Reconocimiento de Lengua de Señas Utilizando Aprendizaje Profundo

Juan Fernando Cardona Ruiz
Departamento de Ingeniería de Sistemas
Universidad de Antioquia
Medellín, Colombia
juan.cardona32@udea.edu.co

Daniela Vásquez Londoño
Departamento de Ingeniería de Sistemas
Universidad de Antioquia
Medellín, Colombia
daniela.vasquezl1@udea.edu.co

***Abstract*—This project focuses on developing an American Sign Language (ASL) recognition system using computer vision technologies such as OpenCV and MediaPipe, along with deep learning techniques implemented in TensorFlow and Keras. Two datasets were employed to train a classification model that achieved an accuracy of 97.89% in detecting hand gestures corresponding to letters of the sign language alphabet. The application allows real-time interaction to form words through recognized gestures, with the ability to save the generated words in a text file. The system can be useful in facilitating communication for individuals with hearing disabilities.**

Keywords*—*Sign Language, hand gesture recognition, Deep Learning, OpenCV and MediaPipe, TensorFlow and Keras, real-time interaction.

***Resumen*—Este proyecto se enfoca en desarrollar un sistema de reconocimiento de Lengua de Señas Americana (ASL) utilizando tecnologías de visión por computadora como OpenCV y MediaPipe, junto con técnicas de aprendizaje profundo implementadas en TensorFlow y Keras. Se emplearon dos conjuntos de datos para entrenar un modelo de clasificación que logró una precisión del 97.89% en la detección de gestos de manos correspondientes a letras del alfabeto de señas. La aplicación permite la interacción en tiempo real para formar palabras mediante gestos reconocidos, con la capacidad de guardar las palabras generadas en un archivo tipo texto. El sistema puede ser de utilidad para facilitar la comunicación de personas con discapacidad auditiva.**

Palabras clave*—*Lengua de Señas, reconocimiento de gestos manuales, Deep Learning, OpenCV y MediaPipe, TensorFlow y Keras, interacción en tiempo real.

I. INTRODUCCIÓN

La lengua de señas es una forma importante de comunicación para las personas con discapacidad auditiva. Sin embargo, la comunicación efectiva puede ser un desafío en entornos donde no hay intérpretes disponibles. En este contexto, el desarrollo de sistemas automáticos de reconocimiento de lengua de señas puede brindar una solución invaluable.

Por tal motivo, este proyecto se centra en el desarrollo de un sistema de reconocimiento de lengua de señas, específicamente orientado a reconocer letras de la Lengua de Señas Americana (ASL). Para lograr este objetivo, se emplearon tecnologías de visión por computadora y técnicas de aprendizaje profundo. La combinación de OpenCV [1] y MediaPipe [2] permite la detección precisa de gestos de manos en tiempo real, mientras que TensorFlow [3] y Keras [4] facilitan el entrenamiento de un modelo de clasificación para reconocer letras del alfabeto de señas.

En este informe, se presentarán los detalles del proyecto, incluida la descripción de la metodología utilizada, el desempeño del modelo, la estructura del código y las funcionalidades implementadas. Además, se discutirán las posibles aplicaciones y beneficios de este sistema, así como las oportunidades para futuras mejoras y expansiones.

II. METODOLOGÍA

A. Descripción del conjunto de datos

Se emplearon dos conjuntos de datos para entrenar el modelo de clasificación. Uno de ellos es el conjunto de datos del lenguaje de señas MNIST [5], que consta de 24 clases que representan las letras del alfabeto de la Lengua de Señas estadounidense (ASL), excluyendo las letras J y Z que requieren movimiento. Cada clase representa una letra específica del alfabeto y consiste en imágenes en escala de grises de 28x28 píxeles. Este dataset proporciona un reto adicional para los métodos de aprendizaje automático basados en imágenes, ya que las letras del lenguaje de señas tienen formas y características únicas que deben ser reconocidas por el modelo.

El segundo conjunto de datos utilizado es el Spanish Sign Language Alphabet (Static) [6], que también consiste en imágenes en escala de grises de 28x28 píxeles. Este conjunto de datos representa el alfabeto del lenguaje de señas español y consta de 19 clases diferentes. Estas imágenes representan gestos estáticos de las manos que corresponden a letras específicas.

