Neiva, 29 de agosto de 2022

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

César Augusto Cerón Manrique, con C.C. No. 1003806733, Tannia Lucía Hernández Rojas, con C.C. No. 1010112732 , Autor(es) de la tesis y/o trabajo de grado titulado Interfaz gráfica para traducción de Lengua de Señas Colombiano (LSC) usando Python presentado y aprobado en el año 2022 como requisito para optar al título de Ingeniero electrónico; Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE:

Firma: César A. Cerón M.

EL AUTOR/ESTUDIANTE:

Firma: Tannia Hernández

TÍTULO COMPLETO DEL TRABAJO: Interfaz gráfica para traducción de Lengua de Señas Colombiano (LSC) usando Python

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
Cerón Manrique Hernández Rojas	César Augusto Tannia Lucía

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Robayo Betancourt	Faiber Ignacio

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre

PARA OPTAR AL TÍTULO DE: Ingeniero electrónico

FACULTAD: Ingeniería

PROGRAMA O POSGRADO: Ingeniería electrónica

CIUDAD: Neiva **AÑO DE PRESENTACIÓN: 2022** **NÚMERO DE PÁGINAS:** 63

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas___ Fotografías___ Grabaciones en discos___ Ilustraciones en general_x_ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros_x_

SOFTWARE requerido y/o especializado para la lectura del documento: Microsoft Word

MATERIAL ANEXO:

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

Español	Inglés	Español	Inglés
1. Aprendizaje profundo	Deep learning	6. Matriz de confusión	Confusion matrix
2. Visión por computador	Computer Vision	7. Tkinter	Tkinter
3. Redes neuronales convolucionales	Convolutional neural network		
4. Inteligencia artificial	Python		
5. Python	Python		

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

El objetivo de este proyecto es crear una interfaz gráfica que realice la traducción del lenguaje de señas colombiano mediante el uso del lenguaje de programación Python aplicado al desarrollo de redes neuronales convolucionales.

Este trabajo se realiza con el fin de mitigar las dificultades que se presentan en una comunicación entre personas con y sin dificultades auditivas. Para cumplir este propósito se crea un amplio conjunto de datos capturando alrededor de 5000 imágenes por seña. Con este conjunto se diseña una red neuronal convolucional de 13 capas que permite entrenar el modelo el cual realiza la predicción de la seña.

La interfaz se ejecuta en tiempo real con un botón de encendido y apagado de la cámara para iniciar y finalizar la predicción y cuenta con diferentes botones interactivos que permiten al usuario conocer el uso y funcionamiento de la interfaz, además de permitir mediante botones conocer cuál es la representación de cada letra del alfabeto en lengua de señas, haciendo la aclaración que solo se trabaja con 21 de las 27 letras del alfabeto, debido a que su traducción es una seña estática.

Este proyecto representa una contribución en la inclusión de personas con discapacidad auditiva, generando

una reducción a la brecha social presente en esta minoría. Además, de su contribución al campo tecnológico, en especial en el proceso de innovación e inmersión a la inteligencia artificial en la región del Huila.

ABSTRACT: (Máximo 250 palabras)

The aim of this project is to create a graphical interface that performs the translation of Colombian sign language using Python programming language applied to the development of convolutional neural networks. This project is done in order to mitigate the difficulties that arise in communication between people with and without hearing difficulties.

To fulfill this purpose, a large dataset is created by capturing about 5000 images per sign. With this set, a 13-layer convolutional neural network is designed to train the model that performs the signal prediction. The interface is executed in real time with an on and off button of the camera to start and end the prediction.

Also, has different interactive buttons that allow the user to know the use and operation of the interface, in addition to allowing through buttons to know what is the representation of each letter of the alphabet in sign language, making the clarification that only works with 21 of the 27 letters of the alphabet, because its translation is a static sign.

This project represents a contribution to the inclusion of people with hearing disabilities, generating a reduction of the social gap present in this minority. In addition, it is contribution to the technological field, especially in the process of innovation and immersion in artificial intelligence in the region of Huila.

APROBACION DE LA TESIS

Nombre Jurado: Martin Diomedes Bravo Obando

Firma: 

Nombre Jurado: José de Jesús Salgado Patrón

Firma: 

INTERFAZ GRÁFICA PARA TRADUCCIÓN DE LENGUA DE SEÑAS COLOMBIANO (LSC) USANDO PYTHON

CÉSAR AUGUSTO CERÓN MANRIQUE
TANNIA LUCÍA HERNÁNDEZ ROJAS

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA, COLOMBIA
2022

INTERFAZ GRÁFICA PARA TRADUCCIÓN DE LENGUAJE DE SEÑAS COLOMBIANO (LSC) USANDO PYTHON

CÉSAR AUGUSTO CERÓN MANRIQUE Cod. 20171154848
TANNIA LUCÍA HERNÁNDEZ ROJAS Cod. 20171154840

Trabajo de grado para aplicar
al título de ingeniero electrónico

Director:
Mag. Faiber Robayo Betancourt

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA, COLOMBIA
2022

Notas de aceptación

_____
_____
_____
_____
_____
_____

_____
Firma del director de Tesis

_____
Firma del Jurado

_____
Firma del Jurado

Neiva, 29 de julio de 2022.

En primer lugar, dar gracias a Dios por guiarme
en esta etapa de mi vida, ofreciendo lo mejor para
lograr esta meta y que sin importar lo difícil que sea
el camino siempre se puede llegar a su fin.

A mis padres por enseñarme la importancia de la educación
en nuestras vidas, por sus sacrificios y esfuerzos para
brindarme un buen futuro y por su acompañamiento
en mi desarrollo personal. A mi hermana por todo su apoyo
durante este proceso siendo mi mayor ejemplo para seguir,
resaltando sus consejos para superarme y construir mi vida profesional.

A Tannia, que aparte de ser mi compañera de tesis fue mi
acompañamiento durante todo mi proceso formativo sin
importar los obstáculos que se presentaron.

A mis amigos, en especial a Pedro, Paola, Esteban y Joseph
que sin importar las buenas o malas ocasiones siempre estaban
presentes.

César

Doy comienzo dando gracias a Dios, que ha obrado en mí, con cada uno de los dones y regalos con los que me ha bendecido, en especial mi familia, que es el obsequio más preciado. Agradezco a mi padre José Alonso Hernández que, con dedicación, responsabilidad y absoluta entrega, me ha enseñado el valor de la resiliencia, firme para enfrentar con fortaleza cada adversidad. A mi madre Amparo Rojas, que con su amor, devoción y cuidado me ha permitido reposar y encontrar aliento, me protege y me orienta siempre a actuar con la mejor intención y de la mejor manera. Mi hermana, Karen Sofía que ha sido mi guía, mi compañera, mi mejor amiga, soy fiel admiradora de su complicidad, su bondad y generosidad gracias por caminar conmigo siempre. A mi novio Marlon Alejandro Rojas, mi compañero, que me abraza con su nobleza y paciencia, que me acompaña y me sostiene en cada paso, gracias por su fidelidad y compromiso. A los amigos que me regalo la Universidad, Julián Zúñiga, Juan Pablo Monje, Paola Hernández, Joseph Caicedo, Nicolas Sánchez, Pedro Contreras, Juan Esteban Narváez, a todos y cada uno de ellos y a los que me hace falta nombrar infinitas gracias, sin ellos este proceso no hubiera sido tan divertido y enriquecedor, no solo he ganado un título, sino algo mucho más valioso que es una lista de experiencias inolvidables y una verdadera amistad. Exaltando principalmente a mi amigo y compañero de tesis Cesar Augusto Cerón, ha sido un verdadero honor recorrer a tu lado este proceso, confío en que nos augura una carrera exitosa y si es juntos estaría encantada. Finalmente, a mis docentes, gracias por su generosidad y loable labor al compartir sus conocimientos y formar a los futuros ingenieros, especialmente al Ingeniero Faiber Robayo que ha sido un tutor del cuál he aprendido incontables conocimientos y espero seguir haciéndolo. Gracias a la ciudad de Neiva, por adoptarme como opita y por permitirme formarme en esta maravillosa Universidad.

Tannia

AGRADECIMIENTOS

Agradecemos a Dios por permitirnos terminar una etapa en nuestras vidas por medio de este proyecto representando un aporte a la sociedad, por brindarnos la sabiduría para afrontar todas las circunstancias adversas que se presentaron durante nuestra formación académica y por poner en nuestro camino a las personas indicadas que nos acompañaron en todos los momentos.

Agradecemos a nuestros compañeros y amigos que han sido un pilar indispensable para cumplir nuestro objetivo, haciendo más agradable y memorable este recorrido.

Agradecemos a todos los docentes del programa de ingeniería electrónica por contribuir en nuestro desarrollo como profesionales al servicio de la sociedad, principalmente al ingeniero Faiber Robayo por su acompañamiento y compromiso en su labor como docente y tutor de nuestro proyecto de grado.

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

LISTA DE ANEXOS

GLOSARIO

CNN: (Convolutional Neural Networks) Red Neuronal Convolucional, tipo de red neuronal artificial usada para identificar características (líneas, curvas, formas, etc.) mediante campos receptivos similares a la corteza visual del ojo humano. Son efectivas en tareas de visión artificial como la clasificación y segmentación de imágenes[1].

IA: (Inteligencia Artificial) Todo sistema o máquina que imitan la inteligencia humana para realizar tareas y que pueden mejorar iterativamente mediante la información que recopilan[2].

LSC: Lenguaje de Señas Colombiano[3].

MAX POOLING: Proceso encargado de reducir el tamaño de la imagen mediante un filtro que recorre toda la imagen en bloques de píxeles para agruparlos y extraer el mayor valor dicho bloque de imagen. Este proceso es usado para reducir la carga computacional de la capa posterior al generar como salida una imagen de menor tamaño[4].

ML: (Machine Learning) Una forma de inteligencia artificial que aprende de los datos, no por programación directa[5].

OPENCV: (Open Computer Vision) Biblioteca líder y de código abierto que es usada en visión por computador, procesamiento de imágenes y aprendizaje automático[6].

RGB: (Red-Green-Blue) Modelo cromático que permite representar distintos colores mediante la mezcla de estos 3 colores primarios[7].

---

[1] BAGNATO, Juan Ignacio. ¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador [blog]. Aprende Machine Learning en español. Coruña, España. 29 de noviembre de 2018. Disponible en: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/.

[2] ORACLE COLOMBIA. ¿Qué es la Inteligencia Artificial-IA? [En línea.]. Oracle Colombia web site. Disponible en: https://www.oracle.com/co/artificial-intelligence/what-is-ai.

[3] MINISTERIO DE EDUCACIÓN NACIONAL, INSTITUTO NACIONAL PARA SORDOS, INSOR. Diccionario Básico de la Lengua de Señas Colombiana por Instituto Nacional para Sordos INSOR [En línea.]. INSOR web site. Disponible en: http://www.insor.gov.co/descargar/diccionario_basico_completo.pdf.

[4] CIENCIACINÉTICA. Redes Neuronales Convolucionales en Inteligencia Artificial (CNN) [blog]. INTELIGENC[IA]. 6 de junio de 2018. Disponible en: https://inteligencia.tech/2018/06/06/redes-convolutivas-en-inteligencia-artificial/.

[5] IBM. ¿Qué es Machine Learning? [En línea.]. IBM Analítica web site. Disponible en: https://www.ibm.com/co-es/analytics/machine-learning.

[6] NVIDIA DEVELOPER. What is OpenCV? [En línea.]. NVIDIA Corporation web site. Disponible en: https://developer.nvidia.com/opencv.

[7] CASTILLO, José Antonio. RGB qué es esto y para qué se utiliza en Informática [blog]. Profesional Review. 20 de enero de 2019. Disponible en: https://www.profesionalreview.com/2019/01/20/rgb-que-es/.

RESUMEN

El objetivo de este proyecto es crear una interfaz gráfica que realice la traducción del lenguaje de señas colombiano mediante el uso del lenguaje de programación Python aplicado al desarrollo de redes neuronales convolucionales. Este trabajo se realiza con el fin de mitigar las dificultades que se presentan en una comunicación entre personas con y sin dificultades auditivas.

Para cumplir este propósito se crea un amplio conjunto de datos capturando alrededor de 5000 imágenes por seña. Con este conjunto se diseña una red neuronal convolucional de 13 capas que permite entrenar el modelo el cual realiza la predicción de la seña.

La interfaz se ejecuta en tiempo real con un botón de encendido y apagado de la cámara para iniciar y finalizar la predicción y cuenta con diferentes botones interactivos que permiten al usuario conocer el uso y funcionamiento de la interfaz, además de permitir mediante botones conocer cuál es la representación de cada letra del alfabeto en lengua de señas, haciendo la aclaración que solo se trabaja con 21 de las 27 letras del alfabeto, debido a que su traducción es una seña estática.

Este proyecto representa una contribución en la inclusión de personas con discapacidad auditiva, generando una reducción a la brecha social presente en esta minoría. Además, de su contribución al campo tecnológico, en especial en el proceso de innovación e inmersión a la inteligencia artificial en la región del Huila.

ABSTRACT

The aim of this project is to create a graphical interface that performs the translation of Colombian sign language using Python programming language applied to the development of convolutional neural networks. This project is done in order to mitigate the difficulties that arise in communication between people with and without hearing difficulties.

To fulfill this purpose, a large dataset is created by capturing about 5000 images per sign. With this set, a 13-layer convolutional neural network is designed to train the model that performs the signal prediction.

The interface is executed in real time with an on and off button of the camera to start and end the prediction. Also, has different interactive buttons that allow the user to know the use and operation of the interface, in addition to allowing through buttons to know what is the representation of each letter of the alphabet in sign language, making the clarification that only works with 21 of the 27 letters of the alphabet, because its translation is a static sign.

This project represents a contribution to the inclusion of people with hearing disabilities, generating a reduction of the social gap present in this minority. In addition, it is contribution to the technological field, especially in the process of innovation and immersion in artificial intelligence in the region of Huila.

INTRODUCCIÓN

La comunicación es el medio que poseen los seres humanos para convivir como sociedad. “El hombre es un ser naturalmente sociable”[8] y, por ende, nace con la necesidad de caracterización social y relación humana. Esta habilidad permite el avance en sociedad y cualquier alteración de este proceso, afecta considerablemente la calidad de vida del individuo y de la comunidad que lo rodea.

Entre las limitaciones posibles en la comunicación, se encuentran las personas con discapacidades auditivas. Esta situación que presenta parte de la comunidad ya sea por una condición de nacimiento o por un desafortunado evento, afecta el proceso adecuado de comunicación y excluye a las personas que lo presentan.

Entre las personas con discapacidades que afectan la expresión oral, se ha adaptado un grupo lingüístico particular, en el cual su medio de comunicación es la lengua de señas. La lengua de señas está compuesta por códigos desarrollados de manera visual y gestual, en el que la persona depende de sus manos, rostro y parte superior del cuerpo para lograr comunicarse[9]. Aunque dicho lenguaje, es popular entre esta comunidad, sigue siendo una minoría excluida en un contexto social.

En general la comunicación es imprescindible para el desarrollo de la personalidad, el consentimiento en las acciones y sobre todo en la coexistencia en el mundo. Es por lo que, entendiendo la dificultad que presentan algunas personas para lograr comunicarse y las innumerables opciones que brinda la tecnología en todas las áreas que involucran al ser humano, se han creado técnicas, procedimientos y/o herramientas[10,11] que son de ayuda para el reconocimiento de gestos correspondientes al lenguaje de señas con el fin de mitigar los obstáculos que imposibilitan una fluida comunicación.

El proyecto que se propone en este documento, pretende ser un aporte a los trabajos referenciados previamente, con la característica adicional y principal del conveniente uso de la inteligencia artificial, planteando como principal objetivo el desarrollo de una interfaz para la clasificación del lenguaje de señas colombiano (LSC) mediante el uso de redes neuronales convolucionales, desarrollando una

---

[8] ARISTOTELES. Política, Libro I. De la Sociedad Civil. De la Esclavitud. De la Propiedad. Del Poder Doméstico.

[9] ROZO MELO, Nancy. La Lengua de Señas Colombiana [En línea.]. Portal De Lenguas de Colombia: Diversidad y Contacto. Disponible en: https://lenguasdecolombia.caroycuervo.gov.co/contenido/Lenguas-de-senas-colombiana/introduccion.

