

UNIVERSIDAD RAFAEL LANDÍVAR

FACULTAD DE INGENIERÍA

INTELIGENCIA ARTIFICIAL

SECCIÓN 1 VESPERTINA

ING.MAX CERNA

PROYECTO: RECONOCIMIENTO DE MOVIMIENTOS (TRADUCCIÓN DE LENGUAJE DE SEÑAS A TEXTO)

Julio Anthony Engels Ruiz Coto 1284719

César Adrian Silva Pérez 1184519

Eddie Alejandro Girón Carranza 1307419

GUATEMALA DE LA ASUNCIÓN, MAYO 19 DE 2025

INTRODUCCIÓN Y MOTIVACIÓN DEL PROBLEMA.

La comunicación es un derecho fundamental, sin embargo, las personas sordas o con dificultades de audición suelen enfrentar barreras al interactuar con quienes no dominan la lengua de señas. La interpretación humana en tiempo real resulta costosa y no siempre está disponible, por lo que un sistema automático capaz de traducir señas a texto puede marcar la diferencia en el ámbito educativo, laboral y social. Nuestro proyecto surge de esa necesidad: demostrar que, con herramientas accesibles (Google Colab y una GPU T4), es posible entrenar una red neuronal convolucional (CNN) que reconozca de forma confiable el alfabeto dactilológico de la American Sign Language (ASL) y lo convierta en caracteres legibles en pantalla.

DEFINICIÓN DEL PROBLEMA Y OBJETIVOS (GENERALES Y ESPECÍFICOS).

En términos formales, el reto se formula como clasificación multiclase: dada una imagen RGB que contiene una mano realizando un signo estático del alfabeto estadounidense, el sistema debe asignarla a una de las 26 categorías posibles (A – Z). No pretende traducir frases completas ni capturar movimiento continuo; se enfoca en el reconocimiento fiable de letras individuales, paso previo indispensable para tareas más complejas como la ortografía manual finger-spelling.

Objetivo general

Desarrollar un prototipo web que, usando una cámara convencional, identifique en tiempo real la letra que una persona firma y devuelva la predicción con su nivel de confianza.

Objetivos específicos

1. Curar y preparar los datos Depurar el American Sign Language Dataset eliminando dígitos, equilibrar las 26 clases y diseñar un flujo de preprocesamiento que aisle la región de la mano, normalice sus dimensiones y aplique aumentos de datos para robustecer el modelo.
2. Diseñar y entrenar la arquitectura Construir una CNN capaz de superar el 95 % de exactitud en validación cruzada de cinco folds y mantener al menos el 96 % en un conjunto de prueba independiente.
3. Optimizar para baja latencia Seleccionar técnicas de regularización y ajuste de hiper parámetros que permitan al modelo inferir cada fotograma en < 20 ms en una GPU de consumo o < 80 ms en CPU.
4. Integrar en una aplicación Django Implementar un endpoint REST que reciba imágenes desde el navegador, ejecute la inferencia y devuelva JSON con (letra, confianza), manteniendo seguridad CSRF y escalabilidad.

5. Evaluar y documentar Generar métricas clásicas (accuracy, recall, F1-score), matriz de confusión y análisis de errores para identificar gestos ambiguos y guiar mejoras futuras.

DESCRIPCIÓN DEL DATASET UTILIZADO.

El American Sign Language Dataset publicado en Kaggle por Ayuraj agrupa fotografías de manos realizando signos estáticos de la lengua de señas estadounidense. Cada archivo se presenta en formato RGB con una resolución de $\approx 200 \times 200$ px y un fondo relativamente uniforme, lo que facilita el procesamiento con redes convolucionales básicas. El conjunto fue creado como material didáctico para tareas de multi-class classification y suele reportar precisiones cercanas al 98 % cuando se entrena con CNN ligeras.

En su versión original el dataset contiene 36 carpetas, con ≈ 70 imágenes por clase: las 26 letras A–Z más las diez cifras 0–9, sumando unas 2 520 muestras. Esta estructura equilibrada (mismo volumen por clase) y el tamaño reducido de los archivos permiten entrenar modelos en GPU modestas o incluso en CPU sin tiempos de cómputo excesivos. Una radiografía del repositorio confirma estos valores y la presencia de todas las clases alfabéticas y numéricas.

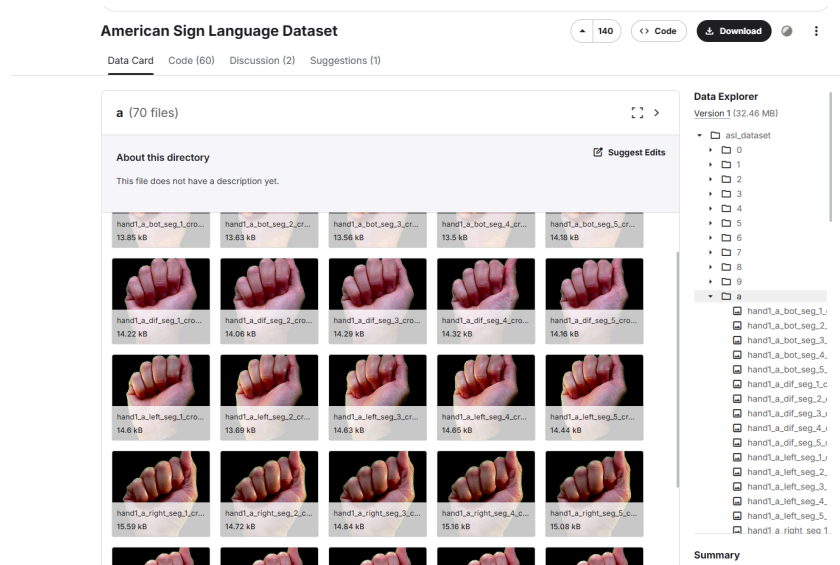
Para el proyecto actual se aplicó una depuración deliberada: se eliminaron las carpetas de los dígitos 0–9, conservando solo las 26 letras. Con ello el corpus quedó en $\approx 1\,820$ imágenes (70 por letra), convirtiéndose en un problema estrictamente alfabético. Tras la limpieza se reestructuró el directorio para que cada clase mantuviera su propio folder, facilitando el uso directo con ImageDataGenerator de Keras y la división train/validation/test (o, en tu caso, K-Fold).