B. Descripción de la estructura de los notebooks

El proyecto se estructura en diferentes directorios y archivos para mantener una organización clara y coherente. En el directorio principal, se encuentran varios subdirectorios y archivos, cada uno con un propósito específico en el desarrollo

y la implementación del sistema de reconocimiento de lengua de señas.

En primer lugar, el directorio **data** almacena todos los datos relevantes utilizados en el proyecto. Esto incluye conjuntos de datos de entrenamiento, validación y prueba, así como cualquier otro archivo de datos necesario para el preprocesamiento y la construcción del modelo.

El directorio **models** contiene los modelos entrenados que se guardan después del proceso de entrenamiento. Estos modelos son archivos en formato h5 que representan la arquitectura y los pesos del modelo después del entrenamiento.

El directorio **out** se utiliza para almacenar cualquier resultado o salida generado durante el proceso de implementación o evaluación del sistema. Esto puede incluir imágenes generadas, archivos de texto, o cualquier otro tipo de salida relevante.

El archivo **.gitignore** específica los archivos y directorios que deben ser ignorados por el control de versiones Git, lo que ayuda a mantener el repositorio limpio y organizado.

Además, se encuentran otros archivos como **README.md**, **licence.md**, y **requirements.txt** que proporcionan información adicional sobre el proyecto, la licencia de uso y las dependencias del entorno, respectivamente.

Por último, pero no menos importante, los archivos **Preproceso.ipynb**, **Modelo.ipynb** e **Implementacion.ipynb** son cuadernos Jupyter que contienen el código fuente utilizado para el preprocesamiento de datos, la construcción y entrenamiento del modelo, y la implementación del sistema, respectivamente. A continuación, se presenta una descripción detallada de cada uno:

1) Preproceso

En el cuaderno de preprocesamiento (01 Preproceso.ipynb), se realizan una serie de tareas críticas para preparar los datos antes de su alimentación al modelo de aprendizaje automático. Esto implica la carga inicial de conjuntos de datos y la exploración para comprender su estructura y distribución. Además, se llevan a cabo transformaciones necesarias, como redimensionamiento, normalización o codificación de etiquetas.

Dentro de este cuaderno, se abordan problemas específicos del conjunto de datos, como la manipulación de imágenes, el procesamiento de texto y la limpieza de datos. También se incluyen visualizaciones que ayudan a comprender los datos y detectar posibles anomalías que puedan afectar el rendimiento del modelo.

En cuanto al procesamiento de imágenes, se realizan pasos críticos como la normalización para garantizar una representación coherente de los datos. Esto implica operaciones como la eliminación de ruido, la corrección de color y el ajuste de la intensidad. Además, se lleva a cabo la segmentación de la mano para aislarla del fondo, lo que facilita la extracción de características.

Para ilustrar estos procesos, se muestran dos imágenes generadas durante el preprocesamiento. La Fig. 1 muestra el proceso de normalización, incluidas las operaciones de suavización, conversión a escala de grises y normalización de intensidad, junto con histogramas de cada canal de color antes y después del proceso.

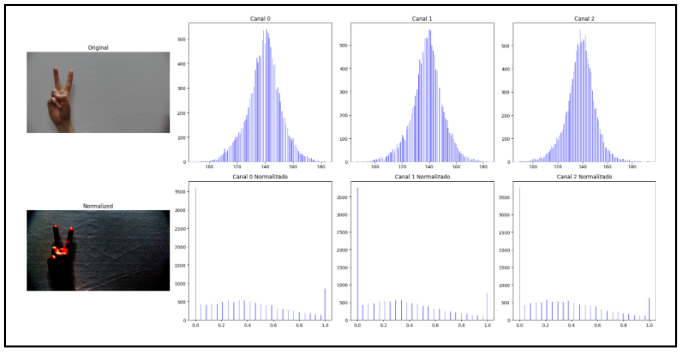

Fig. 1. Histograma de normalización de imagen.

La Fig. 2 muestra el resultado del proceso de segmentación de manos, utilizando un enfoque basado en umbrales y operaciones morfológicas, junto con características extraídas como descriptores locales (HOG) [7] y características de textura (GLCM) [8].

