Prototipo basado en visión artificial para el reconocimiento del alfabeto de la lengua de señas colombiana

Miguel Ángel Villamil Huertas

Asesor

Javier Hernán Jiménez Beltrán

Universidad Nacional Abierta y a Distancia UNAD

Escuela de Ciencias Básicas Tecnología e Ingeniería ECBTI

Ingenierías de Sistemas

2025

Dedicatoria

Este trabajo lo dedico a mi madre, fuente inagotable de amor, paciencia y fortaleza, quien siempre me ha ensenado a no darme por vencido y a luchar por las cosas que quiero. También dedico este proyecto que busca tender puentes de comunicación donde antes existían barreras en la comunidad sorda, que con su perseverancia y valentía enfrenta dificultades que muchos no alcanzan a imaginar, les expreso mi respeto y admiración. Sé que la falta de accesibilidad no es solo un obstáculo técnico, sino también una prueba constante de resiliencia. Mi compromiso es aportar, desde mis conocimientos y mi trabajo y apoyando con mis conocimientos en soluciones que faciliten su interacción con la sociedad y que reconozcan plenamente su derecho a ser comprendidos y escuchados, aun sin palabras habladas.

Agradecimientos

Agradezco a Dios, quien me ha guiado por el buen camino, dándome fuerzas en los momentos más difíciles. Él me ha enseñado a no rendirme, a levantarme cuando caigo y a enfrentar cada circunstancia con valentía.

Mi madre fue y sigue siendo la voz de esa fe, la mujer que, con paciencia, dedicación y amor me transmitió la confianza y me enseño que no hay barreras que no se puedan superar que la constancia, disciplina no se puedan superar. Si a mi madre quien cuando se da cuenta que me quede sordo se personaliza aún más y desde su lucha y acompañamiento me enseña a creer que todo es posible.

Mis padres. Ellos me formaron con principios sólidos, disciplina y un amor profundo que ha sido el motor de mis logros. Este nuevo triunfo que celebro hoy es fruto de su constancia, de esas palabras de aliento que ellos me dijeron tantas veces y que me impulsaron a seguir, incluso cuando parecía más fácil detenerme.

Resumen

Esta propuesta tiene como objetivo desarrollar un prototipo de traductor digital del alfabeto de la Lengua de Señas Colombiana (LSC) para la comunidad sorda, utilizando inteligencia artificial. A partir de la problemática identificada, se definen los requisitos funcionales y el modelo de proceso necesario para su construcción. La implementación de esta solución busca facilitar la comunicación y promover la inclusión, proporcionando una herramienta eficaz para la traducción del alfabeto de LSC. Esta propuesta tiene como objetivo desarrollar un prototipo de traductor digital del alfabeto de la Lengua de Señas Colombiana (LSC) para la comunidad sorda, utilizando inteligencia artificial. A partir de la problemática identificada, se definen los requisitos funcionales y el modelo de proceso necesario para su construcción. La implementación de esta solución busca facilitar la comunicación y promover la inclusión, proporcionando una herramienta eficaz para la traducción del alfabeto de LSC.

Palabras clave**:** Software, inteligencia artificial, lengua de señas colombiana, inclusión, visión artificial, redes neuronales

Abstract

This proposal aims to develop a prototype digital translator of the Colombian Sign Language (CSL) alphabet for the deaf community, using artificial intelligence. Based on the identified problem, the functional requirements and the process model necessary for its construction are defined. The implementation of this solution seeks to facilitate communication and promote inclusion by providing an effective tool for translating the CSL alphabet. This proposal aims to develop a prototype digital translator of the Colombian Sign Language (CSL) alphabet for the deaf community, using artificial intelligence. Based on the identified problem, the functional requirements and the process model necessary for its construction are defined. The implementation of this solution seeks to facilitate communication and promote inclusion by providing an effective tool for translating the CSL alphabet.

Keywords**:** Software, artificial intelligence, Colombian sign language, inclusion, computer vision, neural networks

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Introducción

La comunidad sorda en Colombia enfrenta barreras de comunicación significativas debido a la falta de herramientas accesibles que faciliten la interacción con personas oyentes. Esta propuesta busca desarrollar un prototipo de traductor digital del alfabeto en Lengua de Señas Colombiana (LSC), utilizando inteligencia artificial (IA) para convertir señas estáticas (letras del alfabeto dactilológico) en texto, promoviendo así una mayor inclusión social.

El presente trabajo describe el desarrollo de una aplicación diseñada para fomentar la inclusión y reducir las barreras de comunicación con personas con discapacidad auditiva. Esta plataforma digital utiliza tecnologías de inteligencia artificial y visión por computadora para la traducción de señas.

La Inteligencia Artificial (IA) ha transformado la manera en que los sistemas tecnológicos interactúan con el ser humano, permitiendo desarrollar herramientas capaces de reconocer, interpretar y responder a distintos tipos de información. En el contexto del Lenguaje de Señas Colombiano (LSC), la IA, junto con las técnicas de visión por computadora, ofrece una oportunidad significativa para mejorar la comunicación entre personas sordas y oyentes.

La visión por computadora permite que las máquinas comprendan e interpreten las señales visuales captadas por cámaras, identificando patrones, movimientos y gestos de las manos en tiempo real. Mediante algoritmos de aprendizaje automático (machine learning) y redes neuronales convolucionales (CNN), los sistemas pueden detectar la posición de los dedos, reconocer expresiones faciales y clasificar señas específicas del LSC.

Presentación del proyecto aplicado

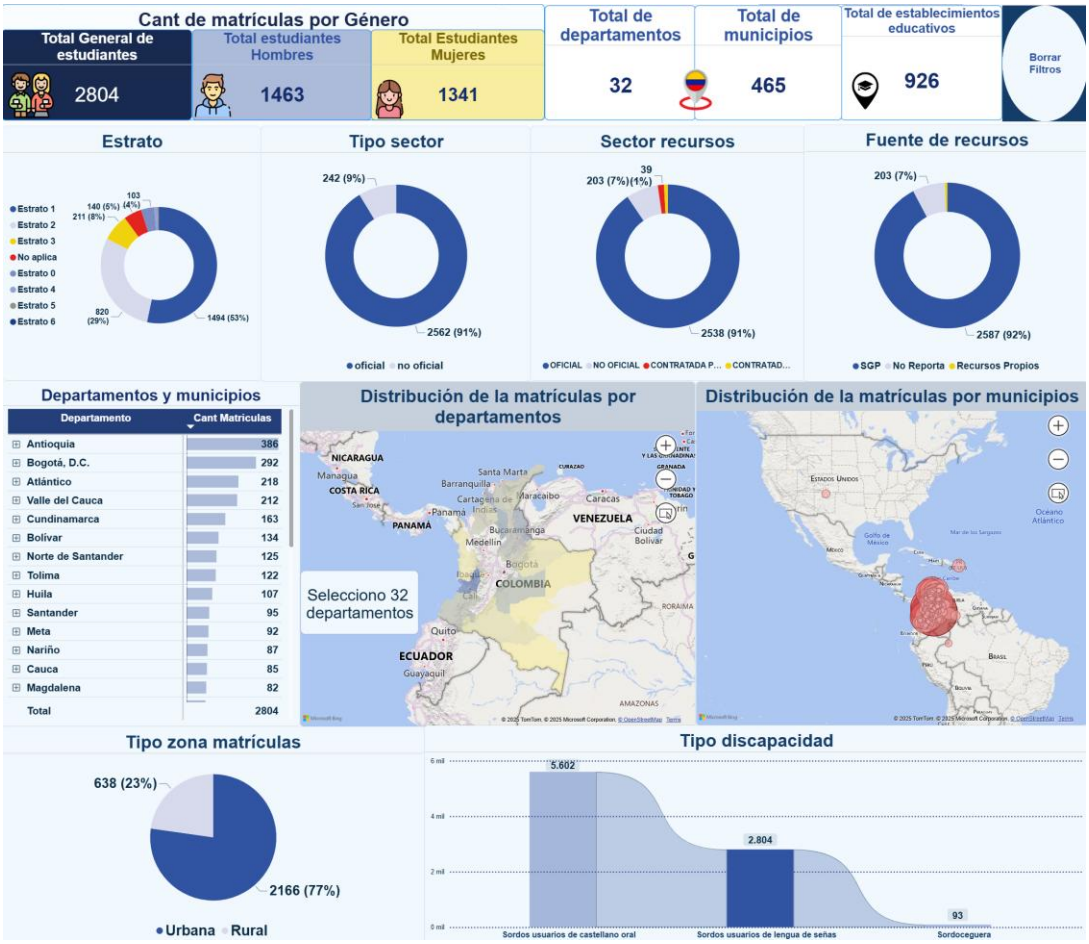
Problema de Investigación

En Colombia, la comunicación entre personas sordas y oyentes que no dominan la Lengua de Señas Colombiana (LSC) continúa siendo una barrera significativa en contextos educativos, laborales y sociales. Esta limitación se debe, en parte, a la falta de herramientas tecnológicas que permitan una traducción rápida y precisa del alfabeto dactilológico de la LSC hacia texto o voz, y viceversa. La ausencia de soluciones accesibles y adaptadas a la realidad lingüística del país perpetúa desigualdades y restringe la participación plena de la comunidad sorda en la sociedad.

En Colombia, según datos entregados por Departamento Nacional de Estadísticas (DANE) en el 2021, en la encuesta nacional de calidad de vida, existen 459.784 personas sordas a nivel nacional. De esos, según cifras del Instituto Nacional para Sordos INSOR, 8989 tienen acceso a educación básica y media y 899 a educación superior (INSOR, 2025); como se ve en la Figura 1, aunque hay esfuerzos para brindar educación a esta población, todavía falta cobertura porque existen brechas en la comunicación entre hablantes y la comunidad sorda.

Figura 1.

Visualización de datos sobre matrícula de estudiantes sordos en Colombia.



Nota: Visión integral de las matrículas de estudiantes sordos, desglosándolas por género, estrato, tipo de sector, fuentes de recursos, ubicación geográfica. *Fuente.* Instituto Nacional para Sordos – INSOR.

En el ámbito laboral, según el Departamento Nacional de Estadísticas (DANE), para el último trimestre del 2024 (octubre a diciembre), la tasa global de participación de población con discapacidad en el mercado laboral fue del 22,4% que evidencia una diferencia respecto a la población con discapacidad, cuya participación fue del 66,2%, esto muestra que aunque existen esfuerzos por disminuir las barreras comunicativas que impiden a la comunidad sorda acceder a oportunidades de estudio y de trabajo, aún existen brechas, asociadas a la comunicación entre

personas hablantes y la comunidad sorda, entre otras causas porque no siempre se cuenta con un intérprete de señas de lengua colombiana o porque falta aún más inclusión de esta población en diferentes ámbitos de la vida cotidiana generando brechas de accesibilidad, comunicación y desigualdad de oportunidades.

Ser sordo en Colombia no solo implica desafíos en la comunicación y el aprendizaje, sino también repercusiones en los ámbitos económicos, laborales y sociales. Según el INSOR, el 70% de las personas sordas registradas en el RLCPD (Registro para la Localización y Caracterización de Personas con Discapacidad) se encuentran en situación de pobreza o vulnerabilidad, y solo el 15% de la población en edad productiva está vinculada laboralmente. Además, el 89% de estas personas desempeñan trabajos de baja cualificación, mientras que solo el 11% accede a empleos que requieren mayor especialización. Estos datos evidencian las deficiencias en la implementación de estrategias y herramientas que garanticen el acceso equitativo a la educación, el trabajo y la salud para la comunidad sorda en Colombia.

Ante esta problemática, el uso de tecnologías emergentes, como la inteligencia artificial permite plantear una solución aprovechando el uso de herramientas tecnológicas. Abordar este problema mediante un prototipo basado en visión artificial representa una oportunidad para combinar la innovación tecnológica con el compromiso social. Un sistema de este tipo no solo contribuiría a la inclusión y a la accesibilidad, sino que también abriría nuevas posibilidades para el aprendizaje de la Lengua de Señas Colombiana (LSC) por parte de personas oyentes, promoviendo así una interacción más equitativa y respetuosa entre comunidades

Pregunta de Investigación

Teniendo en cuenta la problemática planteada, se presenta la siguiente pregunta de investigación:

¿Cómo utilizar inteligencia artificial para reconocer el alfabeto de la Lengua de Señas Colombiana, facilitando su traducción y reduciendo las barreras comunicativas?

Objetivos

Objetivo General

Desarrollar un prototipo basado en visión artificial para el reconocimiento del alfabeto de la Lengua de Señas Colombiana, favoreciendo la accesibilidad comunicativa entre personas sordas y oyentes.

Objetivos Específicos

Realizar un análisis de software para definir los requisitos funcionales y no funcionales del prototipo de reconocimiento del alfabeto de la Lengua de Señas Colombiana.

Crear un conjunto de datos de entrenamiento para la red neuronal, las cuales serán imágenes representativas del alfabeto de la Lengua de Señas Colombiana.

Desarrollar un modelo de inteligencia artificial basado en visión por computador para la identificación y clasificación de las señas del alfabeto de la Lengua de Señas Colombiana

Validar el algoritmo con inteligencia artificial para el reconocimiento de imágenes, mediante pruebas con usuarios.

Justificación

La comunidad sorda enfrenta barreras significativas en la comunicación con la sociedad oyente, lo que limita su participación plena en la sociedad. Aunque existen los intérpretes que ayudan a la traducción, no siempre se cuenta con estas personas de manera inmediata o en todo lugar, lo que dificulta el acceso a la educación, el empleo y otros servicios. Por ello, el uso de tecnologías basadas en inteligencia artificial y procesamiento de imágenes representa una oportunidad para mejorar la accesibilidad y autonomía de las personas sordas en su interacción

con el entorno.

Un traductor digital puede ayudar a superar estas barreras y promover la inclusión y la igualdad de oportunidades. Por eso, mediante esta propuesta de traductor digital de lengua de señas de Colombia, apoyado en inteligencia artificial, se presenta una herramienta que busca facilitar la comunicación entre la comunidad sorda y la sociedad oyente. Este sistema utiliza tecnologías de reconocimiento de gestos, inteligencia artificial y aprendizaje automático para traducir la lengua de señas a lenguas orales y viceversa. El impacto potencial del traductor digital es significativo. Puede mejorar la calidad de vida de la comunidad sorda, incrementar su participación en la sociedad y fomentar la inclusión social. Además, puede ayudar a reducir las barreras y promover la igualdad de oportunidades en áreas como la entidad, el empleo y la vida pública

Marco Conceptual y Teórico

Discapacidad

La discapacidad es un concepto amplio que incluye tres dimensiones: las deficiencias, que afectan funciones o estructuras corporales; las limitaciones de la actividad, que dificultan la realización de tareas o acciones; y las restricciones de la participación, que impiden la inclusión en situaciones de la vida cotidiana.

El Congreso de la República en la ley 762 de 2002 decreta: El término “discapacidad” significa una deficiencia física mental o sensorial, ya sea de naturaleza permanente o temporal, que limita la capacidad de ejercer una o más actividades esenciales de la vida diaria, que puede ser causada o agravada por el entorno económico y social. (Congreso de la República de Colombia, 2002).

Discapacidad Auditiva

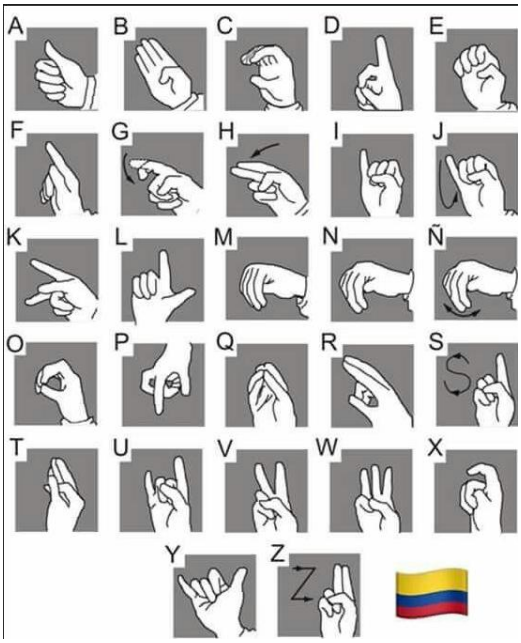
La discapacidad auditiva se refiere a la pérdida total o parcial de la capacidad de percibir sonidos. Esta condición impacta negativamente la calidad de vida de quienes la padecen, ya que dificulta la comunicación, la identificación de sonidos y la interacción social, afectando especialmente los ámbitos emocional, social y educativo.

Lenguaje de Señas

Las lenguas de señas son lenguas viso-gestuales-espaciales, basadas en el uso de las manos, los ojos, el rostro, la boca y el cuerpo. Son lenguas para ser vistas, no para ser oídas y cumplen las mismas funciones que la lengua oral para los oyentes (Marzo Peña, Rodríguez Fleitas, y Fresquet Pedroso, 2022). En la imagen se muestra lengua de señas:

Figura 2.

Alfabeto dactilológico



Nota: Esta imagen presenta el alfabeto dactilológico (abecedario manual) de la lengua de señas colombiana (LSC). *Fuente*. https://co.pinterest.com/pin/4574037113878604/.

Según la RAE, lenguaje es: "Facultad del ser humano de expresarse y comunicarse con los demás a través del sonido articulado o de otros sistemas de signos." (RAE, 2019)11, seña es: "indicio o gesto para dar a entender algo o venir en conocimiento de ello" (RAE, 2019)12, por lo tanto, según el Diccionario panhispánico del español jurídico, la lengua de señas es: "Sistema lingüístico de carácter visual, espacial, gestual y manual en cuya conformación intervienen factores históricos, culturales, lingüísticos y sociales, utilizado tradicionalmente como lengua por las personas sordas, con discapacidad auditiva y sordociegas signantes en España"(Diccionario panhispánico del español jurídico, 2019).

La lengua de señas es una forma de expresión del lenguaje humano que se organiza en un sistema visoespacial, a diferencia de la lengua oral, cuya materialidad es acústica. Su uso otorga

a la comunidad sorda una identidad lingüística y cultural propia, fortaleciendo su sentido de pertenencia y comunicación (Roso, 2015).

La lengua de señas se estructura en distintos niveles lingüísticos: sintáctico, semántico, morfológico y fonológico. En el nivel fonológico, las señas pueden descomponerse en parámetros específicos que definen su forma y significado dentro del sistema de comunicación visual-gestual.

- La configuración de la mano, la forma que adopta la mano (puño, palma arriba).
- El movimiento.
- La orientación de la mano (palma arriba, palma abajo).
- La ubicación, el lugar en el espacio donde se realiza la seña.
- La expresión facial (Roso, 2015).

La lengua de señas no es universal, ya que varía según el país y la comunidad lingüística en la que se desarrolla. Cada región tiene su propio sistema de señas con estructuras y significados únicos.

Inteligencia Artificial

La Inteligencia Artificial se define como una disciplina de la informática que busca crear sistemas que imiten la capacidad humana para percibir problemas, identificar sus componentes y, en consecuencia, resolverlos y tomar decisiones (Aguilar, Gavilanes, Freire, Quincha, 2023).

La Inteligencia Artificial (IA) engloba un conjunto de técnicas y modelos computacionales diseñados para simular capacidades cognitivas humanas. Entre sus subcampos más relevantes están el aprendizaje automático (machine learning) y el aprendizaje profundo (deep learning). En visión por computadora, estos modelos (por ejemplo, redes neuronales

convolucionales y arquitecturas basadas en transformadores o modelos de poses) aprenden automáticamente representaciones visuales a partir de datos imágenes y secuencias de video.

Machine Learning

El aprendizaje automático o Machine learning se puede definir como es una rama de la inteligencia artificial que permite a las computadoras aprender de los datos y mejorar de forma autónoma sin ser programadas explícitamente para cada tarea. En Machine learning existen diferentes tipos de aprendizaje, destacando el supervisado y el no supervisado.

Aprendizaje Supervisado

El entrenamiento supervisado en *Machine Learning* se basa en asociar los datos de entrada con una etiqueta o valor de salida conocido. El aprendizaje supervisado consiste en una serie de algoritmos que utilizan datos de entrada junto con sus salidas asociadas para entrenar un modelo. Esta metodología divide la información en datos de entrenamiento y etiquetas, permitiendo que la máquina aprenda a clasificar o predecir respuestas a medida que se expone a más ejemplos durante el proceso de entrenamiento.

Aprendizaje no Supervisado

El aprendizaje no supervisado se enfoca en extraer información útil de los datos sin necesidad de etiquetas o resultados predefinidos. Su objetivo principal es que la máquina identifique patrones, estructuras o relaciones dentro de la data de forma autónoma, sin intervención humana directa durante el proceso de análisis.

