Instituto Tecnológico Superior de Abasolo
Organismo Público Descentralizado del Gobierno del Estado de Guanajuato.

Desarrollo de una interfaz gráfica para el reconocimiento de rostros

Tesis profesional que para obtener el título de:

Ingeniera Mecatrónica

Presentan:

María Dolores González Mosqueda

Rosa Kassandra Rodríguez Martínez

Directora de Tesis:

Aztatzi Pluma Dalyndha

ABASOLO, GTO. 18/06/2024

Autorización de presentación de trabajo de titulación

Se autoriza a las estudiantes González Mosqueda María Dolores y Rodríguez Martínez Rosa Kassandra inscritas en la Ingeniería Mecatrónica del Instituto Tecnológico Superior de Abasolo, con los números de control AS19110196 Y AS19110114, respectivamente, a defender su proyecto de trabajo de titulación Desarrollo de una interfaz gráfica para el reconocimiento de rostros, generados en el ITESA. Ante el comité sinodal asignado.

La presente se expide en Abasolo Guanajuato al 18 de junio de 2024.

_____
Dra. Dalyndha Aztatzi Pluma
Presidenta del sínodo

_____
Dr. Jesús Norberto Guerrero Tavares
Secretario del sínodo

_____
M.E.E.E. José Sabás Segura
Vocal del sínodo

Agradecimientos

Al finalizar esta tesis titulada "Reconocimiento facial por medio de una red neuronal convolucional", quiero expresar mi más profundo agradecimiento a todas las personas que han hecho posible la culminación de este trabajo.

En primer lugar, quiero agradecer a la Doctora Aztatzi Pluma Dalyndha, por creer en nuestro potencial y ayudarnos en cada paso de este proyecto. Su guía y apoyo constante han sido fundamentales para llevar a la culminación exitosa de esta investigación. También extiendo mi gratitud a mi colega y amiga Kassandra por su colaboración en este proyecto, por su amistad que ha sido una fuente constante de motivación y apoyo.

A mis padres, Teresa y Manuel, quiero expresarles mi más sincero agradecimiento. Con mucho amor y esfuerzo, me han brindado la oportunidad de estudiar, impulsándome a ser mejor cada día. Los amo con todo mi corazón. Y a mis hermanos Maria y Juan Pablo que fueron modelos principales en este proyecto y siempre estuvieron ayudándome; a mis tías Alicia, Teresa y Virginia, gracias por motivarme y ayudarme en todo el proceso. Sin ustedes, nada de esto sería posible.

A todos ustedes, mis más sinceros agradecimientos y amor.

Resumen

La combinación de inteligencia artificial (IA) y Redes Neuronales Convolucionales (CNN) ha revolucionado la visión por computadora, permitiendo a las máquinas interpretar y comprender imágenes de manera inteligente. Estas tecnologías han demostrado un impacto significativo en una variedad de aplicaciones prácticas y están transformando fundamentalmente la forma en que se interactúa con el mundo digital y físico. Se desarrollo una interfaz gráfica intuitiva y de fácil acceso para mejorar el reconocimiento facial mediante la implementación de modelos de redes neuronales convolucionales con el lenguaje de programación Python.

Para alcanzar este objetivo, se llevó a cabo la creación y procesamiento de una base de datos de imágenes de individuos, junto con la implementación de OpenCV para abordar los desafíos que presentan los algoritmos de visión. El proyecto se centró en mejorar la eficacia del reconocimiento facial y democratizar el acceso a esta tecnología, permitiendo que usuarios de diferentes niveles de habilidad puedan beneficiarse sin complicaciones. La combinación de inteligencia artificial y redes neuronales convolucionales está impulsando una revolución en la visión por computadora, transformando nuestra sociedad y nuestro mundo de formas nunca vistas. En dicho proyecto se utilizaron redes neuronales convolucionales y perceptrón multicapa como parte final del modelo que, con varias modificaciones en este, la normalización de las imágenes, y un exhaustivo entrenamiento se logró obtener un mejor desempeño de los resultados con un error debajo de 0.7 cercana a cero y una exactitud mayor al 0.95 cercana a 1. Resultando así un reconocimiento facial por medio de 3 técnicas que van desde el reconocimiento en una foto, video pre grabado y un video en vivo, existieron limitantes en cuanto a los videos en vivo, sin embargo los resultados fueron satisfactorios cumpliendo así el propósito del proyecto y sus respectivos objetivos con un reconocimiento en su totalidad de las personas ingresadas a la base de datos así como la creación de las interfaces graficas ayudando así a la manipulación de la captura y reconocimiento de rostros ingresados en la actualidad o en un futuro incremento de rostros.

Abstract

The combination of artificial intelligence (AI) and Convolutional Neural Networks (CNN) has revolutionized computer vision, allowing machines to intelligently interpret and understand images. These technologies have demonstrated significant impact in a variety of practical applications and are primarily transforming the way the digital and physical world is interacted with. An intuitive and easily accessible graphical interface was developed to improve facial recognition by implementing convolutional neural network models with the Python programming language.

To achieve this goal, the creation and processing of a database of images of individuals was carried out, along with the implementation of OpenCV to address the challenges presented by vision algorithms. The project focuses on improving the effectiveness of facial recognition and democratizing access to this technology, allowing users of different skill levels to benefit without complications. The combination of artificial intelligence and convolutional neural networks is driving a revolution in computer vision, transforming our society and our world in ways never seen before. In this project, convolutional neural networks and multilayer perceptron were used as the final part of the model that, with several modifications to it, the normalization of the images, and exhaustive training, it was possible to obtain better performance of the results with an error below 0.7 close to . to zero and an accuracy greater than 0.95 close to 1. Thus resulting in facial recognition through 3 techniques that range from recognition in a photo, pre-recorded video and a live video, there were limitations regarding live videos, However, the results were satisfactory, thus fulfilling the purpose of the project and its respective objectives with a complete recognition of the people entered into the database as well as the creation of graphical interfaces, thus helping to manipulate the capture and recognition of faces entered currently or in a future increase of faces.

Índice

Índice de tablas

Índice de figuras

INTRODUCCIÓN

En el contexto del vertiginoso avance tecnológico, el reconocimiento facial emerge como un campo de gran interés debido a su amplio espectro de aplicaciones, que van desde la seguridad hasta la autenticación de identidad. En este documento, titulado "Desarrollo de una interfaz gráfica para el reconocimiento de rostros", nos sumergimos en la elaboración de una solución basada en inteligencia artificial (IA) para la identificación y clasificación de rostros humanos. Nuestro enfoque principal recae en el uso de redes neuronales convolucionales, una técnica de aprendizaje automático especialmente efectiva en el análisis de imágenes.

El reconocimiento facial, una habilidad inherente a los seres humanos, ha sido reproducido y mejorado gracias al desarrollo tecnológico, permitiendo que la IA desempeñe un papel crucial en este proceso de identificación. En este proyecto, exploramos el vasto campo del aprendizaje automático, haciendo uso específico del enfoque del Machine Learning para entrenar algoritmos capaces de reconocer y distinguir entre diferentes rostros con una precisión notable. La elección de Python como lenguaje de programación principal se justifica por su versatilidad y el amplio abanico de librerías disponibles, especialmente aquellas diseñadas para tareas de aprendizaje automático. Estas herramientas, combinadas con la implementación de un sistema de entrenamiento basado en datos divididos en conjuntos de entrenamiento, validación y pruebas, proporcionan la estructura necesaria para el desarrollo eficaz de la interfaz gráfica de reconocimiento facial.

Una de las piedras angulares del proyecto fue la optimización del algoritmo de entrenamiento. Esto implicó ajustar, mejorar las técnicas utilizadas para procesar, analizar las imágenes de rostros, con el objetivo de aumentar la precisión y la eficiencia del sistema. Este enfoque meticuloso en la optimización garantizo que la interfaz gráfica desarrollada sea capaz de reconocer rostros con una precisión notable en una variedad de condiciones y contextos.

Para lograr este objetivo, se empleo el uso de diversas técnicas al igual que estrategias, incluida la división de datos en conjuntos de entrenamiento, validación y pruebas, así como

la implementación de capas convolucionales en el modelo de red neuronal. Estas capas permitieron que a la IA extraer características clave de las imágenes, facilitando así el proceso de reconocimiento facial. El documento también resalta la importancia de la calidad de los datos y las imágenes utilizadas en el proceso de entrenamiento. La selección de imágenes claras tal como de alta resolución garantizo que el sistema pueda aprendiera de manera efectiva las características distintivas de cada rostro, lo que resulto en un reconocimiento facial más preciso además de confiable.

En resumen, este proyecto represento un esfuerzo integral para desarrollar una interfaz gráfica para el reconocimiento facial utilizando técnicas avanzadas de aprendizaje automático. A lo largo de este documento, se detallan los pasos seguidos, los métodos empleados así como los resultados obtenidos, con el objetivo de demostrar la efectividad y la viabilidad de esta solución en el campo del reconocimiento facial.

CAPÍTULO 1. ESTADO DEL ARTE

1.1 Antecedentes históricos

La evolución de las Redes Neuronales Convolucionales (CNN) se remonta a la década de 1940 con la propuesta de modelos de neuronas artificiales y el desarrollo del perceptrón en 1958 por Frank Rosenblatt. A lo largo de las décadas siguientes, surgieron técnicas para entrenar redes neuronales profundas, aunque enfrentaron desafíos como el desvanecimiento del gradiente. En 1980, Kunihiko Fukushima propuso el Neocognitron, una de las primeras CNN, seguido por LeNet-5 de Yann LeCun en la década de 1990, marcando hitos en la visión por computadora. A partir de entonces, las CNN han evolucionado con mejoras en la arquitectura, demostrando su potencial en la clasificación de imágenes con el éxito de AlexNet en 2012, seguido por otras arquitecturas como VGG, Inception, ResNet y EfficientNet, que han mejorado aún más el rendimiento en tareas de reconocimiento de imágenes y visión por computadora.

1.1.1 Redes Neuronales Convolucionales (CNN)

A continuación se menciona un poco de la evolución de las Redes Neuronales Convolucionales:

- En la década de 1980, Yann LeCun desarrolló la primera arquitectura de red neuronal convolucional llamada LeNet, diseñada para tareas de reconocimiento de caracteres escritos a mano.
- Las CNN introdujeron capas convolucionales y de agrupación que permiten a la red aprender características locales y jerárquicas en imágenes, reduciendo la cantidad de parámetros y mejorando la eficiencia computacional.
- A principios de la década de 2010, las CNN demostraron un rendimiento sobresaliente en competiciones de reconocimiento de imágenes, como el Desafío de Reconocimiento de Imágenes de Imagenet (ImageNet Large Scale Visual Recognition Challenge), catalizando un resurgimiento en el interés por las redes neuronales profundas.

Con respecto a los antecedentes del proyecto, se encontró un trabajo titulado "Implementación de una red neuronal de convolución para el reconocimiento de poses en imágenes de rostros", realizado por Méndez e Ibarra (2014). Este trabajo presento una exploración del uso de redes neuronales profundas de convolución para el reconocimiento de poses horizontales en imágenes de rostros, (Méndez, P., & Ibarra, J., 2014).

También, se destaco un trabajo de pase de lista en un aula, donde se realizó un análisis exhaustivo del desempeño de una red neuronal convolucional en el reconocimiento facial. Se observó que el uso de distintos factores, como la cantidad o calidad de las imágenes en el conjunto de datos, influía significativamente en la precisión del sistema. Los experimentos demostraron una precisión superior al 88.7% para un grupo de alumnos y mayor al 90% para otro grupo, destacando la importancia de la calidad del conjunto de datos en el entrenamiento del modelo, (Morales Velasco, L. R., 2023).

Además, otro proyecto similar al que se realizó describe el desarrollo de un sistema de reconocimiento facial para el control de accesos mediante inteligencia artificial. Utilizó algoritmos de Redes Neuronales Convolucionales (CNN) además el lenguaje de programación Python, junto con las bibliotecas Numpy, Os, OpenCV e Imutils para su implementación. El sistema logro una precisión de aproximadamente el 88% en la predicción por persona, utilizando un dataset de 4500 imágenes. Este proyecto destaco en la aplicación de técnicas de inteligencia artificial, específicamente redes neuronales convolucionales, en la identificación biométrica facial (Reyes Campos, J. E. M., Castañeda Rodríguez, C. S., Alva Luján, L. D., & Mendoza de los Santos, A. C., 2023).

1.1.2 Avances Recientes

Desde entonces, ha habido numerosos avances en el diseño y entrenamiento de CNN, incluida la introducción de arquitecturas más profundas y complejas como VGG, ResNet, Inception y EfficientNet. Estos avances han llevado a mejoras significativas en el rendimiento de las CNN en una amplia gama de aplicaciones de visión por computadora, incluida la detección de objetos, el reconocimiento facial, la segmentación semántica y más.

1.1.3 Impacto y Aplicaciones

Las CNN han revolucionado la visión por computadora y han encontrado aplicaciones en diversas industrias, incluida la medicina (diagnóstico médico asistido por computadora), la automoción (conducción autónoma), la seguridad (reconocimiento facial) y la industria (inspección automatizada).

Los antecedentes de las CNN se basan en décadas de investigación en redes neuronales, aprendizaje automático así mismo procesamiento de imágenes; Por lo que, han evolucionado hasta convertirse en una de las tecnologías más influyentes actualmente. En la actualidad existe un gran número de sistemas de reconocimiento facial que utilizan la clasificación de imágenes o fotografías de los rostros que son digitales hoy en día. Puesto que, anteriormente eran digitalizados a mano usando distintas técnicas, tomando en consideración los rasgos más característicos como los ojos, boca, nariz y el pelo. Cabe destacar que la base de datos era realizada a mano tomando en cuenta el nombre de la persona y algunas características.

1.2 Reconocimiento facial

El reconocimiento facial es una tecnología capaz de identificar o verificar a un sujeto a través de una imagen, considerando diferentes características faciales particulares de los rostros tales como el tamaño de ojos, nariz, orejas, mentón y boca, también en diferentes ambientes de iluminación así como la posición del sujeto.

El proceso de captura de rostro transformo la información analógica (rostro) en un conjunto de información digital. Sus aplicaciones pueden ir desde la seguridad de un sistema, búsqueda de personas desaparecidas, mercadotecnia, publicidad, hasta el sector de salud donde se puede ser de ayuda con la atención de los pacientes, o incluso ser capaz de detectar enfermedades genéticas específicas. En la siguiente Figura 1.1 se muestra el rastreo de rasgos de una persona centrándose en los ojos, nariz, y comisuras de los labios, (Kaspersky, s.f.).

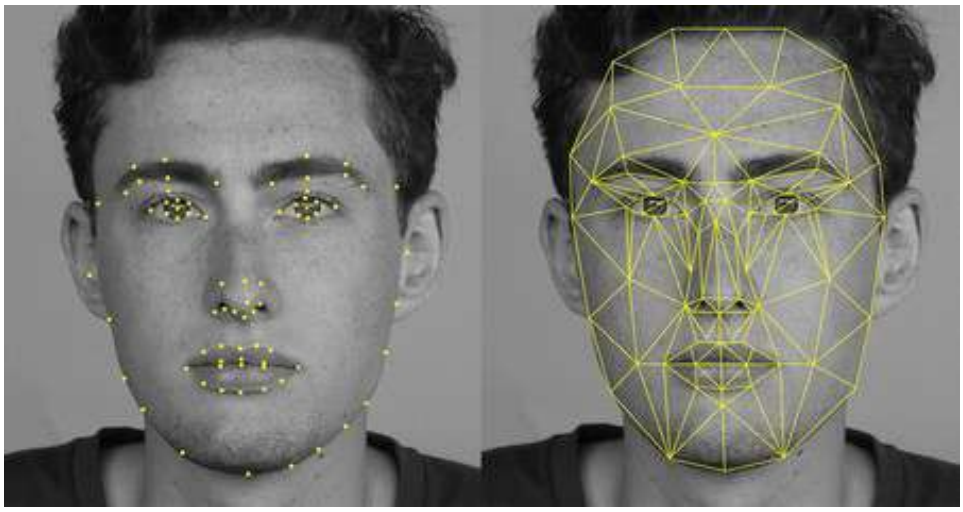

Figura 1.1 Reconocimiento facial y el proceso de reconocimiento facial, tomando en cuenta características básicas del rostro como cejas, ojos, nariz, labios y el contorno del rostro.

Como conclusión el reconocimiento facial fue posible gracias al desarrollo tecnológico que a su vez ha permitido que la inteligencia artificial (IA) sea empleada para esta aplicación definiendo a la inteligencia artificial como la habilidad de un sistema para interpretar datos externos a este, aprender de esos datos que emplearán el conocimiento adquirido para alcanzar metas y tareas específicas. Los sistemas de inteligencia artificial funcionan con algoritmos, al usar técnicas como el aprendizaje profundo al igual que el aprendizaje automático para demostrar conductas «inteligentes», (Hewlett Packard, 2019).

1.2.1 Detección Facial

Detrás del reconocimiento facial existen un sin fin de procedimientos y herramientas que lo hacen posible; Sin embargo, uno de los principales pilares es la (Detección), para una predicción es la manipulación de una imagen para la detección de un rostro en la misma, gracias al lenguaje de programación Python, con sus respectivas librerías utilizadas para el reconocimiento facial principalmente OpenCV. En la Figura 1.2 se muestra la imagen original, además el recorte del rostro detectado. Como punto importante en este apartado es que también se pueden cambiar las propiedades de la imagen tal como el color,

escalamiento, tamaño, posición, etc. Características que más adelante en la metodología del proyecto fueron de gran ayuda.

Figura 1.2 Ejemplo de manipulación de imagen como base principal del uso de OpenCV para detectar el rostro de una persona en la imagen, (Haar Cascade face detecction).

1.2.1.1 1.2.1.1 Open CV y Haar Cascade face detection

La función con la que Python por medio de *OpenCv* permitió la captura de imágenes y videos en tiempo real por medio de la cámara del equipo estuvo dada por **cap = cv2.VideoCapture** ya sea que se utilice para leer o captar una imagen como anteriormente se uso, o para implementar un video Streaming y un video del directorio. Esto facilitó la multiple detección de rostros tal como se muestra en la Figura 1.3 en donde la presencia de 4 sujetos detectados son guardados, almacenados con un tamaño de 150 x150 pixeles y una etiqueta dada para la base de datos.

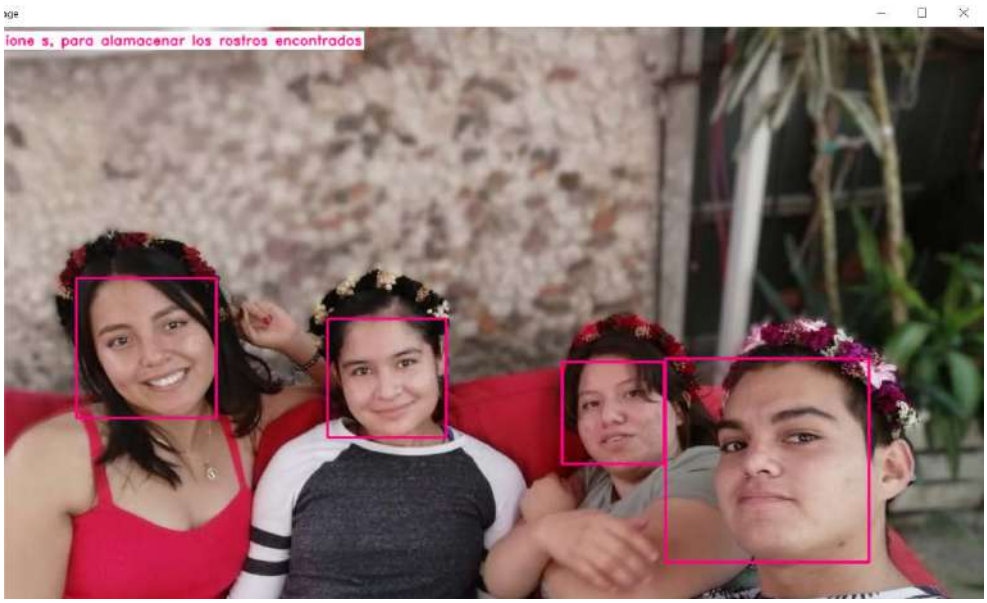

Figura 1.3 Detección de rostros múltiple por medio de (Haar Cascade face detecction) en la cual reconoce los cuatro rostros de la imagen.

Como parte complementaria se ha comprobado el porcentaje de confiabilidad de la librería OpenCV en la detección de rostros con los modelos, usando diferentes imágenes de entrada así como de video streaming donde se aplicarón diferentes movimientos al igual que expresiones para verificar su funcionamiento, como se muestra el la Figura 1.4 la deteción y confiabilidad de los rostros fue alta por encima de un 90 porciento, con el rostro descubrierto igualmente con accesorios que en este ejemplo fueron una gorra y lentes. Esto impacto directamente en el reconocimiento facial final, ya que la herramienta de detección fue parte principal para una buena predicción.

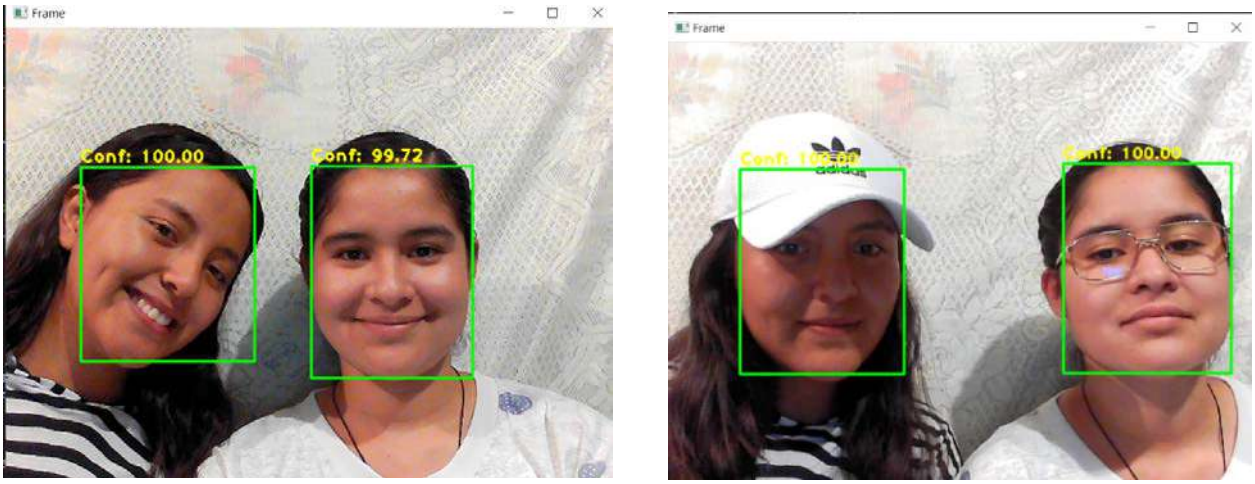

Figura 1.4 Detección de rostros utilizando modelos pre-entrenados de OpenCV con un porcentaje de alta confiabilidad, se detectaron dos rostros aún con accesorios como gorra y lentes.

