Desarrollo de una aplicación móvil para traducción de Lenguaje de Señas Mexicano (LSM)

Dorisol de los Ángeles Pereda Catana
Facultad de Ingeniería Eléctrica y Electrónica
Universidad Veracruzana
Veracruz, México
zS21002408@estudiantes.uv.mx

***Resumen*—El presente proyecto detalla el desarrollo e implementación de una aplicación móvil en Android capaz de traducir gestos de la Lengua de Señas Mexicana (LSM) a texto en tiempo real. La solución integra técnicas de visión por computadora para la extracción de puntos clave (keypoints) utilizando MediaPipe Holistic, y el procesamiento de secuencias temporales mediante una Red Neuronal Recurrente (RNN) con arquitectura LSTM (Long Short-Term Memory). El sistema fue entrenado con un conjunto de datos propio y posteriormente optimizado y convertido a formato TensorFlow Lite para su ejecución eficiente en dispositivos Android. Los resultados experimentales dieron como resultado una precisión del 98.23 % en la etapa de validación del modelo. Sin embargo, las pruebas de despliegue en el dispositivo móvil mostraron variaciones en el rendimiento debido a las condiciones de captura en tiempo real, logrando identificar eficazmente palabras aisladas con niveles de confianza entre el 30 % y 50 %.**

***Index Terms*—LSM, Visión Artificial, RNN, LSTM, Aplicación móvil.**

I. Introducción

El lenguaje de señas es la lengua natural de las personas con discapacidad auditiva, basado en las expresiones faciales y en diversos movimientos de las manos, los brazos y el cuerpo. Cada país cuenta con su propio lenguaje de señas, y el Lenguaje de Señas Mexicano (LSM) es la utilizada en la comunidad de no oyentes en México. El LSM está compuesto por dos técnicas. La primera es la dactilología, que es el deletreo manual de letras para palabras. La segunda son los ideogramas, que representan una palabra con una o varias configuraciones de mano [3] . Tener un conocimiento básico sobre el lenguaje de señas mexicano es crucial para promover la inclusión y la comunicación con aquellas personas no oyentes en México.Es por ello que este trabajo propone el desarrollo de una aplicación móvil capaz de reconocer e interpretar gestos de la LSM en tiempo real. Esta solución tecnológica implementa visión por computadora y redes neuronales recurrentes con arquitectura LSTM para clasificar secuencias de movimiento. El sistema está diseñado para operar en dispositivos Android, permitiendo traducir señas dinámicas a texto utilizando únicamente la cámara del smartphone, sin necesidad de sensores externos ni conexión permanente a internet.

II. Objetivos del Proyecto

- **Objetivo General:** Desarrollar un prototipo de aplicación móvil para la traducción de LSM utilizando redes neuronales recurrentes LSTM.
- **Objetivos Específicos:**
  - Construir un conjunto de datos (dataset) de señas dinámicas normalizadas utilizando extracción de características corporales.
  - Diseñar y entrenar un modelo de clasificación de secuencias temporales basado en arquitectura LSTM.
  - Implementar un sistema de procesamiento en Android que gestione la captura de video, la inferencia del modelo y la visualización de resultados en tiempo real.

III. Estado del Arte

Investigaciones previas han implementado diferentes enfoques para la interpretación de lenguas de señas. Algunos ejemplos son el desarrollo de redes neuronales multicapa (MLP) y convolucionales (CNN) para reconocer el lenguaje de señas boliviano (LSB). Toma como entrada los fotogramas más significativos de un video utilizando un algoritmo basado en movimiento y aplicando un algoritmo de detección de bordes en los fotogramas seleccionados. Como resultado, se encontró que MLP tiene una precisión que varía entre 65 % y 88 %, y CNN varía entre 95 % y 99 %, dependiendo del número de neuronas y capas internas utilizadas. [1] En México, algunos trabajos han desarrollo de una aplicación móvil basada en algoritmos de Deep Learning (aprendizaje profundo), diseñada para reconocer y traducir la Lengua de Señas Mexicana al español, obteniendo el 90 % en la detección de señas fijas y dinámicas en tiempo real. [2] El análisis de los estudios previos sugiere que, si bien los modelos de reconocimiento de lenguas de señas han logrado avances significativos, aún existen desafíos en la implementación de soluciones móviles eficientes para la LSM. El uso de modelos de Deep Learning en dispositivos móviles representa una alternativa prometedora para mejorar la accesibilidad, pero requiere optimización en términos de consumo energético y capacidad de procesamiento.