En el aprendizaje no supervisado, los datos de entrada no tienen etiquetas ni categorías predefinidas. La máquina analiza la información de forma autónoma, identificando patrones o similitudes sin intervención humana. Un ejemplo de esto es la clasificación de verduras mediante

imágenes, donde el modelo agrupa los elementos según características como color o tamaño. (Bobadilla, 2020, p. 17).

Deep Learning

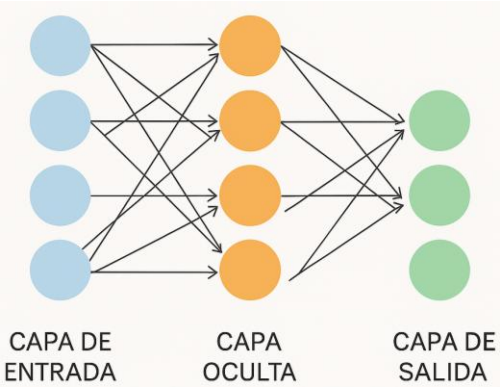
El aprendizaje profundo o Deep learning es una técnica de inteligencia artificial que utiliza redes neuronales artificiales con múltiples capas para procesar datos, identificar patrones complejos y realizar tareas de inteligencia artificial.

Redes neuronales Artificiales (RNA)

Es un modelo computacional inspirado en el cerebro humano que procesa la información a través de nodos denominados neuronas. Estos nodos o neuronas se organizan en capas: una capa de entrada que recibe los datos del mundo exterior, una o más capas ocultas donde se realiza la mayor parte del procesamiento y una capa de salida que genera el resultado final. En la figura 3 se puede ver la estructura de una red neuronal artificial.

Figura 3

Estructura de una Red Neuronal Artificial.



Nota: Arquitectura fundamental de una Red Neuronal Artificial (RNA) simple, que es la base de los sistemas de aprendizaje automático (Machine Learning). *Fuente.*

https://interactivechaos.com/

Red neuronal Convolucional (CNN)

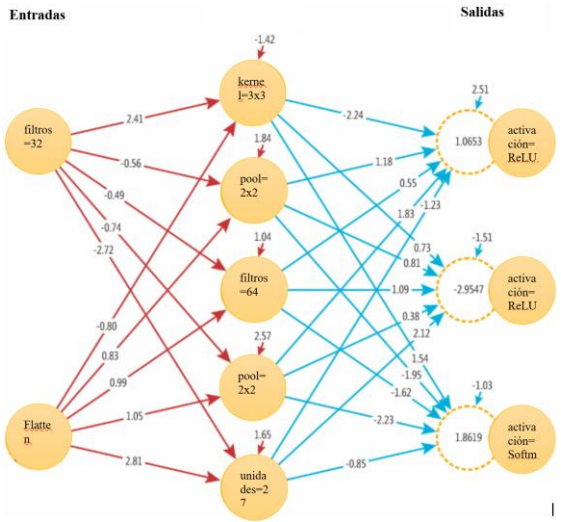
Las redes neuronales convolucionales (CNN) son un tipo de redes neuronales diseñadas específicamente para analizar imágenes y videos. Las CNN procesan imágenes de forma jerárquica a través de tres tipos de capas:

- La capa convolucional que es aquella que realiza la operación de convolución para extraer características de la imagen.
- La capa de agrupamiento que reduce la dimensionalidad de los mapas de características
- La capa totalmente conectada realiza la clasificación final o la predicción basándose en las características que fueron extraídas por las capas anteriores.

En la figura 4 se muestra un ejemplo de arquitectura de una CNN.

Figura 4.

Ejemplo de una arquitectura CNN



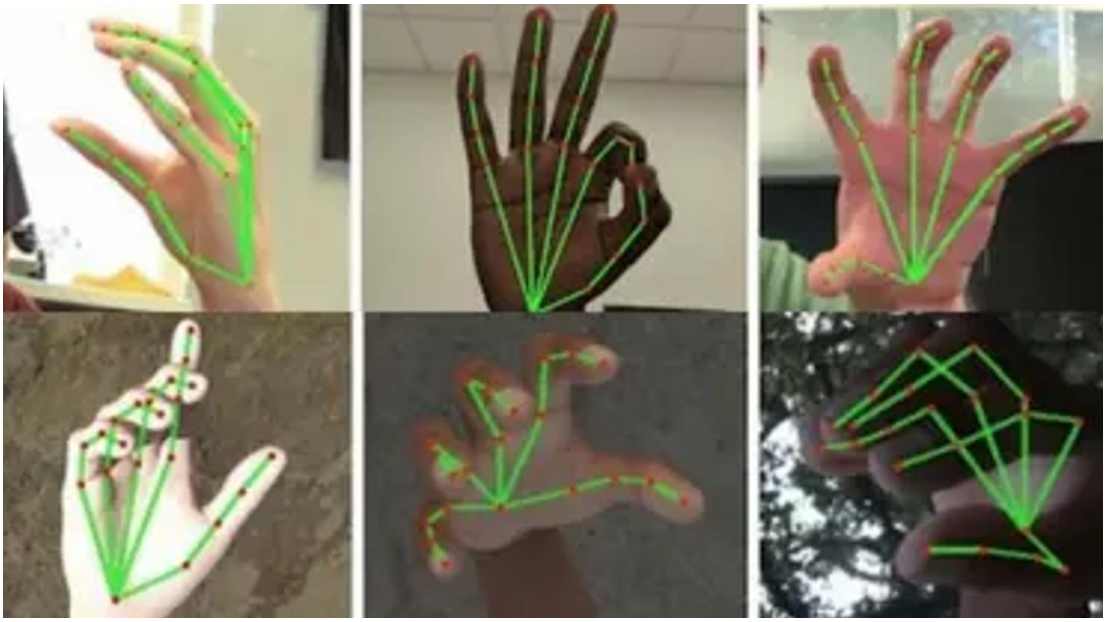
Nota: La imagen muestra una sección de una Red Neuronal Convolucional (CNN), específicamente su porción de clasificación o red densa. *Fuente*. https://antonio-richaud.com/

Visión Computacional

La visión computacional es un área fundamental de la inteligencia artificial dedicada a emular la capacidad de percepción visual humana (Padrón, Planchat y Reina, 2021). Dentro de la visión computacional, la estimación de poses es una tarea de visión artificial que captura y representa de forma gráfica la postura y posición del cuerpo humano. Esta técnica se utiliza a menudo para predecir las posiciones de las partes y articulaciones del cuerpo humano (Mira Abad, 2023), en la figura 5 se presenta la estimación de poses para manos:

Figura 5.

Detección de gestos manuales mediante visión computacional.



Nota: La imagen ilustra un ejemplo de Visión por Computadora y Detección de Hitos (Landmark Detection) aplicada a la mano humana. *Fuente*. https://www.xataka.com/

Mediapipe Studio

MediaPipe Studio es una plataforma web interactiva creada por Google que permite probar, visualizar y ajustar modelos de visión por computador y procesamiento de señales desarrollados con MediaPipe.

Entre sus funcionalidades, permite la detección y seguimiento de múltiples puntos de referencia corporales, incluidas manos, rostro y pose corporal. Para su integración, se definen dos variables clave: una para inicializar el modelo holístico y otra para utilizar las funciones de dibujo. Esta última facilita la visualización en pantalla de los puntos de referencia detectados, así como las expresiones faciales y su probabilidad en tiempo real, lo cual resulta esencial para el análisis dinámico de señas. Para estimar poses de manos se hace uso del modelo HandPose, que tiene 21 puntos clave o landamarks. En la figura 6 se muestra una imagen de MediaPipe Studio

Figura 6.

Detección de puntos de referencia de la mano del mano izquierdo.

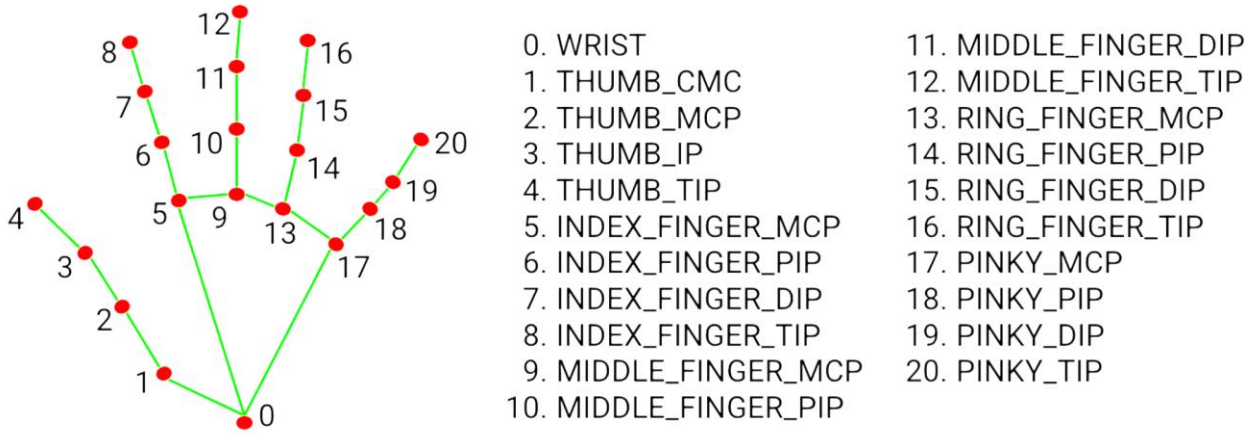
Nota: La imagen muestra la interfaz del Estudio MediaPipe, una plataforma de Google para el desarrollo de soluciones de Machine Learning para visión por computadora. *Fuente*. Autor.

Mediapipe Hands

Este modelo implementa una técnica basada en aprendizaje profundo que permite captar con precisión la forma y el movimiento de las manos. Esta técnica extrae 21 puntos de referencia en 3D a partir de cada fotograma en tiempo real, lo que facilita una interpretación detallada y dinámica de las señas realizadas por el usuario. En la Figura 7 se pueden ver los puntos clave que se identifican mediante MediaPipe Hands.

Figura 7.

Modelo de referencia de puntos clave en la mano – MediaPipe Hands.



Nota: La imagen es un diagrama técnico que representa el "esqueleto" digital de una mano derecha abierta. Este tipo de esquema se utiliza comúnmente en informática para el reconocimiento de gestos y el seguimiento de manos. *Fuente*. https://link.springer.com/article/10.1007/s00138-023-01413-2

Python

Lenguaje de programación versátil, multiplataforma y multiparadigma, reconocido por su sintaxis clara y legible. Su naturaleza de código abierto facilita la integración en sistemas de alto tráfico como Google o YouTube. Este lenguaje permite automatizar procesos con eficiencia,

reduciendo tiempos y complejidad mediante pocas líneas de código, lo cual lo hace ideal para implementaciones en múltiples plataformas y sistemas operativos.

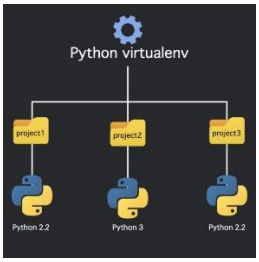
Python como lenguaje de programación debido a su sintaxis simple y cercana al lenguaje natural, lo que facilita un desarrollo rápido, intuitivo y eficiente. Python destaca por su tipado dinámico, gestión automática de memoria y una amplia variedad de librerías especializadas en inteligencia artificial y visión por computador. Aunque no es adecuado para tareas de bajo nivel o que requieran rendimiento extremo, su uso es ampliamente respaldado por organizaciones como Google, NASA y Yahoo, lo que demuestra su confiabilidad y versatilidad en proyectos tecnológicos de alto impacto (González, 2011, p.32).

Venv

Es un módulo incorporado de Python que permite crear entornos virtuales aislados. Cada entorno virtual contiene su propia instalación independiente de Python, junto con sus propios paquetes y dependencias. Esto significa que se puede trabajar en diferentes proyectos sin que sus bibliotecas entren en conflicto entre sí. En la Figura 8 se muestra un ejemplo de creación de un entorno virtual.

Figura 8.

Creación de un entorno virtual en Python con venv.



Nota: Python virtualenv" permite gestionar múltiples proyectos simultáneamente (project1, project2, project3) donde cada uno puede utilizar una versión diferente del lenguaje (versión 2.2 o versión 3) sin que entren en conflicto entre ellas. *Fuente*. https://media.licdn.com/.

OpenCv

OpenCV (Open Source Computer Vision Library), es una biblioteca de código abierto especializada en visión por computador y aprendizaje automático. Esta herramienta proporciona una infraestructura común que facilita el desarrollo de aplicaciones de procesamiento de imágenes, permitiendo integrar capacidades de percepción visual en sistemas inteligentes y acelerando su implementación en productos comerciales.

Integra más de 2500 algoritmos optimizados. Esta biblioteca ofrece un amplio conjunto de herramientas clásicas y avanzadas en visión por computador y aprendizaje automático, lo cual permite realizar tareas como el procesamiento de imágenes, detección de objetos y reconocimiento de patrones de manera eficiente y precisa.

Apoya en algoritmos avanzados de visión por computador que permiten realizar múltiples tareas como la detección y reconocimiento facial, identificación de objetos, clasificación de acciones humanas en video y seguimiento de objetos en movimiento. Estos algoritmos también permiten generar modelos 3D, unir imágenes en panorámicas de alta resolución, extraer información visual para realidad aumentada, e incluso realizar tareas específicas como eliminar ojos rojos o seguir el movimiento ocular. Su versatilidad los hace ideales para aplicaciones complejas y precisas como el reconocimiento de señas.

Tensorflow

Librería especializada que permite la creación, entrenamiento e implementación de modelos de aprendizaje automático. Esta herramienta facilita la integración con otras plataformas y es utilizada en el proyecto para desarrollar el modelo de reconocimiento de gestos, permitiendo así la activación de dispositivos eléctricos a partir de señales visuales.

Una biblioteca de código abierto creada por Google, orientada al aprendizaje automático. Esta herramienta permite construir y entrenar redes neuronales capaces de identificar patrones y relaciones complejas, simulando procesos de aprendizaje humano. TensorFlow es ampliamente utilizado tanto en investigación como en entornos de producción, y ha reemplazado a tecnologías anteriores como DistBelief. Su publicación bajo licencia Apache 2.0 ha impulsado su adopción global y la colaboración de la comunidad de desarrolladores (Gómez, citado en Vera, 2019, p.26).

Keras

Es un framework de alto nivel escrito en Python que permite construir y entrenar modelos de aprendizaje automático de manera sencilla y rápida. Keras funciona sobre plataformas como TensorFlow, CNTK o Theano, y fue diseñado para facilitar la experimentación.

Scikit-learn

La biblioteca scikit-learn permite implementar modelos de aprendizaje automático. En el caso del reconocimiento del alfabeto de la Lengua de Señas Colombiana, scikit-learn facilita la creación, entrenamiento y validación de clasificadores como RandomForestClassifier o SVC, los cuales son adecuados para tareas de clasificación de imágenes procesadas mediante visión artificial.

Su integración en el flujo de trabajo permite aplicar algoritmos robustos sin necesidad de desarrollar modelos desde cero, lo cual acelera el proceso de desarrollo y mejora la precisión del sistema. Además, su compatibilidad con otras bibliotecas como NumPy, OpenCV y Pandas permite un manejo eficiente de los datos capturados por cámara, desde el preprocesamiento hasta la predicción.

Usabilidad en software

La definición más utilizada o reconocida de usabilidad es la que se expone en la norma ISO 9241-11, la cual indica que la usabilidad se describe como el grado con el que un producto puede ser usado por usuarios específicos para alcanzar objetivos específicos con efectividad, eficiencia y satisfacción, en un contexto de uso específico (Pico, López, Lescano, 2023)

Estado arte

A continuación, se presenta estudios relevantes de la literatura científica asociados a soluciones de implementación de lengua de señas apoyada por herramientas tecnológicas:

La trayectoria en el desarrollo de aplicaciones para la lengua de señas ha sido evolutiva, partiendo de soluciones físicas y avanzando hacia la inteligencia artificial. En 2016, se exploró el uso de guantes que, aunque requerían una carga de 15 voltios y una duración de 30 minutos, se enfrentaron a desafíos de precisión relacionados con el color de la piel y los rasgos, dificultando el reconocimiento de señas. Para 2017, se propuso la aplicación de una luz blanca para mejorar el reflejo de la luz, lo que incrementó la precisión de las señas en un 54%, y el uso de un guante quirúrgico logró una mejora cercana al 90%. Estos esfuerzos iniciales buscaban superar barreras ópticas para el reconocimiento (Inca Balseca & Andrade Guaraca, 2022).

Más recientemente, la inteligencia artificial (IA) ha impulsado avances significativos. En el año 2020, se empleó la IA para la traducción de textos a lengua de señas utilizando la librería Google Vision IA, la cual permitió un análisis detallado de las imágenes a través de algoritmos desde el computador (Inca Balseca & Andrade Guaraca, 2022).

También se destaca la importancia del aprendizaje automático y las redes neuronales, logrando sistemas más precisos y eficientes mediante el reconocimiento de imágenes y el

procesamiento en tiempo real (Rasco Miranda, 2025). Los modelos de visión por computadora y aprendizaje profundo, como las Redes Neuronales Convolucionales (CNN) y arquitecturas como YOLOV8 (Rasco Miranda, 2025) y YOLOv10 (Alshahrani et al., 2025), han demostrado gran efectividad. Por ejemplo, en un estudio se emplearon 2,935 imágenes, obteniendo un promedio de precisión (P) de 0.983 y un *recall* (R) de 0.987 con modelos de aprendizaje automático, lo que indica una alta corrección en las detecciones y reconocimiento de objetos (Rasco Miranda, 2025).

Una revisión sistemática también destacó la implementación de modelos de aprendizaje profundo, evaluando 14 artículos y resaltando que ResNet-50 alcanzó una precisión del 99.98%, superando a ResNet-34 CNN (78.5%), aunque se reconoce que la interpretación de gestos y expresiones complejas sigue siendo un desafío (Nadita et al., 2024). ResNet-50, en particular, es una alternativa adoptada por su capacidad de incorporar bloques residuales que mitigan el desvanecimiento del gradiente en redes muy profundas, lo que se ha comprobado en investigaciones enfocadas en lenguas de señas rusa y bengalí (Ashrafi et al., 2024).

Otros modelos arquitectónicos también han mostrado un alto rendimiento. El Sistema de Reconocimiento de Lenguaje de Señas Americano (ASLRS) combina CNN y modelos generativos para identificar formas y movimientos de las manos, logrando una precisión promedio del 92.4% en la clasificación de gestos estáticos de ASL (Karche et al., 2025). Además, el reconocimiento de gestos alfabéticos en múltiples lenguajes (árabe, indonesio, indio y ASL) con YOLOv10 y VGG-19 alcanzó una precisión de 88.07% para YOLOv10 y una precisión de 92.29% para VGG-19 (Alshahrani et al., 2025).

La integración de la IA no solo abarca el reconocimiento estático. La investigación también propone un Sistema de Reconocimiento Automatizado de Señas (SLR) que combina la

extracción de características espaciales con el análisis temporal mediante Long Short-Term Memory networks (LSTM) y mecanismos de atención para lograr una interpretación más precisa de los gestos (Vasumathi et al., 2024). Este modelo, entrenado con un dataset adaptado a señas, alcanzó una precisión del 98.33%, superando a modelos previos (Vasumathi et al., 2024). El papel del Sign Language Translation (SLT) es transformar la secuencia de signos en una secuencia coherente de palabras; para ello, los algoritmos basados en LSTM son los más extendidos, como el modelo que para el lenguaje de señas indio mostró una precisión del 73% en la traducción de oraciones completas y 90% en el reconocimiento de palabras aisladas (Taborri et al., 2023).

Otra investigación denominada SignaSpectrum utiliza redes neuronales profundas y *deep learning* para la detección e interpretación dinámica del lenguaje de señas en tiempo real, traduciendo gestos cotidianos a voz o texto con el modelo SSD MobileNet v2 FPNLite 320x320 (Saini & Kumari, 2024).

Estos avances, en conjunto, establecen las bases para el desarrollo de prototipos para la Lengua de Señas Colombiana (LSC), pudiendo aprovechar la solidez de las CNN para alfabetos estáticos y la capacidad de las LSTM para el reconocimiento de señas dinámicas, incluso integrando servicios de emergencia automatizados (Sari & Utomo, 2023).

