

INTERPRETACIÓN DE GESTOS DINÁMICOS DE LA LENGUA DE SEÑAS UTILIZANDO TÉCNICAS DE VISIÓN ARTIFICIAL

Joshua Arango Fabbri



Universidad del Valle

Escuela de Ingeniería Eléctrica y Electrónica

Facultad de ingeniería

Programa de Ingeniería Electrónica - 3744

Mayo

2025

INTERPRETACIÓN DE GESTOS DINÁMICOS DE LA LENGUA DE SEÑAS UTILIZANDO TÉCNICAS DE VISIÓN ARTIFICIAL

Joshua Arango Fabbri

Documento presentado al
Programa de Ingeniería Electrónica - 3744
como requisito para optar por el título de
Ingeniero Electrónico

Director: Wilfredo Alfonso Morales, Ph.D

Directora: Sandra E. Nope Rodríguez, Ph.D



Universidad del Valle

Escuela de Ingeniería Eléctrica y Electrónica

Facultad de ingeniería

Programa de Ingeniería Electrónica - 3744

Mayo

2025

Aceptación

Nota de aceptación:

Firma del Jurado

Firma del Jurado

Wilfredo Alfonso Morales, Ph.D, Director

Sandra E. Nope Rodríguez, Ph.D, Directora

Cali, Mayo de 2025

Dedicatoria

Dedico este trabajo a todas las personas que, de una u otra forma, han hecho parte de este proceso. A quienes me acompañaron en los momentos de duda y también en los de alegría. A mi familia, por su amor incondicional y su confianza en mí, incluso en los días más difíciles. A mis docentes, por compartir su conocimiento con generosidad y exigirme siempre un poco más. A mis amigos y compañeros, por las conversaciones, los consejos y el apoyo que tantas veces hizo la diferencia.

Agradecimientos

Agradezco profundamente a Wilfredo Alfonso Morales y a Sandra Esperanza Nope Rodríguez, por su acompañamiento constante, su orientación académica y por compartir su experiencia y conocimiento a lo largo de este proceso. Su compromiso y confianza fueron pilares fundamentales para llevar a buen término este proyecto.

A mis amigos, compañeros y docentes de la universidad, gracias por cada conversación, enseñanza y experiencia compartida. Todos dejaron una huella en mi vida académica y personal que siempre llevaré conmigo.

Finalmente, agradezco a mi familia, por su amor, paciencia y respaldo inquebrantable a lo largo de estos años. Este logro también es suyo.

Glosario

AKAZE *Accelerated KAZE*. 22

BRIEF *Binary Robust Independent Elementary Features*. 20, 21

BRISK *Binary Robust Invariant Scalable Keypoints*. 22, 25, 29–33, 67, 69, 70

CNN *Convolutional Neural Network*. 13, 22, 23

DL *Deep Learning*. 3, 9, 16

DNN *Deep Neural Network*. 4

FAST *Features from Accelerated Segment Test*. 3, 19, 20

FN False Negative. 44, 45

FP False Positive. 44, 45

GPU *Graphics Processing Unit*. 72

IA Inteligencia Artificial. 2, 3, 5

KAZE *Nonlinear Scale Space for Feature Detection and Description*. 22

LSC Lengua de Señas Colombiana. 12, 13, 28, 53, 58

LSTM *Long Short-Term Memory*. XI, XII, 24, 28, 40, 41, 50–53

ML *Machine Learning*. 3, 16, 23

OMS Organización Mundial de la Salud. 1

OpenCV *Open Source Computer Vision Library*. 13

ORB *Oriented FAST and Rotated BRIEF*. 3, 19–22, 25, 29–33, 67–70

PLN Procesamiento del lenguaje natural. 23

ProbSom *Probabilistic Self-Organizing Maps*. 10

ReLU *Rectified Linear Unit*. 42

RGB *Red Green Blue*. 29

RNN *Recurrent Neural Networks*. 14, 23, 24, 40

SIFT *Scale-Invariant Feature Transform*. 3, 12, 18, 19, 22

SLAM *Simultaneous Localization and Mapping*. 19

SURF *Speeded-Up Robust Feature*. 3, 18, 19, 22

SVM *Support Vector Machine*. 12

TN True Negative. 44, 45

TP True Positive. 44, 45

TPU *Tensor Processing Unit*. 72

Tabla de Contenido

	Pág.
Aceptación	I
Dedicatoria.....	II
Agradecimientos	III
Glosario	IV
Tabla de Contenido	VI
Lista de Figuras	VIII
Lista de Tablas	X
Resumen	XI
Abstract	XII
1. Introducción.....	1
1.1 Objetivos	5
1.1.1 Objetivo General	5
1.1.2 Objetivos Específicos	6
1.2 Estructura del Documento	6
2. Antecedentes y Fundamentos de la Interpretación de la Lengua de Señas	8
2.1 Lengua de señas.....	8
2.2 El reconocimiento de la lengua de señas	10
2.3 Inteligencia Artificial	14
2.3.1 Revisión y análisis comparativo de técnicas de visión artificial para la extracción de <i>landmarks</i>	18
2.3.2 Arquitecturas de Redes Neuronales para la clasificación.....	22
2.4 Síntesis	24
3. Sistema de Interpretación Inteligente de Gestos usando <i>Landmarks</i>	26
3.1 Bosquejo metodológico	27

3.1.1	Selección del Descriptor	28
3.1.2	Adecuación de frames	33
3.1.3	Extracción y preparación de características	36
3.1.4	Diseño e implementación de la red neuronal.....	40
3.1.5	Evaluación del modelo y métricas de desempeño.....	43
3.2	Análisis con MediaPipe.....	47
3.3	Síntesis	53
4.	Desarrollo Experimental y Análisis de Resultados	54
4.1	Evaluación Experimental.....	54
4.2	Aplicación en Caso Práctico	62
4.3	Resultados usando BRISK y ORB	67
5.	Conclusiones y Trabajos Futuros	71
5.1	Conclusiones Generales.....	71
5.2	Trabajos Futuros.....	72
	Referencias	73

Lista de Figuras

FIGURA	Pág.
1.1 Puntos de correspondencia con técnicas de visión artificial.....	4
2.1 Lengua de señas e inteligencia artificial	9
2.2 Captura de gesto mediante el uso de guantes.....	11
2.3 Ejemplo de la página web con modelo implementado	13
2.4 Composición de la Inteligencia Artificial	16
2.5 Identificación de características utilizando SIFT	18
2.6 Identificación de características utilizando SURF	19
2.7 Identificación de características utilizando FAST	20
2.8 Identificación de características utilizando ORB	20
2.9 Identificación de características utilizando MediaPipe	21
2.10 Ejemplo de red neuronal	23
3.1 Captura de un video dentro de la base de datos externa	27
3.2 Visualización de puntos clave detectados con BRISK	29
3.3 Visualización de puntos clave detectados con ORB	31
3.4 Visualización de puntos clave detectados con MediaPipe	32
3.5 Frame Capturado para Análisis de Puntos Clave.....	34
3.6 Frame Capturado desde otro ángulo	35
3.7 Frame Capturado desde un ángulo diferente.....	35
3.8 Ejemplo de Mediapipe Pose	38
3.9 Normalización de <i>landmarks</i> en función de los hombros	39

3.10	Ejemplo de formato one-hot encoding.....	40
3.11	Ejemplo de una matriz de confusión	44
3.12	Detección de mano con MediaPipe	47
3.13	Primer frame capturado del gesto de <i>Adiós</i>	49
3.14	Estructura interna de LSTM.....	51
3.15	Gesto realizado en base de datos externa	53
4.1	Matriz de confusión del entrenamiento.....	56
4.2	Matriz de confusión de la prueba	57
4.3	Frame Capturado del gesto de <i>Trabajo</i>	59
4.4	Frame Capturado del gesto de <i>Nombre</i>	60
4.5	Fotograma de base de datos externa	63
4.6	Matriz de prueba de la base de datos externa	64
4.7	Matriz de confusión con BRISK	68
4.8	Matriz de confusión con ORB	69

Lista de Tablas

TABLA	Pág.
2.1 Tabla de antecedentes	15
3.1 Posiciones de los puntos clave del gesto Adiós en el <i>Frame 1</i>	48
4.1 Métricas del modelo obtenido.....	61

Resumen

Este trabajo propone el desarrollo de un sistema de reconocimiento de gestos dinámicos de la lengua de señas colombiana mediante el uso de redes neuronales artificiales combinando técnicas de visión por computador con modelos de aprendizaje profundo. En particular, se implementó una arquitectura basada en redes *Long Short-Term Memory* (LSTM), la cual permite modelar de forma eficiente las secuencias temporales características de los gestos dinámicos. Para ello, se utilizaron técnicas de extracción de características a partir de *landmarks* generados mediante herramientas de visión artificial, lo que permitió capturar la posición y el movimiento de las articulaciones clave a lo largo del tiempo.

Se aplicaron también técnicas de preprocesamiento y normalización de datos con el fin de aumentar la robustez del sistema frente a variaciones entre personas, posiciones de cámara y tipos de fondo. El sistema fue evaluado utilizando una base de datos propia compuesta por videos etiquetados de distintos gestos dinámicos, logrando una exactitud del 96 %.

Palabras clave— Lengua de señas, reconocimiento de gestos, redes neuronales, visión por computador.

Abstract

This work proposes the development of a dynamic gesture recognition system for Colombian Sign Language using artificial neural networks by combining computer vision techniques and deep learning models. Specifically, we implemented an architecture based on *Long Short-Term Memory* (LSTM) networks, which efficiently models the temporal sequences of dynamic gestures. To achieve this, feature extraction techniques were applied using *landmarks* generated through computer vision tools, allowing the system to capture the position and movement of keypoints over time.

Preprocessing and data normalization techniques were also applied to increase the robustness of the system against variations among individuals, camera positions, and background. The system was evaluated using a custom database composed of labeled videos of different dynamic gestures, achieving an accuracy of 96 %.

Keywords— Sign language, gesture recognition, neural networks, computer vision.

Capítulo 1

Introducción

Las personas con discapacidad auditiva enfrentan diariamente diversas dificultades para comunicarse y acceder a la información en igualdad de condiciones respecto al resto de la población [1]. De acuerdo al informe de OMS el 4 de febrero de 2024, se estima que más del 5 % de la población mundial vive con algún grado de discapacidad auditiva total o parcial [2]. Estas barreras comunicativas no solo afectan su vida personal y social, sino que también limitan su acceso a servicios esenciales como la educación, el empleo y la atención médica. La falta de soluciones inclusivas y accesibles amplifica estas dificultades, subrayando la necesidad urgente de desarrollar tecnologías y sistemas que faciliten una comunicación efectiva y equitativa para estas personas.

Dentro de este grupo poblacional existen personas que nacen con esta condición o la adquieren tempranamente. Esto implica la pérdida al acceso de una lengua primaria, lo cual es de vital importancia para su desarrollo integral. El síndrome de privación lingüística puede tener un impacto significativo en el desarrollo cognitivo, afectando áreas críticas como el pensamiento abstracto, la resolución de problemas y la adquisición de conocimientos generales [3]. Sin una lengua accesible desde una edad temprana, los niños sordos pueden enfrentar retrasos en el desarrollo de la lengua, lo que a su vez puede influir negativamente en su rendimiento académico y su capacidad para interactuar con el mundo que les rodea.

En 2021, la encuesta nacional de calidad de vida realizada por el Departamento Nacional de Estadísticas (DANE) reveló que 459.784 personas en Colombia tienen discapacidad auditiva. Este dato representa un incremento significativo del 10 % en comparación con las cifras de 2018, cuando la población con discapacidad auditiva se estimaba en aproximadamente 300.000 personas [4]. Este aumento resalta la creciente necesidad de abordar y mejorar las condiciones de accesibilidad y comunicación para este grupo de la población, así como de implementar políticas inclusivas que faciliten

la comunicación y la accesibilidad, y desarrollar tecnologías que faciliten la integración social, educativa y laboral de las personas con esta discapacidad.

Así mismo, relacionarse con personas con discapacidad auditiva puede ser un desafío, especialmente si no se está familiarizado con la lengua de señas. Las personas sordas enfrentan numerosas dificultades al interactuar con individuos que no comprenden la lengua de señas. Esta brecha en la comunicación obstaculiza aun más las interacciones cotidianas y puede llevar a una exclusión social significativa. Las dificultades en la comunicación pueden resultar en aislamiento social y limitar su participación en actividades comunitarias y profesionales.

Una posible solución radica en la implementación de sistemas capaces de reconocer y comprender los gestos de la lengua de señas. Este tipo de sistemas podría reducir la brecha de comunicación al interpretar los gestos y comprender su significado, fomentando futuros desarrollos que mejorarían significativamente la comodidad y accesibilidad para aquellos que utilizan este medio de comunicación. No obstante, es fundamental entender primero que la lengua de señas se basa en una combinación de gestos manuales, así como de expresiones faciales y corporales, para transmitir conceptos y significados de manera efectiva, que además se dividen en gestos dinámicos y gestos estáticos. Los gestos dinámicos involucran movimientos continuos de algunas partes del cuerpo, como las manos y/o los brazos, mientras que los gestos estáticos consisten en una configuración final particular del cuerpo, mantenida en una posición específica [5].

Los movimientos y trayectorias de los gestos dinámicos pueden transmitir matices adicionales y detalles sutiles que son esenciales para la comunicación no verbal, tales como la velocidad del movimiento, la dirección y el ritmo. Estos aspectos proporcionan una capa extra de significado que enriquece la interacción y permite una comprensión más profunda y precisa del mensaje. Es por ello que reconocer y comprender la complejidad de la lengua de señas, incluyendo la importancia de los gestos dinámicos, es esencial para el desarrollo de tecnologías y métodos que reduzcan la brecha de comunicación. Los recientes avances en técnicas de procesamiento de señales e imágenes ubican a la Inteligencia Artificial (IA) como una herramienta poderosa que puede llegar a simplificar el tratamiento de información compleja. La IA abarca un campo amplio y diverso, con numerosas subdisciplinas, aplicaciones y

algoritmos denominados usualmente como algoritmos de *Machine Learning* (ML) y *Deep Learning* (DL).

En el contexto de la interpretación de gestos, un enfoque particularmente relevante es el de la percepción visual. Dentro de este dominio, la visión artificial desempeña un papel crucial al permitir que los ordenadores y sistemas interpreten y extraigan información valiosa de imágenes digitales, videos y otras entradas visuales [6]. A través del uso de algoritmos de ML, las computadoras pueden analizar imágenes y videos para extraer información significativa, identificar objetos, reconocer patrones y tomar decisiones basadas en datos visuales. Este enfoque ha revolucionado numerosas aplicaciones, desde sistemas de conducción autónoma y diagnóstico médico por imágenes hasta vigilancia y reconocimiento facial [7].

Mediante el uso de algoritmos avanzados, la visión artificial puede llevar a cabo una variedad de tareas complejas. Estas tareas incluyen la detección y clasificación de objetos, el reconocimiento facial, la interpretación de gestos y movimientos que pueden ser imperceptibles para el ojo humano. Estas innovaciones que mejoran la eficiencia operativa y la calidad de los productos y servicios, también abren nuevas oportunidades para la creación de soluciones más inclusivas y accesibles para personas con diversas discapacidades [7]. Asimismo, es factible realizar una combinación de algoritmos avanzados de IA con las técnicas de reconocimiento de patrones usadas en aplicaciones de visión artificial y procesamiento de imágenes como *Scale-Invariant Feature Transform* (SIFT), *Speeded-Up Robust Feature* (SURF), *Oriented FAST and Rotated BRIEF* (ORB) y *Features from Accelerated Segment Test* (FAST).

Dichas técnicas son capaces de generar puntos de correspondencia (*landmarks*) de objetos en imágenes, identificando los puntos más destacados y significativos con el fin de analizar diferentes instancias de algún objeto. La Figura 1.1 presenta un ejemplo de puntos de correspondencia. La buena identificación de estos puntos es esencial para tareas de detección y seguimiento de gestos, ya que permiten una comparación precisa entre diferentes poses y movimientos [8], además de ser el conjunto de características que podrían integrarse a los algoritmos avanzados en IA.

El reconocimiento de las particularidades de cada gesto a partir de videos capturados con una sola cámara, incluso en situaciones donde los gestos se superponen parcial o

totalmente, o durante conversaciones rápidas con gestos complejos.

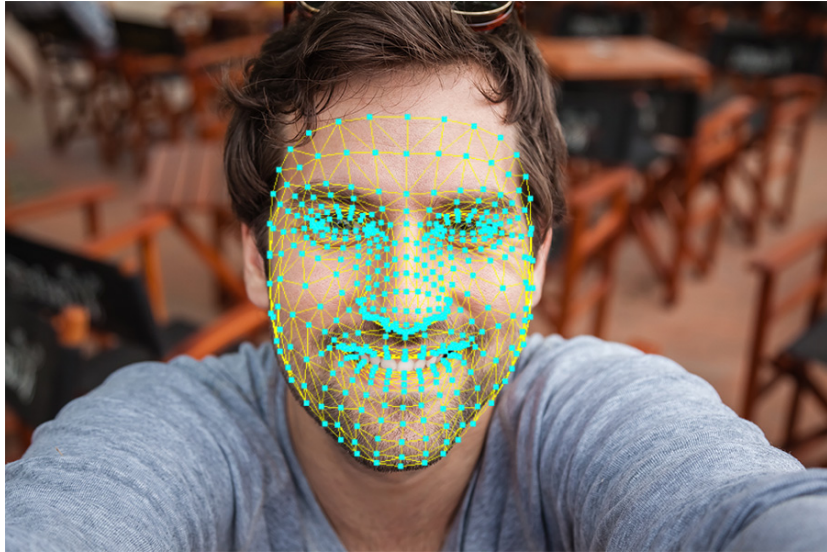


Figura 1.1: Puntos de correspondencia con técnicas de visión artificial. Tomado de [9].

Conforme a lo presentado anteriormente, este trabajo planteó la siguiente pregunta:

¿Qué método de generación de *landmarks* es el más adecuado para interpretar gestos dinámicos a partir de videos capturados por una sola cámara, que pueda sobreponerse a oclusiones temporales parciales?

En la actualidad, se han propuesto investigaciones dedicadas al reconocimiento de gestos en la lengua de señas, empleando una variedad de métodos avanzados para la detección y clasificación. Estas investigaciones abordan múltiples enfoques, desde el uso de algoritmos de aprendizaje automático y *Deep Neural Network* (DNN) hasta la implementación de sensores y cámaras especializadas, con el objetivo de encontrar el método óptimo para un funcionamiento preciso y eficiente [5, 10]. Sin embargo, un aspecto crucial de estas investigaciones es la inclusión de gestos dinámicos, que requieren un enfoque especializado. A diferencia de los gestos estáticos, los gestos dinámicos implican el análisis de una secuencia de imágenes a lo largo del tiempo para comprender completamente su significado. Este desafío adicional demanda el desarrollo de técnicas sofisticadas capaces de rastrear y reconocer estos movimientos con precisión. Implementar métodos avanzados de visión por computadora y procesamiento de video es esencial para capturar la fluidez y continuidad de los

gestos, asegurando una interpretación precisa y contextual de los gestos. Por ello, una alternativa radica en el análisis de secuencias temporales y el uso de técnicas como la segmentación de imágenes y el seguimiento de objetos, tal que se puedan descomponer los gestos en semióticas interpretables a lo largo del tiempo usando una combinación entre las técnicas de visión por computadora y la IA.

La integración de estos avances promete mejorar considerablemente la precisión y utilidad de los sistemas de reconocimiento de la lengua de señas, proporcionando una herramienta valiosa para la comunidad sorda y sus interacciones con oyentes. El reconocimiento de gestos, a través de las técnicas de visión por computadora, tiene el potencial de mejorar significativamente la interacción humana, facilitando una comunicación más natural y eficiente.

Al continuar explorando y refinando en este campo, se buscaría crear sistemas capaces de adaptarse a las variaciones individuales en el estilo de señas de cada persona, ofreciendo una solución verdaderamente inclusiva y accesible. Con el desarrollo de un sistema de interpretación de gestos, además de beneficiar a las personas con discapacidad auditiva, también podría ser aplicado en otros contextos, como las interfaces de usuario gestuales y los sistemas de realidad aumentada y virtual donde sea fundamental una interacción fluida y efectiva.

1.1. Objetivos

1.1.1. Objetivo General

Desarrollar un sistema de interpretación que emplee diversos métodos para generar *landmarks* de 10 gestos dinámicos en la lengua de señas a partir de videos capturados por una sola cámara que pueda sobreponerse a oclusiones temporales y parciales mediante técnicas de visión artificial.

1.1.2. Objetivos Específicos

- Realizar un análisis comparativo de las técnicas de visión artificial y reconocimiento de patrones empleadas en el reconocimiento de gestos dinámicos de la lengua de señas, a través de una revisión bibliográfica.
- Seleccionar tres técnicas de visión artificial que superen oclusiones temporales y parciales en el reconocimiento de gestos dinámicos de la lengua de señas.
- Implementar al menos una técnica de reconocimiento de patrones para la identificación de gestos que presenten oclusiones temporales y parciales.
- Evaluar los alcances y limitaciones del sistema de reconocimiento de gestos desarrollado, teniendo en cuenta la calidad y diversidad de los gestos dinámicos seleccionados, así como de la capacidad de sobreponerse a oclusiones.