[10] BOTINA MONSALVE, et al. Clasificación Automática de las Vocales en el Lenguaje de Señas Colombiano. Disponible en: https://repositorio.itm.edu.co/handle/20.500.12622/1038.

[11] TRIVIÑO LOPEZ, Iván. Sistema para el Aprendizaje del Lenguaje de Señas Colombiano usando Visión por Computador. Disponible en: https://ciencia.lasalle.edu.co/ing_automatizacion/159/.

interacción en tiempo real, que beneficia significativamente la comunicación entre una persona oyente y una con discapacidad auditiva.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Desarrollar e implementar una interfaz gráfica para facilitar la comunicación a personas con discapacidades auditivas, realizando detección y clasificación en tiempo real del alfabeto de lenguaje de señas colombiano mediante el uso de Python.

1.2 OBJETIVOS ESPECÍFICOS

- Crear un algoritmo de predicción en tiempo real del alfabeto del lenguaje de señas colombiano mediante el uso de herramientas de código abierto como OpenCV y Tensorflow.

- Diseñar una red neuronal convolucional que permita la clasificación de imágenes (Alfabeto del lenguaje de señas colombiano).

- Proporcionar una herramienta didáctica e intuitiva que mejore el proceso de comunicación para las personas que hacen uso del alfabeto de lenguaje de señas colombiano.

2. FUNDAMENTOS BÁSICOS

2.1 LENGUA DE SEÑAS COLOMBIANA

La lengua de señas colombiana es la lengua utilizada por la comunidad sorda de Colombia. Reconocida oficialmente en el año 1996, mediante la Ley 324. Esta lengua se caracteriza por ser visual y corporal, es decir la comunicación se establece con el cuerpo en un espacio determinado[12].

Su reconocimiento oficial se debe al desarrollo de proyectos que favorecieron la inclusión de las personas con discapacidades auditivas. Como resultado a investigaciones y de la labor de enseñanza de la lengua, la Federación Nacional de Sordos de Colombia (Fenascol) realizó dos cartillas denominadas “Lenguaje Manual Colombiano” y “Lengua de Señas Colombiana”, publicadas en 1993 y 1996 respectivamente.

En el año 2006 el Ministerio de Educación Nacional, el Instituto Caro y Cuervo y el Instituto Nacional para Sordos presentó el Diccionario Básico de la Lengua de Señas Colombiana como un compendio léxico que facilita los procesos comunicativos y educativos de la comunidad sorda del país, representando una fundamental contribución al estudio de la lingüística de la lengua de señas en Colombia[13].

De esta manera, se estableció el LSC, en el cual se encuentra el alfabeto conformado por 27 señas, 21 de las cuales son estáticas y 6 de ellas requieren de movimiento para su correcta ejecución. Por medio de estas señas es posible comunicar palabras, frases e ideas completas.

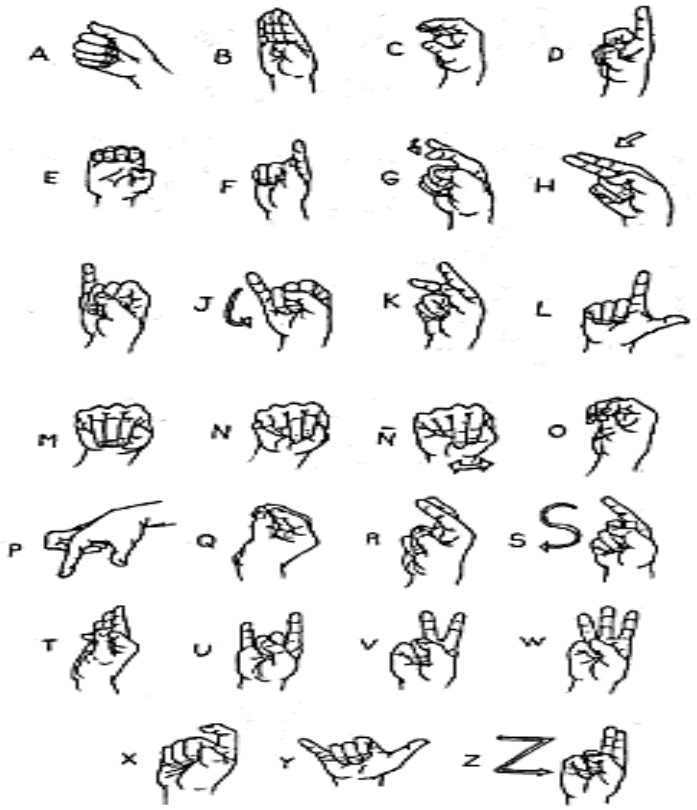
El LSC es el idioma oficial por medio del cual se comunica la comunidad colombiana con discapacidad auditiva, sin embargo, la población en condiciones auditivas normales en su mayoría desconoce su dominio. Es claro que el grupo de individuos que presentan dichas limitaciones auditivas hacen parte de una comunidad de minoría que no ha sido suficientemente resaltada para equilibrar ese factor diferencial. A pesar del desarrollo tecnológico y los constantes proyectos de inclusión en Colombia y el mundo, aún existe una brecha que imposibilita una comunicación fluida sin limitaciones entre personas con y sin discapacidad. Por esta

[12] ROZO MELO, Nancy. La Lengua de Señas Colombiana [En línea.]. Portal De Lenguas de Colombia: Diversidad y Contacto. Disponible en: https://lenguasdecolombia.caroycuervo.gov.co/contenido/Lenguas-de-senas-colombiana/introduccion.

[13] MINISTERIO DE EDUCACIÓN NACIONAL, INSTITUTO NACIONAL PARA SORDOS, INSOR. Diccionario Básico de la Lengua de Señas Colombiana por Instituto Nacional para Sordos INSOR [En línea.]. INSOR web site. Disponible en: http://www.insor.gov.co/descargar/diccionario_basico_completo.pdf.

razón, es imprescindible la educación de la sociedad en este lenguaje y como herramienta indispensable un traductor que favorezca y contribuya su uso.

Figura 1. Alfabeto del lenguaje de señas colombiano[14]



2.2 VISIÓN POR COMPUTADOR

La visión por computador es el campo de estudio enfocado en el proceso de obtención de información de imágenes y videos digitales por medio de un computador. Abarca todas las tareas realizadas por los sistemas de visión biológica, la detección de un estímulo visual, la comprensión de lo que se ve y la extracción de información compleja en una forma que se pueda usar en otros procesos[15].

La visión por computador funciona de manera muy similar a la visión humana, excepto que los humanos tienen una ventaja inicial. La vista humana tiene la primacía de toda una vida de contexto para aprender a diferenciar los objetos,

[14] MINISTERIO DE EDUCACIÓN NACIONAL, INSTITUTO NACIONAL PARA SORDOS, INSOR. Diccionario Básico de la Lengua de Señas Colombiana por Instituto Nacional para Sordos INSOR [En línea.]. INSOR web site. Disponible en: http://www.insor.gov.co/descargar/diccionario_basico_completo.pdf.

[15] DeepAI. Computer Vision: ¿What is Computer Vision? [En línea.]. Disponible en: https://deepai.org/machine-learning-glossary-and-terms/computer-vision.

identificar a que distancia se encuentran, si están en movimiento o si existe error en una imagen[16].

Por otro lado, la visión por computador necesita muchos datos para un óptimo funcionamiento, debe ejecutar un análisis de datos una y otra vez hasta que pueda realizar distinciones y reconocer correctamente imágenes.

En este campo de la inteligencia artificial es requerido dos tecnologías esenciales para lograr su desarrollo: un tipo de aprendizaje automático llamado aprendizaje profundo y una red neuronal convolucional (CNN), que son las tecnologías en las que se desarrolla este proyecto.

2.3 INTELIGENCIA ARTIFICIAL

El ser humano constantemente se cuestiona respecto al desarrollo de sus actividades cotidianas, fenómenos o sistemas que observa en el medio en el que se desenvuelve, sus relaciones interpersonales y en la forma en cómo se pueden resolver problemas mediante el razonamiento. Todas estas preguntas siempre llevan a una situación problema de la cual siempre se buscará encontrar solución, para casos en los que ya se encuentre una solución se indagará sobre la mejor de ellas tomando todos los recursos a disposición[17].

En la actualidad la sociedad se encuentra inmersa en un proceso orientado al desarrollo tecnológico a gran escala, resaltando disciplinas encargadas de crear sistemas que logren simular y/o replicar actividades realizadas por el ser humano como lo es la inteligencia artificial[18]. La inteligencia artificial (IA) se presenta como un campo de la computación encargado del estudio de conocimientos básicos respecto al funcionamiento de la inteligencia humana, la forma en como los seres humanos usan el lenguaje, comprensión de fenómenos, procesos de aprendizaje, formas de percepción y otras acciones para lograr tareas concretas mediante la adaptación[19].

[16] IBM. What is Computer Vision?. [En línea.]. IBM ANALÍTICA web site. Disponible en: https://www.ibm.com/topics/computer-vision.

[17] PONCE GALLEGOS, Julio Cesar, *et al.* Inteligencia Artificial. [s.l.]: Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn), 2014. 225 p.

[18] DOMÍNGUEZ, Néstor A. Ética y Ecoética para la inteligencia artificial. En: Boletín del Centro Naval 856 [En línea.]. 2021. Disponible en: https://www.centronaval.org.ar/boletin/BCN856/856-DOMINGUEZ.pdf

[19] PORCELLI, Adriana Margarita. La Inteligencia Artificial y la Robótica: sus Dilemas Sociales, Éticos y Jurídicos. Derecho glob. Estud. Sobre Derecho Justicia [En línea.]. 2020, vol.6, n.16, pp.49-105. Disponible en: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2448-51362020000300049&lng=es&nrm=iso

La inteligencia artificial se enfoca en automatizar las tareas intelectuales que normalmente realizan los seres humanos, pero al ser un campo en general abarca otros tipos de aprendizaje como lo son[20]:

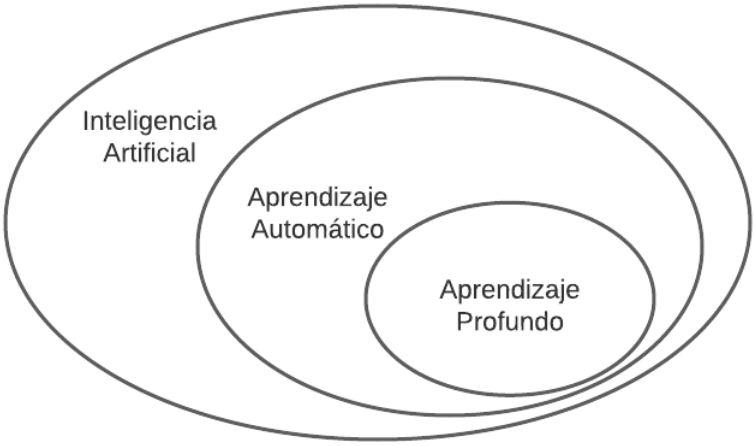
- Aprendizaje automático (Machine Learning).
- Aprendizaje profundo (Deep Learning).

2.3.1 APRENDIZAJE PROFUNDO

Como se observa en la Figura 2, el aprendizaje profundo o Deep Learning es un enfoque del aprendizaje automático. A diferencia de los modelos y/o algoritmos convencionales del aprendizaje automático, en el aprendizaje profundo los sistemas son capaces de mejorar su rendimiento al acceder a un volumen mayor de datos. El aprendizaje profundo es comúnmente usado en las siguientes aplicaciones:

- Reconocimiento de voz.
- Procesamiento de lenguaje natural.
- Tratamiento avanzado de imágenes.
- Reconocimiento facial.
- Clasificación de imágenes.

Figura 2. Inteligencia artificial, aprendizaje automático y aprendizaje profundo[20]



En el desarrollo de este proyecto se hace uso de las redes neuronales convolucionales para obtener un clasificador correspondiente a las señas estáticas del alfabeto de lengua de señas colombiano.

[20] CHOLLET, Francois. Deep learning with python. [s.l.]: Manning Publications Co, 2018. 361 p. ISBN 9781617294433.

2.4 PYTHON

Python es un lenguaje de programación multiplataforma, multiparadigma y de código abierto. Fue creado por Guido Van Rossum a principio de los 90 basándose en el lenguaje de programación ABC que implementó durante los 80. Python presenta características que destacan:

- Tiene una sintaxis simple.
- Alta legibilidad (Sangrado obligatorio).
- Entorno amigable de desarrollo (intérprete interactivo).
- Abstracciones de más alto nivel (mayor nivel de expresividad).
- Potente librería estándar y gran cantidad de módulos de terceros (actualmente son más de 100.000).
- Software libre y comunidad entusiasta.

En la actualidad Python es de los lenguajes más usados en el aprendizaje de la programación, planteado como un lenguaje de propósito general aplicado a desarrollo de aplicaciones web, ciencias de datos, ciencias de computación, inteligencia artificial, internet de las cosas, entre otras más[21].

2.5 OPENCV

OpenCV es una librería código abierto de visión artificial y machine learning, posee cerca de 2500 algoritmos que son capaces de:

- Identificar objetos, caras, colores.
- Clasificar acciones.
- Realizar seguimiento de objetos.
- Extraer modelos 3D.
- Encontrar imágenes similares.
- Importar archivos.
- Realizar captura en tiempo real.

La primera versión alfa de OpenCV aparece el mes de enero de 1999 desarrollada por Intel escrita en lenguaje C y C++, creada para ser multiplataforma (Disponible en sistemas operativos Linux, Mac OS, Windows). Su adaptación OpenCV-Python permite hacer uso de todas las características y/o comandos en el lenguaje de programación Python[22].

---

[21] CHALLENGER PÉREZ, Ivet; DÍAZ RICARDO, Yanet y BECERRA GARCÍA, Roberto Antonio. El lenguaje de programación Python. En: Ciencias Holguín [En línea.] 2014. vol. XX, no. 2. Disponible en: https://www.redalyc.org/articulo.oa?id=181531232001.

[22] ARÉVALO, V. M.; GONZÁLEZ, J. y AMBROSIO, G. La Librería ee Visión Artificial Opencv Aplicación a la Docencia e Investigación. En: MAPIR Research Group [en línea]. 2017. Disponible en: http://mapir.isa.uma.es/varevalo/drafts/arevalo2004lva1.pdf.

2.6 TKINTER

El paquete Tkinter es una adaptación de la biblioteca Tcl/Tk, es la interfaz por defecto de Python que corresponde a un conjunto de herramientas robusta e independiente de la plataforma para administrar ventanas y en general la interfaz gráfica de usuario.

Este marco de trabajo o framework proporciona a los usuarios de Python una forma sencilla de crear elementos GUI utilizando los widgets que se encuentran en el kit de herramientas Tk. Los widgets de Tk se pueden usar para construir botones, menús, campos de datos, etc. en una aplicación de Python. Una vez creados, estos elementos gráficos pueden asociarse o interactuar con características, funciones, métodos, datos o incluso otros widgets[23].

Tkinter maneja interfaces orientadas a objetos, es de fácil desarrollo para programadores que cuenta con experiencia en Python. Se determinó esta librería como fuente para el desarrollo de la interfaz gráfica debido a su adaptabilidad y la característica de proporcionar una plataforma de gráficos independiente de la arquitectura para las aplicaciones.

[23] ActiveState. What is Tkinter used for and How to Install this Python Framework? 21 de septiembre de 2021. Disponible en: https://www.activestate.com/resources/quick-reads/what-is-tkinter-used-for-and-how-to-install-it/.

