Ciencia Latina Revista Científica Multidisciplinar, Ciudad de México, México.
ISSN 2707-2207 / ISSN 2707-2215 (en línea), septiembre-octubre 2024,
Volumen 8, Número 5.

https://doi.org/10.37811/cl_rcm.v8i5

RECONOCIMIENTO DE LA LENGUA DE SEÑAS MEXICANA USANDO DEEP LEARNING MEDIANTE UNA APLICACIÓN MÓVIL

MEXICAN SIGN LANGUAGE RECOGNITION USING DEEP LEARNING THROUGH A MOBILE APP

Gabriel Alejandro Salgado Martínez
Universidad Autónoma de Guerrero, México

René Edmundo Cuevas Valencia
Universidad Autónoma de Guerrero, México

Angelino Feliciano Morales
Universidad Autónoma de Guerrero, México

Arnulfo Catalán Villegas
Universidad Autónoma de Guerrero, México

DOI: https://doi.org/10.37811/cl_rcm.v8i5.14458

Reconocimiento de la Lengua de Señas Mexicana Usando Deep Learning Mediante una Aplicación Móvil

Gabriel Alejandro Salgado Martínez[1]
06316477@uagro.mx
https://orcid.org/0009-0004-8614-5469
Universidad Autónoma de Guerrero
México

René Edmundo Cuevas Valencia
renecuevas@uagro.mx
https://orcid.org/0000-0001-9528-7603
Universidad Autónoma de Guerrero
México

Angelino Feliciano Morales
afmorales@uagro.mx
https://orcid.org/0000-0002-7707-7319
Universidad Autónoma de Guerrero
México

Arnulfo Catalán Villegas
03180@uagro.mx
https://orcid.org/0009-0001-0391-7960
Universidad Autónoma de Guerrero
México

RESUMEN

El reconocimiento de la Lengua de Señas Mexicana (LSM) mediante algoritmos de Machine Learning (ML) ha sido mayormente estudiado en computadoras personales, habiendo menos trabajos enfocados en plataformas móviles. Este estudio tuvo como objetivo evaluar el reconocimiento de señas fijas y con movimiento de LSM mediante una Aplicación móvil, para las señas fijas se entrenó una red neuronal (RN) del tipo Perceptrón Multi Capa (en inglés Multi Layer Perceptron, MLP) con 137,764 imágenes. Para reconocer señas con movimiento, se entrenaron tres RN: Memoria Larga a Corto Plazo (en inglés Long Short Term Memory, LSTM), Unidad Recurrente Cerrada (en inglés Gated Recurrent Unit, GRU) y Transformer, compuesto con una Red Neuronal Convolucional (en inglés Convolutional Neural Network, CNN) y una LSTM, con datos de 1600 videos de 16 señas. En el entrenamiento el modelo MLP logró un 96.2% de Accuracy, mientras que el modelo Transformer tuvo el mejor desempeño en señas con movimiento con 94.16% de Accuracy. En la Aplicación, el reconocimiento de señas fijas fue más preciso a 50 cm de distancia, con tiempo de inferencia de 0.60 milisegundos y un uso de memoria de 116 bytes por fotograma. Las señas con movimiento de ambas manos lograron una precisión superior al 93%, con tiempo de inferencia de 286 milisegundos y un uso máximo de 1 Mb de memoria. Estos hallazgos pueden aportar información potencial para el desarrollo de una herramienta de traducción de la LSM, que puede impactar significativamente en la vida de las personas con discapacidad auditiva.

***Palabras Clave*:** aplicación móvil, deep learning, lengua de señas

[1] Autor principal.
Correspondencia: 06316477@uagro.mx

Mexican Sign Language Recognition Using Deep Learning Through a Mobile App

ABSTRACT

Mexican Sign Language (MSL) recognition using Machine Learning (ML) algorithms has been mostly studied on personal computers, with fewer studies focused on mobile platforms. This study aimed to evaluate the recognition of fixed and moving signs of MSL using a mobile application. For fixed signs, a neural network (NN) of the MLP (Multi Layer Perceptron) type was trained with 137,764 images. To recognize signs with movement, three NNs were trained: LSTM (Long Short Term Memory), GRU (Gated Recurrent Unit) and Transformer, composed of a CNN (Convolutional Neural Network) and an LSTM, with data from 1600 videos of 16 signs. In the training, the MLP model achieved 96.2% Accuracy, while the Transformer model had the best performance in signs with movement with 94.16% Accuracy. In the App, recognition of stationary signs was most accurate at 50 cm distance, with an inference time of 0.60 milliseconds and a memory usage of 116 bytes per frame. Signs with movement of both hands achieved an accuracy of over 93%, with an inference time of 286 milliseconds and a maximum memory usage of 1 Mb. These findings may provide potential information for the development of an LSM translation tool, which can significantly impact the lives of people with hearing impairments.

Keywords: mobile application, deep learning, sign language

Artículo recibido 15 septiembre 2024

Aceptado para publicación: 28 octubre 2024

INTRODUCCIÓN

La comunidad de personas sordas en México es de 1,350,802 personas, según datos del censo del 2020 del Instituto Nacional de Estadística y Geografía (INEGI, 2020), de las que alrededor de 250,000 utilizan la Lengua de Señas Mexicana (LSM) para comunicarse. A pesar de que la LSM fue reconocida de manera oficial desde 2005 (Ley General de las Personas con Discapacidad, 2005), la documentación y el estudio sobre la gramática de este tipo de lengua resulta todavía escaso (Cruz, 2014).

Las personas con discapacidad auditiva de nacimiento, o adquirida posteriormente, enfrentan una limitación significativa para lograr una plena integración social (Gobierno de México, 2016). La barrera de comunicación que enfrenta este sector vulnerable de la sociedad dificulta sus oportunidades de inclusión laboral y educativa (Pérez y Cruz, 2020).

El problema de comunicación entre hablantes y no hablantes de lengua de señas ha sido abordado en numerosos trabajos de investigación, enfocados en el desarrollo de sistemas de reconocimiento de señas gestuales mediante algoritmos de ML capaces de identificar movimientos de los brazos, manos y rostro. Sin embargo, la gran mayoría de estos sistemas fueron desarrollados para su funcionamiento en computadoras personales, ya que su implementación en plataformas móviles enfrenta desafíos significativos debido a las limitaciones en potencia computacional y energía que implican ejecución de modelos de ML en estos dispositivos (Martinez et al., 2022).

En este trabajo, se presenta el reconocimiento de señas de la LSM mediante una aplicación móvil, utilizando la librería MediaPipe Hands para el reconocimiento de la trayectoria de las manos en secuencias cortas de videos y analizando estas trayectorias en modelos de ML, basados en Redes Neuronales Artificiales, previamente entrenados y cargados en la Aplicación para realizar la clasificación de forma local en el dispositivo móvil. Para ello, se entrenaron varias arquitecturas de modelos de Redes Neuronales Artificiales para probar su rendimiento desde la aplicación, así como también se midió el rendimiento General de la Aplicación en el teléfono.

El aporte principal de esta investigación es mostrar el rendimiento de una aplicación móvil que implementa modelos de Deep Learning en un smartphone de gama media, para reconocer y traducir señas de la LSM a su equivalente en español, con la finalidad de servir como una herramienta de traducción entre hablantes y no hablantes de esta lengua de señas.

Estado del arte

Como se mencionó anteriormente, la barrera de comunicación entre hablantes y no hablantes de las diferentes lenguas de señas ha sido ampliamente estudiada, con múltiples investigaciones enfocadas en desarrollar sistemas computacionales capaces de reconocer señas gestuales y traducirlas al idioma oral y escrito, esto mediante algoritmos de ML con la capacidad de interpretar movimientos del cuerpo, como los de los brazos, manos, y rostro.

En particular, dentro de los trabajos dedicados al reconocimiento de la Lengua de Señas Mexicana (LSM), destaca el estudio de Espejel (2021), quien en su tesis doctoral estudió la traducción automática de señas que corresponden al alfabeto dactilológico, los números del 0 al 100, y 249 palabras, de la LSM. El reconocimiento de las señas se realizó a partir de la selección de las regiones de interés, como manos y rostro cara en imágenes y videos, de las cuales se extrajeron características geométricas que se emplean en diferentes clasificadores: Bayesiano, Árboles de decisión, Máquina de Vectores de Soporte (en inglés Support Vector Machine, SPV) y Redes Neuronales, en cada uno el porcentaje de precisión medido fue del 93.7%, 77.2%, 96.2% y 64.9%.

