**UNIVERSIDAD AUTONOMA DE MADRID**

**ESCUELA POLITECNICA SUPERIOR**

** **

**Máster en Big Data y Data Science: ciencia e ingeniería de datos**

**TRABAJO FIN DE MÁSTER**

**Interpretación de la lengua de signos mediante el procesamiento de imagen/vídeo y su exploración para aplicaciones en el mundo real**

**Javier Muñoz Haro**

**Tutor: Juan Carlos San Miguel Avedillo**

**Junio 2023**

**Interpretación de la lengua de signos mediante el procesamiento de imagen/vídeo y su exploración para aplicaciones en el mundo real**

**AUTOR: Javier Muñoz Haro**

**TUTOR: Juan Carlos San Miguel Avedillo**

**Escuela Politécnica Superior**

**Universidad Autónoma de Madrid**

**Junio de 2021**

**Resumen**

Todavía se debate en qué momento de la historia los seres humanos adquirimos la capacidad de comunicarse. El rango de tiempo es muy amplio, ya que existen [indicios](https://www.science.org/content/article/human-language-may-have-evolved-help-our-ancestors-make-tools#:~:text=Researchers%20have%20long%20debated%20when,than%202%20million%20years%20ago.) que datan de hace 2 millones y años y otros de hace solo 50.000 años. Si bien es cierto que la manera más efectiva de comunicación es la verbal, los humanos, al igual que otras especies, tenemos la capacidad de comunicarnos de manera no verbal. Gracias a esto, personas sin la capacidad de comunicación verbal, pueden transmitir mensajes los cuales pueden ser comprendidos por cualquier persona, sin la necesidad de signos lingüísticos orales.

Uno de estos ejemplos es el caso de las personas sordas, mudas o sordomudas. Dada esta discapacidad, utilizan el lenguaje de signos para poder comunicarse con otras personas. La motivación de este trabajo es poder acercar la lengua de signos a aquellas personas que no la conocen, usando el campo del Deep Learning.

El trabajo en el que ha consistido este TFM se basa en explorar las soluciones que existen actualmente para el reconocimiento de acciones y aplicarlas en el reconocimiento de los distintos gestos que componen la lengua de signos. Esto es un reto algo complicado, ya que no solo los modelos están limitados por la cantidad de signos que pueden reconocer, si no que existen muchas lenguas de signos, (tantas como países/comunidades existen en el mundo) por lo que hacer un modelo universal, capaz de entender cualquier signo en cualquier lengua se antoja más que complicado. También se ha de mencionar que los conjuntos de datos no son muy extensos, ni muy completos, lo cual ha dificultado mucho la tarea de crear un primer reconocedor de gestos.

Una vez que se recopilaron los datos, se han diseñado varios modelos para poder realizar las predicciones, y se han comparado los resultados para discernir cuales son los más efectivos, de los menos. Los distintos modelos están basados en arquitecturas como las ConvLSTM o 3D CNN, o se han apoyado en herramientas de predicciones de holísticas, como Mediapipe, que se han encargado de extraer las características más representativas de los signos para poder entrenar al modelo que realiza las predicciones.

Posteriormente a las comparaciones, se hará un estudio de precisión de las predicciones en distintos subconjuntos de gestos, para analizar cuál de los modelos generaliza mejor cuando el conjunto de signos a predecir es mayor.

Finalmente, y con el propósito de poder analizar si con la tecnología existente, se pueden crear herramientas que traduzcan de manera instantánea los signos, se harán pruebas que midan el tiempo de reacción de los modelos hasta que estos realizan la predicción, para evaluar si es factible su utilización como traductores en tiempo real.