Diseño Metodológico

Metodología CRIPS -DM

Como metodología, se usa CRISP-DM (Cross Industry Standard Process for Data Mining) es un enfoque estructurado para el desarrollo de proyectos de minería de datos y aprendizaje automático. Para el prototipo de reconocimiento del alfabeto de la Lengua de Señas Colombiana basado en visión artificial, la metodología se aplica en las siguientes etapas:

Comprensión del Negocio

- Identificar las necesidades y problemáticas de la comunidad sorda en la comunicación con personas oyentes.
- Definir los objetivos del prototipo y los beneficios esperados.
- Establecer los requisitos funcionales y no funcionales del sistema.
- Analizar soluciones previas y enfoques existentes en el reconocimiento de señas.

Comprensión de los Datos

- Recolectar un conjunto de datos de imágenes de las señas del alfabeto de la Lengua de Señas Colombiana (LSC).
- Analizar la calidad de los datos: resolución, iluminación, variabilidad en las posiciones de las manos, etc.
- Identificar posibles sesgos en los datos.
- Explorar las características de las imágenes y definir qué atributos serán útiles para el reconocimiento.

Preparación de los Datos

- Preprocesar las imágenes (escalado, normalización, eliminación de ruido).

- Realizar aumento de datos (data augmentation) para mejorar la capacidad de generalización del modelo.
- Etiquetar correctamente las imágenes con la letra correspondiente del alfabeto de señas.
- Dividir los datos en conjuntos de entrenamiento, validación y prueba.

Modelado

- Seleccionar algoritmos de visión artificial y aprendizaje profundo (por ejemplo, redes neuronales convolucionales - CNN).
- Entrenar diferentes modelos y ajustar hiperparámetros para optimizar el rendimiento.
- Implementar técnicas de regularización y evitar sobreajuste
- Evaluar el desempeño del modelo con métricas como precisión, recall y F1-score.

Evaluación

- Medir la efectividad del modelo en el reconocimiento del alfabeto de la LSC.
- Comparar los resultados obtenidos con los objetivos definidos en la primera fase.
- Realizar pruebas con usuarios para evaluar la usabilidad y precisión del sistema en condiciones reales.
- Identificar posibles mejoras y ajustes antes de la implementación final.

Despliegue

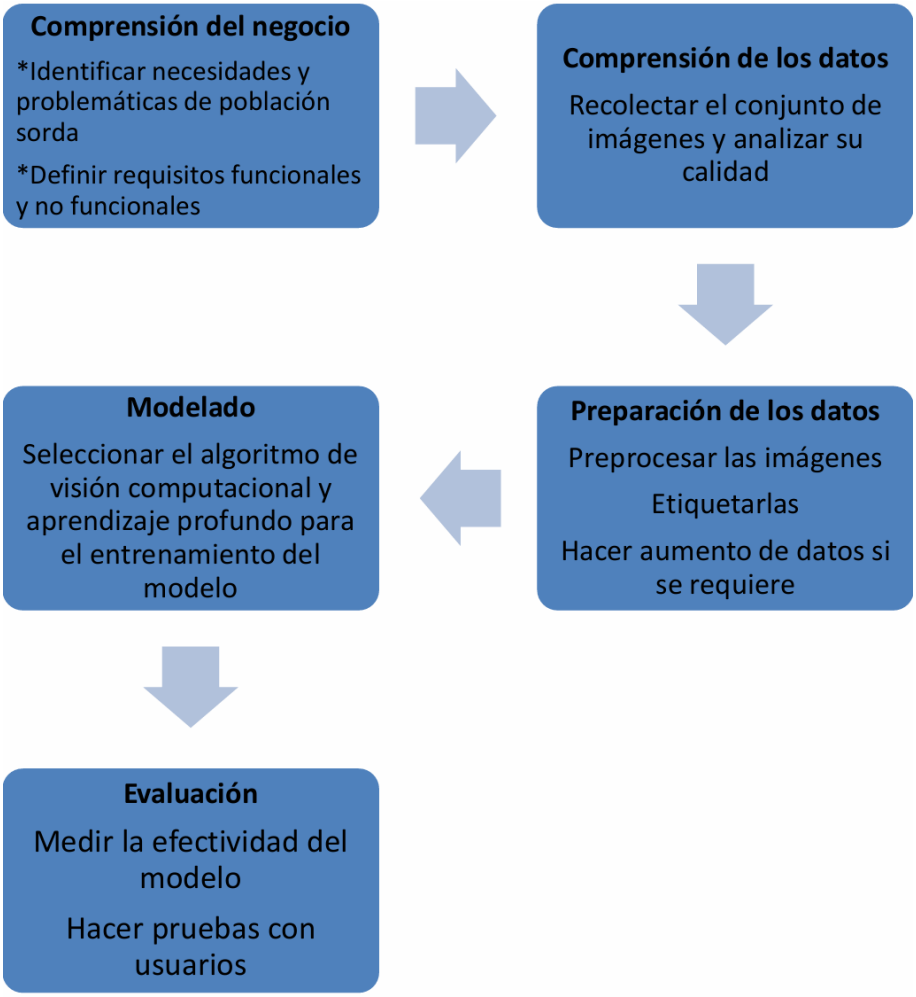
- Integrar el modelo entrenado en una aplicación o sistema accesible para los usuarios.
- Desarrollar una interfaz gráfica para facilitar la interacción con el prototipo.

- Documentar el proceso y proporcionar recomendaciones para futuras mejoras o escalabilidad del sistema

En la Figura 9 se presenta un resumen de la metodología usada.

Figura 9.

Metodología CRIPS aplicada al proyecto



Nota: Diagrama de flujo que detalla las cinco fases metodológicas del proyecto, basado en un enfoque de ciencia de datos (CRISP-DM) aplicado a la visión computacional para la población sorda. *Fuente.* Elaboración propia.

Metodología de Investigación mixta aplicada al Proyecto.

La elección de una metodología con enfoque mixto para esta investigación se justifica en la necesidad de comprender el fenómeno de estudio desde una perspectiva integral. Al combinar herramientas cuantitativas y cualitativas, se logra una visión más completa y profunda, superando las limitaciones que cada enfoque puede tener por separado. Según Torres (2019), este enfoque mixto permite integrar sistemáticamente ambos métodos, manteniendo sus estructuras originales, lo cual favorece una interpretación más rica del objeto de estudio. En este sentido, la complementariedad de los datos numéricos con los relatos, percepciones y contextos cualitativos no solo enriquece el análisis, sino que también permite discernir patrones, contrastar hallazgos y profundizar en las causas y efectos del fenómeno. Esta metodología resulta especialmente valiosa en investigaciones relacionadas con problemáticas sociales o tecnológicas, en las que el impacto humano y los resultados medibles deben considerarse de forma simultánea.

Desde esta perspectiva, el desarrollo del proyecto se estructuró en dos fases fundamentales, concebidas para abordar de manera integral la problemática de comunicación entre personas sordas y oyentes. En la primera fase, se realizó un análisis reflexivo que permitió comprender a profundidad las dinámicas, barreras y desafíos que caracterizan la interacción comunicativa entre ambas comunidades. Esta etapa fue crucial para identificar las necesidades reales de la población sorda y orientar la solución tecnológica hacia un enfoque verdaderamente inclusivo.

Esta disertación se enfoca en la segunda fase del proyecto, cuyo objetivo fue implementar un prototipo de aplicación basado en los requisitos funcionales identificados previamente con la población participante. Dichos requerimientos fueron recolectados mediante encuestas y

entrevistas abiertas estructuradas. El desarrollo de esta fase se organizó en tres etapas fundamentales para la construcción del prototipo.

- Levantamiento de requisitos (a partir de los datos suministrados en la primera fase por los participantes).
- Desarrollo del aplicativo, donde se estableció la plataforma de desarrollo y las tecnologías que son la base del algoritmo y la interfaz gráfica.
- Evaluación de usabilidad efectuada con los usuarios finales puntualizando la función del aplicativo dentro del contexto cotidiano como medio de interacción comunicativa sordo-oyente.

En la siguiente tabla se detalla la metodología Mixta aplicada al proyecto.

Tabla 1.

Metodología Mixta aplicada a la investigación

METODOLOGÍA DE INVESTIGACIÓN	
Tipo de investigación	**Mixta**
FASE 1: Diseño del Prototipo	- Definición del problema - Selección de herramientas (cámara RGB, sensores, etc.) - Planteamiento de objetivos
FASE 2: Recolección de Datos	- Captura de imágenes y videos de señas - Registros de precisión del reconocimiento
FASE 3: Análisis Cuantitativo	- Estadísticas de desempeño del prototipo - Métricas: exactitud, precisión, recall, F1-score
FASE 4: Análisis Cualitativo	- Opiniones de usuarios sordos - Evaluación de accesibilidad y usabilidad - Entrevistas o encuestas
FASE 5: Validación del Modelo	- Comparación de resultados

Nota: Metodología de investigación del proyecto de reconocimiento de lenguaje de señas. El estudio adopta un enfoque mixto (cuantitativo y cualitativo) estructurado en cinco fases.

Fuente. Autor.

Hipótesis

Si se desarrolla un sistema basado en visión por computador para el reconocimiento del alfabeto en Lengua de señas colombiana, entonces se mejorará significativamente la comunicación entre personas sordas y oyentes en comparación con la situación actual sin herramientas tecnológicas.

La interacción con el sistema permitirá reducir la necesidad de intérpretes humanos en contextos cotidianos de comunicación básica (por ejemplo: presentación de nombres, letras, etc.)

Requerimientos del Sistema

Análisis de Requerimientos

Los requisitos funcionales del sistema son fundamentales para garantizar una experiencia de uso adecuada y accesible. Desde una perspectiva crítica, es acertado que estos se hayan definido en función de las necesidades del usuario final, ya que esto permite orientar el desarrollo hacia soluciones prácticas y efectivas. Además, el uso de una escala de priorización del 1 al 5 facilita la organización de tareas y la asignación de recursos, permitiendo enfocar los esfuerzos en las funcionalidades más relevantes para el funcionamiento inicial del prototipo. Esta estrategia demuestra una planificación estructurada y coherente con la metodología del proyecto, favoreciendo una evolución gradual del sistema con base en criterios objetivos de impacto y utilidad. En la siguiente tabla se detallan requerimientos técnicos del sistema.

Tabla 2.

Requerimientos técnicos del sistema

ID	Requerimiento	Descripción
RT01	Tipo de cámara	Cámara RGB con resolución mínima de 720p.
RT02	Framework de desarrollo	Uso de TensorFlow o Python para el modelo CNN.
RT03	Dataset	Conjunto de datos etiquetado con señas del alfabeto colombiano.
RT04	Arquitectura del modelo	Red neuronal convolucional, MediaPipe
RT05	Lenguaje de programación	Python como lenguaje principal.

Nota. Requerimientos técnicos (RT) para el diseño y desarrollo del prototipo de reconocimiento de lenguaje de señas. Los requerimientos especifican los componentes de hardware y software necesarios. *Fuente,* autor

Requerimientos Funcionales y Requerimientos no Funcionales

Requerimientos Funcionales

A continuación, se presentan los requerimientos funcionales asociados al prototipo de reconocimiento del alfabeto de la Lengua de Señas Colombiana (LSC):

Tabla 3.

Requerimiento Funcional – RF-01

N	Código	RF-01
RF-01	Nombre	Captura de imágenes en tiempo real
	Descripción	El sistema debe permitir la captura continua de imágenes mediante una cámara RGB conectada al dispositivo, procesándolas en tiempo real para su posterior análisis y reconocimiento de señas.
	Rol(es)	Usuario
	Entradas	Imagen de una seña perteneciente al Lenguaje de Señas Colombiana (LSC) capturada por la cámara.
	Salidas	Una etiqueta con la letra o palabra identificada por el modelo de reconocimiento.
	Precondiciones	La cámara debe estar conectada y funcionando correctamente; el sistema debe haber iniciado el módulo de captura.
	Postcondiciones	Se muestra en pantalla la etiqueta correspondiente a la seña detectada en tiempo real.
	Prioridad	Alta
	Dependencias	Requiere la inicialización del módulo de visión artificial y del modelo de reconocimiento entrenado.

Nota: Detalle del requerimiento funcional RF-01: Captura de imágenes en tiempo real. Este requerimiento, de prioridad alta, describe la capacidad esencial del sistema para procesar el video de una cámara RGB de forma continua. Fuente. Autor.

Tabla 4.

Requerimiento funcional del sistema RF-02

N	**Código**	**RF-02**
RF-02	Nombre	Detección automática de la mano
	Descripción	El sistema debe detectar la presencia de una mano dentro del flujo de video capturado por la cámara RGB. Esta detección debe ejecutarse en tiempo real utilizando un modelo de MediaPipe Hand Landmarker para identificar la ubicación general de la mano antes del análisis detallado de la seña.
	Rol(es)	Usuario
	Entradas	Flujo de video capturado por la cámara. Mano del usuario dentro del campo de visión.
	Salidas	Señal de detección activa de mano.
	Precondiciones	La cámara debe estar activa y en operación. El módulo de detección de mano debe estar correctamente cargado
	Postcondiciones	El sistema activa el proceso de estimación de puntos clave cuando se detecte una mano
	Prioridad	Alta
	Descripción	RF-01 Captura continua de imágenes. Inicialización del modelo Hand Landmarker de MediaPipe.

Nota: Detalle del requerimiento funcional RF-02: Detección automática de la mano. Este requerimiento de prioridad alta describe el proceso de visión artificial que identifica la presencia y ubicación de una mano dentro del flujo de video capturado. *Fuente*. autor.

Tabla 8.

Requerimiento funcional del sistema RF-03

N	**Código**	**RF-03**
RF-03	Nombre	Extracción de landmarks de la mano
	Descripción	El sistema debe estimar en tiempo real los 21 puntos clave (landmarks) de la mano detectada utilizando el modelo Hand Landmarker de MediaPipe, proporcionando información espacial necesaria para el reconocimiento de la seña.
	Rol(es)	Usuario
	Entradas	Imagen de la mano previamente detectada.
	Salidas	Conjunto de 21 coordenadas normalizadas correspondientes a los puntos clave de la mano. Representación visual opcional de los landmarks en la interfaz.
	Precondiciones	El módulo de detección de mano debe haber identificado una mano en la imagen. El modelo de landmarks debe estar correctamente inicializado.
	Postcondiciones	Los datos de landmarks quedan disponibles para el módulo de reconocimiento de señas.
	Prioridad	Alta
	Descripción	RF-02 Detección automática de la mano. Modelo Hand Landmarker operativo.

Nota: Detalle del requerimiento funcional RF-03: Extracción de landmarks de la mano. Este requerimiento de prioridad alta se enfoca en el preprocesamiento de la imagen, donde el sistema debe estimar los 21 puntos clave (landmarks) de la mano detectada en tiempo real utilizando el modelo MediaPipe Hand Landmarker. *Fuente*. Autor.

Tabla 9.

Requerimiento funcional del sistema RF-04

N	**Código**	**RF-04**
RF-04	Nombre	Identificación de la seña del LSC
	Descripción	El sistema debe clasificar la seña realizada por el usuario utilizando un modelo entrenado que tome como entrada las coordenadas de los landmarks y determine la letra o palabra correspondiente del Lenguaje de Señas Colombiano (LSC).
	Rol(es)	Usuario
	Entradas	Coordenadas de los 21 landmarks estimados por MediaPipe.
	Salidas	Etiqueta con la letra, número o palabra reconocida.
	Precondiciones	El módulo de extracción de landmarks debe haber entregado correctamente los datos. El modelo de reconocimiento debe estar cargado y operativo.
	Postcondiciones	La etiqueta clasificada queda disponible para ser mostrada al usuario o registrada por el sistema.
	Prioridad	Alta
	Descripción	RF-03 Extracción de landmarks. Modelo de clasificación entrenado y disponible.

Nota: Detalle del requerimiento funcional RF-04: Identificación de la seña del LSC. Este requerimiento, de prioridad alta, describe la función principal del sistema: la clasificación del gesto. *Fuente.* Autor.

Tabla 10.

Requerimiento funcional del sistema RF-05

N	Código	RF-05
RF-05	Nombre	Presentación de resultados en la interfaz
	Descripción	El sistema debe mostrar en la interfaz del usuario, en tiempo real, la seña identificada por el modelo, destacando la etiqueta correspondiente
	Rol(es)	Usuario
	Entradas	Etiqueta generada por el módulo de clasificación.
	Salidas	Visualización en pantalla de la letra o palabra reconocida. Visualización de landmarks sobre la imagen en tiempo real.
	Precondiciones	El modelo debe haber generado una etiqueta válida. La interfaz gráfica debe estar activa.
	Postcondiciones	El usuario observa en tiempo real la seña interpretada por el sistema.
	Prioridad	Media
	Descripción	RF-04 Identificación de la seña.

Nota: Detalle del requerimiento funcional RF-05: Presentación de resultados en la interfaz. Este requerimiento de prioridad media describe la función de interfaz de usuario (UI). Su propósito es tomar como entrada la etiqueta clasificada por el modelo (RF-04) y mostrarla en tiempo real. *Fuente*. Autor.

Requerimientos no Funcionales

A continuación, se presentan los requerimientos no funcionales para el reconocimiento del alfabeto de la Lengua de Señas Colombiana:

Tabla 11.

Requerimiento No Funcionales del Sistema RNF-01

N	Nombre	Descripción del Requerimiento
RNF-01	Rendimiento	• El sistema debe procesar cada imagen o fotograma en menos de 0.5 segundos para garantizar una experiencia en tiempo real. • Debe permitir el reconocimiento continuo (streaming) sin pausas perceptibles durante la ejecución.

Nota: Detalle del requerimiento no funcional RNF-01: Rendimiento. Este requerimiento establece las métricas de tiempo real para el prototipo. El sistema debe garantizar que el procesamiento de cada fotograma (detección y clasificación de la seña) se complete en menos de 0.5 segundos (500 milisegundos). *Fuente*. Autor.

Tabla 12.

Requerimiento No Funcionales del Sistema RNF-02

N	Nombre	Descripción del Requerimiento
RNF-02	Usabilidad	• La interfaz debe ser sencilla, intuitiva y permitir al usuario interactuar con el sistema sin requerir conocimientos técnicos. Debe mostrar mensajes claros sobre detecciones, errores y estado de la cámara.

Nota. Detalle del requerimiento no funcional RNF-02: Usabilidad. Este requerimiento se centra en la experiencia del usuario y la interfaz. Estipula que la interfaz debe ser sencilla e intuitiva, diseñada para ser utilizada por cualquier usuario sin requerir conocimientos técnicos. *Fuente*. Autor.

Tabla 13.

Requerimiento No Funcionales del Sistema RNF-03

N	Nombre	Descripción del Requerimiento
RNF-03	Seguridad-Protección de datos capturados	El sistema no debe almacenar imágenes ni video del usuario a menos que sea explícitamente autorizado. El acceso a la cámara debe solicitar permisos y cerrarse correctamente al finalizar el uso

Nota. Detalle del requerimiento no funcional RNF-03: Seguridad - Protección de datos capturados. Este requerimiento establece políticas de privacidad y manejo de la información visual. *Fuente*. Autor.

Tabla 14.

Requerimiento No Funcionales del Sistema RNF-04

N	Nombre	Descripción del Requerimiento
RNF-04	Mantenibilidad	• Se debe entregar documentación clara para desarrolladores, y usuarios mediante los manuales correspondientes.

Nota: Detalle del requerimiento no funcional RNF-04: Mantenibilidad. Este requerimiento establece la necesidad de documentación clara y completa para el proyecto. *Fuente*. Autor.

Actores del sistema

Los actores del sistema, para el prototipo de inteligencia artificial para la interpretación del Lenguaje de Señas Colombiana (LSC) son:

Tabla 14.

Actores del Sistema

Tipo de Actor	Descripción
Usuario Sordo	Persona perteneciente a la comunidad sorda que utiliza el sistema para facilitar la comunicación con oyentes mediante el reconocimiento automático del alfabeto o señas en LSC.

Usuario Oyente /	Persona oyente interesada en comprender la lengua de señas a través del prototipo. Puede ser familiar, docente o profesional en contextos educativos o laborales.
Administrador	Persona encargada de la gestión de usuarios y de gestionar la base de datos de señas.

Nota: Tipos de actores identificados para el sistema de reconocimiento de lenguaje de señas.