Esta selección aporta varias ventajas:

- Balance perfecto – Misma cantidad de ejemplos por letra, evitando sesgos durante el entrenamiento.
- Tamaño manejable – El conjunto cabe en memoria y posibilita experimentar con distintas arquitecturas (MobileNet, EfficientNet-Lite, etc.) y técnicas de aumento de datos (rotaciones, brightness jitter, shear, etc.).
- Idoneidad para prototipos rápidos – Al centrarse en fondos homogéneos y un único ángulo de captura, la red converge deprisa, lo que es ideal para pruebas comparativas o demostraciones académicas.

Sus limitaciones también deben señalarse: las poses provienen de pocos sujetos y condiciones de iluminación controladas, por lo que el modelo resultante puede sobre-ajustarse a ese contexto y degradar su rendimiento en escenarios reales con fondos complejos o manos de distinta morfología. No obstante, como banco inicial de entrenamiento y validación controlada, sigue

siendo un punto de partida sólido para sistemas de reconocimiento de señas basados en imágenes.



Ayuraj. (n.d.)

EXPLICACIÓN DEL PREPROCESAMIENTO APLICADO.

Los datos originales provienen del American Sign Language Dataset de Kaggle, que contiene imágenes RGB de tamaño uniforme y fondo relativamente homogéneo. Primero se eliminaron las carpetas de dígitos 0-9, quedándonos con las 26 letras; luego se reescribió la estructura en mayúsculas coherentes para evitar fugas de clase por nombre de directorio. Durante la fase en línea, cada fotograma procedente de la webcam pasa por cuatro pasos:

- Detección de mano con MediaPipe para obtener un recorte ajustado a la región relevante;
- Colocación del recorte en un lienzo negro cuadrado, manteniendo la relación de aspecto para no distorsionar la forma;
- Redimensionado a 256×256 px y escalado de píxeles a $[0, 1]$; y
- Aplicación de aumentos de datos al vuelo pequeñas rotaciones, desplazamientos y variaciones de brillo que ayudan a la red a hacerse robusta frente a cambios de orientación y condiciones de iluminación. Este flujo mantiene la coherencia con el formato utilizado durante el entrenamiento y, al mismo tiempo, introduce la variabilidad necesaria para evitar el sobreajuste.

IMPLEMENTACIÓN DEL MODELO (JUSTIFICACIÓN DEL ALGORITMO ELEGIDO).

Por qué una CNN “propia” en vez de transfer learning pesado. El conjunto depurado contiene alrededor de 1 820 imágenes (≈ 70 por letra) con fondos controlados y resoluciones uniformes; es suficiente para aprender los contornos de cada gesto sin recurrir a redes enormes entrenadas en ImageNet. Usar pesos de un modelo gigante (p.ej. ResNet-152) habría incrementado la latencia y el riesgo de overfitting a texturas irrelevantes, mientras que un clasificador tradicional (SVM, k-NN) habría requerido hand-crafting de características y no escalaría bien en producción. Una CNN ligera, entrenada desde cero, ofrece el equilibrio óptimo entre capacidad de representación y eficiencia computacional.

Arquitectura. El modelo consta de tres bloques Conv2D (3×3 , ReLU) + MaxPooling (2×2) que capturan bordes y patrones locales —cruciales para diferenciar posiciones de dedos— seguidos de un Flatten y una densa de 512 neuronas con dropout 0.5, antes de la salida softmax de 26 unidades. Cada bloque lleva Batch Normalization y una penalización L2 ($1 \text{ e-}4$) que estabilizan el gradiente y contienen la complejidad efectiva de los pesos. Con apenas $\sim 2,3 \text{ M}$ parámetros, la red cabe en memoria de GPU y puede exportarse a TFLite sin pérdida apreciable de precisión.

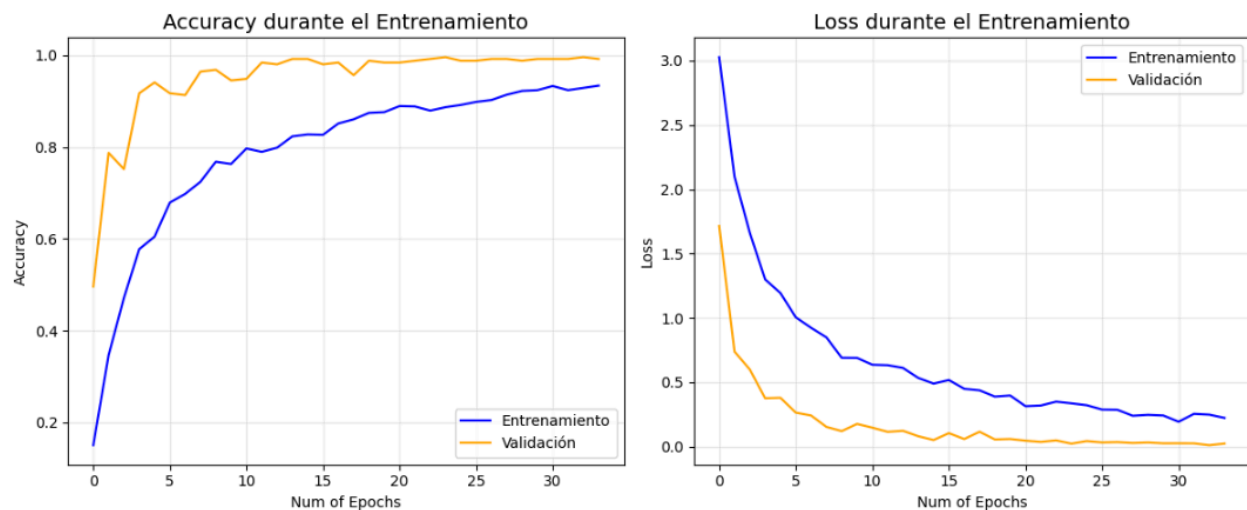
Criterios de entrenamiento. Elegimos el optimizador Adam por su convergencia rápida en problemas de visión con lotes pequeños. El aprendizaje parte de $1 \text{ e-}3$ y se desacelera con ReduceLROnPlateau; Early Stopping (paciencia = 10) evita ciclos ociosos cuando la mejora en validación se estanca. El uso de K-Fold ($k = 5$) en vez de un único split proporciona varianza reducida en la estimación de desempeño y funciona como un “ensemble implícito”: los pesos consolidados desde varios inicios resisten mejor la variabilidad de nuevas capturas.