***Agradecimientos***

*A mi familia por su paciencia, a mis amigos por su apoyo y a Sheila y a Celeste Prieto por introducirme al mundo de la lengua y el habla.*

**INDICE DE CONTENIDOS**

[1 Introducción 1](#_Toc175400053)

[1.1 Motivación 1](#_Toc175400054)

[1.2 Objetivos 1](#_Toc175400055)

[1.3 Organización de la memoria 1](#_Toc175400056)

[2 Estado del arte 3](#_Toc175400057)

[2.1 Subsección 3](#_Toc175400058)

[2.1.1 Subsubsección 3](#_Toc175400059)

[3 Diseño 5](#_Toc175400060)

[3.1 Subsección 5](#_Toc175400061)

[3.1.1 Subsubsección 5](#_Toc175400062)

[4 Desarrollo 7](#_Toc175400063)

[4.1 Subsección 7](#_Toc175400064)

[4.1.1 Subsubsección 7](#_Toc175400065)

[5 Integración, pruebas y resultados 9](#_Toc175400066)

[6 Conclusiones y trabajo futuro 9](#_Toc175400067)

[6.1 Conclusiones 9](#_Toc175400068)

[6.2 Trabajo futuro 9](#_Toc175400069)

[Referencias 11](#_Toc175400070)

[Glosario 13](#_Toc175400071)

[Anexos I](#_Toc175400072)

[Anexo A. Manual de instalación I](#_Toc175400073)

[Anexo B. Manual del programador III](#_Toc175400074)

Anexo [C. Anexo … V](#_Toc175400075)

**INDICE DE FIGURAS**

[Figura 2‑1: Logo EPS 3](#_Toc175399191)

**INDICE DE TABLAS**

# Introducción

## Motivación

La comunicación es esencial de los seres humanos. Es el método que tenemos para, como especie, transmitir información, ideas o datos. La comunicación, de manera genérica, consta de tres actores principales: emisor, que es el encargado de crear el mensaje y mostrar la intención de transmitirlo, el mensaje, que contiene la información a transmitir y el receptor, que es quien recibe la información contenida en el mensaje. Dentro de la comunicación, existen dos subconjuntos principales, la comunicación verbal y la comunicación no verbal. La comunicación verbal usa signos lingüísticos en su mensaje y tiene dos maneras de transmitirse: de manera oral y de manera escrita. La conjunción de ambas es lo que se conoce como “habla”. Por otro lado, la comunicación “no verbal” es precisamente el tipo de comunicación que se realiza sin hacer uso del habla. Algunos ejemplos de esto son el contacto visual, expresiones faciales o gestos.

La primera forma en la que el ser humano conoce la comunicación verbal es mediante el sentido auditivo. Cuando cognitivamente este se desarrolla, se ha de ver expuesto a alguna lengua para poder aprenderla, pero si el sentido auditivo no está desarrollado o se tiene cualquier tipo de discapacidad asociada que impida la escucha, será muy complicado aprender a comunicarse de manera verbal. Precisamente este impedimento de aprender de una manera oral cualquier lengua está asociado a las dificultades en la comunicación que pueden padecer las personas sordas, mudas y sordomudas. Se ha de aclarar que no todas las personas que sufren estos tipos de discapacidad son incapaces de comunicarse de manera oral, pero, aquellas que padecen de sordera prelocutiva (aquellas que perdieron la capacidad auditiva de manera total antes de la adquisición o aprendizaje del lenguaje) tendrán mucha más dificultad a la hora de comunicarse oralmente que aquellas que podían escuchar antes de desarrollar la discapacidad. De esta incapacidad para comunicarse de manera verbal, nace la lengua de signos.

El origen de la lengua de signos varía enormemente en función del país o región donde nos encontremos. En España, por ejemplo, data del siglo XVI, cuando los monjes en los monasterios estaban obligados a guardar silencio. Este impedimento para comunicarse de manera verbal impulsó que empezasen a comunicarse con signos manuales. Estaban, sin saberlo, comenzando a desarrollar la Lengua de Signos Española (LSE). Es interesante saber que, pese a que en España se tiene una lengua oficial principal, el castellano, se tienen una variedad lenguas oficiales como el catalán, valenciano, gallego o euskera. Pues bien, para cada una de estas lenguas, existe una lengua de signos asociada, como la Lengua de Signos Catalana (LSC) o la lengua de signos valenciana (LSCV). Hasta la comunidad de Andalucía, cuya lengua principal es el castellano, tiene variantes con respecto a la LSE. Esto es aplicable al resto de países, donde, sin ir más lejos, la lengua de signos de Gran Bretaña, Estados Unidos y Australia es distinta, pese a que todos estos países tiene como lengua oficial principal el inglés.