Otro ejemplo relevante es el trabajo de maestría de Mejía (2022), quien estudió el reconocimiento de las señas basado en la detección de puntos característicos de las manos, cuerpo y rostro. Para ello utilizó una cámara OAK-D y la librería Mediapipe para capturar coordenadas 3D de 4 señas estáticas y 26 dinámicas. Con los datos obtenidos, evaluó tres modelos de redes neuronales: Red Neuronal Recurrente (en inglés Recurrent Neural Network, RNN), Memoria Larga a Corto Plazo (en inglés Long Short Term Memory, LSTM) y Unidad Recurrente Cerrada (en inglés Gated Recurrent Unit, GRU). El modelo GRU mostró el mejor rendimiento, alcanzando una precisión del 97%.

Por otro lado, respecto a trabajos enfocados en el desarrollo de aplicaciones móviles para la traducción de lengua de señas mediante modelos de ML, se encontró que algunos de los proyectos son sistemas del tipo cliente servidor, donde se realiza la captura de una imagen mediante la cámara del dispositivo y luego se envía a un sistema en línea para ser procesada.

Por ejemplo, Pinzón & Sanabria (2021), en su trabajo de tesis para obtener el título de licenciatura propusieron el desarrollo de una App para el reconocimiento de cinco señas estáticas de la Lengua de Señas Colombiana a partir imágenes tomadas desde la cámara o de la galería del dispositivo. La

aplicación desarrollada es del tipo cliente – servidor, ya que consiste en una captura de una imagen de una seña y se envía a una aplicación web para ser analizada y devolver el resultado, por lo que necesita una conexión a internet para el funcionamiento de la aplicación. La app se desarrolló utilizando el framework Flutter y se utilizó la librería MediaPipe para realizar el proceso de reconocimiento de las manos.

Otro ejemplo es el trabajo de Jin et al. (2016), en donde propusieron un sistema para el reconocimiento de imágenes utilizando el algoritmo, SPV para clasificar un conjunto de datos de imágenes de gestos. El modelo se implementó sobre una aplicación web en un teléfono inteligente, los resultados mostraron que fue capaz de reconocer y traducir 16 gestos diferentes del lenguaje de señas americana con una precisión general del 97.13%.

Por otro lado, Gallego (2021), en su trabajo de tesis de licenciatura muestra el desarrollo de una app para iOS capaz de identificar en tiempo real 24 letras estáticas de la Lengua de Señas Americana (en inglés American Sign Language, ASL). La app se realizó en Swift, utiliza un modelo de ML del tipo MLP para las predicciones, que consiste en una red neuronal entrenada en Python y Keras. El modelo de Deep Learning consiguió un 94.7% en el test de validación y en el reconocimiento de las señas a través de la aplicación obtuvo resultados variados entre el 50% y el 100% en algunos casos.

En el trabajo de tesis de licenciatura de Aquino (2018) se desarrolló en Python un sistema de reconocimiento e interpretación de 27 señas estáticas del alfabeto en lengua de señas boliviano mediante una CNN desarrollada en TensorFlow apoyada por la librería OpenCV. Posteriormente se optimizó este modelo de red neuronal para ser implementado en una aplicación para dispositivos móviles con sistema operativo Android. La red neuronal obtuvo una exactitud final de 77.6%, utilizando un conjunto de pruebas conformado por 144000 imágenes, y el porcentaje de reconocimiento de la seña a en la aplicación fue del 93%.

En la Tabla 1 se muestra un resumen de los principales trabajos de investigación encontrados que se enfocaron en el desarrollo de una aplicación móvil para el reconocimiento y traducción de lenguas de señas, en cada trabajo se describe la plataforma objetivo de la aplicación, la lengua de señas que traduce, la cantidad de señas que puede reconocer, el tipo de señas, el modelo de ML utilizado para la clasificación de las señas reconocidas y por último el porcentaje de reconocimiento alcanzado.

Tabla 1. Resumen de los trabajos de investigación enfocados en el desarrollo de aplicaciones móviles para el reconocimiento de lenguas de señas.

Referencia	Sistema operativo	Lengua de señas	Cantidad de señas reconocidas	Tipo de seña (Fijas/ Dinámicas)	Clasificador	Porcentaje de reconocimiento
(Berea, 2021).	Multiplataforma	ASL	24	Fijas	CNN	99%
(Gallego, 2021)	iOS	ASL	24	Fijas	MLP	94.7%
(Pinzón y Sanabria, 2021).	Multiplataforma	LSC	5	Fijas	No se indica	No se indica
(Aquino, 2018)	Android	LSB	27	Fijas	CNN	93%
(Luna y Minajas, 2016)	Android	LSM	No se indica	Fijas	No se indica	No se indica
(Jin, Omar y Hisham, 2016)	Windows Phone	ASL	16	Fijas	SPV	97.13%

Fuente: elaboración propia

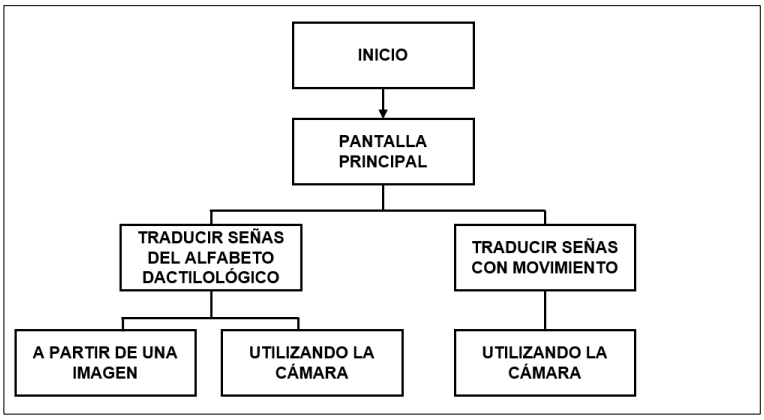
MATERIALES Y MÉTODOS

Metodología de la Aplicación Móvil

En la LSM se puede identificar al grupo de señas que corresponden al alfabeto dactilológico, estas señas se utilizan para formar palabras deletreadas, representando cada letra con una seña única. El alfabeto dactilológico se emplea para expresar palabras que no cuentan con una seña específica, como tecnicismos, o nombres propios (Cruz, 2008). Las señas del alfabeto dactilológico son señas fijas que se realizan con determinada configuración de la mano en una posición estática a la altura de los hombros, y no requieren de algún otro movimiento para su representación. Para el resto de señas que componen a la LSM se requieren de movimientos de los brazos con determinadas configuraciones de las manos, y en algunos casos se emplean también gestos faciales y movimientos del cuerpo.

De acuerdo a lo anterior, se propuso que la Aplicación móvil pueda reconocer señas fijas del alfabeto dactilológico y señas con movimiento de la LSM. En la Figura 1 se muestra el Diagrama General de funcionamiento propuesto para la Aplicación.

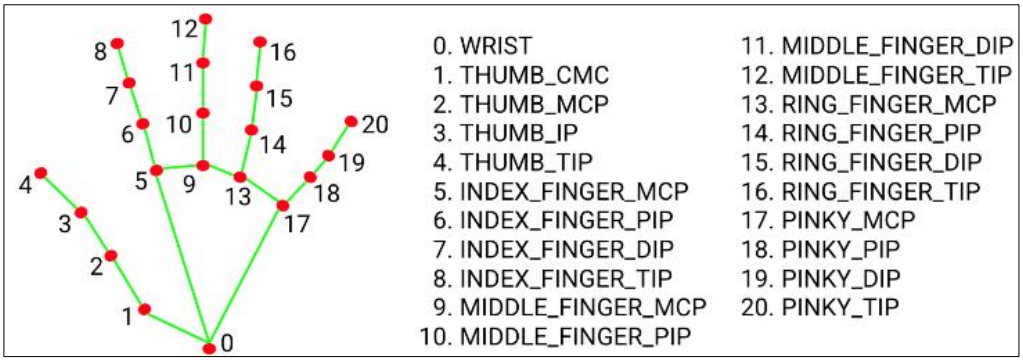
Figura 1. Diagrama General de funcionamiento de la Aplicación Móvil



Elaboración propia

Para desarrollar la Aplicación se utilizó el proyecto de código abierto de MediaPipe Hands para Android (MediaPipe, 2024), el cual implementa modelos de ML previamente entrenados para realizar detección y seguimiento de manos en imágenes y videos almacenados en el dispositivo móvil, así como en video en tiempo real. La solución MediaPipe Hands infiere 21 puntos de referencia 3D de una mano, los cuales se ubican en coordenadas (X, Y, Z) respecto a las dimensiones del fotograma procesado, tal como se muestra en la Figura 2.

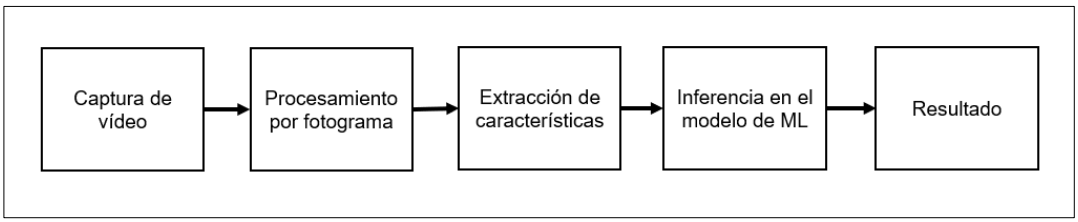
Figura 2. Puntos del Landmark detectado por la librería MediaPipe Hands (MediaPipe, 2024).