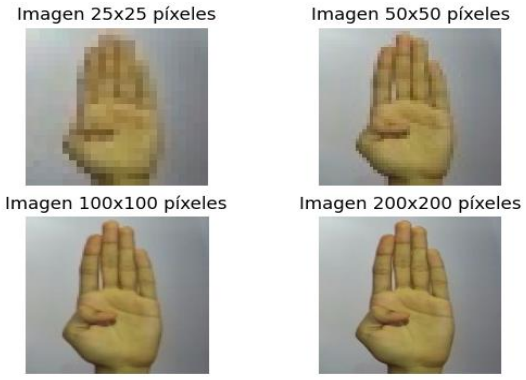
3. ADQUISICIÓN DE DATOS

En el proceso de formación de una imagen intervienen el objeto, fuente radiante y el sistema de formación (Sistema óptico, sensor y digitalizador). La imagen digital se puede representar por una matriz f de dimensiones $N \times M$ donde cada elemento corresponde a un píxel, de esta manera[24]:

$$f = \begin{bmatrix} f_{(1,1)} & \cdots & f_{(1,M)} \\ \vdots & \ddots & \vdots \\ f_{(N,1)} & \cdots & f_{(N,M)} \end{bmatrix} \qquad \text{Ecuación (1)}$$

El muestreo en el que se realiza la imagen digital (Matriz) juega un papel importante en la resolución que esta presenta y la cantidad de información disponible y a procesar, en la siguiente figura se ilustra la variación en la calidad de imagen a diferentes resoluciones:

Figura 3. Información disponible al aumento del muestreo

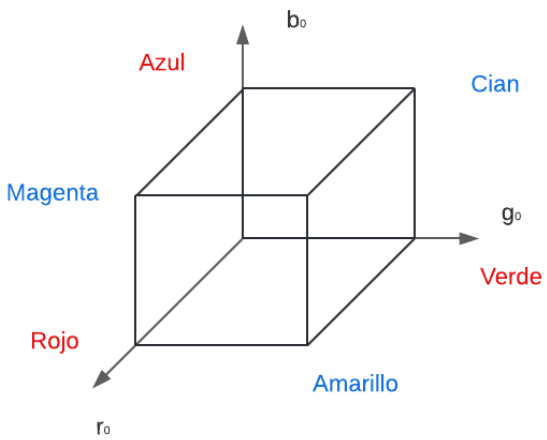


Para la adquisición de los datos se trabajó el espacio de color RGB, que se representa con un sistema cartesiano tridimensional y un cubo unitario (Figura 3). Cada eje corresponde a un color (rojo (R), verde (G), azul (B)), la ausencia de los colores resulta en negro, la presencia de todos los colores resulta en blanco y al superponer colores primarios se obtienen los colores secundarios (Cian, Magenta y Amarillo), este espacio de color presenta utilidad al momento de representar y/o visualizar imágenes mediante adición de colores[25].

[24] GONZÁLEZ MARCOS, Ana, et al. Técnicas y algoritmos básicos de visión artificial [en línea]. [s.l.]: Material didáctico. Ingenierías., 2006. Disponible en: https://publicaciones.unirioja.es/catalogo/online/VisionArtificial.pdf. ISBN 84-689-9345-X.

[25] ALEGRE GUTIÉRREZ, Enrique, et al. Procesamiento Digital de Imagen: Fundamentos y Prácticas Con Matlab [En línea]. Secretariado de Publicaciones y Medios Audiovisuales, 2003. Disponible en: https://www.researchgate.net/publication/229828279_Procesamiento_Digital_de_Imagenes_Fundamentos_y_Practicas_con_Matlab_Digital_image_processing_Fundamentals_and_practices_with_Matlab.

Figura 4. Representación del cubo unitario para el espacio de color RGB[25]



A partir de la imagen original con sus 3 canales respectivos en el espacio de color RGB, se puede obtener una imagen en escala de grises mediante la construcción de señales de luminancia y diferencia de color con el método de promedio ponderado[26]:

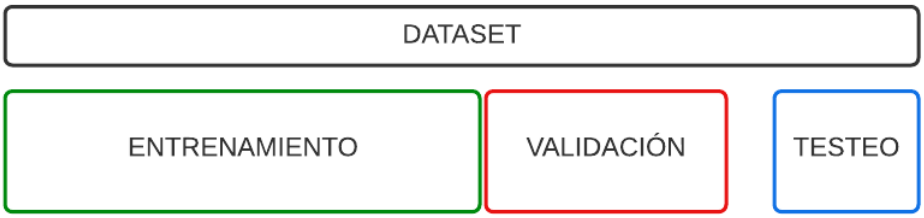
$$E'_Y = 0.299E'_R + 0.587E'_G + 0.114E'_B \qquad \text{Ecuación (2)}$$

Se consideró para el desarrollo de la red neuronal convolucional imágenes de entrada con solo 1 canal (imágenes en escala de gris) para reducir tiempos de compilación de dicha red neuronal. La resolución del recuadro de captura de imágenes es de 200×160 píxeles que posteriormente se reescalan a 50×50 píxeles al construir el set de datos. Se optó por este tamaño de imagen debido a que no presenta pérdida de información a causa de su resolución y el tiempo de compilación de la red neuronal convolucional disminuye a comparación de imágenes de 100×100 o 200×200 píxeles.

El conjunto de datos generado para el proyecto se presenta en 2 etapas, la primera que corresponde al entrenamiento de la red neuronal convolucional (comprende 2 conjuntos, entrenamiento y validación) y la segunda que corresponde al testeo de la red neuronal convolucional obtenida. La distribución se puede observar en la siguiente Figura:

[26] UNIÓN INTERNACIONAL DE TELECOMUNICACIONES. RECOMENDACIÓN UIT-R BT.601-7: Parámetros de Codificación de Televisión Digital para Estudios con Formatos de Imagen Normal 4:3 y de Pantalla Ancha 16:9 [En línea]. Disponible en: https://web.archive.org/web/20220119174709/https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-S.pdf

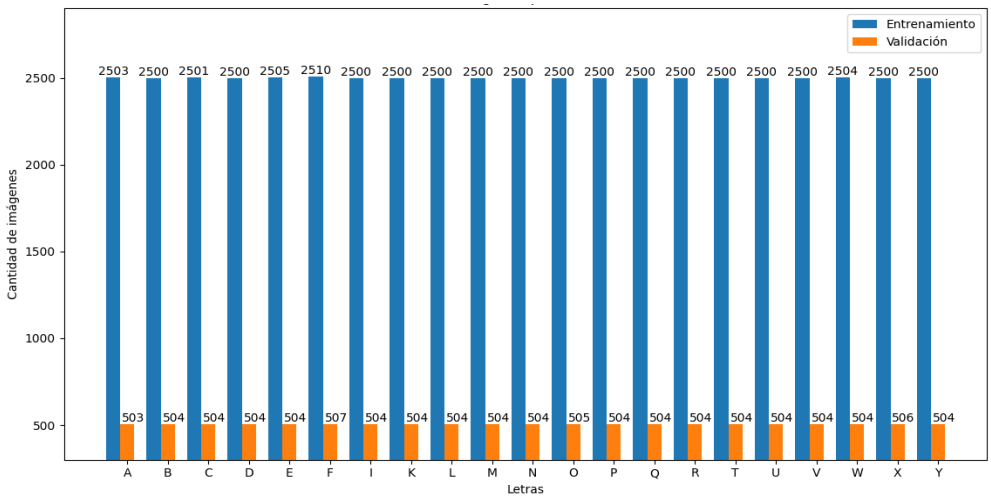
Figura 5. Estructura del conjunto de datos



Para entrenar el modelo de la red neuronal convolucional se usan los subconjuntos de entrenamiento y validación, por lo general se aplica una división de un conjunto general en 80% para entrenamiento y 20% para validación. El subconjunto restante que corresponde al testeo se usa para obtener predicciones del modelo generado en la etapa del entrenamiento y de esta manera generar métricas para evaluar el desempeño de la red neuronal convolucional[27].

Se generó un conjunto de datos balanceado en la cantidad de imágenes generadas por clase en cada uno de los subconjuntos del dataset. La cantidad de imágenes generadas para la primera etapa del proyecto se muestran a continuación:

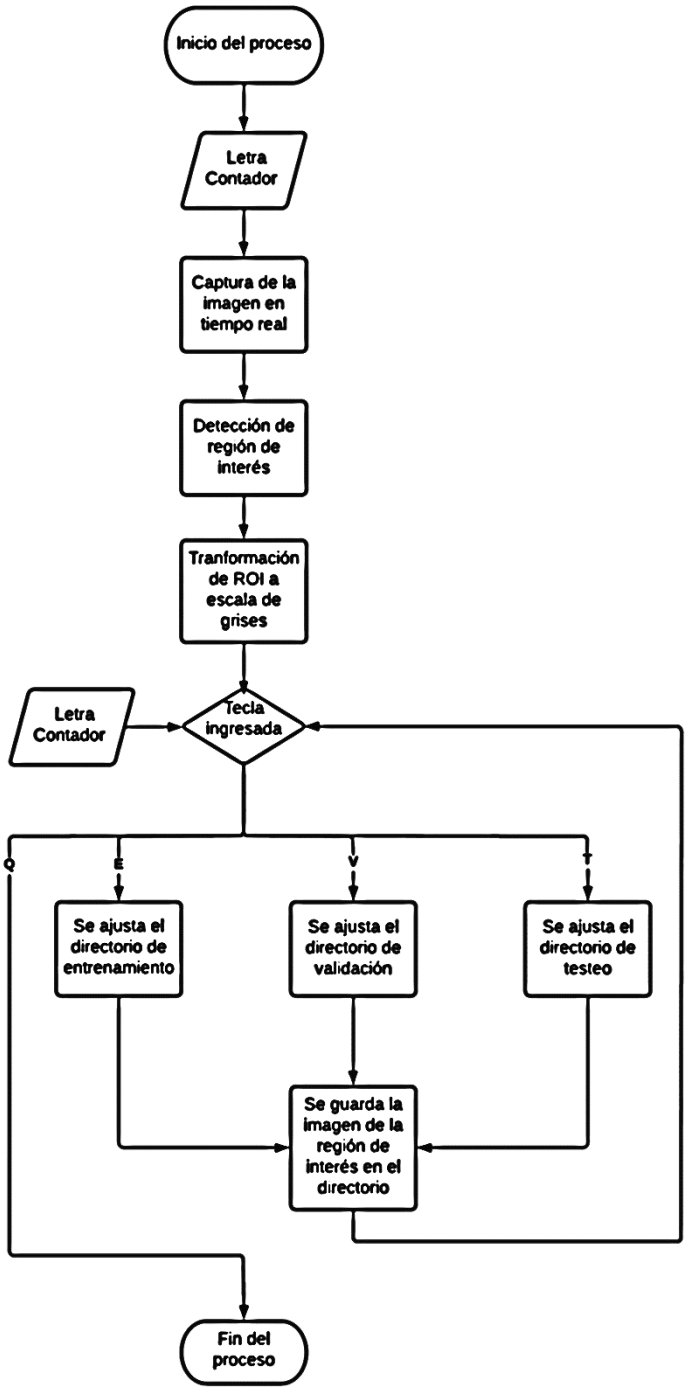
Figura 6. Cantidad de imágenes por clase en cada subconjunto



[27] BAGNATO, Juan Ignacio. Sets de Entrenamiento, Test y Validación [blog]. Aprende Machine Learning en español. Coruña, España. 3 de marzo de 2020. Disponible en: https://www.aprendemachinelearning.com/sets-de-entrenamiento-test-validacion-cruzada/

A continuación, se presenta el diagrama de flujo del ejecutable de Python usado para generar el conjunto de datos:

Figura 7. Diagrama de flujo del archivo generar de datos

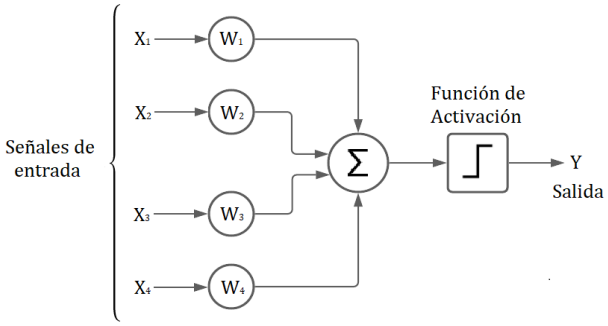


4. DISEÑO DE LA RED NEURONAL CONVOLUCIONAL

Una red neuronal convolucional es un tipo de red neuronal que es una mejora de un perceptrón multicapa (una red de propagación directa que consta de 2 o más capas de neuronas), que incluye el uso de capas convolucionales y submuestreo.[28]

Las redes neuronales convolucionales operan bajo un proceso de imitación respecto a la neurona biológica que para este caso se llamará perceptrón, el cual se puede representar gráficamente[29] de la siguiente manera:

Figura 8. Diagrama del perceptrón[29]



Sumado al perceptrón el uso de la convolución permite la extracción de características, La convolución es una operación lineal denotada como $s_{(t)}$ y que en su forma general es la operación entre 2 funciones:

$$s_{(t)} = (x * w)_{(t)} = \int x_{(a)} w_{(t-a)} da \qquad \text{Ecuación (3)}$$

La ecuación (3) expresada en la nomenclatura de las redes neuronales convolucionales corresponde a:

- $x_{(t)}$ la entrada.
- $w_{(t)}$ el kernel o filtro.
- $s_{(t)}$ la salida producida por la convolución.

[28] GARCÍA SÁNCHEZ, Eugenio. Introducción a las redes neuronales de convolución. Aplicación a la visión por ordenador. Trabajo de grado. [s.l.]: Universidad de Zaragoza, 2019.
[29] TORRES SOLER, Luis. El Perceptrón Redes Neuronales Artificiales [En línea] 20 de enero de 2020. Repositorio Universidad Nacional. Disponible en: https://disi.unal.edu.co/~lctorress/RedNeu/LiRna004.pdf

El proceso de convolución[30] entre una imagen ($x_{(t)}$) y un filtro ($w_{(t)}$) de 3×3 se expresa mediante la siguiente manera:

$$s_{(t)} = \sum_{i=0}^{2} \sum_{j=0}^{2} w_{(i,j)}\, x_{(a-i,b-j)} \qquad \text{Ecuación (4)}$$

Aplicando un filtro de 3×3 sobre una matriz de 6×6 se obtiene en:

$$s_{(1,1)} = \Big(3 * (-1) + (0 * 0) + (1 * 1) + \big(1 * (-1)\big) + (5 * 0) + (8 * 1) + \big(2 * (-1)\big) + (7 * 0) + (2 * 1)\Big) = 5$$

Figura 9. s(1,1) para convolución de imagen con kernel

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

IMAGEN

*

-1	0	1
-1	0	1
-1	0	1

KERNEL

=

5			

SALIDA

Realizando el proceso iterativo de convolución se obtiene:

Figura 10. Resultado de convolución entre imagen y kernel

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

IMAGEN

*

-1	0	1
-1	0	1
-1	0	1

KERNEL

=

5	4	0	-8
10	2	-2	-3
0	2	4	7
3	2	3	16

SALIDA

Entendiendo el proceso base que realizan las redes neuronales convolucionales, se inicia el proceso del desarrollo de la red neuronal convolucional. Para esto se hace uso de frameworks como TensorFlow y Keras, partiendo de la implementación de un generador de imágenes para aplicar la técnica de aumento de datos con el fin de abordar problemáticas como el sobreajuste y escasez de datos[31].

[30] SILVA, Sarahí y FREIRE, Estefanía. Intro a las redes neuronales convolucionales. Bootcamp AI. (23, noviembre, 2019). Disponible en: https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8.