Esto por tanto nos introduce la primera dificultad a la hora de poder crear un traductor de lengua de señas, y es que no existe una lengua de señas universal, por lo que deberíamos tener en cuenta en qué lengua de signos se está comunicando el emisor del mensaje.

Por otro lado, existe otra dificultad, y es la gran cantidad de gestos que contiene cada una de las lenguas de signos. En sematos(ref https://www.sematos.eu/), el portal europeo de lengua de signos, se tienen registrados un total de 6076 palabras para LSE, 3605 en Lengua de Signos Francesa (LSF) y 297 en LSC. Como se puede ver, solo en estas 3 lenguas, se tienen un total de 9978 gestos diferentes a identificar en 3 lenguas distintas lo cual haría de esto un problema multiclase multietiqueta muy complejo.

La motivación, por tanto, de este trabajo será comprobar si hoy en día existen técnicas y diseños que permitan la viabilidad de, en un futuro, poder tener un traductor de lengua de signos universal que permita traducir cualquier mensaje de lengua de signos de cualquier región del planeta al mismo mensaje en otra lengua o su traducción a lenguaje verbal, ya sea de manera oral o escrita. El estudio de la instantaneidad a la hora de realizar las predicciones será otro factor importante, ya que un intérprete de lengua de signos es capaz de traducir de lenguaje verbal a no verbal en cuestión de fracciones de segundo, por lo que, si queremos que se desarrolle un traductor de lengua de signos a

Referencia [1]

## Objetivos

El objetivo principal es crear un clasificador que realice reconocimiento de acciones para predecir los gestos del lenguaje de signos. El reconocimiento correcto de signos se comprobará de dos formas:

* Reconocimiento de videos que el algoritmo no ha visto hasta el momento.
* Reconocimiento de acciones en tiempo real mediante OpenCV

El motivo por el que comprobaremos el correcto funcionamiento del modelo de estas dos formas es debido a que las diferencias entre clasificar un vídeo del conjunto de test no es lo mismo que realizar predicciones en tiempo real, esto se explicará en profundidad más adelante.

Para cumplir estos objetivos, se necesitará principalmente comprender cuales son los algoritmos que mejor funcionan a la hora de reconocer acciones, como preprocesar los datos para que dicho algoritmo aprenda y finalmente evaluar correctamente los resultados obtenidos, sean estos positivos o negativo

## Organización de la memoria

La memoria consta de los siguientes capítulos:

* **Estudio del estado del arte:** El reconocimiento de acciones y el reconocimiento de lengua de señas son campos explorados dentro del mundo del Deep Learning. En esta primera sección explicaremos cuales son los métodos y algoritmos más efectivos encontrados hasta la fecha en esta disciplina, y como estos se van a utilizar como punto de apoyo para resolver el problema.
* **Diseños de la solución:** Para poder elaborar una solución que funcione debidamente, se han implementado varias soluciones para aplicarlas a nuestro problema. Explicaremos en profundidad los fundamentos de dichas soluciones y como se han ajustado para poder resolver el problema.
* **Desarrollo**: Una vez que se han explorado distintas soluciones para nuestro problema, comenzaremos con la explicación de cómo se implementaron las mismas. Se explicarán los inconvenientes encontrados y como estos se subsanaron.
* **Integración, pruebas y resultados**: Ya con los distintos diseños implementados, procedemos a probar como estos funcionan cuando se ven ante nuevos datos. Hay que recordar que, para que un modelo funcione correctamente, este no solo tiene que ser óptimo a la hora de reconocer los datos que se han usado para entrenarlo, si no que, cuando se vea expuesto a datos que nunca ha visto, sea capaz de inferir apropiadamente el resultado correcto. Durante la elaboración de este TFM, esta ha sido la fase donde más tiempo se ha pasado, ya que se han realizado diversas pruebas para poder verificar cual de todas es la solución más eficiente, en base a los objetivos marcados.
* **Conclusiones y trabajo futuro**: Una vez visto cómo se comportan los distintos modelos desarrollados, los compararemos entre ellos y comprobaremos cuales son las fortalezas y flaquezas de cada uno de ellos. Posteriormente, se explorarán nuevas vías de desarrollo para poder mejorar los resultados o ampliar la funcionalidad de los modelos para poder resolver problemas más complejos.