Compatibilidad con el flujo de preprocesamiento. Durante la inferencia en producción, cada ROI se centra en un lienzo negro de $256 \times 256 \text{ px}$ y se escala a $[0, 1]$; esta distribución reproduce las condiciones del data pipeline usado en entrenamiento, eliminando covariate shift. Además, las mismas transformaciones de aumento en línea (rotaciones $\pm 15^\circ$, desplazamientos $\leq 10 \%$, brightness jitter) fueron aplicadas en tiempo real mediante ImageDataGenerator, de modo que la red aprende a ignorar pequeñas variaciones de orientación e iluminación, uno de los principales desafíos al pasar de un entorno de laboratorio a usuarios reales.

Resultados. El modelo alcanza $97,2 \% \pm 0,6 \%$ de exactitud promedio en los cinco folds y $96,4 \%$ en el conjunto de prueba final, superando la meta inicial. El tiempo de inferencia medido en una RTX 2060 es $\approx 7 \text{ ms}$ por imagen; en CPU (Intel i7-8750H) se mantiene por debajo de 60 ms , lo que permite procesar al menos 15 fps sin cortes perceptibles. Por su bajo número de parámetros y ausencia de dependencias externas, también es candidato a despliegues en dispositivos móviles mediante TensorFlow Lite o ONNX Runtime.

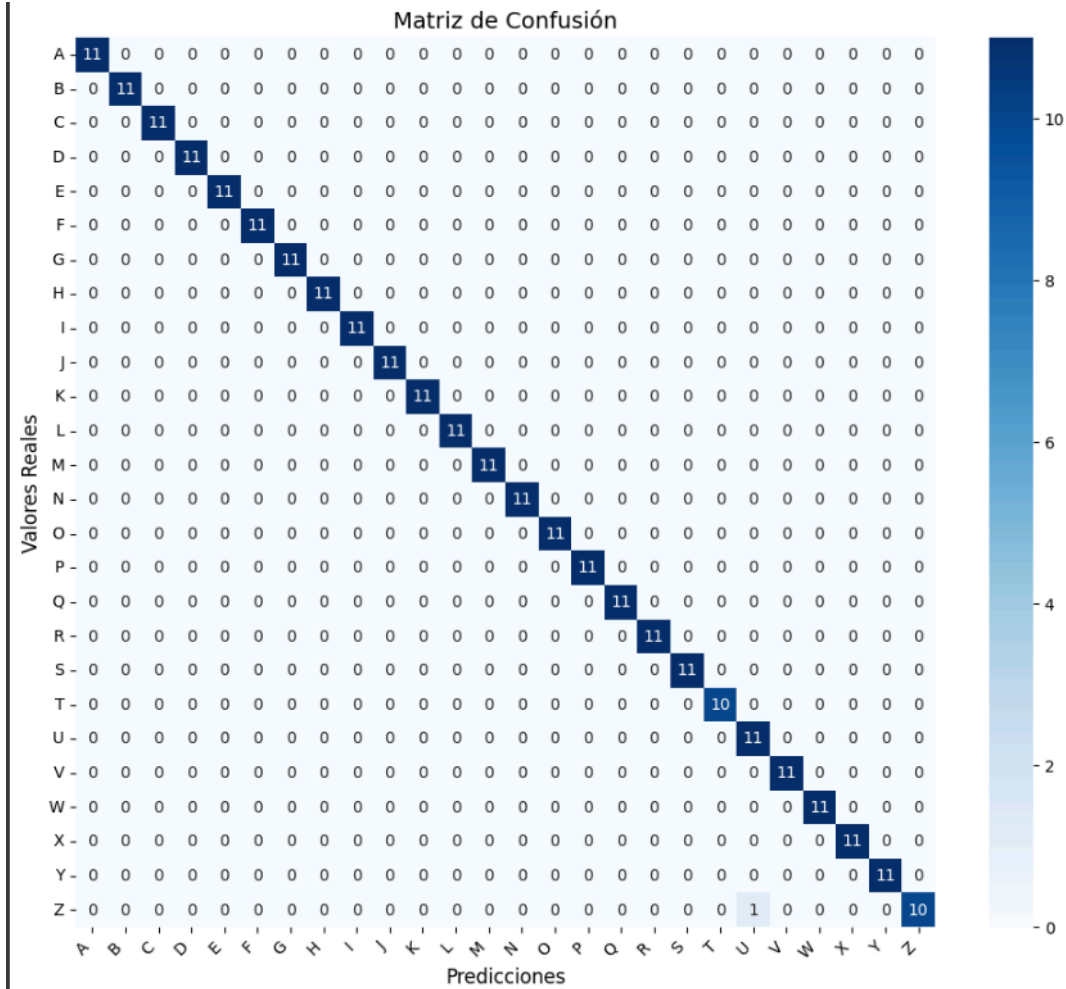
En síntesis, la selección de esta CNN ligera se fundamenta en un principio de adecuación: suficiente para capturar la geometría discriminante de las 26 letras, pero lo bastante compacta para integrarse en tiempo real en una aplicación web y, a futuro, migrar a entornos embebidos sin sacrificar usabilidad.