1.2. Estructura del Documento

Este trabajo de grado se divide en 5 capítulos, cada uno abordando aspectos cruciales de la investigación sobre la interpretación de gestos dinámicos de la lengua de señas utilizando técnicas de visión artificial. En el Capítulo 1, se presenta un contexto general del tema, la formulación del problema de investigación y la pregunta problematizadora. También se delimita el alcance del estudio mediante el establecimiento de un objetivo general y los objetivos específicos.

El Capítulo 2 presenta una revisión de trabajos investigativos afines, permitiendo situar la investigación en el contexto adecuado, destacando su originalidad en el estudio. Además, se presenta el marco teórico, que define la arquitectura, las metodologías y los conceptos fundamentales. Con ello se procede a delinear una línea base, como mapa de ruta para el desarrollo del trabajo y se identifica los insumos necesarios para la construcción del sistema.

En el Capítulo 3, detalla el área de estudio en la que se enfoca la investigación, incluyendo la instrumentación necesaria, las respectivas herramientas y los algoritmos

diseñados para el reconocimiento efectivo de los gestos. Finalmente, se definen las métricas de desempeño y se realiza una síntesis de lo realizado.

El Capítulo 4 presenta el diseño experimental basándose en la metodología previamente establecida. Este capítulo ofrece un análisis de los resultados obtenidos, evaluando la fiabilidad y precisión del sistema. Se examina el desempeño del sistema en diversas condiciones, incluyendo pruebas con variaciones en los gestos. Además, se discuten los resultados obtenidos, identificando tanto los éxitos como las áreas de mejora.

Finalmente, el Capítulo 5 expone las conclusiones generales del trabajo realizado, discutiendo las implicaciones de los resultados obtenidos y proponiendo posibles direcciones futuras.

Capítulo 2

Antecedentes y Fundamentos de la Interpretación de la Lengua de Señas

Este capítulo presenta una revisión de trabajos, enfocados a la interpretación de la lengua de señas, diversidad de gestos y otros aspectos clave para evaluar el estado actual de conocimiento. Esta revisión incluye un análisis de las metodologías empleadas, los resultados obtenidos y los principales hallazgos, con el fin de identificar las áreas que requieren mayor atención. También, se realiza la definición de algunos conceptos fundamentales para lograr una comprensión más profunda en la interpretación de la lengua de señas.

Así mismo, se presentan los elementos teóricos y prácticos necesarios para la construcción de una solución al problema de investigación; esto incluye una discusión sobre las herramientas y técnicas más efectivas, así como una evaluación de las limitaciones y desafíos potenciales. Al integrar estos elementos, el estudio no solo busca contribuir al conocimiento existente, sino también proporcionar una base sólida para futuras investigaciones y aplicaciones en el campo de la interpretación de la lengua de señas y la comunicación gestual.

2.1. Lengua de señas

La lengua de señas es un sistema complejo de expresión y comunicación basado en gestos espaciales, utilizado principalmente por personas con sordera para comunicarse efectivamente con su entorno, utilizando principalmente la percepción visual. Esta forma de comunicación posee una estructura gramatical propia y un léxico específico, utilizando un espacio tridimensional donde los gestos de las manos, movimientos corporales y expresiones faciales son fundamentales para transmitir significados de

manera clara y precisa [11].

Es importante tener en cuenta que la lengua de señas, al igual que los idiomas hablados, varía según la región y la comunidad que la utiliza. Aunque algunos gestos pueden ser similares o compartidos entre distintas lenguas de señas, cada una posee su propia gramática, vocabulario y estructura única. Por esta razón, es fundamental reconocer y tratar cada lengua de señas como un sistema independiente, evitando la suposición de que existe una única lengua de señas universal.

La lengua de señas, al ser un sistema de comunicación que depende principalmente de gestos manuales, se presta de manera ideal para ser interpretada mediante técnicas de visión por computador. Estas técnicas permiten capturar, procesar y analizar los movimientos espaciales y temporales de las personas que utilizan la lengua de señas, facilitando la automatización de su interpretación. En dicho contexto, la extracción de *landmarks* se ha convertido en una herramienta fundamental, como se muestra en la Figura 2.1.



Figura 2.1: Lengua de señas e inteligencia artificial. Tomado de [9].

Estos puntos permiten capturar con precisión las posiciones y movimientos de las manos, el rostro y otras partes del cuerpo. Combinados con técnicas de *Deep Learning* (DL), capaces de identificar patrones complejos a partir de grandes volúmenes de datos no estructurados, se logran avances significativos en la precisión y la capacidad de los sistemas para interpretar la lengua de señas de manera más eficiente. Los *landmarks*

son puntos específicos en un objeto o conjunto de datos que se utilizan como puntos de correspondencia para comparar y analizar diferentes instancias del mismo objeto. Estos *landmarks* se dividen en tres categorías principales:

- Los *landmarks* anatómicos los cuales son puntos biológicamente significativos que se asignan basándose en el conocimiento experto; estos se utilizan comúnmente en estudios médicos y biológicos para analizar estructuras anatómicas y realizar comparaciones entre diferentes sujetos.
- Los *Pseudo-landmark* son puntos contruidos internamente en un objeto utilizando algoritmos específicos. A diferencia de los *landmark* anatómicos, no tienen un significado biológico intrínseco, pero son útiles para la comparación y el análisis. Estos puntos se generan automáticamente basado en características del objeto como bordes, curvas u otros atributos visuales [8].
- Los *landmarks* matemáticos, se definen según propiedades geométricas o matemáticas del objeto. Estos puntos se determinan basándose en criterios matemáticos precisos, como centros de masa, puntos de intersección de ejes de simetría, o vértices de formas geométricas.

Estos puntos de referencia proporcionan una base fundamental para la representación y análisis de estructuras en un amplio rango de aplicaciones científicas y tecnológicas. En el ámbito de la visión por computadora, los *landmarks* facilitan tareas como la reconstrucción 3D, la alineación de imágenes y la identificación de patrones [8].

2.2. El reconocimiento de la lengua de señas

El estudio abordado se enfocó en la extracción de características de la lengua de señas argentina utilizando una variante supervisada de red neuronal conocida como *Probabilistic Self-Organizing Maps* (ProbSom). Este trabajo destaca la importancia de trabajar con una base de datos compuesta por 800 imágenes capturadas de 10 sujetos realizando 5 repeticiones de gestos cotidianos (LSA16). La base de datos incluyó 16

gestos estáticos, ejecutados en diversas posiciones y rotaciones para generar una mayor diversidad y realismo. Los sujetos vistieron prendas de color negro y se ubicaron sobre un fondo blanco con iluminación controlada [12]. Además de utilizar guantes de colores fluorescentes, tal como se muestra en la Figura 2.2. En el procesamiento de imágenes se utilizó la transformada de Radon, que permite analizar imágenes al convertir una imagen 2D en varias proyecciones desde diferentes ángulos. El estudio logró una exactitud del 90 %, capaz de resolver parcialmente el problema del solapamiento, aunque capturando gestos en un entorno ideal que simplificó el proceso de detección.



Figura 2.2: Captura de gesto mediante el uso de guantes. Tomado de [5].

Basado en el estudio anterior se propuso una etapa de segmentación para identificar cada mano, facilitando así la clasificación posterior de la configuración [12]. Los resultados de los clasificadores parciales de las imágenes, obtenidos mediante un sistema de distribución gaussiana, se utilizaron como entrada para un clasificador probabilístico integral. Además de la base de datos del estudio anterior se generaron 3200 videos de 10 personas diferentes (LSA64), que incluía 64 gestos, entre ellos gestos dinámicos, alcanzando una exactitud del 96 % [5]. Es importante señalar que durante la captura de datos se utilizaron guantes de colores y un entorno ideal. Sin embargo, al implementar el sistema en entornos cotidianos donde no se emplean guantes y las condiciones ambientales difieren, su exactitud disminuye. El autor menciona que, a pesar de la abundancia de videos en la base de datos, ésta aún no es lo suficientemente robusta para su uso en escenarios reales más diversos y complejos.

Un estudio desarrolló un sistema que puede reconocer 24 gestos dactilológicos de la lengua de señas peruana. El estudio incluyó la participación de cuatro personas, quienes realizaron la cantidad necesaria de gestos correspondientes a las 24 letras del abecedario, generando un total de 9600 imágenes. La ejecución de los gestos se capturaron con un fondo negro y aplicaron filtro Gaussiano para eliminar el ruido, el operador laplaciano para resaltar los bordes y el algoritmo Canny para detectar los bordes de la mano. Para la clasificación de los gestos, se empleó el algoritmo *Support Vector Machine* (SVM). Como resultado, se logró una exactitud promedio del 70 % [13]. Sin embargo, el sistema no abordó gestos dinámicos y se identificaron algunas inconsistencias en la clasificación de ciertas letras.

Para la clasificación de 20 gestos estáticos de la Lengua de Señas Colombiana (LSC) se empleó la técnica SIFT, que en el procesamiento de imágenes y visión por computadora para detectar y describir características locales. En este estudio, se recopilaron datos a partir de una base de datos que contenía un total de 60 fotogramas, capturados con tres patrones diferentes. Las imágenes se capturaron con una cámara digital Canon PowerShot A4000, que tiene una resolución de 16 megapíxeles. El proyecto se desarrolló utilizando Matlab GUIDE, que permite al usuario interactuar con las herramientas proporcionadas mediante una interfaz intuitiva. La exactitud del sistema desarrollado alcanzó el 77 % [14]. En la etapa de pre-procesamiento se convierte la imagen a escala de grises, para reducir el costo computacional. Dado que las imágenes ya están ajustadas en tamaño, no se requieren más modificaciones. Aunque el proyecto logró una exactitud interesante, es importante destacar que no se abordaron los gestos dinámicos.

Un modelo diseñado para reconocer la LSC, abarcando específicamente 22 señas las cuales 14 gestos son estáticos y 8 gestos dinámicos. Dicho estudio diseñó una base de datos propia, recopilada con la participación de cinco personas y que comprendió un total de 3168 imágenes. Durante la captura de los gestos, las imágenes se convirtieron a escala de grises y se alcanzó una exactitud del 68 % [15]. Adicionalmente, elaboraron una página web con el modelo realizado el cual identifica el gesto y luego proporciona una descripción en forma de texto como se observa en Figura 2.3, permitiendo que cualquier persona pueda practicar esta lengua de manera interactiva.

Se presentó un sistema para el reconocimiento del alfabeto de la LSC utilizando la



Figura 2.3: Ejemplo de la página web con modelo implementado. Tomado de [15].

metodología de visión artificial respaldada por la librería OpenCV y basada en *Convolutional Neural Network* (CNN). En primer lugar, se enfocaron únicamente en 21 letras, ignorando los gestos dinámicos. Después, tomaron fotogramas de 12 personas realizando las señas y se recopilaron imágenes extraídas de 7 videos de YouTube que mostraban el abecedario. Posteriormente, se eliminó el fondo de las imágenes mediante edición, se aplicó un filtro gaussiano y se redimensionaron las imágenes, obteniendo un total de 2364 imágenes, de las cuales se utilizaron 489 como clasificadores para optimizar el modelo de red neuronal, con el objetivo de implementarlo en una aplicación para dispositivos móviles con sistema operativo Android [16]. Aunque se encontraron algunos problemas con la detección de ciertos gestos, resultando en una exactitud del 79.2 %, el método para la toma de datos resulta interesante debido al procesamiento realizado.

El esquema propuesto se centra en la extracción de características espaciales y temporales específicas de cada gesto en la LSC. Para este proceso, se consideraron cinco gestos dinámicos, destacando la utilización de gestos similares. Se empleó MediaPipe, que permite el procesamiento en tiempo real y abarca desde la detección y seguimiento de manos y rostros hasta la segmentación de poses y clasificación de imágenes. Se clasificaron las señas, obteniendo un porcentaje de exactitud variable para cada gesto, utilizando solo 10 muestras por gesto, lo que resultó en una exactitud promedio del 54 % [17]. El trabajo resulta interesante ya que además del

reconocimiento, implementaron una aplicación celular como herramienta de traducción de la lengua de señas.

Un sistema para el reconocimiento del alfabeto de la lengua de señas mexicana tiene en cuenta un total de 21 gestos estáticos y 6 gestos dinámicos. Para llevar a cabo este procedimiento crearon un corpus de 27 gestos, teniendo en cuenta el uso de la cámara OAK-D, que incluye una cámara central para capturar la información RGB y dos cámaras laterales para medir distancias. Luego de capturar los datos, detectan de forma automática los *landmarks* de la persona que está haciendo los gestos, tomando 20 puntos para la cara, 5 para el cuerpo y 21 por cada mano. Luego, se hace uso de *Recurrent Neural Networks* (RNN), el cual logró una exactitud de 80.74 % [10]. Este sistema resulta interesante debido a su detección sin depender de elementos externos, lo cual es esencial para su implementación en situaciones cotidianas. Este sistema resulta interesante debido a su capacidad para la detección sin depender de elementos externos, lo cual es esencial para su implementación en situaciones cotidianas. Sin embargo, el uso de 3 cámaras incrementa los costos y dificulta su aplicación en dispositivos de gama media o baja.

Como puede notarse en la Tabla 2.1, la mayoría de los trabajos de reconocimiento de señas utilizan gestos estáticos. En el caso de los trabajos que hacen uso de gestos dinámicos, como por ejemplo, el estudio [17], se observa una exactitud baja en comparación con otros trabajos. Por otro lado, el trabajo [10] se destaca por alcanzar un porcentaje de exactitud más elevado, aunque usando 6 gestos dinámicos, lo más interesante es el uso de *landmarks*, que podría resultar útil para la interpretación de los gestos dinámicos.

2.3. Inteligencia Artificial

La inteligencia artificial abarca un campo amplio y diverso (ver Figura 2.4), por lo que en este contexto nos enfocaremos en la percepción visual. Dentro de este dominio, la visión artificial adquiere una relevancia significativa al capacitar a ordenadores y sistemas para interpretar y extraer información valiosa de imágenes digitales, videos y otras entradas visuales. El objetivo primordial de la visión artificial es generar

Tabla 2.1: Tabla de antecedentes

Ref.	Lengua de Señas	Gestos	Tamaño BD	Dispositivo de Captura	Características	Exactitud
[12]	Argentino	16 gestos estáticos y dinámicos	800 imágenes con 10 personas y 5 repeticiones (LSA16)	Guantes con colores fluorescentes, camisa negra y fondo blanco, realizado en diferentes posiciones y rotaciones frente a la cámara	Gira la imagen en distintos sentidos para luego aplicar la transformada de Radon para el procesamiento	90 %
[5]	Argentino	64 gestos estáticos y dinámicos	(LSA16) y otra con 3200 videos de 10 personas	Guantes con colores fluorescentes, camisa negra y fondo blanco, realizado en diferentes posiciones y rotaciones frente a la cámara	Realizado en diferentes posiciones y rotaciones frente a la cámara	96 %
[13]	Peruano	24 gestos estáticos	9600 muestras con 4 personas	Toma de datos en fondo negro	En el preprocesamiento se usó filtro Gaussiano, operador laplaciano y Canny	70 %
[14]	Colombiano	20 gestos estáticos	60 fotografías de 3 patrones diferentes	Cámara digital Canon, la cual tiene 16 megapíxeles y zoom 8x	Uso de Matlab GUIDE para trabajar en el reconocimiento de gestos, uso de imagen en escala de grises	77 %
[15]	Colombiano	22 gestos estáticos	Crea repositorio con 5 personas, 3168 imágenes de 22 gestos	Almacenado en blanco y negro	Uso de aplicación web para la traducción y escala de grises para evitar parcialmente la oclusión	68 %
[16]	Colombiano	21 gestos estáticos	12 voluntarios y 7 videos de YouTube con un total de 2364 imágenes	Celular con ruido, se elimina el fondo y se acomoda el tamaño	Aplica desenfoque gaussiano, 489 imágenes de las anteriores se usaron como clasificador	79.2 %
[17]	Colombiano	5 gestos dinámicos	Uso de librería de Google, y a la hora de probar, tomaron 10 datos por gesto	Cámara de celular convencional	Uso de Mediapipe para tomar 21 puntos clave de la mano y realizan aplicación con interfaz móvil	54 %
[10]	Mexicano	21 gestos estáticos y 6 gestos dinámicos	Crearon un corpus con los 27 gestos	Uso de cámara OAK-D	Detectaron puntos característicos de la cara, cuerpo y manos, obteniendo 20, 5 y 21 puntos de interés respectivamente	80.74 %

recomendaciones fundamentadas en el análisis de la información visual. Mediante esta tecnología, las máquinas son entrenadas para realizar diversas funciones, como identificar defectos que pueden ser imperceptibles para el ojo humano [6].

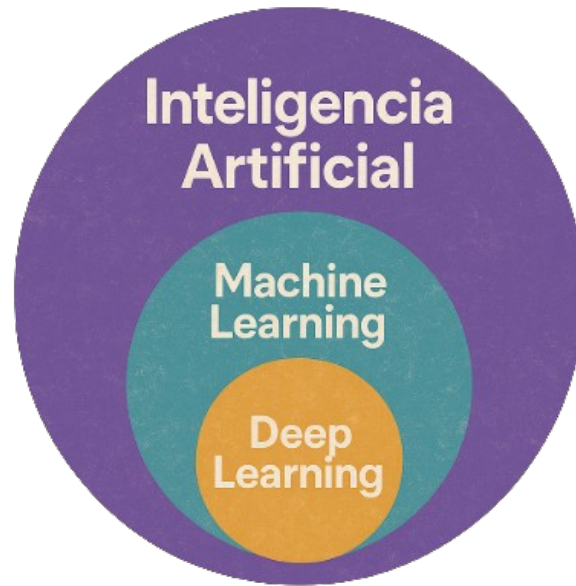


Figura 2.4: Composición de la Inteligencia Artificial. Adaptado de [7]

Machine Learning (ML): Conocido también como aprendizaje automático es una de las ramas fundamentales de la inteligencia artificial que se centra en desarrollar algoritmos y técnicas que permiten a las máquinas aprender de los datos y mejorar su rendimiento con el tiempo sin ser explícitamente programadas para cada tarea específica. Este campo abarca una amplia variedad de métodos, desde los más simples, como la regresión lineal y los árboles de decisión, hasta los más complejos, como las redes neuronales, se puede entender mejor observando la Figura 2.4 [7].

Deep Learning (DL): También conocido como aprendizaje profundo, es una subdisciplina del aprendizaje automático que utiliza algoritmos jerárquicos inspirados en la estructura y el funcionamiento del cerebro humano, conocidos como redes neuronales artificiales. Estas redes están compuestas por múltiples capas de nodos, donde cada capa procesa las entradas recibidas y transmite la salida a la siguiente capa. A través de este proceso, las redes neuronales profundas pueden aprender representaciones de datos de múltiples niveles de abstracción. Este enfoque permite a las máquinas adquirir conocimientos complejos y realizar tareas como el

reconocimiento de voz, la traducción automática, la detección de objetos en imágenes y muchas otras aplicaciones avanzadas. El aprendizaje profundo ha revolucionado el campo de la inteligencia artificial gracias a su capacidad para manejar grandes volúmenes de datos y aprender características relevantes de manera automática, lo que ha llevado a avances significativos en diversas áreas, incluyendo la visión por computadora [7].

Flujo óptico: Se refiere al desplazamiento de objetos entre fotogramas consecutivos en un vídeo. Determinar este movimiento en el tiempo es crucial para diversas aplicaciones, como el seguimiento de objetos y la estimación de la velocidad. El flujo óptico tiene dos variantes notables, el *flujo óptico disperso* el cual proporciona vectores de flujo para características específicas dentro del cuadro, como bordes o esquinas. Es útil para rastrear puntos de interés específicos en la escena y es menos intensivo computacionalmente que su contraparte densa. Por otra parte se encuentra el *flujo óptico denso*, este método abarca todos los píxeles del marco, ofreciendo un vector de flujo para cada píxel. Aunque es más intensivo en términos de cálculo, proporciona una representación completa del movimiento en la escena, lo cual es ventajoso para aplicaciones que requieren una comprensión detallada del entorno visual. Ambas técnicas de flujo óptico son fundamentales para diversas aplicaciones de la visión artificial, como la navegación autónoma, la realidad aumentada y la vigilancia, donde la precisión en la detección y seguimiento del movimiento es esencial [18].

Visión Artificial: La visión artificial es una disciplina que engloba todos los procesos y elementos que dotan a una máquina con la capacidad de ver. Se podría definir como la capacidad de una máquina para comprender imágenes mediante la deducción automática de la estructura y propiedades de un entorno tridimensional, posiblemente dinámico, a partir de una o varias imágenes bidimensionales de ese entorno. Las estructuras y propiedades del mundo tridimensional que la visión artificial intenta deducir incluyen tanto propiedades geométricas como materiales. Ejemplos de propiedades geométricas son la forma, el tamaño y la localización de los objetos, mientras que ejemplos de propiedades materiales incluyen el color, la iluminación, la textura y la composición. Si el entorno cambia durante el proceso de formación de la imagen, es necesario inferir la naturaleza de ese cambio e incluso predecir futuros eventos. En la visión artificial, el ojo del ordenador es la cámara de

video, y su retina es un sensor sensible a la intensidad del espectro. El desafío principal en la visión artificial es, por tanto, interpretar las imágenes, identificar y distinguir los objetos en la escena, extraer información relevante de ellos y resolver problemas específicos según las necesidades [19].

