulo 5

Conclusiones

5.1. Conclusión

En el presente trabajo, hemos diseñado un traductor neuronal básico orientado a la Lengua de Señas Española (LSE), utilizando técnicas de aprendizaje profundo. A lo largo del proyecto, se abordaron varios desafíos asociados al procesamiento y la traducción de videos de señas, logrando implementar una arquitectura funcional que recibe videos en lenguaje de signos a texto escrito.

Uno de los logros más significativos ha sido la creación de un pipeline integral que abarca desde el preprocesamiento de los videos hasta la traducción final. No obstante, es crucial señalar que los resultados alcanzados no son directamente comparables con los de otros estudios, dado que no existe un conjunto de datos de referencia estándar para la LSE. La falta de un dataset unificado y ampliamente reconocido restringe la posibilidad de evaluar de manera objetiva el rendimiento de nuestro modelo en comparación con otros métodos existentes.

A pesar de las restricciones mencionadas, los resultados alcanzados demuestran que hemos cumplido con nuestro objetivo primordial: desarrollar un sistema de traducción neuronal básico que actúa como un precursor para el campo de estudio de la traducción de la LSE a lengua escrita. Este estudio sienta una base valiosa sobre la cual se podrán edificar modelos más sofisticados y precisos.

5.2. Lineas futuras

Para avanzar en este campo, es crucial centrarse en las siguientes áreas de mejora y desarrollo:

- **Creación de un Dataset de Referencia para la LSE:** La ausencia de un dataset robusto y ampliamente aceptado ha sido una barrera significativa en este trabajo. Futuras investigaciones deben priorizar la creación y estandarización de un dataset para la LSE que permita evaluar y comparar de manera justa los diferentes modelos y enfoques.
- **Mejoras en la Arquitectura del Modelo:** Aunque nuestro sistema ha demostrado ser funcional, existen múltiples oportunidades para optimizar la arquitectura de la red neuronal, incluyendo la exploración de modelos más avanzados como

Transformers o modelos híbridos que combinen diferentes enfoques de aprendizaje profundo.

- **Aumento del Tamaño y Diversidad del Dataset:** Además de crear un dataset de referencia, es esencial aumentar la diversidad y el tamaño del dataset utilizado. Esto incluiría videos con mayor variabilidad en términos de contexto, persona que los ejecuta, velocidad de la seña, y diferentes dialectos dentro de la LSE, lo que podría mejorar la robustez y generalización del modelo.
- **Evaluación con Usuarios Reales:** Finalmente, un paso crucial será evaluar el modelo en situaciones del mundo real, utilizando feedback de usuarios sordos para ajustar y mejorar la precisión y usabilidad del sistema. Esta evaluación permitirá comprender mejor las necesidades reales y adaptar el sistema para su uso práctico.

En conclusión, aunque hemos logrado un importante primer paso en la creación de un traductor neuronal para la LSE, queda un camino largo por recorrer para alcanzar un sistema que sea realmente efectivo y útil en aplicaciones del mundo real. La creación de un dataset de referencia y la continua mejora del modelo serán aspectos esenciales para futuros avances en este campo.

Bibliografía

[1] Encuesta de Discapacidad, Autonomía personal y situaciones de Dependencia (EDAD). Año 2008. Instituto nacional de estadística (INE). https://www.ine.es/prensa/np524.pdf

[2] Federación de Personas Sordas de la Comunidad de Madrid. 2019. https://www.fesorcam.org/historia-de-la-lse/

[3] Jarque, M. J. (2012). Las lenguas de signos: su estudio científico y reconocimiento legal. Anuari de Filologia. Estudis de Lingüística, 2, 33-48. https://dialnet.unirioja.es/descarga/articulo/4200049.pdf.

[4] Zeshan, U. (2005). Sign Languages of the World: A Comparative Handbook. De Gruyter Mouton.

[5] Sutton-Spence, R., & Woll, B. (1999). The Linguistics of British Sign Language: An Introduction. Cambridge University Press.

[6] Apuntes de lingüística de la Lengua de Signos Española. Fundación CNSE. https://dialnet.unirioja.es/servlet/libro?codigo=860590

[7] Nicholas, G., & Bhatia, A. (2023). Artificial Intelligence and Language Translation in Scientific Publishing. *Science Editor*. https://www.csescienceeditor.org/article/artificial-intelligence-and-language-translation-in-scientific-publishing/

[8] Bekhit, A. F. (2022). Introduction to Computer Vision. *Computer Vision and Augmented Reality in iOS*. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-7462-0_1

[9] Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., & Saenko, K. (2015). Sequence to Sequence - Video to Text. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December.