Para el reconocimiento de señas del Alfabeto Dactilológico utilizando la cámara del dispositivo, se modificó el proyecto de Android de MediaPipe de reconocimiento de manos en video en tiempo real. Por cada fotograma del video capturado se almacenan en una lista las coordenadas de los 21 puntos reconocidos de la mano señante, posteriormente la lista se convierte a un Tensor Buffer para poder ser comparado en el modelo de ML, entrenado en esta investigación, para el reconocimiento de 21 clases de señas fijas. Del resultado de la predicción se toma la clase de mayor porcentaje y se muestra en pantalla la seña un texto con la seña correspondiente, junto a la posición de la mano efectuando la seña,

esto en el mismo fotograma procesado, con lo que se logra un reconocimiento de señas fijas en tiempo real. En la Figura 3 se muestra el proceso de principal de reconocimiento de las señas fijas.

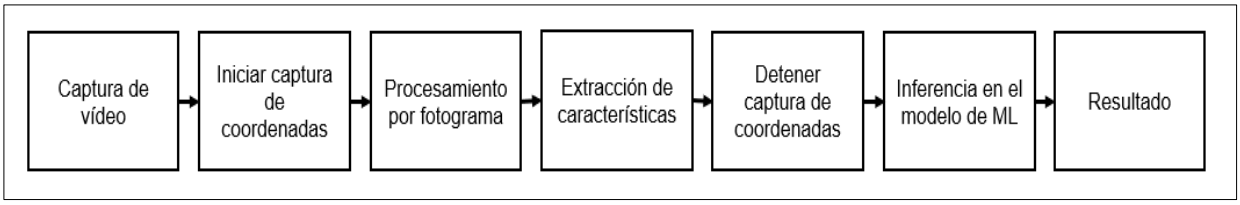
Figura 3. Diagrama de reconocimiento de señas fijas



Elaboración propia.

Para el reconocimiento de señas con movimiento se diseñó un sistema de funcionamiento para capturar secuencias de video utilizando un botón de Inicio/Término a partir de la captura de video en tiempo real, al presionarlo se comienzan a almacenar las coordenadas detectadas por la librería MediaPipe de ambas manos en cada fotograma procesado en una lista, el proceso continúa hasta que se presiona nuevamente el botón para detener la captura. Luego, la lista se convierte a un Tensor Buffer que se compara en el modelo de ML, entrenado en esta investigación para reconocer señas con movimiento, del resultado de la predicción se toma la clase de mayor porcentaje y la seña correspondiente se muestra en pantalla, en un cuadro de texto bajo el video. En la figura 4 se muestra el diagrama propuesto de la secuencia de reconocimiento de señas con movimiento.

Figura 4. Diagrama de reconocimiento de señas con movimiento



Elaboración propia.

Reconocimiento de señas fijas

Obtención de datos de señas fijas

Se recolectó un dataset de las 21 señas fijas del alfabeto dactilológico de la LSM, correspondientes a las señas A, B, C, D, E, F, G, H, I, L, M, N, O, P, R, S, T, U, V, W y Y. Por cada seña se grabaron 8 videos de diferentes personas realizando las señas, con una duración de 1 minuto cada uno. Posteriormente utilizando la librería MediaPipe en Python, por cada fotograma se extrajo la región de

la mano derecha realizando la seña y se guardó como imagen de tamaño 200 x 200 pixeles. En la Figura 5 se muestran ejemplos de las imágenes obtenidas.

Figura 5. Ejemplos de las imágenes de señas obtenidas

Elaboración propia.

- Al conjunto de imagenes resultantes, se agregaron 7 transformaciones aleatorias utilizando la librería Keras, consistentes en:
- rescale=1./255: Para escalar los valores de píxeles de las imágenes de 0-255 a 0-1.
- rotation_range=40: Para rotar de la imagen en un rango de -40 a 40 grados.
- width_shift_range=0.2: Para desplazar la imagen horizontalmente en un rango de -20% a 20% del ancho total de la imagen.
- height_shift_range=0.2: Para desplazar la imagen verticalmente en un rango de -20% a 20% de la altura total de la imagen.
- shear_range=0.2: Para realizar un corte a la imagen, en rango de corte de 20%.
- zoom_range=0.2: Para aplicar un zoom aleatorio a la imagen, en un rango de -20% a 20%.
- horizontal_flip=True: Para invertir la imagen horizontalmente.

Luego, en cada imagen procesada con estas transformaciones, se extrajeron las coordenadas en (X, Y) de los 21 puntos del HandLandmark reconocidos por MediaPipe, estas coordenadas se guardaron en un archivo separado por comas (CSV), resultante en 137,763 filas, que representan a las imágenes procesadas, y 42 columnas para las coordenadas, además de 1 columna para la etiqueta de las señas enumeradas en un rango del 0 al 20. En la Tabla 2 se muestra un resumen del dataset recolectado.

Tabla 2. Composición del dataset recolectado.

Seña	Cantidad de imágenes	Etiqueta
A	6,839	0
B	6,356	1
C	6,240	2
D	6,748	3
E	6,306	4
F	6,702	5
G	6,941	6
H	7,058	7
I	6,146	8
L	7,038	9
M	6,165	10
N	6,103	11
O	6,535	12
P	6,928	13
R	6,416	14
S	6,567	15
T	6,378	16
U	6,449	17
V	6,242	18
W	6,594	19
Y	7,012	20

Fuente: elaboración propia.

Modelo para la clasificación de señas fijas

En un estudio previo (Salgado et al. 2024) para la clasificación de señas del alfabeto dactilológico, se utilizaron 4 técnicas para el entrenamiento de modelos de ML a partir de imágenes de manos representando las señas, posteriormente con los modelos obtenidos en cada técnica se evaluaron 100 nuevas imágenes por cada seña. A continuación se mencionan las técnicas utilizadas y los resultados conseguidos en cada una de ellas:

1. Entrenamiento utilizando Plataforma Teachable Machine. El entrenamiento se realizó con 1,000 imágenes por cada seña. Con el modelo obtenido se obtuvo un 48.80% de clasificación correcta y un 51.19% de clasificación incorrecta.
2. Reentrenamiento del algoritmo Yolo V5. Se utilizaron 1,000 imágenes para el entrenamiento y 300 para la validación. Con el modelo obtenido se obtuvo un 94.95% de clasificación correcta y un 5.05% de clasificación incorrecta.

3. Entrenamiento de una CNN. El entrenamiento se realizó ingresando las imágenes directamente a la red, para cada seña se utilizaron 2,400 imágenes para entrenamiento y 720 imágenes para validación. La red se entrenó a 20 épocas, en la última época se consiguieron los parámetros: loss: 0.0187, accuracy: 0.9956, val_loss: 0.0351 y val_accuracy: 0.9949. Con este modelo se obtuvo un 69.38% de clasificación correcta y un 30.61% de clasificación incorrecta.
4. Entrenamiento de una red MLP. Se utilizó una arquitectura de red conformada en general por 4 capas con 256, 128, 64 y 32 unidades respectivamente. Para entrenar la red se utilizaron las coordenadas de los 21 puntos del Landmark de MediaPipe reconocidos en las imágenes. Para ello se utilizó un dataset correspondiente a 19,956 imágenes, con el 80% de los datos para el entrenamiento y el 20% para la validación. Con el modelo obtenido con esta técnica se obtuvo un 99.14% de clasificación correcta y un 0.85% de clasificación incorrecta.
5. Dado que el modelo de ML obtenido con la técnica 4 obtuvo los mejores porcentajes en la clasificación, se decidió implementar la red MLP para entrenar el modelo para la clasificación de señas fijas, utilizando el dataset correspondiente a 137,764 imágenes, dividido en 80% para entrenamiento y 20% para validación.

Arquitecturas del modelo para la clasificación de señas fijas

De acuerdo con la arquitectura del modelo MLP utilizada en (Salgado et al., 2024) se propuso experimentar con variaciones de esta, aumentando y disminuyendo el número capas y unidades. En la Tabla 2 se muestran 5 variaciones de la arquitectura del modelo MLP elegido para la clasificación de señas fijas, en cada una se muestra también la cantidad de parámetros entrenables.

Tabla 2. Arquitecturas propuestas para el modelo de red MLP para la clasificación de señas fijas.