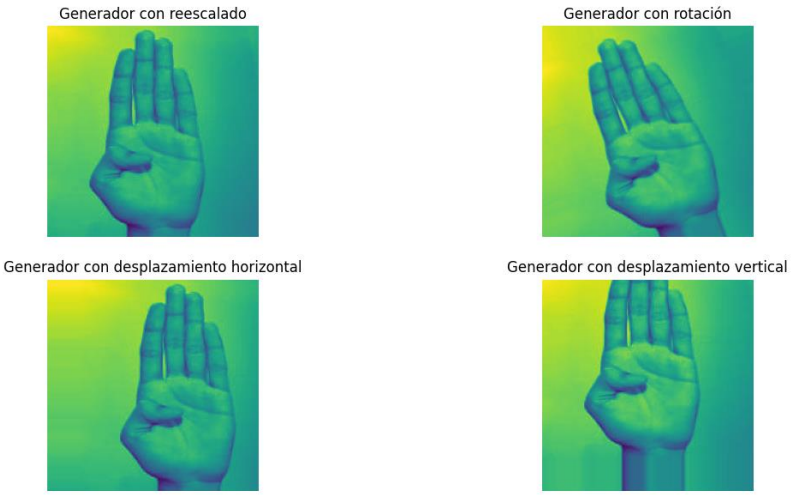
[31] UTRERA BURGAL, Jesús. Tratamiento de imágenes usando ImageDataGenerator en Keras [blog]. Technical thoughts, stories and ideas. 2 de agosto de 2019. Disponible en: https://enmilocalfunciona.io/tratamiento-de-imagenes-usando-imagedatagenerator-en-keras/

El aumento de datos consiste en incrementar la cantidad de datos (en este caso las señas con las que se entrenará la red neuronal) a partir de unos datos ya presentes que serán transformados. El aumento de datos aplicado mediante ImageDataGenerator se aplicó a los siguientes parámetros[32]:

- rescale: El reescalado se aplica con la finalidad de disminuir el valor de las imágenes originales en el rango de $0 - 255$ al rango de $0 - 1$, aplicando un factor de reescalado igual a $1/255$.
- width_shift_range: Desplazamientos horizontales, este parámetro indica el porcentaje o la cantidad de pixeles que se desplazará la imagen horizontalmente. Para el caso de porcentajes, se debe especificar un valor ≤ 1 donde 0 representará 0% de desplazamiento y 1 representará hasta un máximo de 100% de desplazamiento ya sea hacía la izquierda o derecha. Para el caso de pixeles, se debe especificar la cantidad máxima de pixeles a desplazar ya sea hacía la izquierda o derecha.
- height_shift_range: Desplazamientos verticales, este parámetro al igual que width_shift_range indica el porcentaje o la cantidad de pixeles que se desplazará la imagen, pero verticalmente. Para el caso de porcentajes, se debe especificar un valor ≤ 1 donde 0 representará 0% de desplazamiento y 1 representará hasta un máximo de 100% de desplazamiento ya sea hacía arriba o abajo. Para el caso de pixeles, se debe especificar la cantidad máxima de pixeles a desplazar ya sea hacía arriba o abajo.

Aplicando individualmente cada parámetro del generador se obtiene:

Figura 11. Generador de aumento de datos



[32] TF.KERAS.PREPROCESSING.IMAGE.IMAGEDATAGENERATOR. TensorFlow [página web]. Disponible en: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator.

Con el generador de aumento de datos para el conjunto de entrenamiento se leen las imágenes del directorio con un total de 52523 imágenes en 21 clases, el generador de datos en el conjunto de validación solo presenta el parámetro de rescale y se poseen en total 10589 imágenes en el directorio.

Con los datos obtenidos por el generador se definen los hiperparámetros que tendrá la red neuronal convolucional. Los hiperparámetros corresponden a valores y/o configuraciones establecidas para el proceso de entrenamiento[33]:

- Número de épocas: Es la cantidad de ciclos o número de veces que se ejecutan los algoritmos de forwardpropagation y backpropagation. Dependiendo de su valor, la red neuronal convolucional puede actualizar más o menos veces el valor de los pesos.
- Tasa de aprendizaje: Es el porcentaje de cambio con el que se actualizan los pesos en cada iteración realizada por la red neuronal, este se encarga de regular la velocidad en la que el modelo optimiza la función de pérdida[34].
- Optimizador: El optimizador permite ajustar los pesos generados en la red neuronal mediante los valores obtenidos en la función de pérdida[35].
- Función de activación: Las funciones de activación se encargan de retornar un valor de salida en un rango establecido ($(0,1)$ o $(-1,1)$) a partir de un valor de entrada[36].
  A. Función de activación RELU (Rectified Lineal Unit): Esta función de activación permite obtener valores positivos a la salida conservando los valores positivos en la entrada e igualando a cero los valores negativos en la entrada.

$$f_{(x)} \begin{cases} 0 \rightarrow x < 0 \\ x \rightarrow x \geq 0 \end{cases} \qquad \text{Ecuación (5)}$$

  B. Función de activación Softmax (Normalized exponential function): Esta función de activación permite obtener en la salida una representación de probabilidades, donde la sumatoria de todas las salidas debe corresponder

[33] ANALYTICS LANE. ¿Cuál es la Diferencia entre Parámetro e Hiperparámetro? [En línea] diciembre 16 de 209. Recuperado el 26 de marzo de 2022 de ANALYTICS LANE web site: https://www.analyticslane.com/2019/12/16/cual-es-la-diferencia-entre-parametro-e-hiperparametro/#:~:text=Los%20hiperpar%C3%A1metros%20de%20un%20modelo,por%20el%20cient%C3%ADfico%20de%20datos.

[34] VELO FUENTES, Edward Joseph. Introducción a los métodos Deep Learning basados en Redes Neuronales [En línea]. Trabajo Fin de Máster. [s.l.]: Universidad de Coruña, 2020. Disponible en: http://eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_1654.pdf.

[35] LOPEZ BRIEGA, Raul E. Introducción al Deep Learning [blog]. Matemáticas, análisis de datos y python. 13 de junio de 2017. Disponible en: https://relopezbriega.github.io/blog/2017/06/13/introduccion-al-deep-learning/

[36] CALVO, Diego. Función de activación – Redes neuronales [blog]. Diego Calvo. 7 de diciembre de 2018. Disponible en: https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/

a 1. Esta función de activación se usa en capas de salida que contienen la cantidad de neuronas correspondientes a la cantidad de clases.

$$f_{(z)_j} = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \qquad \text{Ecuación (6)}$$

Con los hiperparámetros definidos en la siguiente tabla se entrena la red neuronal convolucional con el conjunto de datos descritos en el capítulo 3.

Tabla 1 Hiperparámetros

Hiperparámetro	Valor
Número de capas convolución	5
Número de capas maxpooling	3
Número de capas flatten	1
Número de capas densas	2
Número de capas de dropout	1
Número de clases	21
Dimensiones de imagen de entrada	50x50
Número de épocas	30
Batch size	32
Tasa de aprendizaje	0.0005
Número de pasos en entrenamiento	1641
Número de pasos en validación	330
Tamaño de filtros de convolución	(4,4)-(3,3)-(2,2)
Tamaño de filtros de maxpooling	(2,2)
Optimizador	Adam
Cantidad de filtros de convolución	32 - 64

El entrenamiento se completa en un total de 56 minutos (Entre 95 y 120 segundos por época), el historial de las métricas obtenidas por iteración se muestra a continuación:

Tabla 2 Métricas generadas por el compilador

Precisión	Precisión de Validación	Pérdida	Pérdida de Validación
0,74464196	0,88001895	0,798173904	0,382220119
0,952982426	0,973863661	0,149043471	0,078163885
0,973671675	0,962405324	0,083311588	0,170352921
0,979653656	0,978787899	0,062613167	0,091233708
0,984054387	0,980113626	0,049434904	0,067923911
0,986759663	0,991287887	0,040513799	0,032048032
0,988969564	0,98939395	0,033699073	0,049360454
0,990836501	0,977083325	0,029224329	0,070468269
0,99182719	0,965151489	0,025970442	0,104407385
0,993637025	0,984943211	0,021206547	0,054955155
0,99350363	0,987026513	0,021299498	0,037437584
0,994703829	0,974337101	0,017502794	0,082698874
0,994741976	0,996022701	0,016163014	0,012254046
0,995027721	0,988352299	0,015421477	0,037031606
0,995865941	0,971306801	0,013590627	0,080084875
0,995618284	0,993371189	0,014300671	0,022129867
0,996380329	0,992613614	0,011459325	0,030743925
0,996685147	0,997064412	0,011794321	0,010002479
0,996875644	0,971875012	0,00991751	0,117075555
0,99664706	0,992045462	0,010567611	0,030129014
0,997028053	0,990151525	0,009905551	0,040683854
0,996913731	0,988731086	0,010236982	0,036500514
0,997123301	0,993276536	0,009122201	0,020242566
0,997237623	0,981439412	0,008784134	0,0577459
0,997999668	0,992803037	0,006350072	0,030312521
0,996970892	0,9969697	0,009578946	0,011562496
0,997752011	0,991003811	0,006829314	0,032311518
0,997561514	0,971969724	0,007874855	0,148606092
0,997713923	0,996401489	0,007778627	0,012424275
0,997732937	0,990056813	0,008159977	0,044452794

Con el historial de datos obtenidos en el entrenamiento de la red neuronal se grafica el comportamiento de la precisión y pérdida para el conjunto de entrenamiento y validación, los resultados se muestran a continuación:

Figura 12. Accuracy

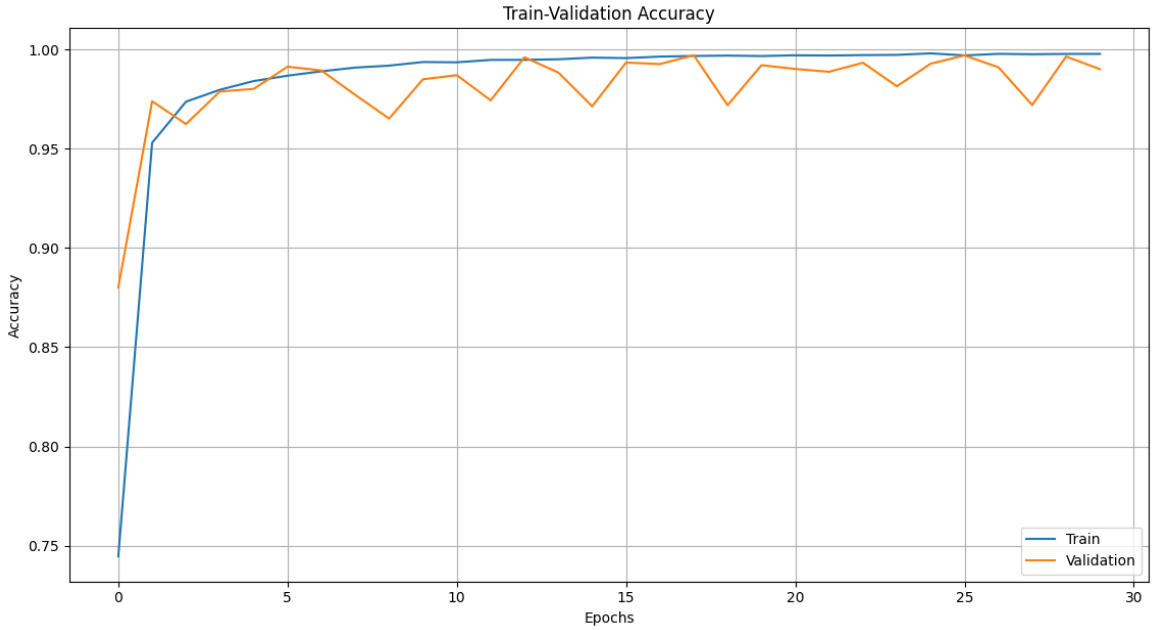
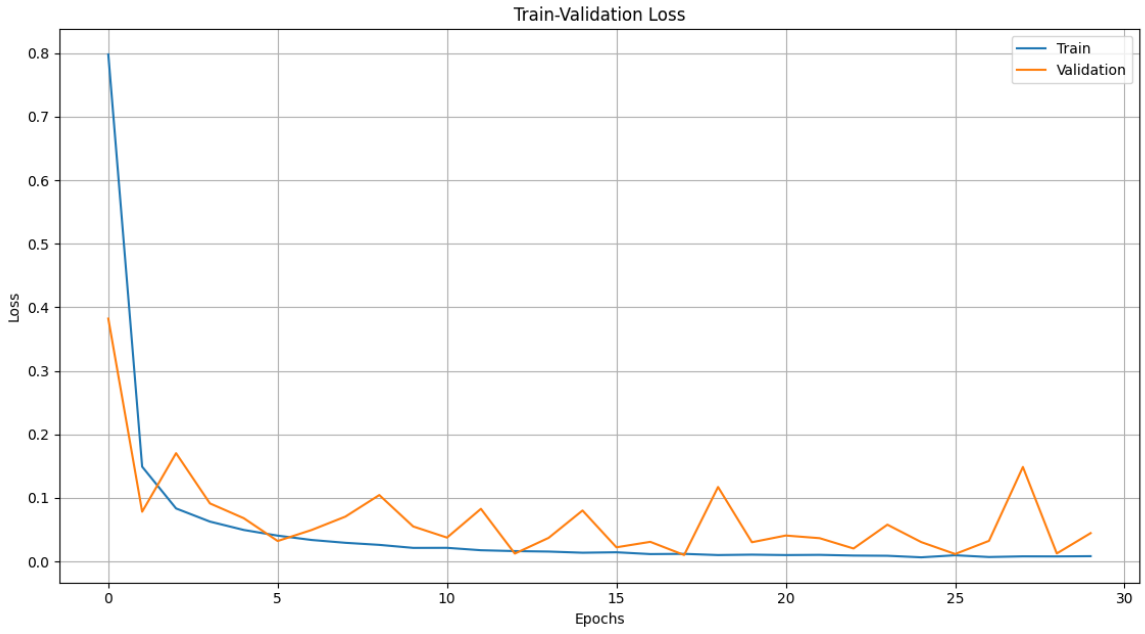
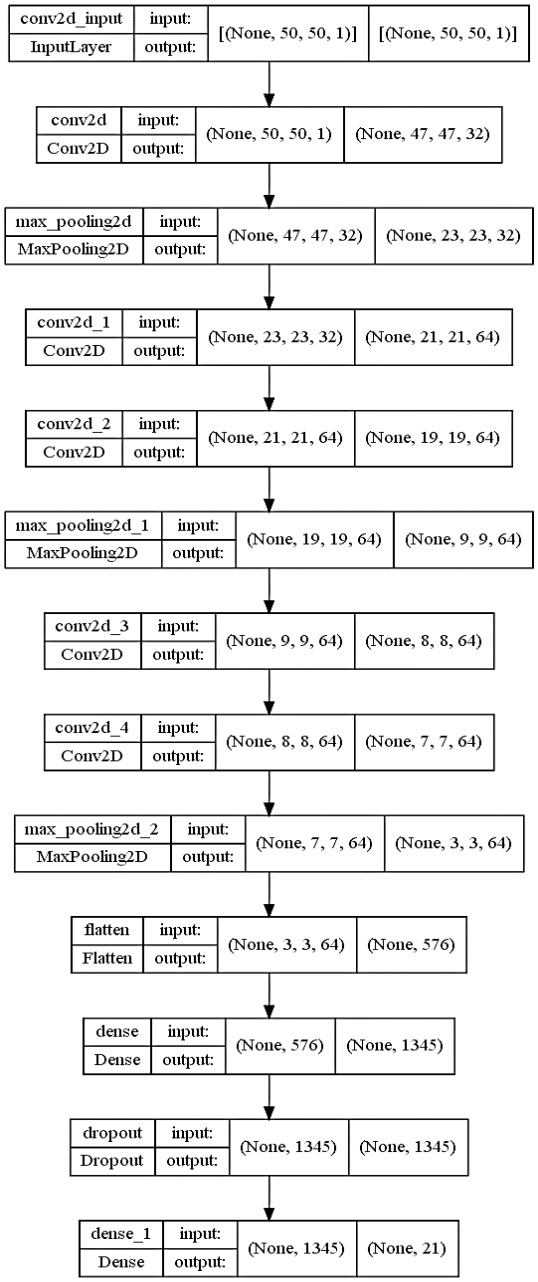


Figura 13. Loss



La composición de la red neuronal convolucional usada en el entrenamiento se muestra en la figura (14), donde es posible relacionar los valores de entrada y salida de cada capa de la red neuronal convolucional.

Figura 14. Composición de la red neuronal



5. RECONOCIMIENTO DE LA MANO

5.1 MEDIAPIPE

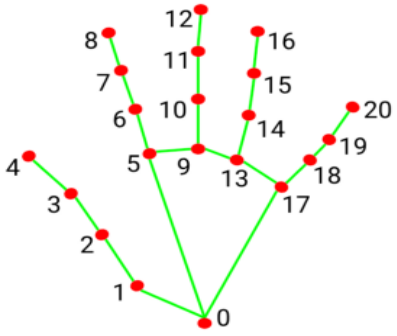
MediaPipe es una solución de detección de objetos en tiempo real. Permite la detección de rostros, detección holística, detección y seguimiento de objetos, reconocimiento de manos, conteo de dedos, entre otras aplicaciones.

Uno de los recursos que ofrece MediaPipe, es MediaPipe Hands, que es una herramienta para el seguimiento de manos y dedos de alta fidelidad, que favorece la percepción de la forma y el movimiento de las manos, representando una significativa contribución al campo de visión por computador.