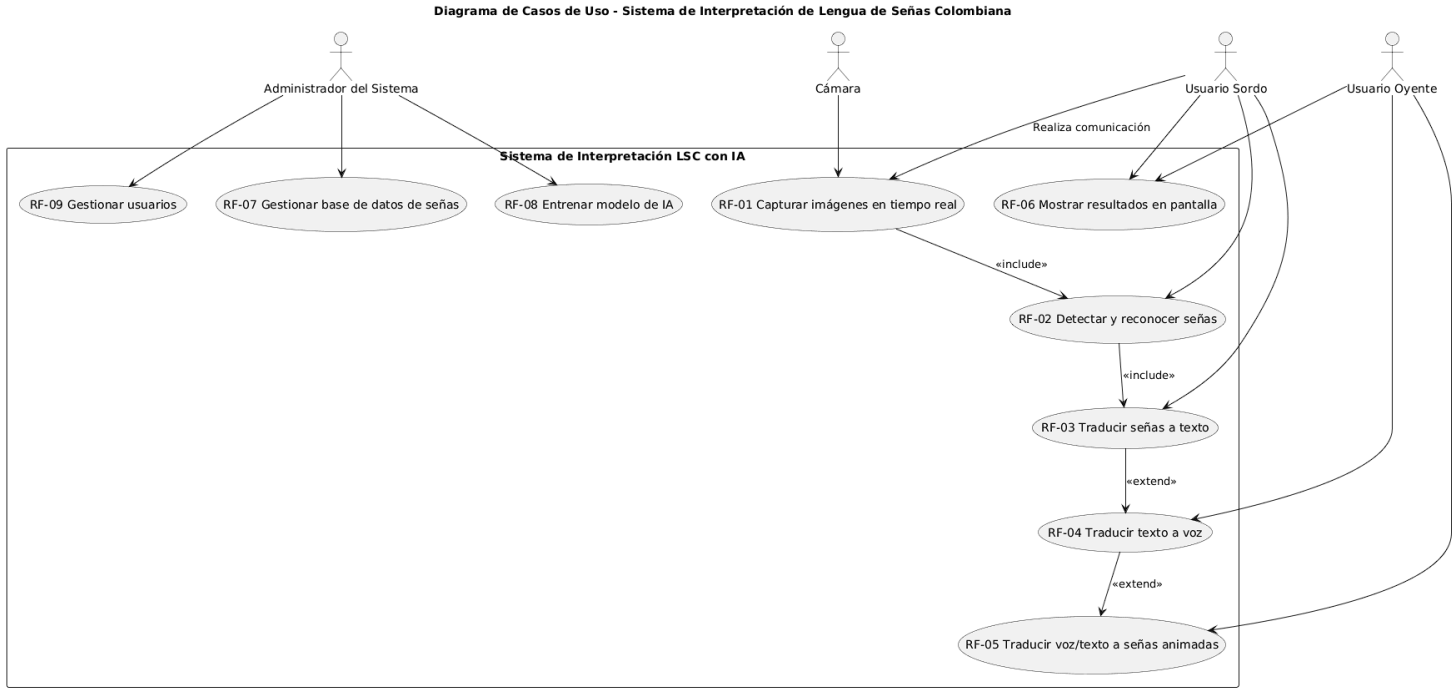
Fuente: Autor.

Diagrama de casos de uso.

A continuación, se presenta el diagrama de casos de uso de una aplicación de inteligencia artificial para la interpretación de la Lengua de Señas Colombiana (LSC). Este diagrama incluye los actores y funcionalidades más comunes del sistema:

Figura10.

Diagrama de Casos de Uso del Sistema.



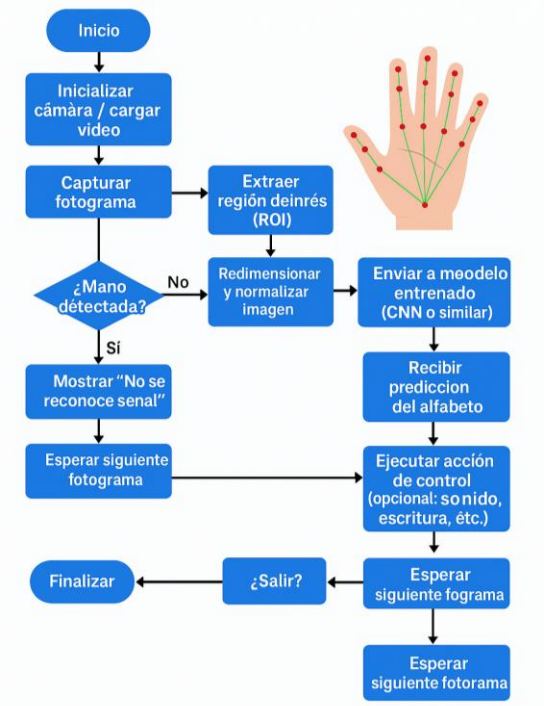
Nota. El diagrama UML modela las interacciones funcionales del sistema de inteligencia artificial. Se detallan los roles de los actores humanos (usuario sordo, usuario oyente, administrador) y de hardware (cámara), así como las relaciones de inclusión (include) y extensión (extend) entre los procesos de traducción de señas, texto y voz. *Fuente:* El autor.

Diagrama de flujo del sistema

En la Figura 11 se presenta el diagrama de flujo que describe el funcionamiento general del prototipo de reconocimiento de señas del Lenguaje de Señas Colombiano (LSC) mediante visión artificial. En él se representa la secuencia lógica de módulos que componen el sistema, iniciando con la captura continua de imágenes a través de la cámara y avanzando hacia la detección de la mano, la estimación de puntos clave con MediaPipe Hand Landmarker y la posterior clasificación de la seña mediante un modelo entrenado.

Figura 11.

Diagrama del Sistema: Reconocimiento de LSC con Visión Artificial.



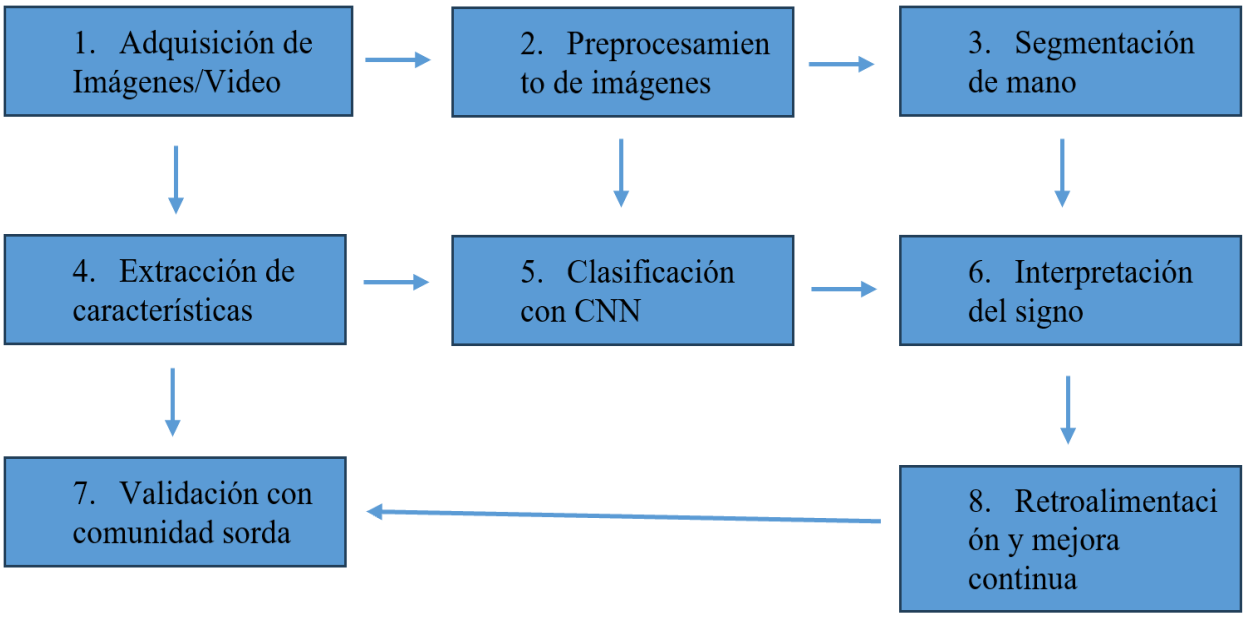
Nota: Diagrama que ilustra el proceso algorítmico para el reconocimiento de señas mediante visión artificial en tiempo real. *Fuente:* Autor.

Arquitectura del Sistema

La arquitectura del sistema para el reconocimiento de señas del Lenguaje de Señas Colombiano (LSC) se compone de un flujo secuencial que integra técnicas de visión artificial y aprendizaje profundo. El proceso inicia con la adquisición de imágenes o video en tiempo real, seguido por un preprocesamiento que optimiza la calidad visual para facilitar la segmentación precisa de la mano. Posteriormente, se realiza la extracción de características relevantes que alimentan un modelo de clasificación basado en redes neuronales convolucionales (CNN) apoyado en MediaPipe Studio, el cual determina la seña realizada. La salida del modelo se interpreta y se traduce en un signo comprensible para el usuario. Además, se incorpora una fase de validación con la comunidad sorda, cuyo propósito es asegurar que las señas reconocidas sean cultural y lingüísticamente correctas.

Figura 12.

Arquitectura del Prototipo de Reconocimiento del Alfabeto LSC.



Nota. Diagrama de flujo que resume las ocho fases de la metodología implementada para el reconocimiento automático de señas, desde la adquisición de datos y el preprocesamiento hasta la clasificación con CNN y el ciclo de validación y mejora continua con la comunidad sorda. *Fuente:* Autor.

A continuación, se detallan los diferentes componentes técnicos usados en el desarrollo del prototipo:

Tabla 5.

Tecnologías Utilizadas.

Componente	**Herramienta**	**Versión**	**Función**
Captura	OpenCV	4.5+,	Adquisición y preprocesamiento de video.
Detección	MediaPipe Hands	0.8.11	Landmarks de manos en tiempo real.
Modelado	TensorFlow/Keras,	2.10+	Entrenamiento de CNN.
Implementación	Flask	2.2.x	API REST para despliegue.

Nota. Arquitectura tecnológica del sistema de reconocimiento de señas. La tabla detalla los componentes de software clave utilizados, su versión y función en el flujo de trabajo. *Fuente.* Autor.

Recolección de datos

La fase de recolección de datos se llevó a cabo mediante la captura sistemática de imágenes de distintas señas utilizando un teléfono inteligente, con el fin de construir un conjunto de datos representativo y adecuado para el entrenamiento de los modelos de visión por computadora. Este proceso consideró variaciones naturales en las condiciones de iluminación, ángulos de captura y características individuales de las manos, garantizando diversidad suficiente para mejorar la capacidad de generalización del sistema. Además, se establecieron procedimientos consistentes para asegurar la calidad de las imágenes y mantener la uniformidad en la adquisición, lo que constituye un insumo fundamental para el posterior preprocesamiento y modelado.

Captura de imágenes de señas (dataset)

La captura de imágenes se realizó teniendo en cuenta:

- **Instrumento**: Cámara digital o cámara de un teléfono inteligente con alta resolución.
- **Finalidad**: Recolectar imágenes de las manos representando las letras del alfabeto dactilológico de la LSC.
- **Formato de datos**: Imágenes en formato JPG o PNG, organizadas por carpeta (una por cada letra del alfabeto).
- **Condiciones de captura**: Fondos neutros, buena iluminación, diferentes ángulos, posiciones de manos, y variabilidad en los usuarios para mejorar la generalización del modelo.

Creación del dataset.

Para la creación del dataset se seleccionaron 25 señas del diccionario del INSOR y algunas palabras básicas de la lengua de señas colombiana como: “gracias”, “bien”, “buenos días”, “buenas tardes”, “qué pasa”, “porqué”, “para qué”, “hola”, “por favor”, “quién”, “dónde”, “qué”, “cómo”. Sin embargo, trabajar con lenguas de señas es intrínsecamente desafiante por la diversidad dialectal, la variabilidad individual y las sutilezas gestuales, esto enfatiza la necesidad de validar el sistema con usuarios reales y de continuar ampliando el conjunto de señas para cubrir mayor representatividad cultural.

La selección de las 25 señas para el prototipo demuestra una estrategia práctica y enfocada en la accesibilidad. Sin embargo, trabajar con lengua de señas sigue siendo un reto considerable, especialmente para la comunidad sorda, pues la precisión y utilidad real dependen de pruebas en contextos reales y de la adaptación a las diversas formas de comunicación existentes. En la Figura 12 se detalla el proceso de creación del Dataset

Figura 4.

Detalle de creación del dataset.

Nota: Diagrama que ilustra las fases de preparación de los datos utilizadas para el entrenamiento del modelo. *Fuente.* Autor.

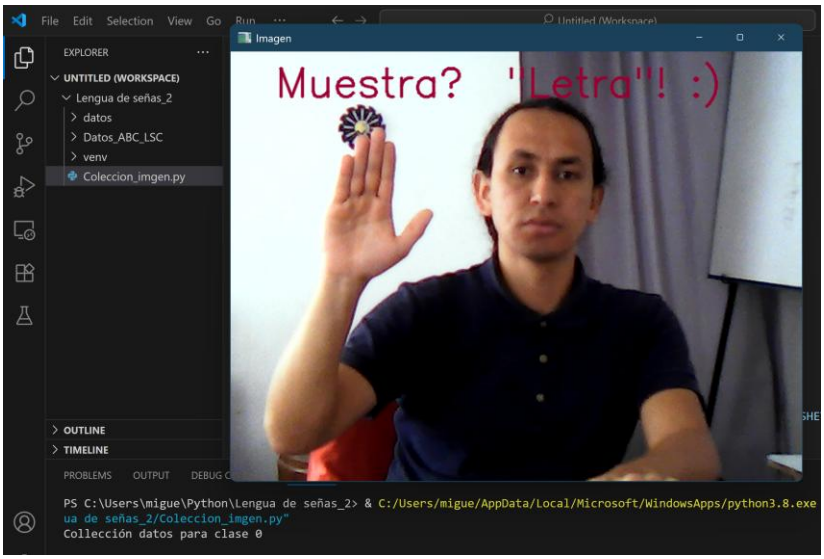
Para la captura de imágenes se tuvieron en cuenta las siguientes recomendaciones:

Recomendaciones de captura:

- Escenario: fondo neutro si es posible; luego incorporar fondos diversos para robustez.
- Iluminación: difusa (evitar contraluces). Capturar en 3–4 niveles de brillo.
- Dispositivo: cámara HD (≥720p). Tasa 30 fps.
- Variabilidad: mano derecha/izquierda, rotaciones ±15°–30°.
- Etiquetado: un archivo Datos con sus etiquetas

Figura 5.

Dataset de imágenes capturadas para el entrenamiento del modelo



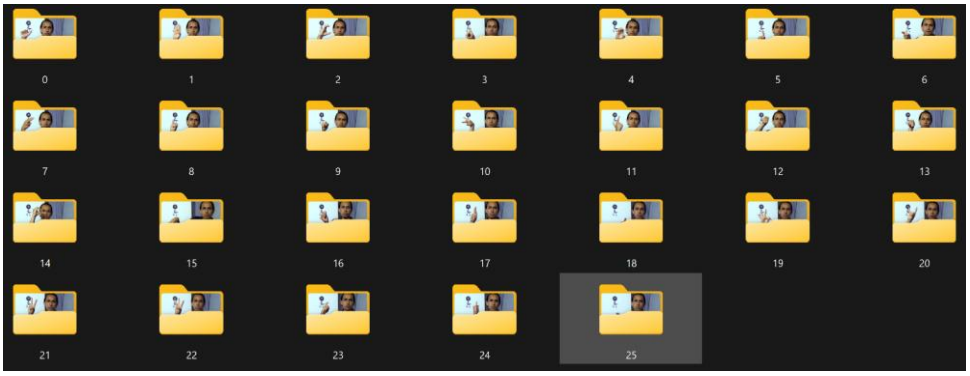
Nota: Captura de pantalla que ilustra la interfaz de la aplicación de colección de imágenes desarrollada en Python y Visual Studio Code. Muestra al usuario realizando una seña de la Lengua de Señas Colombiana (LSC) frente a la cámara. *Fuente.* Autor.

Construcción de la base de datos

El proceso de segmentación y etiquetado por seña permitió organizar el dataset de forma estructurada, asignando una carpeta por cada letra del alfabeto en lengua de señas colombiana. La recolección de alfabeto imágenes provenientes de dos dispositivos distintos supuso un reto técnico por la diferencia en dimensiones, pero la estandarización previa al entrenamiento garantizó uniformidad y calidad en los datos. Este paso fue clave para asegurar que el modelo de visión artificial pueda interpretar correctamente las señas y reducir errores de reconocimiento.

Figura 14.

Dataset de carpeta de imágenes alfabetos.



Nota: Captura de pantalla que muestra la estructura jerárquica de carpetas utilizada para almacenar las imágenes del conjunto de datos. *Fuente*. Autor

Una vez organizado el conjunto de datos, en la tabla, se muestran las letras del abecedario, la letra en lenguaje de señas y el número que la representa en el algoritmo el cual corresponde a la etiqueta de la imagen.

Tabla 6.

Representación del lenguaje de señas y alfabético.

Letra LSC	Letra ABC es A - Z	N° representativo
	A	0
	B	1
	C	2

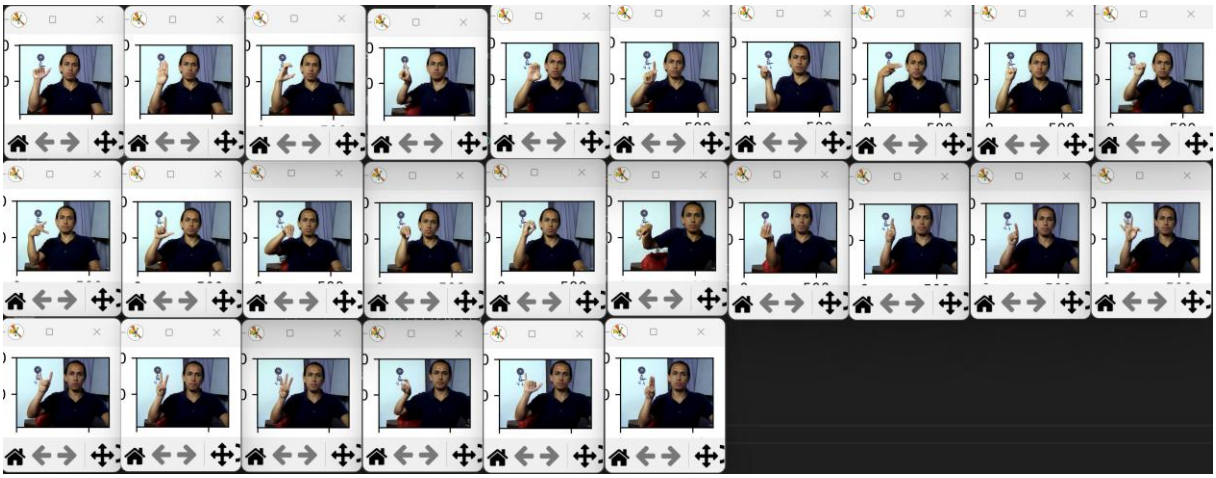
	D	3
	E	4
	F	5
	G	6
	H	7
	I	8
	J	9
	K	10
	L	11
	M	12
	N	13
	O	14
	P	15
	Q	16
	R	17
	S	18
	T	19
	U	20
	V	21
	W	22
	X	23
	Y	24
	Z	25

Nota: La tabla muestra la correspondencia directa entre cada imagen del gesto capturado y su respectiva letra del alfabeto en español, sirviendo como el esquema de etiquetado para el modelo de reconocimiento. *Fuente*. Autor.

A continuación, se presenta un ejemplo del dataset:

Figura 6.

Ejemplos del dataset: variación de letras, ángulos y sujetos.



Nota: Captura de pantalla que muestra una matriz de múltiples fotogramas de video (frames) extraídos de una secuencia de captura. *Fuente*. Autor.