A continuación, se muestran algunas de las ventajas del Clasificador Haar Cascade de OpenCV:

- Eficiencia Computacional: Es conocido por su eficiencia computacional, lo que lo hace adecuado para aplicaciones en tiempo real donde se necesita una detección rápida de rostros.
- Facilidad de Uso: OpenCV proporciona una implementación fácil de usar, lo que facilita su integración en proyectos de visión por computadora.
- Estabilidad: Aunque puede no ser tan preciso como MTCNN, tiende a ser más estable y menos propenso a falsos positivos en ciertas condiciones de entrada.
- Condiciones Controladas: Funciona bien en condiciones controladas donde la iluminación es uniforme y el fondo es simple, proporcionando resultados satisfactorios con una menor carga computacional.
- Bajo Requerimiento de Recursos: Requiere menos recursos computacionales y memoria en comparación con modelos basados en redes neuronales, lo que lo hace adecuado para aplicaciones en dispositivos con recursos limitados.

1.2.1.2 Red Neuronal Convolucional en Cascada Multitares (MTCNN)

Si bien se ha hablado de la importancia de un eficiente detector de rostros en este proyecto, también se hizo uso de un algoritmo por medio de redes neuronales convolucionales en cascada multitares, que por sus siglas es ingles MTCNN (Multi-Task Cascaded Convolutional Neural Network). Es conocido por su capacidad para detectar rápidos rostros en una imagen, incluso en diferentes ángulos, iluminación y expresiones faciales. Este modelo utilizó una cascada de redes neuronales convolucionales para realizar tres tareas principales en la detección facial: detección de rostros, localización de puntos clave faciales (como ojos, nariz y boca) y refinamiento de los cuadros delimitadores de los rostros. Está arquitectura en cascada permitió una detección eficiente y precisa, ya que cada etapa se especializó en una tarea específica.

La evaluación de este reconocimiento facial proporcionó una validación independiente y una perspectiva adicional sobre el rendimiento del modelo. Esta comparación cruzada permitió una evaluación más completa de la capacidad del modelo para generalizar a diferentes escenarios y condiciones de entrada. A continuación se muestrán algunas de estas ventajas:

- Precisión y Robustez: MTCNN utiliza redes neuronales convolucionales, lo que le otorga una alta precisión y robustez en la detección de rostros incluso en condiciones variables como cambios en la iluminación, ángulo de visión y expresiones faciales.
- Detección de Múltiples Rostros: Es capaz de identificar y localizar varios rostros en una sola imagen de manera eficiente.
- Detección de Puntos Clave Faciales: Además de rostros, MTCNN puede localizar puntos clave faciales como ojos, nariz y boca, lo que permite un análisis más detallado de las características faciales.
- Robustez ante Variaciones: Es robusto ante cambios en las condiciones de entrada, como variaciones en la iluminación, fondo complejo o ángulos de visión extremos.
- Implementación Eficiente: A pesar de utilizar redes neuronales convolucionales, está diseñado para ser eficiente en términos de velocidad de procesamiento.

1.3 Inteligencia Artificial

Tomando las bases del reconocimiento facial se introdujo directamente a los origenes tal como es la inteligencia artificial tal como muchas de las veces se relaciona con las redes neuronales, y si, las redes neuronales se han constituido como una técnica proveniente de ella, considerando que busca encontrar o soluciones informáticas, ha imitado la forma en la que actúa el cerebro humano de pensar, recordar, del mismo modo que resolver problemas. Dependiendo del número de neuronas, la red puede ser más o menos simple o profunda. La sinapsis entre neuronas, la transmisión de información, o el valor de salida de la neurona anterior se multiplica por un valor peso.

Estos pesos en los enlaces pueden incrementar o inhibir el estado de activación de las siguientes neuronas. A la salida de una neurona existe un filtro, una función limitadora o umbral, que modifica el valor del resultado o impone un límite que se debe sobrepasar para poder modificar a otra neurona, se conoce a esta función como función de activación , en Figura 1.5 se puede apreciar las cacateristicas basicas de una red neuronal tal como la señal de entrada que al momento de pasar por los pesos sinápticos llegando a la función de activación y hacen que los modelos sean no lineales para finalmente llegar a una función de activación, (Think Blg, 2019).

Así mismo se pueden ver con más detalle más adelante en el subtema 1.6 Funciones de Activación y posteriormente los resultados de la neurona con mayor peso donde son los dominantes en la siguiente neurona, mientras que la información de las neuronas con menos peso no se transmitio. Por lo que se puede decir que la matriz de pesos gobierna todo el sistema neuronal obteniendo una salida.

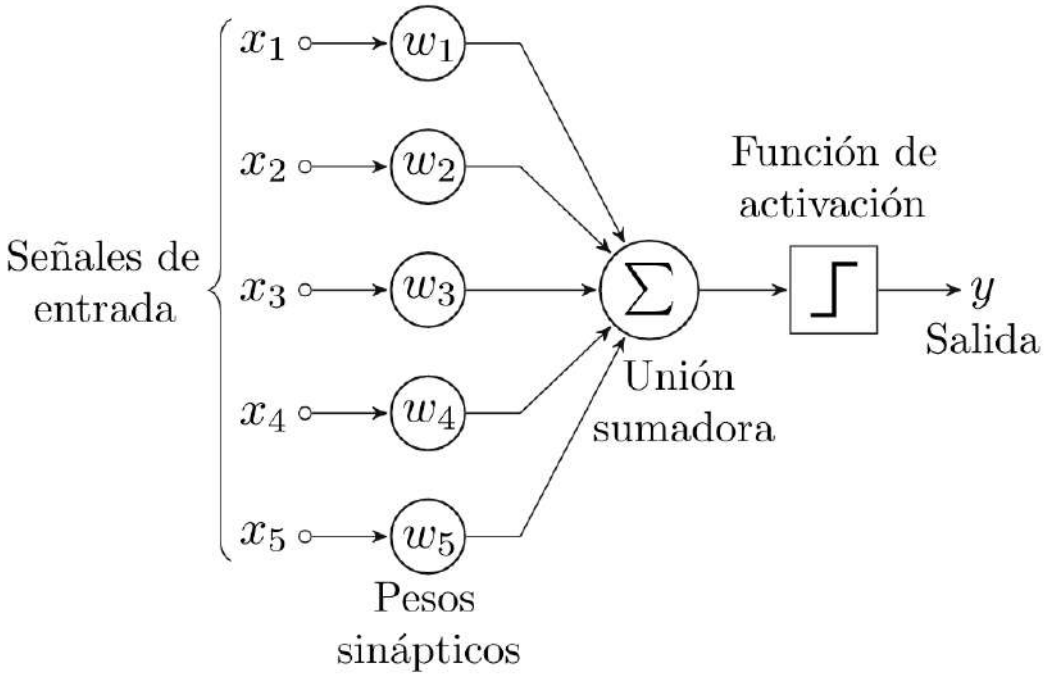

Figura 1.5 Ejemplo de aplicación de la IA con una estructura básica de una red neuronal.

1.3.1 Red Neuronal

Si bien se comento anteriormente que las redes neuronales son un subconjunto del machine learning al igual que el núcleo de los algoritmos de aprendizaje profundo. Están compuestas por capas de nodos, que incluyen una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo está conectado a otro, tiene un peso y un umbral asociados. Si la salida de un nodo individual está por encima del valor de umbral especificado, ese nodo se activa y envía datos a la siguiente capa de la red. De lo contrario, no se pasa ningún dato a la siguiente capa de la red.

Las redes neuronales artificiales comenzaron asignando valores aleatorios a los pesos de las conexiones entre neuronas. La clave para que la red neuronal realizará su tarea de forma correcta y precisa, fue ajustar estos pesos a los números adecuados tal como se muestra en la Figura 1.6 haciendo enfais en la definición de una *Neurona* y como se identifica dentro de una red subconjunta, puede existir una variabilidad en la sinapsis producida en las conexiones entre las dendritas de las neuronas, también pueden influir de

manera distinta de la "activación" de las neuronas, debido a que algunas sinapsis refuerzan la activación mientras que otras inhiben la activación ambos casos con diferentes grados. Se cree que esta "variabilidad" es la base del aprendizaje humano, (González, 2024).

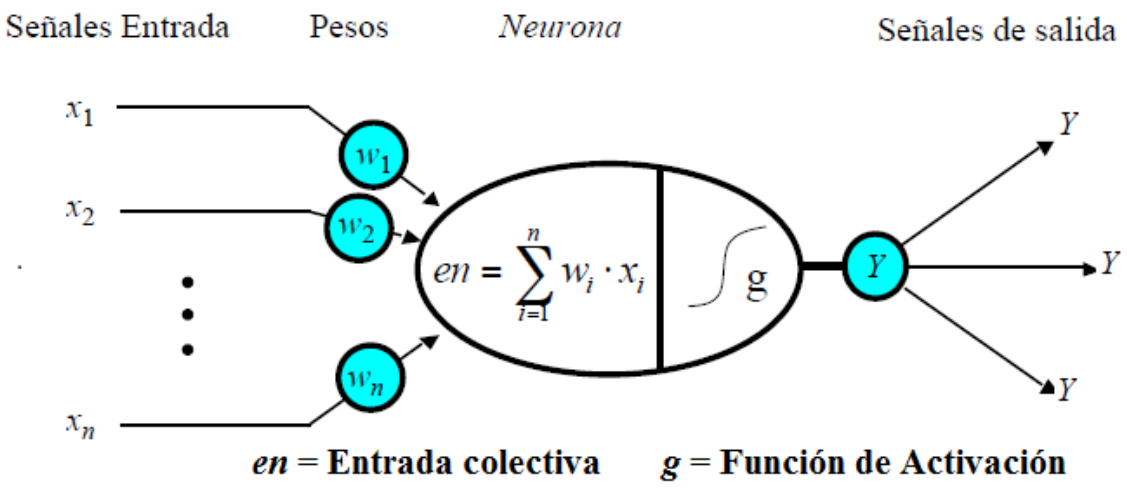

Figura 1.6 Descripción gráfica, matemática de una neurona, el conjunto de una red además de sus respectivas características de entrada y salida.

La neurona artificial más simple es la denominada como "perceptrón" que se ha caracterizado por ser una neurona artificial con una función de activación umbral. Formalmente, el umbral se representa como otro peso más (denominado como "sesgo") cuya entrada constante es '1' y su peso es negativo. Varios perceptrones componen una red neuronal de una "capa", (CMI, 2022).

1.4 Machine Learning (ML)

Existen diferentes tipos de inteligencia artificial como cualquier otra ciencia, como lo son la inteligencia Simbólica, la cual implica reglas codificadas elaboradas por programadores, también los algoritmos genéticos o algoritmos de optimización que imita la sección natural

que sucede en la naturaleza. Los Algoritmos Bayesianos basados en el teorema de bayes siendo una herramienta matemática usada para calcular la probabilidad de que un evento ocurra dado otro evento también conocida como la probabilidad condicional, (Verch, s.f.).

Por ultimo tenemos el Machine Learning o aprendizaje profundo es el campo de estudio en el que esta investigación se centró y se define como el campo de estudio de la inteligencia artificial el cual proporciona a las computadoras la habilidad de aprender sin ser explícitamente programadas creando sistemas que desarrollan un aprendizaje automáticamente, identifica patrones de alta complejidad en millones de datos.

El Machine Learning en otras palabras es el diseño de nuevas máquinas que emulan la inteligencia humana, buscando suplir la necesidad de algunas actividades que realizan los seres humanos a su vez, reducir el margen de error, (Iturburu, 2022). En la **Figura 1.7** se muestran las diferentes ramas y requerimientos del Machine Learning las cuales fueron implementadas en este proyecto, tales como el Deep Learning, AI, Neuronas, Algoritmos, Clasificaciones, etc.

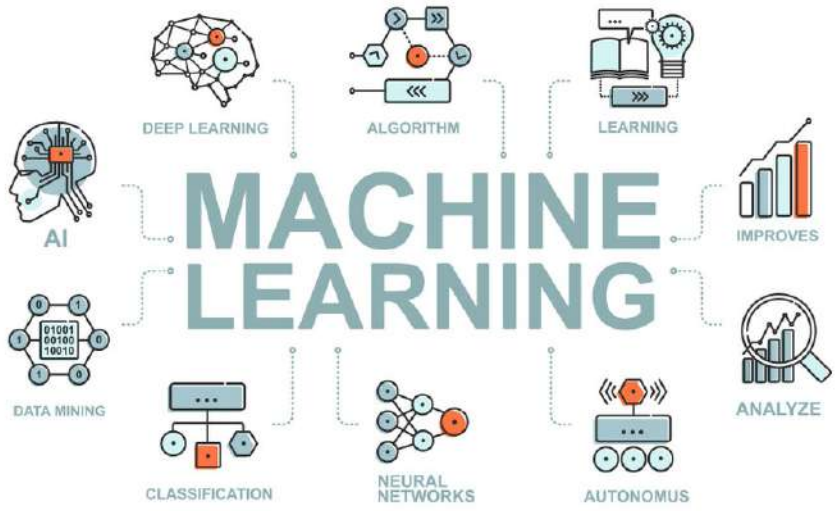

Figura 1.7 Esquema de los requerimientos y ramas del machine Learning dentro del campo de la Inteligencia artificial.

Los algoritmos de aprendizaje automático dentro del Aprendizaje Profundo se dividen ampliamente en cuatro tipos:

1.4.1 Aprendizaje supervisado

La máquina se entrena mediante ejemplos. El operador proporciona al algoritmo de aprendizaje automático, un conjunto de datos predefinido con entradas y salidas especificadas, el algoritmo tiene que averiguar cómo obtener esas entradas y salidas. El algoritmo predice, además es corregido por el operador; Por lo tanto, este proceso se repite hasta que el algoritmo alcanza un alto nivel de precisión.

1.4.2 Aprendizaje no supervisado

El programa de aprendizaje automático examina los datos para detectar tendencias. No hay tecla de respuesta ni interferencia humana que sirva de guía. El programa intenta organizar los datos de forma que describan su estructura. A medida que evalúa datos adicionales, aumenta y se refina su capacidad para tomar decisiones basadas en esos datos.

1.4.3 Aprendizaje semi-supervisado

Similar al aprendizaje supervisado, pero emplea datos etiquetados y no etiquetados. Los datos etiquetados son principalmente información que tiene etiquetas semánticas para que el algoritmo pueda interpretarla, pero los datos no etiquetados no tienen esa información. Los sistemas de aprendizaje automático pueden aprender a categorizar los datos no etiquetados utilizando esta combinación.

1.4.4 1.4.4 Aprendizaje forzado

El aprendizaje por refuerzo se ocupa de los procedimientos de aprendizaje regimentado en los que un algoritmo de aprendizaje automático recibe un conjunto de acciones, variables y valores finales que debe seguir. Siguiendo la definición de las reglas, el algoritmo intenta

explorar varias opciones y perspectivas, supervisando y evaluando cada salida para determinar cuál es la ideal. El aprendizaje por refuerzo instruye a la máquina mediante ensayo y error. Aprende de experiencias anteriores y empieza a cambiar su enfoque de la situación para alcanzar el mejor resultado posible, (Management Blog, 2022).

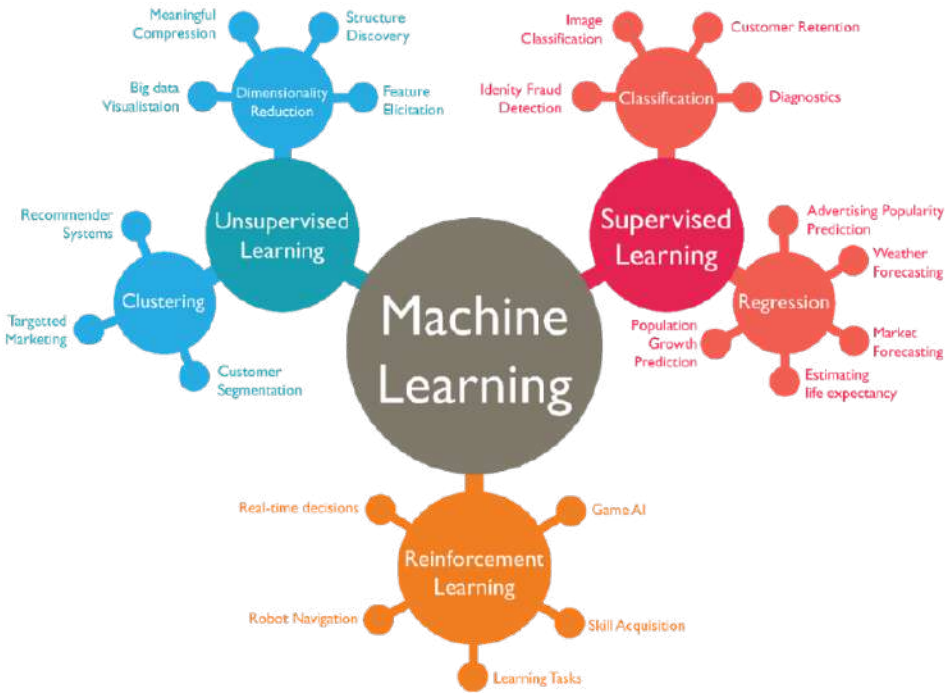

Figura 1.8 Tipos de aprendizaje del Machine Learning y sus diferentes ramas de aplicaciones.

El Machine Learning y el Deep Learning son un subconjunto concéntrico de la IA y la IA autómata. Si bien la IA crea la base para el procesamiento de datos para la creación de proyecciones, los algoritmos de Machine Learning (ML) permiten que la IA aprenda de las experiencias con esos datos, haciéndola una tecnología más inteligente. El Deep Learning profundiza esta capacidad a través de las redes neuronales, permitiendo generar resultados de manera cada vez más autónoma y exhaustiva, (GOTIYE, 2022).

1.5 Deep Learning

Como ya se observo una de las raices de la imitación del aprendizaje de los seres humanos es el Deep Learning que fue introducido por primera vez en 1986, pero no ha sido hasta el siglo XXI cuando su uso se ha puesto realmente de moda. El principal objetivo del Deep Learning es emular los enfoques de aprendizaje de los seres humanos, utilizando para ello una serie de características específicas: transformaciones no lineales y diferentes niveles de abstracción. Podemos enfocar el Deep Learning dentro de otros paradigmas de la Ciencia de Datos como una evolución natural de los mismos.

Concluyendo así con que el Deep Learning es la rama del aprendizaje automático que extrae patrones por medio de redes neuronales con varias capas de profundidad, que son capaces de generalizar bien sobre conjuntos grandes de datos que son intratables cuando se utilizan redes superficiales (pocas capas).

El tamaño de un modelo de Deep Learning o red neuronal lo determina el número de capas intermedias, como el ejemplo de la Figura 1.9 el número de neuronas de estas capas, y el grado de interconexión entre neuronas, el tamaño de las redes son dobladas cada cerca de 2 años y medio; Por lo que, no será hasta 2050 cuando se genere una red con la complejidad del cerebro humano. En la actualidad el trabajo con Deep Learning se ha visto favorecido por el acceso gratuito a un conjunto de librerías y especificaciones técnicas para su uso como son los frameworks: TensorFlow de Google, PyTorch de Facebook, MXNet de Amazon, entre otros, (CMI, 2022).

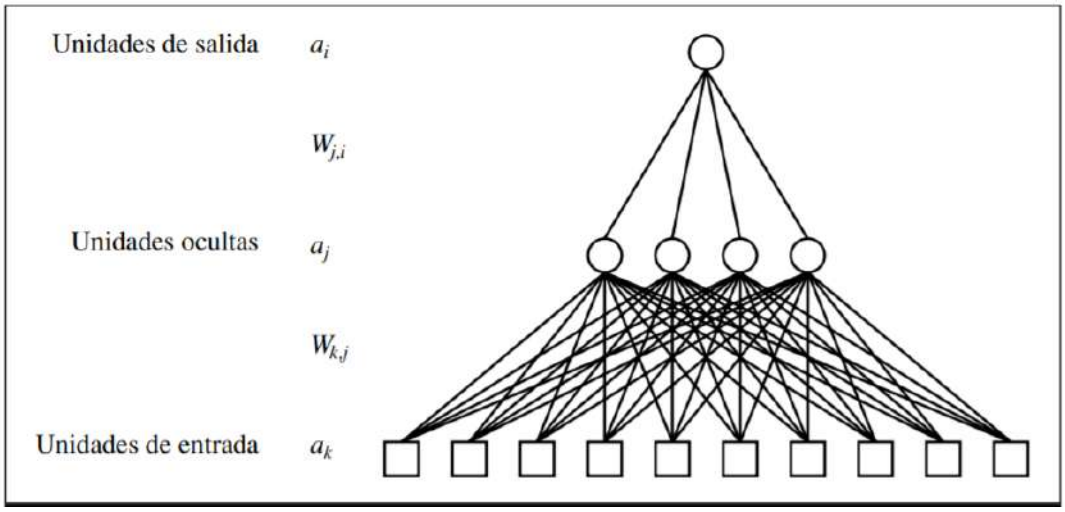

Figura 1.9 Ejemplo de estructura de una red neuronal multicapa del Deep Learning y su extracción de patrones.

1.6 Redes neuronales convolucionales

Las Redes Neuronales Convolucionales (CNN) son un tipo específico de Red Neuronal (Feed Forward) basada en la organización de las neuronas de los cerebros biológicos. Su comienzo se enmarca en un artículo llamado “Gradient-based learning applied to document recognition” publicado en 1988 por Yann LeCun, el cual se basa asimismo en el artículo Neocognitron de Kunihiko Fukushima (1980). Las CNN son un tipo de red neuronal profunda que ha demostrado ser muy efectiva en tareas de visión por computadora, incluida la clasificación de imágenes. Estas redes están especialmente diseñadas para procesar datos de imágenes de manera eficiente, aprendiendo representaciones jerárquicas a través de capas convolucionales y de agrupación.

Las redes neuronales convolucionales modelan de forma consecutiva pequeñas piezas de información, tratando de extraer información sobre diferentes patrones de cada imagen, esto se puede observar de mejor manera en Figura 1.10 donde la primera capa intentará detectar los *bordes* y establecer patrones de detección de bordes, por ejemplo. Todo ello en una secuencia de mapas que constituirán el primer bloque de convolución. Luego, en secuencia de capas posteriores trataran de combinarlos en formas más simples y,

finalmente, en patrones de las diferentes posiciones de los objetos, iluminación, escalas, etc.