MediaPipe Hands emplea el aprendizaje automático (ML) para inferir 21 puntos de referencia 3D de una mano a partir de un solo cuadro en donde se han obtenido múltiples modelos que trabajan juntos. El sistema está basado en un rastreo de la palma y los dedos de la mano para luego detectar 21 puntos clave 3D de un solo fotograma, en una imagen cinematográfica considerada aisladamente. El proceso se divide en un detector de palmas que recorta la forma de la mano del fotograma, un modelo que detecta puntos de referencia en 3D de la imagen recortada y un detector de gestos que clasifica los puntos claves configurados previamente en un set de gestos[37].

MediaPipe se ofrece para trabajar en lenguajes como JavaScript, C++, y Python. En Python, se encuentra como un paquete preconstruido y se ejecuta con comandos de rápido entendimiento.

Figura 15. Puntos de referencia de la mano[37]



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

5.2 DETECCIÓN DE LA MANO

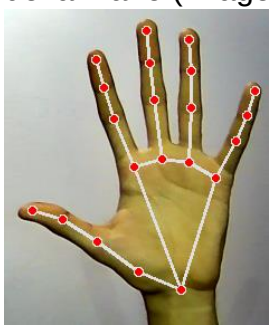
Por medio de la librería MediaPipe, se hace uso del módulo Hands el cual contiene la clase Hands que se utiliza para realizar la detección de puntos de referencia de

[37] MEDIAPIPE HANDS. Mediapipe [página web]. Disponible en: https://google.github.io/mediapipe/solutions/hands.html

manos en una imagen. La clase Hands se instancia creando un objeto “manos” que permite procesar la mano. El constructor de la clase recibe diferentes parámetros, el primer parámetro “STATIC_IMAGE_MODE” se establece en False, ya que trata las imágenes como una trasmisión de video, reduciendo la latencia y es ideal para procesar cuadros de video, el segundo parámetro MAX_NUM_HANDS corresponde al número máximo de manos que pueden ser detectadas, presenta un valor por defecto igual a 2, el tercer parámetro MODEL_COMPLEXITY es la complejidad del modelo de puntos de referencia de la mano, entre 0 o 1 con un valor predeterminado de 1. Los dos parámetros finales MIN_DETECTION_CONFIDENCE y MIN_TRACKING_CONFIDENCE, ambos por defecto 0.5 entre un rango de 0 a 1 que corresponden al valor mínimo de confianza del modelo de detección manual y el valor de confianza mínimo del modelo de seguimiento de puntos de referencia. Se mantiene el valor por defecto del MIN_TRACKING_CONFIDENCE debido a que al aumentar el valor aumenta la solidez de la solución, pero al mismo tiempo, aumenta la latencia.

Con fines ilustrativos se incluye el módulo drawing_utils que incluye funciones auxiliares útiles para dibujar detecciones y los puntos de referencia sobre la imagen. Este módulo se empleó únicamente en el reconocimiento de la imagen para realizar el dibujo de los 21 puntos detectados, ya que para la presentación final de la interfaz no se usó, para evitar mayor carga visual y de procesamiento. Adicionalmente, se realiza la lectura de la cámara, la corrección de color del Frame, se crea una copia del Frame para el procesamiento de las manos y se realiza la detección de puntos de la mano con la llamada del método “process” del objeto “manos” que devuelve como salida un objeto NamedTuple el cual contiene una colección de puntos de referencia de las manos que se encuentran en la imagen y una colección de las manos detectadas. Si se detecta una mano, se realiza una búsqueda de la mano dentro de la colección de manos detectadas multi_hand_landmarks, en donde cada mano se representa como una lista de 21 puntos de referencia de la mano.

Figura 16. Puntos de referencia de la mano (Imagen real)



5.3 EXTRACCIÓN DE COORDENADAS

Para la extracción de coordenadas de la mano, en donde se representa la señal de interés a traducir, fue requerido realizar una función que permitiera obtener los puntos extremos que incluyera toda la mano. La función recibe como parámetro de entrada una lista de listas, donde cada una de las listas contiene la identificación de cada uno de los 21 puntos de la mano y su respectiva coordenada en el eje X y Y. Al principio de la función se definen los puntos iniciales izquierda, derecha, arriba, y abajo que corresponden a los extremos correspondientes al Frame en donde se ubica la cámara.

Esta función recorre la lista de listas, realizando comparaciones por medio de condicionales. El eje X se encuentra en la posición 1 de cada sublista y el eje Y, en la posición 2, de esta manera, los puntos máximos del eje X hacen la comparación con los puntos iniciales derecha e izquierda y los puntos del eje Y lo realizan con los puntos arriba y abajo. Así, se va almacenando el punto más externo con relación a la ubicación de la mano.

La función retorna 4 puntos los cuales permiten crear un rectángulo que contornea toda la seña, por medio del método cv2.rectangle el cual recibe como parámetros la imagen que en este caso es el Frame donde se ubica la cámara, punto de inicio y punto final, donde el punto de inicio corresponde a izquierda, arriba y el punto final derecha, abajo. A cada punto se le adiciona 50 pixeles para tener una mejor cobertura de la mano.

Figura 17. Recuadro generado en extracción de coordenadas

5.4 REGIÓN DE LA MANO

Para obtener la región de la mano, es necesario obtener una copia del Fotograma de la cámara y una vez dibujado el rectángulo que contiene la mano con la seña, con dichos puntos se extrae la región que posee la mano. Dicha región corresponde a una imagen, la cual se ajusta al tamaño de las imágenes con las que fue entrenada la red neuronal. En este proyecto cada imagen de entrenamiento tiene un tamaño igual a 50×50 pixeles.

A continuación, se presenta el diagrama de flujo del ejecutable de Python usado para la detección de la mano:

Figura 18. Diagrama de flujo para el reconocimiento de la mano

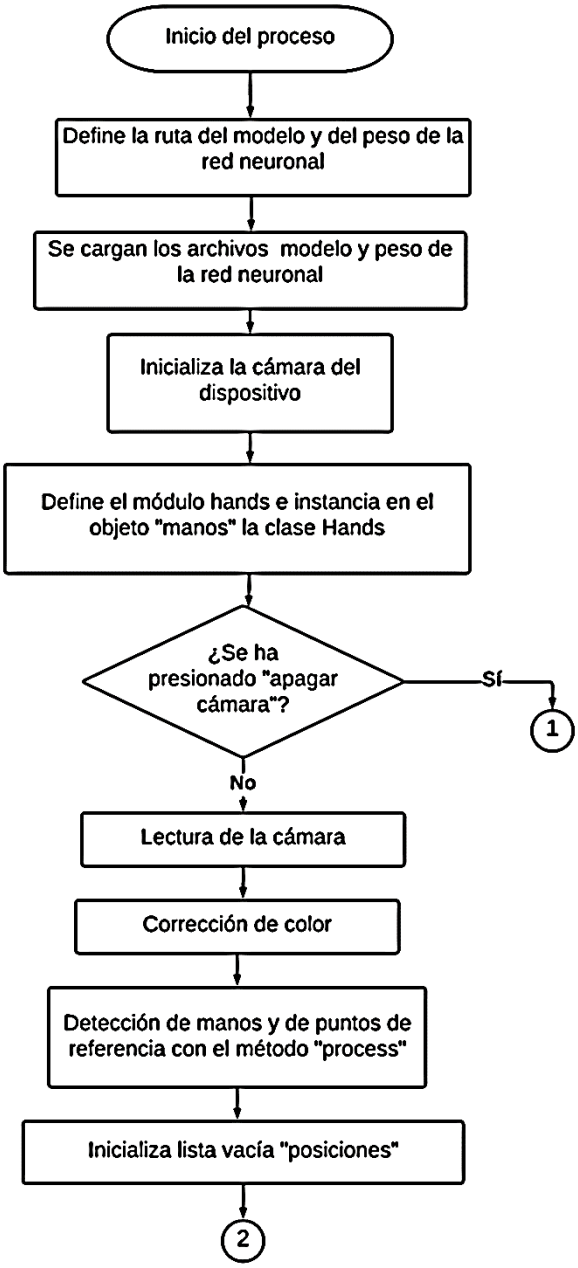


Figura 18. Diagrama de flujo para el reconocimiento de la mano

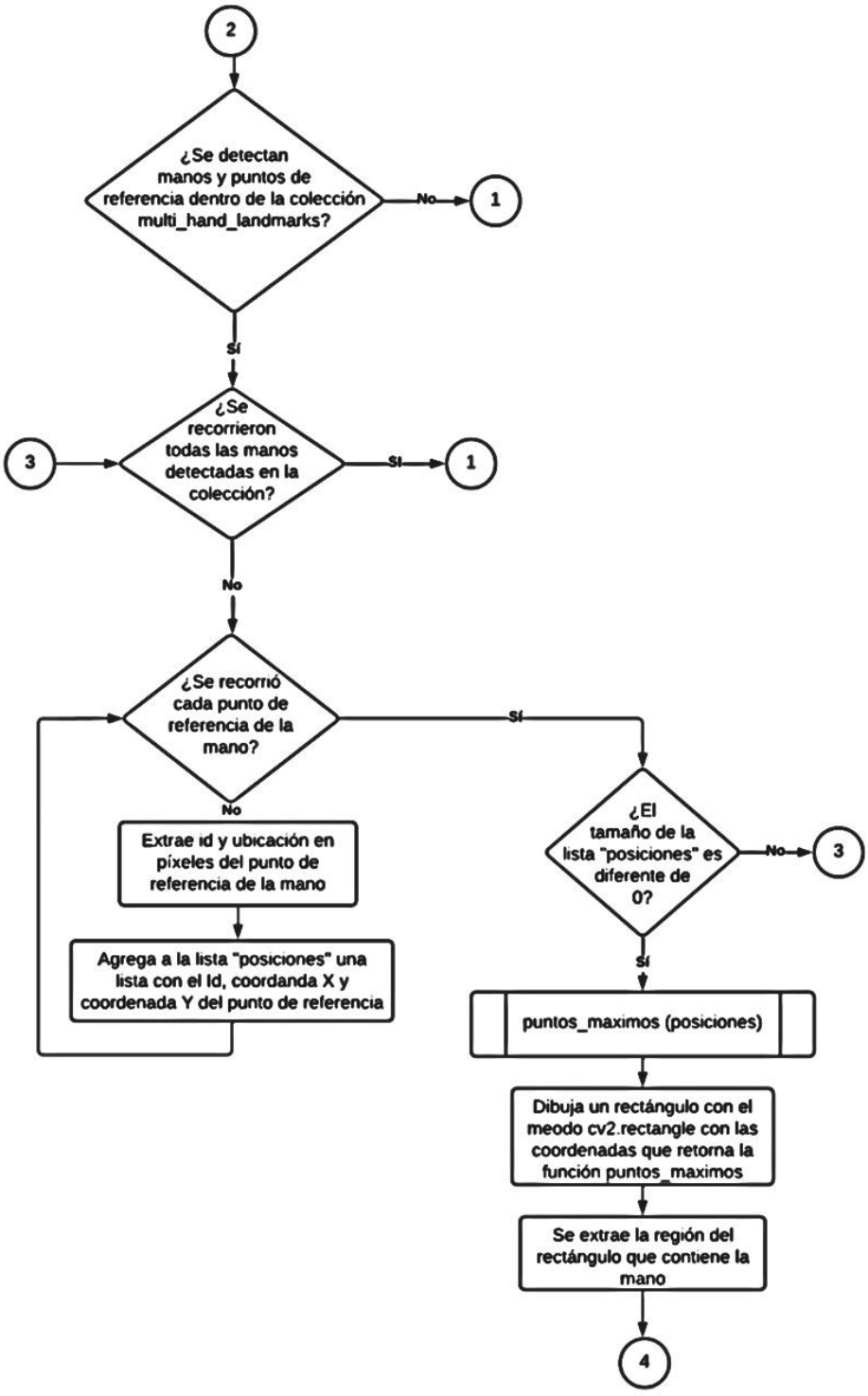


Figura 18. Diagrama de flujo para el reconocimiento de la mano

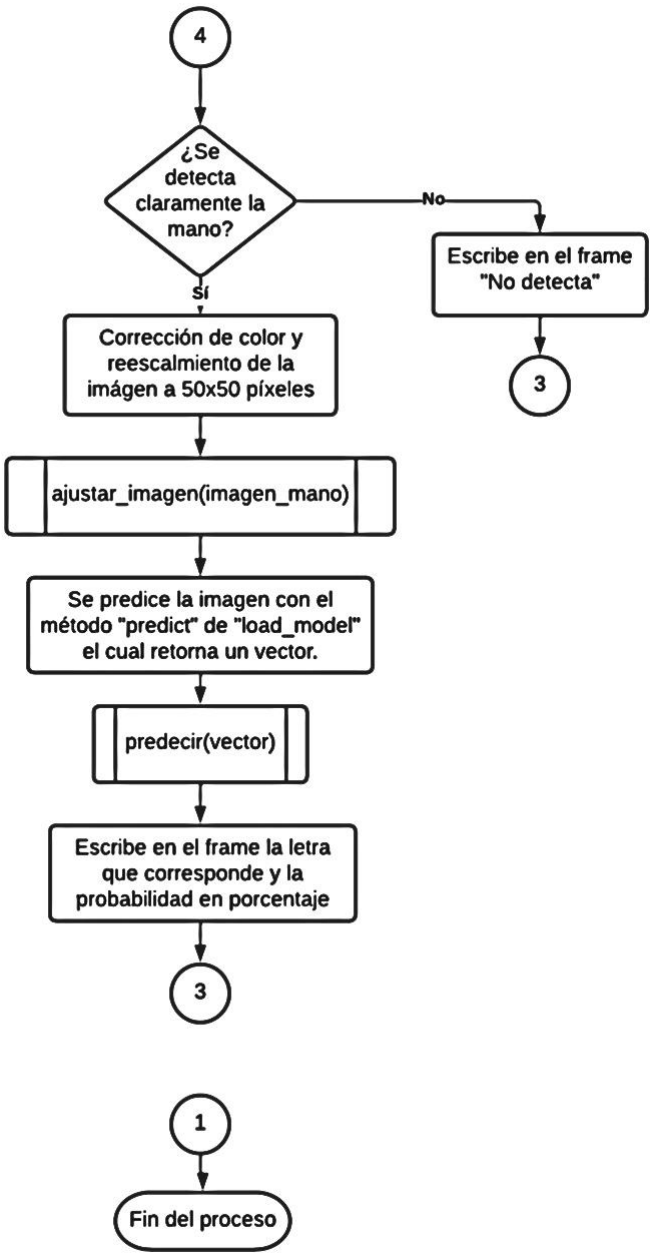
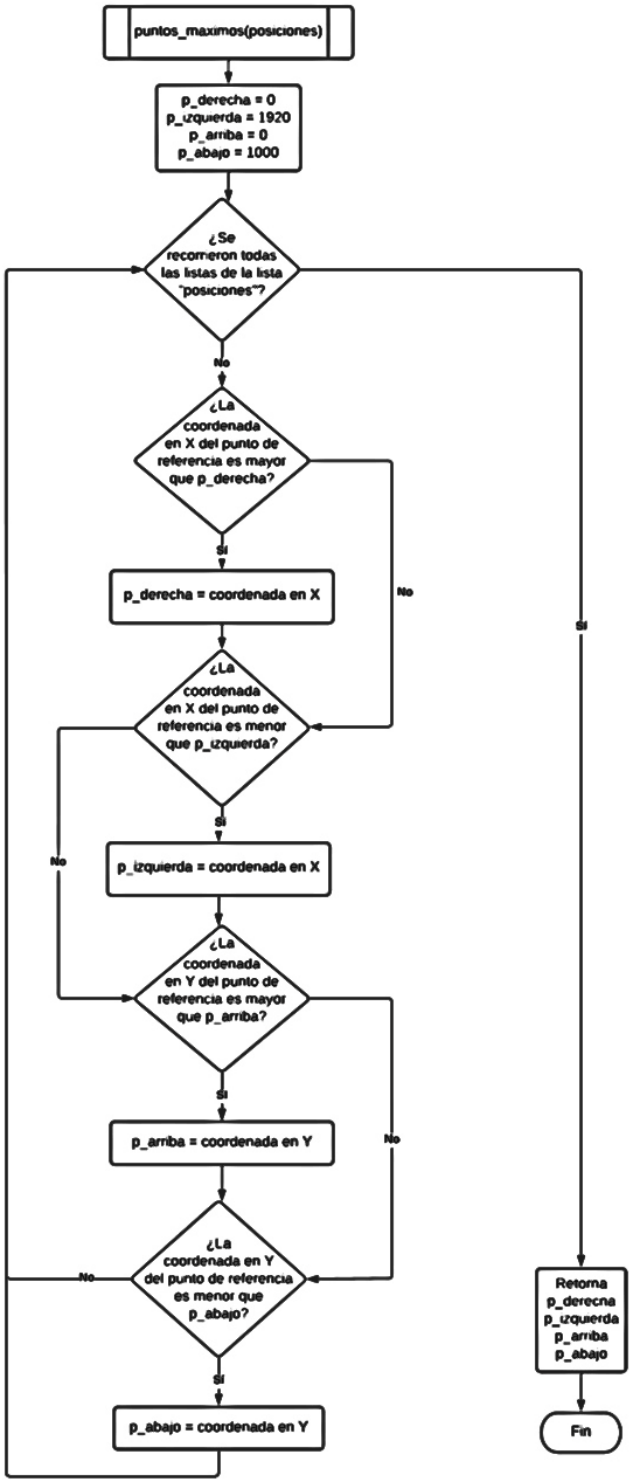


Figura 18. Diagrama de flujo para el reconocimiento de la mano



6. INTERFAZ (GUÍA DE USO)

La interfaz se desarrolló con la librería Tkinter, haciendo uso de programación orientada a objetos con el fin de mejorar el entendimiento del código y su reutilización para futuras mejoras o proyectos relacionados.