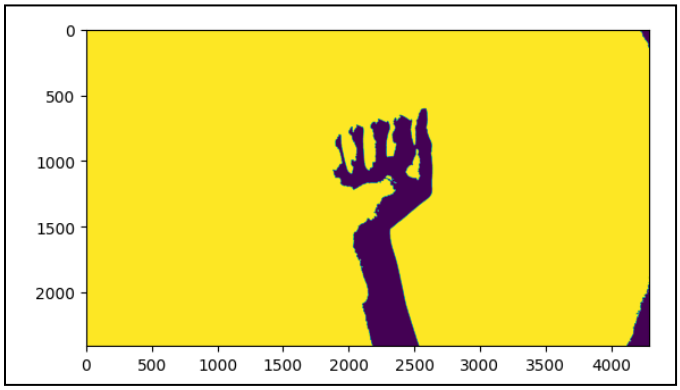

Fig. 2. Segmentación de manos y extracción de características.

Estas imágenes proporcionan una visión detallada de los pasos de preprocesamiento, ayudando a comprender cómo se preparan los datos antes de alimentar al modelo de aprendizaje automático.

2) Modelo

En el cuaderno del modelo (02 Modelo.ipynb), se llevan a cabo diversas etapas cruciales en el desarrollo y evaluación del sistema de reconocimiento de lengua de señas. Este incluye desde la carga y preprocesamiento de los datos hasta la definición, entrenamiento y evaluación del modelo de red neuronal convolucional (CNN). También se implementan funciones para visualizar métricas clave como la precisión y la pérdida durante el entrenamiento y la validación, así como la matriz de confusión para evaluar el rendimiento del modelo.

El modelo de CNN se ha diseñado específicamente para reconocer letras de la Lengua de Señas Americana (ASL), centrándose en las letras correspondientes a las señas ABCDEFGIKLMNOPQRSTU. Su arquitectura consta de capas de convolución y pooling seguidas de capas totalmente conectadas para la clasificación. Se emplea la técnica de

optimización Adam y la función de pérdida de entropía cruzada categórica durante el entrenamiento.

Adicionalmente, se han asignado funciones especiales a las letras 'Q' , 'O' y 'B'. La letra 'Q' permite borrar la última letra ingresada, mientras que la letra 'O' se utiliza para insertar un espacio. Por otro lado, la letra 'B' desempeña la función de guardar el texto reconocido en un archivo para su posterior referencia o uso. Estas asignaciones permiten una interacción más fluida y eficiente con el sistema, facilitando su utilidad y accesibilidad para los usuarios.

Para comenzar, se transforman los datos de entrada en un formato apto para el entrenamiento del modelo, convirtiendo cada registro en una imagen y realizando la codificación one-hot [9] de las etiquetas. Luego, se dividen los datos en conjuntos de entrenamiento, validación y prueba.

Una vez preparados los datos, se exploran visualmente muestras aleatorias de imágenes por clase (Fig. 3), lo que permite apreciar la diversidad de gestos presentes en el conjunto de datos.

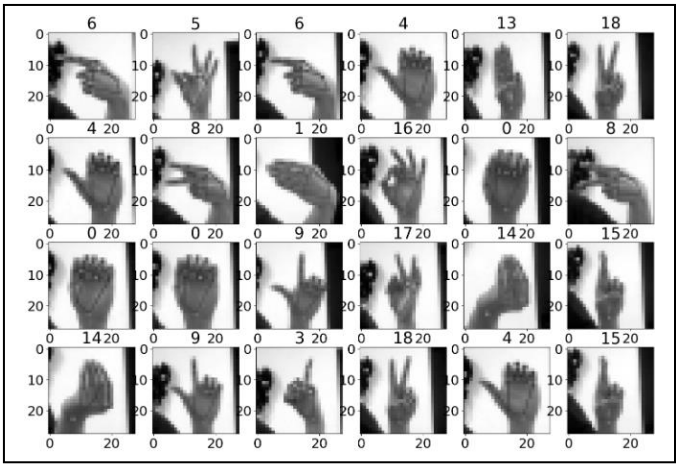

Fig. 3. Muestra aleatoria por clase.

Luego, se profundiza en la distribución de clases en el conjunto de datos mediante un gráfico de barras que muestra la cantidad de imágenes disponibles para cada clase (Fig. 4). Esta representación gráfica resulta fundamental para evaluar el equilibrio del conjunto de datos y comprender si algunas clases están sobrerrepresentadas o subrepresentadas, lo cual podría tener un impacto significativo en el rendimiento del modelo.