Arquitectura	Capa 1	Capa 2	Capa 3	Capa 4	Parámetros
1	128	64	32	16	16,725
2	256	128	64	32	54,933
3	512	256	128	64	195,861
4	1024	512	256	128	735,765
5	2048	1024	512	256	2,847,765

Fuente: elaboración propia

Reconocimiento de señas con movimiento

Obtención de datos de señas con movimiento

Se recolectó un dataset de 16 señas que implican movimiento de uno o ambos brazos para su representación, así como configuraciones específicas de las manos en cada una de ellas. Las señas elegidas corresponden a las expresiones "**Bien**", "**Mal**", "**Dolor**", "**Casa**", "**Baño**", "**Teléfono**", "**Agua**", "**Comer**", "**Hambre**", "**Con**", "**De**", "**Es**", "**Hola**", "**¿Cómo estás?**", "**Buenos días**" y "**Gracias**". Por cada una de estas señas se recolectaron 100 videos capturados con distintos smartphones, con la ayuda de 10 personas voluntarias.

Los videos tienen una duración entre 3 y 4 segundos, dependiendo de la seña efectuada, se procesaron para tener una resolución de 1440x1920 pixeles y 30 fotogramas por segundo en promedio. En cada video se capturó el espacio que va de arriba de la cabeza hasta la parte superior de las rodillas, un ejemplo de los videos capturados se muestra en la Figura 6.

Figura 6. Fotogramas de ejemplo de la seña "Gracias"

Elaboración propia.

A partir de cada video se obtuvieron secuencias de datos, consistentes en las coordenadas de los puntos reconocidos por el HandLanmark de MediaPipe de ambas manos, las secuencias se almacenaron en un archivo separado por comas (CSV), donde cada fila representa las coordenadas reconocidas en un fotograma. Las secuencias resultantes tenían una longitud entre 45 y 220 filas por lo que se fijó la longitud de 220, rellenando con valores de -2 las filas vacías de las secuencias con cantidad de fotogramas inferiores.

Dado que al realizar pruebas de la Aplicación de MediaPipe en el reconocimiento de señas con movimiento, se encontró que la aplicación procesa un promedio de entre 30 y 40 fotogramas por segundo, por lo que se decidió reducir las secuencias en una tercera parte, para que las secuencias de

entrenamiento de los modelos de ML fueran lo más semejante posible en longitud a las secuencias capturadas por la Aplicación Móvil. Para ello, se extrajeron filas de forma distribuida de cada secuencia original y se fijó la longitud de las mismas en 74 filas, rellenando con valores de -2 las filas de secuencias inferiores a esta longitud.

Por lo que finalmente se obtuvo un archivo CSV resultante de 118,400 filas, equivalentes a 1,600 secuencias de longitud de 74 filas, con 84 columnas para las coordenadas (X, Y) y 1 columna más para las etiquetas. Además, al archivo CSV obtenido se le agregaron 3 tipos de Aumento de Datos consistentes en *Desplazamiento Horizontal y Vertical*, *Zoom In* y *Zoom Out*, por lo que se obtuvo un dataset con Aumento de Datos equivalente a 6,400 videos, equivalentes a 400 por seña.

Modelos para la clasificación de señas con movimiento

Para la Clasificación de señas con movimiento se propusieron tres modelos de RN: LSTM, GRU y Transformer, que combina una CNN y una LSTM. Estos modelos se eligieron ya que se especializan en el entrenamiento con datos secuenciales (Mejía, 2022). A continuación, se describen cada uno de los modelos:

Long Short Term Memory (LSTM)

La red LSTM fue propuesta por Hochreiter & Schmidhuber (1997), es una red neuronal capaz de aprender a partir de secuencias largas de datos, teniendo como objetivo la reducción del problema de desaparición del Gradiente presente en su antecesora la Red Neuronal Recurrente (en inglés Recurrent Neural Network, RNN), en la que a medida que los gradientes se propagan durante la retropropagación, su norma tiende a disminuir de manera exponencial, lo que resulta en una actualización de pesos insignificante para las neuronas cercanas al comienzo de la secuencia.

Una unidad de LSTM contiene una celda de memoria, en la que se regula la información mediante una puerta de entrada, una puerta de olvido y puerta de salida. En la puerta de entrada se decide qué fragmentos de información almacenar en el estado de la celda de memoria. En la puerta de olvido se decide que parte de la información debe ser descartada y en la puerta de salida se controla que parte de la información de la celda de memoria se utilizará como salida, tomando en cuenta los estados anterior y actual. Al seleccionar información relevante en cada estado le permite a la red LSTM manejar dependencias a largo plazo. Durante la retropropagación del gradiente, los LSTM pueden mantener un

flujo constante de información a través del tiempo, evitando así el problema del “gradiente que desaparece” y permitiendo un aprendizaje más estable y preciso (Van Houdt et al., 2020).

Gated Recurrent Unit (GRU)

Una red GRU es también una variante de las RNN. La red GRU, al igual que la red LSTM, puede capturar dependencias a largo plazo en secuencias. Sin embargo, las GRU son más simples y más rápidas que las LSTM debido a que tienen menos parámetros (Cho et al., 2014). Una red GRU contiene una Puerta de actualización y una puerta de reinicio. La puerta de actualización es una combinación de las puertas de olvido y la puerta de entrada de una LSTM, en esta se decide cuánta información de la celda anterior debe pasar a la celda actual y cuánto de la nueva información debe ser almacenada. En la puerta de reinicio se controla la cantidad de información antigua que debe olvidarse o "reiniciarse" en cada paso de tiempo. En la red GRU también se integra el estado de la celda y el estado oculto, lo que soluciona problemas de estancamiento local de mínima y descenso de gradiente.

Red Neuronal Transformer Compuesta

Una red neuronal Transformer es una arquitectura de red diseñada para manejar secuencias de datos de manera eficiente y escalable, fue introducida por Vaswani et al. (2017), de acuerdo con estos autores, a diferencia de las redes neuronales recurrentes (RNN y LSTM) que procesan los datos de manera secuencial, los Transformers procesan todas las partes de la secuencia de entrada simultáneamente, utilizando un sistema de Atención que le permite calcular la importancia de diferentes partes de la secuencia en relación con cada elemento. Este proceso permite a los Transformers aprender dependencias a corto y largo plazo de manera eficiente, acelerando el entrenamiento y la inferencia sin sufrir el problema del desvanecimiento del gradiente.

El modelo Transformer Compuesto está basado en la propuesta de solución ganadora en el concurso de Google “Isolated Sign Language Recognition” (Chow, et. al 2023) cuyo objetivo es clasificar señas aisladas de la ASL utilizando datos etiquetados, de puntos de referencia del cuerpo, extraídos mediante la solución Holística MediaPipe, que incluye detección de pose del cuerpo, detección de puntos de las manos y puntos del rostro.

La solución ganadora del concurso (Kaggle, 2024), propuesta por el usuario @hoyso48 (hoyso48, 2024), combina una CNN unidimensional, las cuales son efectivas para modelar secuencias donde hay

una fuerte correlación local entre los datos de entrada, en este caso movimientos en fotogramas de video (Cruz et al., 2021) y una Red Transformer para aprovechar su eficacia para capturar dependencias a largo plazo entre diferentes posiciones en una secuencia. De acuerdo con el usuario @hoyso48, de este modo la CNN puede utilizarse como tokenizador entrenable para el Transformer, en donde la CNN actúa como una capa de preprocesamiento que extrae características locales de los datos de secuenciales y el Transformer utiliza estas características para modelar relaciones a largo plazo y complejas en la secuencia.

Arquitecturas de los modelos para la clasificación de las señas con movimiento

Para cada tipo de modelo se propusieron varias arquitecturas para realizar el entrenamiento, para ello, el set de datos se dividió en 85% para el entrenamiento y 15% para validación. El entrenamiento se realizó el lenguaje de programación Python utilizando las librerías TensorFlow y Keras. En la Tabla 3 se muestran las arquitecturas propuestas para el entrenamiento del modelo LSTM y en la Tabla 4 se muestran las arquitecturas propuestas para el modelo GRU.

Tabla 3. Arquitecturas propuestas para el modelo LSTM.

Arquitectura	Capa 1	Capa 2	Capa Densa	Parámetros
1	32	16	32	19,184
2	64	32	64	53,712
3	128	64	128	168,848
4	256	128	256	583,440
5	512	256	512	2,149,904
6	1024	512	1024	8,231,952

Fuente: elaboración propia

Tabla 4. Arquitecturas propuestas para el modelo GRU.

Arquitectura	Capa 1	Capa 2	Capa Densa	Parámetros
1	32	16	32	14,800
2	64	32	64	41,360
3	128	64	128	129,808
4	256	128	256	448,016
5	512	256	512	1,649,680
6	1024	512	1024	6,314,000

Fuente: elaboración propia

El modelo Transformer Compuesto contiene una parte CNN, para extraer características espaciales de las secuencias, una parte LSTM, para capturar dependencias a largo plazo en las secuencias, y uno o

más bloques Transformer para mejorar la capacidad de la red en modelar relaciones a largo plazo entre los datos secuenciales. La arquitectura propuesta del modelo es la siguiente:

Entradas: La entrada tiene una forma de (74, 84, 1), que representa a la secuencia de 74 fotogramas, con 84 características que representan las coordenadas (X, Y) y una sola dimensión de profundidad.