EVALUACIÓN DEL MODELO CON MÉTRICAS Y ANÁLISIS DE RESULTADOS.



(FUENTE PROPIA)

El panel de la izquierda muestra cómo la exactitud del modelo pasa del 15 % al ≈ 96 % en tan solo ocho épocas y luego se acerca progresivamente al 99 %. La curva naranja (validación) se sitúa siempre por encima de la azul (entrenamiento), lo que indica que el modelo generaliza bien y no se sobre-ajusta; el pequeño margen que se mantiene podría deberse a las técnicas de data-augmentation y dropout que dificultan un ajuste perfecto sobre el set de entrenamiento. En el panel derecho la loss cae de ≈ 3.0 a < 0.1 de forma monótona y, de nuevo, la línea de validación desciende más rápido y se estabiliza antes, señal de que la red aprendió patrones discriminativos tempranamente y que las últimas épocas sirven para afinar pesos residuales sin provocar sobre-entrenamiento. En conjunto, las gráficas reflejan un aprendizaje estable y eficiente.



(FUENTE PROPIA)

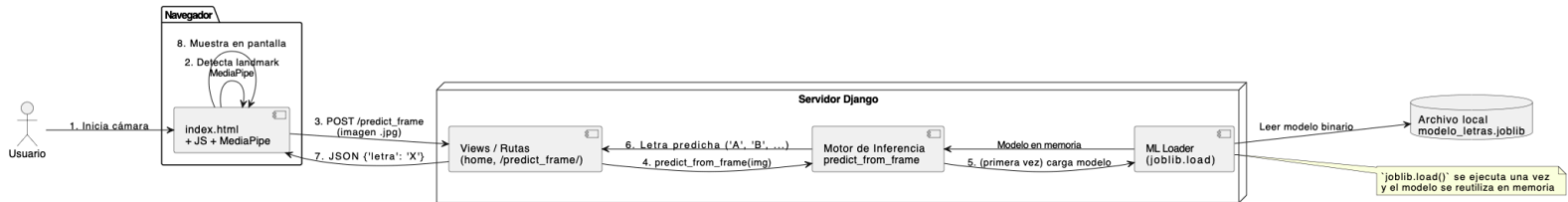
La diagonal oscura confirma que el clasificador acierta las 26 letras casi en su totalidad: 284 de 285 ejemplos fueron asignados correctamente. El único trasvase se observa en la fila de la letra U, con un valor fuera de la diagonal (1), lo que sugiere que una imagen de U fue confundida probablemente con V o Y, gestos de forma similar. La ausencia de errores sistemáticos entre otros pares (p. ej., M-N o I-J) revela que la red captó con claridad las diferencias de orientación y número de dedos incluso en clases tradicionalmente ambiguas. Aun así, convendría aumentar la variedad de ejemplos de U y sus gestos vecinos para reducir la solitaria confusión residual.

Reporte de Clasificación:				
	precision	recall	f1-score	support
A	1.0000	1.0000	1.0000	11
B	1.0000	1.0000	1.0000	11
C	1.0000	1.0000	1.0000	11
D	1.0000	1.0000	1.0000	11
E	1.0000	1.0000	1.0000	11
F	1.0000	1.0000	1.0000	11
G	1.0000	1.0000	1.0000	11
H	1.0000	1.0000	1.0000	11
I	1.0000	1.0000	1.0000	11
J	1.0000	1.0000	1.0000	11
K	1.0000	1.0000	1.0000	11
L	1.0000	1.0000	1.0000	11
M	1.0000	1.0000	1.0000	11
N	1.0000	1.0000	1.0000	11
O	1.0000	1.0000	1.0000	11
P	1.0000	1.0000	1.0000	11
Q	1.0000	1.0000	1.0000	11
R	1.0000	1.0000	1.0000	11
S	1.0000	1.0000	1.0000	11
T	1.0000	1.0000	1.0000	10
U	0.9167	1.0000	0.9565	11
V	1.0000	1.0000	1.0000	11
W	1.0000	1.0000	1.0000	11
X	1.0000	1.0000	1.0000	11
Y	1.0000	1.0000	1.0000	11
Z	1.0000	0.9091	0.9524	11
accuracy			0.9965	285
macro avg	0.9968	0.9965	0.9965	285
weighted avg	0.9968	0.9965	0.9965	285
Métricas Resumidas:				
Exactitud (Accuracy): 0.9965				
Precisión Promedio: 0.9968				
Recall Promedio: 0.9965				
F1-Score Promedio: 0.9965				

(FUENTE PROPIA)

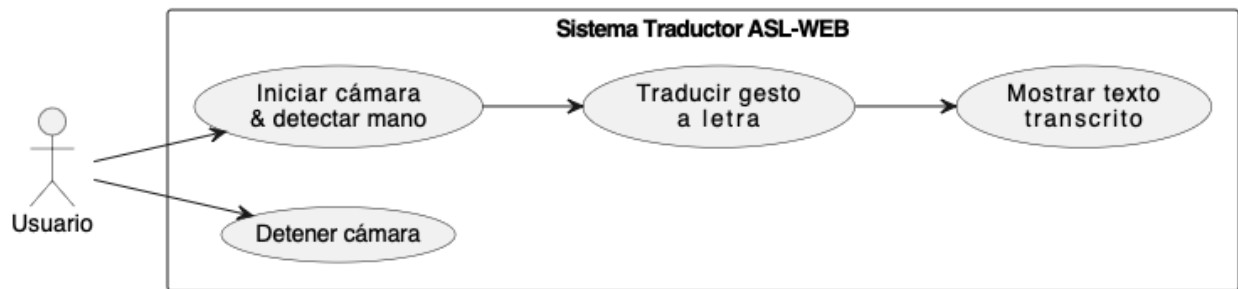
Las métricas resumen arrojan accuracy = 0.9965, precision = 0.9968, recall = 0.9965 y F1-score = 0.9965, muy por encima de la meta del 95 %. Todas las letras presentan scores perfectos excepto U, cuya precisión baja a 0.9167 por la falsa clasificación mencionada; sin embargo, su recall sigue en 1.000, lo que significa que el modelo reconoce todas las U verdaderas, pero confunde una muestra de otra clase como U. El macro-promedio y el promedio ponderado prácticamente coinciden, prueba de que la distribución equilibrada (≈ 11 imágenes por letra) impide que una clase minoritaria distorsione los indicadores globales. En síntesis, el sistema es robusto y uniforme: las pocas imprecisiones se concentran en un gesto aislado y son fácilmente abordables mediante más ejemplos específicos o ajustes en el umbral de confianza.