IV. Diseño e Implementación

IV-A. *Arquitectura del Sistema*

El flujo de información del sistema se divide en cuatro etapas secuenciales:

1. **Captura:** Captura de video mediante la cámara del dispositivo móvil.
2. **Procesamiento:** Detección del cuerpo y manos mediante MediaPipe, seguido de una extracción y normalización de coordenadas (x, y, z).
3. **Predicción:** Una red neuronal LSTM analiza una ventana temporal de frames consecutivos para determinar la probabilidad de cada clase.
4. **Salida:** Interpretación de la predicción y despliegue del resultado en la interfaz de usuario.

IV-B. *Recolección de Datos*

Se generó un dataset propio enfocado en señas dinámicas. El proceso de recolección se realizó mediante un script en Python que automatiza la captura a través de la webcam, con una resolución de 480 x 640 píxeles. Se grabaron **120** secuencias de video por cada palabra (clase). Cada secuencia consta de **45 frames**, lo que permite capturar el inicio, desarrollo y fin de un gesto. No se almacenan imágenes brutas, sino vectores numéricos en formato **.npy**. Se extraen **258** keypoints (puntos clave) correspondiente a la pose y ambas manos. Para garantizar la invariancia a la distancia de la cámara, los datos se normalizaron utilizando el ancho de los hombros como factor de escala y el punto medio entre ellos como origen de coordenadas. De esta manera, se logró obtener los datos correspondientes a un total de 22 palabras: **hola, buenos días, buenas tardes, buenas noches, ¿cómo estás?, nos vemos mañana, ¿quién es?, ¿ya?, adios, bien, bueno, de nada, gracias, mal, no, si, otra vez, por favor, si saber, no saber, yo, nada** (esta palabra es para clasificar movimientos que no son parte del LSM).

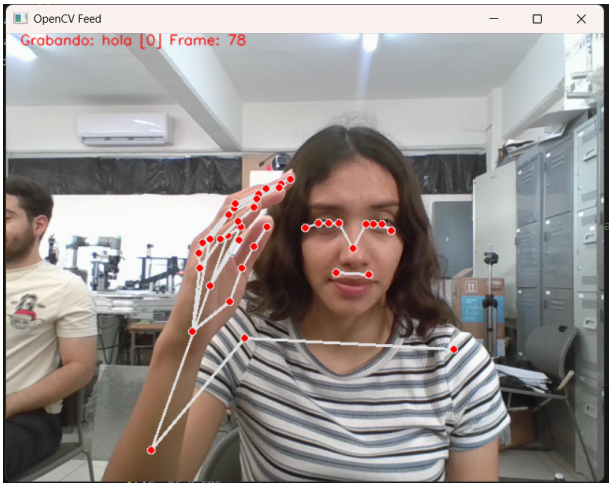

Figura 1. Obtención de dataset con webcam

IV-C. *Entrenamiento del modelo*

Se utilizó la técnica de aumento de datos para multiplicar el tamaño del conjunto de entrenamiento. Por cada secuencia original grabada, el sistema genera automáticamente tres variaciones adicionales, resultando en cuatro versiones por video:

1. **Original:** La secuencia tal cual fue capturada.
2. **Con Ruido:** Se añaden valores aleatorios leves a los puntos clave para simular imperfecciones en la detección de la cámara.
3. **Escalamiento Temporal (Stretch):** Se acelera o ralentiza ligeramente la seña para que el modelo aprenda que la velocidad del gesto no cambia su significado.
4. **Desplazamiento (Shift):** Se mueve la secuencia temporalmente hacia adelante o atrás.

Es importante destacar que esta transformación solo se aplicó a los datos de entrenamiento. El 20 % de los datos reservados para pruebas (Test) se mantuvo intacto para evaluar el modelo con situaciones reales.