Las capas finales intentan hacer coincidir una imagen de entrada con todos los patrones para llegar a una predicción final como una suma ponderada de todos ellos, usando ya una red densa. Las redes convolucionales no dejan de ser redes neuronales, por tanto incluyen una capa de entrada, una capa de salida, y diferentes capas ocultas. Su especialización radica en las capas ocultas, las cuales se relacionan con operaciones de convolución, max-pooling o subsampling, y capas totalmente conectadas (fully-connected), (CMI, 2022).

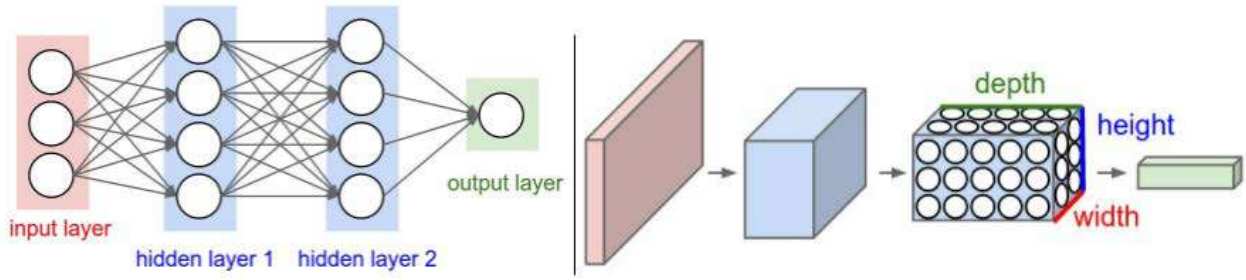

Figura 1.10 Ejemplo gráfico de la estructura de una red neuronal y sus principales características tales como los datos de entrada y las CNN a lo largo de la red.

1.6.1 Capas Convolutivas

Las capas convolutivas son la columna vertebral de las CNN (Convolutional Neural Networks) Consisten en un conjunto de filtros (kernels) que se desplazan sobre la imagen de entrada para calcular características locales , cada filtro es pequeño espacialmente (a lo largo de ancho y alto), pero se extiende a través de toda la profundidad del volumen de entrada. Estos filtros aprenden patrones visuales como bordes, texturas y formas. Los parámetros de la capa CONV consisten en un conjunto de filtros aprendibles.

Tal como en el ejemplo de la Figura 1.11 cada capa se va identificando con los elementos más importantes de la imagen; Por lo que, va aumentando el nivel de abstracción de esta, comenzando desde una capa de entrada para buscar líneas o figuras que en el ejemplo es el número más alto, perteneciente a una capa convolucional que busca los elementos elaborados como objeto y finalmente a una capa de salida, mostrandolo como una calsificación binaria al arrojar dos resultados, (CMI, 2022).

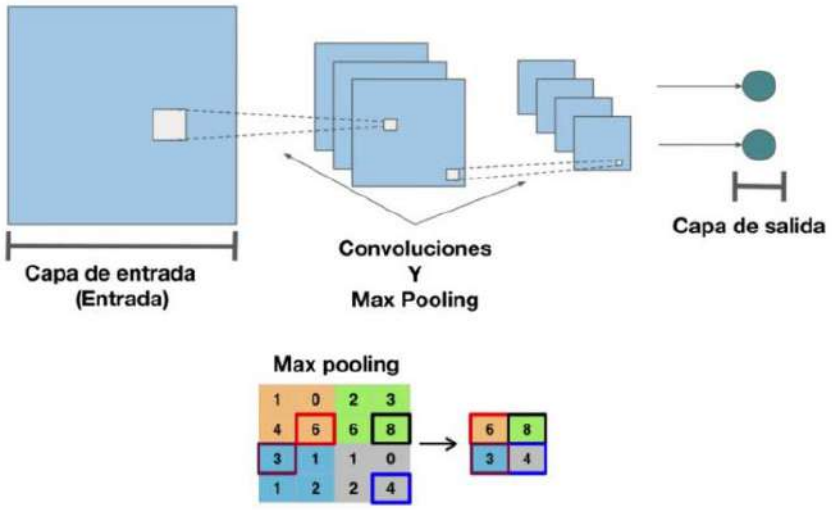

Figura 1.11 Ejemplo de la capa de entrada, capas Convolutivas, Max Pooling y la forma de extracción de datos hasta la capa de salida.

1.6.1.1 Capas de Agrupación (Pooling)

Las capas de agrupación reducen la dimensionalidad de las características extraídas por las capas convolucionales, reduciendo el tamaño espacial de la representación y proporcionando invarianza a pequeñas traslaciones y deformaciones en las características.

Es común insertar periódicamente una capa de agrupación entre capas sucesivas de Conv en una arquitectura de ConvNet. Su función es reducir progresivamente el tamaño espacial de la representación para reducir la cantidad de parámetros al igual que los cálculos en la red; Por lo tanto, controlar también el sobreajuste. La capa de agrupación funciona de forma

independiente en cada segmento de profundidad de la entrada y cambia su tamaño espacialmente, utilizando la operación MAX.

Un ejemplo gráfico de una capa de agrupación *Pooling* se encuentra en la **Figura 1.12** donde se puede observar una capa de tamaño 224x224x64 pixeles, esta se reduce en lo alto y ancho de la capa a fin de reducir parámetros aperte de cálculos en la red. En este proceso se realiza la extracción de las características además de los valores más altos extraídos, en el ejemplo tomaría los números 6, 8, 3, y 4 respectivamente en cada una de las capas.

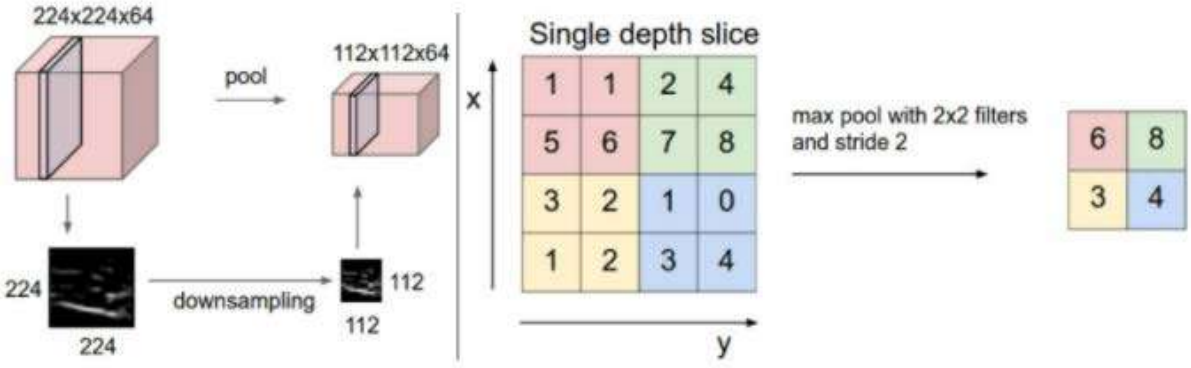

Figura 1.12 Ejemplo gráfico de una capa de agrupación (Pooling) utilizando la operación Max; es decir se extraen los valores máximos de cada capa que fue reducida en lo alto y ancho de esta.

1.6.1.2 Capas Completamente Conectadas

Después de extraer características relevantes mediante capas convolucionales y de agrupación, se utilizan capas completamente conectadas para realizar la clasificación final. Estas capas aprenden a combinar las características extraídas para generar predicciones sobre las clases de las imágenes, (CMI, 2022).

En la Figura 1.13 se muestra un ejemplo de una arquitectura convolucional completa siendo el ejemplo más conocido de las redes neuronales, donde se identifica el número ingresado por un usuario a mano, este será capaz de identificarlo. Esta red muestra cada capa al

igual que su respectivo tamaño, agrupación, comenzando desde una entrada, capas convolucionales, MaxPooling, y finalmente Full Connect para arrojar una salida con los resultados esperados.

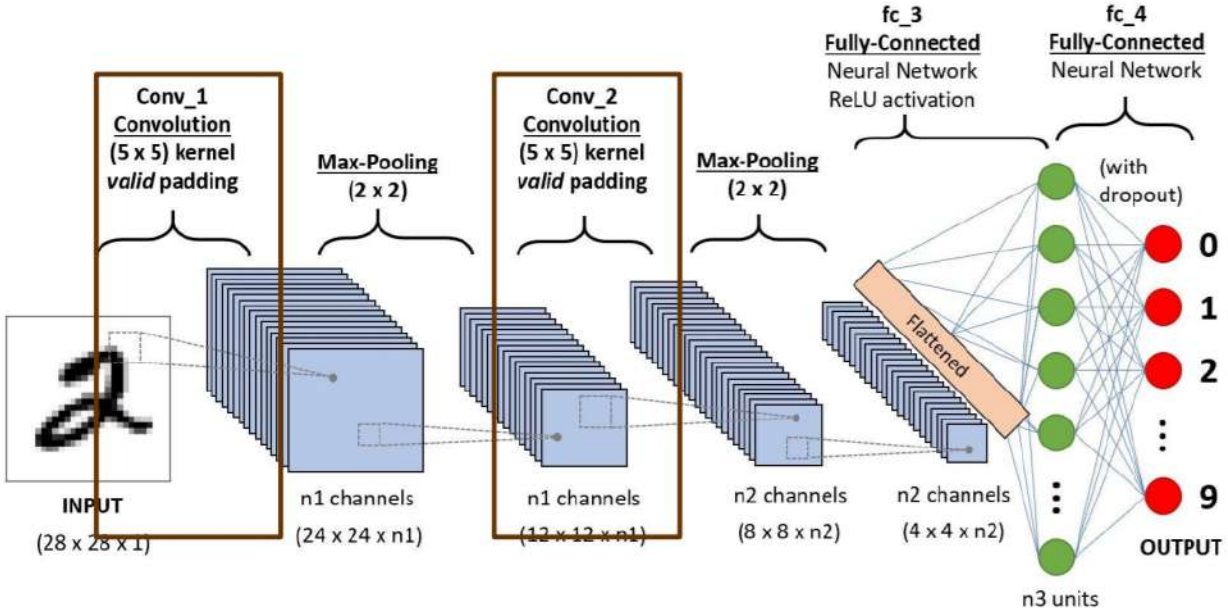

Figura 1.13 Arquitectura de una red convolucional básica tomando como ejemplo de agrupación.

1.6.2 1.6 Funciones de Activación

Las funciones de activación, como ReLU (Rectified Linear Unit), se aplican típicamente después de las operaciones de convolución y agrupación para introducir no linealidad en la red para permitirle aprender relaciones más complejas en los datos. La Figura 1.14 muestra el uso de la función de activación en nuestra red neuronal de forma matemática, el lugar que ocupa dentro de la red, sus caracteristicas como las capas de entrada y pesos.

La función de activación (también llamada función de transferencia, su rango de salida es limitado este se conoce también como función de aplastamiento) se define cómo la suma ponderada de la entrada (con pesos), se transforma en una salida de un nodo o nodos en una capa de la red. La elección de la función de activación tiene un gran impacto en la

capacidad, el rendimiento de la red neuronal, por lo cual podemos utilizar diferentes funciones de activación en distintas partes el modelo.

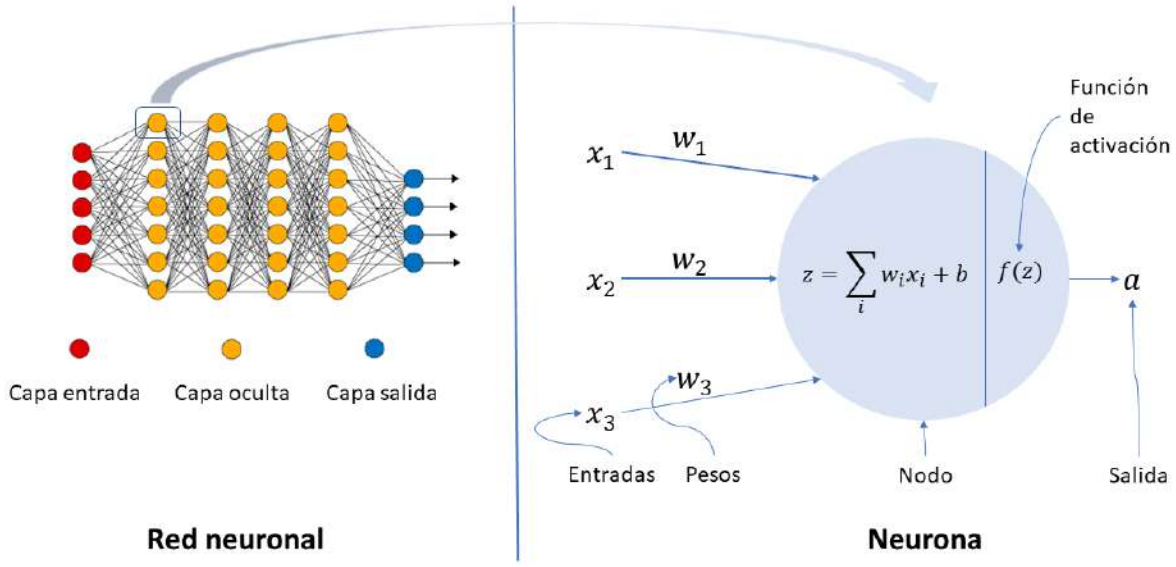

Figura 1.14 Función de activación en una red neuronal considerando las entradas, pesos, nodos, función de activación y salida de la red neuronal.

Por lo general, todas las capas ocultas suelen utilizar la misma función de activación. La capa de salida normalmente utiliza una función de activación distinta y depende del tipo de problema de aprendizaje supervisado (regresión o clasificación, …).

Las funciones de activación deben ser diferenciables. Esto es necesario dado que las redes neuronales se entrenan generalmente utilizando el algoritmo backpropagation donde se requiere la derivada de primer orden del error de predicción para actualizar los pesos del modelo, (Soberanis, 2020).

1.6.3 Diferentes funciones de activación

1.6.3.1 Sigmoide

En los inicios de las redes neuronales se utilizó la función de activación sigmoide debido a la propiedad que tiene de mapear el dominio de los reales al intervalo (0,1), con lo cual se asemeja a una distribución de probabilidad, ampliamente utilizada para clasificaciones binarias. La gráfica de la función de activación se muestra en la **Figura 1.**15 (Soberanis, 2020).

$$sigma(z) = 1(1 + e\text{^}(-z))$$

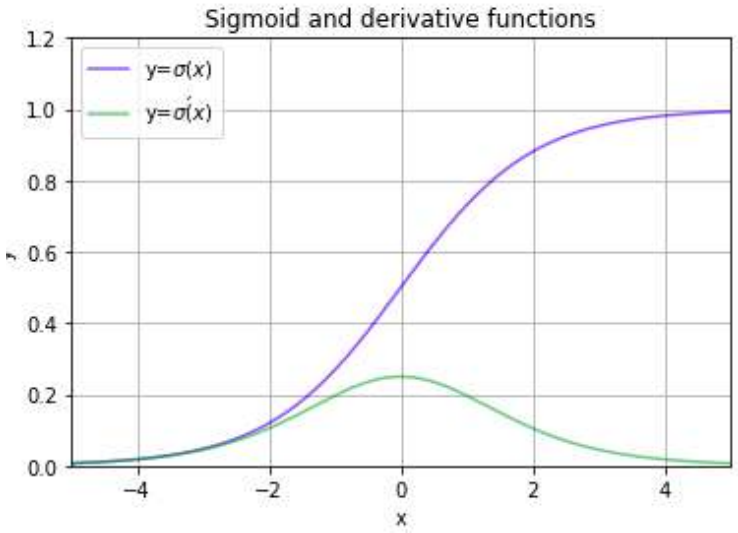

Figura 1.15 Representación gráfica de la función Sigmoide y la derivada de la función**.**

Características:

- La función es una curva en forma de S.
- La salida de la función se centra en 0.5, con un rango de 0 a 1
- Usada como capa final en un problema de clasificación binaria.
- Es una función diferenciable.
- Aunque la función es monótona, su derivada no lo es.

Desventajas:

- Fuga de gradiente: cuando las entradas se vuelven pequeñas o grandes, la función se concentra en 0 o 1, trayendo una derivada cercana a 0. Esto trae que no tenga gradiente a propagarse a través de la red, por lo que casi no queda nada para las capas inferiores.
- Costoso computacionalmente: al tener operaciones exponenciales, (Ponce, 2021).

1.6.3.2 ReLU (Rectified Linear Units):

Es la función de activación más utilizada en el aprendizaje profundo. Es una función extremadamente rápida de calcular ya que ha demostrado su efectividad en varios dominios de aplicación ayudando a mitigar el problema de la saturación de gradiente cuando los valores de la combinación lineal de la entrada son mayores a 0, tal como se muestra en la Figura 1.16 permitiendo una propagación hacia atrás efectiva incluso en redes profundas. Para mitigar el problema de colapso de las neuronas para gradientes negativos se propone una estrategia conocida como Leaky ReLU, la cual utiliza un parámetro alpha para añadir cierta pendiente que permita la propagación de gradientes negativos, (Soberanis, 2020).

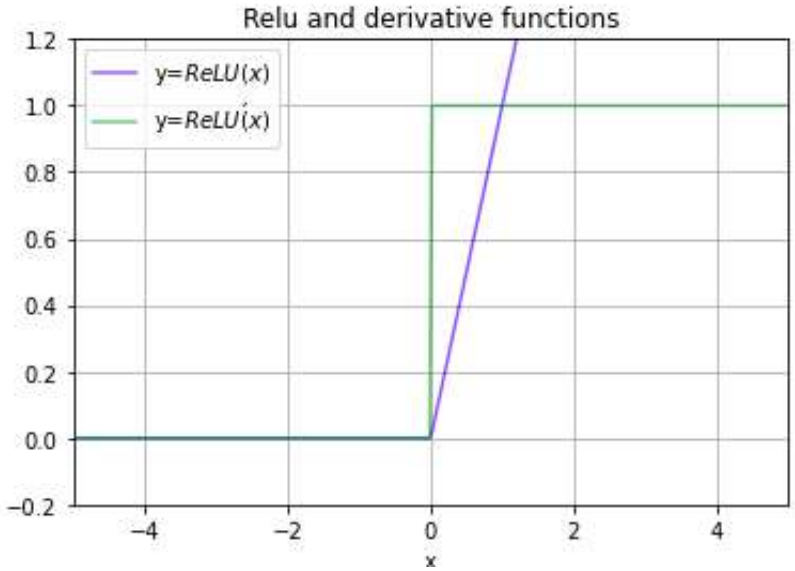

Figura 1.16 Representación gráfica de la función ReLU y la derivada de la función.

Ventajas:

- Activación escasa: por ejemplo, en una red inicializada aleatoriamente, alrededor del 50% de las unidades ocultas están activadas (tienen una salida distinto a cero).
- Mejor propagación del gradiente: menos problemas de fuga de gradiente en comparación con las funciones de activación sigmoide y thanh.
- Cálculo eficiente: ya que solo es comparación, suma y multiplicación.
- Invariante en escala: max(0,a*x)=a*max(0,x)\text{ para }a\geq0.
- Es una función más rápida de cálcular (a diferencia con sigmoide y tanh).

Desventajas:

- No está centrado en cero.
- Es diferenciable en cualquier valor, pero no en 0, el valor de la derivada en este número puede elegirse arbitrariamente a ser 0 o 1.
- ReLU moribundo: Durante el entrenamiento, algunas neuronas mueren efectivamente, lo que significa que dejan de producir algo que no sea 0. En algunos casos, puede encontrar que la mitad de las neuronas de su red están muertas, especialmente si utilizó una tasa de aprendizaje alta.
- Una neurona muere cuando sus pesos se modifican de tal manera que la suma ponderada de sus entradas es negativa para todas las instancias del conjunto de entrenamiento. Cuando esto sucede, sigue generando ceros y el descenso del gradiente ya no lo afecta, ya que el gradiente de la función ReLU es 0 cuando su entrada es negativa.
- Explosión de activaciones: ya que no tiene límite superior, es infinito. Esto a veces conduce a nodos inutilizables, (Ponce, 2021).

1.6.3.3 Softmax

Es la forma más generalizada de la función de activación sigmoide. Se utiliza en problemas de clasificación de múltiples clases. Similar a la sigmoide, los elementos del vector de salida están en el rango 0-1, la diferencia es que realiza la exponencial de cada elemento y la divide entre la suma de las exponenciales del vector su suma es igual a 1. Se utiliza como capa final en los modelos de clasificación.

El resultado es que esta función de activación traduce las salidas de la capa previa a un vector como se observa en la Figura 1.17 que simula la distribución de probabilidades de pertenencia a las diferentes clases en donde la mayor "probabilidad" que en el ejmeplo seria el 0.8412 este se asigna de acuerdo a la intensidad de la entrada con respecto de los valores del vector.

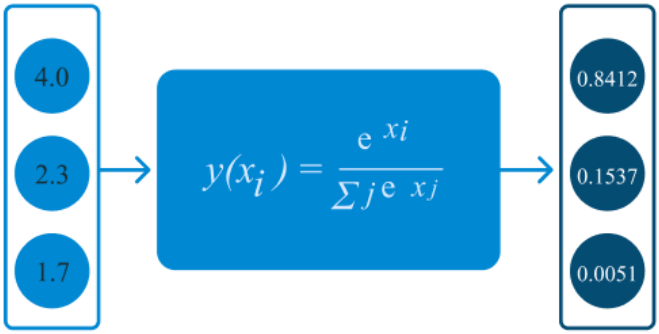

Figura 1.17 Descripción matemática de la función Softmax con su vector de salida 0-1.

1.6.3.4 Tanh

La tangente hiperbólica se puede ver como una versión escalada y desplazada de la sigmoide. De hecho se pude escribir la función de tangente hiperbólica en términos de la sigmoide. En la Figura 1.18 se puede observar como por medio de la función Tanh mapea el dominio de los reales al intervalo abierto (-1, 1). Debido a lo anterior se encuentra centrada en 0 y elimina el problema del covariate shift. De igual forma su derivada es más "activa" que la sigmoide, lo cual perimite reducir tiempos de entrenamiento, convergencia;

Sin embargo, también sufre de saturación de gradiente a los extremos de la función: (Soberanis, 2020).

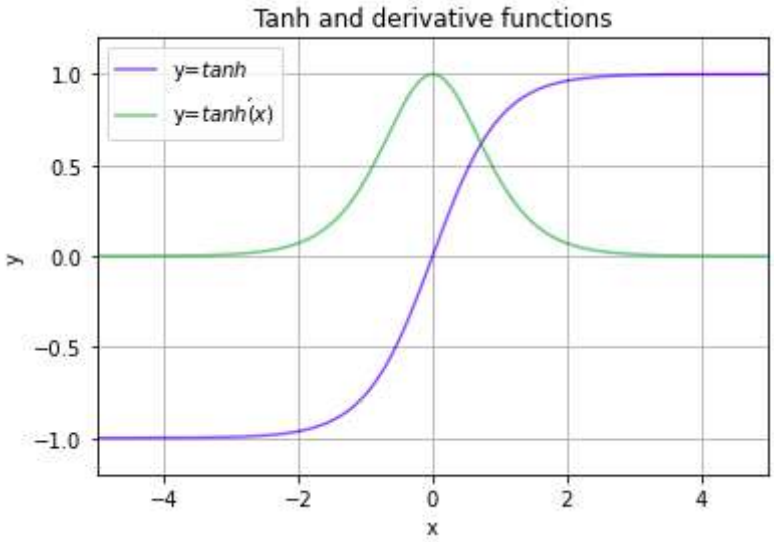

Figura 1.18 Representación gráfica de la función Than y la derivada de la función.