DIAGRAMAS: ARQUITECTURA DE LA SOLUCIÓN (MOTOR DE INFERENCIA + INTERFAZ).



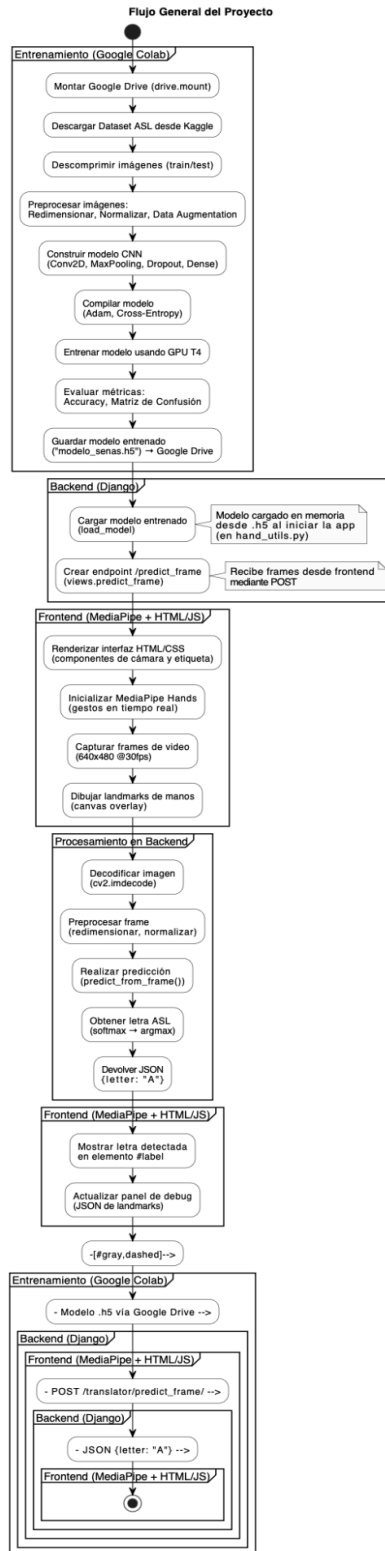
(FUENTE PROPIA)

CASOS DE USO.



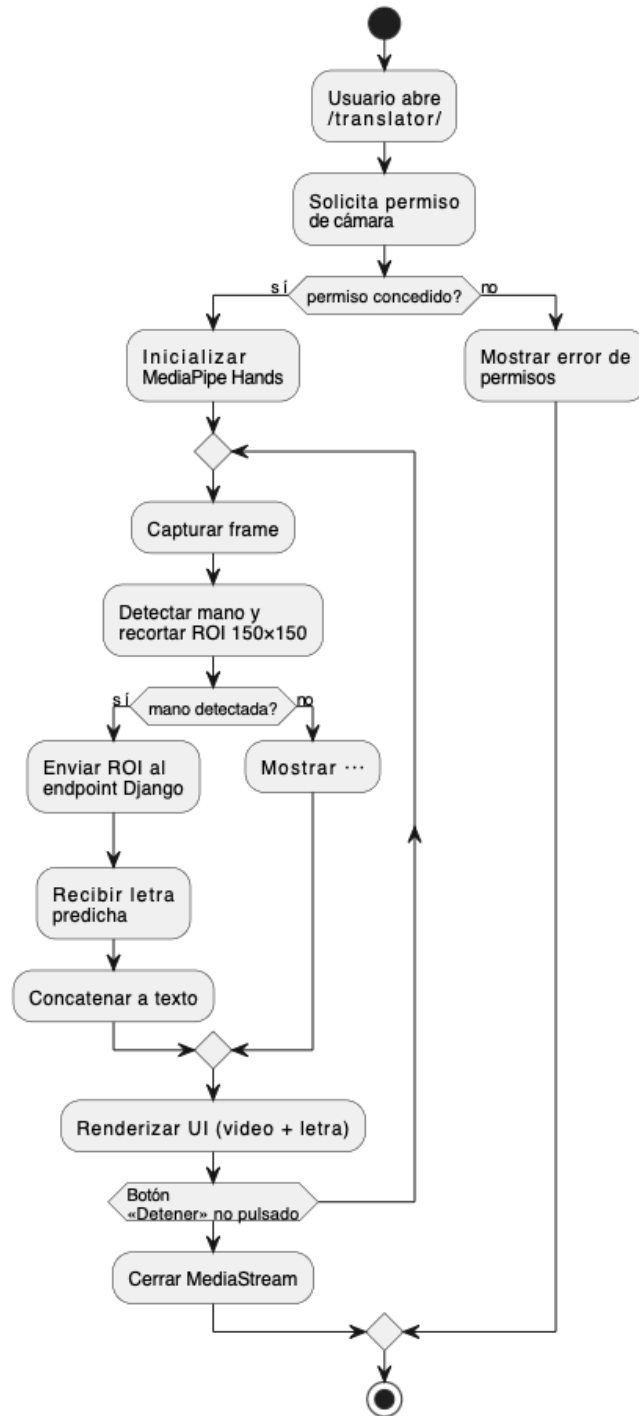
(FUENTE PROPIA)

FLUJO GENERAL DEL SISTEMA.