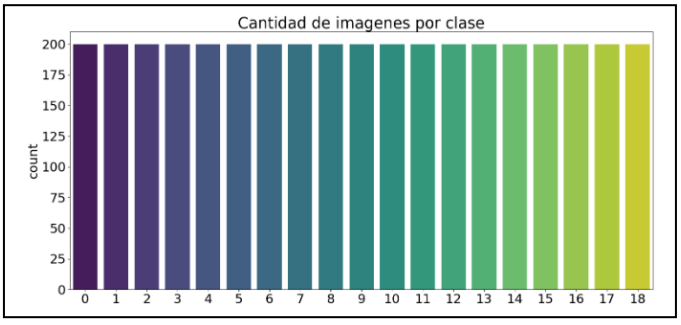

Fig. 4. Distribución de clases.

Continuando con el análisis, se observa la evolución de la precisión del modelo a lo largo del proceso de entrenamiento (Fig. 5). Esta representación gráfica permite seguir el progreso del modelo y evaluar su capacidad para aprender y generalizar patrones en los datos de entrenamiento y validación.

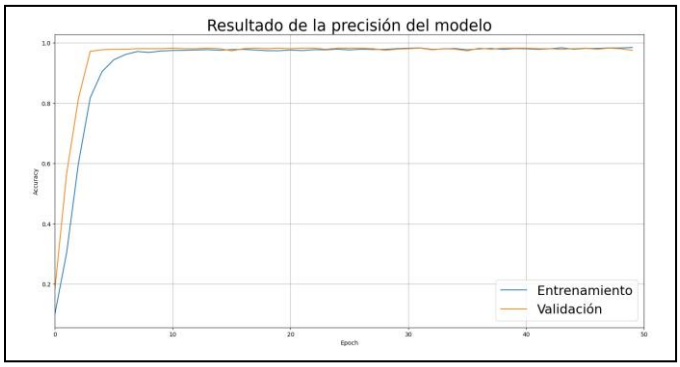

Fig. 5. Precisión del modelo durante el entrenamiento.

Así mismo, se examina la pérdida del modelo durante el entrenamiento (Fig. 6), lo que proporciona información sobre cómo la pérdida del modelo cambia a medida que el modelo se ajusta a los datos de entrenamiento y validación. Esta visualización ayuda a identificar posibles problemas de sobreajuste o subajuste que puedan surgir durante el proceso de entrenamiento.

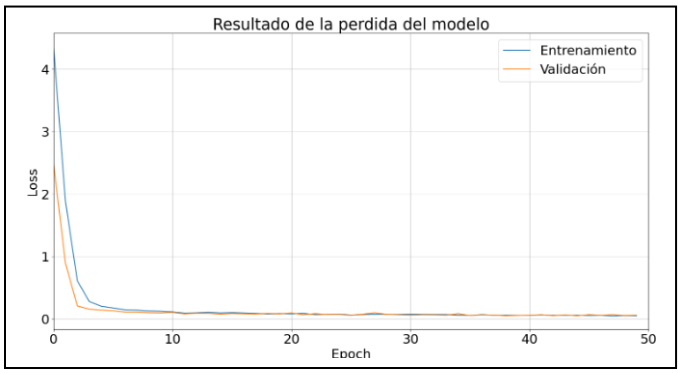

Fig. 6. Pérdida del modelo durante el entrenamiento.

Finalmente, se explora la matriz de confusión del modelo en el conjunto de prueba (Fig. 7). Esta matriz brinda información detallada sobre cómo se clasifican las muestras en cada clase, lo que ayuda a comprender el rendimiento del modelo e identificar áreas específicas donde el modelo puede estar teniendo dificultades.