Características:

- La función también tiene una curva en forma de S, similar a la función de activación logística.
- A diferencia de la función de activación sigmoide, esta función está centrada en 0, con un rango de -1 a 1.
- Similar a la función sigmoide, esta función es diferenciable.
- Similar a la función sigmoide, esta función es monótona, pero su derivada no lo es.

Desventajas:

- Fuga de gradiente: similar a la función sigmoide. Cuando las entradas se vuelven pequeñas o grandes, la función se concentra en 0 o 1, trayendo una derivada

cercana a 0. Esto trae que no tenga gradiente a propagarse a través de la red, por lo que casi no queda nada para las capas inferiores.

- Costoso computacionalmente: al tener operaciones exponenciales, (Ponce, 2021).

1.7 Optimización y Entrenamiento

Durante el entrenamiento, la red se ajusta iterativamente a los datos de entrenamiento utilizando un algoritmo de optimización, como Adam o SGD (Stochastic Gradient Descent), que minimiza una función de pérdida, como la entropía cruzada categórica, entre las predicciones del modelo y las etiquetas verdaderas de las imágenes.

1.7.1 Evaluación y Validación

Después del entrenamiento, el modelo se evalúa en un conjunto de datos de validación independiente para medir su rendimiento en datos no vistos donde se ajustarán los hiperparámetros si es necesario. Finalmente, se prueba en un conjunto de datos de prueba para evaluar su capacidad para generalizar a nuevos datos.

1.7.2 Visualización y Análisis

La visualización de métricas de rendimiento, como la precisión, la pérdida a lo largo del entrenamiento, así como el análisis de las predicciones del modelo en datos de prueba, ayudan a comprender su comportamiento y calidad de las predicciones.

1.7.3 Despliegue y Uso

Una vez que el modelo ha sido entrenado, evaluado satisfactoriamente, puede ser desplegado y utilizado en aplicaciones del mundo real para hacer predicciones sobre nuevas imágenes.

1.8 Multilayer perceptrón

El perceptrón, creado por Frank Rosenblatt en 1958, es el modelo más simple de red neuronal artificial, compuesto por una sola neurona. Funciona recibiendo múltiples entradas, cada una con un peso, que se multiplican y suman. La suma resultante se pasa a través de una función de activación, como la función escalón, que produce una salida binaria, 0 o 1. Durante el aprendizaje, el perceptrón ajusta sus pesos mediante un algoritmo supervisado para minimizar errores, de esta manera mejora su capacidad de clasificación tal como se meustra en la Figura 1.19.

Sin embargo, el perceptrón solo puede resolver problemas linealmente separables, como distinguir entre puntos dentro y fuera de un círculo en un plano, donde una línea puede separar ambas clases. Para resolver problemas más complejos, como el famoso problema XOR (donde las clases no pueden ser separadas por una línea recta), se utiliza el perceptrón multicapa.

Este modelo tiene múltiples capas de neuronas (entrada, ocultas y salida) que usa neuronas sigmoides en lugar de perceptrones simples, permitiendo manejar la no linealidad y resolver problemas más complicados.

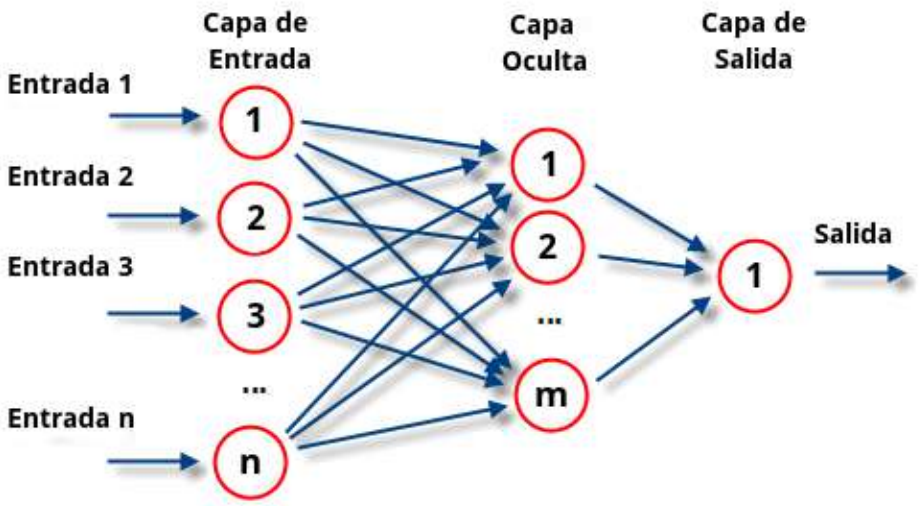

Figura 1.19 Estructura de un perceptrón multicapa, donde se observa la capa de entrada, una capa oculta y una capa de salida.

Un ejemplo práctico es el reconocimiento de dígitos escritos a mano tal como lo vimos en la Figura 1.13. Un perceptrón simple puede diferenciar entre dos tipos de líneas rectas, pero un perceptrón multicapa puede aprender a reconocer patrones complejos en imágenes de dígitos, permitiendo su correcta clasificación.

En resumen, aunque el perceptrón básico es limitado, su concepto es fundamental y ha llevado al desarrollo de redes neuronales más sofisticadas que son la base de muchas aplicaciones modernas en inteligencia artificial.

1.9 Programador PYTHON

Para la programar se uso Python, ya que es un lenguaje de programación interpretado, cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa, aparte en menor medida, programación funcional, además este lenguaje posee una licencia de código libre, puede contar con amplias bibliotecas de librerías que permiten la aplicación a la inteligencia artificial, (Python, s.f.).

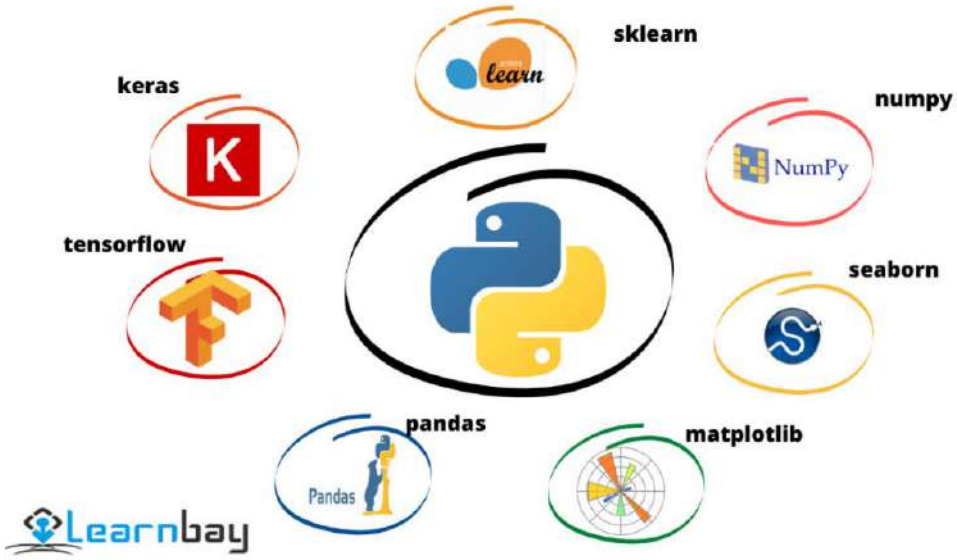

Figura 1.20 Python y sus librerías más destacadas y empleadas para la IA.

Las librerías más destacables, útiles de Python para la inteligencia artificial (IA) y que se usaron en este proyecto son:

1.9.1 Visualización

- **Seaborn**: es una librería basada en matplotlib, pero orientada a la visualización de datos estadísticos. Posee una interfaz gráfica de alto nivel para datos estadísticos.
- **Matplotlib:** Es una librería muy importante que trabaja con Python y una de las más conocidas ya que sirve para generar gráficos de alta calidad. Se pueden realizar diagramas, histogramas, graficas, etc. (InGenio, s.f.)
- **NumPy**: Permite una estructuración de datos de forma universal, ideal para su análisis y la interacción con los algoritmos. Implementa diversos vectores multidimensionales y matrices que pueden manejar altos volúmenes de datos.

1.9.2 Machine Learning

- **Scikit-learn:** Esta librería está basada en NumPy, SciPy y Matplotlib. Se utiliza para el aprendizaje automático supervisado y no supervisado que implementa.

1.9.3 Deep Leraning

- **TensorFlow:** es una biblioteca desarrollada por Google, que permite cálculos numéricos mediante diagramas de flujo de datos para codificar un gráfico. Los nodos de este gráfico pasan a ser operaciones matemáticas y las aristas representan los tensores (matrices de datos multidimensionales).
- **OpenCV:** Se utiliza mucho cuando son redes neuronales Pre – entrenadas utilizando otras funciones junto a Python gracias a que utiliza el módulo dnn.
- **Keras:** Es una interfaz de alto nivel que permite trabajar con redes neuronales. Es más fácil de usar que TensorFlow. Esta interfaz es ideal para verificar si los desarrollos obtendrán los resultados esperados rápidamente. Keras utiliza otras librerías de deep learning (TensorFlow, CNTK o Theano) de forma transparente para hacer el trabajo que se indique, (InGenio, s.f.).

1.10 División de datos

En Machine Learning, típicamente la información se suele dividir en datos para el entrenamiento, validación y prueba. La información de entrenamiento así como la validación, se utilizan para entrenar el modelo o máquina, definir la arquitectura de esta y su sintonización. La información de prueba se emplea para estimar el desempeño final del modelo previamente entrenado aparte sintonizado con la información nunca vista por el algoritmo.

Típicamente se suele dividir la información en 70% para el *entrenamiento*, utilizado para actualizar los pesos a cada *batch*; un 15% pertenece a la *validación* donde comprueba la

capacidad de generalización de la red en cada época; Por lo tanto, a mayor precisión en la validación, existe mejor set de parámetros.

Por último el 15% para *prueba*, parte donde da la intuición de lo buena que es una red al generalizar con un conjunto (más grande que el de validación). Sin embargo, esto no es una regla ya que estos porcentajes pueden variar dependiendo del problema al igual que de la cantidad de información disponible estos datos se pueden observar de manera más grafica en la Figura 1.21, (KeepCoding, 2022).

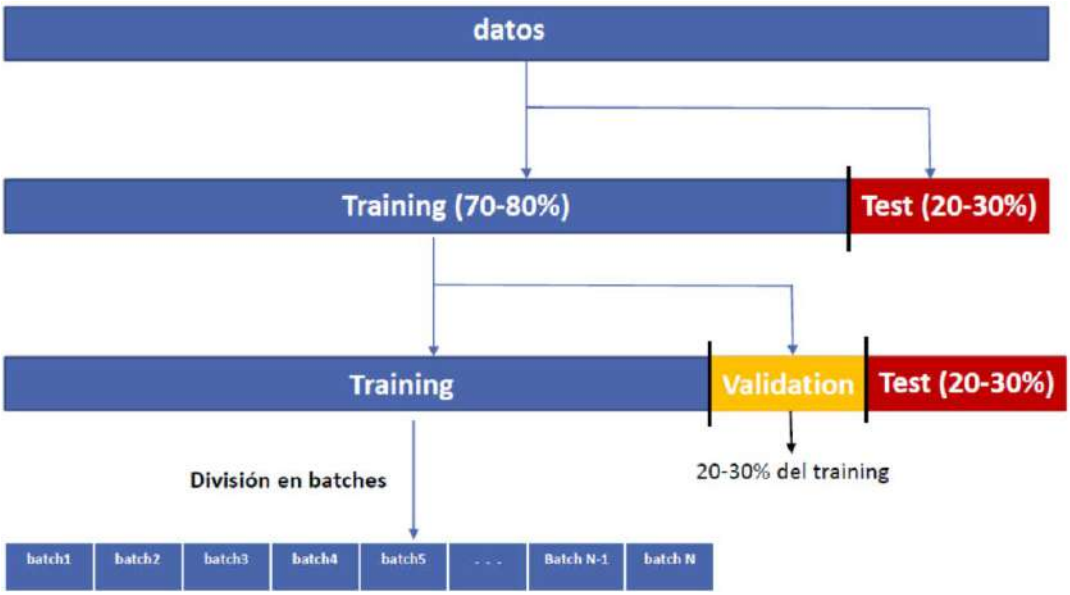

Figura 1.21 Esquemático de la división de datos en Deep Learning, en la etapa de entrenamiento se considera del 60 al 70 % de los datos, en la etapa de validación del 20 al 30% de los datos y en la etapa de prueba se considera del 20 al 30%.

Por último, se llegó al desarrollo de una interfaz gráfica desarrollada en el lenguaje de programación Python. La interfaz gráfica conocida como Graphical User Interface (GUI) es definida como la forma en que un usuario puede interactuar con un dispositivo informático sin introducir comandos de texto ya que al ser un entorno visual permite al usuario realizar cualquier acción sin necesidad de tener conocimientos de programación.

CAPÍTULO 2. GENERALIDADES

2.1 Planteamiento del problema

Los sistemas de reconocimiento facial actualmente presentan tasas elevadas de reconocimiento pero siguen presentando desafíos debido a las múltiples variables y complejidad del rostro humano por ello se han desarrollado diferentes técnicas que ayudan a mejoramiento de eficiencia con respecto a sus aplicaciones , otro de los problemas más comunes es debido a los equipos de un alto costo ya que impiden su comercialización, porque al ser un sistema automático de reconocimiento su precisión puede no ser la mejor ya que es afectado por las variaciones en las imágenes de rostros, como la iluminación, expresión, pose, entre otras.

2.2 Hipótesis

Se espera la comprobación de la eficacia de la técnica de capas convolucionales aplicada al desarrollo y entrenamiento de la red neuronal para el reconocimiento facial a medida que aumente la optimización de la red neuronal con éxito, reduciendo el error en las clasificaciones de las imágenes logrando una mayor exactitud.

2.3 Justificación

El avance constante en el campo del reconocimiento facial, respaldado por el desarrollo tecnológico y la proliferación de la inteligencia artificial (IA), ha generado un gran interés debido a su amplio espectro de aplicaciones. Desde la autenticación de identidad hasta la vigilancia y el control de accesos, las posibilidades son vastas y de gran relevancia en diversos sectores.

Es dentro de este contexto que surge la necesidad de este proyecto. Nos proponemos no solo aprovechar las capacidades de las redes neuronales, sino también poner a prueba su eficacia mediante un riguroso proceso de entrenamiento, pruebas y validación. El objetivo principal es mejorar el reconocimiento facial mediante el desarrollo de una interfaz gráfica intuitiva y de fácil acceso para el usuario. La importancia de este proyecto radica en su

potencial para contribuir significativamente al avance de la tecnología de reconocimiento facial. Al validar y mejorar la eficacia de las redes neuronales en esta área, podemos abrir nuevas puertas para su aplicación en una variedad de escenarios, desde la seguridad en instalaciones hasta la identificación biométrica en dispositivos personales.

Además, al centrarnos en la creación de una interfaz gráfica amigable, buscamos democratizar el acceso a esta tecnología, permitiendo que usuarios de diferentes niveles de habilidad puedan aprovechar sus beneficios de manera efectiva y sin complicaciones.

2.4 Objetivos

2.4.1 Objetivo general

Desarrollar una interfaz gráfica para el reconocimiento de rostros implementando modelos de redes neuronales convoluciones mediante el software programable Python.

2.4.2 Objetivos específicos

- Realizar base de dato de imágenes de individuos y su procesamiento.
- Implementar OpenCV para el reconocimiento facial a pesar de los diferentes desafíos que tienen los algoritmos de visión.
- Desarrollar y optimizar un modelo de red neuronal para el reconocimiento de rostros que proporcione información más precisa considerando el tipo de aplicación empleada.
- Diseñar y optimizar interfaz gráfica para la manipulación del reconocimiento facial.

CAPÍTULO 3. DESARROLLO Y METODOLOGÍA

El desarrollo de este proyecto se comenzó con la introducción a la inteligencia artificial y su implementación en el reconocimiento facial, iniciando desde la verificación facial por medio de la comparación de entre diversos rostros confirmando una sola identidad que recive datos biométricos gracias a una base de datos, para poder continuar con la identificación facial, determinando la identidad de una nueva imagen o rostro ingresado, por ello podemos decir que el reconocimiento facial al menos se divide en cuatro etapas principales que se llevaran a cabo como anteriormente lo mencionamos, comenzando con la detección facial, normalización y alineamiento, extracción de características y por último el reconocimiento.

Este proceso llevara consigo un algoritmo y una función de activación que ayudara al mejoramiento o el entrenamiento de este para disminuir en gran medida los errores al momento de la detección y reconocimiento, por ejemplo, se implementarón redes convolucionales para el reconocimiento facial, siendo una arquitectura de red para Deep Learning que aprende directamente a partir de datos. El procedimiento general es el uso de los modelos como clasificadores siendo entrenados en base a datos con variantes de imágenes de diferentes identidades que más tarde serán las clases por predecir del modelo obtenido y después se obtendrán los vectores de características que ayudarán a realizar la comparación uno a uno requerida en la verificación facial.

Todo el procedimiento previo se podrá mostrar sus resultados en 3 grandes partes investigación comenzando desde la división de la información que estará dada en un 75% destinada al entrenamiento ,15% de validación y un 15% de pruebas, y esperando finalizar con la creación de una interfaz gráfica para la manipulación del reconocimiento de facial. Así mismo a continuación se mostrarán los pasos a seguir en este proyecto, si bien nos centraremos en los puntos clave que llevarón a cabo la culminación de este proyecto.

3.1 Captura de datos

La captura de rostros es una etapa crucial en el proceso de entrenamiento de modelos de reconocimiento facial. Para dicho proyecto se usó la librería OpenCV y el clasificador Haar Cascade ya que fue el principal detector, con el podemos detectar aparte de extraer rostros

de imágenes, videos de manera eficiente. Ya que OpenCV, es una biblioteca de visión por computadora de código abierto que ofrece herramientas robustas para la manipulación, al igual que análisis de imágenes tal como se menciono en el subtema 1.2.1 Detección Facial destacando las ventajas de uso uso y también del detector por medio de redes convolucionales.

En este proceso, se capturaron rostros en diferentes condiciones, incluyendo variaciones en iluminación, poses más la presencia de accesorios, para crear un conjunto de datos diversificado y representativo. Estos datos se utilizaron posteriormente para entrenar una red neuronal, mejorando su capacidad para reconocer rostros en situaciones del mundo real. A continuación, se detallo el procedimiento para la captura de rostros utilizando OpenCV más Haar Cascade, así mismo cómo estos datos se prepararon para el entrenamiento del modelo. Si bien, es oportuno se ha de recalcar que la obtención de datos de este proyecto principalmente fue extraída de videos pregrabados directamente, así mimo a continuación se explicán los pasos a seguir para la detección:

Se inicio con la instalación de librerías esenciales para la captura tal como CV2 ya que esta se utilizo para la captura así mismo para el procesamiento de las imágenes y videos, mientras que OS gestiono los archivos resultantes integrando sin problema el procesamiento de imágenes para que finalmente Imutils fue la que se encargo de simplificar las transformaciones geométricas en las imágenes antes de aplicar las técnicas avanzadas de CV2. Después de ello, se cargo el clasificador Haar Cascade de OpenCV donde se ingreso el video procesado, donde se capturo el rostro visualizándolo con un recuadro verde. Posteriormente, se recorto la imagen en un tamaño de 150x150 píxeles, a continuación se almaceno con una etiqueta específica en aumento, por ejemplo, Juan_1, Juan_2, y así sucesivamente. Dentro de este procedimiento de captura, fue importante considerar las siguientes características al momento de iniciar la captura:

- *Diferentes Ambientes:* Capturar imágenes en diversas condiciones de iluminación, como día, tarde, noche, ambientes claros y oscuros.
- *Posiciones o Expresiones Variadas:* Incluir diferentes ángulos y expresiones faciales frente a la cámara o el video para captar una variedad de perspectivas.

- *Accesorios:* Incluir fotos con accesorios como lentes, gorra, cubrebocas, maquillaje, peinados, etc., para aumentar la diversidad del conjunto de datos.

Estas capturas se almacenaron en un formato adecuado como (JPEG o PNG) y se dividieron en tres partes para el entrenamiento del modelo: entrenamiento (Training), prueba (Testing) y validación (Validation). Esto aseguro que el modelo se entrene, pruebe, valide con un conjunto de datos representativo además diversificado, mejorando su capacidad para reconocer rostros en diferentes condiciones y escenarios.

3.1.1 Red Neuronal

La creación de la red neuronal forma parte del segundo y más importante paso ya que aquí incluyen, aparte de que se modificarón las características del modelo. Las redes neuronales son modelos computacionales inspirados en el funcionamiento del cerebro humano, capaces de aprender patrones complejos a partir de grandes conjuntos de datos. Este proceso metodológico aseguro que el modelo resultante sea robusto, eficiente y adecuado para la tarea específica que se deseo resolver.

Se comenzó con la inserción de librerías tal como: Matplotlib que fue utilizada para generar gráficas aparte de desplegar las imágenes, OS permitió interactuar con el sistema operativo para manejar los directorios al igual que los archivos, OpenCv fue fundamental para la captura, procesamiento de imagenes y videos. *Numpy* fue utilizada para generar tanto como, manipular tensores (matrices de datos). *Keras* es utilizada para construir aparte de entrenar redes neuronales esta misma define igual que entrena la red neuronal convolucional CNN así como también la generación de lotes de imagenes durante el entrenamiento. Finalmente *TensorFlow* es la librería subyacente que permite implementar redes neuronales aparte de ejecutar los modelos.

Después de contar con el entorno de programación se vinculo la correcta información de entrenamiento tal como la obtenida con anterioridad en la base de datos, por consiguiente, arrojó un despliegue de información de entrenamiento con las respectivas etiquetas tal como se muestra en Figura 3.1.