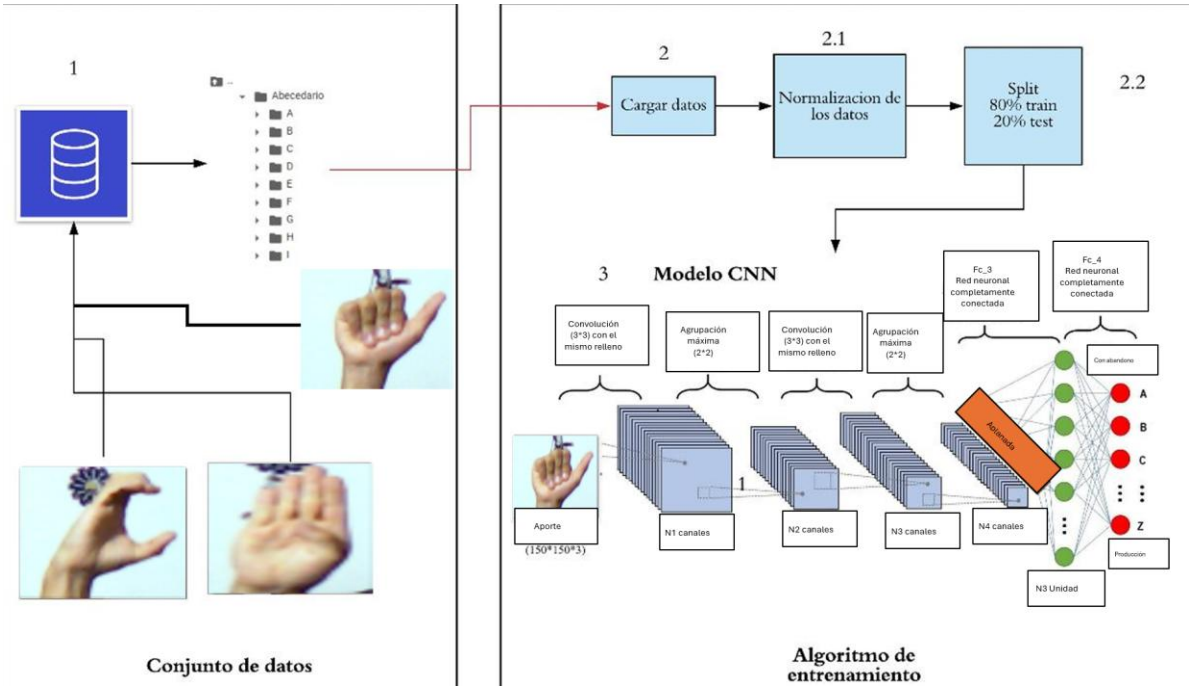
Creación del modelo.

Diseño del modelo

La etapa de modelado se estructuró a partir del flujo ilustrado en la Figura 14, donde se integran los procesos necesarios para entrenar un modelo de clasificación basado en redes neuronales convolucionales (CNN) apoyado por MediaPipe. Inicialmente, los datos del conjunto de imágenes etiquetadas son cargados y sometidos a una normalización que garantiza la homogeneidad en las dimensiones y valores de entrada. Posteriormente, los datos se dividen en subconjuntos de entrenamiento y prueba, empleando una proporción 80/20 con el fin de evaluar el desempeño del modelo de manera objetiva. El modelo CNN se compone de múltiples capas de convolución y agrupación, diseñadas para extraer progresivamente patrones espaciales relevantes de las señas manuales. Estas capas se conectan finalmente con redes completamente conectadas encargadas de la clasificación final de cada imagen.

Figura 7.

Modelo del prototipo de predicción de lenguaje de señas usando CNN.



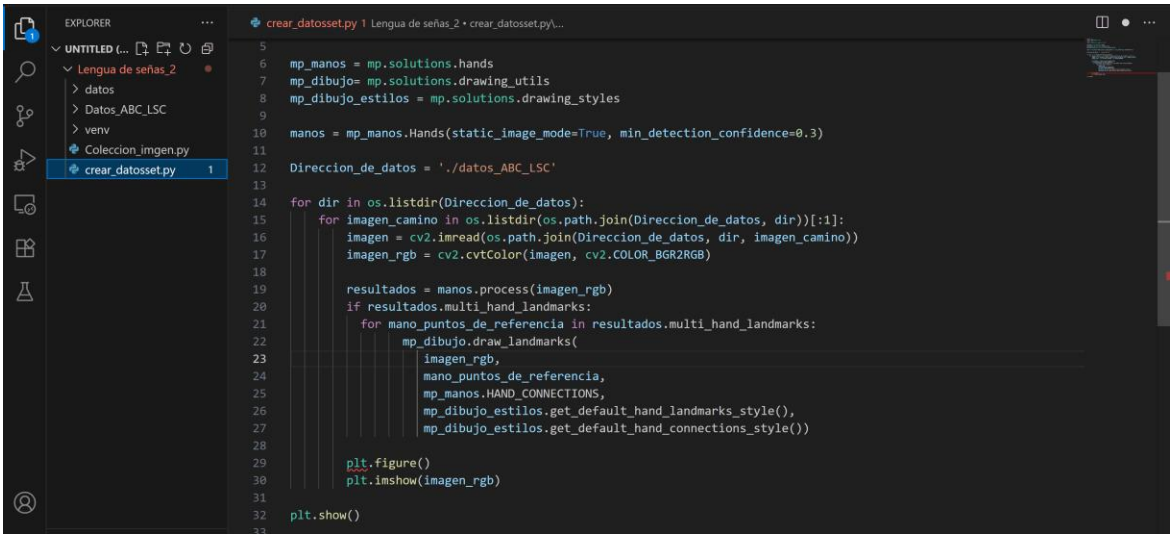
Nota: Diagrama del conjunto de datos y la arquitectura del modelo de Red Neuronal Convolucional (CNN). *Fuente*. Autor

Creación del modelo

Para la creación del modelo se usó Python el cual reúne herramientas maduras (OpenCV, TensorFlow/Keras, PyTorch, albumentations) que aceleran el ciclo de desarrollo. Sin embargo, es importante aclarar que el éxito depende más de la calidad y representatividad del dataset que de la arquitectura del modelo. Invertir tiempo en recolección, etiquetado y preprocesamiento dará mejores resultados que cambiar constantemente la red neuronal.En Python se implementó la funcionalidad de MediaPipe Hands

Figura 8

Código en Python para procesar imágenes con MediaPipe Hands.

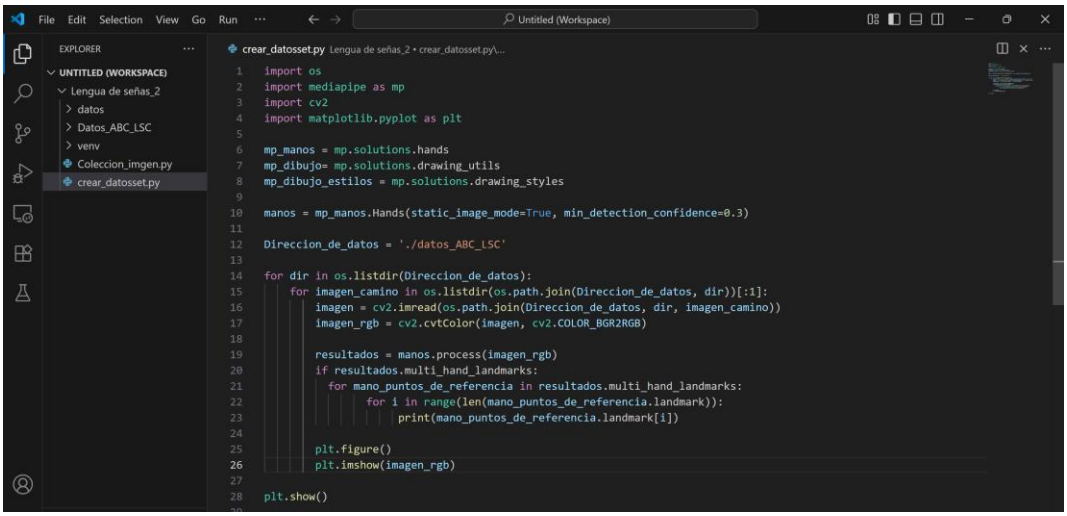


Nota. Captura de pantalla del archivo crear_datosset.py mostrando código para el procesamiento de imágenes de la colección Datos ABC_LSC. El script utiliza la clase mp.solutions.hands de MediaPipe, lee imágenes con cv2.imread, las procesa con manos.process(), e itera para dibujar las conexiones y puntos de referencia (mp_dibujo.draw_landmarks) sobre la imagen original. *Fuente.* Autor.

En la siguiente imagen se muestra un ejemplo de la creación del dataset y la estructura de datos

Figura 9.

Proceso de Creación del Dataset en Python



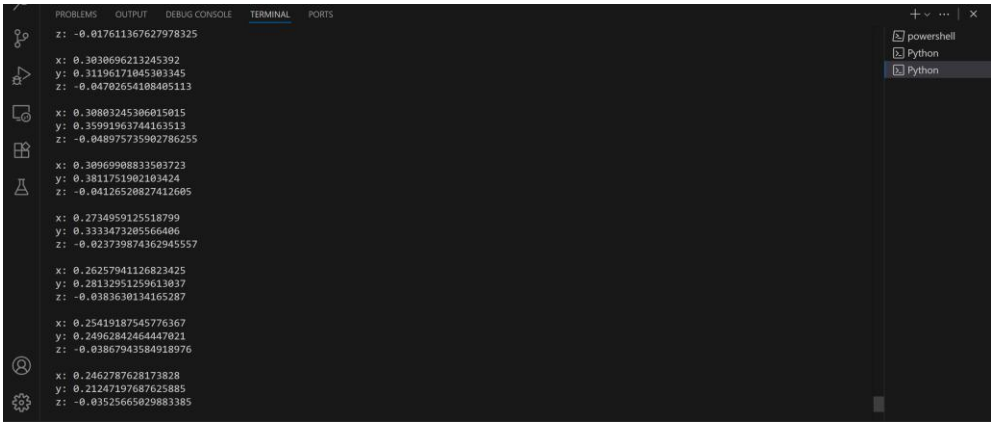
Nota: Captura de pantalla del archivo crear_datosset.py mostrando la parte del código encargada de procesar imágenes (cv2.imread) usando el modelo de detección de manos de MediaPipe (mp.solutions.hands). *Fuente*. Autor.

Descripción:

- Los valores x, y, z mostrados en la terminal corresponden a las coordenadas 3D de los 21 landmarks de mano detectados por MediaPipe.
- x: Posición horizontal (0 a 1, donde 1 = ancho de la imagen).
- y: Posición vertical (0 a 1, donde 1 = altura de la imagen).
- z: Profundidad (valores negativos indican proximidad a la cámara).

Figura 10.

Estructura de Datos de Landmarks.



Nota: Cada grupo de coordenadas X, Y y Z representa la posición normalizada (valores entre 0.0 y 1.0) de un punto de referencia clave (landmark) detectado en la mano por el modelo de MediaPipe. *Fuente*. MediaPipe Hand Landmarks Model (Documentación Oficial)

Train prueba modelo

El archivo Train_prueba.py contiene el código para entrenar y evaluar un modelo de clasificación basado en aprendizaje automático. A continuación, se describe el flujo del proceso:

Preparación de Datos

Se cargan los datos previamente procesados desde un archivo. pkl usando pickle.

Los datos (dato) y sus etiquetas (etiquetas) se convierten en arrays de NumPy para facilitar su manipulación.

División de Datos

Se utiliza train_test_split para dividir los datos en conjuntos de entrenamiento (80%) y prueba (20%), asegurando una distribución balanceada con stratify.

Entrenamiento del Modelo

Se entrena un modelo RandomForestClassifier con los datos de entrenamiento.

Este modelo es adecuado para clasificación multiclase como en el caso del alfabeto en lengua de señas.

Evaluación

Se realizan predicciones sobre el conjunto de prueba.

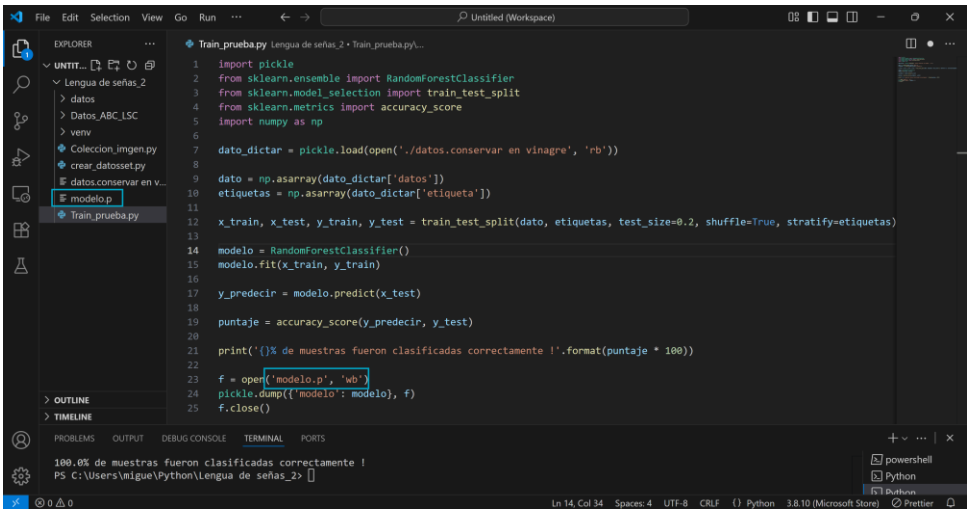
Se calcula la precisión con accuracy_score, mostrando un resultado del 100% en la terminal, lo que indica que todas las muestras fueron clasificadas correctamente en esta ejecución.

Guardado del Modelo

En la siguiente figura se detalla el archivo Train_prueba.py que detalla el proceso de clasificación. El código carga datos previamente procesados (línea 8), los divide en conjuntos de entrenamiento y prueba (train_test_split), inicializa y entrena un clasificador de Bosque Aleatorio (RandomForestClassifier), y evalúa su rendimiento con accuracy_score. Finalmente, el modelo entrenado es serializado y guardado en un archivo binario (modelo.p) utilizando la librería pickle para su posterior uso.El modelo entrenado se guarda en un archivo modelo.p para su uso posterior en la etapa de inferencia.

Figura 11.

Salida en la terminal mostrando la precisión del modelo.



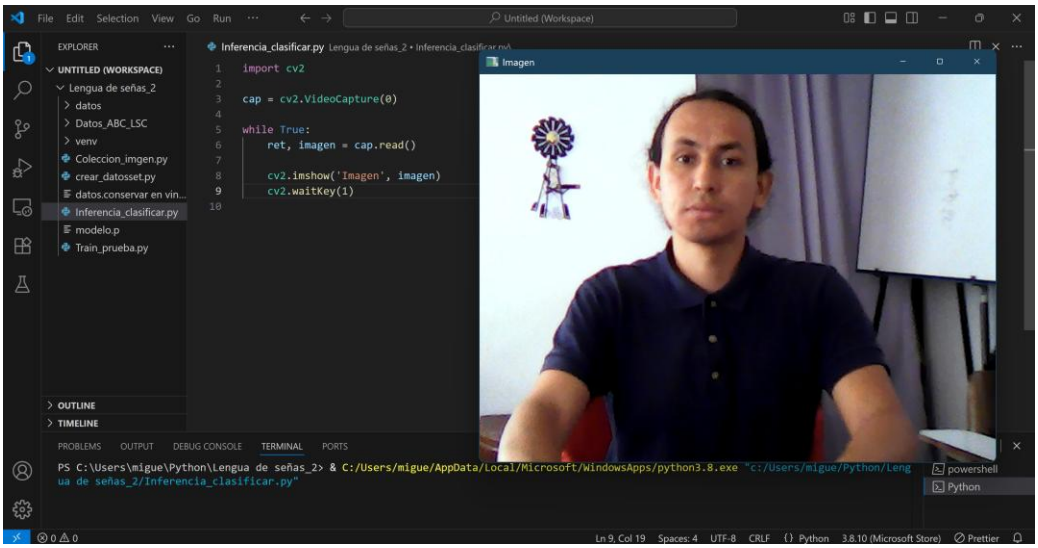
Nota. El archivo Train_prueba.py entrena un Bosque Aleatorio con datos procesados, evalúa su precisión y luego serializa el modelo entrenado en modelo.p para su uso futuro. Ejecución del script Train_prueba.py. *Fuente.* Autor.

Inferencia

La etapa de inferencia corresponde al momento en el que el modelo previamente entrenado es utilizado para clasificar nuevas imágenes capturadas en tiempo real o provenientes de un conjunto de prueba. En este proceso, la imagen de entrada es preprocesada para mantener las mismas condiciones utilizadas durante el entrenamiento, incluyendo el escalado de dimensiones, normalización de valores de píxeles y transformación a formato tensor.

Figura 12.

Proceso de inferencia y clasificación del modelo CNN.

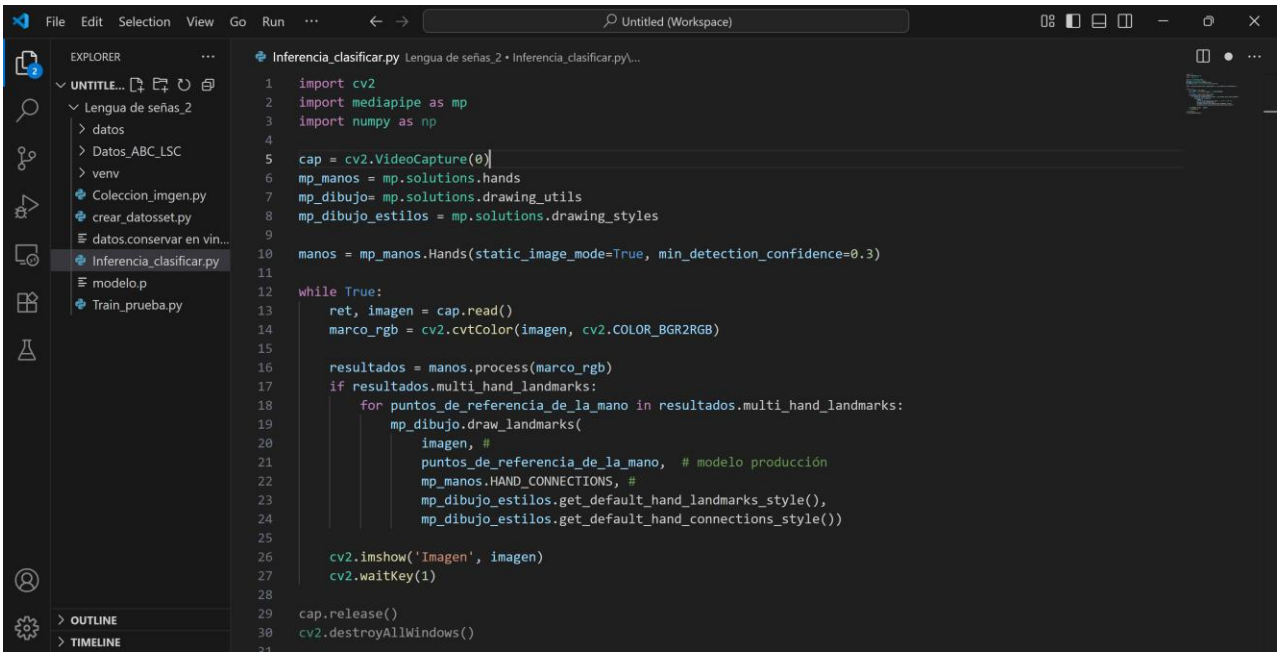


Nota: Captura de pantalla del archivo Inferencia_clasificar.py en ejecución. El código utiliza la librería OpenCV (cv2.VideoCapture) para inicializar y leer fotogramas (cap.read()) desde la cámara de video en un bucle continuo (while True:). *Fuente.* Autor.

Este enfoque permite que el prototipo funcione en tiempo real, integrando captura, detección, extracción de características y clasificación en un único flujo de ejecución, optimizando así el rendimiento y la precisión del sistema.

Figura 13.

Script para la clasificación de señas usando MediaPipe y OpenCV.



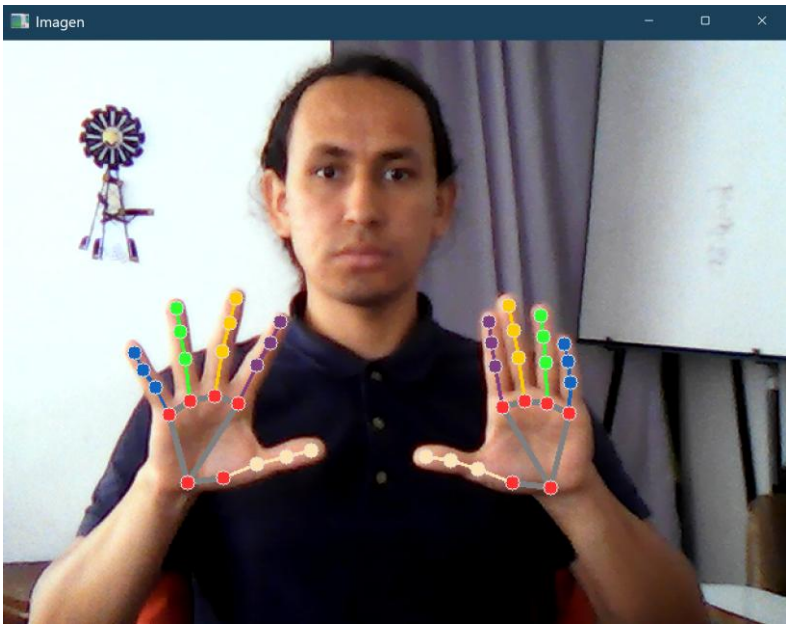
Nota. Captura de pantalla del script en Python (Inferencia_clasificador.py) dentro del entorno de desarrollo Visual Studio Code (VS Code). El código implementa un sistema de detección de puntos de referencia de la mano (hand landmark detection) utilizando la librería MediaPipe (mp.solutions.hands) y OpenCV (cv2) para el manejo del video. *Fuente.* Autor

Visualización

- La letra reconocida se muestra en pantalla junto con la imagen procesada.
- Se puede incluir retroalimentación visual como el nombre de la letra y un borde de color para indicar éxito en la clasificación.