El componente principal del sistema es una Red Neuronal Recurrente diseñada con la API de Keras/TensorFlow. La arquitectura seleccionada es una topología LSTM, cuya estructura tiene como entrada las secuencias de keypoints normalizados. Se utilizaron **dos capas LSTM principales** con activacion tahn. La primera (64 neuronas) extrae patrones temporales generales y pasa la secuencia completa a la siguiente capa. La segunda (128 neuronas) condensa la información temporal para la clasificación final. Cada una de estas capas tiene un Dropout de 0.3. También se añade una **capa Densa** con 64 neuronas y activación relú, con un Dropout de 0.4. Después de cada capa LSTM y Densa, se añadieron capas de BatchNormalization para estabilizar el aprendizaje y acelerar la convergencia. El uso de Dropout permite apagar neuronas aleatoriamente durante el entrenamiento, obligando a la red a no depender de un solo camino y evitando que memorice los datos (overfitting). Finalmente, se agrega una capa densa final con activación Softmax que entrega la probabilidad de que el gesto pertenezca a cada una de las clases definidas. Se utilizó el optimizador Adam con una tasa de aprendizaje de 0.0005. El entrenamiento se supervisó mediante EarlyStopping con una paciencia de 15 épocas, monitoreando la pérdida de validación.

IV-D. *Configuraciones en la aplicación móvil*

La integración en Android requirió adaptaciones para replicar el entorno de entrenamiento en un dispositivo móvil. El modelo .keras fue convertido a **TensorFlow Lite (.tflite)**. Debido a la naturaleza dinámica de las capas LSTM, fue necesario configurar el convertidor para utilizar operaciones selectas de TensorFlow (SELECT_TF_OPS), habilitando un modo híbrido que soporta operaciones complejas no nativas de la versión “lite” estándar.

Se utilizó la cámara frontal con una resolución de 640x480, para que coincidiera con la resolución utilizada al obtener los datos para el entrenamiento. Fue crítico implementar una rutina de preprocesamiento de mapas de bits (Bitmap) para

rotar la imagen según los grados de orientación del sensor y aplicar un efecto espejo (flip) para que la experiencia de usuario fuese natural (como un espejo).

Se implementó una lista mutable que actúa como buffer FIFO (First In, First Out) para almacenar siempre los últimos 45 frames de keypoints normalizados y se estableció una ventana de estabilidad de 10 frames, el mismo valor usado en la fase de obtención de datos.

V. Resultados

El modelo alcanzó un rendimiento excelente con una precisión global del **98.23 %**. Las gráficas de aprendizaje (Figura 2 y Figura 3) muestran una convergencia estable y la matriz de confusión (Figura 4) confirma que el sistema aprendió a distinguir correctamente casi todas las clases, con errores mínimos entre señas muy similares.

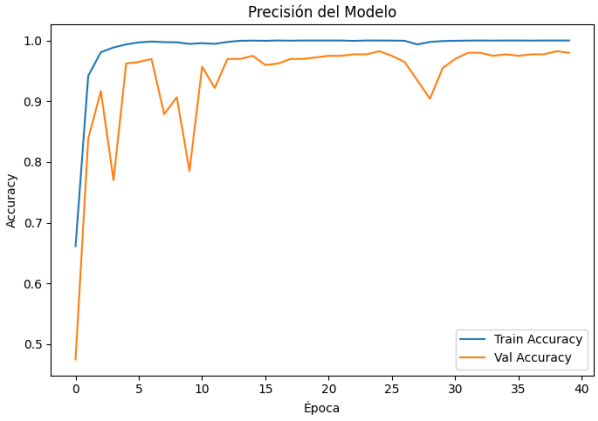

Figura 2. Gráfica de precisión

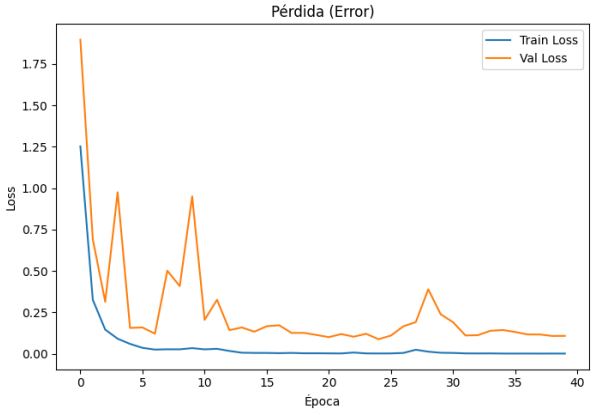

Figura 3. Gráfica de pérdida

Al desplegar el modelo en el dispositivo móvil, el desempeño real disminuyó respecto a las pruebas teóricas. Se identificaron dos problemas principales: El sistema falla al detectar saludos largos como ”buenos días”, ”buenas tardes.º ”buenas noches”, perdiendo la secuencia del movimiento.En palabras simples y cortas (hola, yo, sí), aunque la predicción