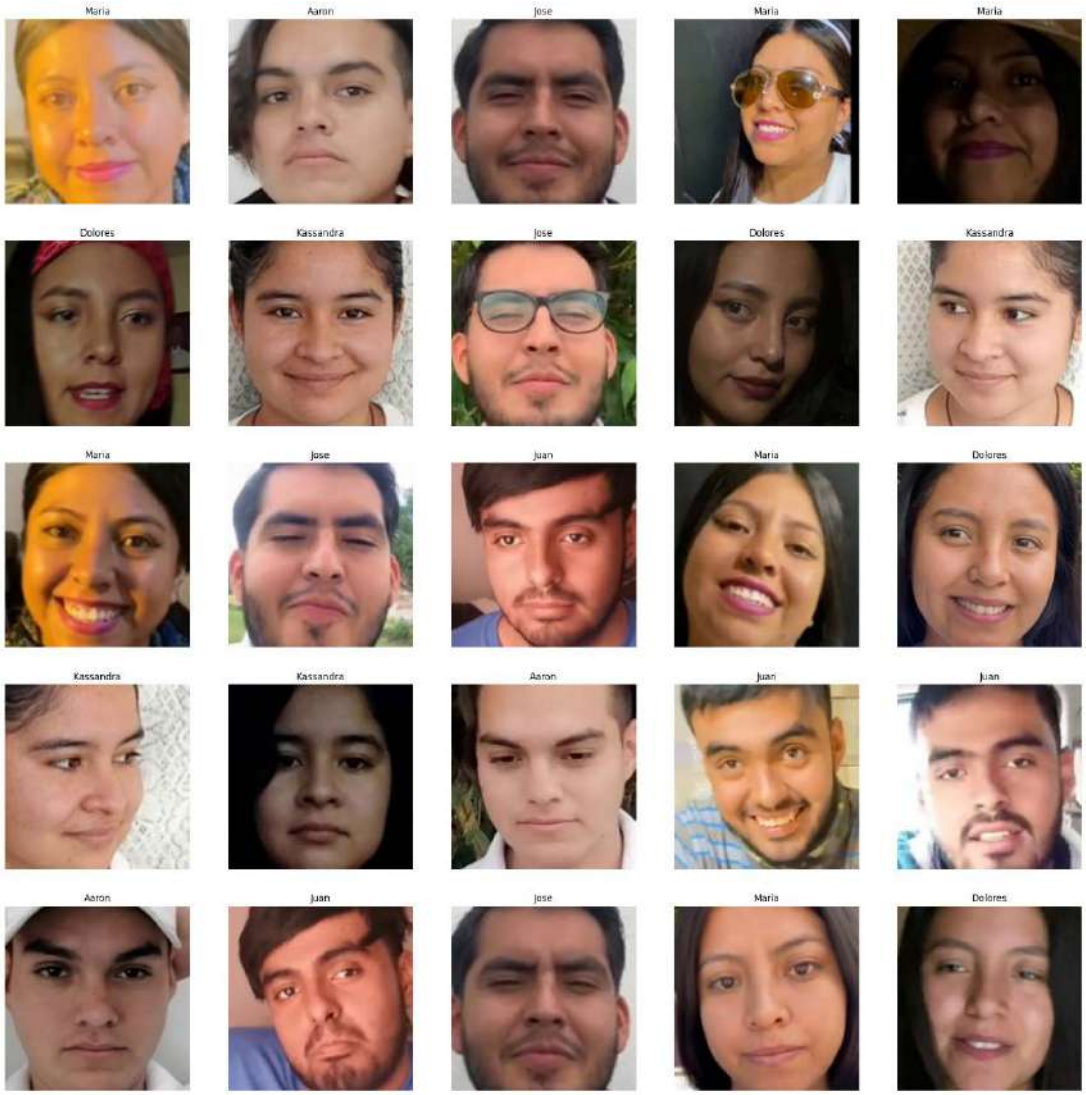

Figura 3.1 Ejemplo de batch de muestra de la información utilizada para el entrenamiento.

Posteriormente se continuo con el reprocesamiento, lo mismo que el aumento artificial de la información donde con esta normalización se adapto, además de mejorar para que el entrenamiento sea más eficiente y por consiguiente el reconocimiento facial.

Los siguientes pasos para la normalización fueron:

- Se importo la librería tensorFlow.keras.reprocessing.image
- Se define el tamaño de lote = 25

- Se configuró el generador de imágenes para el entrenamiento.

  - rescale=1.0/255.0
  - zoom_range=0.3 # Zoom aleatorio hasta un 30%
  - rotation_range=20 # Rotación aleatoria hasta 20 grados
  - shear_range=0.3 # Corte aleatorio hasta 30 grados
  - horizontal_flip=True # Volteo horizontal aleatorio
  - vertical_flip=True #Volteo vertical aleatorio
  - brightness_range=[0.5, 1.5] # Rango de brillo aleatorio entre 0.5 y 1.5
  - channel_shift_range=0.1 # Desplazamiento aleartorio

La Configuración del generador de imágenes para la validación, prueba, entrenamiento como la generación de datos para el entrenamiento donde redireccionamos el directorio con estos datos, después se redimensionaron las imágenes (target_size = 150 x 150), posteriormente el tamaño de lote y finalmente el modo de clasificación categórical.

A continuación en la Figura 3.2 se muestrá de forma visual procesamiento de las imágenes con las características antes vistas, siendo esta una de las partes más importantes de las mejoras del modelo, ya que al otorgarle al entrenamiento una imagen bien procesada fue capaz de detectar mejor las características de cada modelo para que al final por medio de los filtros o capas en el modelo llegó a una predicción lo más acercada posible.

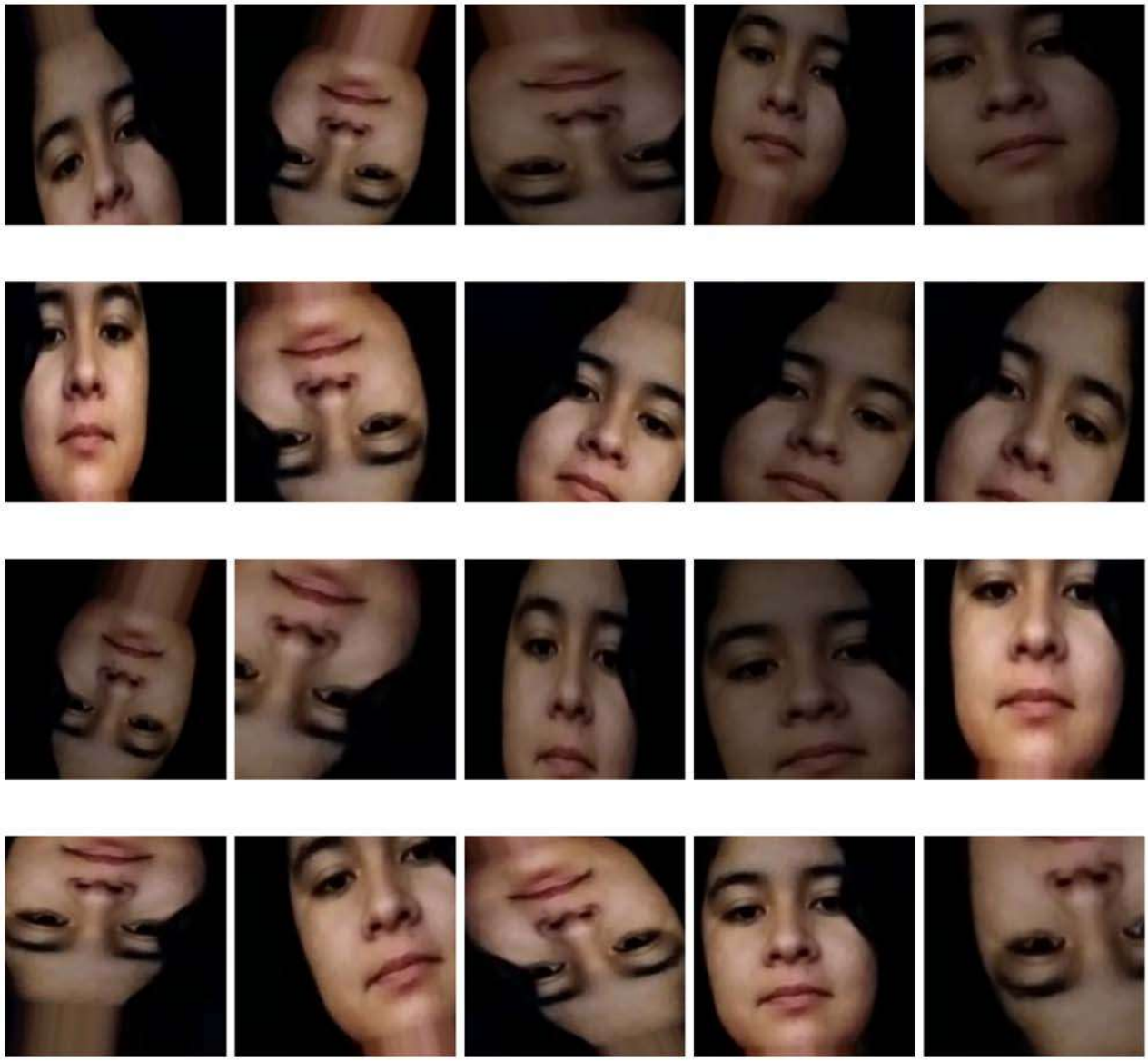

Figura 3.2 Ejemplo de la normalización de la información y sus diferentes características aplicadas a los rostros de cada persona.

3.1.1.1 Modelo

Para la creación del modelo fue importante tener claro el tipo de problema enfrentado, aquello genero cambios en la red actual ya que solamente funcionaba de forma binaria, por lo que se adaptó a un multi clasificador, y que, por medio de las funciones de activación, perdida, o evitando un reajuste en el modelo se ejecutará el modelo. Si bien para la creación de esta red se tiene

principalmente que importar las librerías necesarias para después definir la arquitectura e inicializar el modelo secuencial. Para ello, se creó una instancia del modelo mismo que se muestra a continuación en la Figura 3.3 donde más adelante en la Tabla 3-1 se muestra la representación gráfica del modelo con los parámetros específicos de las capas considerando los filtros, su tamaño, función de activación y finalmente un conteo de los parámetros que son entrenados.

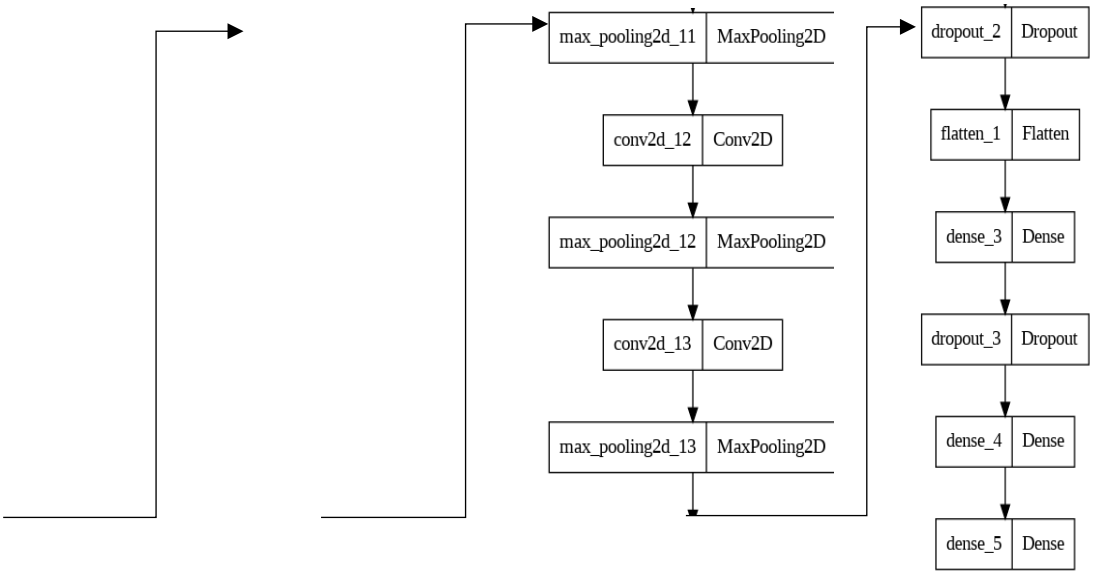

Figura 3.3 Descripción gráfica final del modelo utilizado en el proyecto tras las modificaciones aplicadas a lo largo de todo el proceso.

Tabla 3-1 Representación gráfica del modelo final con las especificaciones pertinentes de las capas utilizadas en el modelo implementado.

	Model: "sequential_1"		
	Layer (type)	Output Shape	Param #
1	conv2d_7 (Conv2D)	(None, 148, 148, 32)	896
	max_pooling2d_7(MaxPooling2D)	(None, 74, 74, 32)	0
2	conv2d_8 (Conv2D)	(None, 74, 74, 64)	18496
	max_pooling2d_8(MaxPooling2D)	(None, 37, 37, 64)	0
3	conv2d_9 (Conv2D)	(None, 37, 37, 64)	36928
	max_pooling2d_9(MaxPooling2D)	(None, 18, 18, 64)	0
4	conv2d_10 (Conv2D)	(None, 18, 18, 128)	73856
	max_pooling2d_10(MaxPooling2D)	(None, 9, 9, 128)	0
5	conv2d_11 (Conv2D)	(None, 9, 9, 128)	147584
	max_pooling2d_11 (MaxPooling2D)	(None, 4, 4 , 128)	0
6	conv2d_12 (Conv2D)	(None, 4, 4, 128)	147584
	max_pooling2d_12 (MaxPooling2D)	(None, 2, 2, 128)	0
7	conv2d_13 (Conv2D)	(None, 2 , 2 , 256)	295168
	max_pooling2d_13 (MaxPooling2D)	(None, 1, 1, 256)	0

8	dropout_2 (Dropout)	(None, 1, 1, 256)	0
9	flatten_1 (Flatten)	(None, 256)	0
10	dense_3 (Dense) dropout_3 (Dropout)	(None, 512) (None, 512)	131584 0
11	dense_4 (Dense)	(None, 256)	131328
12	dense_5 (Dense)	(None, 6)	1542

Total params: 984,966
Trainable params: 984,966
Non-trainable params: 0

3.1.1.2 Compilación del Modelo

Para la compilación de modelo se comenzó por definir el optimizador al igual que la función de pérdida, una vez selecionado el optimizador Adam con una tasa de aprendizaje adecuada (por ejemplo, 0.0001), posteriormente se utilizó la función de pérdida categorical_crossentropy para problemas de clasificación multiclase de modo que se establecen métricas de evaluación, como la precisión (accuracy), para finalmente dar paso al entrenamiento.

3.1.1.3 Entrenamiento

El entrenamiento se inicio con la preparación de datos para ello se configuró generadores de datos utilizando ImageDataGenerator para cargar imágenes además de eso aplicar preprocesamiento en tiempo real definiendo los directorios que contienen las imágenes de entrenamiento, validación así pues, finalmente ajustar las transformaciones necesarias

como la normalización de imágenes, aumentación de datos, etc. A continuación se paso a la configuración del entrenamiento donde los puntos clave a considerar fueron:

- Establecer el número de épocas en 25.
- Calcular steps_per_epoch y validation_steps.
- steps_per_epoch = número total de imágenes de entrenamiento / tamaño del batch.
- validation_steps = número total de imágenes de validación / tamaño del batch.

Teniendo definidos estos datos fue posible pasar al entrenamiento del modelo donde se llamó al método fit del modelo con los siguientes parámetros:

- train_generator: generador de datos de entrenamiento.
- validation_data: generador de datos de validación.
- steps_per_epoch: número de pasos por época.
- validation_steps: número de pasos de validación por época.
- epochs: número de épocas para entrenar (25).
- verbose: nivel de verbosidad (1 para mostrar barra de progreso).

Finalmente, se guardo el modelo de entrenamiento con el método *save del modelo* para guardar el modelo entrenado en un archivo especifícado aparte de ajustar la arquitectura y visualización de resultados.

3.1.1.4 Ajuste de arquitectura y visualización de resultados

Para la definición de parámetros y la visualización se configuraron los parámetros de las gráficas, estableciendo el tamaño de fuente de las gráficas, por otro lado, extrayendo los datos de exactitud y error (pérdida) del historial del entrenamiento (history), creando un rango de épocas basado en la longitud del historial de pérdida. Después se crean las figuras para las graficas del error y la exactitud a base del entrenamiento y la validación, aquí se

podra observar el comportamiento de estas a traves del entrenamiento y si se aserco a cero el error y a 1 la exactitud respectivamente.

3.2 Predicciones de la red neuronal

La parte principal de la predicción comenzó con la obtención de datos de prueba, para obtener este lote se utilizó el generador (test_generator). Después, las imágenes fueron extraídas, también etiquetas separando las imágenes así como las etiquetas del lote obtenido, con ello se pudo obtener la predicción del modelo.

El método predict del modelo se aplicó a las imágenes de prueba para obtener las predicciones. Cada imagen del lote se pasa a través del modelo, que realizó una extracción de características de cada imagen como sus respectivos filtros dados por las capas aplicadas, esto resulto finalmente en un valor máximo de cada interación además de cálculos matriciales para generar un vector de salida con probabilidades para cada clase estas son distinguidas por arrojar un valor cercano o igual a 1 gracias a la función de activación aplicada.

Como ya se comento el procesamiento de las predicciones son matrices de probabilidad para cada clase, arrojando un batch de 25 pruebas, que a continuación solo se muestran las primeras 5 de un ejemplo tomado del modelo perteneciente.

resultados = model.predict (images, verbose = 0)

$$[[2.19352870e-21\ 1.81777537e-07\ 1.31305472e-14\ 9.62823774e-14\ 4.87332548e-18\ 9.99999762e-01]$$

$$[4.71066492e-08\ 1.28851332e-10\ 9.99999881e-01\ 3.00193048e-09\ 1.12844217e-15\ 1.12015847e-07]$$

$$[1.38381654e-28\ 1.95274751e-11\ 5.95278993e-21\ 2.42249835e-16\ 6.60465470e-24\ 1.00000000e+00]$$

$$[1.04248131e-10\ 1.00000000e+00\ 1.68970619e-16\ 5.55639866e-12\ 1.90877553e-14\ 5.28746658e-10]$$

$$[1.00000000e+00\ 5.28898925e-10\ 2.18374294e-11\ 1.56536611e-12\ 6.34740108e-11\ 1.08746609e-16]\ldots$$

La función argmax se aplica a cada predicción para determinar la clase con la probabilidad más alta dada por : *posicion=np.argmax(resultados, axis=1)*. La posición de la clase con la probabilidad más alta se convierte en la predicción para esa imagen, como se podrá observar con anterioiridad los valores reslatados con color verde son los valores más cercanos o iguales a 1, que con respecto a su posición seran los rostros predichos. Tal como se muestra a continuación en la siguiente matriz de posiciones maximas.

Posiciones-max : [5 2 5 1 0 0 0 5 1 1 0 3 2 4 0 3 0 4 1 5 2 5 4 2 0]

Estas posiciones equivalen a {0: 'Aaron', 1: 'Dolores', 2: 'Jose', 3: 'Juan', 4: 'Kass', 5: 'Maria'}, claro esta que en este ejemplo solo muestra las primeras 5 predicciones resaltadas con color verde tal como Maria, Jose, Maria, Dolores, Aaron, respectivamente.

Des esta manera se realizan todas las predicciones obtenidas por nuestro programa creando una figura para visualizar las imágenes que Iteran sobre las imágenes del lote y muestra cada imagen con su etiqueta predicha utilizando un diccionario mapeando las posiciones de las clase predichas a etiquetas legibles y asi finalmente guardar las predicciones con una ruta definida.

3.3 Desempeño del modelo

Para la evaluación del modelo en el conjunto de prueba obteiendo un lote de datos de prueba con el generador de datos de prueba (test_generator). Desues se evalúa el modelo en el conjunto de prueba utilizando el método evaluate del modelo: *loss, accuracy = model.evaluate(test_generator, verbose=1). Este* Imprime los resultados de pérdida y exactitud: *print (f'Loss: {loss}, Accuracy: {accuracy}').*

Finalmente se visualiza las métricas de evaluación imprimiendo la pérdida y la exactitud en el conjunto de prueba generando gráficas para la precisión y el error durante el entrenamiento y la validación guardando los resultados de la evaluación en una ruta definida.

3.4 Interfaces graficas

En esta parte de la metodología, se considera la culminación del proyecto ya que se desarrollaron dos interfaces gráficas, una para capturar datos de nuestra base de datos y otra que hará el reconocimiento facial de las personas u rostros agregados a este entrenamiento. Para la interfaz de captura de datos, se diseña una ventana usando Tkinter con campos para ingresar datos, como el nombre, dirección de carpetas, paro, reinicio y finalización de esta, de esta manera las imágenes (rostros) que captara con OpenCV son guardados con su respectiva etiqueta y ruta en la base de datos.

Y para la interfaz de reconocimiento facial, se creó otra ventana con un botón para iniciar el reconocimiento facial usando el modelo pre entrenando con su respectiva detección de rostros que con la predicción usada con anterioridad en la parte de la predicción del modelo implementando el reconocimiento facial cargando las imágenes guardadas y sus codificaciones faciales, utilizando OpenCV y face-recognition, también MTCNN para detectar y reconocer rostros en tiempo real, una imagen pre cargada o video pregrabado. La recopilación de datos de rostros fue un aspecto crucial del proyecto, como se detalló anteriormente estos datos se dividieron en tres conjuntos: entrenamiento, validación y prueba. Inicialmente, se identificarón dos rostros correspondientes a Kassandra y Dolores.

Posteriormente, se amplió la base de datos agregando imágenes de otras cuatro personas: Aaron, Maria, Juan y José. Para optimizar el proceso de captura y minimizar el uso de memoria, las imágenes se extrajeron directamente de videos pregrabados, cada una con un tamaño de 150 x 150 píxeles y centradas en el rostro identificado. Esto fue posible gracias a OpenCV, que facilita la implementación de algoritmos para el reconocimiento de patrones en imágenes.

Aunque la captura en tiempo real también es una opción viable, se optó por los videos pregrabados por su eficiencia y la poca capacidad computaciónal que necesita, ya que con la que contamos es un área de oportunidad en el proyecto. Esta interfaz se desarrolló con el fin de simplificar la incorporación de más personas a la base de datos en el futuro y poder ser utilizadas para el entrenamiento y reconocimiento facial.

A continuación, se presenta una descripción detallada de cómo se realizó la captura de datos a partir de la interfaz gráfica y cómo se organizaron para su posterior uso en el entrenamiento de la red neuronal, en la Figura 3.4 se muestra el mensaje de inicio para la captura de datos de la persona designada en donde nos da pasos a seguir y sugerencias de la ideal captura comenzando con el respectivo nombre, la carga del video pregrabado o el inicio de la cámara en vivo, después de ello la forma en que sería optima la captura desde una buena iluminación, una buna posición y jugando un poco con las constantes agregando accesorios tanto como lentes, gorras, cubrebocas y maquillaje cosas que ayudarían al entrenamiento y posterior su reconocimiento.

Estos serían almacenados con el nombre del personaje a captar en las 3 carpetas designadas tales como Testing, Training y Validation pertenecientes a la base de datos.