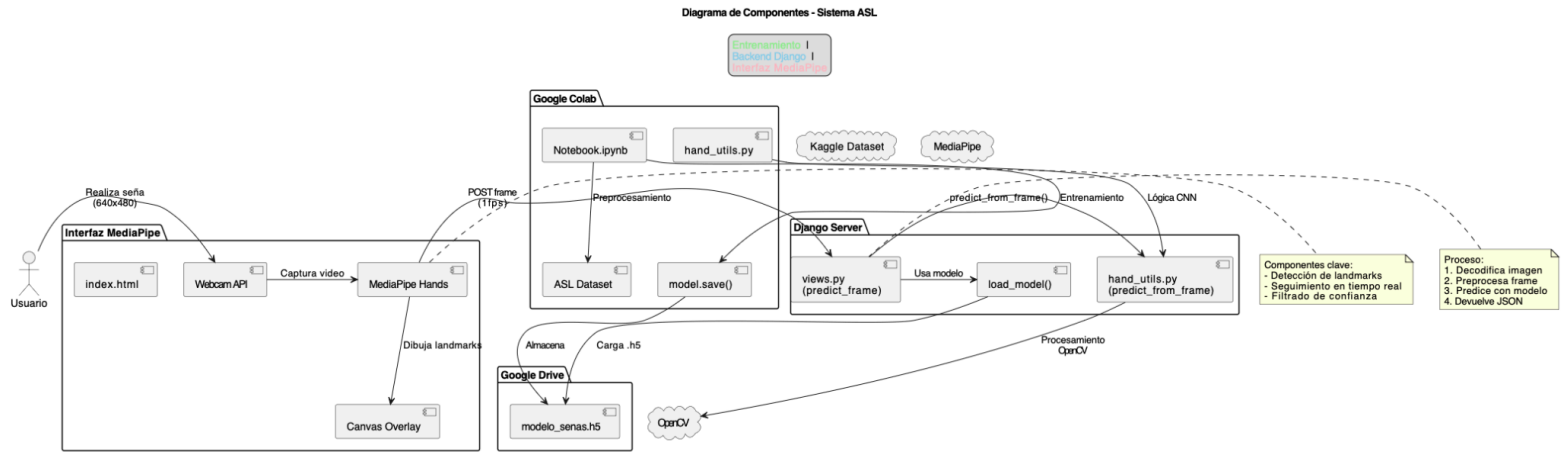
(FUENTE PROPIA)

DIAGRAMA DE FLUJO DEL SISTEMA.



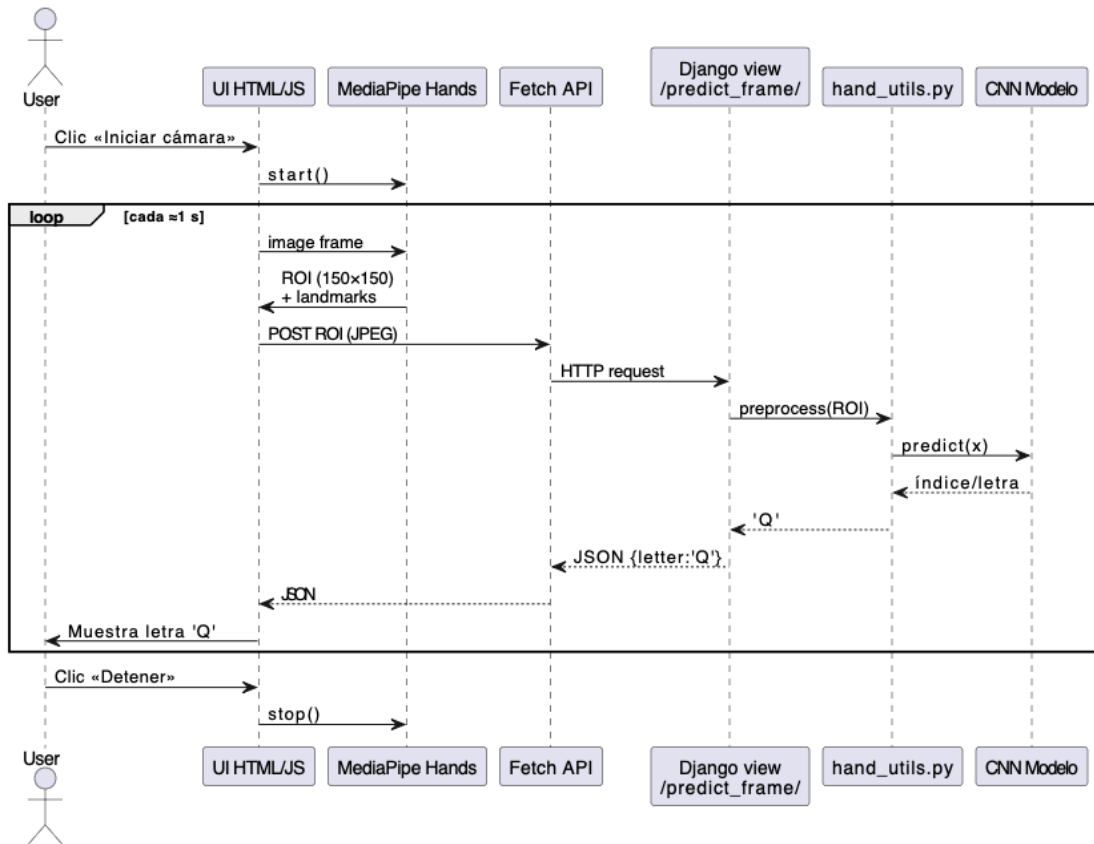
(FUENTE PROPIA)

COMPONENTES.