Para ello fue requerido la creación de diferentes widgets, en donde cada widget es representado como un objeto de Python. Los widgets empleados en la Interfaz correspondieron a Frame, Label y Button.

La interfaz en principio se conforma de dos Frames. El primero concierne a un Frame correspondiente a la barra de título en donde se encuentran los iconos de minimizar y cerrar la aplicación. En este espacio se eliminó la barra que ofrece Tkinter por defecto y se diseñó una bajo los requerimientos considerados como necesarios y visualmente correctos. Como segundo Frame, se creó un Frame Principal que abarca los puntos 1,2 y 3 de la figura (19-a) en donde se aborda el contenido completo de la interfaz, este Frame contiene cuatro columnas y dos filas en las cuales, se crean tres nuevos Frames.

Dentro del primer Frame, se posiciona en principio la portada del proyecto, en donde se presenta el título del proyecto, nombre de los autores y la universidad a la que pertenecen, se puede visualizar en el punto 1 de la figura (19-a). Una vez se presiona uno de los botones que está ubicado en el segundo Frame, se enciende la cámara que se posiciona entonces, en el primer Frame ocultando la portada anteriormente expuesta, lo que se presenta en el punto 1 figura (19-b).

En el segundo Frame, punto 2 figura (19-a), se observan cuatro botones, los tres primeros son de carácter informativo con el fin de que el usuario tenga acceso al objetivo del desarrollo de la interfaz, de su funcionamiento y el cómo interactuar adecuadamente. El último botón es el ya antes mencionado para encender la cámara.

Cabe resaltar, que al igual que el primer Frame, este Frame también cambia al momento de encender la cámara, lo cual se puede observar en el punto 2 figura (19-b). En este Frame, los botones desaparecen y se presenta un espacio en donde se podrá observar en tamaño grande la letra que se está detectando y debajo de ella, el correspondiente botón para apagar la cámara si el usuario lo requiere, retornando así, al estado inicial de la Interfaz.

Finalmente, en la parte inferior, ocupando las cuatro columnas de la segunda fila del Frame principal, se encuentra el alfabeto, ubicado en la figura (19-a) y (19-b), el punto 3. Cada una de las letras corresponde a un botón que al ser presionado muestra su respectiva traducción al lenguaje de señas. Adicionalmente, se dispone de dos flechas, una en sentido izquierdo y la otra en sentido derecho, que permite al usuario desplazarse por todo el alfabeto.

Sobre la letra seleccionada se puede observar un punto rojo, que es empleado como indicador sobre cuál es la seña que se está mostrando.

Es de mencionar, que esta última sesión cuenta únicamente con las letras que en su traducción a lenguaje de señas corresponde a letras estáticas, debido a que son estas letras las abordadas en este proyecto.

La lógica que se encuentra detrás de esta interfaz refiere una serie de métodos que son accionados al presionar los botones. Dentro de este código también se encuentra la lógica correspondiente a la detección de la mano gracias al paquete MediaPipe que se ejecuta una vez inicializada la cámara, junto con el modelo entrenado para la predicción de las señas.

Figura 19. De arriba a abajo, Interfaz con cámara apagada (a) - Interfaz con cámara encendida (b)

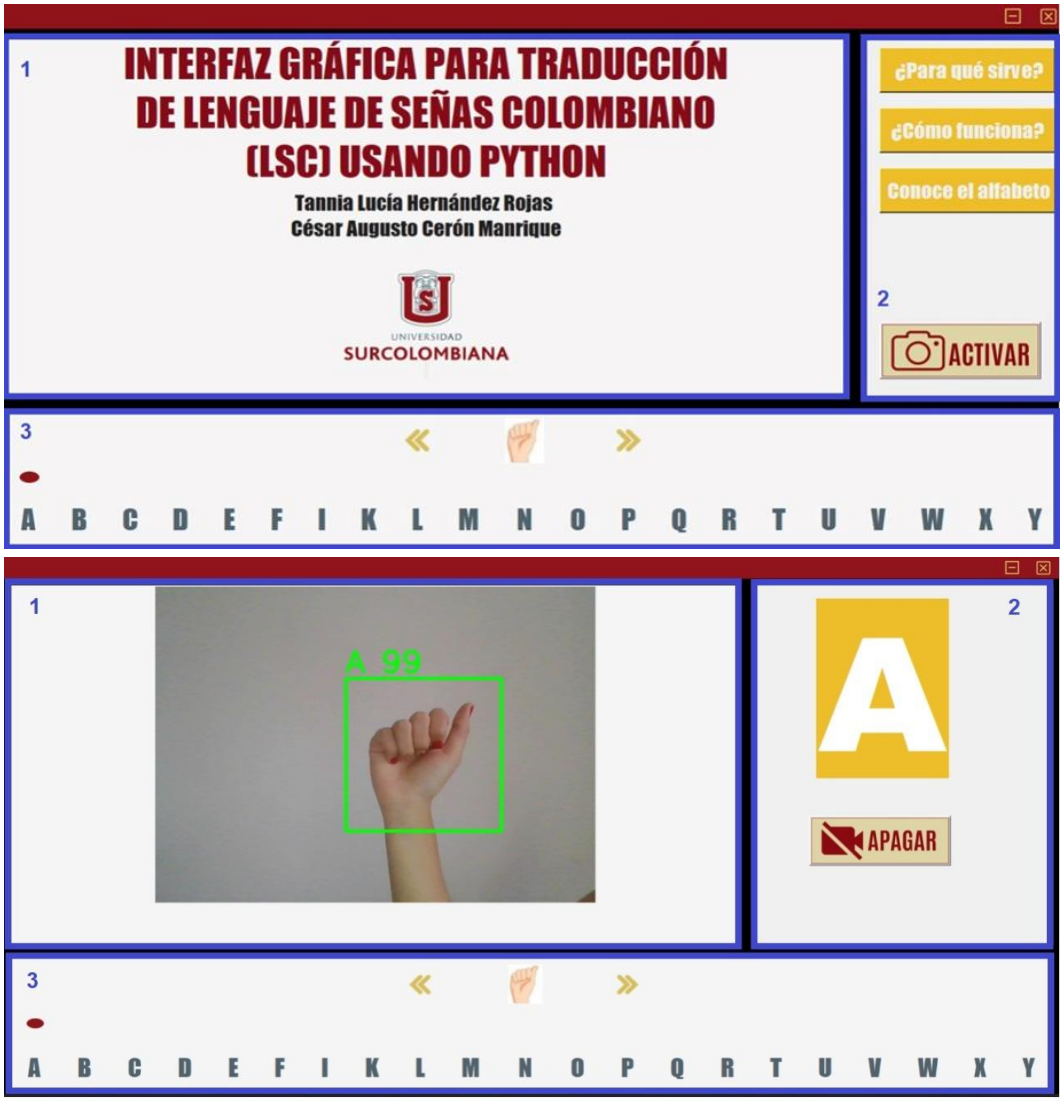


Figura 20. Diagrama de flujo de la interfaz

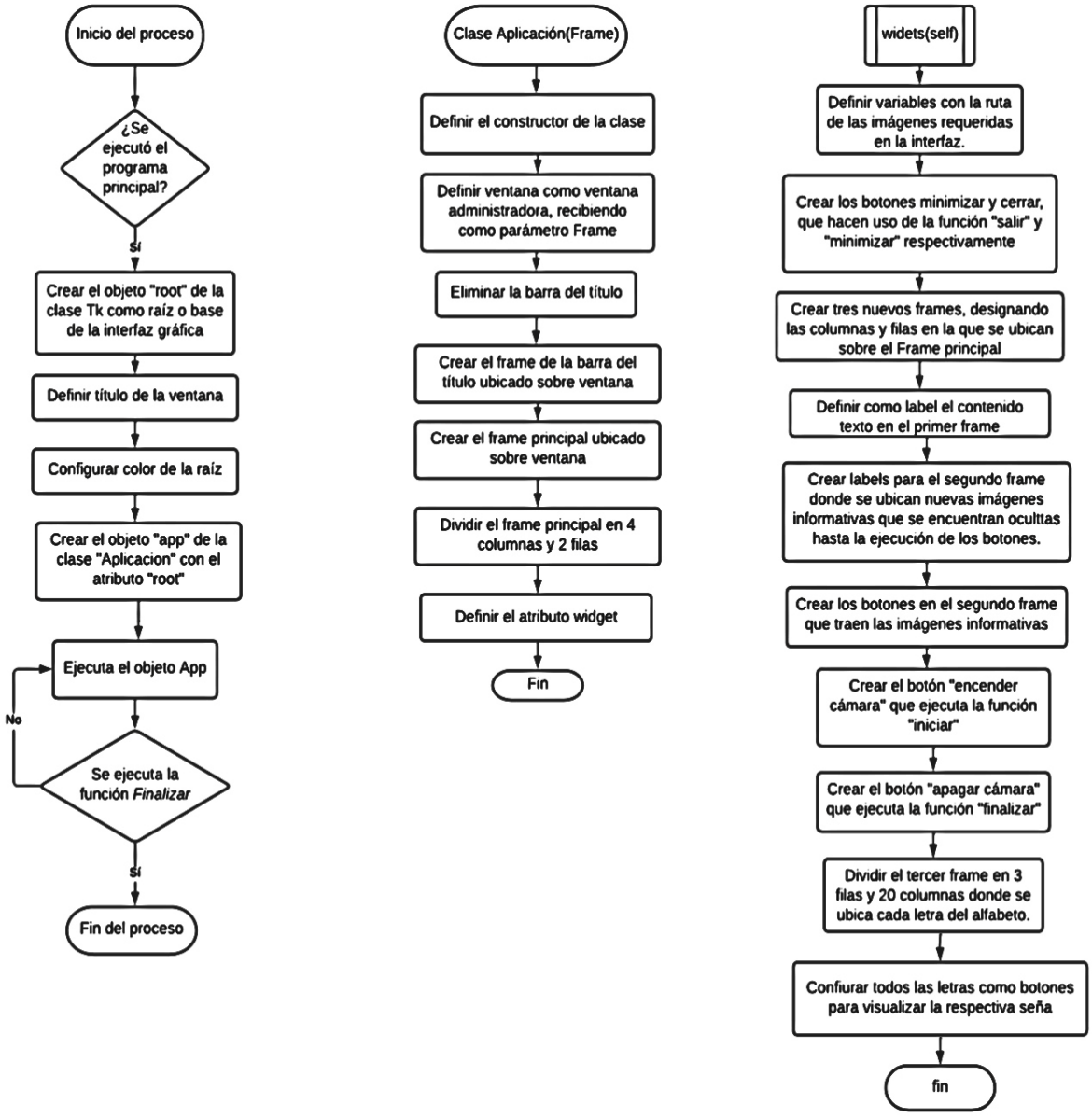


Figura 21. Diagrama de flujo para los métodos de la clase Aplicación

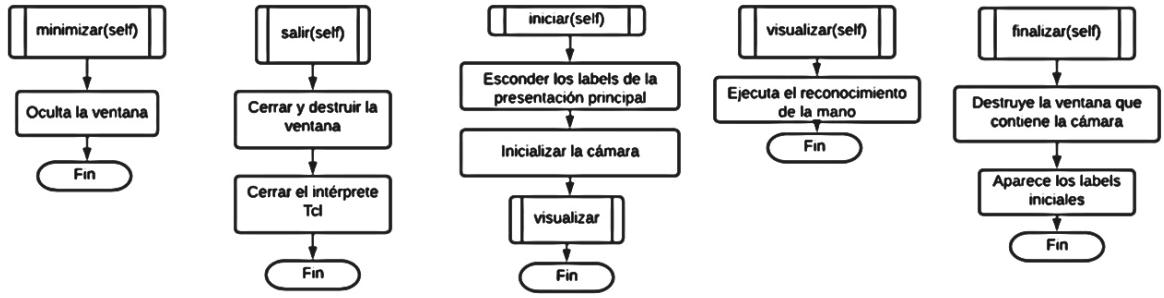
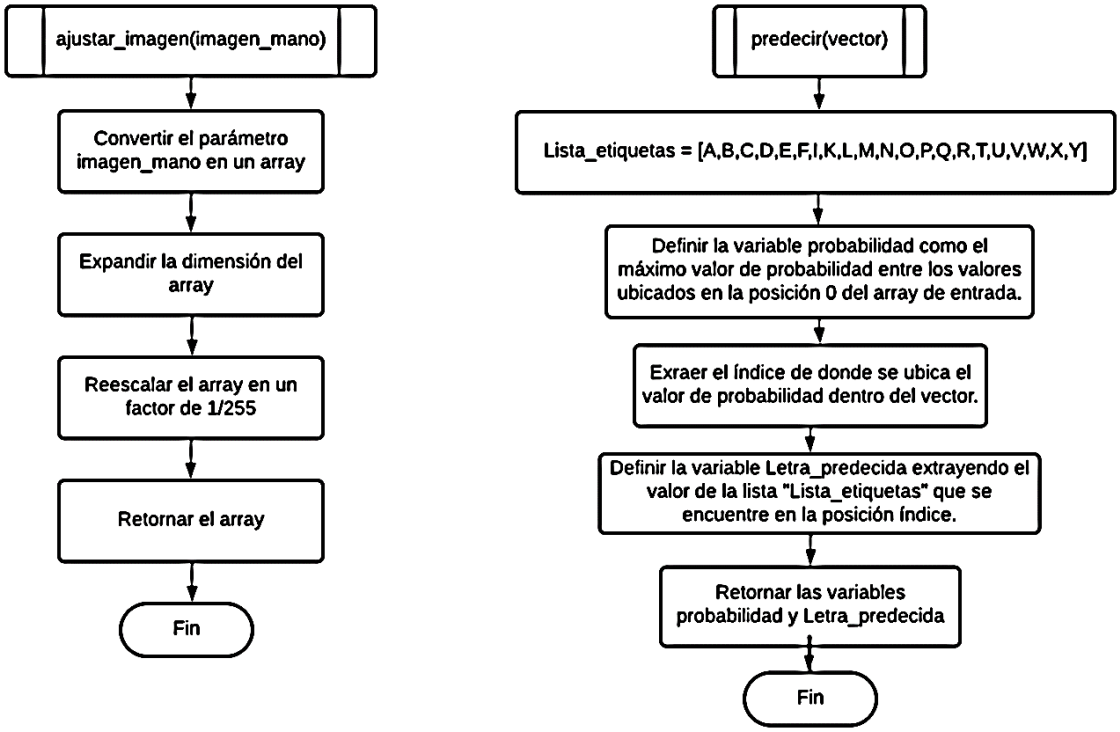


Figura 22. Diagrama de flujo para las funciones de la clase Aplicación

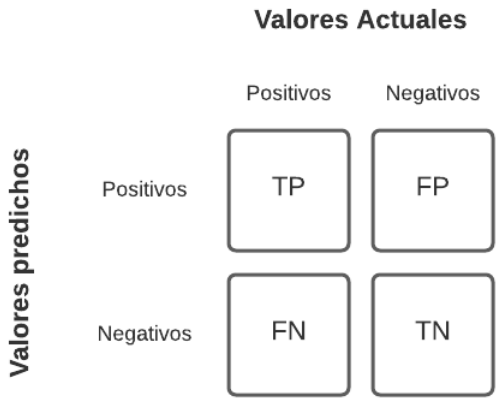


7. RESULTADOS

Para analizar el comportamiento de las predicciones de la red neuronal convolucional desarrollada, es imperativo el conocimiento de las métricas de evaluación debido a que nos permite medir el rendimiento que esta posee. En primer lugar, se tiene la matriz de confusión que permite conocer información relevante del rendimiento del clasificador y a partir de esta se pueden extraer las demás métricas.