# Estado del arte

## Reconocimiento de acciones

El reconocimiento de acciones (*action recognition*) es un campo dentro del Deep Learning que se encarga automatizar la clasificación de distintas acciones que realizan (usualmente) los humanos. Normalmente, para poder resolver este tipo de problemas, se hace uso de datos basados en vídeos, donde dichas acciones a reconocer se ven representadas por un sujeto que las realiza. Sus aplicaciones son variadas, desde sistemas de vigilancia, monitorización para la medicina o sistemas autónomos de conducción. Actualmente existen varios modelos que se han desarrollado para poder realizar reconocimiento de acciones, como redes neuronales convolucionales o redes neuronales recurrentes y sus potenciales combinaciones.

Dentro del campo de reconocimiento de lengua de signos, existen diversos modelos que se están utilizando para poder atajar este problema. Los más implementados son las redes neuronales convolucionales 3D (3D-CNNs)(ref), las redes convolucionales basadas en grafos (GCN)(ref) o las redes convolucionales LSTM(ref) (ConvLSTM)(ref) entre otros.

### Estimadores de poses

Actualmente existen varias librerías o frameworks que realizan el trabajo de estimación de poses por nosotros. Quizás los más conocidos sean Openpose(ref) y Mediapipe(ref).

Openpose, es un sistema de visión por computador que realiza estimaciones de poses humanas en tiempo real. Para ello utiliza una red neuronal convolucional profunda, que analiza la imagen estimando primero las personas que aparecen el frame y posteriormente infiriendo los puntos clave o “keypoints” de la pose de los sujetos, este tipo de inferencia es conocida como bottom-up. Es una herramienta donde la velocidad de inferencia es uno de sus puntos clave, ya que es capaz de detectar poses en fracciones de segundo para frames donde existen varios sujetos, cosa que, hasta la fecha, no se había conseguido, ya que las redes basadas en “bottom-up” tardaban hasta minutos en analizar un único frame y extraer las poses.

Imagen que contiene parado, sostener, posando, mujer

Descripción generada automáticamente

(Figura en el índice más referencia: http://naoki.io/portfolio/person\_descrip.html)

Mediapipe también es un sistema de visión por computador, desarrollado por el equipo de Google Research para la estimación de poses basado en landmarks. Un landmark es un punto en el espacio que representa una característica del sujeto al que se le está extrayendo la pose para un frame dado. Mediapipe se divide en distintos estimadores de landmarks, ya que no solo detecta las poses corporales, si no que también estima landmarks en las facciones de la faciales de los sujetos entre otras, dando información muy precisa de los movimientos de los sujetos. Adicionalmente, Mediapipe tiene una API donde se pueden extraer esos puntos (landmarks). Esto es de especial conveniencia, debido a que, podemos utilizar esos landmarks como un vector de características que representa poses espaciotemporales para pasárselo a una red basada en aprendizaje profundo y que aprenda patrones en las poses para poder clasificarlas.

Imagen que contiene interior, persona, sostener, vistiendo

Descripción generada automáticamente

(Añadir al índice de figuras y referencia: <https://github.com/AshishPandey88/Pose-Detection>)

Mediapipe, en su versión holística, evalúa al sujeto como un “todo” analizando no solo las poses, si no todo el conjunto características extraíbles del sujeto, como las expresiones faciales o los dedos y las palmas de las manos, características muy útiles para la detección de lengua de signos. Estas características, como se ha explicado previamente, pueden ser extraídas como un vector.