Figura 14

Interfaz de salida mostrando la letra clasificada en tiempo real.



Nota. Ventana de visualización de video (cv2.imshow) mostrando el resultado de la ejecución del script de clasificación. La imagen captura a una persona con ambas manos levantadas, sobre las cuales el modelo de inteligencia artificial ha aplicado con éxito la detección de puntos de referencia de la mano (Hand Landmark Detection). *Fuente.* Autor.

En esta etapa, se realiza la inferencia y clasificación en tiempo real del alfabeto de la Lengua de Señas Colombiana. El script carga el modelo previamente entrenado desde un archivo serializado (modelo.p) y accede a la cámara para capturar fotogramas. Con MediaPipe Hands se detectan y extraen los puntos de referencia (landmarks) de la mano, que luego son utilizados como entrada para el modelo de predicción. Finalmente, se muestra en pantalla la imagen con las manos detectadas y se obtiene la predicción correspondiente a la seña capturada.

Figura 15.

Código de inferencia para la clasificación del modelo.

Nota: Captura de pantalla del script en Python (Inferencia_clasificador.py) dentro del entorno de desarrollo Visual Studio Code. Este código muestra la implementación completa de un sistema de clasificación de gestos de lenguaje de señas, integrando la detección de puntos de referencia de la mano (MediaPipe) con un modelo de predicción de Machine Learning serializado. *Fuente*: Autor.

La inferencia es la etapa en la que el sistema reconoce en tiempo real las letras del alfabeto colombiano en lengua de señas. El proceso se realiza mediante el siguiente flujo:

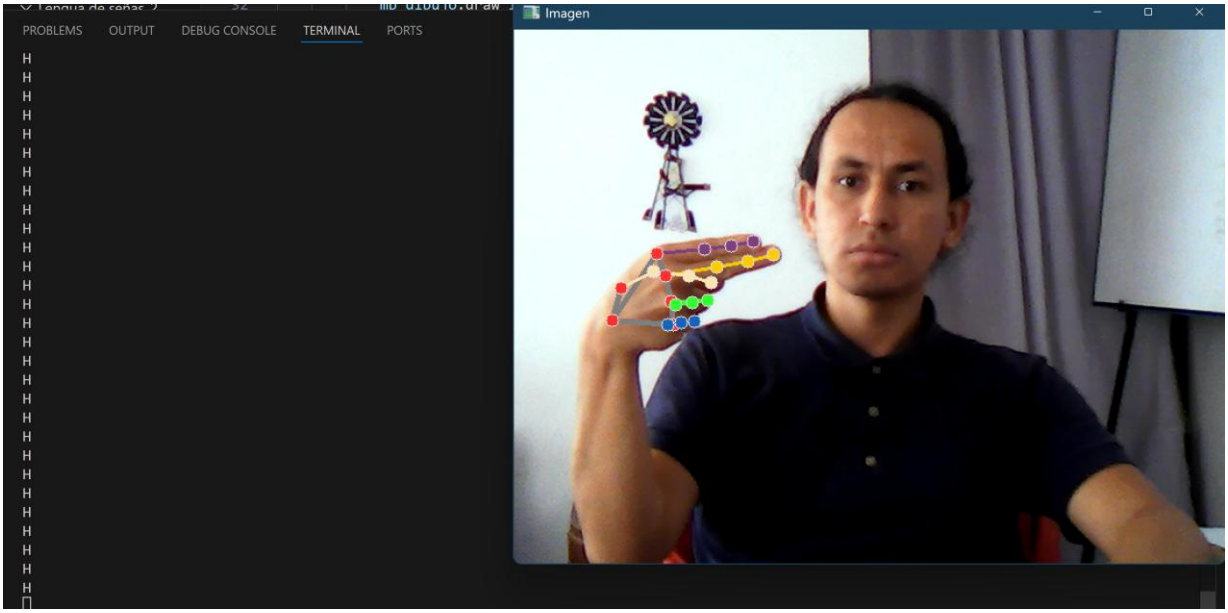
Figura 16.

Script inicializa la captura de video - cv2.VideoCapture.

Nota: El script inicializa la captura de video (cv2.VideoCapture), carga un modelo de clasificación previamente entrenado (línea 6) y un diccionario de etiquetas (etiquetas_dictar, líneas 17-19), e implementa el bucle principal (while True) *Fuente.* Autor.

Figura 17.

Keypoints detectados por visión artificial.



Nota. Captura de pantalla mostrando la ventana de video y la consola de depuración del entorno Visual Studio Code durante la ejecución del script de clasificación de gestos. La imagen presenta a una persona realizando el gesto correspondiente a la letra "H" del alfabeto manual. *Fuente.* Autor.

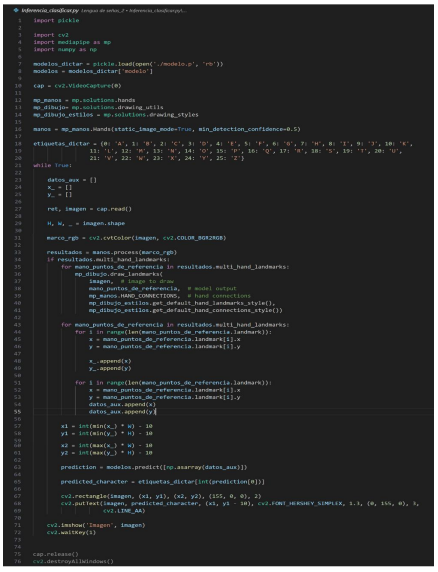
La etapa de inferencia permite reconocer en tiempo real las letras del alfabeto de la Lengua de Señas Colombiana mediante visión artificial y aprendizaje automático. El proceso se realiza como sigue:

- Se inicia la cámara con cv2.VideoCapture(0) para capturar imágenes en tiempo real.
- Se utiliza MediaPipe Hands para detectar los 21 puntos clave de la mano.
- Los landmarks se dibujan sobre la imagen con estilos personalizados usando mp_dibujo.draw_landmarks.

- Se recorren los landmarks y se extraen las coordenadas x e y de cada punto.
- Estas coordenadas se almacenan en un vector datos_aux, que representa la postura de la mano.
- El vector se pasa al modelo previamente entrenado (modelo.p) mediante modelos.predict(...).
- El índice predicho se traduce a una letra usando el diccionario etiquetas_dictar.
- Se dibuja un rectángulo alrededor de la mano y se muestra la letra predicha sobre la imagen.
- La salida se visualiza en una ventana de OpenCV (cv2.imshow('Imagen', imagen)), actualizada en tiempo real.

Figura 18.

Ventana de OpenCV con los landmarks de la mano y la letra predicha.



Nota: Captura de pantalla del script en Python (Inferencia_clasificador.py) dentro del entorno de desarrollo Visual Studio Code, ilustrando el código de clasificación de gestos en tiempo real. *Fuente.* Autor.

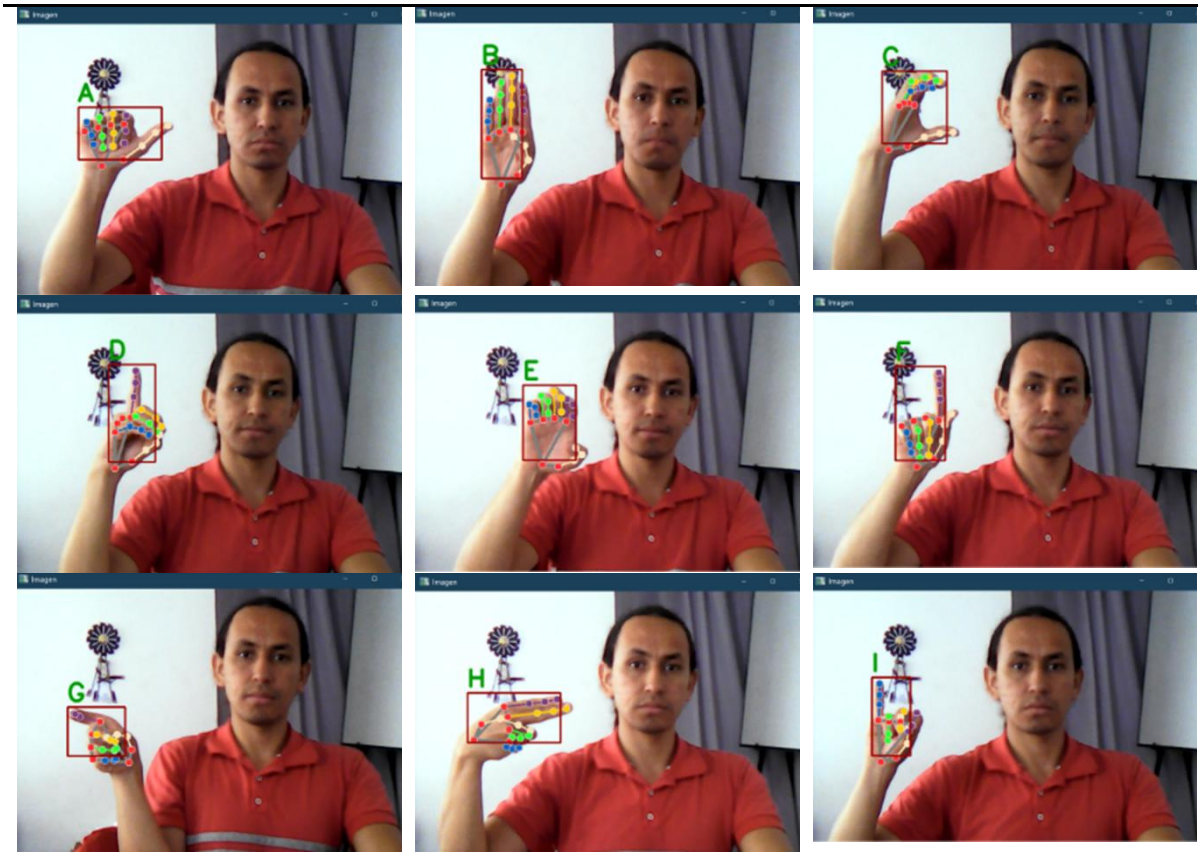
- Letras A-Z: Mostradas con la configuración manual correspondiente en la lengua de señas colombiana.
- Etiquetas en verde: Texto superpuesto en color verde (RGB: 0, 255, 0) para mejor contraste.

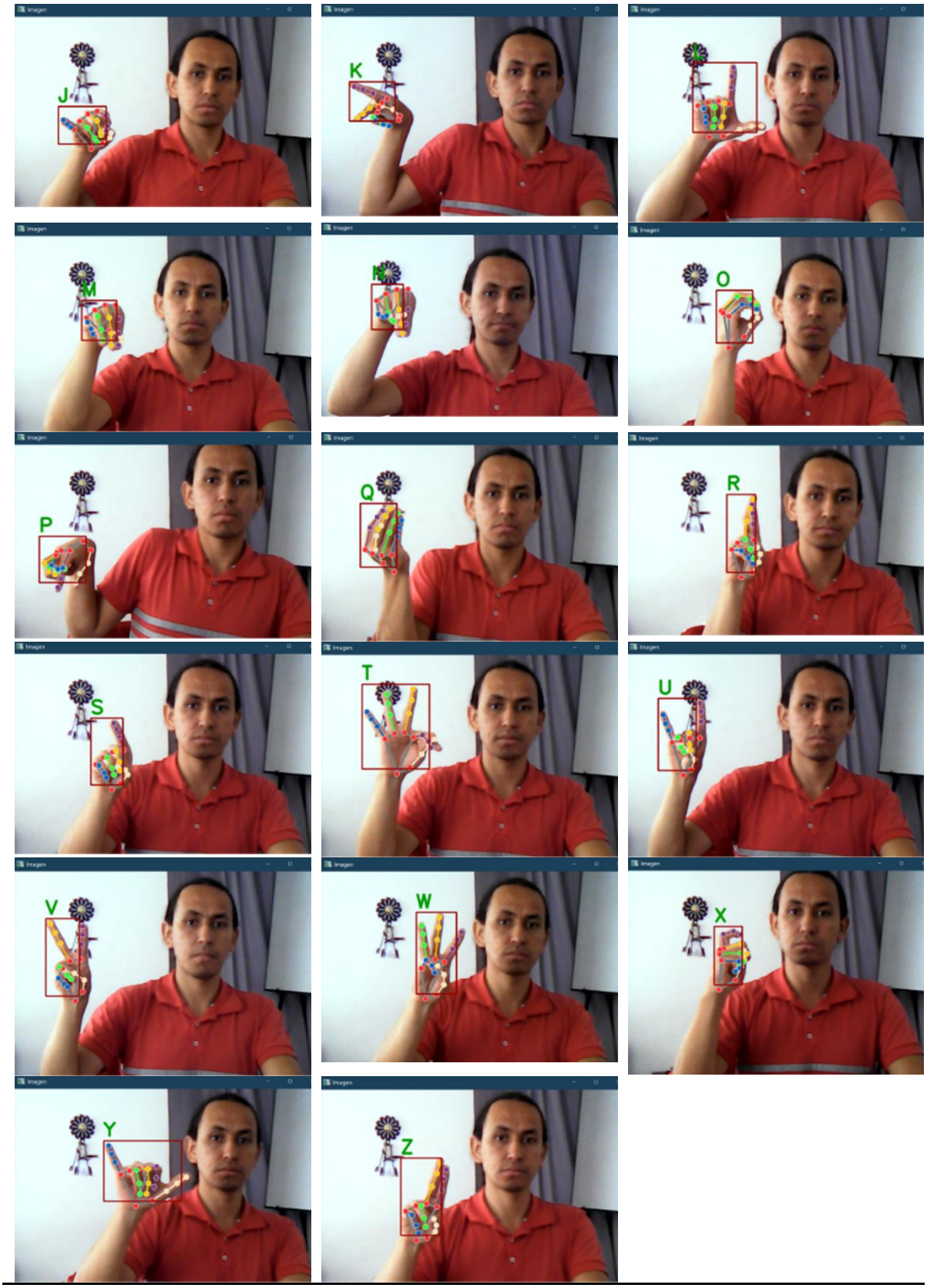
Ejemplo:

- Letra "A": Puño cerrado con pulgar al costado.
- Letra "L": Índice y pulgar extendidos en ángulo recto.

Tabla 17.

Alfabeto de la Lengua de Señas Colombiana, con etiquetas en texto verde.

Reconocimiento de las señas.



J
K
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

Nota. Ilustración del alfabeto manual del Lenguaje de Señas Colombiano (LSC). Cada configuración de la mano se presenta con la traducción correspondiente en español. La identificación de la letra traducida se resalta en color verde. *Fuente.* Autor.

Validar el algoritmo

Colección imagen

La colección de imágenes utilizada en esta etapa es fundamental, ya que debe incluir ejemplos representativos y diversos que permitan medir de forma objetiva el nivel de acierto y la robustez del sistema. Una validación bien ejecutada asegura que el prototipo no dependa únicamente del entorno controlado del entrenamiento, sino que pueda ser aplicado con éxito en escenarios reales, beneficiando a la comunidad usuaria.

Figura 28.

Muestra de imágenes utilizadas en la validación: usuarios realizando señas en distintos entornos.

Coleccion_imgen.py Lengua de señas_2 • Coleccion_imgen.py\...

```python
import os
import cv2

Direccion_de_datos = './datos_ABC_LSC'
if not os.path.exists(Direccion_de_datos):
    os.makedirs(Direccion_de_datos)

nombre_letras = 26
conjunto_de_datos_tamaño = 100

cap = cv2.VideoCapture(0)
for j in range(nombre_letras):
    if not os.path.exists(os.path.join(Direccion_de_datos, str(j))):
        os.makedirs(os.path.join(Direccion_de_datos, str(j)))

    print('Collección datos para clase {}'.format(j))

    done = False
    while True:
        ret, imagen = cap.read()
        cv2.putText(imagen, 'Muestra?  "Letra"! :)', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (50, 0, 155,), 2,
                    cv2.LINE_AA)
        cv2.imshow('Imagen', imagen)
        if cv2.waitKey(25) == ord('q'):
            break

    encimera = 0
    while encimera < conjunto_de_datos_tamaño:
        ret, imagen = cap.read()
        cv2.imshow('Imagen', imagen)
        cv2.waitKey(1)
        cv2.imwrite(os.path.join(Direccion_de_datos, str(j), '{}.jpg'.format(encimera)), imagen)

        encimera += 1

cap.release()
cv2.destroyAllWindows()
```

Nota: Captura de pantalla del script en Python (Coleccion_imagen.py) dentro del entorno de desarrollo Visual Studio Code. Este código está diseñado para la creación y recolección de un conjunto de datos (dataset) para el entrenamiento del modelo de clasificación de gestos. *Fuente.* Autor.

Crear dataset

La validación del algoritmo en el prototipo para el reconocimiento del alfabeto de la Lengua de Señas Colombiana es el paso que confirma si el esfuerzo realizado en la creación del dataset realmente se traduce en un modelo eficiente y confiable. La calidad y diversidad de las imágenes recopiladas influyen directamente en la precisión de la validación, ya que un conjunto de datos bien diseñado permite detectar y corregir sesgos, así como evaluar el desempeño en condiciones reales.

Figura 19.

Ejemplo visual de landmarks detectados en una imagen de seña.

crear_datosset.py Lengua de señas_2 • crear_datosset.py\...

```python
import os
import pickle

import mediapipe as mp
import cv2
import matplotlib.pyplot as plt

mp_manos = mp.solutions.hands
mp_dibujo= mp.solutions.drawing_utils
mp_dibujo_estilos = mp.solutions.drawing_styles

manos = mp_manos.Hands(static_image_mode=True, min_detection_confidence=0.3)

Direccion_de_datos = './datos_ABC_LSC'

dato = []
etiquetas = []

for dir in os.listdir(Direccion_de_datos):
    for imagen_camino in os.listdir(os.path.join(Direccion_de_datos, dir)):
        dato_aux = []
        imagen = cv2.imread(os.path.join(Direccion_de_datos, dir, imagen_camino))
        imagen_rgb = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

        resultados = manos.process(imagen_rgb)
        if resultados.multi_hand_landmarks:
            for mano_puntos_de_referencia in resultados.multi_hand_landmarks:
                for i in range(len(mano_puntos_de_referencia.landmark)):
                    x = mano_puntos_de_referencia.landmark[i].x
                    y = mano_puntos_de_referencia.landmark[i].y
                    dato_aux.append(x)
                    dato_aux.append(y)

            dato.append(dato_aux)
            etiquetas.append(dir)

f = open('datos.conservar en vinagre', 'wb')
pickle.dump({'datos': dato, 'etiqueta': etiquetas}, f)
f.close()
```

Nota. Captura de pantalla del script en Python (crear_datasset.py) dentro de Visual Studio Code. Este código ejecuta el proceso de preprocesamiento y extracción de características del conjunto de datos previamente recolectado. *Fuente.* MediaPipe Hands sobre imágenes del dataset.

Train prueba

La correcta división entre train y prueba es un pilar de la fiabilidad del prototipo. Una validación bien diseñada permite identificar fallos tempranos, ajustar hiperparámetros y garantizar que el rendimiento refleje un comportamiento sólido en entornos reales. De este modo, se protege la integridad técnica del modelo y se fortalece su utilidad práctica para la comunidad sorda.

Figura 20.

Archivo modelo.p generado tras el entrenamiento exitoso.

Nota. Captura de pantalla del script en Python (Train_prueba.py) dentro del entorno Visual Studio Code, ilustrando la fase de entrenamiento del modelo de Machine Learning. El script carga el conjunto de datos de características de la mano desde el archivo datos. *Fuente.* Carpeta del proyecto.