Máscara: Dado que las secuencias son de longitud variable y se rellenaron para tener una longitud de 74 con valores de -2 se utilizó una capa de MAsking para ignorar este valor.

Parte CNN:

- Se aplica una primera capa convolucional unidimensional con 32 filtros y un kernel de tamaño 3 a cada paso temporal.
- Se aplica una segunda capa convolucional unidimensional con con 64 filtros y kernel de tamaño 3.

Después de aplicar las convoluciones, la salida de cada paso temporal es aplanada para reducir la dimensionalidad

Parte LSTM: Se aplica una capa LSTM con 128 unidades, que procesa y retorna las secuencias completas para que puedan ser procesadas por el bloque Transformer.

Parte Transformer: Se aplican uno o más bloques Transformer, cada bloque tiene dos componentes clave:

1. Capa de atención MultiHeadAttention: Utiliza varias cabezas de atención para aprender relaciones entre diferentes partes de la secuencia, permitiendo que el modelo se enfoque en partes relevantes de la secuencia.
2. Capa de red neuronal completamente conectada (feedforward) densa: Después de la atención, se pasa la salida a una red feedforward, que ajusta las características aprendidas. En este caso, se aplica una activación *'relu'* para la primera capa.

Cada bloque Transformer incluye la normalización por capas y el uso de conexiones residuales (la salida de la atención se suma a la entrada original). Para ayudar a estabilizar el entrenamiento y a preservar las señales de información a través de la red.

Clasificación Final

- Se aplica *GlobalAveragePooling1D* para obtener un vector de características fijo a partir de la salida del Transformer.
- Se aplica una capa densa con 256 neuronas, con activación *'relu'* para combinar las características extraídas por los bloques anteriores.
- Se aplica un *Dropout* con valor de 0.5 para apagar aleatoriamente el 50% de las neuronas para prevenir el sobreajuste.
- Se aplica una capa de salida densa con 21 neuronas (el número de clases) y una función de activación *'softmax'* para realizar la clasificación final.

En la Tabla 5 se muestra el resumen de la arquitectura propuesta.

Tabla 5. Arquitectura propuesta para el modelo de red neuronal Transformer Compuesta.

Arquitectura	Capa CNN 1	Capa CNN 2	Capa LSTM	Capa Densa	Parámetros
1	32	64	128	256	3,061,968

Fuente: elaboración propia

3.4 Evaluación

3.4.1 Evaluación de los modelos

Para evaluar el rendimiento de los modelos tanto de reconocimiento de señas fijas y señas con movimiento se midieron las métricas Accuracy, Precision, Recall y F1-Score, que son las más frecuentes utilizadas en estudios de clasificación predictiva mediante algoritmos de Machine y Deep Learning (Borja, 2020), ejemplo de ello son los trabajos para el reconocimiento de la LSM realizados por Mejía, (2022) y (Espejel, 2021). Estas métricas se midieron durante el entrenamiento y posteriormente en un conjunto de nuevos datos, para ello se utilizó la función *classification_report* de la librería *Scikit-learn* (Scikit-Learn, 2024). Los valores de estas métricas están definidas mediante el número de Verdaderos Positivos (VP), Falsos Positivos (FP), Verdaderos Negativos (VN) y Falsos Negativos (FN), descritos a continuación:

- Verdaderos Positivos (VP): Se refiere al número de ejemplos que fueron correctamente clasificados como la clase positiva.

- Verdaderos Negativos (VN): Número de ejemplos en los que el modelo predijo correctamente la clase negativa.
- Falsos Positivos (FP): Se refiere al número de ejemplos que fueron incorrectamente clasificados como clase positiva (es decir, la clase negativa como positiva).
- Falsos Negativos (FN): Número de ejemplos que fueron incorrectamente clasificados como la clase negativa (es decir, clase positiva como negativa).

La métrica Accuracy mide el porcentaje de predicciones correctas sobre el total de ejemplos, se calcula con la ecuación:

$$Accuracy = \frac{VP + VN}{\mathrm{VP} + \mathrm{VN} + \mathrm{FP} + \mathrm{FN}}$$

La métrica Precision indica que proporción de las predicciones positivas son correcta. La Precision se calcula con la ecuación:

$$Precision = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FP}}$$

La métrica Recall mide qué proporción de los ejemplos positivos fueron correctamente identificados. La métrica Recall se calcula con la ecuación:

$$Recall = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FN}}$$

La métrica F1-Score mide el balance entre las métricas Precision y Recall, es decir qué tan bien el modelo equilibra la precisión y la exhaustividad. Es útil cuando existe un desbalance entre estas dos métricas o cuando las clases están desbalanceadas. El F1-Score se calcula mediante la ecuación:

$$F1 = 2 * \left(\frac{\mathrm{Recall} * \mathrm{Precision}}{\mathrm{Recall} + \mathrm{Precision}}\right)$$

El F1-Score solo será alto si tanto la precisión como la exhaustividad son altas. Un F1-Score cercano a 1 indica que el modelo está balanceando bien la precisión y la exhaustividad, mientras que un F1-Score bajo indica un desequilibrio entre ambas métricas.

Evaluación de los modelos en la Aplicación Móvil

El rendimiento de aplicaciones móviles en entornos de reconocimiento de objetos en tiempo real se evalúa midiendo el uso de los recursos del dispositivo que impactan directamente en la velocidad de

funcionamiento (Martinez et al., 2022), en esta investigación se implementó una arquitectura destinada a sistemas portátiles para el reconocimiento de objetos mediante un modelo de ML basado en CNN, el rendimiento se calculó midiendo aspectos como el tiempo de inferencia, la velocidad de procesamiento (en fotogramas por segundo) y el uso de memoria RAM.

De acuerdo con lo anterior se proponen estas métricas para evaluar el rendimiento de los modelos de reconocimiento de señas fijas y señas con movimiento en la Aplicación móvil, además se evaluó el número de inferencias correctas y el porcentaje de reconocimiento. La forma en cómo se miden las métricas se menciona a continuación:

- Tiempo de inferencia: medido desde que se compara el Tensor Buffer de coordenadas con el modelo de ML hasta que éste devuelve el Tensor Buffer con los porcentajes de reconocimiento.
- Velocidad de procesamiento en Fotogramas Por Segundo (FPS): para las señas fijas es el promedio de los fotogramas procesados en un determinado lapso. Para las señas con movimiento es el promedio de fotogramas procesados en un determinado número de señas capturadas.
- Uso de memoria RAM en el dispositivo: Obtenido de la diferencia de memoria utilizada justo antes de comparar el Tensor Buffer de coordenadas con el modelo de ML, con la memoria utilizada después que el modelo devuelve el Tensor Buffer con los porcentajes de reconocimiento.
- Número de inferencias correctos: mide el número de veces que se logró reconocer de forma correcta la seña capturada en un determinado número de señas capturadas.
- Porcentaje en el reconocimiento: es el promedio de los porcentajes de las señas inferidas en una determinada cantidad de señas capturadas.

RESULTADOS

Reconocimiento de señas fijas

Una vez realizado el entrenamiento de las arquitecturas propuestas para la red MLP se obtuvieron las métricas de rendimiento propuestas en la sección **3.4.1** para la evaluación de los modelos. En la Tabla 6 se muestran los resultados de las métricas por cada arquitectura.

Tabla 6. Resultados de las métricas de las arquitecturas propuestas para la red MLP.

Arquitectura	Accuracy	Precision	Recall	F1-Score
1	85.6494	0.8670	0.8558	0.8449
2	93.2421	0.9335	0.9313	0.9315
3	94.6321	0.9462	0.9455	0.9456
4	95.4451	0.9543	0.9537	0.9538
5	96.2835	0.9627	0.9622	0.9623

Fuente: elaboración propia

Como se puede apreciar en la Tabla 6, la arquitectura 5 tuvo los valores más altos en cada una de las métricas por lo que se decidió utilizar esta arquitectura para realizar pruebas en la Aplicación Móvil.

La Aplicación se probó en un teléfono modelo POCO X5 PRO 5G, del año 2023. Para ello se tomaron secuencias de video de 10 segundos de tiempo por cada seña. En el lapso mencionado se evaluaron las métricas propuestas en la sección **3.4.2**. En la Tabla 7 se muestran los resultados conseguidos en una distancia de 100 cm entre en dispositivo móvil y la persona señante, y en la Tabla 8 se muestran los resultados conseguidos en una distancia de 50 cm entre en dispositivo móvil y la persona señante.