[10] Sutskever, I., Vinyals, O., & Le, Q., V. (2014, September 10). Sequence to Sequence Learning with Neural Networks. arXiv.org. https://arxiv.org/abs/1409.3215

[11] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[12] Machine health monitoring using local Feature-Based gated Recurrent unit networks. (2018, February 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/7997605

[13] Learning long-term dependencies with gradient descent is difficult. (1994, March 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/279181

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). Attention is all you need. arXiv.org. https://arxiv.org/abs/1706.03762

[15] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, January 16). Efficient estimation of word representations in vector space. arXiv.org. https://arxiv.org/abs/1301.3781

[16] Pennington, J., Socher, R., & Manning, C. (2014, October). GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543. Association for Computational Linguistics. https://aclanthology.org/D14-1162

[17] Goldberg, Y., & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1411.2738*. https://arxiv.org/abs/1411.2738

[18] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*. https://arxiv.org/abs/1301.3781

[19] Otani, A., Hashiguchi, R., Omi, K., Fukushima, N., & Tamaki, T. (2022). Performance evaluation of action recognition models on low quality videos. IEEE Access, 10, 94898–94907. https://doi.org/10.1109/access.2022.3204755

[20] Han, H., Shan, S., Chen, X., & Gao, W. (2013). A comparative study on illumination preprocessing in face recognition. Pattern Recognition, 46(6), 1691–1699. https://doi.org/10.1016/j.patcog.2012.11.022

[21] Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020, February 13). A simple framework for contrastive learning of visual representations. *arXiv.org*. Retrieved from https://arxiv.org/abs/2002.05709

[22] Srivastava, S., Gangwar, A., Mishra, R., & Singh, S. (2022). Sign language recognition system using TensorFlow Object Detection API. In *Communications in Computer and Information Science* (pp. 634–646). https://doi.org/10.1007/978-3-030-96040-748

[23] Cao, Z., Simon, T., Wei, S., & Sheikh, Y. (2016, November 24). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *arXiv.org*. https://arxiv.org/abs/1611.08050v2

[24] Newell, A., Huang, Z., & Deng, J. (2016, November 16). Associative Embedding: End-to-End learning for joint detection and grouping. *arXiv.org*. https://arxiv.org/abs/1611.05424v2

[25] Zimmermann, C., Ceylan, D., Yang, J., Russell, B. C., Argus, M., & Brox, T. (2019). *FreiHAND: A Dataset for Markerless Capture of Hand Pose and Shape from Single RGB Images*. Retrieved from https://lmb.informatik.uni-freiburg.de/resources/datasets/FreihandDataset.en.html

[26] Newell, A., Yang, K., & Deng, J. (2016, March 22). Stacked hourglass networks for human pose estimation. *arXiv.org*. https://arxiv.org/abs/1603.06937

[27] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 303-338. https://link.springer.com/article/10.1007/s11263-009-0275-4

[28] ISBI Challenge 2015, in *International Symposium on Biomedical Imaging*, 2015. [Online]. Available: https://biomedicalimaging.org/2015/program/isbi-challenges/

[29] Ronneberger, O., Fischer, P., & Brox, T. (2015, May 18). U-NET: Convolutional Networks for Biomedical Image Segmentation. *arXiv.org*. https://arxiv.org/abs/1505.04597

[30] Cai, Y., Wang, Z., Luo, Z., Yin, B., Du, A., Wang, H., Zhang, X., Zhou, X., Zhou, E., & Sun, J. (2020, March 9). Learning delicate local representations for Multi-Person pose estimation. *arXiv.org*. https://arxiv.org/abs/2003.04030v3

[31] Mao, W., Ge, Y., Shen, C., Tian, Z., Wang, X., Wang, Z., & Van Den Hengel, A. (2022, January 19). Poseur: Direct Human Pose Regression with Transformers. *arXiv.org*. https://arxiv.org/abs/2201.07412v2

[32] Google. "MediaPipe: Cross-platform, customizable ML solutions."[En línea]. Disponible en: https://ai.google.dev/edge/mediapipe/solutions/guide?hl=es-419.