La matriz de confusión corresponde a una tabla que resume el número de predicciones correctas e incorrectas de un conjunto de datos (Datos de testeo)[38]. La matriz de confusión se representa de la siguiente manera:

Figura 23. Representación de la matriz de confusión[38]



Las predicciones se establecen como se especifica a continuación:

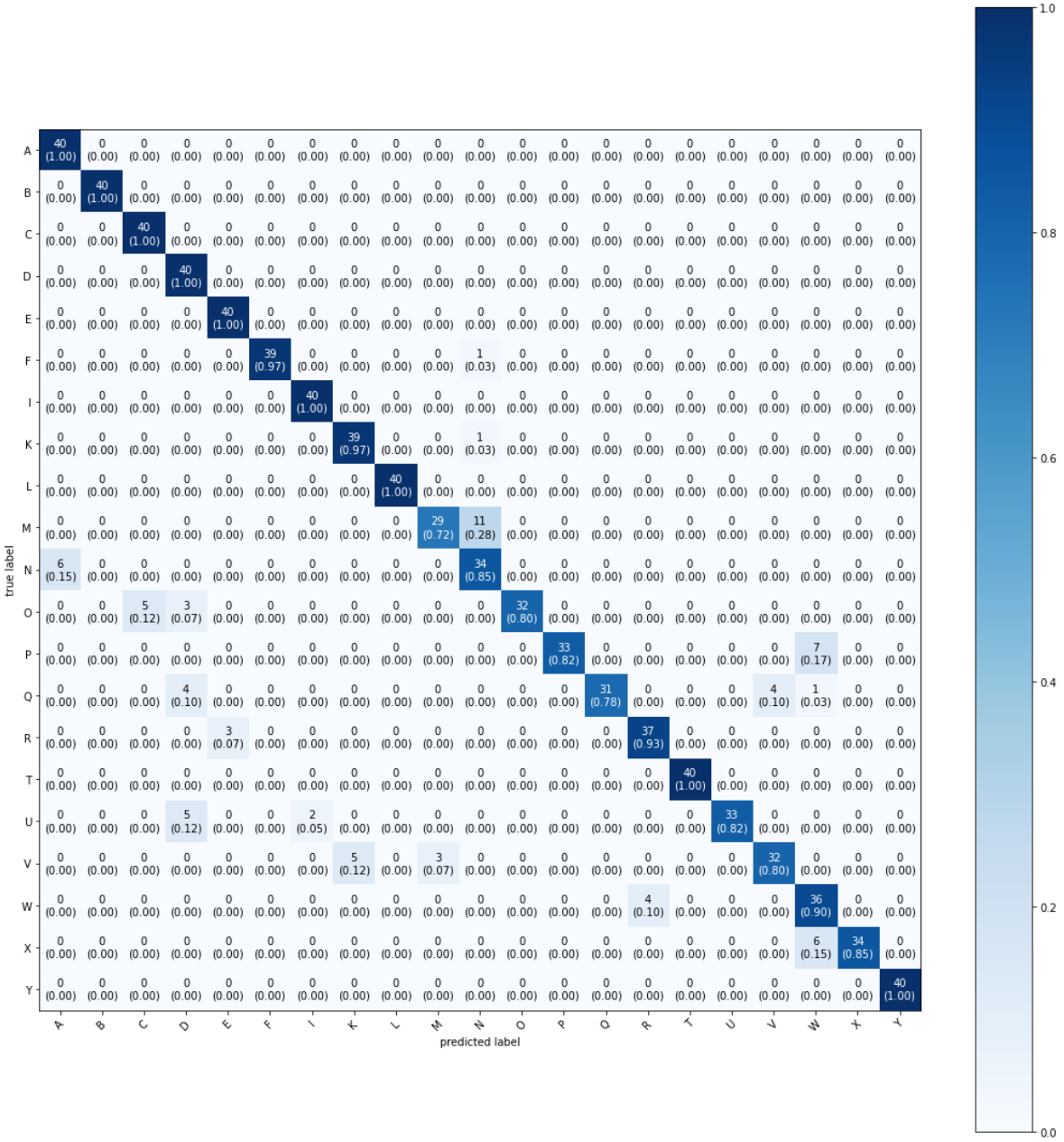
- Verdaderos positivos (TP): Es la cantidad de predicciones verdaderas y correctas.
- Falsos negativos (FN): Es la cantidad de predicciones falsas incorrectas.
- Falsos positivos (FP): Es la cantidad de predicciones verdaderas incorrectas.
- Verdaderos negativos (TN): Es la cantidad de predicciones falsas incorrectas.

Con los 4 valores obtenidos en la matriz de confusión es posible extraer la exactitud, la precisión, la sensibilidad, entre otras métricas que facilitan el análisis del rendimiento del clasificador. Con ayuda de la librería Scikit-learn es posible obtener

[38] SHIN, Terence. Comprensión de la Matriz de Confusión y Cómo Implementarla en Python. DataSource.AI. 20 de mayo de 2020. Disponible en: https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python

la matriz de confusión, para esto se hace uso del subconjunto de datos correspondiente a testeo que posee 40 imágenes de cada clase.

Figura 24. Matriz de confusión



A. EXACTITUD: La exactitud representa el número total de predicciones correctas respecto al número total de muestras clasificadas. Se calcula de la siguiente manera:

$$exactitud = \frac{\#\ predicciones\ correctas}{\#\ total\ de\ predicciones} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Ecuación (7)}$$

Esta métrica se usa solo en escenarios donde se presenta una cantidad de datos balanceados en el entrenamiento para cada clase.

B. PRECISIÓN: La precisión representa el número total de clasificaciones correctas en cada clase. Se calcula de la siguiente manera:

$$precision = \frac{TP}{TP + FP} \qquad \text{Ecuación (8)}$$

C. SENSIBILIDAD: O también llamado recall representa el número de clasificaciones que corresponde a una clase respecto a las clases evaluadas. Se calcula de la siguiente manera:

$$sensibilidad = \frac{TP}{TP + FN} \qquad \text{Ecuación (9)}$$

D. VALOR F1: O también llamado F1-Score representa una relación entre la precisión y la sensibilidad, por lo general se usa para obtener el rendimiento general del modelo. Se calcula de la siguiente manera:

$$valor\ f1 = 2 * \frac{precision * sensibilidad}{precision + sensibilidad} \qquad \text{Ecuación (10)}$$

Las métricas nombradas anteriormente toman valores en un rango de 0 a 1, donde 0 representaría una exactitud, precisión, sensibilidad o valor F1 muy bajo y 1 representaría una exactitud, precisión, sensibilidad o valor F1 muy alto[39].

[39] SITIOBIGDATA. Machine Learning: Seleccion Metricas de clasificacion. sitiobigdata.com [página web]. Disponible en: https://sitiobigdata.com/2019/01/19/machine-learning-metrica-clasificacion-parte-3/.

Mediante Scikit-learn se obtiene el reporte de clasificación para cada clase, los resultados se muestran a continuación:

Tabla 3 Reporte de clasificación

CLASE	PRECISION	RECALL	F1-SCORE	SUPPORT
A	0.87	1.00	0.93	40
B	1.00	1.00	1.00	40
C	0.89	1.00	0.94	40
D	0.77	1.00	0.87	40
E	0.93	1.00	0.96	40
F	1.00	0.97	0.99	40
I	0.95	1.00	0.98	40
K	0.89	0.97	0.93	40
L	1.00	1.00	1.00	40
M	0.91	0.72	0.81	40
N	0.72	0.85	0.78	40
O	1.00	0.80	0.89	40
P	1.00	0.82	0.90	40
Q	1.00	0.78	0.87	40
R	0.9	0.93	0.91	40
T	1.00	1.00	1.00	40
U	1.00	0.82	0.90	40
V	0.89	0.80	0.84	40
W	0.72	0.90	0.80	40
X	1.00	0.85	0.92	40
Y	1.00	1.00	1.00	40

Por otro lado, las métricas obtenidas en general en la red neuronal mediante el reporte de clasificación:

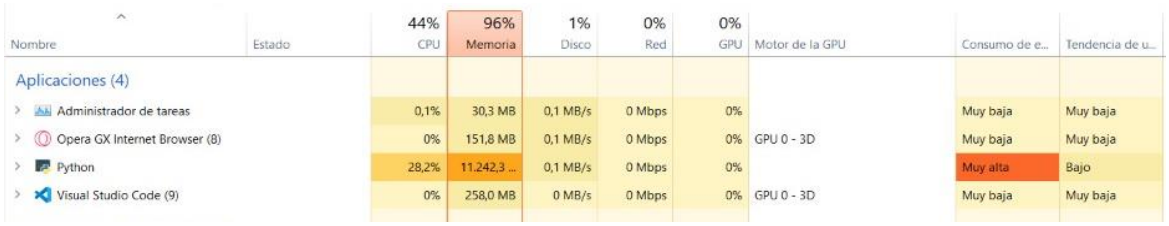
- Exactitud: 0.92.
- Precisión: 0.93.
- Sensibilidad: 0.92.
- Valor F1: 0.92.

El desarrollo e implementación del proyecto se ejecutó en un equipo que presenta las siguientes especificaciones:

- Procesador Intel® Core™ i5-10300H.
- Memoria RAM instalada de 16Gb.
- Unidad de estado sólido PCIe® NVMe™ M.2 de 256 GB.

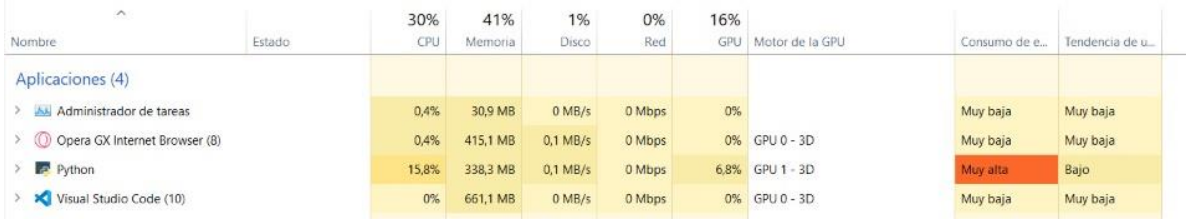
A continuación, se presenta el administrador de tareas el cual permite realizar la comparación del consumo computacional del equipo ejecutando la interfaz gráfica respecto a la ventana de OpenCV:

Figura 25. Consumo computacional con interfaz



Nombre	Estado	44% CPU	96% Memoria	1% Disco	0% Red	0% GPU	Motor de la GPU	Consumo de e...	Tendencia de u...
Aplicaciones (4)									
Administrador de tareas		0,1%	30,3 MB	0,1 MB/s	0 Mbps	0%		Muy baja	Muy baja
Opera GX Internet Browser (8)		0%	151,8 MB	0,1 MB/s	0 Mbps	0%	GPU 0 - 3D	Muy baja	Muy baja
Python		28,2%	11.242,3 ...	0,1 MB/s	0 Mbps	0%		Muy alta	Bajo
Visual Studio Code (9)		0%	258,0 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D	Muy baja	Muy baja

Figura 26. Consumo computacional sin interfaz



Nombre	Estado	30% CPU	41% Memoria	1% Disco	0% Red	16% GPU	Motor de la GPU	Consumo de e...	Tendencia de u...
Aplicaciones (4)									
Administrador de tareas		0,4%	30,9 MB	0 MB/s	0 Mbps	0%		Muy baja	Muy baja
Opera GX Internet Browser (8)		0,4%	415,1 MB	0,1 MB/s	0 Mbps	0%	GPU 0 - 3D	Muy baja	Muy baja
Python		15,8%	338,3 MB	0,1 MB/s	0 Mbps	6,8%	GPU 1 - 3D	Muy alta	Bajo
Visual Studio Code (10)		0%	661,1 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D	Muy baja	Muy baja

Como se puede apreciar, la ejecución de la interfaz presenta un alto consumo de recursos computacionales (CPU y memoria RAM) debido al trabajo que debe realizar Tkinter en el procesamiento de la imagen en tiempo real capturada por la cámara.

8. CONCLUSIONES

El proyecto de grado expuesto en este documento muestra el proceso para el desarrollo de una interfaz gráfica de lenguaje de señas colombiano mediante el lenguaje de programación Python. Este proyecto exhibe el uso de diferentes tecnologías para lograr su ejecución y presenta toda la investigación y estudio para obtener un óptimo funcionamiento. Las conclusiones obtenidas del presente trabajo se indican a continuación.

- Mediante las métricas se puede concluir que el rendimiento obtenido en el clasificador es óptimo debido a que presenta porcentajes superiores al 90% en la predicción general del modelo.
- Se logró construir un set de datos completo de 63.112 imágenes, con un 16,78% perteneciente a los datos de validación y el porcentaje restante a entrenamiento. Normalmente el conjunto de datos suele ser dividido en un 80% de datos de entrenamiento y un 20% de datos de validación, se eligió este porcentaje con el fin de evitar el subajuste o underfitting que ocurre cuando los datos de entrenamiento son insuficientes. La correcta partición de los datos permitió generar un modelo eficiente.
- Se encontraron dificultades con respecto a las condiciones ambientales, debido a que es importante un ambiente controlado que cuente con una incidencia de luz directa hacia la mano, pero no una sobrexposición de esta. También se debe evitar la sombra que realiza la mano y el cuerpo del usuario en espacios oscuros, para que la librería MediaPipe pueda realizar la correcta detección de los 21 puntos de referencia de la mano.
- Mediapipe dispone de las librerías y acceso a bases de datos requeridos para la identificación de la mano, lo que facilita el proceso debido a que, de no contar con esta herramienta, hubiese sido necesario crear un modelo específico para dicha tarea. Mediapipe en conjunto con OpenCV representan un complemento ideal para el desarrollo de proyectos que requieren detección en tiempo real y todo lo correspondiente al área de visión por computador.
- Se creó un algoritmo de predicción en tiempo real mediante el uso de las librerías de código abierto, Tensorflow y Keras que favorecieron en la implementación de las redes neuronales y en general, la creación del modelo de aprendizaje. Estas librerías por excelencia se han definido como idóneas para el aprendizaje profundo ya que facilitan el proceso y mejoran la rapidez de los sistemas.
- Se presentaron dificultades con algunas señas debido a la similitud con otras imágenes en la posición de los dedos. Cuando la red neuronal establece los pesos lo hace para determinar características como detección de bordes, formas, inclinación entre otras. Al existir señas en las que dichas características presentan patrones repetidos o similares, el modelo predice menor probabilidad como se expone en la matriz de confusión. Este factor se puede solucionar incrementando el conjunto de datos, mejorando la posición de la mano, creando

una red más profunda o ubicando la mano en una posición conveniente con respecto a la incidencia de luz.

- Se implementó una interfaz dinámica e intuitiva por medio de la librería Tkinter, que permite interactuar con el alfabeto de señas colombiano y mostrar la traducción de la seña al usuario en tiempo real, lo que favorece a una comunicación fluida entre personas con discapacidad auditiva y personas con condición auditiva normal.
- La interfaz presenta como problemática un alto consumo de recursos computacionales, debido a la constante transformación que se realiza sobre el video, capturando la imagen en alta resolución y convirtiéndola en arreglos para poder ser ubicada sobre el Frame de Tkinter. Por esta razón se recomienda interactuar con la interfaz con fines informativos y ejecutar el traductor directamente desde la ventana de OpenCV.
- Este proyecto representa una contribución significativa, desde el sentido social incluyente y el desarrollo tecnológico. Es una base sólida para futuros proyectos relacionados a la inteligencia artificial, aplicados principalmente en la región del Huila que está abordando estas tecnologías de vanguardia.