Por otro lado, Mediapipe ofrece dos versiones a la hora de realizar inferencias. La primera está basada en CUDA(ref) y TensorRT(ref), librerías propietarias de NVIDIA que facilitan las operaciones matriciales y la velocidad en la inferencia de modelos. La segunda, basada en tensorflow para CPU y modelos ligeros (lite) que permite usar mediapipe en cualquier tipo de dispositivo que tenga un intérprete de Python con Tensorflow instalado en él (Ordenadores sin GPU dedicada, Raspberry Pi, teléfonos Android… etc.)

Estas utilidades hicieron que finalmente se utilizase Mediapipe a la hora de implementar uno de los varios modelos propuestos en este trabajo.

### Redes Convolucionales 3D (3D-CNNs)

Este modelo es una extensión de las redes convolucionales 2D. Una red convolucional 2D es un modelo es capaz de capturar características espaciales. Esta característica se debe principalmente a la operación principal que realiza el modelo, la convolución.

En un espacio continuo, una convolución se define de la siguiente manera:

Texto

Descripción generada automáticamente

(Añadir al índice de figuras y mencionar a Deep Learning Book puesto que esta fórmula es extraída de la sección 9.1 The convolution Operation”)

Matemáticamente, una convolución es una operación matemática que realiza una transformación de dos funciones (*f* y *g*) en una función que representa una superposición de las características de ambas. Esta característica es de especial funcionalidad debido a que, en el tratamiento de señales, es capaz de combinar dos señales mezclando las características de ambas en una sola.

Los colores de las imágenes o “frames” son codificados en 3 canales rojo, verde y azul (RGB). Estos tienen, por lo general 256 niveles (8 bits), siendo el 0 el nivel más bajo de intensidad de color y 255 el máximo. Si queremos realizar cualquier tipo de transformación sobre dichos colores, necesitaremos otra señal o función. Dentro del mundo de tratamiento de imágenes, esa “señal” adicional será lo que se denomina un kernel o filtro. Esto hace que podamos aplicar la función vista previamente al espacio discreto:



(Añadir al índice de figuras y mencionar a Deep Learning Book puesto que esta fórmula es extraída de la sección 9.1 The convolution Operation”)

Un kernel o filtro, puede ser considerado una matriz de números cuya función es aplicarse en un subconjunto de pixeles que forman un frame o imagen para poder extraer características del mismo. Una de las aplicaciones más generalizadas es la de extraer los “edges” o bordes de una imagen, como se puede ver en la siguiente imagen:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

(Añadir al índice de figuras y mencionar a 3Blue1Brown, dado que la imagen es de su video “What is a Convolution?”)

En este caso, la convolución pasaría por multiplicar coordenada a coordenada los valores de cada uno de los pixeles a los valores del kernel y luego sumar cada uno de esos valores de la multiplicación. Si los píxeles a los que aplican el kernel tienen todos los mismos valores, el valor resultante será neutro (0) ya que la primera columna del kernel son todo valores negativos y la tercera todos positivos. En cambio, si los colores son diferentes (bordes de una superficie en la imagen), los valores obtenidos de la convolución serán diferentes de 0, por lo que el valor será o positivo o negativo en el caso que el píxel analizado forme parte de un borde vertical. Si se quisiesen detectar los bordes horizontales se deberá de aplicar una transposición al kernel, ya que los valores distintos de 0 se darían precisamente en aquellos bordes que marcan las líneas horizontales. Una vez extraídas las características, estas servirán a la red para aprender los distintos patrones asociados a las predicciones que nuestro modelo realizará.

Una convolución 3D, realiza el mismo proceso, pero añadiendo una dimensión adicional, que puede ser o bien la profundidad de la imagen captura como es el caso de (ref a paper de action recognition UAH), ya que permite extraer características de entornos más complejos, o puede ser aplicado para vídeos, añadiendo esa dimensión extra (temporal) y así poder capturar las dependencias temporales.