[33] Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F., & Grundmann, M. (2020, June 17). BlazePose: On-device Real-time Body Pose tracking. *arXiv.org*. https://arxiv.org/abs/2006.10204

[34] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Lecture notes in computer science (pp. 21–37). https://doi.org/10.1007/978-3-319-46448-0_2

[35] Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL) (pp. 311–318). *Association for Computational Linguistics*. https://dl.acm.org/doi/10.3115/1073083.1073135.

[36] Lin, C. (2004, July 1). ROUGE: a package for automatic evaluation of summaries. *ACL Anthology*. https://aclanthology.org/W04-1013/.

[37] Hariri, R. H., Fredericks, E. M., & Bowers, K. M. (2019). Uncertainty in big data analytics: survey, opportunities, and challenges. *Journal of Big Data, 6(1)*. https://doi.org/10.1186/s40537-019-0206-3

[38] O. Koller et al., RWTH-PHOENIX-Weather 2014 T: Public Benchmark Dataset,.Available: https://www-i6.informatik.rwth-aachen.de/~koller/RWTH-PHOENIX-2014-T/.

[39] University of Science and Technology of China (USTC), ÇSL-Daily: Chinese Sign Language Corpus,.Available: http://home.ustc.edu.cn/~zhouh156/dataset/csl-daily/.

[40] Camgoz, Necati Cihan and Hadfield, Simon and Koller, Oscar and Ney, Hermann and Bowden, Richard (2018). Neural Sign Language Translation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://openaccess.thecvf.com/content_cvpr_2018/html/Camgoz_Neural_Sign_Language_CVPR_2018_paper.html

[41] Yin, K., & Read, J. (2020, April 1). Better Sign Language Translation with STMC-Transformer. *arXiv.org*. https://arxiv.org/abs/2004.00588v2

[42] Voskou, A., Panousis, K. P., Kosmopoulos, D., Metaxas, D. N., & Chatzis, S. (2021, September 1). Stochastic Transformer Networks with Linear Competing Units: Application to end-to-end SL Translation. *arXiv.org*. https://arxiv.org/abs/2109.13318v2

[43] Panousis, K. P., Chatzis, S., & Theodoridis, S. (2021, December 5). Stochastic local Winner-Takes-All networks enable profound adversarial robustness. *arXiv.org*. https://arxiv.org/abs/2112.02671

[44] Chen, Y., Zuo, R., Wei, F., Wu, Y., Liu, S., & Mak, B. (2022, November 2). Two-Stream network for sign language recognition and translation. *arXiv.org*. https://arxiv.org/abs/2211.01367v2

[45] Convolutional Two-Stream Network fusion for video action recognition. (2016, June 1). *IEEE Conference Publication | IEEE Xplore*. https://ieeexplore.ieee.org/document/7780582

[46] Zolfaghari, M., Oliveira, G. L., Sedaghat, N., & Brox, T. (2017, April 3). Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection. *arXiv.org*. https://arxiv.org/abs/1704.00616

[47] Feichtenhofer, C., Fan, H., Malik, J., & He, K. (2018, December 10). SlowFast networks for video recognition. *arXiv.org*. https://arxiv.org/abs/1812.03982

[48] Xie, S., Sun, C., Huang, J., Tu, Z., & Murphy, K. (2017, December 13). Rethinking spatiotemporal feature learning: Speed-Accuracy Trade-offs in video classification. *arXiv.org*. https://arxiv.org/abs/1712.04851

[49] Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification. *ACM*. https://doi.org/10.1145/1143844.1143891

[50] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan & Xinggang Wang.Deep High-Resolution Representation learning for visual recognition. (2021, October 1). *IEEE Journals & Magazine | IEEE Xplore*. https://ieeexplore.ieee.org/document/9052469

[51] Guan, M., Wang, Y., Ma, G., Liu, J., & Sun, M. (2024, May 9). Multi-Stream Keypoint Attention Network for sign language recognition and translation. *arXiv.org*. https://arxiv.org/abs/2405.05672v1

[52] Dactilología del Lenguaje de Signos Español (LSE). Kaggle. https://www.kaggle.com/datasets/rub3ntercero/lenguaje-de-signos-espaol

[53] Centro de Normalización Lingüística de la Lengua de Signos Española. Dilse (Diccionario de la lengua de sig-

nos española). Publicado en 2019. https://cnlse.es/es/recursos/otros/linguistica/materiales-lexicograficos/dilse-diccionario-de-la-lengua-de-signos-espanola