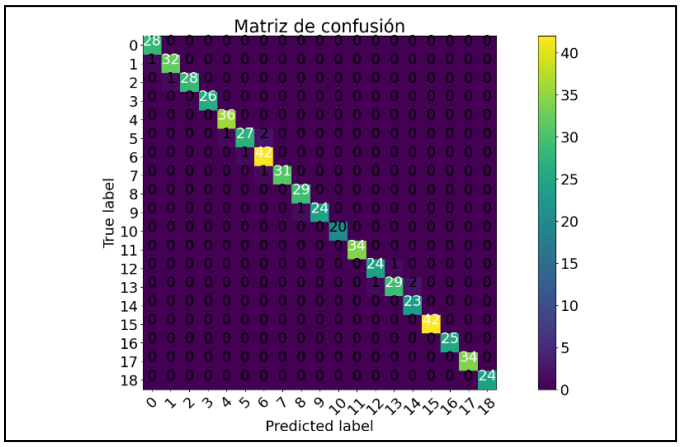

Fig. 7. Matriz de confusión del modelo.

En conjunto, estas visualizaciones ofrecen una comprensión completa del conjunto de datos y del rendimiento del modelo, permitiendo tomar decisiones informadas y mejorar continuamente su enfoque de reconocimiento de lengua de señas.

3) Implementación

En el cuaderno de implementación (03 Implementacion.ipynb), se lleva a cabo la integración del modelo entrenado en un entorno de producción. Aquí, se establecen las conexiones con la cámara en tiempo real o se cargan datos de video pregrabados, según sea el caso. Se utiliza el modelo previamente entrenado para realizar la inferencia sobre los datos de entrada y generar las predicciones correspondientes.

Este cuaderno abarca desde la carga del modelo preentrenado hasta la detección de gestos de manos en tiempo real. Para comenzar, se carga el modelo desde un archivo almacenado en la ruta especificada. Además, se inicializa el modelo de detección de manos de MediaPipe, lo que permite capturar y procesar los gestos de las manos en tiempo real.

El siguiente paso implica la captura de los fotogramas de video y su procesamiento para detectar y clasificar los gestos de la lengua de señas (Fig. 8). Para ello, se utiliza el modelo de detección de manos de MediaPipe junto con el modelo de reconocimiento previamente entrenado. Cada 15 fotogramas se captura un nuevo fotograma y se clasifican los gestos presentes en él. Luego, se guardan 200 fotogramas por letra en una carpeta específica, lo que permite recopilar datos para entrenar y mejorar el modelo en el futuro.

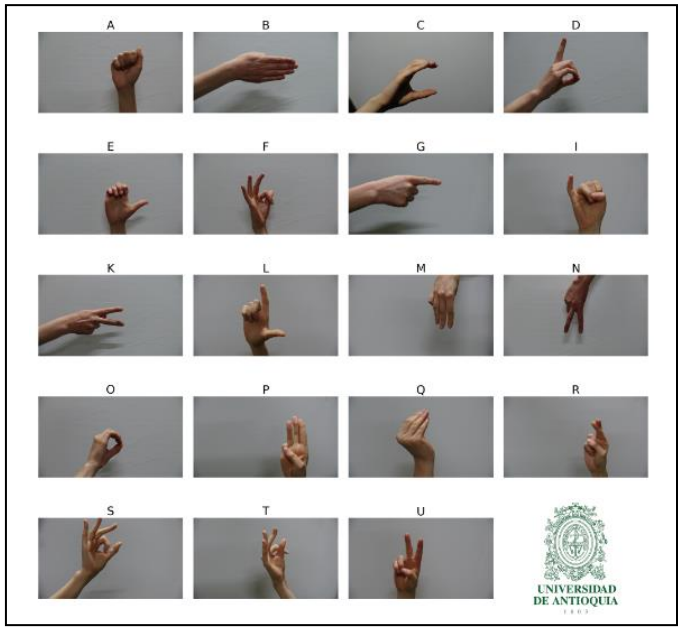

Fig. 8. Captura y procesamiento de gestos de lengua de señas.

En resumen, la implementación de este sistema de reconocimiento de lengua de señas combina el uso de modelos preentrenados con la detección de gestos de manos en tiempo real, permitiendo la captura y clasificación eficiente de los gestos de la lengua de señas en entornos dinámicos.

En síntesis, los cuadernos de preprocesamiento, modelo e implementación son esenciales para el desarrollo del sistema de reconocimiento de lengua de señas. El primero se encarga de preparar los datos, el segundo construye y entrena el modelo, y el tercero integra el modelo en un entorno práctico. Colectivamente, abordan aspectos críticos desde la preparación de datos hasta su implementación funcional.