2.3.1. Revisión y análisis comparativo de técnicas de visión artificial para la extracción de *landmarks*

Existen varias técnicas avanzadas capaces de construir patrones generando *landmarks*. Entre las más destacadas se encuentra el algoritmo *Scale-Invariant Feature Transform* (SIFT), un método altamente efectivo para extraer puntos característicos y distintivos de una imagen. Este algoritmo identifica características clave que son invariantes a escala, rotación y, en cierta medida, a la iluminación y el ruido. SIFT mejora la correspondencia entre dos vistas diferentes de un objeto o una escena, facilitando aplicaciones en visión artificial, reconocimiento de patrones y reconstrucción tridimensional, como se puede observar en la Figura 2.5. Su capacidad para identificar y preservar características clave lo convierte en una herramienta invaluable en estos campos [20].



Figura 2.5: Identificación de características utilizando SIFT. Tomado de [21].

El algoritmo *Speeded-Up Robust Feature* (SURF), es un método avanzado diseñado para optimizar el proceso de detección y descripción de características en imágenes.

Este algoritmo es más rápido y eficiente que SIFT, manteniendo una alta precisión. SURF logra esto combinando los aspectos más destacados de los principales detectores y descriptores existentes, utilizando aproximaciones rápidas y eficaces como la convolución de cajas integrales para la detección de características, como se puede observar en la Figura 2.6. Esto lo hace especialmente útil en aplicaciones en tiempo real y en dispositivos con limitaciones de recursos [22].



Figura 2.6: Identificación de características utilizando SURF. Tomado de [21].

El algoritmo *Features from Accelerated Segment Test* (FAST), propuesto por Edward Rosten y Tom Drummond en 2006 y mejorado en 2010, es un método de detección de esquinas diseñado para ser rápido y eficiente (como indica su nombre), aunque sacrifica robustez ante variaciones en la escala y la rotación. Es especialmente útil en aplicaciones de visión artificial en tiempo real, ya que permite identificar puntos clave en imágenes con un bajo costo computacional, como se observa en la Figura 2.7. FAST surgió como una solución a los problemas de velocidad en sistemas como *Simultaneous Localization and Mapping* (SLAM), donde los recursos computacionales son limitados y es fundamental contar con algoritmos capaces de procesar información visual en tiempo real sin comprometer el rendimiento del sistema [21].

Oriented FAST and Rotated BRIEF (ORB) es un algoritmo desarrollado en 2011 por Ethan Rublee, Vincent Rabaud, Kurt Konolige y Gary R. Bradski como una alternativa eficiente a SIFT y SURF, evitando sus altos costos computacionales y problemas de

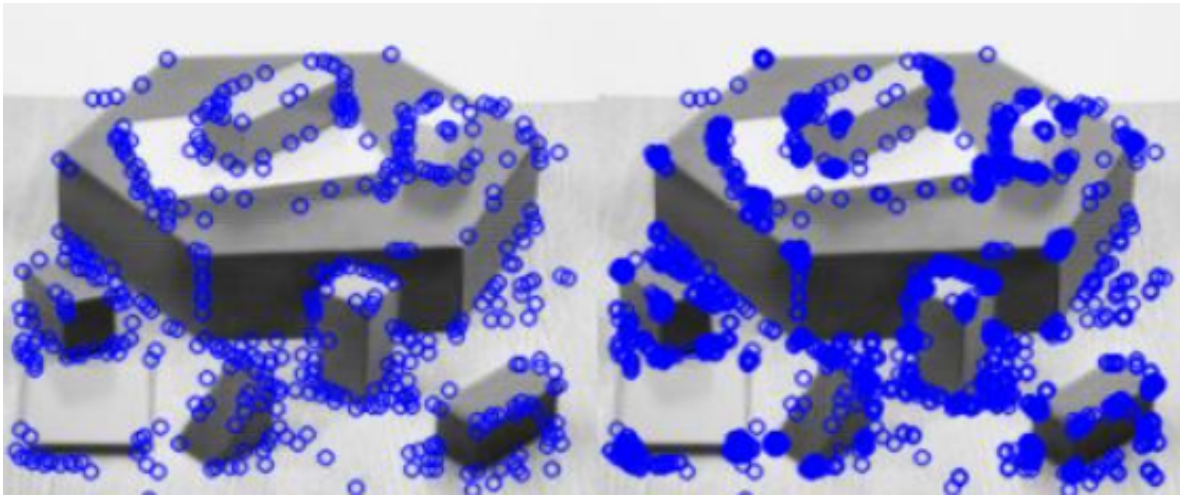


Figura 2.7: Identificación de características utilizando FAST. Tomado de [21].

patentes. Fue creado en *OpenCV Labs* y combina dos técnicas clave: FAST para la detección de puntos clave y *Binary Robust Independent Elementary Features* (BRIEF) para la descripción de características, con varias mejoras para optimizar el rendimiento. ORB primero usa FAST para identificar esquinas en la imagen y luego aplica la medida de *esquina de Harris* para seleccionar los puntos más significativos como se ilustra en la Figura 2.8.

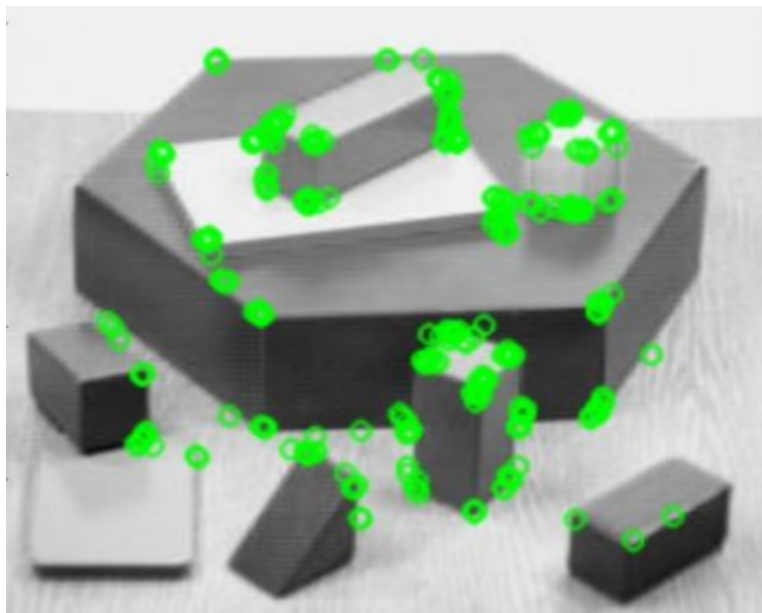


Figura 2.8: Identificación de características utilizando ORB. Tomado de [21].

Como FAST no proporciona información sobre la orientación, ORB la calcula

midiendo el centroide de intensidad en una ventana alrededor del punto detectado, donde la dirección del vector desde la esquina al centroide define la orientación. Para describir las características, ORB emplea BRIEF, pero debido a que BRIEF no es robusto a la rotación, ORB lo ajusta rotando su patrón según la orientación del punto clave.

Finalmente, se encuentra MediaPipe, un framework de código abierto desarrollado por Google para la creación y despliegue de pipelines de aprendizaje automático, especialmente en el procesamiento de datos multimedia como texto y video en tiempo real. Este framework se utiliza para desarrollar soluciones en visión por computadora, procesamiento de señales y análisis de datos. MediaPipe se basa en un modelo de programación basado en flujo de datos, donde la información fluye a través de una serie de nodos interconectados, cada uno ejecutando una tarea específica antes de transmitir los datos al siguiente nodo, como se puede observar en la Figura 2.9 [23].

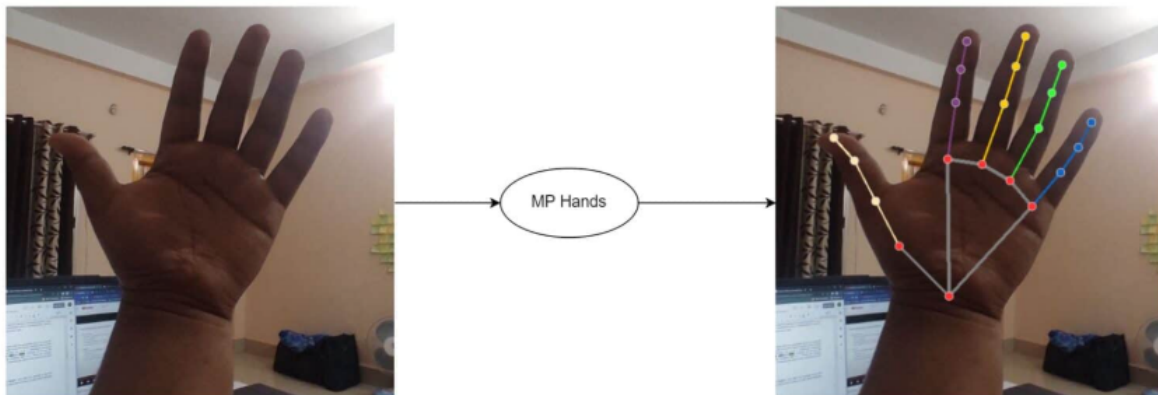


Figura 2.9: Identificación de características utilizando MediaPipe. Tomado de [24].

Es ampliamente utilizada para tareas como la detección de rostros, el seguimiento de manos, la estimación de posturas corporales, el seguimiento ocular y la detección de objetos. Su versatilidad y eficiencia le permiten funcionar incluso en dispositivos con recursos limitados, como teléfonos móviles. El funcionamiento interno de MediaPipe se basa en un sistema de gráficas de procesamiento, donde cada imagen o fotograma de un video atraviesa una serie de pasos organizados en nodos. Primero, se recibe la imagen o video como entrada. Luego, un detector localiza el objeto de interés, como una mano o un rostro. Una vez detectado, se realiza una estimación de puntos clave (*landmarks*) [25].

Un estudio ofrece un análisis detallado de los descriptores de características SIFT, SURF, KAZE, AKAZE, ORB y BRISK. Los resultados experimentales proporcionan información útil para tomar decisiones críticas en aplicaciones de visión por computadora. El estudio determinó que SIFT, SURF y BRISK son los descriptores con mayor estabilidad e invarianza a la escala, incluso cuando son significativas. ORB mostró menos invarianza a la escala y demostró ser más robusto frente a cambios en la forma en que se presenta la imagen en comparación con otros descriptores. SIFT, KAZE, AKAZE y BRISK se destacaron por su alta precisión en presencia de grandes rotaciones. Aunque ORB y BRISK son más eficientes al detectar un elevado número de características, el tiempo de emparejamiento se extiende debido a la gran cantidad de características detectadas. La precisión general de SIFT y BRISK resultó ser superior en todas las transformaciones geométricas, siendo SIFT el descriptor más preciso [26]. Adicionalmente, la evaluación cuantitativa indicó que la capacidad de los descriptores para detectar una gran cantidad de características se clasifica de la siguiente manera:

$$\text{ORB} > \text{BRISK} > \text{SURF} > \text{SIFT} > \text{AKAZE} > \text{KAZE}$$

Basado en el estudio anterior, las técnicas de detección de características eficientes en cuanto a precisión, invarianza a la escala y robustez ante cambios en los descriptores, así como en tiempo de procesamiento son los modelos BRISK, ORB y SIFT.

2.3.2. Arquitecturas de Redes Neuronales para la clasificación

Las redes neuronales son un subconjunto del aprendizaje automático y constituyen la base de los algoritmos de aprendizaje profundo. Como se muestra en la Figura 2.10, están formadas por capas de nodos, que incluyen una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo está interconectado con otros y posee un peso y un umbral asociados, lo que permite ajustar la influencia de cada conexión en el proceso de aprendizaje [27].

Existen varios tipos de redes neuronales que se utilizan para diferentes casos de uso y tipos de datos. Por ejemplo, las *Convolutional Neural Network* (CNN) se utilizan con mayor frecuencia para tareas de clasificación y visión artificial. Antes de

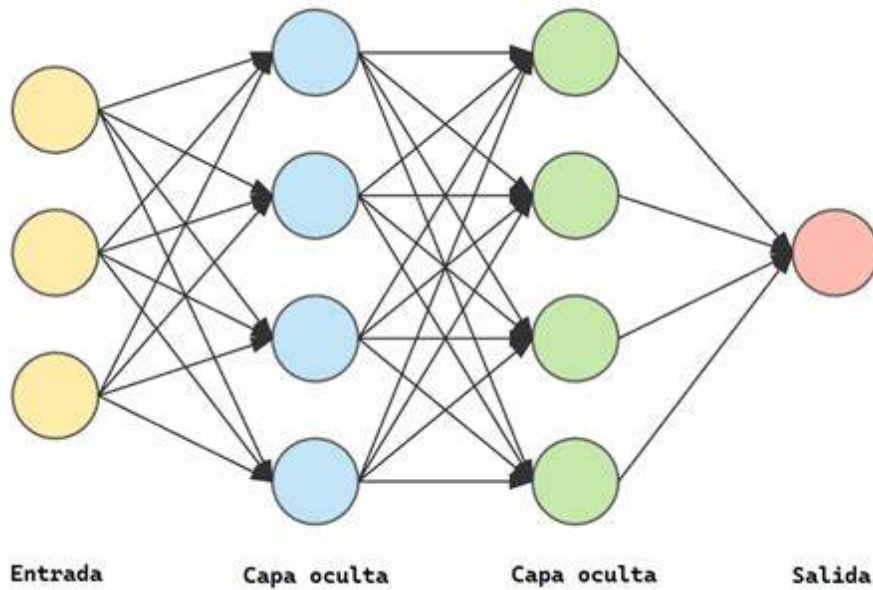


Figura 2.10: Ejemplo de red neuronal. Tomado de [28].

las CNN, se usaban métodos manuales de extracción de características con el fin de identificar objetos en las imágenes. Ahora, las CNN proporcionan un enfoque más escalable para las tareas de clasificación de imágenes y reconocimiento de objetos, aprovechando principios del álgebra lineal, específicamente la multiplicación de matrices, para identificar patrones dentro de una imagen. Sin embargo, pueden ser muy exigentes desde el punto de vista informático, ya que demandan recursos computacionales para entrenar los modelos [29].

Una *Recurrent Neural Networks* (RNN) es una red neuronal profunda que se entrena con datos secuenciales o de series temporales para crear un modelo de ML que pueda hacer predicciones o conclusiones secuenciales basándose en entradas secuenciales. Las RNN también se pueden utilizar para resolver problemas ordinales o temporales, como la traducción de idiomas, el Procesamiento del lenguaje natural (PLN), el análisis de sentimientos, el reconocimiento de voz y el subtitulado de imágenes. Las redes neuronales recurrentes utilizan los datos de entrenamiento para aprender. Se distinguen por su memoria, puesto que toman la información de las entradas anteriores para influir en la entrada y la salida actuales [27].

Las CNN están diseñadas para procesar datos con una estructura de cuadrícula,

como imágenes. Utilizan capas convolucionales que aplican filtros para extraer características espaciales y capas de agrupamiento (pooling) para reducir la dimensionalidad. Por otro lado, las RNN están diseñadas para procesar datos secuenciales, ya que poseen conexiones recurrentes que les permiten mantener información a lo largo del tiempo. Esto las hace adecuadas para tareas donde el contexto temporal es clave, como en procesamiento de lenguaje natural, reconocimiento de voz y análisis de series temporales [30].

La principal diferencia entre CNN y RNN radica en el tipo de datos que procesan. Mientras que las CNN son más adecuadas para datos espaciales como imágenes, las RNN funcionan mejor con datos secuenciales como texto o audio. Además, las CNN procesan la información de manera jerárquica y local, enfocándose en características espaciales, mientras que las RNN utilizan conexiones recurrentes para mantener información de pasos anteriores y capturar relaciones temporales [30].

Una arquitectura específica de RNN es la *Long Short-Term Memory* (LSTM), la cual es capaz de procesar tanto datos individuales como secuencias completas, como por ejemplo un video. Ayuda a evitar problemas relacionados con la dependencia a largo plazo, mientras que otras redes repiten el módulo cada vez que la entrada recibe nueva información. Sin embargo, la LSTM recordará el problema durante más tiempo y tiene una estructura similar a una cadena para repetir el módulo. El proceso de transferencia de datos es el mismo que el de las redes neuronales recurrentes estándar. Sin embargo, la operación de propagación de la información es diferente. Cuando la información pasa a través de ella, la operación decide qué información procesar más y qué información debe dejar ir. Hay varias puertas en el proceso de LSTM. Las puertas contienen varias activaciones llamadas sigmoides, que contienen algunos valores. Estos valores van de ceros a uno. Estos valores ayudan a olvidar y mantener la información [31].

2.4. Síntesis

Al analizar los antecedentes y la tabla resumen, se evidencia que la mayoría de los trabajos de reconocimiento de señas se centran en el uso de gestos estáticos. Sin embargo, algunos estudios han explorado el reconocimiento de gestos dinámicos,

enfrentando desafíos significativos en términos de exactitud. El trabajo [10] también se enfoca en gestos dinámicos, destaca por lograr un porcentaje de exactitud más elevado, con un conjunto reducido de seis gestos dinámicos. Lo más notable de este estudio es el uso de *landmarks*. Estos proporcionan una base sólida para la interpretación y reconocimiento de gestos dinámicos, ya que permiten un seguimiento detallado de las posiciones y movimientos de las manos en el espacio.

También se presentan algunos términos clave para facilitar la comprensión del trabajo, donde se hace una breve explicación de la lengua de señas, la cual es un sistema de comunicación basado en gestos espaciales, utilizado principalmente por personas sordas. Por otro lado, la visión artificial también juega un papel fundamental, permitiendo que los sistemas interpreten y extraigan información de imágenes y videos. Dentro de este campo, el flujo óptico es crucial para aplicaciones que requieren el seguimiento de movimientos.

Finalmente, los *landmarks* son puntos específicos que se emplean para comparar y analizar distintas instancias de un mismo objeto o gesto. En este trabajo se han seleccionado tres técnicas de visión artificial que han demostrado un buen desempeño frente a oclusiones temporales y parciales: BRISK, ORB y MediaPipe. Estas técnicas se distinguen por su alta precisión y su invarianza frente a cambios de escala. Su elección se fundamenta en la revisión presentada en [26], donde se analizaron diversos enfoques que comparten estas características.

Capítulo 3

Sistema de Interpretación Inteligente de Gestos usando *Landmarks*

La implementación del sistema de interpretación inteligente de gestos basado en *landmarks* requiere seguir una serie de pasos para garantizar su robustez y precisión. El primer paso es la recopilación de una base de datos con videos de gestos dinámicos, asegurando una representación suficiente de cada movimiento. Para este propósito, se han seleccionado 10 gestos distintos, cada gesto es registrado con 3 videos realizados por tres personas: dos hombres y una mujer. Todas las grabaciones se efectuaron en un entorno controlado, utilizando una pantalla verde como fondo, dentro del Laboratorio de Inteligencia Computacional de la Universidad del Valle.

Es importante destacar que los videos fueron grabados desde diferentes ángulos y posiciones, lo que permite al sistema aprender variaciones en la ejecución de los gestos. Esta variabilidad contribuye a mejorar la capacidad del modelo para reconocer los gestos en distintos escenarios y condiciones de iluminación, haciendo que el sistema final sea más robusto y generalizable.

En total, la base de datos esta compuesta por 300 videos, distribuidos equitativamente entre las distintas clases de gestos. Estos videos se dividen en dos subconjuntos: entrenamiento y prueba. Para el entrenamiento del modelo, se asignan 22 videos por gesto, mientras que los 8 restantes se reservan para la fase de prueba. Esta distribución permite que el sistema aprenda patrones distintivos en los datos de entrenamiento y evalúe su desempeño en ejemplos no utilizados durante el proceso de aprendizaje.

Asi mismo, es necesario prestar atención al preprocesamiento de los videos. Este paso es vital para garantizar que la carga de la base de datos en el sistema no sea

excesivamente pesada, lo que podría ralentizar el rendimiento del programa. El preprocesamiento permite optimizar los recursos y mejorar la velocidad de ejecución, asegurando que el sistema sea capaz de funcionar en tiempo real sin comprometer su precisión.

3.1. Bosquejo metodológico

Antes de construir el sistema de reconocimiento de gestos dinámicos, se contaba con una base de datos de la Universidad del Valle que incluye 11 gestos dinámicos: *Adiós*, *Bonito*, *Buenos Días*, *Buenas Tardes*, *Buenas Noches*, *Gracias*, *Hola*, *Nombre*, *Por Favor*, *Trabajo* y *Univalle*, cada gesto realizado una vez por tres personas diferentes, en la Figura 3.1 se observa un fotograma representativo de uno de los videos de esta base de datos. Su propósito principal era servir como referencia para la elaboración de una base de datos propia.



Figura 3.1: Captura de un video dentro de la base de datos externa.

Tomando esta base de datos como referencia se toman 10 gestos para la creación de un nuevo conjunto de datos, excluyendo el gesto *Buenas Tardes*. Estos gestos fueron escogidos debido a que la base de datos mencionada tiene los videos de los gestos que sirven como referencia. A diferencia de la base anterior, esta nueva base fue grabada con un fondo semicontrolado, lo que aporta mayor realismo a las condiciones de uso. Además, todos los gestos fueron realizados por el estudiante Joshua Arango Fabbri.

Se elaboró una base de datos para el desarrollo de un sistema de reconocimiento de gestos. Esto incluye 10 gestos de la Lengua de Señas Colombiana (LSC): *Adiós, Bonito, Buenos Días, Buenas Noches, Gracias, Hola, Nombre, Por Favor, Trabajo y Univalle*.

Esta base de datos fue desarrollada por el estudiante Joshua Arango Fabbri donde cada gesto cuenta con 30 repeticiones, para un total de 300 videos. Las grabaciones se realizaron en un entorno semicontrolado, lo que añade variabilidad a las condiciones de iluminación y fondo. Además, para los gestos que se ejecutan con una sola mano, se utilizaron tanto la mano derecha como la izquierda con el fin de aportar mayor diversidad y robustez al sistema.