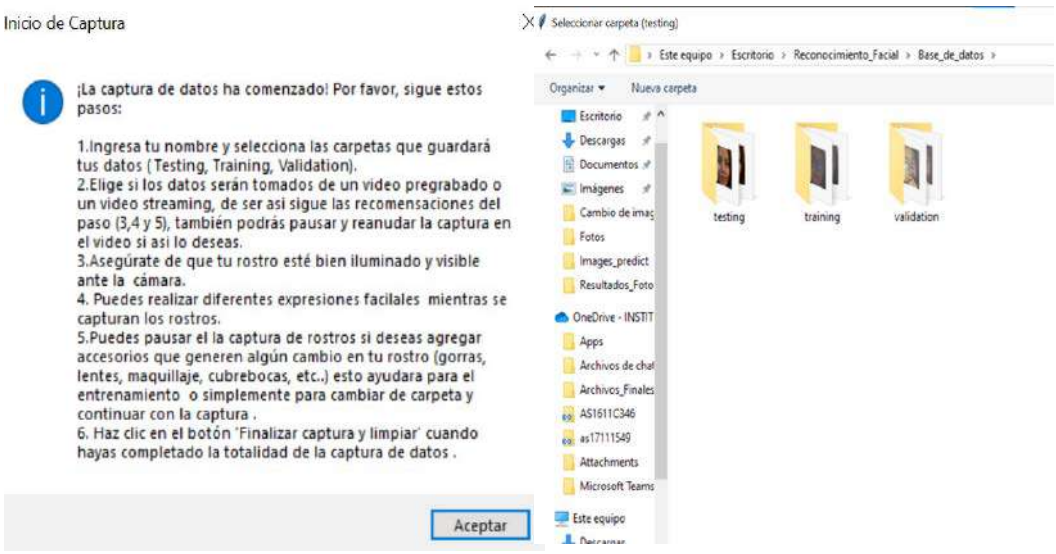

Figura 3.4 Mensaje de entrada de la interfaz gráfica para la base de datos y la selección de divisiones testing, trainng, validation para el entrenamiento del modelo creado.

En la Figura 3.5 y Figura 3.6 se muestra de manera más visual la interfaz gráfica para la captura de datos tomando como ejemplo a Maria:

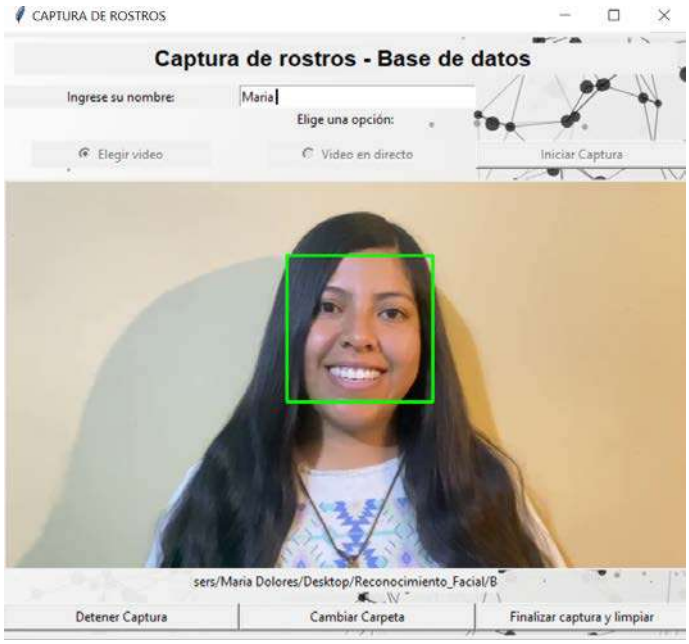

Figura 3.5 Ejemplo de Interfaz gráfica para la base de datos tomando como muestra a Maria y su captura por medio de un video pregrabado.

Figura 3.6 Inicio de la base de datos, que será clasificado y almacenado en una carpeta llamada (Data) con diferentes imágenes extraídas en diferentes poses del rostro de la persona.

De igual forma, se capturó y almacenarón los rostros restantes, totalizando seis personas representadas en 6 clases distintas creando variantes en su rostro para que el programa capture aproximadamente 5000 fotos de cada rostro en una dimensión de 150x150 pixeles dándole una etiqueta a cada una por ejemplo (rostro0_1, rostro0_2, rostro0_3).

Así consecutivamente hasta que tenga almacenada las capturas en las respectivas carpetas de training, testing y validation pertenecientes a la base de datos. Este proceso de entrenamiento ha sido crucial, ya que ha generado un punto de partida y ha catalizado mejoras en nuestros resultados.

Para la interfaz del reconocimiento facial se uso el modelo guardado con su respectivo entrenamiento guardado en un archivo H5, aplicando las funciones pertinentes para poder realizar el reconocimiento por medio de una foto cargada del sistema, un video cargado del sistema o un video en directo. En la Figura 3.7 se mostrára el diseño de la interfaz utilizada y más adelnate en el siguiente capitulo se mostrarán los resultados obtenidos directamente de la interfaz.

Figura 3.7 Interfaz gráfica para el reconocimiento facial por medio de sus 3 funciones (Cargar imagen, Video Pregrabado y Video Streaming)

CAPÍTULO 4. RESULTADOS Y DISCUSIÓN

4.1 Resultados del reconocimiento facial

Los resultados que se expondrán a continuación reflejan el progreso del proyecto desde su inicio con una clasificación binaria hasta la capacidad de reconocer múltiples rostros. Se enfoca en los pilares fundamentales de la metodología utilizada: la recopilación y organización de una base de datos de imágenes, el riguroso entrenamiento del modelo, que implicó ajustes y características específicas para optimizar su desempeño, y finalmente, la implementación del reconocimiento facial en el proceso. Además, se destaca la creación de interfaces gráficas diseñadas para agilizar y mejorar cada etapa del proyecto, con la mira puesta en futuras actualizaciones y seguimiento de este. Sin más dilación, presentamos los resultados obtenidos.

Partiendo de una eficiente obtención de datos para el entrenamiento del modelo se puede pasar a los puntos clave que hicieron cambios significativos en el desempeño de nuestro modelo como a continuación se mostrará.

4.1.1 Optimización y Mejoras en el Reconocimiento Facial: Estrategias y Consideraciones

4.1.1.1 Normalización de la información

El sobreajuste en el dimensionamiento de las imágenes fue la implementación de la normalización de la información siendo crucial para evitar el sobreajuste del modelo. Al agregar más características y dimensionar adecuadamente las imágenes, se logró una representación más equilibrada de los datos, lo que mejoró la capacidad del modelo para generalizar y adaptarse a nuevas instancias.

La normalización de imágenes es un proceso fundamental en el preprocesamiento de datos para el entrenamiento de modelos de aprendizaje automático, especialmente en aplicaciones de visión por computadora. En este proyecto específicamente en la parte de la normalización se utilizó TensorFlow para realizar la normalización de los datos de entrada. Destacando las caracteristicas antes mencionadas en la metodologia (3.2 Red

Neuronal) siendo ejemplo la escala, zom aleartorio, roitación, desplazamientos, cortes, brilllos, etc.

En la Figura 4.1, se mostrarán imágenes comparativas de la normalización y las características agregadas antes y después de las modificaciones de mejora, con el fin de continuar mejorando los resultados de manera consistente, en esta parte también nos muestra la cantidad de datos que utilizara en la prueba con 12000 imágenes , validación 60000 imágenes y entrenamiento 12000 imágenes , más la forma de clasificación que en este caso es Categorical dando por encontrado 6 clases distintas pertenecientes a las personas de nuestra base de datos.

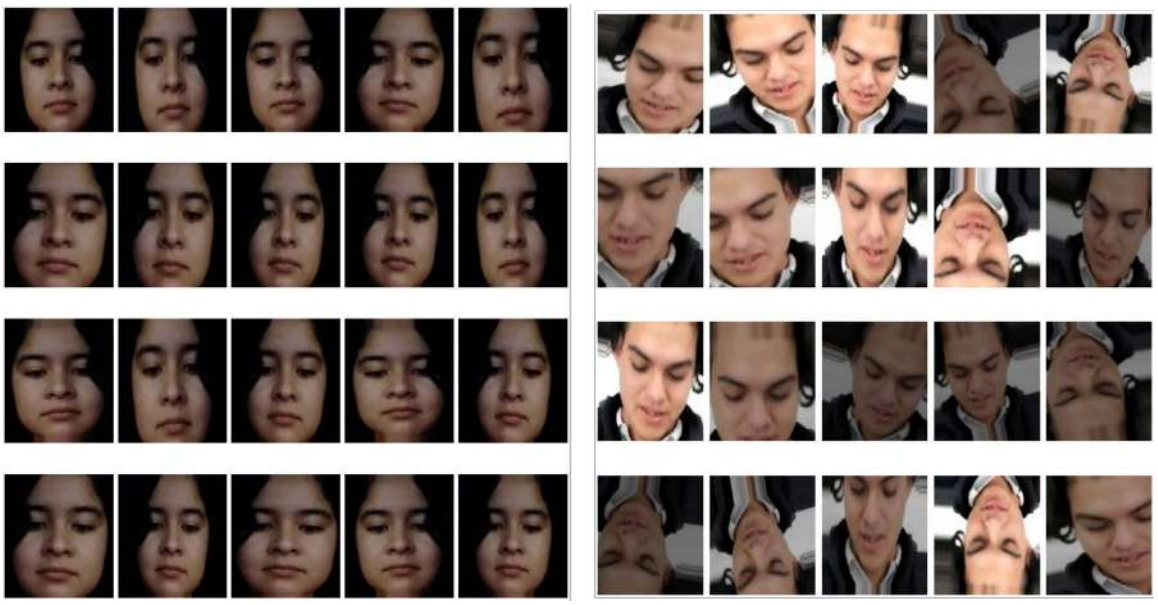

Figura 4.1 Ejemplo de comparación de la normalización de la información y lo cambios aparentes en el proceso de entrenamiento y mejoras del modelo en conjunto pertenecientes a Kass y Aaron.

En resumen, la normalización de imágenes es un paso crítico en el reprocesamiento de datos para el entrenamiento de modelos de aprendizaje automático, y los ajustes aplicados en este proceso contribuyen significativamente a mejorar la robustez y el rendimiento del modelo tal y como se puede observar en la imagen anterior donde al agregar más

cacateristicas de manipulación en la imagen tal como el giro y alargamiento ayudaron a captar de mejor manera los datos de cada imagen contribuyendo a una futura predicción con el mínimo error.

Estas mejoras en la normalización fueron un punto importante ya que se notaron cambios considerables en la predicción del modelo en un nivel de código y dentro de las interfaces graficas en especial para la de reconocimiento facial.

4.1.1.2 Modelo final para el reconocimiento

En los cambios del modelo, se realizaron ajustes significativos para mejorar su capacidad de clasificación. Aquí hay una explicación de los puntos clave:

- Transición de un modelo binario a uno categórico:

Inicialmente, el modelo estaba diseñado para realizar una clasificación binaria (binary_crossentropy), pero se modificó para admitir la clasificación de múltiples clases utilizando la función de pérdida categorical_crossentropy. Este cambio permitió al modelo distinguir entre más de dos clases de rostros, lo que amplió su capacidad de reconocimiento.

- Descripción grafica de los modelos usados en el proyecto y sus respectivos cambios.

En las siguientes imágenes se muestran los dos modelos utilizados el primero como base para una clasificación binaria y ahora con su respectiva modificación para un caso multiclasificador y el como fueron impactando estas caracteristicas en los resultados finales.

- Modelo 1

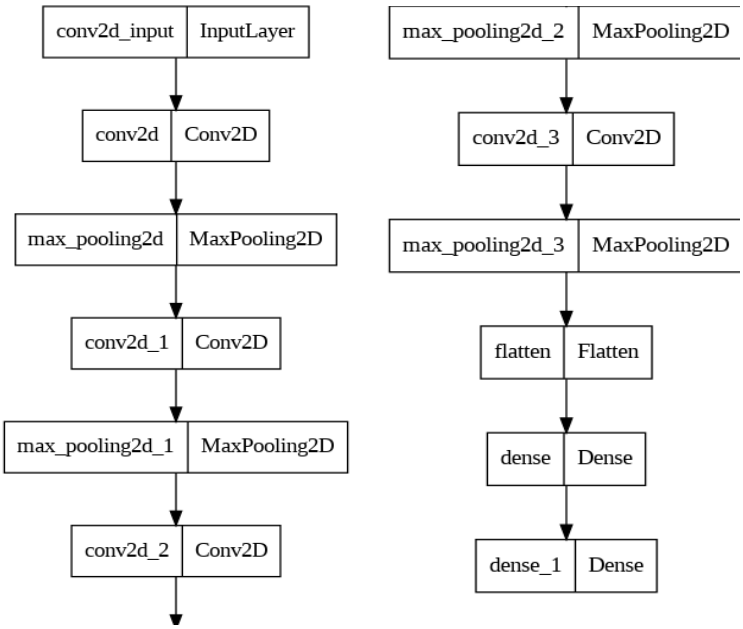

Figura 4.2 Descripción grafica del modelo utilizado inicialmente en el proyecto.

- Modelo 2

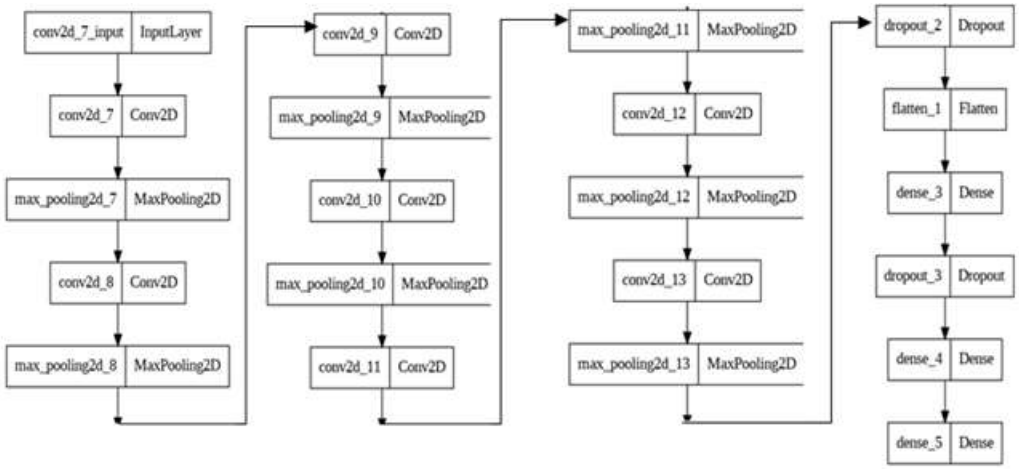

Figura 4.3 Descripción grafica final del modelo utilizado en este proyecto tras las modificaciones aplicadas a lo largo del proceso.

4.1.1.3 Modificación de capas en el modelo

Se realizaron ajustes en las capas convolucionales y de agrupación máxima (MaxPool2D) del modelo. Estas modificaciones implicaron cambios en la arquitectura del modelo, lo que podría incluir la adición, eliminación o ajuste de parámetros en estas capas para mejorar la extracción de características y la capacidad de discriminación entre diferentes clases de rostros. Hay varios cambios significativos realizados en el modelo tales como:

- Capas de Convolución y Agrupación Máxima: En el modelo 1, se utilizaron tres capas de convolución seguidas de agrupaciones máximas. Sin embargo, en el modelo 2, se han agregado más capas de convolución y agrupación máxima para capturar características más complejas y abstracciones en los datos de entrada.
- Funciones de Activación y Padding: En el modelo 1, se utilizó la activación 'relu' en todas las capas de convolución y se utilizó el padding 'valid'. En el modelo 2, se mantuvo la activación 'relu', pero se cambió el padding a 'same' en algunas capas de convolución para mantener el tamaño de la salida.
- Capas de Convolución y Agrupación Máxima: En el modelo 1, se utilizaron tres capas de convolución seguidas de agrupaciones máximas. Sin embargo, en el modelo 2, se han agregado más capas de convolución y agrupación máxima para capturar características más complejas y abstracciones en los datos de entrada.
- Funciones de Activación y Padding: En el modelo 1, se utilizó la activación 'relu' en todas las capas de convolución y se utilizó el padding 'valid'. En el modelo 2, se mantuvo la activación 'relu', pero se cambió el padding a 'same' en algunas capas de convolución para mantener el tamaño de la salida.
- Regularización Dropout: En el modelo 1, se agregó una capa de regularización Dropout con una tasa de 0.5 después de la última capa de convolución. Esto ayuda a prevenir el sobreajuste del modelo al desactivar aleatoriamente una fracción de las unidades durante el entrenamiento.

- Aumento del Número de Unidades en las Capas Densas: En el modelo 2, se aumentó el número de unidades en las capas densas a 512 y 256, respectivamente, en comparación con el modelo 1 que tenía una sola unidad en la capa densa final.
- Función de Activación de la Capa de Salida: En el modelo 1, la capa de salida utilizaba una función de activación 'sigmoid' para la clasificación binaria. Sin embargo, en el modelo 2, se utiliza una función de activación 'softmax' para la clasificación de varias clases.
- Optimización de funciones para evitar el sobreajuste: Se implementaron técnicas para evitar el sobreajuste del modelo, lo que garantiza que el modelo pueda generalizar correctamente a datos no vistos. Estas técnicas podrían incluir regularización, abandono (dropout), o técnicas específicas de optimización de la función de pérdida.
- Aumento del número de épocas de entrenamiento: El modelo se entrenó durante un número mayor de épocas, lo que permitió una mayor convergencia y ajuste a los datos de entrenamiento. Este enfoque de entrenamiento prolongado ayudó al modelo a capturar patrones más complejos en los datos y mejorar su rendimiento general.
- Ajuste de parámetros de BATCH: Se realizaron ajustes en los parámetros de BATCH para optimizar el proceso de entrenamiento del modelo. Estos ajustes incluyeron cambios en el tamaño del lote de datos, la tasa de aprendizaje, o la técnica de optimización utilizada, con el objetivo de mejorar la eficiencia y la estabilidad del entrenamiento.

4.1.1.4 Adaptación y cambio de la detección de rostros MTCNN

En este apartado es importante resaltar la utilización de los detectores de rostros aplicados en el reconocimiento facial, como lo vimos anteriormente en la captura de datos el principal detector es Haar Cascade de OpenCV por su excelente manipualción de imágenes y

captura sin embargo al momento de hacer las pruebas en el reconocimeinto se mostro una baja de rendimiento en la detección de rostros, a pesar de que las prediciones fueron buenas en su mayoria, se opto por ultilizar el detector MTCNN demostrando ser el más adecuado para aplicaciones que requieran una detección precisa y robusta de rostros en diversas condiciones.

Se concluye asi que mientras que el clasificador Haar Cascade es ideal para aplicaciones en tiempo real con recursos computacionales limitados y en entornos controlados fue nuestra opción más viable de aplicación por las eficiencias computacionales, sin embargo se espera en un seguimiento del proyecto futurocon mayor capacidad computacional la comprobación del funcionamiento del MTCNN en un cien porciento de efectividad.

4.2 Reconocimiento facial - Características de mejora esenciales

En cuanto a los resultados del reconocimiento facial, otros factores importantes que fueron considerados son:

1. Calidad de la imagen seleccionada para el reconocimiento: La calidad y la claridad de las imágenes de entrada desempeñaron un papel crítico en la precisión y la confiabilidad del reconocimiento facial. La selección de imágenes de alta calidad garantizó una representación precisa de los rasgos faciales y minimizó la ocurrencia de errores de clasificación.
2. Calidad de los videos seleccionados para el reconocimiento: La calidad de los videos de entrada influyó directamente en la capacidad del modelo para extraer características relevantes y realizar predicciones precisas. Los videos de alta resolución y definición proporcionaron una representación más fiel de los rostros y facilitaron la identificación y clasificación de las personas.
3. Calidad de la cámara para reconocimiento en streaming: La calidad y las características técnicas de la cámara utilizada para la transmisión en vivo impactaron significativamente en la precisión y la velocidad del reconocimiento facial en tiempo real. El uso de cámaras de alta calidad garantizó una captura precisa y detallada de

los rostros, lo que mejoró la capacidad del modelo para realizar predicciones precisas y en tiempo real.

4. Pruebas clave para el reconocimiento: La realización de pruebas exhaustivas y variadas fue fundamental para evaluar la robustez y la capacidad de generalización del modelo. La exploración de diferentes ángulos, condiciones de iluminación, colores de piel, accesorios y expresiones faciales permitió identificar y abordar posibles limitaciones y sesgos del modelo, garantizando su rendimiento óptimo en una amplia variedad de escenarios del mundo real.

4.3 Reconocimiento facial en diferentes casos de prueba

Por ahora se mostrará a continuación los resultados obtenidos del reconocimiento facial directamente de la interfaz gráfica creada con anterioridad. Como parte inicial del reconocimiento las pruebas realizadas directamente en nuestro código realizado llamado *DoloresVSKass* mismo que se anexara al final del documento, obtuvimos las predicciones de estas clases con la que el modelo después de ser entrenado nos arroja un batch de predicción en donde toma el valor máximo de un arreglo matricial resultado de cada imagen, después de ello toma el valor máximo dándolo como el más cercano o certero a la clase donde a continuación es capaz de predecir una etiqueta designada, esta misma metodología será utilizada más adelante en nuestra interfaz gráfica para el reconocimiento.

Resultados = model. predict (images, verbose = 0) y las etiquetas están dadas por: Etiquetas = {0: 'Aaron', 1: 'Dolores', 2: 'José', 3: 'Juan', 4: 'Kass', 5: 'Maria'} igual como se mostró en la metodología solo que ahora visualizaremos un batch completo comprobando cada resultado dado por el modelo.

4.2.1 Resultados en forma matricial

En estos siguientes arreglos se mostrarán los valores más cercanos o iguales a 1 de color verde y posteriormente la posición en la que se encuentran.