Imagen que contiene Diagrama

Descripción generada automáticamente

(ref <https://www.tensorflow.org/tutorials/video/video_classification> y Figura)

Como se puede ver en la siguiente figura, a la izquierda tenemos una serie de matrices, donde cada matriz, aplicado a nuestro problema, puede ser un frame en un instante de tiempo t, y el conjunto de matrices, el vídeo en sí. Los kernels, o filtros, tendrán ahora una dimensión extra, para poder capturar las dependencias temporales y extraer, no solo las características de un frame si no de varios a la vez, siendo estos secuenciales.

### Redes neuronales recurrentes: LSTM

Las redes neuronales recurrentes (RNN) son un tipo de redes neuronales diseñadas para manejar datos secuenciales. Estas redes tienen “memoria”, ya que la salida de un cálculo en un instante t, afecta a la entrada del cálculo de la salida para t+1, pudiendo así procesar secuencias de datos a lo largo del tiempo. A esta memoria, se le suele denominar el estado oculto de la red, y es un vector que guarda información de los estados anteriores de la red para poder tenerlo en cuenta en futuras iteraciones. Una red neuronal tiene la siguiente forma:

Diagrama

Descripción generada automáticamenteAplicación

Descripción generada automáticamente con confianza baja

Izda (Ref<https://datascientest.com/es/wp-content/uploads/sites/7/2021/07/Sans-titre-1-Recupere-08.png>) Dcha(referencia <https://datascientest.com/es/recurrent-neural-network-rnn-de-que-se-trata>) y meter como figura

Donde vemos que la red tiene una única entrada *Xt* que hace referencia a la entrada de los datos secuenciales *X* en un instante de tiempo *t*. En dicha célula se realizan las operaciones pertinentes, que dependerán del tipo de algoritmo que se implemente dentro de estas, detalle que veremos más adelante. La salida de la célula, *t*, será por tanto el resultado de las computaciones internas de la célula, junto con el estado oculto que se va actualizando en cada instante de tiempo. El valor resultado en el instante t será la predicción del valor de *Yt+1*

Si desenrollásemos la recursividad, tendríamos un esquema similar al de la (figura de la derecha), donde vemos que cada célula

### Subsubsección

### Subsubsección

****

Figura 2‑1: Logo EPS

# Diseño

## Subsección

### Subsubsección

# Desarrollo

## Subsección

### Subsubsección

# Integración, pruebas y resultados

# Conclusiones y trabajo futuro

## Conclusiones

## Trabajo futuro

# Referencias

1. En las referencias figurarán los autores (opcionalmebte los editors), el título del artículo, el nombre de la revista o libro, el volumen y número de la revista, las páginas del artículo, la fecha de edición,. A continuación se listan algunos ejemplos
2. K.N. Platanioitis, C.S. Regazzoni (eds.), “Special Issue in Visual-centric Surveillance Networks and Services”, IEEE Signal Processing Magazine, 22(2), Marzo 2005.
3. B.S. Manjunath, P. Salembier, T. Sikora (eds.), “Introduction to MPEG 7: Multimedia Content Description Language,”, John Wiley and Sons, 2002
4. G. R. Bradski, “Computer vision face tracking as a component of a perceptual user interface,” en Proc.IEEE Workshop on Applications of Computer Vision, Princeton, NJ, October 1998, pp. 214–219.
5. A. D. Bue, D. Comaniciu, V. Ramesh, and C. Regazzoni, “Smart cameras with real-time video object generation,” in Proc. IEEE Intl. Conf. on Image Processing, Rochester, NY, volume III, 2002, pp. 429–432.
6. P. Anandan. “A computacional cuadrowork and an algorithm for the measurement of visual motion”, International Journal of Computer Vision, 2(3):283-310, January, 1989.
7. W.J. Ruckelidge. “Efficient Computation of the minimum Hausdorff Distance for Visual Recognition”, Phd thesis, Cornell Universitym 1995. CS-TR1454
8. “Extensible Markup Language (XML) 1.0 (Second Edition)”, W3C Recommendation 6 October 2000 <http://www.w3.org/TR/REC-xml>
9. William H. Press, Saul A.Teukolsky, William T. Vetterling, Brian P. Flannery. “Numerical Recipes in C – The art of Scientific Computing 2nd Edition”. Cambridge University Press