Los scripts de programación se desarrollaron en Python, versión 3.11.3, utilizando el entorno de desarrollo *Visual Studio Code*. Además se cuenta con librerías como *OpenCV*, *NumPy*, *TensorFlow* y *Scikit-learn*. *OpenCV* es una herramienta poderosa para el procesamiento de imágenes y videos, así como para tareas relacionadas con la visión artificial. Esta librería permite procesar los videos al aplicar filtros y detectar objetos en los videos. *NumPy* es fundamental para realizar cálculos eficientes y manejar datos en forma de matrices y arreglos multidimensionales. En este contexto, *NumPy* procesa y manipula los datos de las imágenes, facilitando la extracción de características. *TensorFlow* es una herramienta fundamental para la construcción y entrenamiento de modelos de aprendizaje profundo. En este proyecto, *TensorFlow* se emplea para diseñar una red neuronal *Long Short-Term Memory* (LSTM), para procesar las secuencias de puntos clave extraídos de los videos. La librería *scikit-learn* es una herramienta ampliamente utilizada en el campo del aprendizaje automático para el preprocesamiento de datos, selección de características, clasificación y evaluación de modelos. En este proyecto, *scikit-learn* se emplea para convertir las etiquetas categóricas de los gestos en un formato numérico utilizando *LabelBinarizer*, lo que permitirá representar cada gesto en un formato *one-hot encoding*.

3.1.1. Selección del Descriptor

Es esencial llevar a cabo una evaluación comparativa para determinar cuál es la técnica de descriptores más adecuada para la implementación del sistema. La selección se basa en la calidad de los *landmarks*. Para ello, todos los descriptores contienen la

misma cantidad de *landmarks*, asegurando un balance óptimo entre *landmarks* relevantes y la exclusión de aquellos innecesarios, evaluando su estabilidad y robustez ante los movimientos en la ejecución de los gestos dinámicos. Se priorizará aquella técnica que ofrezca una mayor exactitud en el seguimiento del movimiento de los gestos y presente menos ruido.

Cabe destacar que los videos utilizados para comprobar la técnica de ORB y BRISK han sido convertidos a escala de grises antes del procesamiento. Esta decisión responde a la necesidad de disminuir la cantidad de datos analizados por fotograma, ya que al eliminar los tres canales de color *Red Green Blue* (RGB), se reduce significativamente la complejidad computacional, lo que se traduce en una mejora del rendimiento global del sistema.

Inicialmente, BRISK se configura con un umbral de respuesta mínimo que define la sensibilidad del detector, donde solo se consideran como *landmarks* aquellos puntos cuya respuesta supere este umbral, lo que garantiza una mayor robustez frente a detecciones poco confiables. Una vez extraídos los *landmarks* en cada fotograma, como se puede observar en la Figura 3.2, estos se ordenan en función de la confianza que el algoritmo BRISK asigna a cada uno, es decir, según su valor de respuesta.



Figura 3.2: Visualización de puntos clave detectados con BRISK.

Para seleccionar los mejores puntos, BRISK detecta candidatos a diferentes escalas y les asigna una “respuesta” basada en la diferencia de intensidad entre el píxel central y sus vecinos en un patrón circular. Los puntos con respuestas más altas, correspondientes a esquinas o bordes bien definidos, son los que se eligen [32]. Solo se conservan los puntos más significativos, priorizando aquellos con mayor relevancia para la representación del gesto. Posteriormente, estos puntos se superponen sobre los fotogramas originales para visualizar directamente el resultado del procesamiento, facilitando así el análisis visual.

Observando la imagen preprocesada, así como mediante la revisión directa en el programa, se evidencia que una porción significativa de los puntos clave se genera en la región de la mano, lo cual es deseable para el reconocimiento de gestos. También se detectan múltiples puntos distribuidos en otras zonas, en particular, se observa una alta concentración en el rostro, cabello, ojos y la camisa, lo cual indica que el algoritmo tiende a responder a texturas y contrastes marcados.

El detector *Oriented FAST and Rotated BRIEF* (ORB) fue seleccionado por su buen desempeño en entornos con variaciones de iluminación y ligeros movimientos, tal como suelen ser los gestos dinámicos. Se aplica un umbral para eliminar el ruido de baja intensidad y se realiza la sustracción de fondo, esta estrategia contribuye a reducir detecciones irrelevantes y enfoca el procesamiento en los movimientos corporales clave.

El detector ORB se configura para seleccionar los *keypoints* más significativos en función del valor de respuesta calculado por el algoritmo. Para ello, ORB detecta múltiples puntos mediante el método FAST, a los cuales les asigna una orientación que los hace invariantes a la rotación, luego evalúa su calidad con una medida tipo Harris, y finalmente selecciona los mejores puntos en función de su puntuación y su distribución espacial. Esto le permite ser un método rápido y robusto para detectar puntos clave en imágenes. como se puede observar en la Figura 3.3. Estos puntos son ordenados en función de su relevancia y proyectados sobre los fotogramas originales, lo cual facilita un análisis visual detallado de la calidad y distribución espacial de los puntos generados.

Al analizar la imagen preprocesada y corroborar en el programa, se observa que una proporción considerable de los puntos clave se concentra en la región de la mano, lo cual es favorable para la tarea de reconocimiento de gestos. Sin embargo, también se

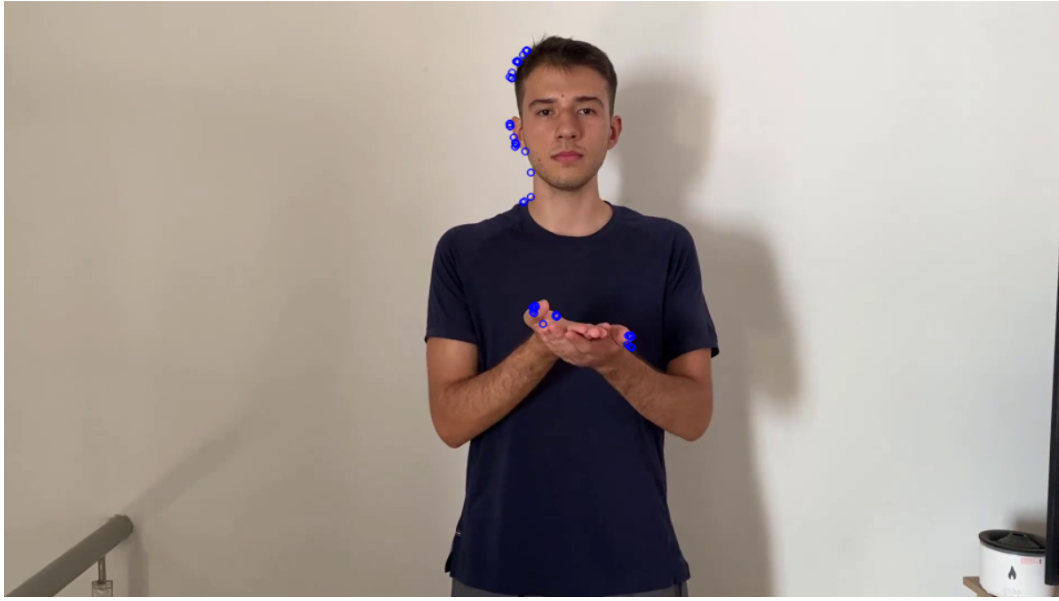


Figura 3.3: Visualización de puntos clave detectados con ORB.

identifican puntos distribuidos en otras áreas del cuerpo, especialmente en la cara y el cabello. Esto sugiere que el algoritmo responde con mayor intensidad a regiones con texturas definidas.

Finalmente, se utiliza el framework *MediaPipe Hands* como herramienta principal para la detección de puntos clave en los videos de la base de datos. Primero, detecta la región aproximada de la mano mediante un modelo de detección rápida. Luego, aplica un segundo modelo que estima 21 puntos clave anatómicos de la mano, como las articulaciones y las puntas de los dedos, a partir de la imagen recortada [33].

A diferencia de los detectores clásicos basados en texturas o bordes, como BRISK o ORB, el enfoque basado en redes neuronales de MediaPipe permite identificar *landmarks* de forma estructurada, donde la posición de los dedos es esencial para una correcta clasificación.

Previo al preprocesamiento, los videos son leídos fotograma por fotograma mediante OpenCV, preservando la resolución. El modelo de detección está configurado para identificar hasta dos manos simultáneamente, utilizando un umbral de confianza mínimo de 0.5, ya que este valor mantiene un equilibrio adecuado entre precisión y sensibilidad. Un umbral demasiado bajo puede generar mayor ruido en los resultados,

mientras que un valor demasiado alto podría impedir la detección de manos en condiciones menos favorables.

A diferencia de los detectores clásicos que generan cientos de puntos en regiones con alta textura, el modelo de MediaPipe limita su salida a un número fijo de 21 puntos por mano, como se puede observar en la Figura 3.4. Esta reducción controlada del número de puntos no solo mejora el rendimiento computacional del sistema, sino que también reduce el ruido en la representación del gesto, al evitar puntos innecesarios en zonas como el fondo, el rostro o la ropa, tal como ocurre con descriptores como ORB o BRISK.

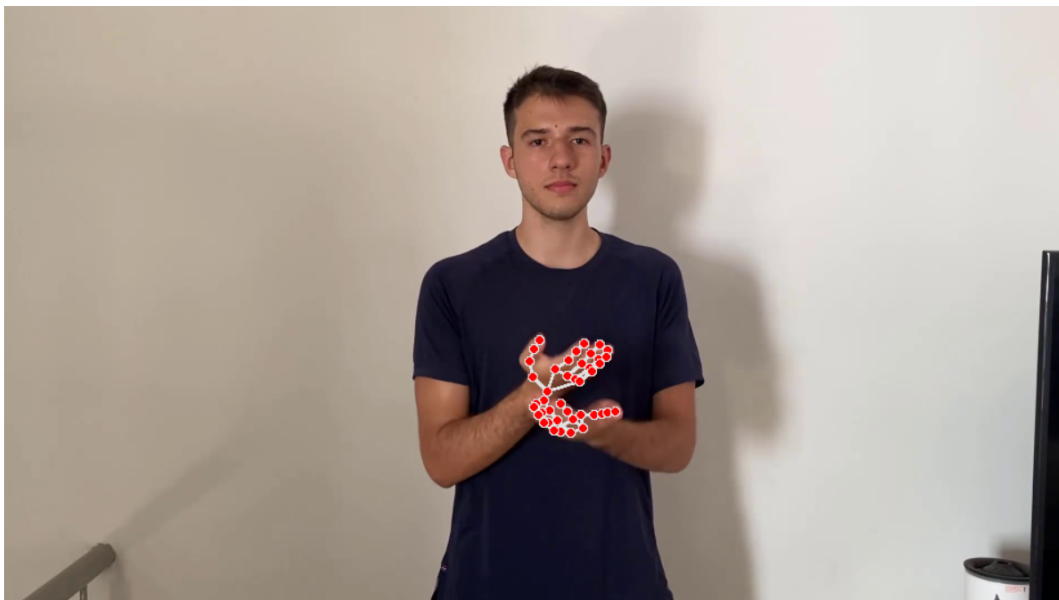


Figura 3.4: Visualización de puntos clave detectados con MediaPipe.

Los 21 puntos de referencia por mano del modelo MediaPipe Hands se distribuyen a lo largo de la mano de la siguiente manera: el punto 0 corresponde a la muñeca; los puntos del 1 al 4 representan las articulaciones del pulgar; los puntos del 5 al 8 corresponden al dedo índice; los puntos del 9 al 12 aluden al dedo medio; los puntos del 13 al 16 se encuentran en el dedo anular; y finalmente, los puntos del 17 al 20 corresponden al dedo meñique [33].

El uso de MediaPipe Hands permite una detección estructurada de puntos clave de la mano en cada fotograma de los videos procesados. Sin embargo, en algunos casos en los que la mano no es detectada, no se generan puntos clave, y los valores correspondientes a estos puntos se asignan como cero.

Al analizar visualmente los resultados obtenidos, se observa que el sistema identifica correctamente ambas manos, generando un conjunto de *landmarks* bien definidos que representan la estructura anatómica de la mano. Sin embargo, en ciertos casos, la alta velocidad del movimiento provoca que la mano no se visualice claramente en algunos fotogramas, lo que impide su detección adecuada, esta situación genera inestabilidades en el reconocimiento y afecta la continuidad en la extracción de los puntos clave.

Tras analizar ORB, BRISK y MediaPipe, se observa que MediaPipe resulta significativamente más preciso al generar los landmarks. Esto se debe a varias razones fundamentales. En primer lugar, MediaPipe permite una detección completa y estructurada de la mano, lo que facilita una representación más precisa de la forma, la posición y la orientación de los dedos, importante para diferenciar gestos similares entre sí. Además, MediaPipe no genera puntos irrelevantes en zonas como la ropa o el rostro, lo que reduce el ruido en la detección. Por otro lado, tanto ORB como BRISK son algoritmos genéricos de detección de características que se centran principalmente en esquinas y bordes de la imagen, sin un enfoque específico en anatomía humana. Debido a que ORB y BRISK no están especializados en manos, frecuentemente detectan puntos en regiones no relevantes del cuerpo, lo que introduce ruido en la detección y disminuye la exactitud del modelo. Esto representa una desventaja significativa frente a MediaPipe, que localiza con mayor exactitud las zonas de interés, permitiendo un reconocimiento más robusto y confiable de los gestos.

3.1.2. Adecuación de frames

Para iniciar el preprocesamiento, los videos de la base de datos se recortan a una duración fija de 3 segundos, lo que equivale a 90 fotogramas (frames), dado que fueron grabados a 30 fps (frames por segundo). Este recorte se realiza de forma manual mediante edición de video con el fin de que la duración de los gestos sea igual para garantizar que todos los ejemplos tengan la misma cantidad de información temporal, lo cual es fundamental para el correcto entrenamiento del modelo de reconocimiento, especialmente en arquitecturas que consideran la secuencia de frames. Al mantener constante el número de frames, se facilita el procesamiento en lotes y se evita que la variabilidad temporal afecte la consistencia de los vectores de entrada. En la Figura 3.5 se muestra un ejemplo de uno de estos fotogramas seleccionados.



Figura 3.5: Frame Capturado para Análisis de Puntos Clave.

Durante la edición, se tuvo en cuenta que algunos gestos son muy rápidos a la hora de la ejecución, por lo que, en caso de que no haya detección de la mano en ciertos cuadros, se asigna un valor de 0. Cuando se utilizan videos externos a la base de datos para probar el funcionamiento del modelo, primero se detecta la presencia de una mano mediante MediaPipe Hands. A partir de ese momento, se comienza a contar el número de fotogramas hasta alcanzar un total de 90, lo que permite capturar una secuencia completa del gesto para su posterior análisis.

El procesamiento comienza recorriendo los videos almacenados en la base de datos, los cuales están organizados en dos carpetas principales: Entrenamiento y Prueba. Dentro de cada una, hay 10 subcarpetas, donde cada subcarpeta representa una clase de gesto. Recordando que en la carpeta de entrenamiento, cada clase contiene 22 videos, mientras que en la carpeta de prueba, cada clase cuenta con 8 videos.

Es importante destacar que los videos de la base de datos fueron grabados desde diferentes ángulos, con el fin de aumentar la robustez del sistema, como se puede observar en la Figura 3.6 y Figura 3.7. Los videos fueron capturados desde cinco ángulos distintos. Inicialmente, la cámara se ubicó frente al estudiante. Posteriormente, se posicionó ligeramente por encima y a la derecha, y luego en una posición más baja. Para completar los cinco ángulos, a estos videos se les aplicó una transformación de espejo horizontal, generando así dos perspectivas adicionales que

enriquecen la variabilidad del conjunto de datos.



Figura 3.6: Frame Capturado desde otro ángulo



Figura 3.7: Frame Capturado desde un ángulo diferente

Esta variación en la perspectiva permite que el modelo generalice mejor ante distintas condiciones de visualización y sea menos sensible a los cambios de orientación, lo cual mejora su desempeño en escenarios del mundo real. No obstante, esta misma variabilidad puede ocasionar que en algunos frames no se detecten correctamente las manos, especialmente cuando se presentan posiciones menos visibles.

Aunque pueden observarse algunos elementos mínimos en el entorno, estos no interfieren de manera significativa en la detección de los gestos. Por otro lado, en los gestos que se realizan únicamente con una mano, se incluyó tanto la ejecución con la mano derecha como con la mano izquierda, con el ánimo de asegurar que el sistema sea capaz de reconocer los gestos realizados por ambas manos.

3.1.3. Extracción y preparación de características

Debido a que en cada frame se genera un vector de coordenadas, cada video contiene 90 vectores de coordenadas; esto se traduce en una gran cantidad de datos, acumulando 19800 vectores de coordenadas en el conjunto de entrenamiento y 7200 vectores de coordenadas en el conjunto de prueba. El volumen de datos implica un procesamiento que puede ser computacionalmente costoso, más si se tiene en cuenta la cantidad de puntos clave generados en cada frame, ya que esto también afectará el costo computacional.

Cada vector de coordenadas tiene 21 puntos por mano (42 puntos en total, considerando ambas manos, aunque no se haga uso de ambas), y este vector guarda la posición en X y en Y de cada punto. Entonces, el vector para un frame queda representado de la siguiente manera:

$$\text{Frame } n = [(x_0, y_0), (x_1, y_1), \dots, (x_{41}, y_{41})]$$

Este vector representa las posiciones X y Y para cada uno de los 42 puntos. Donde desde el punto (x_0, y_0) al punto (x_{20}, y_{20}) representa la mano derecha y del punto (x_{21}, y_{21}) al punto (x_{41}, y_{41}) representa la mano izquierda. Dado que un video está compuesto por múltiples frames, se tendrá una secuencia de estos vectores para cada frame del video. En este caso, el video tiene 90 frames, entonces se tendrá una matriz de datos de la siguiente manera:

$$\begin{bmatrix} \text{Frame 0:} & [X_0, Y_0, X_1, Y_1, \dots, X_{41}, Y_{41}] \\ \text{Frame 1:} & [X_0, Y_0, X_1, Y_1, \dots, X_{41}, Y_{41}] \\ \vdots & \vdots \\ \text{Frame 89:} & [X_0, Y_0, X_1, Y_1, \dots, X_{41}, Y_{41}] \end{bmatrix}$$

La matriz resultante tiene un tamaño de (90, 84), donde 90 corresponde a la cantidad de fotogramas por video y 84 es el total de valores obtenidos, producto de 42 *landmarks* por 2 coordenadas. Esta matriz representa la información extraída de un solo video. En total, la base de datos cuenta con 300 matrices de este tipo, que se utilizarán como datos de entrada para entrenar y evaluar el modelo.

Tras extraer las coordenadas, se aplica un proceso de normalización a todas las coordenadas para mejorar la estabilidad y eficiencia del sistema. Para ello, se hace uso de *MediaPipe Pose*, lo primero que hace es detectar la figura del cuerpo humano, luego, mediante una red neuronal, detecta 33 *landmarks* distribuidos en las principales articulaciones del cuerpo humano, los cuales están distribuidos para representar las principales articulaciones, como se puede observar en la Figura 3.8. Estos puntos se organizan de la siguiente manera: 11 en la cabeza, incluyendo la nariz, ojos (internos, centrales y externos), orejas (estimación del canal auditivo) y comisuras de la boca; 2 en los hombros (izquierdo y derecho); 8 en los brazos, que corresponden a codos, muñecas y bases del pulgar y el meñique de cada mano; 2 en la cadera, ubicados en los laterales de la pelvis; y 10 en las piernas, cubriendo rodillas, tobillos, talones y puntas de los pies en ambos lados.

Una vez identificados los puntos clave del cuerpo, se calcula el punto medio entre los hombros izquierdo y derecho para estimar la posición del torso. Posteriormente, todos los landmarks de las manos detectadas se restan con respecto a este punto de referencia. Esta operación traslada el sistema de coordenadas, de modo que el torso pase a ubicarse en el origen (0,0). Esta normalización por traslación permite que las coordenadas de las manos sean relativas al cuerpo, eliminando la influencia de la posición absoluta del sujeto en la imagen o video.