III. RESULTADO

El proyecto desarrollado se centra en la creación de un sistema de reconocimiento de lengua de señas que opera en tiempo real. Utilizando una cámara, el sistema captura frames de video, detecta los puntos de referencia de la mano y clasifica los gestos en letras del alfabeto de la lengua de señas. Este proceso permite que las letras reconocidas se muestren en pantalla, posibilitando la interacción en tiempo real para formar palabras y guardarlas en un archivo de texto.

Durante el desarrollo del proyecto, se obtuvieron dos imágenes que ilustran el funcionamiento del sistema. La Fig. 9 muestra el inicio del proceso de reconocimiento de gestos cuando se prueba inicialmente el modelo. En esta etapa, el sistema comienza a capturar frames de video y detectar los puntos de referencia de la mano.

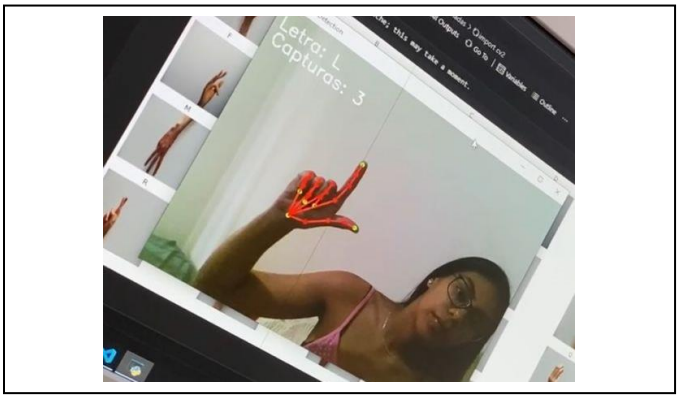

Fig. 9. Inicio del proceso de reconocimiento de gestos.

La Fig. 10 muestra la detección de la mano en un recuadro y la clasificación de la señalización correspondiente. Esta imagen ilustra el proceso de identificación de los gestos de la lengua de señas en tiempo real.

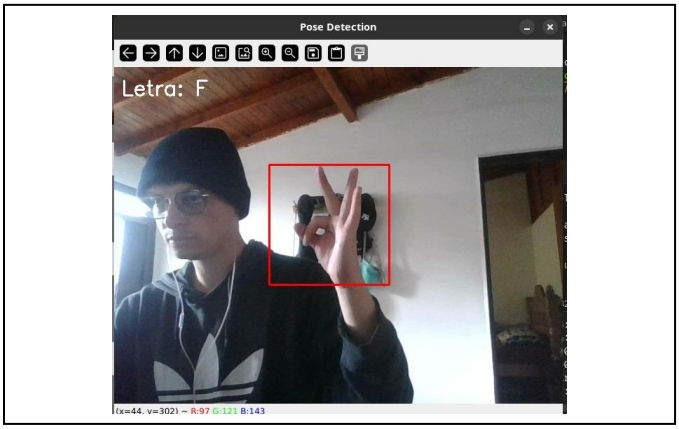

Fig. 10. Detección y clasificación de señas.

Por otro lado, la Fig. 11 representa la vista final del proyecto cuando se ejecuta completamente. Aquí se muestra la detección de los puntos de referencia de la mano y el almacenamiento de las palabras identificadas en el archivo de texto. Esta imagen destaca el paso final del proceso, donde las palabras formadas a partir de los gestos reconocidos se guardan para su posterior uso.

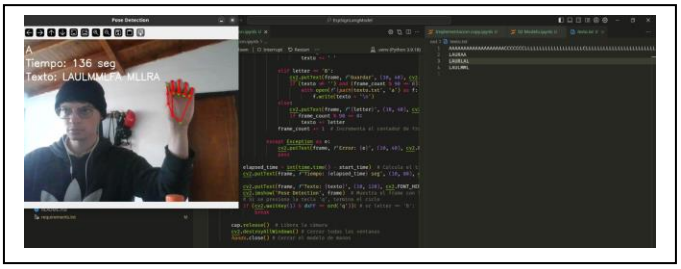

Fig. 11. Vista final del proyecto en ejecución.