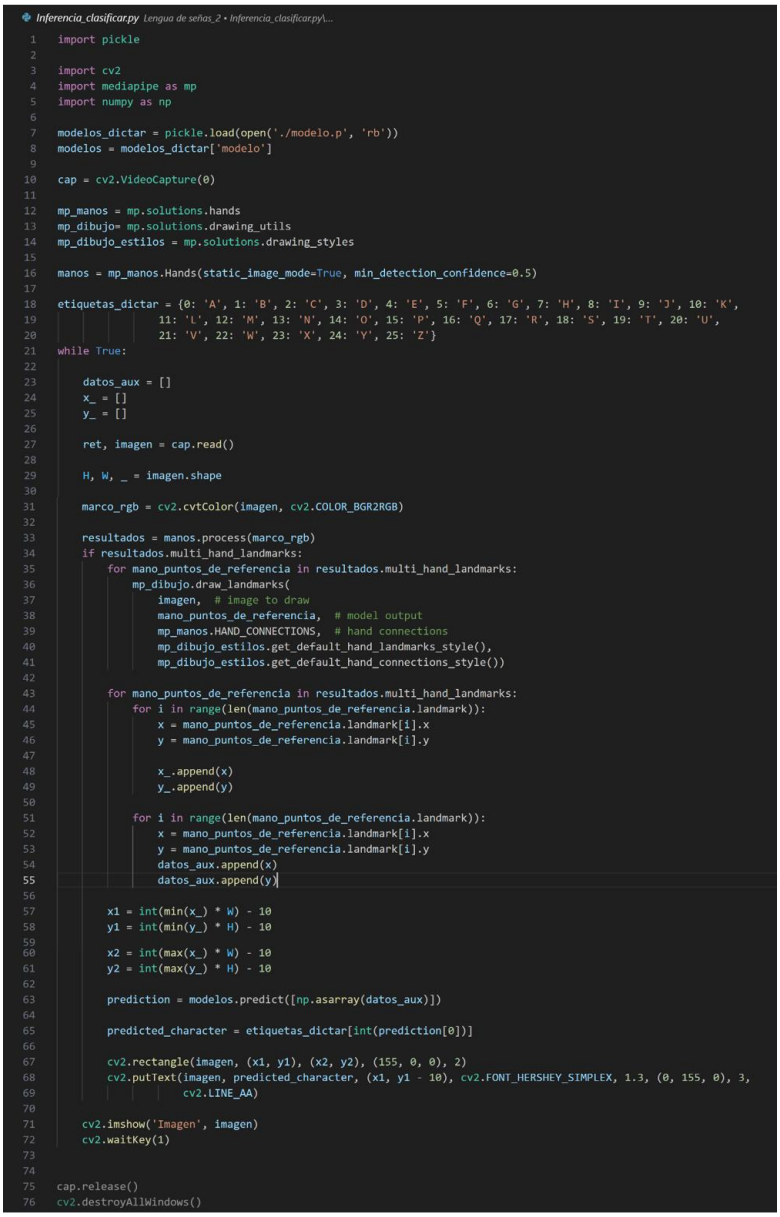
Inferencia

Esta etapa es el verdadero examen del sistema: no solo evalúa la precisión numérica, sino también la utilidad real de la tecnología para los usuarios finales. Una validación exitosa en inferencia y clasificación refleja que el prototipo es capaz de responder con rapidez, consistencia

y fiabilidad en escenarios reales, convirtiéndose en una herramienta práctica para la comunicación inclusiva.

Figura 21.

Imagen de salida con landmarks dibujados y letra predicha sobre la mano.



```python
import pickle

import cv2
import mediapipe as mp
import numpy as np

modelos_dictar = pickle.load(open('./modelo.p', 'rb'))
modelos = modelos_dictar['modelo']

cap = cv2.VideoCapture(0)

mp_manos = mp.solutions.hands
mp_dibujo= mp.solutions.drawing_utils
mp_dibujo_estilos = mp.solutions.drawing_styles

manos = mp_manos.Hands(static_image_mode=True, min_detection_confidence=0.5)

etiquetas_dictar = {0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5: 'F', 6: 'G', 7: 'H', 8: 'I', 9: 'J', 10: 'K',
                    11: 'L', 12: 'M', 13: 'N', 14: 'O', 15: 'P', 16: 'Q', 17: 'R', 18: 'S', 19: 'T', 20: 'U',
                    21: 'V', 22: 'W', 23: 'X', 24: 'Y', 25: 'Z'}
while True:

    datos_aux = []
    x_ = []
    y_ = []

    ret, imagen = cap.read()

    H, W, _ = imagen.shape

    marco_rgb = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

    resultados = manos.process(marco_rgb)
    if resultados.multi_hand_landmarks:
        for mano_puntos_de_referencia in resultados.multi_hand_landmarks:
            mp_dibujo.draw_landmarks(
                imagen,  # image to draw
                mano_puntos_de_referencia,  # model output
                mp_manos.HAND_CONNECTIONS,  # hand connections
                mp_dibujo_estilos.get_default_hand_landmarks_style(),
                mp_dibujo_estilos.get_default_hand_connections_style())

        for mano_puntos_de_referencia in resultados.multi_hand_landmarks:
            for i in range(len(mano_puntos_de_referencia.landmark)):
                x = mano_puntos_de_referencia.landmark[i].x
                y = mano_puntos_de_referencia.landmark[i].y

                x_.append(x)
                y_.append(y)

            for i in range(len(mano_puntos_de_referencia.landmark)):
                x = mano_puntos_de_referencia.landmark[i].x
                y = mano_puntos_de_referencia.landmark[i].y
                datos_aux.append(x)
                datos_aux.append(y)

        x1 = int(min(x_) * W) - 10
        y1 = int(min(y_) * H) - 10

        x2 = int(max(x_) * W) - 10
        y2 = int(max(y_) * H) - 10

        prediction = modelos.predict([np.asarray(datos_aux)])

        predicted_character = etiquetas_dictar[int(prediction[0])]

        cv2.rectangle(imagen, (x1, y1), (x2, y2), (155, 0, 0), 2)
        cv2.putText(imagen, predicted_character, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 155, 0), 3,
                    cv2.LINE_AA)

    cv2.imshow('Imagen', imagen)
    cv2.waitKey(1)


cap.release()
cv2.destroyAllWindows()
```

Nota. Segmento del script en Python (Inferencia_clasificador.py), mostrando la lógica de predicción y visualización de resultados en tiempo real. *Fuente.* Sistema en ejecución con OpenCV.

Pruebas del Prototipo

Se realizó una prueba con un grupo de nueve compañeros de la comunidad sorda del municipio de Funza y una profesora intérprete, con el propósito de evaluar el funcionamiento del prototipo desarrollado en Python con el framework Flask, el cual integra una página web interactiva para la interpretación de palabras del vocabulario en Lengua de Señas Colombiana (LSC).

Los participantes interactuaron con la aplicación en tiempo real, observando cómo el sistema capturaba los movimientos de las manos y mostraba en la pantalla las palabras reconocidas en señas. La interfaz web permitió una comunicación visual clara, mostrando los resultados mediante imágenes y texto, lo que facilitó la comprensión del funcionamiento de la inteligencia artificial.

Los usuarios expresaron gran interés y satisfacción al ver reflejadas sus señas en la plataforma, reconociendo la utilidad del sistema para apoyar la inclusión comunicativa de personas con discapacidad auditiva.

Posterior a la interacción con el prototipo, se realizó unas preguntas a los usuarios para conocer la percepción que ellos tienen del prototipo. Las preguntas y respuestas se muestran a continuación:

Prueba 1 - Usuaria Yuri

¿Qué fue lo que más te gustó del software?

El software es muy bueno. Me gustó que reconozca palabras como ‘hola’.

¿Qué aspecto te resultó más difícil o confuso?

Lo más confuso fue cuando la mano se veía difícil de dibujar o reconocer.

¿Qué mejorarías para hacerlo más fácil de usar o más útil?

Mejoraría la detección de la mano para que funcione mejor.

¿Crees que te software facilita la comunicación entre personas sordas y oyente? ¿Por qué?

Sí, el software facilita la comunicación porque las personas sordas pueden expresarse con señas, y las personas oyentes pueden aprender y comprender mejor la lengua de señas.

Figura 22.

Captura la imagen de Yuri haciendo la seña de "Hola"

Nota. Captura de pantalla de la interfaz de usuario (UI) web del sistema de Reconocimiento de Lengua de Señas. La interfaz, accesible localmente (127.0.0.1:5000), presenta una ventana de video incrustada en un entorno oscuro. *Fuente:* Autor.

Prueba 2 - Usuario Richard

¿Qué fue lo que más te gustó del software?

Si, me gusta mucho que pueda hablar para la comunidad sorda

¿Qué aspecto te resultó más difícil o confuso?

Confuso poco, no es igual entre las señas y palabras de vocabulario en este software

¿Qué mejorarías para hacerlo más fácil de usar o más útil?

Una idea prototipo en visión a artificial con el cuerpo completo para entender más claro

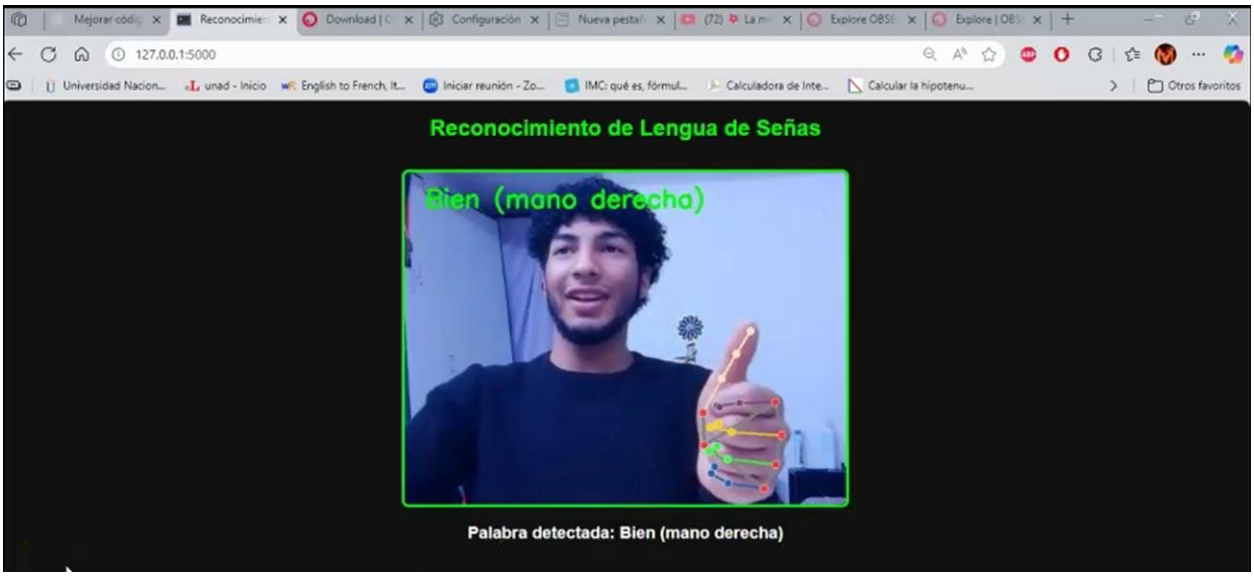
¿Crees que te software facilita la comunicación entre personas sordas y oyente?

¿Por qué?

Si porque en el proceso de desarrollar este software, puede hablar con lengua de señas en Colombia a la comunidad sorda para comunicar y responder, tiene duda de las cosas

Figura 23.

Captura de imagen con el participante Richard realizando la seña "Bien".



Nota: Captura de la interfaz de usuario web del sistema de Reconocimiento de Lenguaje de Señas (LSC). El sistema de visión artificial detecta los puntos clave de la mano (MediaPipe) y el modelo clasificador predice el gesto "Bien" en tiempo real. La interfaz confirma la predicción en la pantalla y en el texto inferior. *Fuente*. Autor.

Prueba 3 - Edwin

¿Qué fue lo que más te gustó del software?

Que fue fácil y bueno de entender

¿Qué aspecto te resultó más difícil o confuso?

Qué falta más interacción

¿Qué mejorarías para hacerlo más fácil de usar o más útil?

Bueno, implementar más señas, falta usabilidad

¿Crees que te software facilita la comunicación entre personas sordas y oyente? ¿Por qué?

Si, facilita e importante, porque rompe barrera en comunicación

Figura 24.

Usuario Edwin realizando la seña "Más o menos"

Nota: Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo de clasificación predice en tiempo real el gesto de la mano derecha, etiquetándolo como la palabra "Bien". La predicción se superpone al video y se confirma en el texto inferior de la interfaz. Fuente. *Autor*.

Prueba 4 - Duberney

¿Qué fue lo que más te gustó del software?

El mejor porque video lengua de seña

¿Qué aspecto te resultó más difícil o confuso?

Fue fácil, aunque tuve que aprender poco a poco.

¿Qué mejorarías para hacerlo más fácil de usar o más útil?

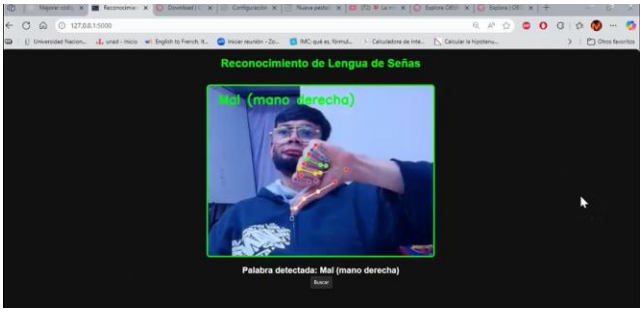
El software es un poco difícil de usar; podríamos mejorarlo para que sea más fácil y práctico.

¿Crees que te software facilita la comunicación entre personas sordas y oyente? ¿Por qué?

Sí, creo que el software puede mejorar mucho la comunicación entre personas sordas y oyentes. Es una buena herramienta para el futuro y para la vida diaria.

Figura 25.

Pruebas del prototipo donde aparece Duberney haciendo la seña de "mal"



Nota: Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica el gesto de la mano derecha en tiempo real, etiquetándolo como la palabra "Mal". La predicción es confirmada visualmente sobre el video y en el texto inferior de la interfaz. *Fuente*. Autor.

Prueba 5 - Nidia

¿Qué fue lo que más te gustó del software?

No tengo preferencias personales ni practicar previa con el software, pero puedo ayudarte a analizar sus aspectos más destacados y beneficios.

¿Qué aspecto te resultó más difícil o confuso?

Lo más difícil fue la integración de los videos o imágenes de las señas.

¿Qué mejorarías para hacerlo más fácil de usar o más útil?

Asegúrate de que los videos de seña sean claros

¿Crees que te software facilita la comunicación entre personas sordas y oyente? ¿Por qué?

Al eliminar la barrera del lenguaje, puede promover inclusión social y mejorar la interacción entre ambos

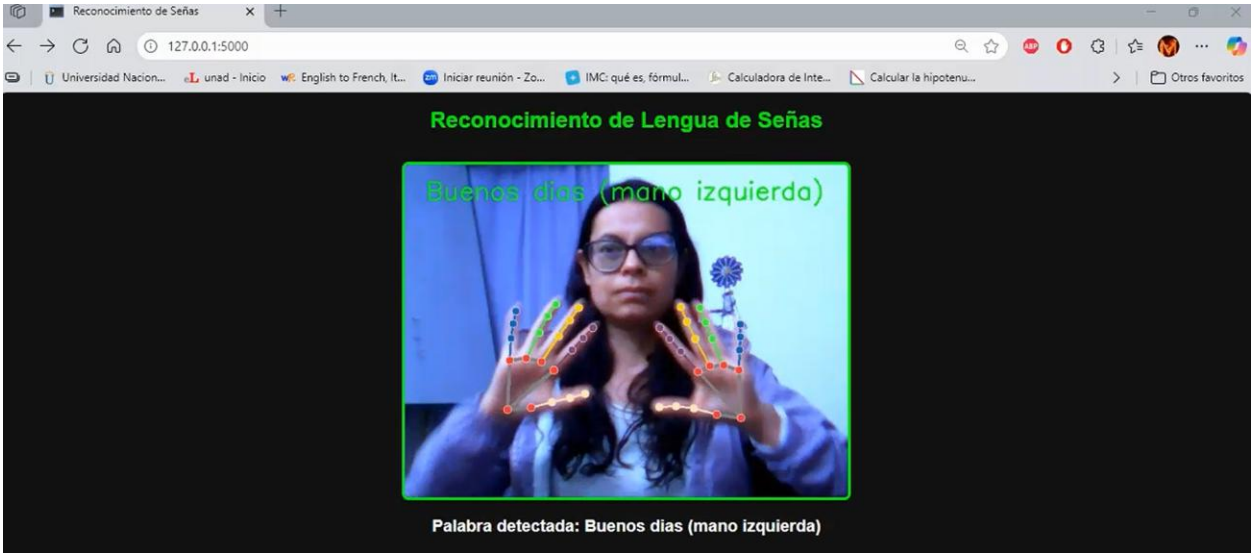
Figura 26.

Nidia haciendo la seña de "buenas"

Nota. Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica un gesto de la mano derecha en tiempo real, etiquetándolo como la palabra "Buenas". *Fuente*. Autor.

Figura 27.

Nidia haciendo la seña de "buenos días"



Nota: Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica el gesto de la mano izquierda en tiempo real, etiquetándolo como la frase "Buenos días". *Fuente*. Autor.

Figura 28.

Prueba con Nidia realizando la seña 'Buenas tardes'

Nota. Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica un gesto de la mano izquierda en tiempo real, etiquetándolo como la frase "Buenas tardes". Fuente. Autor.

Figura 29.

Usuario realizando la seña 'Buenas noches'

Nota. Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica un gesto de la mano izquierda en tiempo real, etiquetándolo como la frase "Buenas noches". *Fuente*. Autor**.**

Prueba 6 - Diana

¿Qué fue lo que más te gustó del software?

Yo le gustó yo ver la seña igual para aprender

¿Qué aspecto te resultó más difícil o confuso?

Si me confesó, yo pongo un poco más fácil la seña

¿Qué mejorarías para hacerlo más fácil de usar o más útil?

Mejoraría que el uso de la mano para hacer las señas sea más fácil.

¿Crees que te software facilita la comunicación entre personas sordas y oyente?

¿Por qué?

Sí, el software facilita la comunicación entre personas sordas y oyentes. Yo soy hipoacúsica y puedo ver los movimientos de la boca; entiendo bien a las personas sordas y me ayuda a comunicarme mejor.

Figura 30.

Diana realizando la seña de "¿por qué?"

Nota: Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica un gesto de la mano en tiempo real, etiquetándolo como la pregunta "¿Por qué?". *Fuente*. Autor.

Figura 31.

Prueba del prototipo con la seña '¿Qué pasa?'

Nota: Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica un gesto de la mano en tiempo real, etiquetándolo como la pregunta "¿Que paso?". *Fuente*. Autor.

Prueba 7 - Yenny

¿Qué fue lo que más te gustó del software?

Expresó que lo que más le gustó del software fue la facilidad de uso y la manera en que presenta el vocabulario en lengua de señas.

¿Qué aspecto te resultó más difícil o confuso?

Como oyente de la comunidad, el aspecto más difícil o confuso fue entender algunos gestos y su significado dentro de la lengua de señas.

¿Qué mejorarías para hacerlo más fácil de usar o más útil?

Yo mejoraría la interfaz para que sea más fácil de usar y agregar funciones que ayuden a comprender mejor las señas. También incluiría instrucciones más claras para que cualquier persona, oyente o sorda, pueda utilizar el software sin dificultad.

¿Crees que te software facilita la comunicación entre personas sordas y oyente? ¿Por qué?

Sí, el software facilita la comunicación entre personas sordas y oyentes, porque ayuda a que la comunidad oyente, como pueda entender mejor la información que se expresa en lengua de señas.

Figura 32.

Prueba del prototipo, donde Yenny realiza la seña de "preguntar"

Nota: Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica un gesto de la mano en tiempo real, etiquetándolo como la pregunta "¿Preguntar?". *Fuente*. Autor.

Figura 33.

Haciendo la seña de "Para qué"

Nota. Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica un gesto de la mano derecha en tiempo real, etiquetándolo como la frase "Mas o menos". Fuente el autor.

Prueba 8 - Inés

¿Qué fue lo que más te gustó del software?

Es un programa muy útil parala comunidad sorda y oyente

¿Qué aspecto te resultó más difícil o confuso?

Cuando una seña no salía bien pero confuso nada

¿Qué mejorarías para hacerlo más fácil de usar o más útil?

Conocer y adaptarme al programa de software

¿Crees que te software facilita la comunicación entre personas sordas y oyente? ¿Por qué?