Además, se calcula la distancia euclidiana entre los hombros como una medida de

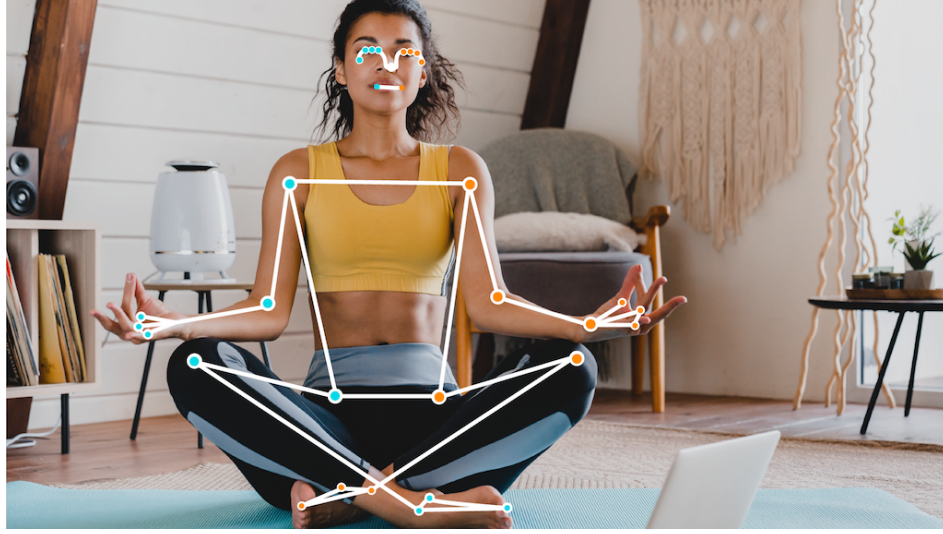


Figura 3.8: Ejemplo de Mediapipe Pose. Tomado de [25].

escala corporal, como se muestra en la Ecuación 3.1. Esta distancia se utiliza para normalizar las coordenadas, dividiendo todas ellas por dicho valor, lo que permite ajustar el tamaño relativo del cuerpo en cada cuadro. De esta forma, se reduce el impacto de variaciones debidas a la distancia del sujeto frente a la cámara o a diferencias en su contextura física, permitiendo una representación más uniforme y comparable entre distintos videos.

$$d = \sqrt{(x_{der} - x_{izq})^2 + (y_{der} - y_{izq})^2} \quad (3.1)$$

Para garantizar que los valores de las coordenadas no presenten variaciones excesivas y asegurar un entrenamiento estable del modelo, se lleva a cabo un proceso de normalización. En esta etapa, cada coordenada de los *landmarks* en un video se divide por el valor absoluto máximo de todas las coordenadas de ese mismo video, escalando así los valores al rango $[-1,1]$, esto se consigue aplicando la Ecuación 3.2 donde x representa todos los valores de una muestra. Para evitar que los valores con magnitudes más grandes dominen la optimización del modelo, que podría afectar negativamente el desempeño.

$$\tilde{x} = \frac{x}{\max(|x|)} \quad (3.2)$$

En la Figura 3.9, se ilustra cómo los hombros se utilizan como referencia para escalar y trasladar los datos. Asimismo, se muestra el punto central del torso, que se toma como origen del nuevo sistema de coordenadas.

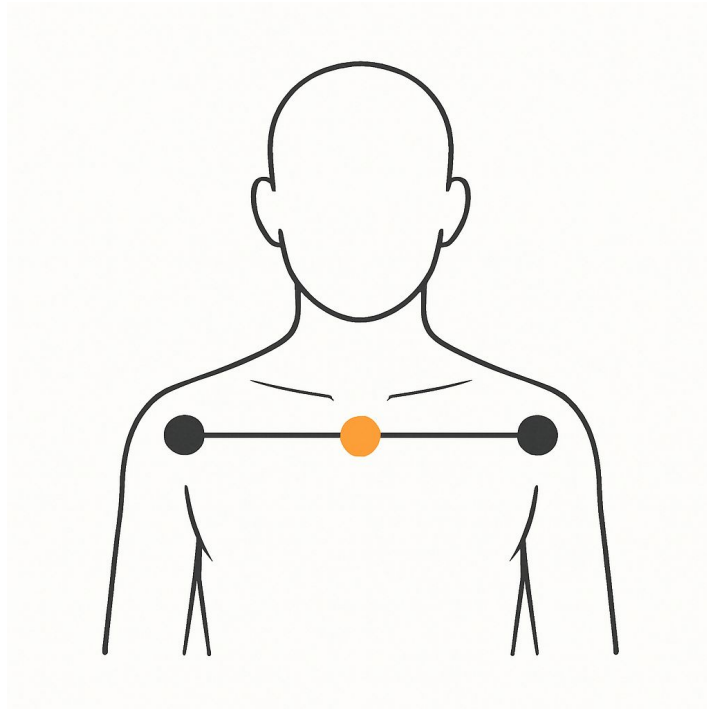


Figura 3.9: Normalización de *landmarks* en función de los hombros.

Finalmente, se convierten las etiquetas de las clases de los videos al formato *one-hot encoding*. En este proceso, cada clase se representa mediante un vector binario, en el que solo una posición tiene el valor 1 y el resto son 0, como se puede observar en la Figura 3.10. Esto facilita la interpretación de las etiquetas, ya que cada clase queda bien diferenciada en un formato numérico adecuado para el aprendizaje automático. Además, esta representación permite que el modelo utilice funciones de activación como *softmax*, lo que mejora su capacidad de asignar probabilidades a cada clase y en consecuencia, optimiza el proceso de clasificación.



Figura 3.10: Ejemplo de formato one-hot encoding. Tomado de [34].

3.1.4. Diseño e implementación de la red neuronal

Se implementa un modelo de red neuronal recurrente basado en la arquitectura LSTM utilizando TensorFlow y Keras. TensorFlow es una biblioteca de código abierto desarrollada por Google ampliamente utilizada en el aprendizaje automático para la construcción, entrenamiento y despliegue de modelos de redes neuronales (abarcando todas las arquitecturas). Keras es una interfaz de alto nivel integrada en TensorFlow que simplifica la implementación de estos modelos al proporcionar una estructura más intuitiva y fácil de usar.

A diferencia de otras bibliotecas como PyTorch, que si bien también es potente y más dinámica para investigación, requiere una mayor gestión manual de sesiones y estructuras para producción. Por otro lado, mientras que frameworks más ligeros como Scikit-learn no están diseñados para trabajar con secuencias o redes neuronales profundas. Así, TensorFlow con Keras representa una solución equilibrada entre facilidad de uso, rendimiento computacional y soporte a largo plazo, ideal para proyectos aplicados de reconocimiento de gestos mediante aprendizaje profundo [35].

Un modelo *Long Short-Term Memory* (LSTM) es un tipo de *Recurrent Neural Networks* (RNN) especializada en el procesamiento de secuencias de datos. Su

arquitectura está diseñada para retener información relevante a lo largo de intervalos temporales prolongados, lo que la hace especialmente adecuada para tareas en las que el orden y la evolución temporal de los datos son fundamentales. El modelo consta de seis capas. Es importante destacar que los datos deben conservar un orden secuencial, ya que las LSTM están concebidas para procesar secuencias temporales. Por ello, no es posible alimentar directamente los videos al modelo sin antes extraer previamente los *landmarks* u otras características relevantes.

Para estructurar la red de manera ordenada, se emplea la clase *Sequential*. La primera capa del modelo es una capa LSTM con 64 unidades, configurada para devolver secuencias completas, lo que resulta útil al apilar múltiples capas LSTM.

Para mejorar la estabilidad y velocidad del entrenamiento, la segunda capa utiliza *BatchNormalization*, que normaliza las activaciones por lote ajustando su media y varianza. Esto facilita entrenar modelos profundos de forma más eficiente y reduce el riesgo de sobreajuste. A diferencia de otras normalizaciones que actúan a nivel de muestra, *BatchNormalization* opera a nivel de mini-batch, siendo muy efectiva con lotes grandes, aunque su rendimiento puede disminuir con lotes muy pequeños, donde otras técnicas podrían ser mejores.

También se aplica *Dropout* en la tercera capa con un valor de 0.5, lo que implica que el 50 % de las conexiones de la red se desactivan aleatoriamente en cada iteración del entrenamiento. Este valor es comúnmente utilizado porque ofrece un buen equilibrio entre reducir el sobreajuste y mantener suficiente información para que el modelo aprenda. Al desactivar la mitad de las neuronas, se fuerza a la red a no depender excesivamente de neuronas específicas, promoviendo una representación más robusta y generalizable.

Luego, en la cuarta capa se incorpora una segunda capa LSTM con 64 unidades, la cual solo devuelve el último estado oculto en lugar de una secuencia completa. Esto reduce la dimensionalidad de la salida y facilita la extracción de características clave antes de pasar a las capas densas, optimizando así el rendimiento del modelo en el reconocimiento de gestos dinámicos.

Después de las capas LSTM, se utiliza una quinta capa completamente conectada

(*Dense*) con 128 neuronas y función de activación ReLU. Se escoge el número de neuronas mediante prueba y error, la capa *Dense* es un tipo de capa neuronal donde cada neurona está conectada a todas las neuronas de la capa anterior, permitiendo que la red aprenda combinaciones complejas de las características extraídas. La función de activación *Rectified Linear Unit* (ReLU) se define como $f(x) = \max(0, x)$, modifica cada valor negativo a cero, mientras que los valores positivos permanecen igual. Esto introduce no linealidad en la red, evitando problemas como la desaparición del gradiente y mejorando la capacidad del modelo para aprender representaciones complejas.

Finalmente, en la última capa se incorpora una capa de salida *Dense* con 10 neuronas, correspondientes al número de clases del problema. En esta capa se utiliza la función de activación *Softmax*, la cual convierte las salidas de la red en un conjunto de probabilidades normalizadas. Esta función garantiza que todas las salidas sean valores entre 0 y 1 y que la suma de todas las probabilidades sea igual a 1. De esta manera, *Softmax* permite interpretar la salida del modelo como la probabilidad de que la entrada pertenezca a cada una de las 10 clases, facilitando la clasificación de los gestos dinámicos.

El modelo se compila utilizando el optimizador *Adam*, una técnica avanzada de optimización que combina las ventajas de *RMSprop* y *Momentum* para lograr un buen ajuste de los pesos. *Momentum* ayuda a acelerar el entrenamiento al acumular información de gradientes previos y suavizar las actualizaciones, mientras que *RMSprop* ajusta la tasa de aprendizaje de manera adaptativa, permitiendo que el modelo converja de forma más estable y rápida. Se utiliza *Adam* porque, requiera poca configuración manual de hiperparámetros, además funciona bien en una amplia gama de problemas y datos, lo que lo convierte en una opción robusta y eficiente para tareas como el reconocimiento de gestos.

Para entrenar el modelo, se emplea la función de pérdida *Categorical Crossentropy*, la cual es opción estándar para problemas de clasificación multiclase donde las etiquetas están codificadas en formato *one-hot* (es decir, representadas como vectores en los que solo una posición tiene un valor de 1 y las demás son 0). Esta función mide la diferencia entre las distribuciones de probabilidad predichas y las reales, penalizando con mayor fuerza aquellas predicciones alejadas del valor correcto. En TensorFlow y

Keras, *Categorical Crossentropy* es la única función específicamente diseñada para este tipo de representación, por lo que su uso es fundamental cuando se trabaja con este tipo de codificación de etiquetas.

Se utiliza la métrica *accuracy* (exactitud) para evaluar el rendimiento del modelo durante el entrenamiento y la prueba. *Accuracy* mide la proporción de predicciones correctas sobre el total de muestras evaluadas, proporcionando una forma intuitiva de entender qué tan bien el modelo está clasificando los gestos dinámicos.

Para mejorar el entrenamiento y evitar el sobreajuste, se implementa *EarlyStopping*, que supervisa la función de pérdida calculada sobre los datos de validación. Si esta métrica no muestra mejoras durante 5 épocas consecutivas (*patience=5*), el entrenamiento se detiene automáticamente y se restauran los pesos correspondientes al mejor desempeño en validación (*restore best weights=True*). Esto ayuda a prevenir el sobreentrenamiento del modelo y optimiza el tiempo de procesamiento al detener el entrenamiento cuando ya no se observan mejoras significativas. El modelo se entrena con un máximo de 80 épocas y un tamaño de lote (*batch size*) de 32 muestras por iteración.

3.1.5. Evaluación del modelo y métricas de desempeño

El siguiente objetivo es evaluar un modelo de clasificación en un conjunto de prueba, utilizando la métrica de exactitud. En primer lugar, se realizan predicciones sobre los datos de prueba, obteniendo una matriz de probabilidades en la que cada fila representa una muestra y cada columna indica la probabilidad de pertenencia a una clase específica.

La matriz de confusión es una tabla que permite visualizar cómo se distribuyen las predicciones del modelo en relación con las clases reales, como se puede observar en la Figura 3.11. Cada fila representa una clase real, mientras que cada columna representa una clase predicha, lo que permite identificar errores de clasificación. Por ejemplo, una matriz con valores altos en su diagonal principal indica que el modelo clasifica correctamente la mayoría de los ejemplos, mientras que valores altos fuera de la diagonal sugieren errores en la clasificación.

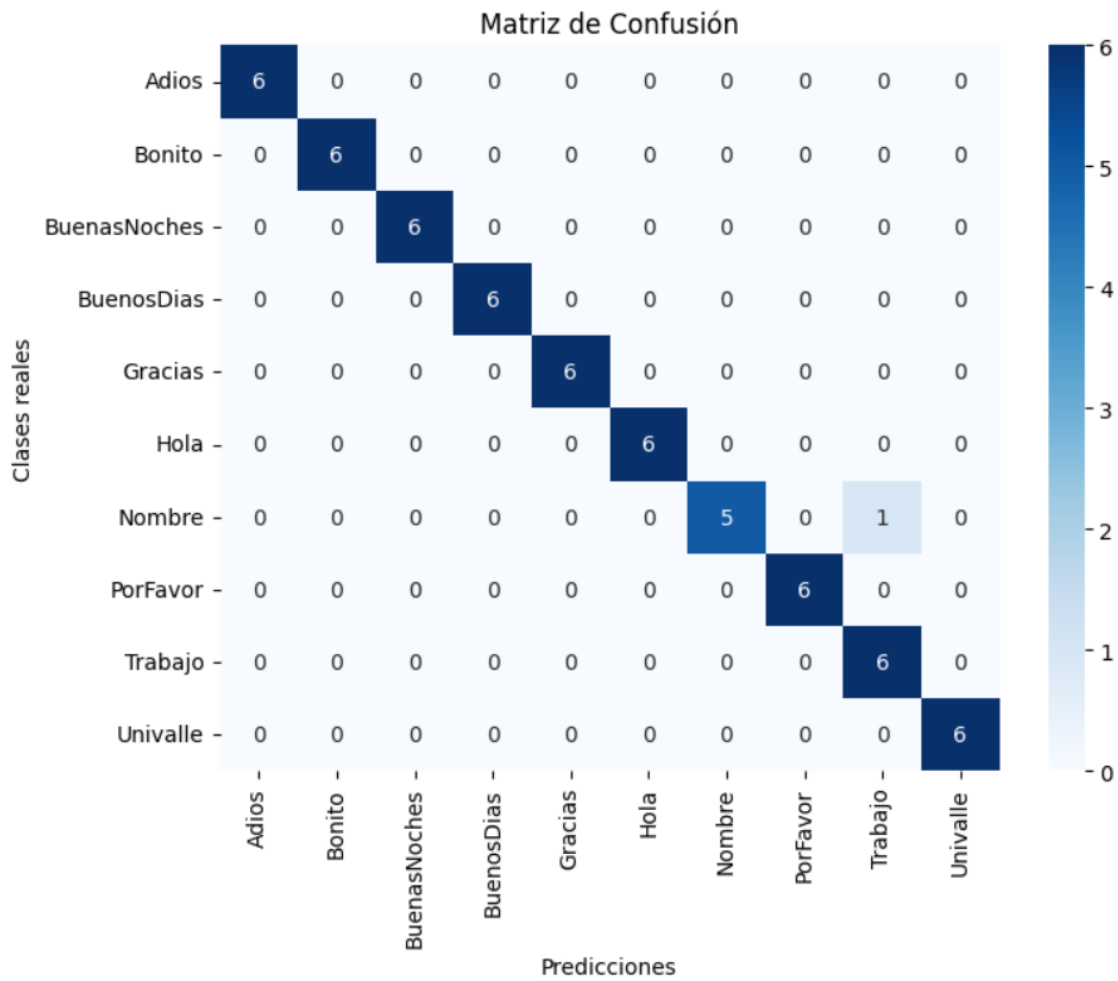


Figura 3.11: Ejemplo de una matriz de confusión.

Para facilitar la interpretación de la matriz de confusión, se genera una visualización mediante un mapa de calor. Además, la matriz se representa con etiquetas en los ejes, donde el eje horizontal indica las predicciones del modelo y el eje vertical representa las clases reales. Se utilizan colores para resaltar los valores más altos y se incluyen anotaciones con los números correspondientes. Finalmente, se añade un título descriptivo, junto con las etiquetas de los ejes para mejorar la claridad de la información antes de mostrar la figura.

En un problema de clasificación multiclase, los términos True Positive (TP), True Negative (TN), False Positive (FP) y False Negative (FN) se interpretan para cada clase individual considerando un enfoque uno contra todos (*one-vs-all*). Para una clase

específica, TP representa la cantidad de veces que el modelo predijo correctamente esa clase cuando realmente pertenecía a ella. FN son los casos en los que el modelo no predijo esa clase, aunque sí era la correcta. FP se refiere a los casos en los que el modelo predijo erróneamente esa clase cuando en realidad pertenecía a otra. Finalmente, TN es el número de veces que el modelo predijo correctamente cualquier otra clase que no sea la clase en cuestión.

Adicionalmente, se genera un reporte de clasificación que proporciona métricas clave para evaluar el desempeño del modelo. Este reporte incluye la exactitud, la sensibilidad y el F1-score para cada clase. La exactitud indica la proporción de predicciones correctas sobre el total de predicciones realizadas para una clase específica (Ecuación 3.3).

La sensibilidad mide la capacidad del modelo para identificar correctamente todas las instancias de una clase (Ecuación 3.4). En este contexto, un valor alto de sensibilidad para una clase específica indica que el modelo reconoce correctamente la mayoría de las veces ese gesto cuando realmente se presenta. Por el contrario, una sensibilidad baja implica que el modelo frecuentemente no detecta correctamente ese gesto cuando ocurre.

El *F1-score* es el promedio armónico entre precisión y sensibilidad, ofreciendo un equilibrio entre ambas métricas (Ecuación 3.5). Este valor es especialmente útil cuando existe un desequilibrio en la cantidad de muestras por clase o cuando los errores de clasificación tienen un impacto importante en la interpretación del gesto. Por ejemplo, un F1-score bajo en una clase puede revelar que el modelo tiene dificultades tanto para identificar correctamente ese gesto como para no confundirlo con otros.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.3)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3.4)$$

$$F1_score = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \quad (3.5)$$

También se muestra la matriz de confusión generada a partir de los datos de entrenamiento, la cual permite evaluar el rendimiento del modelo al identificar cuántas muestras han sido clasificadas correctamente y cuáles han sido confundidas entre clases. A partir de esta visualización, es posible identificar clases que el modelo confunde con mayor frecuencia.

Para comprobar el funcionamiento del modelo entrenado en un caso práctico, se utiliza un video extraído de una base de datos externa con el objetivo de verificar si el sistema es capaz de reconocer correctamente un gesto realizado por una persona distinta y en un entorno diferente al empleado durante el entrenamiento. Se consideran 90 fotogramas para analizar cada video, lo que equivale aproximadamente a 3 segundos de duración del gesto.

El conteo de estos fotogramas comienza en el instante en que el programa detecta la presencia de manos. En caso de que no se detecten manos en alguno de los fotogramas, se asignan vectores con valores de cero, lo cual garantiza que todas las entradas tengan un tamaño uniforme. A continuación, los datos recolectados se normalizan dividiéndolos por su valor máximo, en concordancia con el procedimiento aplicado durante la etapa de entrenamiento del modelo.

Posteriormente, se carga el video desde una ubicación específica y se procesan sus imágenes para extraer los puntos clave, los cuales se almacenan en una lista. Con el fin de asegurar la consistencia temporal, si el video contiene más fotogramas de los necesarios, se eliminan los más antiguos; por el contrario, si hay menos, se agregan vectores nulos hasta alcanzar la cantidad requerida.

Una vez completado este proceso de extracción y normalización, el modelo analiza la secuencia de puntos clave y determina el gesto más probable representado en el video. Finalmente, el nombre del gesto identificado se muestra en pantalla, permitiendo así interpretar el significado del movimiento capturado.

3.2. Análisis con MediaPipe

Una vez recopilados los videos, se procede a su procesamiento para identificar y extraer los *landmarks* de las manos. Para ello, se utiliza MediaPipe, una biblioteca especializada en el procesamiento de datos visuales en tiempo real. En este caso, se implementa MediaPipe Hands, un modelo de detección y seguimiento de manos que permite extraer 21 puntos clave por mano en cada cuadro del video. Estos puntos representan la posición de las articulaciones de la mano, distribuidos de la siguiente manera: cuatro en cada dedo (20 en total) y uno adicional en la base de la palma, como se observa en la Figura 3.12.

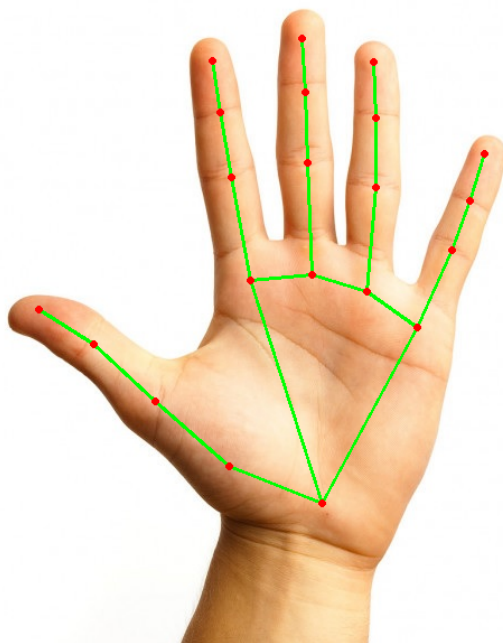


Figura 3.12: Detección de mano con MediaPipe. Tomado de [36].

Tras la obtención de los *landmarks*, se lleva a cabo la extracción de características. En esta etapa, se analizan las coordenadas de los puntos generados con el objetivo de identificar patrones distintivos que permitan diferenciar los distintos gestos.