IV. CONCLUSIÓN

En conclusión, el desarrollo de este proyecto nos ha brindado una oportunidad invaluable para desafiarnos a nosotros mismos y explorar nuevas tecnologías. El proceso de aprendizaje continuo involucrado en la implementación de un sistema de reconocimiento de lengua de señas nos ha permitido expandir nuestros conocimientos y habilidades técnicas, así como mejorar nuestras habilidades blandas, como la comunicación y la colaboración en equipo. Además, al centrarnos en un proyecto que busca abordar las necesidades de la sociedad, hemos reafirmado nuestro compromiso con la creación de soluciones tecnológicas significativas y orientadas al impacto social. El trabajo en equipo desempeñó un papel crucial en el éxito de este proyecto, destacando la importancia de la colaboración y la coordinación efectiva para lograr nuestros objetivos comunes. Este proyecto está licenciado bajo la Licencia MIT, lo que refleja nuestro compromiso con la disponibilidad y accesibilidad de nuestro trabajo para otros desarrolladores y la comunidad en general.

V. TRABAJO FUTURO

Hay varias áreas que podrían explorarse para mejorar y expandir este proyecto. Una de las posibilidades es la optimización del rendimiento del modelo de reconocimiento de lengua de señas, ya sea mediante la mejora de la arquitectura de la red neuronal convolucional o la exploración de técnicas de preprocesamiento de datos más avanzadas. Además, se podrían incorporar nuevas funcionalidades al sistema, como la detección de gestos más complejos o la traducción automática de las señas a texto o voz.

Otro aspecto importante a considerar es la accesibilidad del sistema, asegurándose de que sea fácil de usar para personas con discapacidades auditivas. Esto podría implicar la creación de una interfaz más intuitiva y amigable, así como la optimización del sistema para diferentes dispositivos y entornos.

En resumen, hay muchas oportunidades para continuar desarrollando y mejorando este proyecto en el futuro. Invitamos a otros a contribuir al proyecto y a participar en su evolución, ya que creemos en el valor de la colaboración y la contribución abierta a la comunidad de código abierto. Si se presentan oportunidades para recibir mejoras o retomar el proyecto en el futuro, estaremos encantados de hacerlo para seguir avanzando hacia nuestro objetivo inicial de ayudar a las personas que presentan discapacidades auditivas.

REFERENCES

[1] Á. Torrenti, «Procesamiento de imágenes con OpenCV en Python», *Imagina Formación*, 2 de junio de 2024. [En línea]. Disponible en: https://imaginaformacion.com/tutoriales/opencv-en-python

[2] C. Writer, «What is MediaPipe? A Guide for Beginners», *Roboflow Blog*, 15 de abril de 2024. [En línea]. Disponible en: https://blog.roboflow.com/what-is-mediapipe/

[3] J. Larkin Alonso, «¿Qué es TensorFlow y para qué sirve?», *Incentro*, 15 de junio de 2022. [En línea]. Disponible en: https://www.incentro.com/es-ES/blog/que-es-tensorflow

[4] DataScientest, «Keras: todo sobre la API de Deep Learning», *DataScientest*, 30 de octubre de 2023. [En línea]. Disponible en: https://datascientest.com/es/keras-la-api-de-deep-learning

[5] «Sign Language MNIST», *Kaggle*, 20 de octubre de 2017. [En línea]. Disponible en: https://www.kaggle.com/datasets/datamunge/sign-language-mnist/data

[6] «Spanish Sign Language Alphabet (Static)», *Kaggle*, 22 de julio de 2022. [En línea]. Disponible en: https://www.kaggle.com/datasets/kirlelea/spanish-sign-language-alphabet-static

[7] WordPress, «HOG Histograma de gradientes orientados», *WordPress*, 5 de diciembre de 2017. [En línea]. Disponible en: https://carlosjuliopardoblog.wordpress.com/2017/12/04/hog-histograma-de-gradientes-orientados/

[8] Pavel Note, «Implementación de GLCM en PyTorch para el análisis de texturas», *Pavel Note*, 22 de febrero de 2024. [En línea]. Disponible en: https://pavelnote.com/implementaci%C3%B3n-de-glcm-en-pytorch-para-el-an%C3%A1lisis-de-texturas/

[9] J. M. Alvarez, «Categorías y la codificación One-Hot», *Blog de Jose Mariano Alvarez*, 15 de marzo de 2018. https://blog.josemarianoalvarez.com/2018/03/15/categorias-y-la-codificacion-one-hot