Tabla 7. Resultados del reconocimiento de señas fijas en tiempo real en Aplicación Móvil, en un lapso de 10 segundos, a una distancia de 100 cm entre el dispositivo y el señante, en un teléfono modelo POCO X5 PRO 5G.

Seña	Inferencias Correctas	Inferencias Incorrectas	Promedio de porcentaje en el reconocimiento	Tiempo de Inferencia promedio (MS)	Velocidad de procesamiento (FPS)	Uso de Memoria Promedio (bytes)
A	194	0	0.900812544	0.75257732	19.4	118.2989691
B	0	162	0.391272322	0.691358025	16.2	112.84
C	0	213	0.763101818	0.5	21.3	114.6391753
D	143	37	0.65547896	0.484512	21.4	114.451263
E	136	81	0.469004538	0.456221198	21.7	113.4147465
F	159	29	0.324079556	0.510638298	18.8	116.143617
G	215	0	0.628380498	0.43255814	21.5	116.855814
H	0	210	0.211964799	0.452380952	21	116.0761905
I	183	0	0.920968745	0.453551913	18.3	114.4480874
L	0	174	0.485380295	0.557471264	17.4	112.2126437
M	0	181	0.233402692	0.497237569	18.1	112.8618785
N	0	200	0.202359575	0.46	20	112.86

O	207	0	0.825763099	0.429951691	20.7	114.4541063
P	0	203	0.349151556	0.512315271	20.3	112.8965517
R	0	209	0.909794473	0.454545455	20.9	114.9282297
S	0	185	0.502915011	0.594594595	18.5	113.8378378
T	205	0	0.663224616	0.487804878	20.5	114.9073171
U	196	0	0.908930222	0.484693878	19.6	112.8418367
V	26	166	0.596139832	0.411458333	19.2	113.765625
W	0	194	0.396151436	0.391752577	19.4	113.3402062
Y	0	174	0.259523149	0.477011494	17.4	115.2873563

Fuente: elaboración propia

Tabla 8. Resultados del reconocimiento de señas fijas en tiempo real en Aplicación Móvil, en un lapso de 10 segundos, a una distancia de 50 cm entre el dispositivo y el señante, en un teléfono modelo POCO X5 PRO 5G.

Seña	Inferencias Correctas	Inferencias Incorrectas	Promedio de porcentaje en el reconocimiento	Tiempo de Inferencia promedio (MS)	Velocidad de procesamiento (FPS)	Uso de Memoria Promedio (bytes)
A	207	0	0.938497948	0.45410628	20.7	116.0821256
B	223	0	0.819925307	0.475336323	22.3	116.0538117
C	120	95	0.57468335	0.418604651	21.5	113.7488372
D	212	0	0.98908578	0.424528302	21.2	115.1509434
E	203	0	0.917684382	0.413793103	20.3	113.3399015
F	174	0	0.942394692	0.442528736	17.4	112.5977011
G	167	0	0.927169286	0.538922156	16.7	114.0658683
H	194	0	0.821242517	0.453608247	19.4	111.7680412
I	171	0	0.979813902	0.514619883	17.1	112.9122807
L	167	0	0.814033432	0.526946108	16.7	113.1257485
M	156	24	0.381430845	0.505555556	18	114.2888889
N	155	16	0.611092944	0.479532164	17.1	111.122807
O	166	0	0.956574215	0.542168675	16.6	113.1325301
P	150	13	0.82306638	0.496932515	16.3	115.0429448
R	18	125	0.705625864	0.475524476	14.3	112.7272727
S	123	36	0.528699479	0.477987421	15.9	114.8805031
T	160	7	0.924769283	0.556886228	16.7	113.9221557
U	168	0	0.906979567	0.56547619	16.8	113.1190476
V	163	0	0.891876237	0.601226994	16.3	113.196319

W	173	0	0.912853312	0.485549133	17.3	112.416185
Y	174	0	0.687680835	0.465517241	17.4	113.1954023

Fuente: elaboración propia

En la Tabla 7 se muestra que 5 señas (A, I, G, O, T, U) se detectaron correctamente en el lapso medido, 3 señas (D, E, F, V) tuvieron un reconocimiento aceptable, mientras que 11 señas (B, C, H, L, M, N, P, R, S, W, Y) no se detectaron de forma correcta en ninguna ocasión, lo que indica la Aplicación no pudo detectar más de la mitad de las señas a 100 cm de distancia. Mientras que en la Tabla 8 se muestra que 14 señas (A, B, D, E, F, G, H, I, L, O, U, V, W, Y) se detectaron correctamente en el lapso medido y 7 señas (C, M, N, P, R, S, T) en su mayoría se detectaron correctamente, esto muestra la Aplicación reconoce mejor las señas a una distancia de 50 cm de la persona señante.

Además, los promedios de reconocimiento también mejoraron a una distancia de 50 cm, dado que en la tabla 1 se muestra que solo 4 señas (A, I, R, U) alcanzaron un promedio mayor a 0.90, mientras que en la tabla 2, 10 señas (A, D, E, F, G, I, O, T, U, W) alcanzaron promedios mayores a 0.90, lo que es indicación que el modelo puede predecir mejor a corta distancia.

En la Tabla 7 los promedios de tiempo de inferencia oscilan entre 0.391752577 MS y 0.75257732 MS, y en la Tabla 8 entre 0.413793103 MS y 0.601226994 MS, esto indica que también hubo una mejoría en el tiempo que le toma al modelo reconocer las señas. Además, se puede notar también que la velocidad de procesamiento disminuyó de manera general a una distancia de 50 cm, ya que a excepción de 4 señas (A, B, C, Y) el promedio de FPS disminuyó, un ejemplo claro es la seña G que a una distancia de 100 cm tuvo una velocidad de 21.5 FPS en promedio, mientras que a una distancia de 50 cm tuvo una velocidad de 16.7 FPS en promedio.

Finalmente, en el uso de memoria también hubo una variación, ya que de acuerdo con estos resultados la Aplicación consume menos recursos a una distancia de 50 cm, ya que en 14 señas (A, C, E, F, G, H, I, N, O, R, T, V, W, Y) hubo una reducción en el promedio de uso de memora en el dispositivo.

Reconocimiento de señas con movimiento

Posterior al entrenamiento de las arquitecturas propuestas para los modelos, se evaluaron las métricas propuestas en el apartado **3.4.1**. Los resultados del entrenamiento de la red LSTM se muestran en la

Tabla 9, además al set de datos se agregó aumento de datos y se realizó el entrenamiento con las mismas arquitecturas propuestas, en la Tabla 10 se muestran los resultados de las métricas.

Tabla 9. Resultados de las métricas de las arquitecturas propuestas para el modelo LSTM.

Arquitectura	Accuracy	Precision	Recall	F1-Score
1	75.83333333	0.761865448	0.762889912	0.75553552
2	88.75	0.888047246	0.895242723	0.887937045
3	90.83333333	0.912822497	0.913299247	0.909066418
4	93.33333333	0.934132081	0.936964387	0.932959272
5	93.75	0.939569896	0.942185471	0.93963269
6	93.75	0.938728989	0.940643939	0.936402235

Fuente: elaboración propia

Tabla 10. Resultados de las métricas de las arquitecturas propuestas para el modelo LSTM entrenado con Aumento de Datos.

Arquitectura	Accuracy	Precision	Recall	F1-Score
1	84.0625	0.83898197	0.839837337	0.837903188
2	91.875	0.919629378	0.916835952	0.916879541
3	93.22916667	0.934997393	0.930321213	0.93103289
4	94.58333333	0.949315855	0.943871442	0.944633106
5	96.875	0.968529974	0.967573108	0.967697611
6	97.29166667	0.974917411	0.9724453	0.973126409

Fuente: elaboración propia

De igual forma para la red GRU se realizó el entrenamiento con el dataset original y con aumento de datos, en la Tabla 11 se muestran los resultados de las métricas del entrenamiento con el dataset original y la Tabla 12 muestra los resultados del entrenamiento con Aumento de Datos.

Tabla 11. Resultados de las métricas de las arquitecturas propuestas para el modelo GRU.

Arquitectura	Accuracy	Precision	Recall	F1-Score
1	86.66666667	0.871124892	0.874808673	0.864894381
2	93.33333333	0.928724783	0.939971947	0.931116078
3	94.58333333	0.948169472	0.947119076	0.945216945
4	95.0	0.949574914	0.950257035	0.947803491
5	97.08333333	0.969092947	0.971781549	0.968754132
6	97.91666667	0.977573529	0.980904356	0.97888692

Fuente: elaboración propia

Tabla 12. Resultados de las métricas de las arquitecturas propuestas para el modelo GRU con Aumento de Datos.