Para verificar la correcta extracción de características, se selecciona 10 videos de entrenamiento y se muestra la estructura de los datos obtenidos tras extraer los *landmarks*. Donde se puede observar los primeros 5 frames de cada video y se logra

observar su funcionamiento de acuerdo a la normalización realizada. También se observa que algunos frames no detectaron las manos, por lo tanto aparece un vector de ceros. En cada fotograma se registran los valores correspondientes a la posición de los 42 puntos clave, los cuales representan diferentes partes de ambas manos, como se muestra en la Tabla 3.1.

Tabla 3.1: Posiciones de los puntos clave del gesto Adiós en el *Frame 1*

Índice	Valor X	Valor Y
0	-0.47598270	0.79139966
1	-0.49772903	0.84969240
2	-0.53184557	0.88274026
3	-0.55330807	0.90468955
4	-0.56621057	0.92091435
5	-0.56348250	0.86126715
6	-0.59281677	0.90492475
7	-0.60187110	0.94738466
8	-0.60295060	0.98732890
9	-0.55914290	0.85289650
10	-0.59401920	0.89474404
11	-0.60398420	0.94342405
12	-0.60452010	0.98903680
13	-0.54933670	0.84392965
14	-0.58323870	0.88499606
15	-0.59440583	0.93338600
16	-0.59775930	0.97673600
17	-0.53477000	0.83359987
18	-0.55917895	0.87135060
19	-0.57142910	0.90789485
20	-0.57631695	0.94129940

Al observar la Tabla 3.1, se puede verificar la correcta extracción de los 21 puntos clave correspondientes a la mano. En este caso, se hace alusión al primer video del gesto *Adiós*, el cual se realiza con la mano derecha. Por esta razón, los primeros 21 *landmarks* contienen valores representativos de su posición, mientras que los 21 restantes aparecen como vectores de ceros debido a la ausencia de detección en la mano izquierda.

Teniendo en cuenta estos valores, se calculará el sesgo con respecto a X y Y. Para ello, se toman todos los valores de X y Y de los keypoints y se obtiene el promedio de cada uno, resultando en $[-0.569, 0.900]$. El valor negativo de X indica que la mano se

encuentra en la parte izquierda del video (mano derecha de la persona). Si el valor fuera positivo, la mano estaría en la parte derecha del video. Por su parte, el valor positivo de Y indica que la mano se encuentra en la parte superior del video; un valor negativo indicaría que estaría en la parte inferior del video.

A continuación, se calcula la varianza de las coordenadas respecto a X y Y, obteniendo [0.0012, 0.0025]. Al ser valores bajos, esto indica que los dedos están relativamente cercanos al centro de la mano, con respecto a la vista de la cámara. Por el contrario, valores altos indicarían que la mano está completamente extendida.

Es importante destacar que estos valores están normalizados respecto al torso, lo que asegura consistencia en las coordenadas obtenidas, independientemente de la escala del video. En particular, se evidencia el correcto funcionamiento del sistema al observar que, en el primer fotograma del video correspondiente al gesto “Adiós”, los valores de X son negativos y los de Y positivos. Esto indica que la mano se encuentra hacia la izquierda y ligeramente por debajo del torso (según la vista de la cámara), lo cual coincide con la postura observada. En este punto, la mano derecha apenas inicia su elevación para realizar el gesto, tal como se aprecia en la Figura 3.13.



Figura 3.13: Primer frame capturado del gesto de *Adiós*

Sin embargo, en ciertos fotogramas, la detección de los puntos clave puede fallar, lo que impide su identificación adecuada. Este problema puede deberse a varios factores, como baja resolución del video, presencia de movimientos rápidos o ángulos de

visualización poco favorables, los cuales afectan la precisión del proceso de extracción.

Es fundamental considerar el uso de ambas manos en el proceso de extracción por lo que la matriz se expande para incluir un total de 42 puntos clave, 21 por cada mano. Los primeros 21 índices de la matriz contienen los valores correspondientes a la mano derecha, mientras que los 21 restantes representan la mano izquierda. De esta forma, cuando se emplea únicamente una mano, la matriz incluye 21 puntos reales y 21 nulos; en cambio, si se utilizan ambas manos, se completan los 42 puntos clave con los datos correspondientes.

Este enfoque garantiza que la red neuronal reciba una matriz de datos de tamaño igual, independientemente de si se utiliza una o ambas manos, lo que permite una mayor flexibilidad en el reconocimiento de gestos y una mejor generalización en el modelo. Además, este método facilita la interpretación y el procesamiento de los datos, ya que se mantiene una estructura consistente para las entradas de la red, sin importar si ambas manos están involucradas en el gesto.

Las coordenadas obtenidas de cada frame se comparan con la base de datos previamente recopilada, con el fin de determinar a qué gesto se asemeja más el video de entrada. La clave para la precisión de este procedimiento radica en la comparación detallada entre las coordenadas del video de entrada y las de los videos almacenados. Una vez extraídas, se tomarán las coordenadas de todos los videos y se entrenarán en la red neuronal LSTM. Ideal para procesar datos secuenciales. El modelo generará una salida única para cada clase, permitiendo al sistema determinar a qué clase corresponde con mayor probabilidad y, como resultado, identificar el gesto realizado.

En la Figura 3.14 se puede observar un ejemplo de la arquitectura LSTM donde la celda central tiene la capacidad de preservar y manipular información relevante durante secuencias largas. También se muestran tres bloques que corresponden a tres momentos en el tiempo: $t-1$, t y $t+1$. La celda LSTM recibe dos entradas principales: el vector de entrada actual x_t , y el estado oculto de la celda anterior h_{t-1} . A partir de estas entradas, la celda evalúa tres compuertas esenciales: la *compuerta de olvido*, la *compuerta de entrada* y la *compuerta de salida*, cada una de las cuales aplica una función sigmoide (σ) que produce valores entre 0 y 1, representando qué cantidad de cierta información se debe olvidar, agregar o mostrar [37].

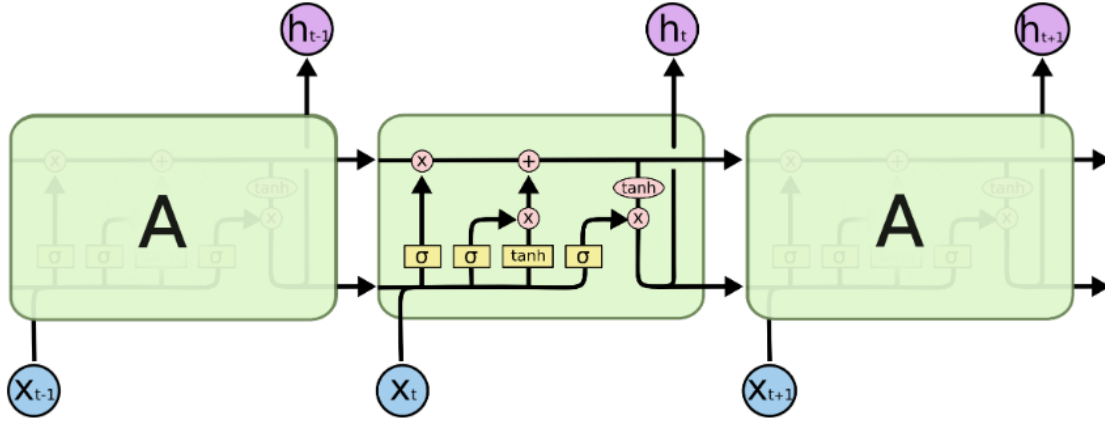


Figura 3.14: Estructura interna de LSTM. Tomado de [37].

La *compuerta de olvido* es la primera en actuar y decide qué parte del estado anterior de la celda C_{t-1} se debe conservar. Luego, la *compuerta de entrada* regula qué nueva información se añadirá al estado de la celda. Para esto, se calcula una “candidata a memoria nueva” mediante una función tangente hiperbólica ($\tanh$), y su resultado se multiplica por la salida de la compuerta de entrada, controlando cuánto de esta información se incorporará. Con estos valores, se actualiza el estado de la celda mediante una combinación entre la parte conservada de C_{t-1} y la nueva información relevante. Esta operación asegura que la celda LSTM pueda recordar datos importantes de manera persistente. Finalmente, la *compuerta de salida* determina cuál será el nuevo estado oculto h_t , que se enviará como salida y servirá como entrada para la siguiente celda en el tiempo. Esta salida se genera modulando el estado actualizado de la celda C_t con una función $\tanh$, para luego multiplicarlo por la señal de la compuerta de salida. De esta forma, el LSTM puede controlar cuánta información se revela en la salida de cada paso de tiempo, manteniendo otras partes ocultas para el futuro si así lo considera útil [37].

Gracias a este diseño cuidadosamente estructurado, una celda LSTM puede aprender a decidir qué información es importante, cuándo debe almacenarse y cuándo debe olvidarse, incluso en secuencias largas. Esto resuelve una de las principales limitaciones de las redes neuronales recurrentes tradicionales, que tienden a olvidar rápidamente las señales antiguas.

Ya con el modelo LSTM entrenado, se genera una matriz de confusión utilizando los datos de prueba con el fin de evaluar su capacidad de generalización. Esta matriz permite visualizar el comportamiento del modelo ante gestos no vistos durante el entrenamiento, identificando cuáles clases son reconocidas correctamente y cuáles no. Además, se calcula y presenta la exactitud obtenida para cada gesto de forma individual, así como la exactitud general del modelo.

La matriz de confusión correspondiente a los datos de entrenamiento se utiliza para evaluar el rendimiento del modelo sobre los datos que ha aprendido directamente, permitiendo identificar posibles casos de *overfitting* hacia alguna de las clases. Si el modelo presenta un desempeño notablemente superior en entrenamiento respecto a la prueba, esto puede ser un indicio de que ha memorizado los datos en lugar de aprender patrones generalizables. Esta comparación entre ambas matrices es clave para entender el equilibrio entre aprendizaje y generalización.

El modelo se guarda en un archivo con extensión .h5, para su reutilización sin necesidad de recompilar todo el sistema cada vez que sea necesario realizar una predicción. Esta estrategia no solo ahorra tiempo en futuras ejecuciones, sino que también asegura la consistencia en los resultados, ya que se conserva exactamente la arquitectura y los pesos obtenidos tras el entrenamiento. De esta forma, el sistema de reconocimiento de gestos estará listo para ser implementado en aplicaciones reales o para ser evaluado por otros usuarios sin requerir nuevamente el proceso de entrenamiento.

Una vez que el modelo ha sido guardado, es posible verificar su funcionamiento realizando predicciones sin necesidad de entrenarlo nuevamente. Para ello, se utiliza un video tomado de una base de datos externa que incluye los mismos gestos, aunque realizados por diferentes personas. Este video es procesado cuadro por cuadro mediante la herramienta MediaPipe, la cual permite extraer los puntos clave de las manos en cada fotograma. Posteriormente, estos datos se normalizan y se organizan en una estructura adecuada para ser ingresada al modelo. De esta forma, se puede verificar si el sistema es capaz de identificar correctamente el gesto presente en el video, evaluando así su desempeño en condiciones externas a las de entrenamiento y prueba inicial.



Figura 3.15: Gesto *Hola* realizado en base de datos externa.

3.3. Síntesis

En este capítulo se describe el desarrollo de un sistema inteligente para interpretar gestos de la Lengua de Señas Colombiana (LSC) utilizando una base de datos propia con 300 videos que representan 10 gestos dinámicos, grabados bajo diversas condiciones de ángulo, iluminación y distancia para asegurar la robustez del sistema. Se evaluaron tres técnicas de visión artificial para la detección de características: BRISK, ORB y MediaPipe Hands. Aunque BRISK y ORB ofrecieron ciertos beneficios, MediaPipe Hands fue seleccionada como la herramienta principal por su alta precisión en la detección de 21 puntos clave por mano.

Tras la extracción de las coordenadas de los landmarks, estas fueron normalizadas y las etiquetas correspondientes de los videos se codificaron en formato one-hot. Posteriormente, se implementó una red neuronal basada en LSTM utilizando TensorFlow y Keras, la cual incluyó capas de *BatchNormalization* y *Dropout* para mejorar la estabilidad del entrenamiento y reducir el riesgo de sobreajuste. Una vez finalizado el entrenamiento, se procedió a evaluar el desempeño del modelo con un conjunto de prueba. Para ello se plantean la matriz de confusión, que permite identificar los aciertos y errores de clasificación, y la exactitud.

Capítulo 4

Desarrollo Experimental y Análisis de Resultados

Este capítulo está dedicado a la validación experimental del sistema propuesto para el reconocimiento automático de gestos en lengua de señas, mediante el uso de técnicas de visión por computador y aprendizaje profundo. Con base en la metodología previamente establecida, se presentan las pruebas diseñadas para evaluar el desempeño del modelo en condiciones reales y controladas, considerando variaciones en los gestos, en los usuarios y en los entornos de grabación.

El análisis realizado abarca tanto los aspectos cuantitativos como cualitativos del sistema, incluyendo las métricas de evaluación. Asimismo, se detallan los resultados obtenidos en cada una de las pruebas, identificando los principales aciertos del modelo y señalando las limitaciones observadas durante su ejecución.

Además, este capítulo busca evidenciar el grado de generalización del sistema frente a gestos no vistos durante el entrenamiento, utilizando para ello videos externos. De esta manera, se establece un marco comparativo que permite valorar el aporte del modelo desarrollado y su aplicabilidad en contextos diversos, sentando las bases para futuras mejoras en el ámbito de la interpretación automática de la lengua de señas.

4.1. Evaluación Experimental

El primer paso dentro de la evaluación experimental consiste en analizar los resultados obtenidos por el modelo durante el proceso de entrenamiento. En las primeras épocas, el modelo presentó una exactitud de entrenamiento por debajo del 20% y un costo mayor de 2, lo cual es común al inicio del aprendizaje, ya que el

modelo aún está ajustando sus parámetros internos. Sin embargo, a medida que avanzaron las épocas, se evidenció una mejora progresiva por encima del 90 %.

El valor de la función de costo refleja qué tan lejos están las predicciones del modelo con respecto a los valores reales esperados. Un valor menor indica que el modelo está aprendiendo adecuadamente. Por otro lado, la exactitud representa el porcentaje de predicciones correctas sobre el total de ejemplos evaluados. Es una métrica especialmente útil cuando todas las clases tienen la misma importancia en el proceso de clasificación.

La representación gráfica de la matriz, facilita la interpretación visual de los resultados y permite detectar, por ejemplo, si el modelo tiende a confundir ciertos gestos entre sí de forma sistemática. Este tipo de análisis resulta fundamental en tareas como el reconocimiento de gestos, donde algunas clases pueden presentar similitudes visuales importantes, lo que podría requerir una mayor cantidad de muestras.

Una vez completado el entrenamiento, se analiza la matriz de confusión correspondiente al conjunto de datos de entrenamiento, lo que permite examinar con mayor precisión cómo el modelo está aprendiendo durante la fase de ajuste. Esta información es clave para identificar patrones específicos de confusión que podrían no ser evidentes al observar únicamente el valor global de exactitud.

Adicionalmente, se calcula la exactitud general sobre los datos de entrenamiento, lo que brinda una idea inicial del rendimiento del modelo en este conjunto. Sin embargo, al combinar esta métrica con la matriz de confusión, es posible evaluar si el modelo ha memorizado los datos de entrenamiento sin lograr una adecuada generalización, o por el contrario, está aprendiendo de forma balanceada y robusta.

Observando la Figura 4.1 matriz de confusión obtenida, es fundamental considerar el rendimiento general del modelo, teniendo en cuenta los conceptos de *overfitting* (sobreajuste) y *underfitting* (subajuste). El *overfitting* ocurre cuando el modelo aprende en exceso los datos de entrenamiento, incluyendo detalles específicos o ruido que no se generalizan bien a nuevos datos, lo que se traduce en un alto desempeño en entrenamiento pero un bajo rendimiento en la prueba. Por el contrario, el *underfitting*

se presenta cuando el modelo no logra capturar adecuadamente los patrones presentes en los datos, resultando en un rendimiento deficiente tanto en entrenamiento como en prueba.

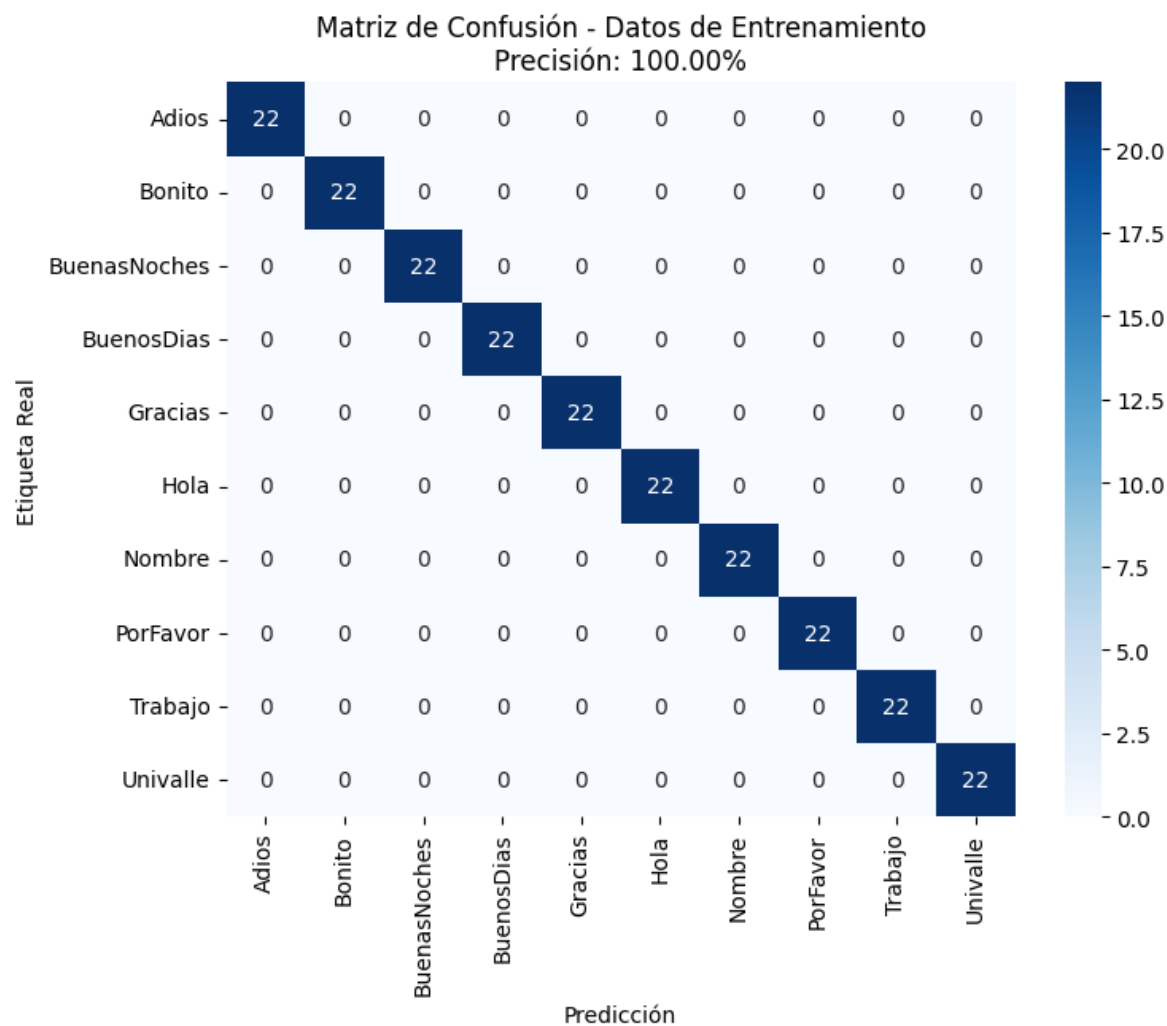


Figura 4.1: Matriz de confusión del entrenamiento

Al examinar la matriz de confusión correspondiente al conjunto de entrenamiento, se evidencia una clasificación completamente correcta de todos los gestos, lo que demuestra una excelente capacidad de discriminación por parte del modelo. Este comportamiento se atribuye a que los gestos utilizados en el entrenamiento presentan características visuales consistentes y suficientemente diferenciables entre sí.

A continuación, se procedió a evaluar el rendimiento del modelo utilizando los datos

del conjunto de prueba. Esta evaluación es fundamental para determinar qué tan bien generaliza el modelo frente a datos no vistos durante el aprendizaje. En esta etapa, se generaron las predicciones correspondientes y se compararon con las etiquetas reales.

La matriz de confusión de la Figura 4.2 permite evaluar el rendimiento del modelo de clasificación sobre el conjunto de datos de prueba. En esta matriz, las filas representan las clases reales y las columnas las predicciones realizadas por el modelo. Los valores diagonales indican la cantidad de aciertos por clase, mientras que los valores fuera de la diagonal representan errores de clasificación.