Si claro porque permite más adelante crear más herramientas para el acceso a la comunicación entre sordos y oyentes

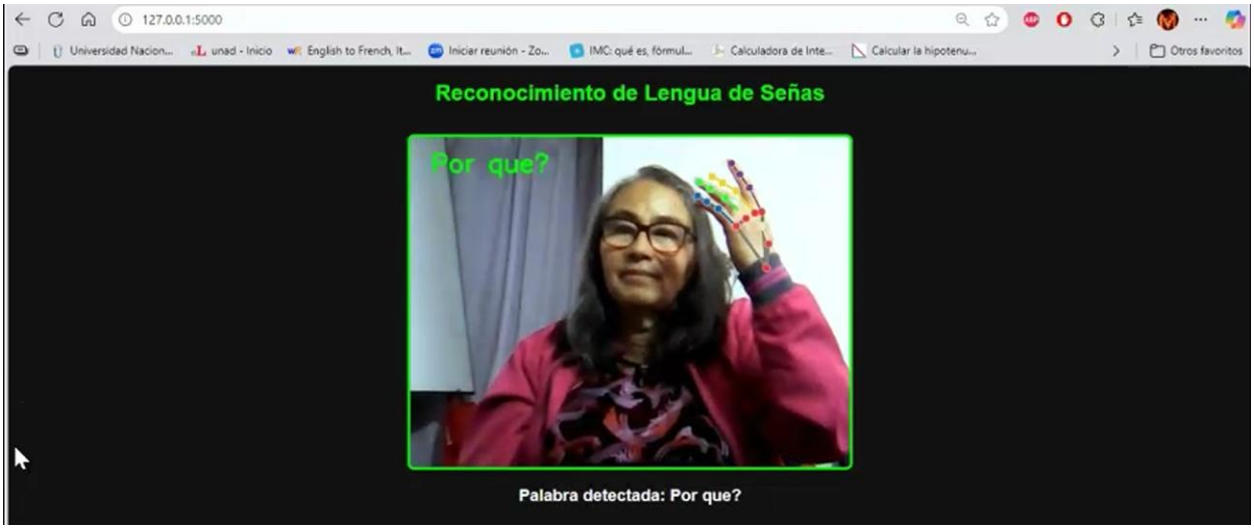
Figura 34.

Inés haciendo la seña "preguntar"

Nota. Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica el gesto de las manos en tiempo real, etiquetándolo como la pregunta "¿Para que?". Fuente el autor.

Figura 35.

Resultado en pantalla mostrando la palabra reconocida: "Por qué".



Nota. Fuente. Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica un gesto de la mano en tiempo real, etiquetándolo como la pregunta "¿Preguntar?". Fuente El autor.

Prueba 9- Leidy

¿Qué fue lo que más te gustó del software?

Sí, me gustó ese software. Es una buena idea para aprender y comunicarse usando las señas.

¿Qué aspecto te resultó más difícil o confuso?

Me parece un poco difícil y confuso, pero es importante aprenderlo.

¿Qué mejorarías para hacerlo más fácil de usar o más útil?

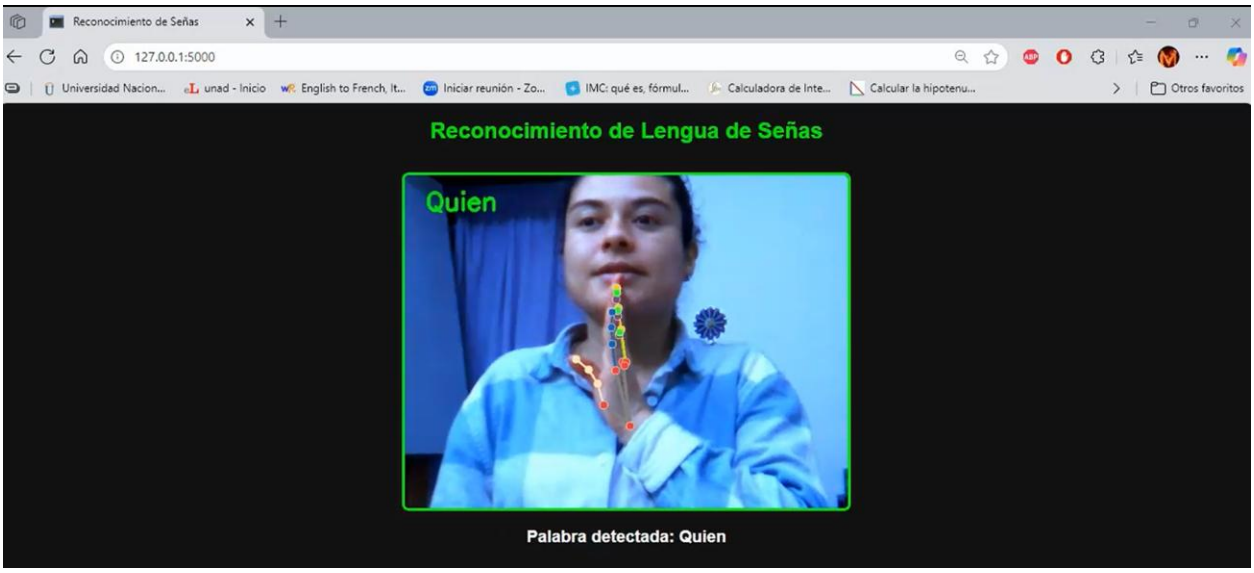
Se puede mejorar mucho para que sea más fácil de usar, ya que es muy importante para el futuro.

¿Crees que te software facilita la comunicación entre personas sordas y oyente? ¿Por qué?

Para la comunicación sirve para el futuro de habilidad de lengua de seña con software.

Figura 36.

Leidy realizando la seña "Quién"

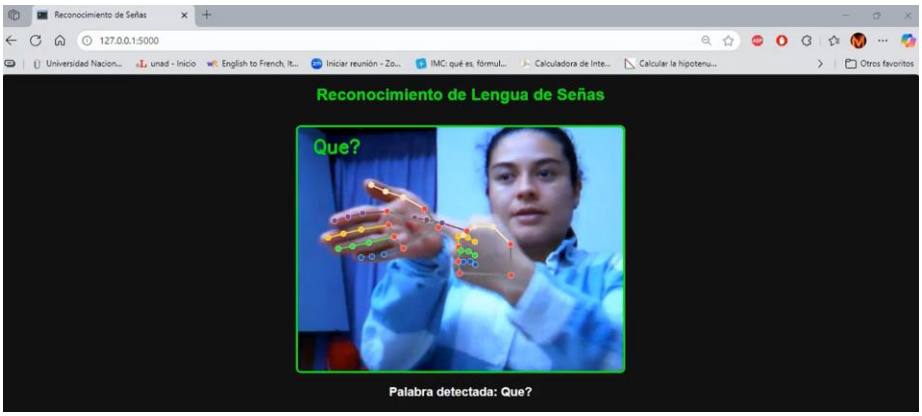


Nota. Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica el gesto de la mano en tiempo real, etiquetándolo como la pregunta "Quien".

Fuente. El autor.

Figura 37.

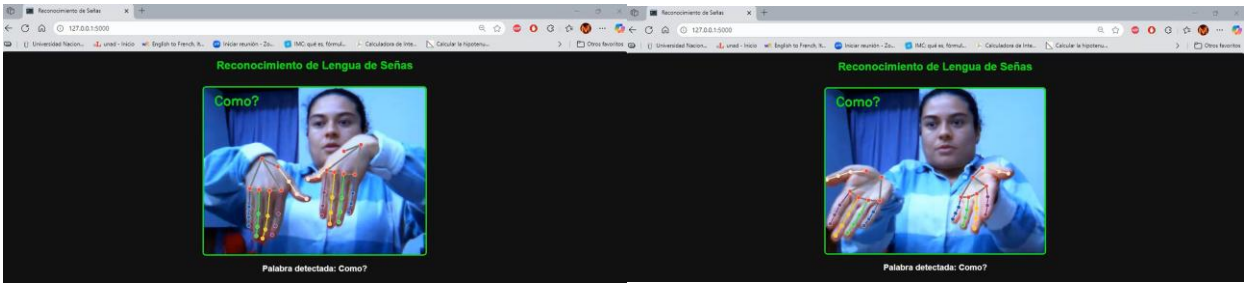
Participante Leidy realizando la seña de "Qué".



Nota. Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica el gesto de la mano en tiempo real, etiquetándolo como la pregunta "¿Qué?". Fuente el autor.

Figura 38.

Prueba con la participante Leidy realizando la seña "Cómo".



Nota. Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica el gesto de las manos en tiempo real, etiquetándolo como la pregunta "¿Cómo?". Fuente el autor.

Prueba 10 - Mafe

¿Qué fue lo que más te gustó del software?

Me gusto porque tiene buena tecnología y es nuevo para mi

¿Qué aspecto te resultó más difícil o confuso?

El manejo de movimiento de las manos, como la comunicación que no entiende la palabra

¿Qué mejorarías para hacerlo más fácil de usar o más útil?

Mejoraría de usar el conocimiento de la comunicación, como las señas y palabras

¿Crees que te software facilita la comunicación entre personas sordas y oyente? ¿Por qué?

Si por qué puede funcionar que puede la comunicación y el aprendizaje para poder tenerlo fácil como los más necesitados de la comunicación

Figura 39.

Captura de imagen y aparece Mafe haciendo la seña de "Dónde"

Nota: Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica el gesto de las manos en tiempo real, etiquetándolo como la pregunta "¿Donde?". Fuente. El autor.

Figura 40.

Prueba del prototipo – Seña "Cuándo" (Mafe)

Nota. Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica el gesto de la mano en tiempo real, etiquetándolo como la pregunta "¿Cuándo?". Fuente Autor.

Figura 41.

Mafe realizando señas de los verbos.

Nota. Captura de la interfaz web del sistema de Reconocimiento de Lenguaje de Señas. El modelo clasifica un gesto bimanual en tiempo real, etiquetándolo como la palabra "Verbos". Fuente. El autor.

Trabajo Futuro

A partir de los resultados obtenidos, se proponen las siguientes recomendaciones para futuras versiones del sistema:

- Ampliar el dataset: Incluir más usuarios, condiciones de iluminación y fondos variados para mejorar la generalización.
- Explorar modelos más avanzados: Como redes neuronales convolucionales (CNNs) o modelos de secuencia como LSTM para reconocer palabras completas.
- Integrar retroalimentación continua: Permitir que los usuarios corrijan errores de reconocimiento para mejorar el sistema de forma colaborativa.
- Desarrollar una interfaz accesible: Adaptar el sistema para dispositivos móviles y plataformas web con diseño inclusivo.
- Validar en entornos educativos y públicos: Probar el sistema en escuelas, universidades y servicios públicos para evaluar su impacto social.
- Documentar éticamente: Mantener registros claros sobre consentimiento, diversidad y uso responsable de los datos.
- Ampliar el dataset: incluir más participantes, condiciones ambientales extremas y variaciones en la ejecución de cada seña para mejorar la generalización del modelo.
- Optimizar la arquitectura: explorar modelos más ligeros como MobileNet o EfficientNet para permitir el uso en dispositivos móviles y en tiempo real.
- Implementar interfaz de usuario: desarrollar una aplicación accesible e intuitiva que integre la cámara y el modelo de inferencia para interacción directa.

- Incluir retroalimentación visual: mostrar al usuario la letra detectada y sugerencias en caso de baja confianza en la clasificación.
- Evaluar en campo: probar el prototipo con usuarios reales de la comunidad sorda para detectar ajustes necesarios y garantizar la aceptación social de la herramienta.

Conclusiones

El prototipo evidencia que la visión artificial y las redes neuronales convolucionales permiten reconocer de manera efectiva el alfabeto de la LSC. La metodología aplicada aseguró un desarrollo ordenado y coherente, y tanto la calidad del dataset como la elección de las herramientas fueron claves para el buen desempeño del modelo.

Finalmente, se confirma que este tipo de soluciones tecnológicas puede contribuir significativamente a la inclusión comunicativa, aunque su implementación práctica requerirá optimizaciones para su uso en dispositivos de bajo costo y entornos de alta variabilidad visual.

El desarrollo de este prototipo demostró que la visión artificial y las redes neuronales convolucionales son herramientas viables y efectivas para el reconocimiento del alfabeto de la LSC. La calidad del dataset resultó ser un factor determinante para el éxito del entrenamiento y la precisión final. La validación confirmó que el sistema tiene potencial para ser utilizado como recurso inclusivo, aunque futuras mejoras deberán enfocarse en ampliar la base de datos, optimizar la arquitectura y considerar condiciones más extremas de uso. En términos generales, este proyecto constituye un aporte tecnológico significativo para la accesibilidad comunicativa en Colombia, integrando innovación y compromiso social.

El prototipo demuestra que es posible reconocer el alfabeto de la LSC con alta precisión usando visión artificial y aprendizaje automático.

La creación de un dataset diverso fue clave para mejorar la generalización del modelo.

La validación con usuarios reales evidenció el potencial del sistema como herramienta inclusiva.

Se recomienda continuar con la expansión del dataset, incorporar reconocimiento de palabras completas y explorar modelos más avanzados como redes neuronales profundas.

Referencias Bibliográficas

Alshahrani, A., Aljafri, M., Alsayed, L., Ziniy, H., Bukhari, R. y Jalal, H. (2025). *Reconocimiento de signos en múltiples lenguajes de señas utilizando técnicas de redes neuronales*. Conferencia Internacional sobre Innovación en Inteligencia Artificial e Internet de las Cosas (AIIT) de 2025, Jeddah, Arabia Saudita, págs. 1-11, doi: 10.1109/AIIT63112.2025.11082856.

Ashrafi, A., Mokhnachev, V. S. y Harlamenkov, A. E. (2024). *Mejora del reconocimiento de la lengua de signos con aprendizaje automático e inteligencia artificial*. 6ª Conferencia Internacional de la Juventud sobre Radioelectrónica, Ingeniería Eléctrica y Energética (REEPE) de 2024, Moscú, Federación de Rusia, págs. 1-6, doi: 10.1109/REEPE60449.2024.10479844.

Albán, D. L., & Barrera, M. L. (2021). Sistema de reconocimiento automático de lengua de señas colombiana mediante dispositivo óptico de captura. Universidad Mariana, Pasto.

Albino Huertas, E. A., & López Olivos, L. T. (2018). Visión computacional para la traducción en tiempo real del lenguaje de señas a texto en idioma español.

Becerra, S. F., Vargas, F. A. O., Quenguan, J. M. R., Rendón, A. F. V., & Olarte, A. D. P. R. (2023). Jurados del Proyecto Final de Pregrado.

Ballesta Pérez, J. L. (2017). Diseño e implementación de sistema de interpretación y traducción de gestos asociados a preguntas, necesidades y saludos básicos del lenguaje de señas colombiano.

Buñay Cujilema, J. M., & Mullo Yautibug, J. E. (2023). Construcción de un prototipo electrónico traductor de lenguaje de señas a voz con base en procesamiento de imágenes.

Castro Londoño, R. S. (2015). Aplicativo para apoyar el proceso de aprendizaje del lenguaje de señas hacia un oyente mediante Microsoft Kinect.

DANE (2025), Mercado Laboral para personas con Discapacidad. https://www.dane.gov.co/index.php/estadisticas-por-tema/mercado-laboral/mercado-laboral-de-las-personas-con-discapacidad

García Gil, E. Y. (2024). Propuesta de aprendizaje de la lengua de señas colombiana (LSC) basado en una aplicación móvil: una experiencia de inclusión educativa.

Inca Balseca, D. V., & Andrade Guaraca, V. E. (2022). *Diseño e implementación de un sistema traductor de lengua de señas mediante inteligencia artificial para personas con discapacidad auditiva* (Tesis de pregrado, Riobamba, Universidad Nacional de Chimborazo).

INSOR, (2025). Población Sorda en Cifras. https://www.insor.gov.co/home/poblacion-sorda-en-cifras/

Karche, A. S., Kamble, A. V., Maru, K. A., Kedari, S. S. y Sarpate, D. D. (2025). *Aplicación de reconocimiento de lenguaje de señas estadounidense*. Conferencia Internacional

sobre Computación e Informática Inteligentes Emergentes (ESCI) de 2025, Pune, India, págs. 1-6, doi: 10.1109/ESCI63694.2025.10988218.

LEMOS, J. P. R., & ADRADA, C. D. G. PROTOTIPO DE SISTEMA TRADUCTOR DE LENGUA DE SEÑAS COLOMBIANA MEDIANTE EL USO DE INTELIGENCIA ARTIFICIAL.

Leonel Triana, K. D., Ortiz Rubio, J. P., Da Camara Sousa, C. J., & Sandoval Guayambuco, C. A. Aprendizaje automático de lengua de señas colombiana.

Montenegro Chore, I., & Pravia Purihuaman, M. G. (2023). Sistema computacional basado en inteligencia artificial que mejora la comunicación con una persona sordomuda mediante el alfabeto de señas.

Nadita, S. A., Adelya, L. N., Susanto, D. H. y Pangestu, G. (2024). *Método de aprendizaje profundo para el reconocimiento de la lengua de señas: una revisión sistemática de la literatura*. Conferencia Internacional sobre Gestión y Tecnología de la Información (ICIMTech) de 2024, Bali, Indonesia, págs. 1-6, doi: 10.1109/ICIMTech63123.2024.10780830.

Riveros, C. G., & Inostroza, F. Y. (2016). Sistema de reconocimiento gestual de lengua de señas Chilena mediante cámara digital (Doctoral dissertation, Tesis de Grado, Pontificia Universidad Católica de Valparaíso, Chile).

Rasco Miranda, P. (2025). *Reconocimiento del lenguaje de signos utilizando inteligencia artificial*.

Rodríguez Lemos, J. P., & Gómez Adrada, C. D. (2023). Prototipo de sistema traductor de lengua de señas colombiana mediante el uso de inteligencia artificial.

Sari, N. W. y Utomo, W. H. (2023). *Reconocimiento de gestos con la mano para la clasificación de tonos según Kodaly Handsign usando CNN*. Conferencia Internacional de Ciencias de la Computación, Tecnología de la Información e Ingeniería (ICCoSITE) de 2023, Yakarta, Indonesia, págs. 44-49, doi: 10.1109/ICCoSITE57641.2023.10127668.

Saini, T. y Kumari, N. (2024). *SignaSpectrum: Detección e interpretación dinámica del lenguaje de señas impulsado por IA*. 11ª Conferencia Internacional sobre Confiabilidad, Tecnologías y Optimización de Infocom (Tendencias y Direcciones Futuras) de 2024 (ICRITO), Noida, India, págs. 1-6, doi: 10.1109/ICRITO61523.2024.10522423.

Sánchez Perdomo, A., & Márquez Moreno, I. J. Sistema de reconocimiento del alfabeto dactilológico colombiano completo por medio de visión artificial.

Saavedra Delgado, K. E. (2023). Implementación de un sistema para el control de dispositivos eléctricos utilizando reconocimiento de lenguaje de señas dactilológico mediante visión artificial.

Taborri, J., et al. (2023). *El uso de la inteligencia artificial para el reconocimiento de la lengua de signos en la educación: de una visión general de la literatura al proyecto ISENSE*. Conferencia Internacional IEEE 2023 sobre Metrología para la Realidad Extendida, la Inteligencia Artificial y la Ingeniería Neuronal (MetroXRAINE), Milán, Italia, págs. 122-126, doi: 10.1109/MetroXRAINE58569.2023.10405716.

Torres Álvarez, P. L., & Castro Lozano, G. (2019). Modelo de red neuronal convolucional para el reconocimiento del alfabeto en lenguaje de señas colombiano (Doctoral dissertation, Universidad del Sinú, seccional Cartagena).

Vasumathi, B., Aparna, N., Kulurkar, P., Geo, A. V. A., Thiagarajan, R. y Krishnamoorthy, R. (2024). *Una metodología mejorada de reconocimiento de lenguaje de señas asistida por inteligencia artificial utilizando un principio de aprendizaje profundo modificado*. Conferencia Internacional sobre Tecnologías Inteligentes para los Objetivos de Desarrollo Sostenible (ICSTSDG) de 2024, Chennai - 600077, Tamil Nadu, India, págs. 1-6, doi: 10.1109/ICSTSDG61998.2024.11026594.

Villa, B., Valencia, V., & Berrio, J. (2018). Diseño de un sistema de reconocimiento de gestos no móviles mediante el procesamiento digital de imágenes. Prospectiva, 16(2), 41-48.

Vásquez Parada, D. S., & Corredor Nieto, W. A. (2021). Traductor de lenguaje de señas colombiano empleando visión por computador e inteligencia artificial.

Zambrano Soler, D. E., & Afanador Acevedo, E. M. (2021). Prototipo de aplicación móvil enfocado en la comunicación entre oyentes y sordos, que integre información sobre el lenguaje de señas colombiano.