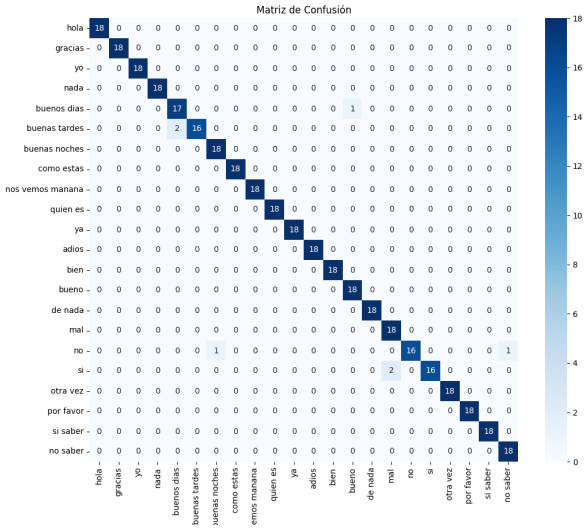

Figura 4. Matriz de confusión

suele ser acertada, el nivel de confianza apenas oscila entre el 30 % y el 50 %. Esto indica que el modelo tiene dificultades para generalizar lo aprendido cuando cambian las condiciones de la cámara (resolución y ángulo) del celular.

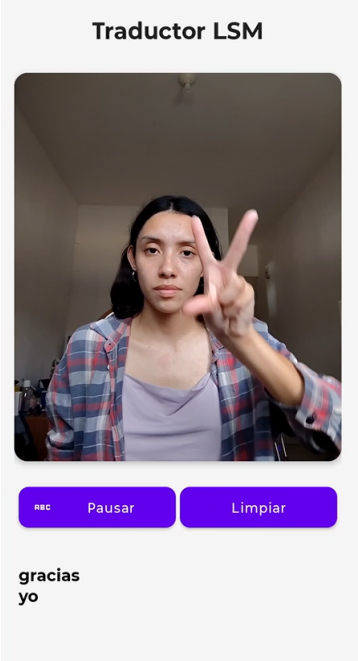

Figura 5. Pruebas en la aplicación móvil

VI. Conclusiones y Trabajo Futuro

En este proyecto se desarrolló e implementó un sistema de traducción de Lengua de Señas Mexicana (LSM) utilizando redes neuronales recurrentes (LSTM) y visión por computadora (MediaPipe), desplegado en una aplicación móvil Android.

Los resultados experimentales demostraron la viabilidad de la arquitectura propuesta. En la etapa de entrenamiento, el mo-

delo alcanzó una precisión del 98.23 %, validando la eficacia de la extracción de puntos clave (keypoints) normalizados para distinguir entre las diferentes clases de señas.

Sin embargo, la implementación en el entorno real presentó desafíos significativos. A pesar del alto rendimiento en el modelo, la aplicación móvil mostró una caída en la robustez de la detección. Las discrepancias en el ángulo de visión, la óptica de la cámara frontal del dispositivo y las condiciones de iluminación provocaron que el modelo operara con niveles de confianza bajos (entre 30 % y 50 %) para palabras simples y fallara en la interpretación de frases con mayor duración. Esto concluye que, aunque la lógica del modelo es correcta, la normalización de datos actual no es suficiente para compensar las diferencias físicas entre una webcam de escritorio y un dispositivo móvil.

Para solucionar las limitaciones detectadas y mejorar la usabilidad del sistema, se proponen las siguientes líneas de acción:

1. Generar el conjunto de datos directamente utilizando la cámara del dispositivo móvil en lugar de una webcam. Esto eliminará la brecha entre el entrenamiento y la inferencia, mejorando los niveles de confianza.
2. Entrenar el modelo con una mayor variabilidad de fondos, distancias e iluminación, aplicando técnicas de Data Augmentation más agresivas que simulen el movimiento natural de un teléfono sostenido en la mano.

REFERENCIAS

[1] Cuevas, R., Salgado, G., "Aplicación Móvil Basada en Deep Learning para la Inclusión Educativa de Personas Sordas: Reconocimiento y Traducción de la Lengua de Señas Mexicana," Revista Iberoamericana para la Investigación y el Desarrollo Educativo, Diciembre 2025.

[2] Serafin, M., González, R, "Manos con voz. Diccionario de lengua de señas.", SEP, 2011.

[3] Rodríguez, J., Ponce de León, N., Arteaga, W, "Multilayer and convolutional neural networks for Bolivian Sign Language recognition: an empirical evaluation.",SCIELO, Marzo 2021.