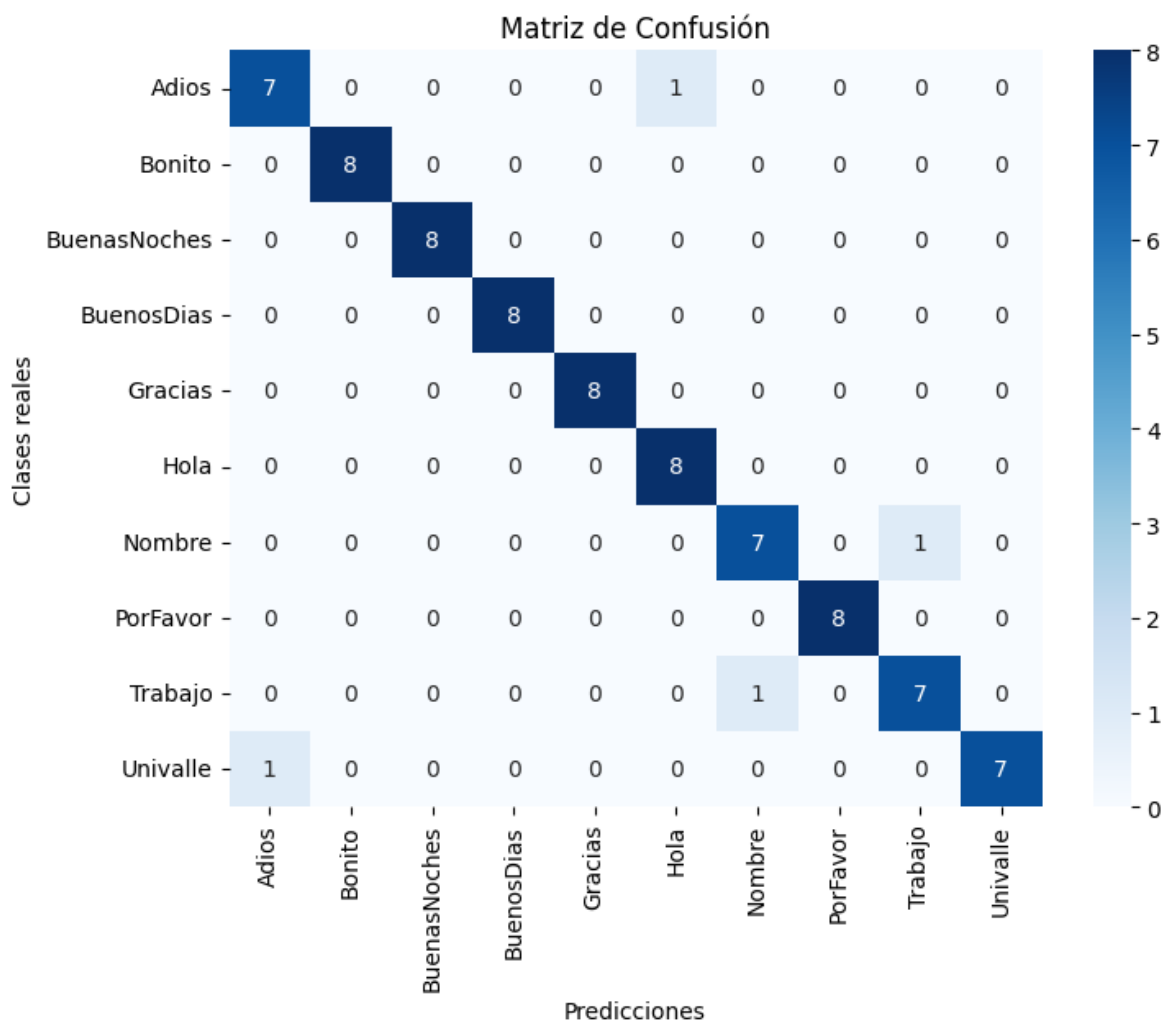


Figura 4.2: Matriz de confusión de la prueba

Observando la matriz de confusión, se evidencia que el modelo logró una predicción

perfecta para los siguientes gestos: *Bonito*, *BuenosDias*, *BuenasNoches*, *Gracias*, *Hola* y *PorFavor* en las cuales los 8 videos de prueba correspondientes fueron clasificados correctamente, sin presentar errores.

Dentro de la clase *Adios*, una muestra fue clasificada incorrectamente como *Hola*, esta equivocacion podrían deberse tanto a la similitud entre los gestos como a la variabilidad en los ángulos de grabación, ya que fueron capturados desde diferentes perspectivas, lo que puede dificultar la correcta identificación de las diferencias entre ellos.

En la Lengua de Señas Colombiana (LSC), el gesto de *Adiós* se caracteriza por un movimiento de la mano, con la palma abierta y orientada hacia el interlocutor. Este gesto se realiza agitando la mano de un lado a otro, de forma breve y repetitiva, como señal de despedida. Por su parte, el gesto de *Hola* también emplea la mano con la palma abierta, pero en este caso el movimiento consiste en una breve extensión hacia adelante desde la frente o desde el costado del rostro, como si se estuviera saludando con un leve empuje de la mano.

El gesto de *Adiós* comparte similitudes con el de *Hola*, particularmente en la postura inicial de la mano y su orientación hacia el interlocutor. La distinción principal entre ambos gestos radica en la dirección y repetición del movimiento: mientras que *Adiós* implica un movimiento lateral repetido, *Hola* se ejecuta como un único gesto hacia adelante.

La clase *Univalle* fue confundida una vez con el gesto *Adios*; el gesto de *Univalle* se realiza con una sola mano elevándola a la altura del pecho con los dedos meñique e índice extendidos, extiende la mano al pecho y luego, al retirar la mano, el meñique se encoge y se extiende el dedo medio, formando una “V” con el dedo índice y el medio. La mano vuelve brevemente al costado del pecho antes de regresar a una posición neutra.

La confusión entre ambos gestos por parte del modelo podría deberse a que se ejecutan en una zona similar del cuerpo, aproximadamente a la altura del pecho y del rostro, lo que genera una coincidencia espacial significativa. Esta cercanía en la ubicación puede dificultar la diferenciación entre gestos, especialmente cuando existen variaciones en el ángulo de grabación. Dichas variaciones pueden impedir una correcta

visualización de la posición de los dedos o provocar una pérdida de precisión en la detección de los puntos clave, afectando así el rendimiento del modelo.

El mayor desafío se presenta en la diferenciación entre los gestos *Nombre* y *Trabajo*, siendo esta la combinación que más afecta la exactitud del sistema. Ambos gestos comparten ciertas similitudes en términos de ubicación espacial y dirección del movimiento, lo que puede dificultar su distinción por parte del modelo, especialmente cuando hay variaciones en la ejecución.

Específicamente, el gesto de *Nombre* consiste en chocar de forma perpendicular los dedos índice y corazón de una mano con los de la otra, mientras que el gesto de *Trabajo* se realiza chocando los puños entre sí, como se puede observar en Figura 4.3 y en la Figura 4.4. Aunque existen diferencias claras en la configuración de las manos, la cercanía espacial y la similitud en la estructura del movimiento pueden generar confusiones en el sistema de reconocimiento. A esto se suman factores como las variaciones en el ángulo de grabación o la ausencia de detección en algunos cuadros, los cuales pueden llevar al modelo a interpretar estos gestos como equivalentes.



Figura 4.3: Frame Capturado del gesto de *Trabajo*.

Estas confusiones revelan que, a pesar del buen desempeño general del modelo, persisten ciertos patrones compartidos entre algunas clases que introducen ambigüedad en el proceso de predicción. Para mitigar este problema, podrían explorarse estrategias como el aumento de datos, incorporando más muestras por clase



Figura 4.4: Frame Capturado del gesto de *Nombre*.

y asegurando una mayor variabilidad en los ángulos, velocidad de ejecución y participantes. Esto permitiría que el modelo aprenda mejor las diferencias sutiles entre gestos similares y mejore su capacidad de generalización frente a nuevas muestras.

Estas confusiones revelan que persisten ciertos patrones compartidos entre algunas clases que introducen ambigüedad en el proceso de predicción. Para mitigar este problema, podrían explorarse estrategias como el aumento de datos, incorporando más muestras por clase y asegurando una mayor variabilidad en los ángulos, velocidad de ejecución y participantes. Esto permitiría que el modelo aprenda mejor las diferencias sutiles entre gestos similares y mejore su capacidad de generalización frente a nuevas muestras.

Complementando este análisis, se evaluaron las métricas estándar de clasificación: exactitud, sensibilidad, F1-score. La exactitud mide qué tan precisas fueron las predicciones del modelo para una clase dada (es decir, cuántas de las predicciones realizadas realmente pertenecían a esa clase). La sensibilidad indica la capacidad del modelo para identificar correctamente todos los ejemplos reales de una clase. Por su parte, el F1-score ofrece un equilibrio entre ambas métricas, siendo especialmente útil cuando existe un desbalance entre clases.

Analizando los resultados de la Tabla 4.1, se destaca que cinco clases lograron

métricas perfectas (1.00 en precisión, sensibilidad y F1-score), lo que refleja la capacidad del modelo para distinguir adecuadamente esos gestos específicos. Además, se observan valores consistentes tanto en el promedio macro como en el ponderado. El promedio macro de precisión, sensibilidad y F1-score se sitúa en 0.97, 0.96 y 0.96 respectivamente, lo que indica que el modelo presenta un rendimiento equilibrado entre las distintas clases, sin importar cuántas muestras tenga cada una. Esta métrica es especialmente útil para detectar si el modelo está sesgado hacia clases específicas, ya que otorga el mismo peso a cada clase.

Tabla 4.1: Métricas del modelo obtenido

Clase	Precisión	Sensibilidad	F1-Score
Adios	0.88	0.88	0.88
Bonito	1.00	1.00	1.00
BuenasNoches	1.00	1.00	1.00
BuenosDias	1.00	1.00	1.00
Gracias	0.89	1.00	0.94
Hola	0.89	1.00	0.94
Nombre	1.00	0.88	0.93
PorFavor	1.00	1.00	1.00
Trabajo	1.00	1.00	1.00
Univalle	1.00	0.88	0.93
Exactitud	0.96		
Prom. macro	0.97	0.96	0.96
Prom. ponderado	0.97	0.96	0.96

Por otro lado, el promedio ponderado tiene en cuenta la cantidad de muestras por clase al calcular las métricas, proporcionando así una visión más realista del rendimiento general del modelo en función del desbalance de clases. En este caso, los valores de exactitud, sensibilidad y F1-score ponderados también se ubican en 0.97, 0.96 y 0.96 respectivamente, lo que indica que el modelo se comportó de manera eficiente incluso al considerar el tamaño relativo de cada clase dentro del conjunto de prueba. Cabe destacar que ambos promedios arrojaron resultados idénticos debido a que todas las clases cuentan con la misma cantidad de muestras, por lo que el peso asignado a cada una fue equitativo en los cálculos.

Teniendo en cuenta algunos trabajos previos que también utilizaron la lengua de señas e incluyeron gestos dinámicos [10, 17], se revisaron las métricas de desempeño

empleadas para evaluar los modelos. En [10], la métrica principal considerada fue la exactitud, aunque también se analizaron otras métricas mencionadas anteriormente para realizar una evaluación más completa del sistema, donde se obtuvo una exactitud de 80.74%, considerando un total de 27 gestos: 21 estáticos y 6 dinámicos. Por su parte, en [17] se tuvo en cuenta la cantidad de muestras de gestos clasificadas correcta e incorrectamente, lo cual también hace referencia a la exactitud del modelo, donde se obtuvo una exactitud de 54 %, utilizando 5 gestos dinámicos con 10 muestras por gesto.

El modelo alcanzó una exactitud global del 96 %, lo que refleja un desempeño sólido dadas las complejidades del reconocimiento de gestos en video. Esta métrica indica que el sistema clasificó correctamente 76 de los 80 videos de prueba, y representa la proporción de predicciones correctas con respecto al total de muestras evaluadas, ofreciendo una visión general del rendimiento del sistema en términos de aciertos globales.

El modelo alcanzó una exactitud del 100 % en los datos de entrenamiento y del 96 % en los datos de prueba, lo que sugiere un buen desempeño tanto en la fase de aprendizaje como en la de generalización. Esta ligera diferencia entre ambas métricas indica que el modelo ha logrado aprender adecuadamente los patrones presentes en los datos sin llegar a sobreajustarse, y que es capaz de adaptarse correctamente a datos nuevos sin comprometer significativamente su exactitud. Además, el buen rendimiento en las pruebas refuerza la idea de que el modelo no solo memoriza las muestras de entrenamiento, sino que también ha aprendido representaciones significativas que le permiten identificar correctamente nuevas instancias.

4.2. Aplicación en Caso Práctico

Una vez realizado el análisis del modelo y evaluadas sus respectivas métricas de desempeño, se procede a guardar el modelo en un archivo con extensión .h5. Este formato permite almacenar no solo la arquitectura del modelo, sino también los pesos entrenados, el optimizador y su estado, lo que facilita su posterior carga y utilización sin necesidad de reentrenamiento.

A continuación, se evalúa la capacidad del modelo para generalizar sobre datos no vistos previamente, utilizando una base de datos externa. En la Figura 4.5 se muestra un fotograma representativo de uno de los videos pertenecientes a esta base de datos. Para ello, se seleccionan 20 videos, dos videos por cada clase disponible. Estos videos no formaron parte del conjunto de entrenamiento ni del de validación, por lo que constituyen una fuente independiente para comprobar el rendimiento real del sistema.



Figura 4.5: Fotograma de base de datos externa

Es importante destacar que la base de datos utilizada en esta etapa está compuesta por videos grabados por tres personas diferentes. A diferencia de la base de datos empleada durante el entrenamiento, esta nueva colección de datos ofrece una mayor robustez gracias a la diversidad de usuarios y a la variabilidad en la velocidad de ejecución de los gestos. No obstante, cabe mencionar que los videos fueron capturados en un entorno con fondo controlado.

Cada video será procesado mediante el mismo procedimiento de extracción de coordenadas utilizado durante la etapa de entrenamiento, incluyendo la normalización correspondiente. Dado que los videos tienen diferentes duraciones, se tomarán en cuenta únicamente los 90 fotogramas posteriores al momento en que se detecta por primera vez una mano en el video.

De esta manera, se garantiza que las entradas que recibe el modelo mantengan el mismo formato y coherencia. Posteriormente, el modelo previamente guardado será

cargado y utilizado para predecir la clase correspondiente a cada video. Las predicciones serán comparadas con las clases reales para determinar si el modelo logra reconocer correctamente cada uno de los gestos representados.

La matriz de confusión mostrada en la Figura 4.6 corresponde al desempeño de un modelo de reconocimiento de señas evaluado con la base de datos externa. Esto permite analizar la capacidad de generalización del sistema frente a datos nuevos y no vistos previamente, lo cual es un criterio fundamental para determinar su utilidad en escenarios reales.

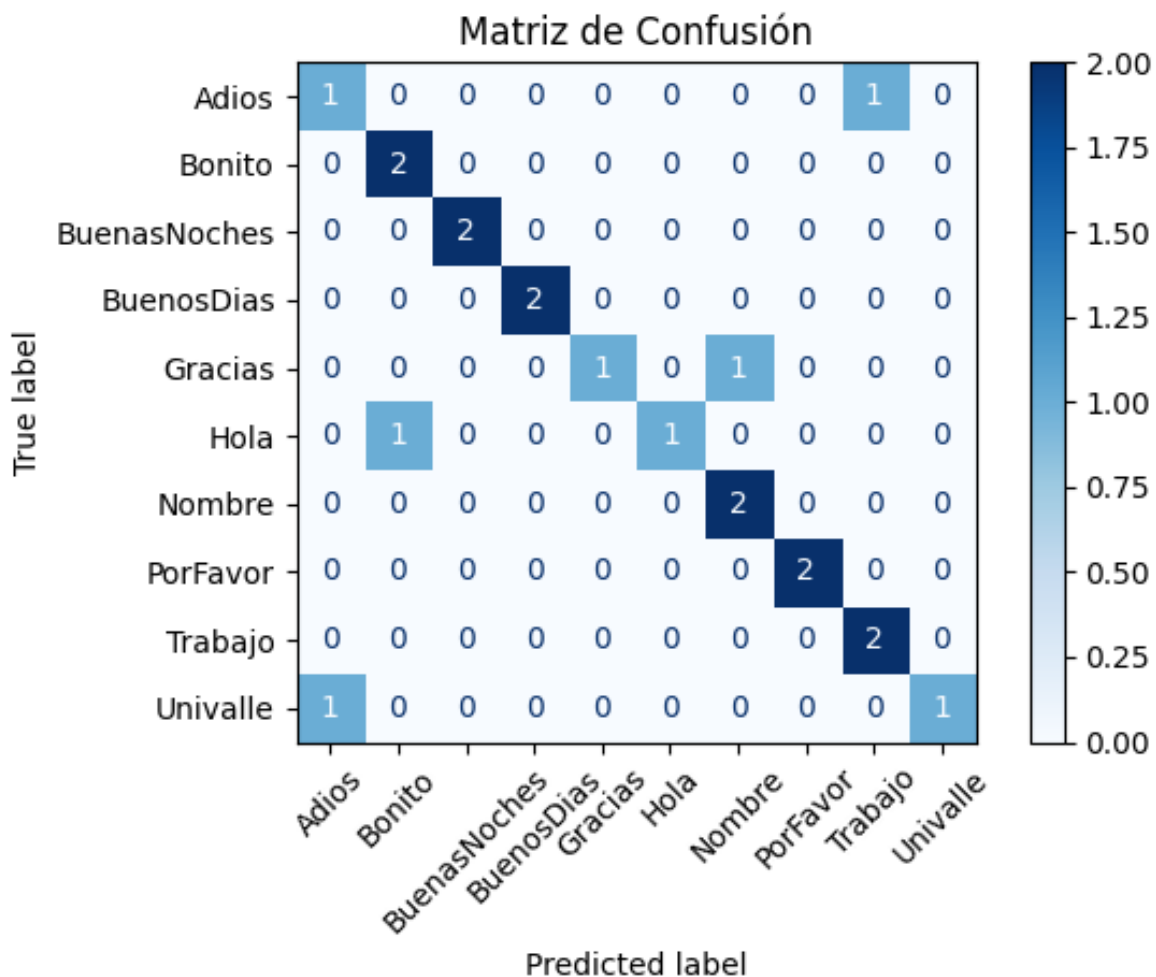


Figura 4.6: Matriz de prueba de la base de datos externa

Se evaluaron los 10 gestos del modelo entrenado: *Adios*, *Bonito*, *BuenasNoches*, *BuenosDias*, *Gracias*, *Hola*, *Nombre*, *PorFavor*, *Trabajo* y *Univalle*. En general, el

modelo mostró un buen desempeño, con una alta tasa de aciertos en la mayoría de las clases. Gestos como *Bonito*, *BuenasNoches*, *BuenosDias*, *Nombre*, *PorFavor* y *Trabajo* fueron clasificados correctamente en todos los casos, lo que indica una buena precisión en su reconocimiento.

No obstante, se presentaron algunas confusiones puntuales. Por ejemplo, el gesto *Adios* fue clasificado incorrectamente una vez como *Trabajo*. *Gracias* fue confundido con *Nombre*, *Hola* con *Bonito* y *Univalle* con *Adios*.

El sistema comienza a capturar los 90 fotogramas correspondientes al gesto cuando las manos del usuario ya son visibles, incluso antes de iniciar propiamente la ejecución del gesto. Por ejemplo, en la confusión del gesto *Adios*, esta detección prematura provocó que el sistema comenzara a registrar movimientos previos irrelevantes, los cuales no forman parte del gesto. Esta captura anticipada puede alterar la secuencia de entrada y afectar negativamente el proceso de reconocimiento.

En el caso del gesto *Gracias*, la confusión con *Nombre* podría deberse a la similitud en el patrón de movimiento: ambos involucran un desplazamiento ascendente y descendente de las manos, lo cual puede llevar al modelo a asociarlos incorrectamente si no se aprecian diferencias claras en la ejecución.

La confusión entre *Hola* y *Bonito* también puede explicarse por la proximidad de la mano al rostro al inicio del gesto *Hola*, lo que podría inducir al modelo a interpretar el inicio como correspondiente al gesto *Bonito*, el cual también implica movimiento facial o cercano al rostro.

Finalmente, el error de clasificación de *Univalle* como *Adios* puede deberse a una ejecución atípica por parte del usuario, quien realizó el gesto de forma acelerada y simplificada, omitiendo etapas clave del movimiento. Esta variación en la forma de ejecución puede reducir la capacidad del modelo para identificar correctamente el gesto.

La exactitud del modelo es del 80 %, habiendo clasificado correctamente 16 videos de un total de 20. Este buen desempeño general frente a una base de datos externa indica que el sistema posee un grado aceptable de capacidad de generalización, lo cual es un resultado positivo. Sin embargo, las confusiones observadas revelan que aún existe

margen de mejora, especialmente en términos de robustez ante entradas variadas o inesperadas.

Los errores de clasificación identificados en la matriz de confusión evidencian la importancia de contar con una base de datos robusta y representativa. Esto se debe a que los errores observados en esta matriz de prueba no coinciden con los errores presentes durante la evaluación del modelo con los datos utilizados en el entrenamiento. Esta discrepancia sugiere que el modelo puede no estar generalizando adecuadamente. Para mejorar el rendimiento y la capacidad de generalización del sistema, es fundamental ampliar la base de datos incluyendo mayor diversidad de ejemplos por clase, con diferentes usuarios, escenarios y estilos de ejecución. Esto permitiría al modelo aprender patrones más robustos y adaptarse mejor a nuevas muestras en condiciones reales.

También se sugiere incluir ejemplos negativos o gestos fuera de la base (para una futura clase de rechazo), lo que permitiría al modelo aprender a distinguir entre gestos válidos y entradas no pertenecientes al vocabulario definido. Estos ajustes contribuirían significativamente a mejorar la generalización del modelo y su aplicabilidad en escenarios reales.

Es importante tener en cuenta el tiempo de procesamiento si se desea implementar el sistema en entornos reales. Con videos de 7 a 8 segundos, el programa tarda entre 13 y 17 segundos en procesarlos, lo que puede ser una limitación para aplicaciones en tiempo real.