[[8.93743150e − 03 1.93731743e − 04 1.01677524e − 05 1.29826914e − 03 9.89468038e − 01 9.23373300e − 05]

[7.08694284e − 11 1.00000000e + 00 4.88217032e − 16 6.21327317e − 13 5.50864354e − 14 1.21725152e − 09]

[4.73827589e − 03 8.05163562e − 01 2.94375187e − 03 9.02053062e − 03 2.50697439e − 03 1.75626859e − 01]

[1.41319804e − 04 3.20820604e − 04 1.19949720e − 04 9.90668833e − 01 3.53600539e − 04 8.39546975e − 03]

[4.30148747e − 03 1.00080362e − 02 3.98309808e − 03 8.83172035e − 01 8.84125941e − 03 8.96941572e − 02]

[3.49960750e − 12 1.00000000e + 00 1.68313451e − 20 3.61519125e − 15 5.94499537e − 17 4.11014755e − 13]

[5.47939317e − 06 4.35509219e − 06 2.09119084e − 04 9.99162078e − 01 4.50110711e − 05 5.73909725e − 04]

[0.00000000e + 00 2.06466385e − 16 1.39578632e − 30 1.89142813e − 24 6.77693802e − 36 1.00000000e + 00]

[6.86870248e − 04 4.92854379e − 05 8.95090523e − 07 4.18278505e − 04 9.98767734e − 01 7.69205726e − 05]

[9.99808371e − 01 1.11767200e − 04 6.01340935e − 06 1.26590439e − 05 6.12318690e − 05 4.30805365e − 08]

[1.24707203e − 02 1.50120482e − 02 1.07914070e − 03 9.55136240e − 01 9.36289318e − 03 6.93910476e − 03]

[1.01444493e − 04 4.44663243e − 07 5.02684916e − 09 1.52383182e − 05 9.99882102e − 01 7.66798337e − 07]

[3.65426384e − 10 4.74096334e − 04 1.43540220e − 07 4.97514839e − 06 8.10167045e − 09 9.99520779e − 01]

[7.20248139e − 03 2.13723164e − 03 3.97189474e − 03 9.77480292e − 01 3.45285982e − 03 5.75529877e − 03]

[1.00000000e + 00 3.98033100e − 11 1.45153431e − 12 1.35316386e − 13 1.16170988e − 11 2.17352521e − 18]

[6.83847475e − 07 3.93819999e − 09 7.04493668e − 12 3.31493112e − 07 9.99998927e − 01 4.50813147e − 08]

[4.87023935e − 05 8.28556804e − 05 5.47489617e − 05 9.95773256e − 01 5.42701346e − 05 3.98626970e − 03]

[2.05827773e − 13 1.61543265e − 15 1.00000000e + 00 2.86176857e − 14 6.06574181e − 25 2.57756400e − 10]

[2.02863102e − 05 9.26054418e − 06 1.62884248e − 08 2.29093603e − 05 9.99928117e − 01 1.94806526e − 05]

[6.65354989e − 08 9.99999523e − 01 8.16052954e − 12 9.15186327e − 09 4.25060653e − 10 3.35065806e − 07]

[5.47244099e − 06 4.34988988e − 06 2.08927682e − 04 9.99162793e − 01 4.49938925e − 05 5.73474390e − 04]

[3.33126890e − 03 1.04813734e − 02 2.65557738e − 03 9.06881452e − 01 2.73851468e − 03 7.39118084e − 02]

[1.00000000e + 00 8.21141466e − 09 1.82934035e − 09 1.11418846e − 10 1.63221596e − 08 5.56497360e − 14]

[2.00720615e − 06 1.36951257e − 06 4.45763862e − 06 9.99768317e − 01 3.70693806e − 06 2.20241520e − 04]

[9.99812543e − 01 6.92960748e − 05 5.47507298e − 05 4.65169323e − 05 1.67624239e − 05 1.14400741e − 07]]

Los números marcados con color verde resultan ser el valor máximo que tiende a 1 de cada arreglo matricial en donde nos arroja un batch de 25 pruebas, y así con la función

(**Posiciones máximas** = [**4 1 1 3 3 1 3 5 4 0 3 4 5 3 0 4 3 2 4 1 3 3 0 3 0**]) es más fácil apreciar en que posición se encuentra y con respecto a esto a quien pertenece la predicción guiándonos por las etiquetas de las personas en nuestra base de datos que va de 0 a 5 respectivamente. A continuación, se muestran los resultados de la predicción de este entrenamiento en específico y sus respectivas etiquetas.

Posiciones maximas = [Kass, Dolores, Dolores, Juan, Juan, Dolores, Juan, Maria, Kass, Aaron, Juan, Kass, Maria, Juan ,Aaron, Kass, Maria, Juan...]

En la Figura 4.4 se puede observar en su totalidad de los aciertos de cada persona y su respectiva etiqueta, también destacando las diferentes características de estos sujetos de prueba en donde existen diferentes tonalidades, expresiones y accesorios con los datos anteriores se estipula una predicción y un reconocimiento facial. Para la optimización se creó una interfaz gráfica con la que podríamos acceder o reconocer un rostro mediante 3 métodos: con una imagen cargada del sistema, un video pregrabado con anterioridad y el reconocimiento en vivo. Para agilizar este procedimiento se descargó el modelo que creamos y entrenamos antes en un formato h5 cargándolo al código de reconocimiento, detectando los rostros y después predecir una clase y asignar la variable mayor conveniente, como recordatorio la predicción está dada por el valor máximo de un arreglo matricial, de ello se extrae la posición y finalmente se le otorga un valor perteneciente a una clase.

En el reconocimiento final se probaron dos detectores de rostro Clasificador Haar Cascade de OpenCV y MTCCN el algoritmo de detección de rostros por medio de redes convolucionales tal como se explicó anteriormente, en este apartado de mostrarán los resultados del reconocimiento comprobando la robustes de la detección siendo parte esencial para que nuestro modelo clasifique nuestros datos y finalmente obtener una etiqueta con el respectivo nombre. Sin más vamos a ver los diferentes reconocimientos de rostros probando los 3 métodos:

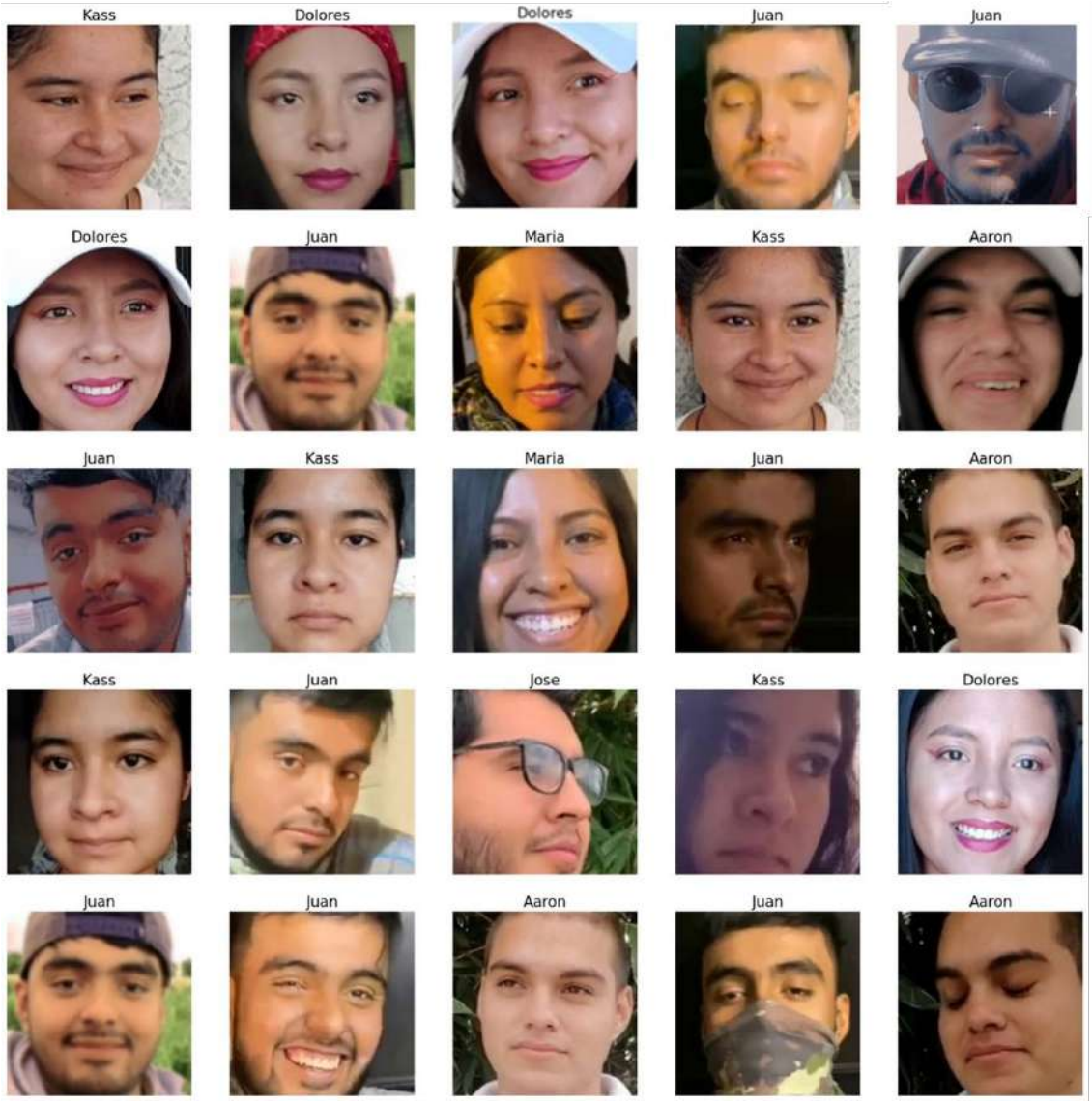

Figura 4.4 Resultados de la predicción de rostros directamente del código, resaltando sus diferentes características en cada modelo y la efectividad del reconocimiento.

4.2.2 Imagen cargada directamente del ordenador

En las siguientes imágenes se muestra el reconocimiento de rostros pertenecientes a la base de datos. Para el reconocimiento, se utilizaron fotos tomadas en diversos entornos, con variaciones en iluminación, accesorios, poses y con múltiples personas en una sola imagen. Nuestro modelo es capaz de detectar los diferentes rostros, demostrando así su

eficacia. Recordando las etiquetas dadas: {0: 'Aaron', 1: 'Dolores', 2: 'José', 3: 'Juan', 4: 'Kass', 5: 'Maria'}

En la Figura 4.5 se muestra a Juan en dos diferentes posiciones, con una relativa expresión, pero con diferente iluminación lo cual nos da un acierto en su reconocimiento.

Figura 4.5 Reconocimiento de Juan en diferentes posiciones, accesorio y expresiones.

En la siguiente Figura 4.6 se muestra la predicción de Aaron a diferentes distancias, iluminación, accesorios, posiciones y expresiones, dando un acierto en su reconocimiento.

Figura 4.6 Reconocimiento de Aaron a diferentes distancias y accesorios complementarios a sus facciones.

La predicción siguiente pertenece a Dolores cabe mencionar que este rostro lleva más entrenamiento que los anteriores lo que permitió un el reconocimiento más robusto en cuanto la iluminación y accesorios tal como el cubrebocas, como muestra en la Figura 4.7.

Figura 4.7 Reconocimiento de Dolores en diferentes ambientes, expresiones, posiciones

De la misma forma la predicción de Kassandra en la Figura 4.8 ha sido más certera gracias al entrenamiento previo, permitiendo detectarla a una distancia considerable y con cubrebocas.

Figura 4.8 Reconocimiento de Kassandra en diferentes ambientes, expresiones, posiciones.

En el caso de Maria se obtienen resultados buenos considerando que la base de datos de ella perteneció a un ambiente totalmente diferente al de las fotos presentes, reconociéndola así con el cabello suelto y a diferentes distancias apreciándolo en Figura 4.9.

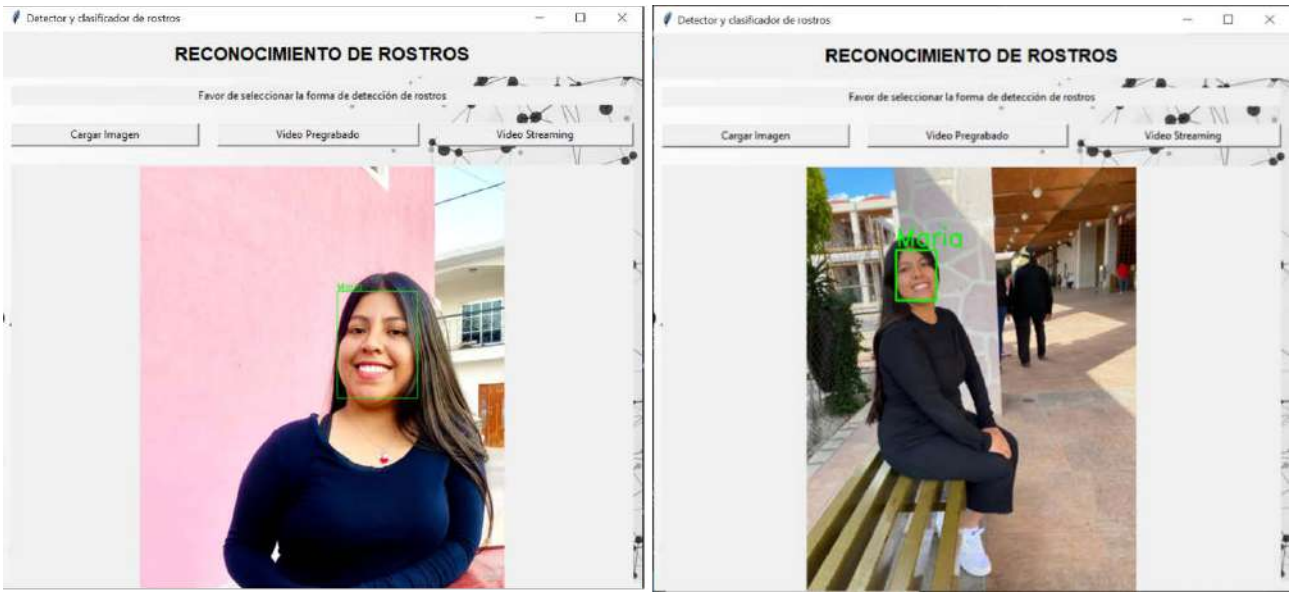

Figura 4.9 Reconocimiento de Maria en diferentes ambientes, distancias y posiciones

Hasta este punto de los resultados solo se hicieron pruebas individuales de cada persona, sin embargo, al comprobar la predicción y detección de los rostros comprobamos la eficacia en grupos de personas, no a su totalidad del 100% pero si las bases para un futuro seguimiento, sin más a continuación se mostraran reconocimiento en grupo en diferentes ambientes, expresiones y accesorios que en este caso aplicaría el maquillaje.

En la Figura 4.10 se puede apreciar dos esenarios que muestra el reconocimiento 3 personas, Dolores, Kassandra y Aaron , como se puede apreciar reconoce perfectamente a Dolores y Kass pero mantiene ese margen de error en cuanto a Aaron, ya que lo predice como Jose, y en la parte del costado derecho se encuentra Dolores y Maria predicciendo también de manera erronea, sin embargo aquí se pueden recalcar los parecidos entre ellas ya que son hermanas, su maquillaje es igual en cuanto al delineado y color de labios, sin contar su expresión que las hace ver como personas iguales.

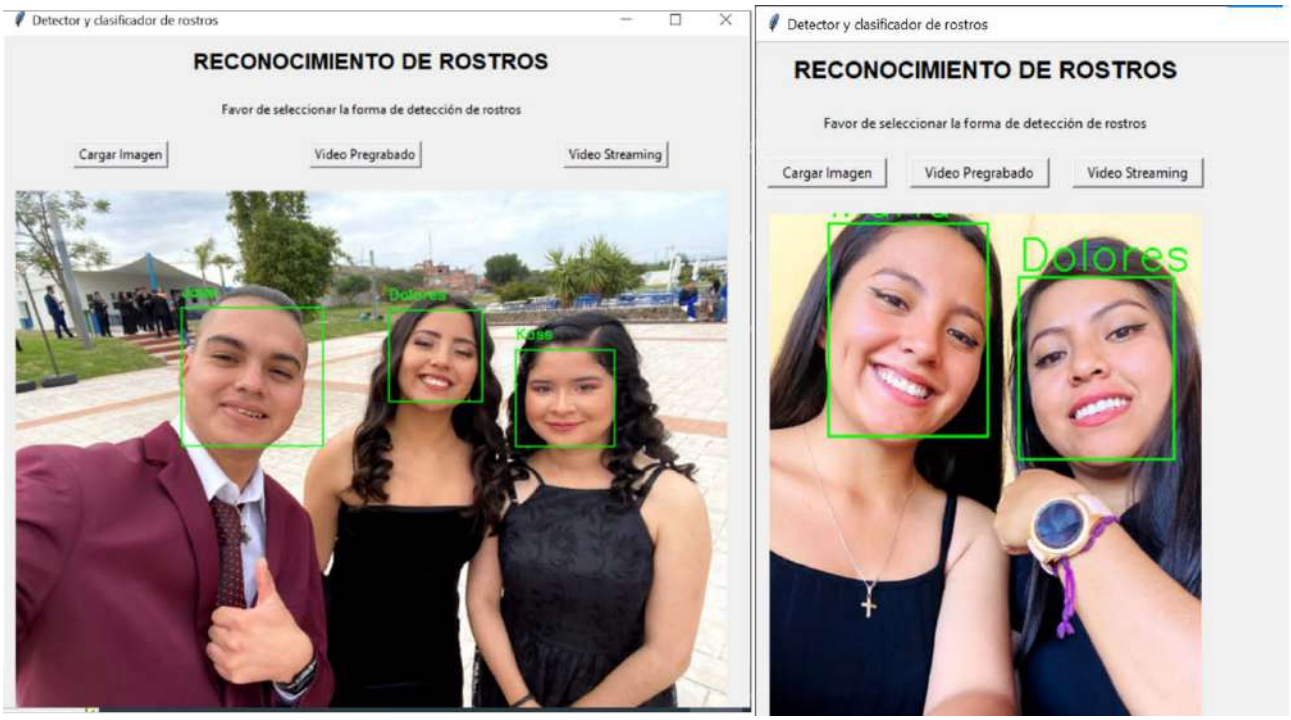

Figura 4.10 Reconocimiento en grupos con diferentes ambientes, expresiones, posiciones y accesorios externos como el maquillaje.

En las figuras anteriores se muestra de nuevo a Dolores y Kass juntas con diferentes expresiones e incluso se resalta la tonalidad de la piel, sin embargo la predicción es correcta en su totalidad recordando su constante entrenamiento previo en la clasificación binaria.

También se observa a Jose y Dolores en la Figura 4.11 en un ambito totalmente diferente a su entrenamiento, aquí se pude observar el parecido en las tonalidades de piel y las expresiones dadas por una sonrisa, a pesar de ello se nota la diferencia en cuanto al maquillaje de Dolores y rasgos particulares de Jose tales como la barba y la ceja poblada, aún con estas características se logró un resultado satisfactorio.

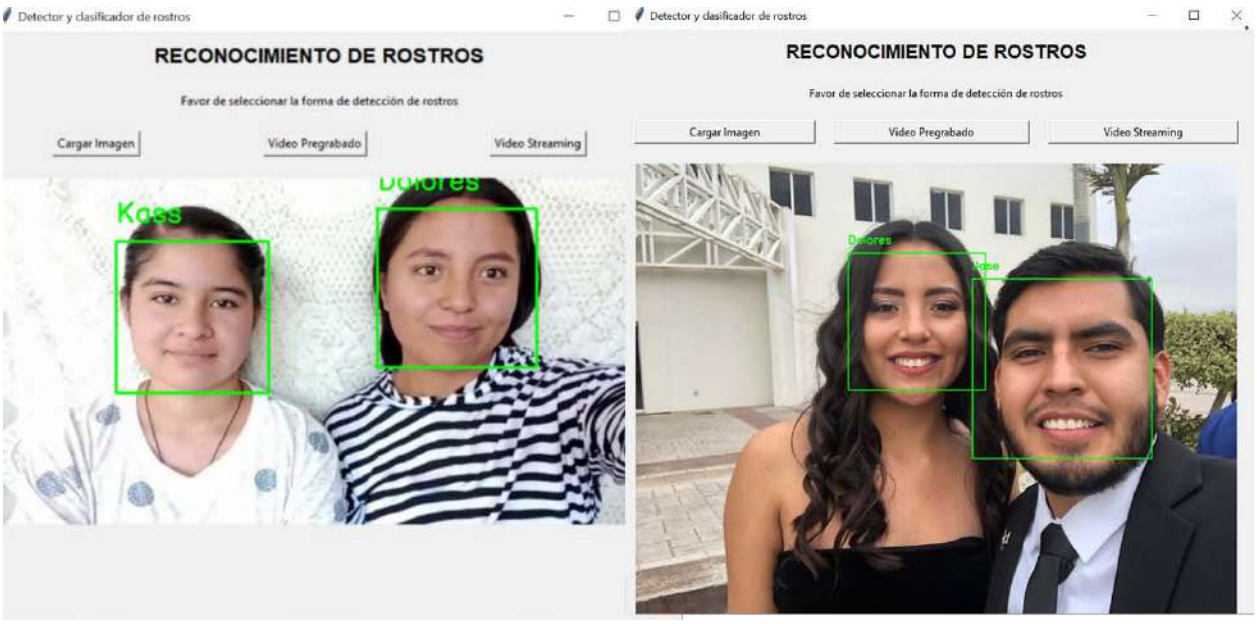

Figura 4.11 Reconocimiento en pareja con aprobación de los 3 rostros presentados

Finalmente se muestra una parte mas de este proyecto para el reconocimeinto de rostros donde al no detectar o no llegar a una predicción acercada a las ya obtenidad, marcaria este como un rostro desconocido como se muestra a continuación en Figura 4.12.

En la primer parte se muestra una persona (Alicia) de manera individual resaltando la detección de este rostro en un perfil de lado con tonalidades relativamente contrastadas y un maquillaje sutil, lo cual pudo haberlo confuncido con alguna otra de nuestros modelos, sin embargo se obtiene una prediccion correcta, en la imagen del costado derecho se observan a dos personas (Teresa y Manuel) en un enfoque oscuro dado por el ambiente externo, sn embargo se mantiene correcta la predicción a pesar de las tonalidades y rasgos parecidos a los de nuestros modelos.

Figura 4.12 Validación de rostros desconocidos al ser aplicado al reconocimiento facial, mostrándolo con un recuadro rojo y la etiqueta de “desconocido”.

4.3 Desempeño del Modelo

El modelo de reconocimiento facial fue evaluado en tres conjuntos de datos: entrenamiento, validación y prueba. Los resultados en cada conjunto se describen a continuación:

a) Conjunto de Entrenamiento

- Error (pérdida) en entrenamiento: 0.1993. Este valor representa la cantidad de error que el modelo cometió al aprender de los datos de entrenamiento. Un error de 0.1993 indica que, aunque el modelo ha aprendido bastante bien, todavía hay margen para mejorar.
- Exactitud en entrenamiento: 93.20%. Este valor indica que el modelo clasificó correctamente el 93.20% de las imágenes en el conjunto de entrenamiento. Una exactitud alta en el entrenamiento muestra que el modelo ha aprendido bien a reconocer los rostros en los datos de entrenamiento.

b) Conjunto de Validación

- Error (pérdida) en validación: 0.1375. Este valor refleja el error del modelo en el conjunto de validación. Un error de 0.1375 es más bajo que el error en el conjunto de entrenamiento, lo cual es positivo y sugiere que el modelo generaliza bien.
- Exactitud en validación: 95.54%. Esta métrica indica que el modelo clasificó correctamente el 95.54% de las imágenes en el conjunto de validación. Una alta exactitud en la validación es un buen indicador de que el modelo tiene un buen rendimiento y no está sobreajustado.

c) Conjunto de Prueba

- Error (pérdida) en prueba: 0.1132. Este es el error del modelo al predecir en datos completamente nuevos que no fueron usados ni en el entrenamiento ni en la validación. Un error de 0.1132 muestra que el modelo sigue siendo preciso encondiciones nuevas.
- Exactitud en prueba: 96.41%. Este valor representa la proporción de imágenes correctamente clasificadas por el modelo en el conjunto de prueba. Con una exactitud de 96.41%, el modelo demuestra una alta capacidad para generalizar a nuevos datos.