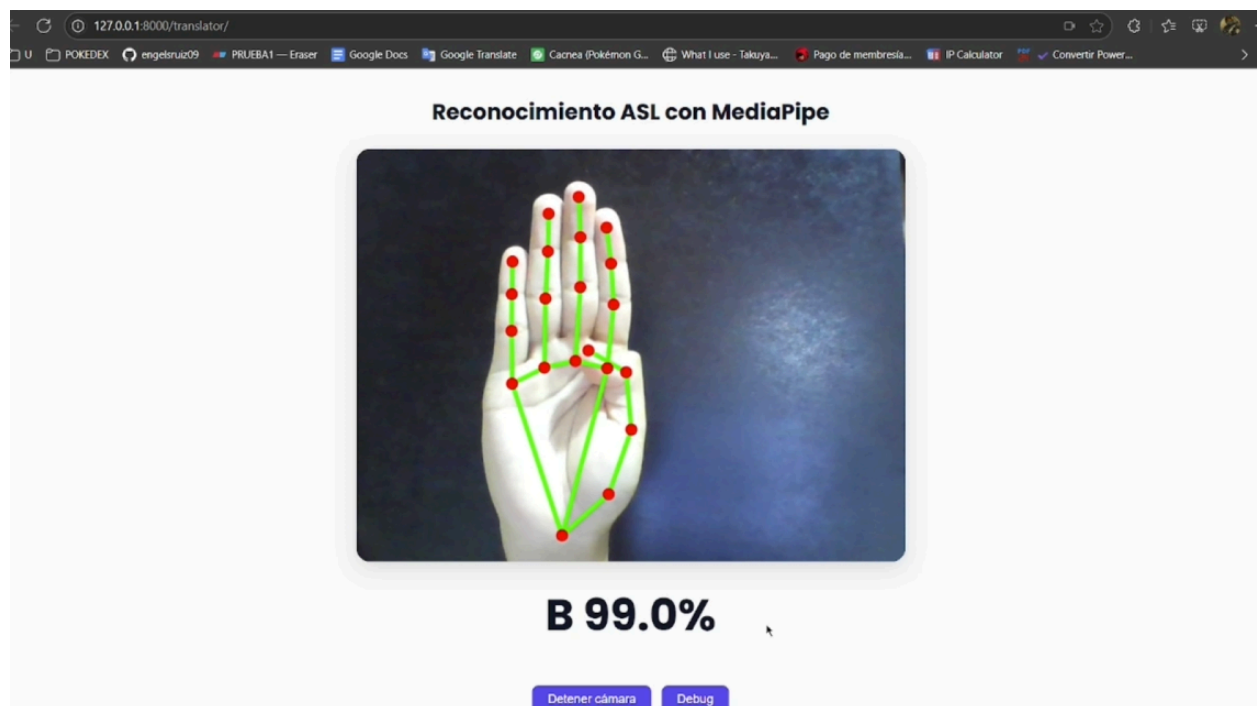
(FUENTE PROPIA)

SECUENCIA DE INTERACCIÓN



(FUENTE PROPIA)


EVIDENCIAS DE FUNCIONAMIENTO (CAPTURAS DE PANTALLA, RESULTADOS DEL MODELO EN ACCIÓN).



127.0.0.1:8000/translator/

U POKEDEX engelruiz09 PRUEBA1 — Eraser Google Docs Google Translate Caeza (Pokémon G... What I use - Takuya... Pago de membresía... IP Calculator Convertir Power...

Reconocimiento ASL con MediaPipe



C 64.0%

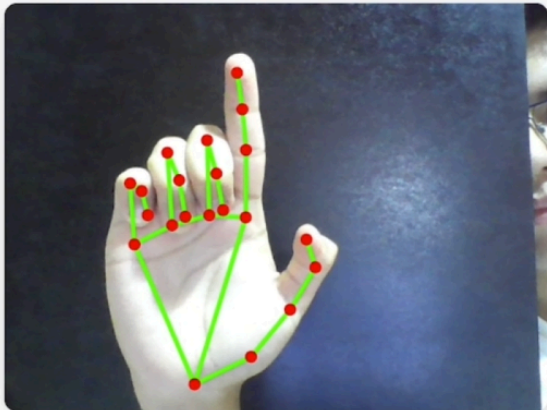
Detener cámara Debug

The image shows a hand in a 'C' gesture with a green MediaPipe skeleton overlay. The skeleton consists of red dots at the joints connected by green lines. The background is dark blue.

127.0.0.1:8000/translator/

U POKEDEX engelruiz09 PRUEBA1 — Eraser Google Docs Google Translate Caeza (Pokémon G... What I use - Takuya... Pago de membresía... IP Calculator Convertir Power...