[54] Centro Aragonés para la Comunicación Aumentativa y Alternativa. Videos LSE. http://old.arasaac.org/videos_lse.php?pg=0&

[55] Selenium with Python. Baiju Muthukadan. https://selenium-python.readthedocs.io/

[56] Klein, G., Kim, Y., Deng, Y., Nguyen, V., Senellart, J., & Rush, A. M. (2018, May 28). OpenNMT: Neural Machine Translation Toolkit. *arXiv.org*. https://arxiv.org/abs/1805.11462v1

Anexo

.1. Código de Web scrapping

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Inicializar el navegador
driver = webdriver.Chrome()  # Asegúrate de tener el driver de Chrome descargado

# Variable para el número de páginas
num_paginas = 414  # Esto incluye la página 0 hasta la 413

# Iterar sobre todas las páginas
for pg in range(num_paginas):
    url = "http://old.arasaac.org/videos_lse.php?pg={}".format(pg)
    driver.get(url)

    try:
        # Esperar a que aparezca el botón "Seleccionar todos"
        select_all_button = WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.NAME, "boton_seleccionar_todos"))
        )

        # Hacer clic en "Seleccionar todos"
        select_all_button.click()

        # Esperar a que aparezca el botón "Añadir archivos seleccionados a mi se
        add_to_selection_button = WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.NAME, "boton_descargar"))
        )

        # Hacer clic en "Añadir archivos seleccionados a mi selección"
        add_to_selection_button.click()

        # Esperar un momento para que se procese la acción
        driver.implicitly_wait(2)
```

```
    except Exception as e:
        print("No se pudo realizar la acción en la página {}: {}".format(pg, e))

# Cerrar el navegador
# driver.quit()
```

.2. Código de extracción de keypoints

```
import cv2
import mediapipe as mp
import numpy as np
import os
import pandas as pd
from tqdm import tqdm

# Mediapipe setup
mp_holistic = mp.solutions.holistic
mp_drawing = mp.solutions.drawing_utils

# Directory setup
input_dir = './Diccionario'
output_dir = './DiccionarioKeypoints'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# Helper functions
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image.flags.writeable = False
    results = model.process(image)
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    return image, results

def draw_styled_landmarks(image, results):
    mp_drawing.draw_landmarks(
        image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
        mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
        mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2))
    mp_drawing.draw_landmarks(
        image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
        mp_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),
        mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
    mp_drawing.draw_landmarks(
        image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
        mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4)
        mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)

def extract_keypoints(results):
```

```
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.po
    lh = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.
    rh = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks
    return np.concatenate([pose, lh, rh])

# Inicialización de las columnas del DataFrame
num_pose_keypoints = 33 * 4  # 33 pose landmarks each with x, y, z, visibility
num_hand_keypoints = 21 * 3  # Each hand has 21 landmarks each with x, y, z

# Total keypoints per frame
total_keypoints = num_pose_keypoints + num_hand_keypoints * 2  # Two hands

# Create column names based on the keypoints data structure
columns = ([f'pose_{i}' for i in range(num_pose_keypoints)] +
           [f'left_hand_{i}' for i in range(num_hand_keypoints)] +
           [f'right_hand_{i}' for i in range(num_hand_keypoints)])

# Initialize DataFrame with column names
df_keypoints = pd.DataFrame(columns=columns)

# Video processing function
def process_video(video_path, output_path):
    cap = cv2.VideoCapture(video_path)
    frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    out = cv2.VideoWriter(output_path, cv2.VideoWriter_fourcc(*'mp4v'), 20, (fra

    with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confide
        frame_num = 0
        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break

            image, results = mediapipe_detection(frame, holistic)
            draw_styled_landmarks(image, results)
            out.write(image)

            keypoints = extract_keypoints(results)
            df_keypoints.loc[len(df_keypoints)] = keypoints

            frame_num += 1

    cap.release()
    out.release()
    cv2.destroyAllWindows()

# Process all videos
for video_file in tqdm(os.listdir(input_dir)):
```

```
    if video_file.endswith('.mp4'): #and (video_file not in os.listdir(output_di
        video_path = os.path.join(input_dir, video_file)
        output_path = os.path.join(output_dir, video_file)
        process_video(video_path, output_path)

# Save keypoints to CSV
df_keypoints.to_csv('./keypoints.csv')
```