Como conclusión de los resultados tenemos que la disminución del error desde el entrenamiento (0.1993) hasta la prueba (0.1132) indica que el modelo mejora su desempeño cuando se le presentan datos nuevos, lo que sugiere una buena generalización. Y el Incremento de la Exactitud desde el entrenamiento (93.20%) hasta la prueba (96.41%) es un indicador de que el modelo no solo aprende bien los patrones en los datos de entrenamiento, sino que también es capaz de aplicarlos correctamente a datos nuevos.

Después de aplicar las mejoras del proyecto tanto en la normalización de la información, el modelo, el modelo de detección MTCCN y finalmente las características de las imágenes, aumentarón considerablemente nuestros resultados reflejándolo en el desempeño del modelo y la predicción.

A continuación en la Figura 4.13, se mostrarán lo diferentes cambios en cuanto a la exactitud y el error, y también los comportamientos de cada entrenamiento en cuanto a estos factores tomando una muestra de 40 corridas de entrenamiento donde se pueden ver reflejados los diferentes cambios del desempeño del modelo.

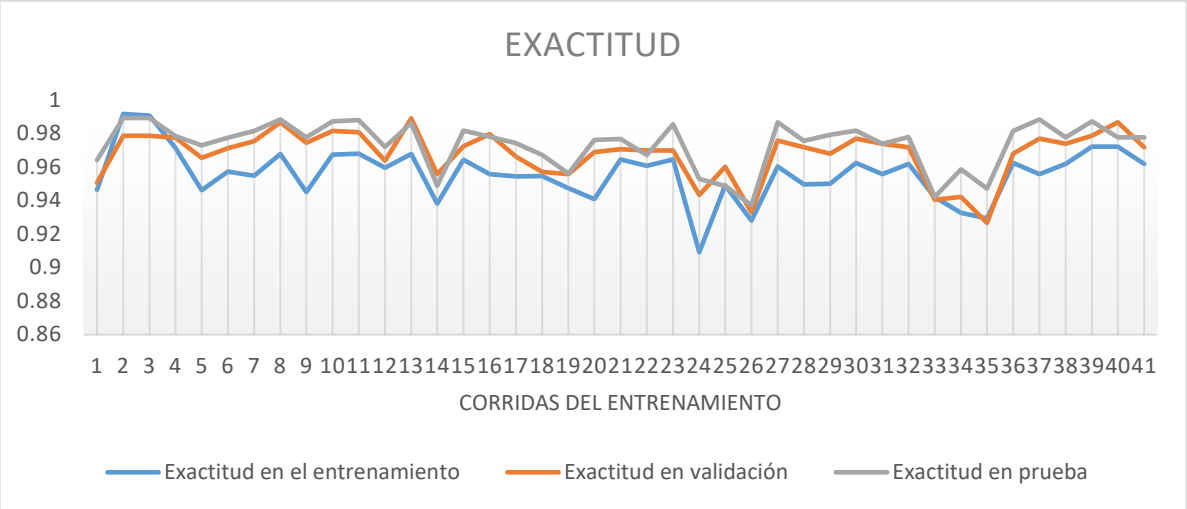

Figura 4.13 Gráficas de la exactitud del desempeño del modelo tomando una batch de 40 corridas de entrenamiento con un rango de 20 a 30 épocas en esta etapa. Los resultados se mantuvieron en un rango de 0.95 a 0.97 mismos que se pueden visualizar de mejor forma en la parte derecha de la imagen.

De esta misma manera en la Figura 4.14 se observa el error en el entrenamiento, validación y prueba, demostrando los cambios en el modelo y como fue su comportamiento repecto una de la otra, ya que el error en el entrenamiento se mantuvo ligeramente más allto con respecto a la validación y prueba, esto porque el entrenamiento fue mantenido al inicio en un rango de 15 a 20 epocas, después se subio de 25 a 30 lo que también provoco mejoras en nuestro desempeño.

Sin embargo, como se muestra en la grafica el error se mantuvo en un rango de un 0.2 a 0.05, cabe resaltar que hubo corridas donde se alejaban de estos limites sin embargo el

constante entrenamiento mantuvo a su mayoria en estos limites favoreciendo nuestras predicciones.

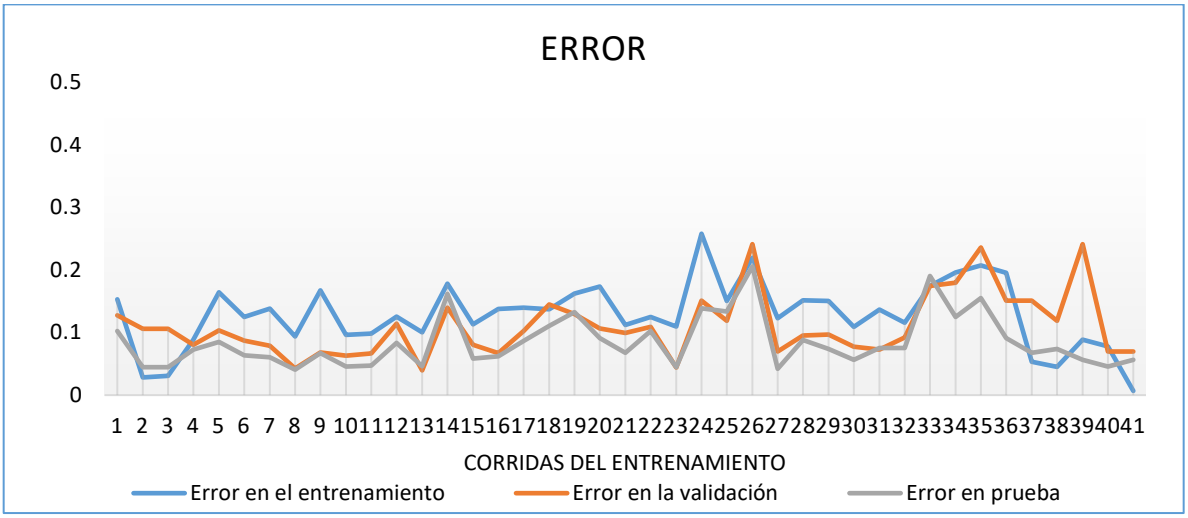

Figura 4.14 Graficas del error del desempeño del modelo tomando una batch de 40 corridas de entrenamiento con un rango de 20 a 30 épocas en esta etapa.

Hay que recordar también que un error cercano a cero y una exactitud cercana a 1 en el entrenamiento del modelo indica que las predicciones tinene poca desviación respecto a las etiquetas verdaderas y que casi todad sus predicciones son correctas. Esto sugeriendo que el modelo ha aprendido a representar los datos de manera acertiva, sin embargos e nevesita tener una evaluación y análisis de la prueba y validación para verificar que no este sobre ajustando (pertencientes al ruido en las imágenes).

Finalmente se puede mostrar las graficas correspondientes al error y la exactitud del último entrenamiento tomado como prueba, ya con las modificaciones antes mencionadas se puede observar una mejora significativa en la prediccion de rostros ya que el error esta muy cercano al cero por debajo de 0.05 al menos en esta corrida de 25 epocas, la exactitud también se muestra muy cercana a uno teniendo una diferenciación de rango en el entrenamiento pero finalmente logra estabilizarse de manera correcta teniendo a 1 como se observa en la Figura 4.15 y Figura 4.16.

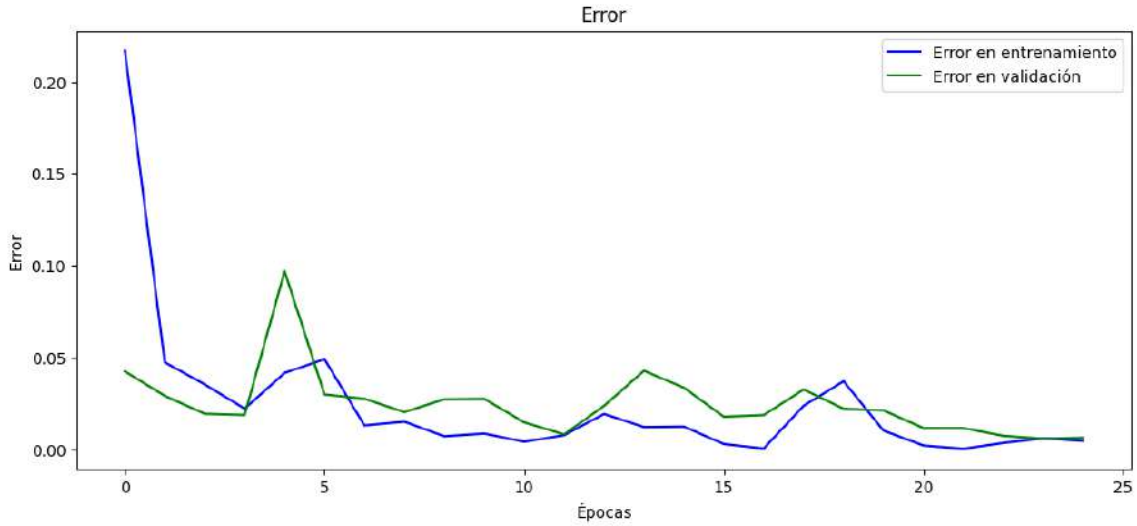

Figura 4.15 Grafica del error con respecto al último entrenamiento de nuestro modelo, mostrando la tendencia acero por debajo de 0.05.

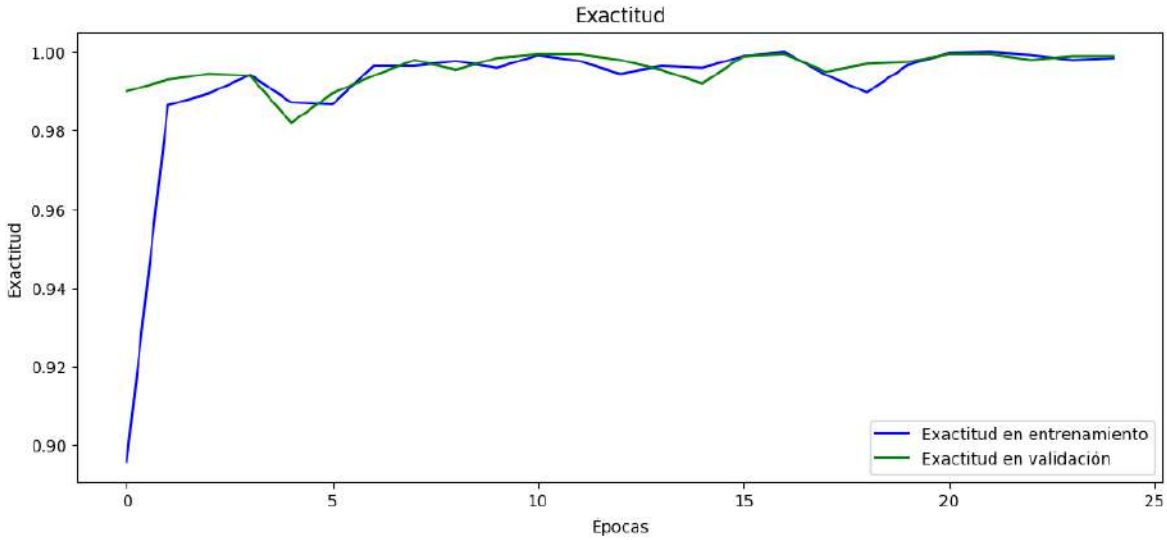

Figura 4.16 Grafica de la exactitud con respecto al último entrenamiento de nuestro modelo, mostrando la tendencia a uno por encima de 0.98.

En general, se observa que los errores en el conjunto de prueba tienden a ser más bajos en los conjuntos de entrenamiento y validación, lo que sugiere que el modelo es apropiado. La exactitud en el conjunto de prueba se mantiene constante y cercana a uno; lo que indica

que el modelo es efectivo en su capacidad para reconocer caras. La variabilidad en los errores y la exactitud entre diferentes corridas del modelo es relativamente baja, lo que sugiere una consistencia en su desempeño.

En cuanto a la comparación entre conjuntos de datos: La exactitud en el conjunto de prueba es ligeramente más alta que en los conjuntos de entrenamiento y validación, lo que sugiere que el modelo no está sobre ajustando demasiado a los datos de entrenamiento y no hay una discrepancia significativa entre las métricas de desempeño en los diferentes conjuntos de datos, lo que indica que el modelo generaliza bien.

Asi las tendencias a lo largo del tiempo o del número de iteraciones de entrenamiento, se observa una mejora gradual en la exactitud y una disminución en el error, lo que sugiere que el modelo está aprendiendo de manera efectiva y asi al visualizar las métricas de desempeño a lo largo del tiempo, vemos una tendencia general de mejora, con algunas fluctuaciones menores, lo que indica una convergencia del modelo hacia un mejor rendimiento.

Por ello cuando los errores cometidos por el modelo, no se observan patrones claros o consistentes, lo que sugiere que los errores pueden ser aleatorios y no están asociados con características específicas de las imágenes si no mas bien al nuero de epocas aplicadas o la continuidad del entrenamiento ya que mucho de este fue gracias a un servidor en linea (GOOGLE COLAB) en donde no siemppre estuvimos conectada a un servidor consistente o tambien la falta de internet generaba un retrazo significativo.

Sin embargo, el análisis sugiere que el modelo de reconocimiento facial tiene un buen desempeño, con una alta exactitud en el conjunto de prueba y una capacidad para generalizar bien a partir de los datos de entrenamiento.

CONCLUSIÓN

Como conclusión de este proyecto podemos resaltar el análisis exhaustivo de los resultados del modelo desarrollado obtenido tras múltiples iteraciones de entrenamiento y evaluación del sistema de reconocimiento facial reflejando de manera contundente la efectividad y el cumplimiento de los objetivos planteados en el proyecto en cuanto a la precisión del modelo y generalización. Se puede resaltar resaltar la disminución progresiva del error desde el conjunto de entrenamiento hasta el conjunto de prueba, junto con el incremento correspondiente en la exactitud, subrayando la capacidad del modelo para generalizar a datos nuevos con una alta precisión. Este fenómeno indica una sólida comprensión de los patrones subyacentes en los datos de entrenamiento y una habilidad excepcional para aplicar ese conocimiento a situaciones no vistas previamente.

El impacto de las mejoras realizadas a lo largo del proyecto, incluyendo la normalización de la información, ajustes en la arquitectura del modelo, la integración del detector de rostros MTCNN y la optimización de las características de las imágenes, han generado un impacto significativo en el desempeño general del sistema. Esto se refleja en la notable mejora en la precisión (cercano a 100%) y la capacidad de predicción del modelo, validando la efectividad de las estrategias implementadas, más al cumplimiento del objetivo general del proyecto al desarrollar una interfaz gráfica funcional y eficiente a la cual se realizo la implementación del modelo de red neuronal convolucional.

Además se desarrollo una metodología para la generación de una base de datosde imágenes organizada. La congugación de los distintos puntos demuestra el cumplimiento exitoso de los objetivos especificos y el objetivo general del proyeto. Con el desarrollo del proyecto se logró implementar una interfaz gráfica intuitiva y eficiente para realizar el reconocimiento facial de 6 individuos con imágenes nunca antes vista con una exactitud de aproximadamente el 96%.

Si bien el proyecto ha alcanzado resultados prometedores, existen áreas para futuras investigaciones y mejoras. Una de ellas es la exploración de técnicas de aumento de datos

con una mejor calidad de fotos ya que esto impacta directamente en una mejor normalización de los datos, ya que podrá mapear cada caracteristica encontrada por los filtros de cada capa convolucional y finalmente obtener un valor resultante en los Pooling.

Es importante mencionar que la evaluación de arquitecturas de redes neuronales más avanzadas y/o complejas que permiten incrementar la exactitud en el reconocimiento facial y disminuir sus errores es una de las áreas más relevantes a estudiar en trabajos futuros. El modelo presentado en este proyecto tiene un sinfín de áreas de oprtunidad en cuanto a sus capas y funciones de activación, en un futuro se espera su mejora y optimización.

Este proyecto se llevo a cabo principalmete por la accesibilidad de Google Colab que es un colaborador de programación en línea, ayudandonos al entrenamiento de nuestra red casi en su totalidad, pero para ello se requirió una buena conexión a internet lo que limitó el entrenamiento de la red; por lo que seria recomendable tener capacidad computacional que soporte un entrenamiento más exahutivo directamente en Python, esto para aumetar las epocas preferentemente mayor a 50.

BIBLIOGRAFIA

Arimetrics. (s.f.). Recuperado el 25 de 02 de 2023, de Qué es Interfaz Gráfica de Usario (GUI): https://keepcoding.io/blog/division-datos-deep-learning/

Delgado, R. (2 de 12 de 2015). *Revista digital INESEM*. (INESEM, Editor) Recuperado el 24 de 10 de 2022, de . https://www.inesem.es/revistadigital/gestion-integrada/conexion-arranque-motores-trifasico/#:~:text=Los%20motores%20trifásicos%20son%20motores,sistema%20trifásico%20de%20corriente%20alterna.

Diario Electronicohoy. (25 de 1 de 2019). Recuperado el 20 de 10 de 2022, de https://www.diarioelectronicohoy.com/controladores-inteligentes-para-motores-trifasicos/

Eléctrica, R. d. (2 de 2021). *EDITORES*. Recuperado el 20 de 10 de 2022, de https://www.editores.com.ar/revistas/ie/324/motores_electricos_dafa

Equipo editorial, E. D. (11 de 11 de 2020). *concepto*. Recuperado el 20 de 10 de 2022, de https://concepto.de/interfaz/

Geek Electrónica. (s.f.). Recuperado el 22 de 10 de 2022, de "¿Qué son los microcontroladores PSoC?: https://geekelectronica.com/que-es-psoc/

GOTIYE. (18 de 04 de 2022). Recuperado el 20 de 02 de 2023, de https://www.algotive.ai/es-mx/blog/machine-learning-que-es-el-aprendizaje-autom%C3%A1tico-y-c%C3%B3mo-funciona

Hewlett Packard. (07 de 2019). Recuperado el 18 de 02 de 2023, de Inteligencia artificial: https://www.hpe.com/mx/es/what-is/artificial-intelligence.html

InGenio. (s.f.). Recuperado el 25 de 02 de 2023, de InGenio Learning: https://ingenio.edu.pe/blog/en-que-se-relaciona-python-con-la-inteligencia-

artificial/#:~:text=Python%20y%20la%20inteligencia%20artificial,de%20sistemas%20de%20inteligencia%20artificial.

Iturburu, I. G. (19 de 02 de 2022). *Univerisdad Politecnica SALESIANA*. Recuperado el 25 de 02 de 2023, de https://dspace.ups.edu.ec/bitstream/123456789/22605/1/UPS-GT003738.pdf

Kaspersky. (s.f.). Recuperado el 20 de 02 de 2023, de Reconocimiento facial: definición y explicación: https://latam.kaspersky.com/resource-center/definitions/what-is-facial-recognition

KeepCoding. (12 de 08 de 2022). Recuperado el 25 de 02 de 2023, de La división de datos en Deep Learning: https://keepcoding.io/blog/division-datos-deep-learning/

Laboratorio de Electrónica. (s.f.). Recuperado el 20 de 10 de 2022, de http://labelectronica.weebly.com/proteus.html

Management Blog. (31 de 08 de 2022). Recuperado el 18 de 02 de 2023, de https://www.fsm.ac.in/blog/an-introduction-to-machine-learning-its-importance-types-and-applications/

Méndez, P., & Ibarra, J. (2014). Implementación de una red neuronal de convolución para el reconocimiento de poses en imágenes de rostros. *Avances en Ciencias e Ingenierías*, 6(2), A1-A6.Python. (s.f.). Recuperado el 25 de 02 de 2023, de https://www.python.org/

Morales Velasco, L. R. (2023). Conocimiento de rostros mediante una red neuronal para el pase de lista (Tesis de pregrado, Universidad Tecnológica de la Mixteca). Huajuapan de León, Oaxaca.

Rashid, M. H. (1995). Electrónica de potencia, Circuitos , Dispositivos y Aplicaciones . En *Electrónica de potencia* (Segunda ed.). Pearson Educación.

Reyes Campos, J. E. M., Castañeda Rodríguez, C. S., Alva Luján, L. D., & Mendoza de los Santos, A. C. (2023). Sistema de reconocimiento facial para el control de accesos mediante Inteligencia Artificial. *Innovación y Software, 4*(1).

Riuz, C. (s.f.). Recuperado el 20 de 10 de 2022, de https://www.estudiaraprender.com/2021/01/17/todo-sobre-pcb-wizard/

Sanabria, C. E. (05 de 2017). *Revista Colombiana de Tecnologias de Avanzada (RCTA)*. Recuperado el 25 de 02 de 2022, de https://www.researchgate.net/publication/319958509_RECONOCIMIENTO_FACIAL_BASADO_EN_EIGENFACES_LBHP_Y_FISHERFACES_EN_LA_BEAGLEBOARD-xM

Scielo . (7 de 1 de 2015). Recuperado el 20 de 10 de 2022, de https://scielo.conicyt.cl/pdf/infotec/v26n3/art15.pdf

tecnología + informatica. (s.f.). Recuperado el 20 de 10 de 2022, de https://www.tecnologia-informatica.com/que-es-interfaz/

Think Blg. (26 de 11 de 2019). Recuperado el 20 de 02 de 2023, de https://empresas.blogthinkbig.com/las-matematicas-del-machine-learning-redes-neuronales-parte-i/

Valdés, R. S. (06 de 11 de 2021). Obtenido de https://rene-silva-valdes.medium.com/introducci%C3%B3n-a-la-programaci%C3%B3n-en-python-2-funciones-y-librer%C3%ADas-5fd3da9046cf

Verch, M. (s.f.). *sitiobigdata*. Recuperado el 20 de 02 de 2023, de Teorema de bayes en Machine Learning: https://sitiobigdata.com/2019/12/24/teorema-de-bayes-en-machine-learning/#:~:text=El%20teorema%20de%20Bayes%20proporciona%20un%20manera%20de%20calcular%20la,P%C3%A1gina%20156%2C%20Machine%20Learning%201997.