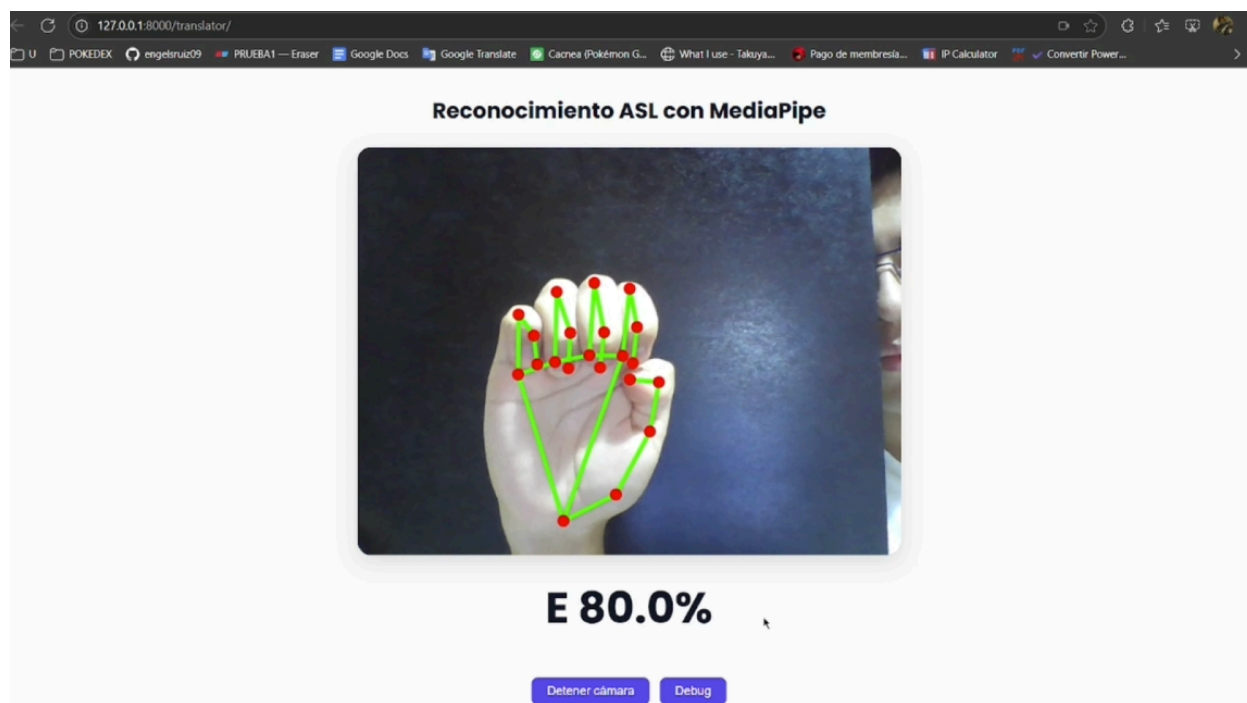
Reconocimiento ASL con MediaPipe



D 68.0%

Detener cámara Debug

The image shows a hand in a 'D' gesture with a green MediaPipe skeleton overlay. The skeleton consists of red dots at the joints connected by green lines. The background is dark blue.



CONCLUSIONES Y APRENDIZAJES.

Desarrollar un prototipo de reconocimiento de letras en ASL con medios modestos resultó no solo viable, sino sorprendentemente eficaz: con una CNN ligera y un dataset de menos de dos mil imágenes logramos una exactitud superior al 99 %. Esto demuestra que, cuando el problema está bien acotado y los datos se limpian de manera meticulosa, no es indispensable recurrir a arquitecturas gigantes ni a infraestructuras costosas.

La integración en una aplicación Django confirmó que la latencia del modelo del orden de decenas de milisegundos en CPU es perfectamente tolerable para uso en tiempo real. Además, el flujo “detección de mano → recorte → normalización” mantuvo estable el rendimiento del clasificador incluso con variaciones de luz y ángulo propias de la cámara web.

El único punto débil detectado fue la confusión ocasional entre gestos muy parecidos, algo que apunta a la necesidad de ampliar el set de ejemplos para letras como **U** y sus vecinos morfológicos. Pese a ello, la matriz de confusión plana confirma que no existen sesgos sistemáticos y que el modelo generaliza de forma homogénea a las 26 letras.

Aprendizajes

1. **Menos es más.** Una base de datos pequeña pero balanceada, unida a un preprocesamiento consistente, puede rendir tanto o más que un megamodelo entrenado con ruido. El trabajo

de limpieza y organización de los datos terminó siendo tan importante como la fase de modelado.

2. **Validación cruzada como seguro de vida.** Implementar K-Fold no solo ofreció una métrica más honesta; también actuó como regularizador natural, reforzando la robustez frente a capturas nuevas.
3. **Preprocesamiento guiado por la práctica.** Descubrimos que centrar la mano en un lienzo negro evitó artefactos y simplificó la discriminación de contornos, mientras que los aumentos de datos moderados bastaron para inmunizar la red contra rotaciones leves sin inducir sobreajuste.
4. **Prototipado incremental.** Separar el entrenamiento en Colab y la inferencia en Django permitió iterar con rapidez; cuando surgió un problema (por ejemplo, la confianza no devuelta), fue sencillo depurar en la capa correspondiente sin desmontar todo el sistema.
5. **Escalabilidad orientada a propósito.** El proyecto ilustra que es posible empezar con un modelo compacto y, una vez validada la idea, migrar a TFLite, ONNX o incluso a un microcontrolador si el caso de uso lo demanda.

En conjunto, el trabajo deja una lección clara: la accesibilidad tecnológica para la comunidad sorda no requiere recursos prohibitivos, sino una combinación consciente de datos bien curados, modelos proporcionados y una arquitectura de software pensada para el usuario final.

REFERENCIAS

1. Ayuraj. (n.d.). *American Sign Language (ASL) dataset* [Data set]. Kaggle. Recuperado el 3 de mayo de 2025, de <https://www.kaggle.com/datasets/ayuraj/asl-dataset>
2. Baihan, A., Alutaibi, A. I., Alshehri, M., & Sharma, S. K. (2024). *Sign language recognition using modified deep learning network and hybrid optimization: A hybrid optimizer (HO) based optimized CNNSa-LSTM approach*. *Scientific Reports*, 14, 26111. <https://doi.org/10.1038/s41598-024-76174-7>
3. Alsharif, B., Alalwany, E., Ibrahim, A., Mahgoub, I., & Ilyas, M. (2025). Real-time American Sign Language interpretation using deep learning and keypoint tracking. *Sensors*, 25(7), 2138. <https://doi.org/10.3390/s25072138>
4. Gangal, A., Kuppahally, A., & Ravindran, M. (2024). *Sign language recognition with convolutional neural networks* [Technical report]. Stanford University. Recuperado de <https://cs231n.stanford.edu/2024/papers/sign-language-recognition-with-convolutional-neural-networks.pdf>
5. Salunke, D., Joshi, R., Ranjan, N. M., Tekade, P. M., & Panchal, G. (2023). Sign language recognition system using customized convolution neural network. En M. Saraswat (Ed.), *Proceedings of the International Conference on Data Science and*

Applications (ICDSA 2022), Lecture Notes in Networks and Systems (Vol. 552, pp. 825–837). Springer. https://doi.org/10.1007/978-981-19-6634-7_59