El modelo se puede mejorar si solo se toman en cuenta los segundos importantes del gesto, es decir, el segmento donde realmente se ejecuta la acción significativa. De esta manera, no solo se reduciría el tiempo de procesamiento, sino que también se podría mejorar la precisión del modelo, ya que se eliminaría el ruido causado por los segundos iniciales o finales sin información útil.

Estos errores podrían deberse a la similitud en los movimientos entre estos gestos. No obstante, el desempeño general del modelo demuestra una buena capacidad de generalización frente a datos no vistos. Sin embargo, estas fallas evidencian la necesidad de seguir fortaleciendo el sistema, ya sea incrementando la cantidad y diversidad de los

datos de entrenamiento, optimizando la arquitectura del modelo o incorporando clases de rechazo para disminuir la probabilidad de clasificaciones erróneas.

Es importante tener en cuenta el tiempo de procesamiento si se desea implementar el sistema en entornos reales. Con videos de 7 a 8 segundos, el programa tarda entre 13 y 17 segundos en procesarlos, lo que puede ser una limitación para aplicaciones en tiempo real.

El modelo se puede mejorar si solo se toman en cuenta los segundos importantes del gesto, es decir, el segmento donde realmente se ejecuta la acción significativa. De esta manera, no solo se reduciría el tiempo de procesamiento, sino que también se podría mejorar la precisión del modelo, ya que se eliminaría el ruido causado por los segundos iniciales o finales sin información útil.

4.3. Resultados usando BRISK y ORB

Se extrajo vectores de características utilizando los descriptores BRISK y ORB, aunque no fueron seleccionados como método principal, sirven para contrastar el desempeño general del sistema. Para ello, se aplicó un preprocesamiento basado en la sustracción de fondo, con el fin de aislar las regiones con mayor movimiento en cada cuadro del video. Posteriormente, los descriptores se emplearon para detectar un máximo de 50 puntos clave exclusivamente dentro de estas zonas activas, siguiendo el mismo enfoque descrito en el Capítulo 2. Las coordenadas (X, Y) de los puntos detectados se tomaron y se organizaron en vectores unidimensionales de longitud 100 por cuadro, correspondientes a las posiciones de cada punto clave.

A continuación se presentan las matrices de confusión correspondientes a los descriptores BRISK y ORB, ilustradas en la Figura 4.7 y la Figura 4.8, respectivamente. En ambos casos, se evidencia un desempeño deficiente por parte del sistema de reconocimiento, con un patrón de clasificación erróneo bastante marcado. Para el descriptor BRISK, se observa que la mayoría de gestos, fueron clasificados como el gesto *Univalle*. Esto incluye gestos claramente diferentes como *Adiós* o *BuenasNoches*, este último realizado con ambas manos. En ambos casos, se obtuvo un

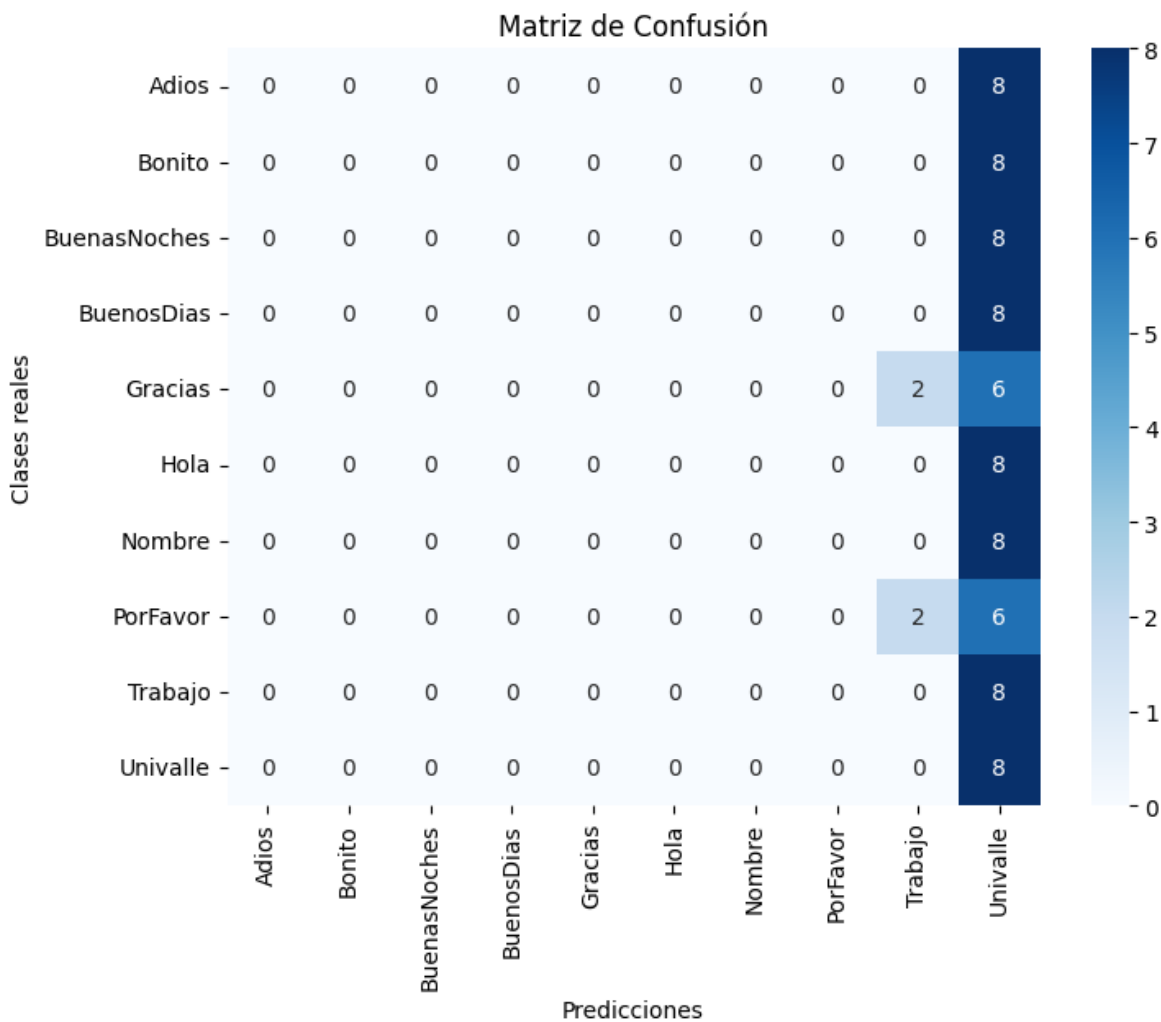


Figura 4.7: Matriz de confusión con BRISK

100% de error, lo que evidencia que el modelo no fue capaz de distinguir adecuadamente entre las distintas clases durante la fase de entrenamiento.

En el caso del descriptor ORB, si bien se aprecia una ligera mejora en el comportamiento general del sistema, los resultados siguen siendo insatisfactorios. La mayoría de los gestos fueron clasificados como *BuenasNoches*, con excepciones puntuales como *Bonito*, *BuenosDias*, *Gracias*, *Nombre* y *Trabajo*, donde se observa cierta dispersión en las predicciones. Sin embargo, los gestos *Adios*, *Hola*, *PorFavor* y *Univalle* fueron completamente clasificadas erróneamente como *BuenasNoches*, lo cual sugiere que en ciertos casos el descriptor ORB logró capturar características visuales

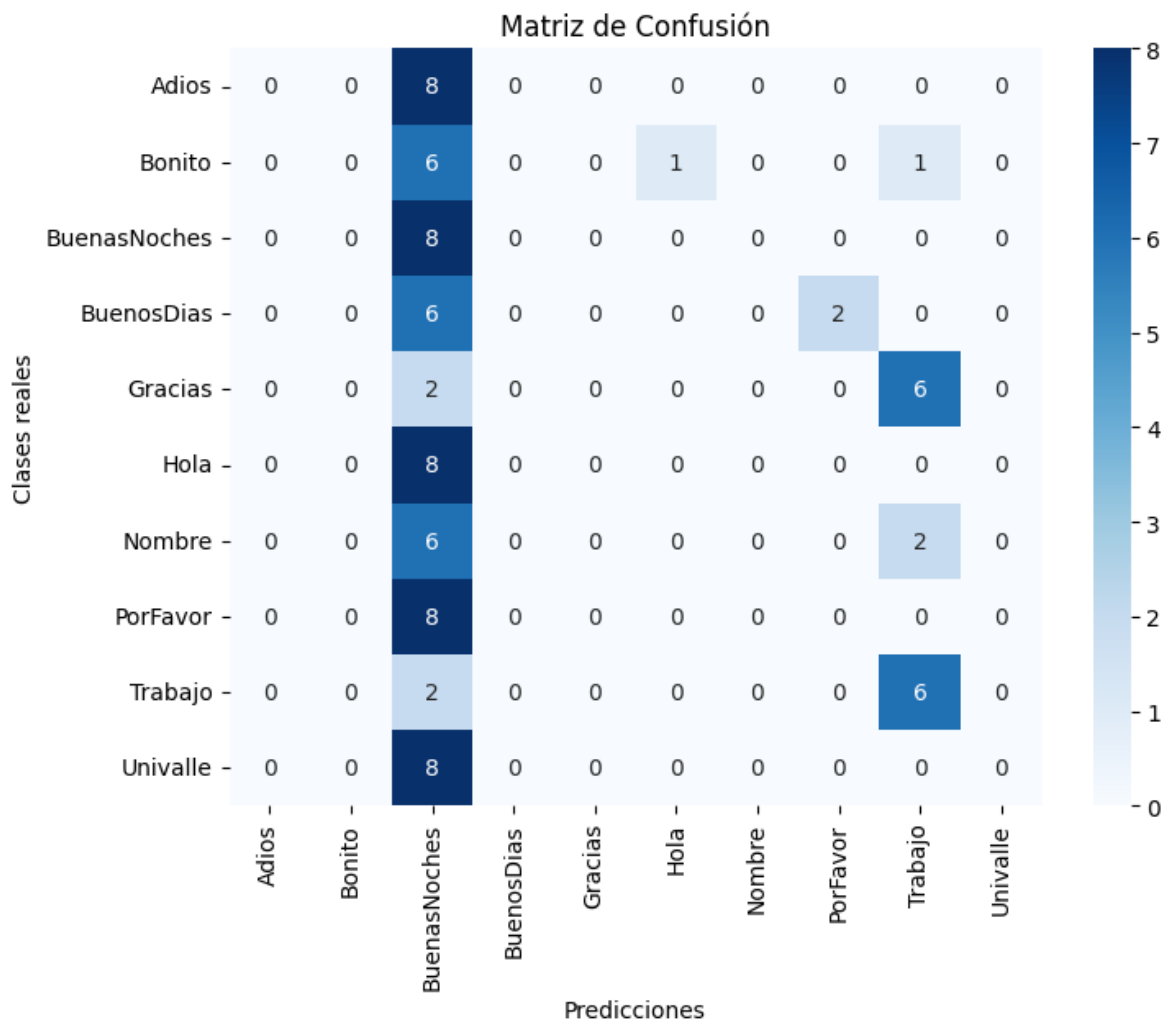


Figura 4.8: Matriz de confusión con ORB

relevantes, aunque no de manera suficiente para distinguir correctamente entre las clases.

Esto puede explicarse por diversas razones, tanto BRISK como ORB son descriptores locales diseñados para la extracción de características en imágenes estáticas, por lo que no están optimizados para capturar información temporal, la cual es fundamental en el reconocimiento de gestos dinámicos. Aunque el sistema procese los videos cuadro por cuadro, se pierde el contexto del movimiento y la esencia misma del gesto. En otras palabras, tratar cada frame como una entidad aislada, sin una interpretación de su secuencia, equivale a observar fotogramas de un gesto sin entender el orden ni la

progresión del movimiento.

Además, estos descriptores no están especializados en detectar únicamente las manos o las partes del cuerpo que participan activamente en el gesto. BRISK y ORB tienden a localizar puntos clave en regiones con alto contraste o variación local, lo que puede incluir elementos irrelevantes como la ropa o incluso el rostro del sujeto. La captación de estas regiones introduce ruido en los vectores de características, dificultando que el modelo de clasificación logre identificar patrones representativos y distinguibles entre gestos.

Por último, la cantidad de keypoints extraídos por cuadro también juega un papel importante. Un número excesivamente bajo de puntos puede hacer que se pierdan detalles clave del gesto, especialmente si los puntos detectados no corresponden a las regiones importantes como las manos. Por otro lado, un número demasiado alto puede aumentar el ruido, ya que incrementa la probabilidad de capturar zonas irrelevantes o estáticas.

Capítulo 5

Conclusiones y Trabajos Futuros

5.1. Conclusiones Generales

- La propuesta desarrollada alcanzó una exactitud del 96 % en el reconocimiento de señas, lo que demuestra un alto rendimiento. Este resultado valida el enfoque y resalta el potencial de aplicar estos sistemas en contextos reales de comunicación entre personas sordas y oyentes. Se destaca su posible integración en plataformas de atención al cliente, entornos educativos inclusivos y sistemas de traducción automática en tiempo real. Sin embargo, el modelo fue entrenado con una base de datos limitada en cantidad de videos y diversidad de usuarios, lo que puede afectar su capacidad de generalización.
- Herramientas como MediaPipe (gratuita y de código abierto) han tenido un papel clave en la democratización del desarrollo de soluciones basadas en visión artificial. Gracias a su capacidad para detectar manos y *landmarks* de forma eficiente, estas librerías han simplificado procesos que antes requerían hardware especializado. Esto ha permitido que investigadores y desarrolladores puedan crear sistemas funcionales y accesibles sin grandes inversiones en infraestructura, impulsando así la innovación y la exploración de nuevas aplicaciones.
- La variabilidad entre personas y elementos como la posición de la cámara siguen siendo factores que afectan significativamente el desempeño de los sistemas de reconocimiento. Por esta razón, resulta indispensable contar con bases de datos lo suficientemente diversas y robustas, así como aplicar técnicas de normalización de datos que mejoren la capacidad de generalización del sistema frente a estas variaciones.
- El desarrollo de sistemas de reconocimiento de señas mediante visión artificial tiene un alto potencial de impacto social. Estas tecnologías pueden ser utilizadas

para eliminar barreras de comunicación entre personas sordas y oyentes, permitiendo traducción automática en tiempo real. Además, abren nuevas oportunidades para la educación inclusiva y una atención más equitativa en servicios públicos, médicos y educativos.

5.2. Trabajos Futuros

Existen diversas oportunidades para mejorar y ampliar el sistema desarrollado. Un primer paso sería incrementar el tamaño y la diversidad de la base de datos, incorporando una mayor cantidad de gestos para aumentar la cantidad de clases reconocidas por el modelo. Esto permitiría aplicar el sistema en entornos más prácticos y con una mayor variedad de señales.

Además, sería beneficioso incrementar el número de videos por clase, lo que contribuiría a mejorar la robustez y exactitud del modelo, reduciendo la posibilidad de errores en la clasificación. Para manejar esta mayor cantidad de datos de manera eficiente, se requeriría una máquina con mayor capacidad de procesamiento, lo que optimizaría los tiempos de entrenamiento y ejecución del sistema.

Una futura implementación permitiendo que el sistema sea utilizado en aplicaciones interactivas o en entornos donde la detección instantánea de gestos sea necesaria. Para ello, sería importante investigar estrategias de optimización en hardware, como el uso de *Graphics Processing Unit* (GPU) o *Tensor Processing Unit* (TPU), que permitan acelerar el procesamiento sin sacrificar la calidad de los resultados.

Otra alternativa útil para implementar un sistema de reconocimiento de gestos en entornos reales es la incorporación de una clase de rechazo, que permite identificar gestos no pertenecientes a la base de datos de entrenamiento. Esto evita clasificaciones erróneas ante entradas desconocidas, mejorando la fiabilidad y seguridad del sistema en aplicaciones prácticas. Además, pueden explorarse diversas estrategias para su implementación en contextos reales.

Referencias

- [1] J. Loro, “Personas con discapacidad auditiva: realidad y principales adaptaciones,” *Balance Sociosanitario*, 2021.
- [2] P. J. C. Carrasco, “Aprendizaje del lenguaje en niños sordos: fundamentos para la adquisición temprana de lenguaje de señas,” *Temas de Educación*, vol. 19, pp. 14–27, 2002.
- [3] P. C. Carrasco, “Aprendizaje del lenguaje en niños sordos: fundamentos para la adquisición temprana de lenguaje de señas,” *Cultura Sorda*, 2002.
- [4] L. L. Munar, “Colombia tiene porcentaje bajo en la implementación de los derechos de las personas sordas,” *Infobae*, 2023.
- [5] F. Ronchetti, “Reconocimiento de gestos dinámicos y su aplicación al lenguaje de señas,” in *XX Workshop de Investigadores en Ciencias de la Computacion (WICC 2018, Universidad Nacional del Nordeste).*, 2018.
- [6] M. B. Stegmann and D. D. Gomez, “A brief introduction to statistical shape analysis,” *Informatics and mathematical modelling, Technical University of Denmark, DTU*, vol. 15, no. 11, 2002.
- [7] U. C. de Madrid, “¿qué es el machine learning?,” *Universidad Complutense de Madrid*, 2021.
- [8] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, pp. 91–110, 2004.
- [9] A. Preet, “Facial landmarks detection using mediapipe library,” 22 Mar, 2022.
- [10] K. M. Pérez and D. M. C. Esparza, “Reconocimiento de la lengua de señas mexicana mediante redes neuronales,” 2023.
- [11] B. V. and Gabriela, “Lenguaje de señas,” 2020.

- [12] F. Ronchetti, F. Quiroga, C. A. Estrebou, and L. C. Lanzarini, “Clasificación de configuraciones de manos del lenguaje de señas argentino con probsom,” in *XXI Congreso Argentino de Ciencias de la Computación (Junín, 2015)*, 2015.
- [13] E. A. Albino Huertas and L. T. López Olivos, “Visión computacional para la traducción en tiempo real del lenguaje de señas a texto en idioma español,” 2018.
- [14] B. Villa, V. Valencia, and J. Berrio, “Digital image processing applied on static sign language recognition system/diseño de un sistema de reconocimiento de gestos no móviles mediante el procesamiento digital de imágenes,” *Prospectiva*, vol. 16, no. 2, pp. 41–48, 2018.
- [15] N. Ortiz-Farfán and J. E. Camargo-Mendoza, “Modelo computacional para reconocimiento de lenguaje de señas en un contexto colombiano,” *TecnoLógicas*, vol. 23, no. 48, pp. 191–226, 2020.
- [16] N. E. S. Rojas, B. S. M. Serna, E. M. P. Velásquez, and O. S. R. Galeano, “Reconocimiento del abecedario de la lengua de señas colombiana con redes neuronales convolucionales,” *Orinoquia*, vol. 25, no. 1, pp. 25–30, 2021.
- [17] G. Pinzón-Bayona and Y. G. Sanabria-Orjuela, “Desarrollo de una aplicación móvil para traductor de lenguaje de señas mediante el uso de servicios web,” 2022.
- [18] T. Articles, “Introducción a la estimación de movimiento con flujo Óptico,” *Tanger Articles*, 2021.
- [19] A. González Marcos, F. J. Martínez de Pisón Ascacíbar, F. Alba Elías, M. Castejón Limas, J. B. Ordieres Meré, E. P. Vergara González, *et al.*, “Técnicas y algoritmos básicos de visión artificial,” *Técnicas y Algoritmos Básicos de Visión Artificial*, 2006.
- [20] P. Kishore, M. Prasad, D. A. Kumar, and A. Sastry, “Optical flow hand tracking and active contour hand shape features for continuous sign language recognition with artificial neural networks,” in *2016 IEEE 6th international conference on advanced computing (IACC)*, pp. 346–351, IEEE, 2016.
- [21] O. Documentation, “Feature detection and description - opencv,” 2024.

- [22] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.
- [23] A. Vina, “What is mediapipe? a guide for beginners,” *Roboflow*, 2024.
- [24] K. Pranav Durai, “Mediapipe – the ultimate guide to video processing,” *LearnOpenCV*, 2022.
- [25] Google, “Guía de detección de puntos de referencia de posiciones,” 2025.
- [26] S. A. K. Tareen and Z. Saleem, “A comparative analysis of sift, surf, kaze, akaze, orb, and brisk,” in *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pp. 1–10, 2018.
- [27] IBM, “¿qué es una red neuronal recurrente?.”
- [28] P. Huet, “Qué son las redes neuronales y sus aplicaciones,” 2023.
- [29] IBM, “Redes neuronales convolucionales (cnn).”
- [30] C. Shidling, “Difference between CNN and RNN architecture in deep learning,” 2022.
- [31] D. S. Team, “Introducción al concepto de lstm,” 2020.
- [32] S. Leutenegger, M. Chli, and R. Y. Siegwart, “Brisk: Binary robust invariant scalable keypoints,” in *2011 International Conference on Computer Vision*, 2011.
- [33] Google, “Mediapipe hands,” 2020.
- [34] FineProxy, “One-hot encoding,” 2024.
- [35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “Tensorflow: A system for large-scale machine learning,” in *12th*

USENIX Symposium on Operating Systems Design and Implementation (OSDI '16), (Savannah, GA, USA), USENIX Association, 2016.

- [36] programacionpython80889555, “Detector de manos en python, con «opencv» y «mediapipe».” 2021.
- [37] C. Olah, “Understanding lstm networks,” 2015.