Arquitectura	Accuracy	Precision	Recall	F1-Score
1	90.625	0.906577307	0.907029039	0.90549601
2	89.0625	0.892048235	0.890115785	0.888964082
3	92.60416667	0.926850746	0.926022432	0.925650028
4	96.45833333	0.963804498	0.963753261	0.963282544
5	97.5	0.974782789	0.974876494	0.97464007
6	77.39583333	0.859528952	0.773175175	0.795416246

Fuente: elaboración propia

El entrenamiento de la arquitectura de la red Transformer se realizó de igual forma que los modelos de las redes anteriores, en el Tabla 13 se muestran los resultados de las métricas en el entrenamiento con el dataset original y en la Tabla 14 se muestran los resultados de las métricas del entrenamiento con aumento de datos.

Tabla 13. Resultados de las métricas de las arquitecturas propuestas para el modelo Transformer.

Arquitectura	Accuracy	Precision	Recall	F1-Score
1	94.16666667	0.944263156	0.943371212	0.94121961

Fuente: elaboración propia

Tabla 14. Resultados de las métricas de las arquitecturas propuestas para el modelo Transformer con Aumento de Datos.

Arquitectura	Accuracy	Precision	Recall	F1-Score
1	93.375	0.936335653	0.93496251	0.934732465

Fuente: elaboración propia

Posteriormente al entrenamiento, todos los modelos se evaluaron en 100 nuevas secuencias de datos y se midieron las métricas propuestas para la evaluación de los mismos. En la Tabla 15 se muestran los resultados de los modelos cuya arquitectura obtuvo el mejor desempeño, para los modelos entrenados con el dataset original y con Aumento de Datos.

Tabla 15. Resultados de las métricas en 100 nuevos datos.

Model	Arquitectura	Accuracy	Precision	Recall	F1-Score
LSTM	5	0.81	0.81	0.81	0.78
LSTM (Data Aug.)	2	0.81	0.79	0.81	0.78
GRU	6	0.74	0.79	0.74	0.71

GRU (Data Aug.)	4	0.83	0.9	0.83	0.8
Transformer	1	0.90	0.93	0.9	0.89
Transformer (Data Aug.)	1	0.92	0.94	0.92	0.91

Fuente: elaboración propia

De acuerdo con estos resultados mostrados en la Tabla 15, se puede observar que el modelo Transformer con Aumento de Datos tuvo el mejor rendimiento, por lo que se eligió este modelo para evaluar su funcionamiento en la Aplicación Móvil. La aplicación se probó en un teléfono modelo POCO X5 PRO 5G, del año 2023. Para ello, se tomaron capturas de 50 secuencias por cada seña desde la Aplicación Móvil, en cada una se midieron los aspectos de evaluación propuestos en el apartado 3.4.2. Los resultados se muestran en la Tabla 16.

Tabla 16. Resultados del reconocimiento de señas con movimiento en la Aplicación Móvil, en un teléfono modelo POCO X5 PRO 5G.

Seña	**1 / 2 manos**	**Inferencias Correctas**	**Promedio de porcentaje en el reconocimiento**	**Tiempo De Inferencia Promedio (MS)**	**Promedio fotogramas procesados**	**Uso De Memoria Promedio (bytes)**
Agua	1	0	0.846183799	273.9	18.82	940196.64
Baño	1	34	0.844926762	242.54	20.86	620488.64
Bien	1	31	0.613038004	262.76	29.08	472787.04
Buenos Días	2	8	0.689014679	250.32	34.92	138924.00
Casa	2	21	0.890057083	235.62	35.34	357954.72
Comer	1	1	0.695806874	286.82	31.58	700282.88
¿Cómo estás?	2	35	0.934179095	246.66	29.1	41322.88
Con	2	39	0.930367277	230.24	29.42	859323.84
De	1	5	0.796003784	232.6	33.16	34790.72
Dolor	1	11	0.719690505	237.48	37.8	687834.56
Es	1	3	0.767289079	232.62	37.64	1315477.12
Gracias	2	38	0.944447134	240.72	33.06	859732.00
Hambre	2	36	0.941854351	232.38	34.94	1439584.32

Hola	1	2	0.849352877	238.02	41.36	979467.04
Mal	1	22	0.660801736	236.26	36.74	339006.56
Teléfono	1	0	0.71717658	233.6	34.4	1007166.72

Fuente: elaboración propia

Como se puede apreciar en la Tabla 16, las señas "**Con"**, "**Gracias"**, "**Hambre"** y "**¿Cómo estás?"** tuvieron mayor cantidad de inferencias correctas, es importante señalar que estas señas implementan ambas manos en su ejecución. En las que se utiliza una sola mano, las señas que mejor resultado obtuvieron fueron "**Baño"**, "**Bien"** y "**Mal"**, mientras que las señas "**Agua"** y "**Teléfono"** no obtuvieron ninguna inferencia correcta, esto pudiera deberse a que en estas señas el movimiento del brazo es muy parecido, en estos casos la mano sube a la altura del rostro y luego baja a la cintura, la diferencia está en la configuración de la mano y movimiento de los dedos.

En las señas "**Con"**, "**Gracias"**, "**Hambre"** y "**¿Cómo estás?"** se alcanzaron promedios de porcentajes de reconocimiento mayores a 0.93, en estos casos la confianza en el reconocimiento fue más alta, mientras que en las señas "**Bien"** y "**Mal"** tuvieron los porcentajes de confianza más bajos, estas señas se realizan con una sola mano. En las señas que se realizan con una sola mano el porcentaje fue menor a 0.85, esto podría deberse a la similitud entre las señas, ya que todas se realizaron con el brazo derecho, además de que el número de porcentajes procesados indica que no se pudieron capturar detalles de la posición de la mano, de los dedos y movimientos de los mismos.

En cuanto al tiempo de inferencia, la seña que tuvo un promedio más alto fue la seña "**Comer"** con un promedio de 286.82 milisegundos, mientras que la seña "**Con"** tuvo un promedio de 230.24 milisegundos, estos tiempos impactarían directamente en la fluidez y en la experiencia de usuario en una futura producción de la Aplicación.

Lo que más llama la atención es el promedio de fotogramas procesados por la Aplicación, que varía entre 18.82 y 41.36, estas cantidades son casi una tercera parte de los fotogramas procesados por la Librería MediaPipe en el lenguaje Python, esto tiene un impacto directo en el rendimiento del reconocimiento de señas, ya que como se mencionó en la sección 3.3.1 fue necesario reducir la longitud de las secuencias obtenidas originalmente, para que fueran semejantes a las obtenidas por parte de la

Aplicación, lo que conlleva a la pérdida de detalles en la posición de la mano, de los dedos y movimientos de los mismos

En cuanto al uso de memoria, las señas "**Es"**, "**Hambre"** y "**Teléfono"** tuvieron promedios superiores a 1 Mb, y la seña con menor promedio fue "**Es**" con un equivalente a 0.03 Mb, esto resulta al considerar el rendimiento de la Aplicación en otros dispositivos con menor capacidad.

DISCUSIONES

El modelo MLP para el reconocimiento de las señas fijas del alfabeto dactilológico obtuvo un mejor rendimiento a una distancia de 50 cm, entre el teléfono y la persona señante, sin embargo, en las señas donde se obtuvo un bajo rendimiento en el reconocimiento debe trabajarse sobre ello, aumentando la cantidad de datos en estas señas.

El desempeño de la Aplicación en conjunto con el modelo de ML de reconocimiento de señas fijas demuestra un buen rendimiento en el reconocimiento de este tipo señas fijas, ya que el tiempo máximo de reconocimiento de seña es de 0.601226994 MS y el promedio de memoria utilizada máximo es de 116.0821256 bytes, por lo que se puede concluir que la aplicación no consume muchos recursos de hardware al realizar el proceso de inferencia.

Dado que en este estudio se entrenó un modelo de red neuronal MLP para el reconocimiento de señas fijas, en un trabajo futuro se puede considerar entrenar otros modelos de Redes Neuronales para explorar su rendimiento de reconocimiento de este tipo de señas.

Respecto al reconocimiento de señas con movimiento, de los tres modelos de redes neuronales propuestos fue evidente que el mejor desempeño lo tuvo el modelo Transformer Compuesto con CNN y LSTM, ya que este modelo mejoró mucho el desempeño mostrado por los modelos LSTM y GRU.

Los resultados del reconocimiento de señas con movimiento en la Aplicación muestran que el reconocimiento de señas es mejor cuando utilizan ambas manos en lugar de solo una, dado que en las señas con una sola mano el movimiento del brazo es muy similar, aunado a que la Aplicación de MediaPipe procesa un promedio de entre 30 y 40 FPS, por lo que no puede capturar detalles finos sobre la posición y movimiento de las manos y los dedos.

En este trabajo únicamente se utilizaron los 21 puntos reconocidos por MediaPipe Hands en cada mano para el proceso de reconocimiento, por lo que en trabajos fututos se podría explorar el reconocimiento de señas utilizando más puntos de interés, como cuerpo y rostro.