9. DISCUSIONES Y TRABAJOS FUTUROS

Es imprescindible continuar en el avance tecnológico y en especial en esta área de la inteligencia artificial, por esta razón este proyecto puede continuar en su proceso de evolución y mejora. A continuación, se plantean posibles mejoras que se pueden implementar a futuro.

- Permitir el ingreso mediante voz de la letra del alfabeto y que el traductor presente la respectiva seña.
- Adicionar una salida de audio que favorezca en la interpretación de la seña para el usuario oyente.
- Implementar las letras dinámicas del alfabeto de señas colombiano, aumentando la complejidad del proyecto con la detección de video.
- Con el fin de mejorar la efectividad y eficacia de la interfaz, es necesario ejecutar el proyecto en un equipo con un buen procesador y con un buen espacio de memoria RAM, debido al alto consumo de recursos que genera la interfaz. En este sentido, como mejora se plantea trabajar en reducir este gasto computacional.
- Compilar la red neuronal convolucional haciendo uso de GPU ya sea de manera local o en la nube, para obtener tiempos de entrenamiento menores o trabajar con un conjunto de datos de 3 canales.

BIBLIOGRAFÍA

ActiveState. What is Tkinter used for and How to Install this Python Framework?. 21 de septiembre de 2021. Disponible en: https://www.activestate.com/resources/quick-reads/what-is-tkinter-used-for-and-how-to-install-it/.

ALEGRE GUTIÉRREZ, Enrique, et al. Procesamiento Digital de Imagen: Fundamentos y Prácticas Con Matlab [En línea]. Secretariado de Publicaciones y Medios Audiovisuales, 2003. Disponible en: https://www.researchgate.net/publication/229828279_Procesamiento_Digital_de_Imagenes_Fundamentos_y_Practicas_con_Matlab_Digital_image_processing_Fundamentals_and_practices_with_Matlab.

ARÉVALO, V. M.; GONZÁLEZ, J. y AMBROSIO, G. La Librería de Visión Artificial Opencv Aplicación a la Docencia e Investigación. En: MAPIR Research Group [en línea]. 2017. Disponible en: http://mapir.isa.uma.es/varevalo/drafts/arevalo2004lva1.pdf.

ARISTOTELES. Política, Libro I. De la Sociedad Civil. De la Esclavitud. De la Propiedad. Del Poder Doméstico.

BAGNATO, Juan Ignacio. ¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador [blog]. Aprende Machine Learning en español. Coruña, España. 29 de noviembre de 2018. Disponible en: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

BAGNATO, Juan Ignacio. Sets de Entrenamiento, Test y Validación [blog]. Aprende Machine Learning en español. Coruña, España. 3 de marzo de 2020. Disponible en: https://www.aprendemachinelearning.com/sets-de-entrenamiento-test-validacion-cruzada/

BOTINA MONSALVE, et al. Clasificación Automática de las Vocales en el Lenguaje de Señas Colombiano. Disponible en: https://repositorio.itm.edu.co/handle/20.500.12622/1038.

CALVO, Diego. Función de activación – Redes neuronales [blog]. Diego Calvo. 7 de diciembre de 2018. Disponible en: https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/

CASTILLO, José Antonio. RGB qué es esto y para qué se utiliza en Informática [blog]. Profesional Review. 20 de enero de 2019. Disponible en: https://www.profesionalreview.com/2019/01/20/rgb-que-es/

CHALLENGER PÉREZ, Ivet; DÍAZ RICARDO, Yanet y BECERRA GARCÍA, Roberto Antonio. El lenguaje de programación Python. En: Ciencias Holguín [En línea.] 2014. vol. XX, no. 2. Disponible en: https://www.redalyc.org/articulo.oa?id=181531232001.

CHOLLET, Francois. Deep learning with python. [s.l.]: Manning Publications Co, 2018. 361 p. ISBN 9781617294433

CIENCIACINÉTICA. Redes Neuronales Convolucionales en Inteligencia Artificial (CNN) [blog]. INTELIGENC[IA]. 6 de junio de 2018. Disponible en: https://inteligencia.tech/2018/06/06/redes-convolutivas-en-inteligencia-artificial/

DeepAI. Computer Vision: What is Computer Vision? [En línea.]. Disponible en: https://deepai.org/machine-learning-glossary-and-terms/computer-vision.

DOMÍNGUEZ, Néstor A. Ética y Ecoética para la inteligencia artificial. En: Boletín del Centro Naval 856 [En línea.]. 2021. Disponible en: https://www.centronaval.org.ar/boletin/BCN856/856-DOMINGUEZ.pdf

GARCÍA SÁNCHEZ, Eugenio. Introducción a las redes neuronales de convolución. Aplicación a la visión por ordenador. Trabajo de grado. [s.l.]: Universidad de Zaragoza, 2019.

GONZÁLEZ MARCOS, Ana, et al. Técnicas y algoritmos básicos de visión artificial [en línea]. [s.l.]: Material didáctico. Ingenierías., 2006. Disponible en: https://publicaciones.unirioja.es/catalogo/online/VisionArtificial.pdf. ISBN 84-689-9345-X.

IBM. ¿Qué es Machine Learning? [En línea.]. IBM ANALÍTICA web site. Disponible en: https://www.ibm.com/co-es/analytics/machine-learning

IBM. What is Computer Vision?. [En línea.]. IBM ANALÍTICA web site. Disponible en: https://www.ibm.com/topics/computer-vision.

LOPEZ BRIEGA, Raul E. Introducción al Deep Learning [blog]. Matemáticas, análisis de datos y python. 13 de junio de 2017. Disponible en: https://relopezbriega.github.io/blog/2017/06/13/introduccion-al-deep-learning/

MEDIAPIPE HANDS. Mediapipe [página web]. Disponible en: https://google.github.io/mediapipe/solutions/hands.html

MINISTERIO DE EDUCACIÓN NACIONAL, INSTITUTO NACIONAL PARA SORDOS, INSOR. Diccionario Básico de la Lengua de Señas Colombiana por Instituto Nacional para Sordos INSOR [En línea.]. INSOR web site. Disponible en: http://www.insor.gov.co/descargar/diccionario_basico_completo.pdf

NVIDIA DEVELOPER. What is OpenCV? [En línea.]. NVIDIA Corporation web site. Disponible en: https://developer.nvidia.com/opencv

ORACLE COLOMBIA. ¿Qué es la Inteligencia Artificial-IA? [En línea.]. Oracle Colombia web site. Disponible en: https://www.oracle.com/co/artificial-intelligence/what-is-ai

PONCE GALLEGOS, Julio Cesar, et al. Inteligencia Artificial. [s.l.]: Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn), 2014. 225 p.

PORCELLI, Adriana Margarita. La Inteligencia Artificial y la Robótica: sus Dilemas Sociales, Éticos y Jurídicos. Derecho glob. Estud. Sobre Derecho Justicia [En línea.]. 2020, vol.6, n.16, pp.49-105. Disponible en: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2448-51362020000300049&lng=es&nrm=iso

RODRÍGUEZ, Daniel. ¿Cuál es la diferencia entre Parámetro e Hiperparámetro?. Analytics Lane. 16 de diciembre de 2019. Disponible en: https://www.analyticslane.com/2019/12/16/cual-es-la-diferencia-entre-parametro-e-hiperparametro/#:~:text=Los%20hiperpar%C3%A1metros%20de%20un%20modelo,por%20el%20cient%C3%ADfico%20de%20datos

ROZO MELO, Nancy. La Lengua de Señas Colombiana [En línea.]. Portal De Lenguas de Colombia: Diversidad y Contacto. Disponible en: https://lenguasdecolombia.caroycuervo.gov.co/contenido/Lenguas-de-senas-colombiana/introduccion

SHIN, Terence. Comprensión de la Matriz de Confusión y Cómo Implementarla en Python. DataSource.AI. 20 de mayo de 2020. Disponible en: https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python

SILVA, Sarahí y FREIRE, Estefanía. Intro a las redes neuronales convolucionales. Bootcamp AI. 23 de noviembre de 2019. Disponible en: https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8.

SITIOBIGDATA. Machine Learning: Seleccion Metricas de clasificacion. sitiobigdata.com [página web]. Disponible en: https://sitiobigdata.com/2019/01/19/machine-learning-metrica-clasificacion-parte-3/.

TF.KERAS.PREPROCESSING.IMAGE.IMAGEDATAGENERATOR. TensorFlow [página web]. Disponible en: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator.

TORRES SOLER, Luis. El Perceptrón Redes Neuronales Artificiales [En línea] 20 de enero de 2020. Repositorio Universidad Nacional. Disponible en: https://disi.unal.edu.co/~lctorress/RedNeu/LiRna004.pdf

TRIVIÑO LOPEZ, Iván. Sistema para el Aprendizaje del Lenguaje de Señas Colombiano usando Visión por Computador. Disponible en: https://ciencia.lasalle.edu.co/ing_automatizacion/159/.

UNIÓN INTERNACIONAL DE TELECOMUNICACIONES. RECOMENDACIÓN UIT-R BT.601-7: Parámetros de Codificación de Televisión Digital para Estudios con Formatos de Imagen Normal 4:3 y de Pantalla Ancha 16:9 [En línea]. Disponible en: https://web.archive.org/web/20220119174709/https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-S.pdf

UTRERA BURGAL, Jesús. Tratamiento de imágenes usando ImageDataGenerator en Keras [blog]. Technical thoughts, stories and ideas. 2 de agosto de 2019. Disponible en: https://enmilocalfunciona.io/tratamiento-de-imagenes-usando-imagedatagenerator-en-keras/

VELO FUENTES, Edward Joseph. Introducción a los métodos Deep Learning basados en Redes Neuronales [En línea]. Trabajo Fin de Máster. [s.l.]: Universidad de Coruña, 2020. Disponible en: http://eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_1654.pdf.

ANEXO

Anexo A. Código del desarrollo de la red neuronal convolucional

```
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from keras.layers import Convolution2D, MaxPooling2D
from keras.preprocessing import image
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import plot_model
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow

K.clear_session()

datos_entrenamiento = ".\Entrenamiento"
datos_validacion = ".\Validacion"

Generador1 = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 20,
    width_shift_range = 0.1,
    height_shift_range = 0.1,
    horizontal_flip = True,
)

Generador2 = ImageDataGenerator(
    rescale = 1./255
)

imagen_entrenamiento = Generador1.flow_from_directory(
    datos_entrenamiento,
    target_size = (altura, longitud),
    batch_size = batch_size,
    color_mode = 'grayscale',
    class_mode = "categorical"
)

imagen_validacion = Generador2.flow_from_directory(
    datos_validacion,
```

```
    target_size = (altura, longitud),
    batch_size = batch_size,
    color_mode = 'grayscale',
    class_mode = "categorical"
)

plt.imshow(next(imagen_entrenamiento)[0][1,...,0])
plt.show()
print(imagen_entrenamiento.class_indices)

epocas = 30
pasos_entrenamiento = imagen_entrenamiento.n//batch_size
pasos_validacion = imagen_validacion.n//batch_size
print("Los pasos en entrenamiento son: " + str(pasos_entrenamiento))
print("Los pasos en validacion son: " + str(pasos_validacion))
filtrosconv0 = 32
filtrosconv1 = 64
filtrosconv2 = 64
tam_filtro0 = (4, 4)
tam_filtro1 = (3, 3)
tam_filtro2 = (2, 2)
tam_pool = (2, 2)
clases = 21
altura, longitud = 50, 50
batch_size = 32

cnn = Sequential()
cnn.add(Convolution2D(filtrosconv0, tam_filtro0, input_shape=(altura, longitud, 1),
activation="relu"))
cnn.add(MaxPooling2D(pool_size=tam_pool))

cnn.add(Convolution2D(filtrosconv1, tam_filtro1, activation="relu"))
cnn.add(Convolution2D(filtrosconv1, tam_filtro1, activation="relu"))
cnn.add(MaxPooling2D(pool_size=tam_pool))

cnn.add(Convolution2D(filtrosconv2, tam_filtro2, activation="relu"))
cnn.add(Convolution2D(filtrosconv2, tam_filtro2, activation="relu"))
cnn.add(MaxPooling2D(pool_size=tam_pool))

cnn.add(Flatten())
cnn.add(Dense(1345, activation="relu"))
cnn.add(Dropout(0.20))
cnn.add(Dense(clases, activation="softmax"))

cnn.summary()
```

```
plot_model(cnn, to_file="Modelo.png", show_shapes=True)

optimizador = tensorflow.keras.optimizers.Adam(learning_rate = 0.0005)
cnn.compile(loss="categorical_crossentropy",                optimizer=optimizador,
metrics=["accuracy"])

clasificador = cnn.fit(imagen_entrenamiento,
                       validation_data = imagen_validacion,
                       steps_per_epoch = pasos_entrenamiento,
                       validation_steps = pasos_validacion,
                       batch_size = batch_size,
                       epochs = epocas)

cnn.save("Modelo.h5")
cnn.save_weights("Pesos.h5")

plt.figure(figsize=(12,12))
plt.subplot(2,1,1)
plt.plot(clasificador.history['accuracy'], '-')
plt.plot(clasificador.history['val_accuracy'], '-')
plt.xlabel('Épocas')
plt.ylabel('Precisión')
plt.legend(['Entren','Valid'], loc='upper left')
plt.title('Precisión de entrenamiento y validación')
plt.grid()

plt.subplot(2,1,2)
plt.plot(clasificador.history['loss'], '-')
plt.plot(clasificador.history['val_loss'], '-')
plt.title('Pérdida del modelo')
plt.xlabel('Épocas')
plt.ylabel('Pérdida')
plt.legend(['Entren','Valid'], loc='upper left')
plt.title('Pérdida de entrenamiento y validación')
plt.grid()
plt.show()

Lista_Precision = clasificador.history['accuracy']
Lista_Precision.insert(0, 'Precisión')
Lista_Precision_Validacion = clasificador.history['val_accuracy']
Lista_Precision_Validacion.insert(0, 'Precisión de Validación')
Lista_Perdida = clasificador.history['loss']
Lista_Perdida.insert(0, 'Pérdida')
Lista_Perdida_Validacion = clasificador.history['val_loss']
```

```
Lista_Perdida_Validacion.insert(0, 'Pérdida de Validación')
Lista_Final = [Lista_Precision, Lista_Precision_Validacion, Lista_Perdida, Lista_Perdida_Validacion]

Dataframe = pd.DataFrame(Lista_Final).transpose()
Dataframe.to_excel('Datos_Red.xlsx', index = False)
```

Anexo B. Código de obtención de métricas del rendimiento del clasificador

```
from keras import backend as K
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
from mlxtend.plotting import plot_confusion_matrix
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt

K.clear_session()

Nombre_Clases = ['A', 'B', 'C', 'D', 'E', 'F', 'I', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'T', 'U', 'V', 'W', 'X', 'Y']
altura, longitud = 50, 50
batch_size = 32

Directorio_Testeo = './Testeo'
Generador_Testeo = ImageDataGenerator(
        rescale = 1./255
)
Datos_Testeo = Generador_Testeo.flow_from_directory(
        Directorio_Testeo,
        target_size = (longitud, altura),
        batch_size = batch_size,
        color_mode = 'grayscale',
        class_mode = ''categorical'',
        shuffle = False
)

modelo = 'Modelo.h5'
pesos_modelo = 'Pesos.h5'
cnn = load_model(modelo)
cnn.load_weights(pesos_modelo)

Predicciones = cnn.predict(Datos_Testeo)
```

```
Datos_Prediccion = np.argmax(Predicciones, axis=1)
Datos_Reales = Datos_Testeo.classes
accuracy_score(Datos_Reales, Datos_Prediccion)

precision_score(Datos_Reales, Datos_Prediccion, average='macro')

recall_score(Datos_Reales, Datos_Prediccion, average='macro')

f1_score(Datos_Reales, Datos_Prediccion, average='macro')

Matriz = confusion_matrix(Datos_Reales, Datos_Prediccion)
plot_confusion_matrix(conf_mat = Matriz, figsize=(15, 15), class_names = Nombre_Clases, show_normed = True, colorbar = True)
plt.tight_layout()

print(classification_report(Datos_Reales, Datos_Prediccion, target_names=Nombre_Clases))
```

Para conocer más acerca del código del sistema, contactar a los autores al correo u20171154848@usco.edu.co y u20171154840@usco.edu.co.