CONCLUSIONES

En este estudio se propuso una aplicación móvil para el reconocimiento de la LSM, empleando algoritmos de ML novedosos que mostraron un rendimiento prometedor, esta propuesta considera el reconocimiento de señas fijas y señas con movimiento, lo que brinda una ventaja sobre otros trabajos similares en los que únicamente se enfocan en las señas fijas.

Gracias a la implementación del modelo Transformer Compuesto para el reconocimiento de movimientos complejos de las manos, se consiguieron resultados aceptables en el reconocimiento de señas mediante una Aplicación Móvil, aunque también se destacan áreas específicas que podrían mejorarse para aumentar la precisión del reconocimiento de algunas señas.

El aporte principal de este estudio es la implementación del reconocimiento de señas gestuales en general en un entorno móvil en tiempo real. Sin embargo, una limitación importante es que la muestra utilizada se basa en un conjunto limitado de secuencias y condiciones controladas, lo cual puede afectar la generalización del modelo y de la Aplicación. En estudios futuros, sería recomendable aumentar el número de secuencias utilizadas en el entrenamiento, así como explorar el rendimiento de la Aplicación en otros dispositivos para optimizar su rendimiento en escenarios diversos.

Los hallazgos de esta investigación brindan la posibilidad de desarrollar una herramienta de traducción de lengua de señas en tiempo real mediante una aplicación móvil, por lo que resulta viable el desarrollo de una herramienta de traducción de la LSM que funcione como un intérprete entre hablantes y no hablantes de la misma, lo que puede impactar significativamente en la vida de las personas con discapacidad de escuchar.

REFERENCIAS BIBLIOGRÁFICAS

Chow, G. Cameron, M. Sherwood, P. Culliton, S. Sepah, S. Dane, & T. Starner. (2023). Google - Isolated Sign Language Recognition. Kaggle. https://kaggle.com/competitions/asl-signs . Consultado en septiembre 2024.

Aquino Castro R.A. (2018). *Reconocimiento e interpretación del alfabeto dactilológico de la lengua de señas mediante tecnología móvil y redes neuronales artificiales*. (Tesis inédita de licenciatura). Universidad Mayor De San Andrés, Facultad De Ciencias Puras Y Naturales Carrera De Informática, La Paz, Bolivia.

Berea Barcia, C. (2021). *Reconocimiento del alfabeto de lenguaje de señas*. (Tesis inédita de licenciatura). Universidad Autónoma de Barcelona, Barcelona, España.

Borja-Robalino, R., Monleon-Getino, A., & Rodellar, J. (2020). Estandarización de métricas de rendimiento para clasificadores Machine y Deep Learning. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (E30), 184-196.

Cámara de Diputados (10 de febrero de 2021). *Aprueban reformas para que personas con discapacidad auditiva reciban educación bilingüe en lengua de señas*. https://comunicacionnoticias.diputados.gob.mx/comunicacion/index.php/boletines/aprueban-reformas-para-que-personas-con-discapacidad-auditiva-reciban-educacion-biling-e-en-lengua-de-se-as#gsc.tab=0 ,

fecha de consulta: 19 de octubre de 2022.

Cruz Aldrete, M. (2008). Gramática de la Lengua de Señas Mexicana. (Tesis de doctorado, Colegio de México. Recuperado de https://repositorio.colmex.mx/concern/theses/kk91fk72t?locale=es cruz_m_000203845.pdf

Cruz Aldrete, M. (2014). Hacia la construcción de un diccionario de Lengua de Señas Mexicana. *Revista de Investigación 38(83)*, 57-80. Recuperado de http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S1010-29142014000300005&lng=es&tlng=es. Consultado en 06 de octubre de 2024.

Cruz, Y.J., Rivas, M., Quiza, R., Villalonga, A., Haber, R.E., & Beruvides, G. (2021). *Ensemble of convolutional neural networks based on an evolutionary algorithm applied to an industrial welding process. Computers in Industry*. 133. doi:10.1016/j.compind.2021.103530.

Espejel, J. (2021). Desarrollo de algoritmos para traductor automático de lenguaje de señas mexicanas (LSM). *Repositorio Institucional Universidad Autónoma del Estado de México*. http://hdl.handle.net/20.500.11799/111836

Salgado Martínez, G. A., Cuevas Valencia, R. E., Feliciano Morales, A., & Catalán Villegas, A. (2024). Reconocimiento de señas de la Lengua de Señas Mexicana mediante técnicas de Machine Learning. *XIKUA Boletín Científico De La Escuela Superior De Tlahuelilpan, 12* (Especial), 33-39. https://doi.org/10.29057/xikua.v12iEspecial.12696

Gallego Peralta D. (2021). *Intérprete del lenguaje de signos en tiempo real mediante Machine Learning*. (Tesis inédita de licenciatura). Universidad de Málaga, Escuela Técnica Superior De Ingeniería Informática, Málaga, España.

Gobierno de México. (2016). *La dificultad de las personas con discapacidad auditiva para comunicarse con los demás, dificulta su desarrollo educativo, profesional y humano, por consecuencia se ven limitadas sus oportunidades de inclusión*. Consejo Nacional para el Desarrollo y la Inclusión de las Personas con Discapacidad. https://www.gob.mx/conadis/articulos/lengua-de-senas-mexicana-lsm?idiom=es#:~:text=La%20Lengua%20de%20Se%C3%B1as%20Mexicana,propia%20sintaxis%2C%20gram%C3%A1tica%20y%20l%C3%A9xico , consultado en octubre de 2024.

Instituto Nacional de Estadística y Geografía. (2020). *Población con discapacidad o limitación en la actividad cotidiana por entidad federativa y tipo de actividad realiza según sexo, 2020*. https://www.inegi.org.mx/app/tabulados/interactivos/?pxq=Discapacidad_Discapacidad_02_2c111b6a-6152-40ce-bd39-6fab2c4908e3&idrt=151&opc=t, consultado en octubre de 2024.

Jin, M., Omar, Z. & Hisham M. (2016). A Mobile Application of American Sign Language Translation via Image Processing Algorithms. *2016 IEEE Region 10 Symposium*, DOI: 10.1109/TENCONSpring.2016.7519386

Kaggle. (2024). Google - Isolated Sign Language Recognition. 1st place code with reproducibility. Kaggle. https://www.kaggle.com/competitions/asl-signs/discussion/406978. Consultado en septiembre 2024.

Cho, B. van Merrienboer, DZ. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio. (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". Association for Computational Linguistics.

Luna Buendia Y. & Minajas Carbajal F. J. (2016). *Sistema de reconocimiento del alfabeto del Lenguaje de Señas Mexicano usando dispositivos móviles*. (Tesis inédita de licenciatura). Instituto Politécnico Nacional, Escuela Superior de Cómputo, Ciudad de México, México.

Martinez-Alpiste, I., Golcarenarenji, G., Wang, Q. & Alcaráz-Calero, J.M. (2022). Smartphone-based real-time object recognition architecture for portable and constrained systems. *J Real-Time Image Proc* 19, 103–115. https://doi.org/10.1007/s11554-021-01164-1

MediaPipe. (2024). MediaPipe Hands. https://github.com/google-ai-edge/mediapipe/blob/master/docs/solutions/hands.md, fecha de consulta: septiembre 2024.

Mejía-Peréz, K., Córdova-Esparza, D. M., Terven, J.,Herrera-Navarro, A. M., García-Ramírez, T., Ramírez-Pedraza, A. (2022). Automatic Recognition of Mexican Sign Language Using a Depth Camera and Recurrent Neural Networks. Appl. Sci. 2022, 12, 5523. https://doi.org/10.3390/app12115523

Pinzón Bayona G. & Sanabria Orjuela Y. G. (2021). *Desarrollo de una aplicación móvil para traductor de lenguaje de señas mediante el uso de servicios web*. (Tesis inédita de licenciatura). Universidad Católica De Colombia, Facultad De Ingeniería, Ingeniería De Sistemas Y Computación, Bogotá, Colombia.

Pérez, J. & Cruz, J. (2020). Experiencias de Inclusión-Exclusión de un Grupo de Sordos Usuarios de la Lengua de Señas Mexicana. Revista Latinoamericana de Educación Inlcusiva, 15(1), pp. 39-54. DOI: https://doi.org/10.4067/s0718-73782021000100039

S. Hochreiter, J. Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276. S2CID 1915014.

Scikit-learn. (2024). Metrics and scoring: quantifying the quality of predictions. Recuperado de https://scikit-learn.org/1.5/modules/model_evaluation.html#classification-metrics. Consultado en septiembre de 2024.

Van Houdt, G., Mosquera, C. & Nápoles, G. (2020). A review on the long short-term memory model. *Artif Intell Rev* 53, 5929–5955. https://doi.org/10.1007/s10462-020-09838-1