Pontificia Universidad Católica del Ecuador

Facultad De Ingeniería

Modelo de Reconocimiento Facial con Redes Neuronales: Un enfoque de visión por Computador

AUTOR

LUIS TARQUINO MOLINA GUAJÁN

TRABAJO PREVIA A LA OBTENCIÓN DEL TÍTULO DE MAGISTER EN SISTEMAS DE INFORMACIÓN

MENCIÓN DATA SCIENCE

Dirección: MSC. EDUARDO MONTERO

Quito, agosto 2024

Dedicatoria

A mi querida esposa, por su amor, su paciencia y su constante apoyo en cada paso de este camino. Gracias por ser mi compañera y por creer en mí incluso en los momentos más difíciles.

A mis hijos, por ser mi fuente de inspiración diaria y por recordarme siempre la importancia de luchar por nuestros sueños. Este logro es tanto de ustedes como mío.

Agradecimientos

En primer lugar, a Dios por permitirme seguir avanzando y superarme en todos los ámbitos de la vida, a mi tutor por guiarme e impulsarme para que el proyecto se efectúe exitosamente, a mis profesores de la maestría, quienes con sus conocimientos han ampliado mis horizontes a lo largo de toda la maestría.

Resumen

La inteligencia artificial ha emergido como una herramienta crucial en la mejora de la seguridad informática. Este estudio se centra en la implementación de un modelo de redes neuronales convolucionales (CNN) utilizando *TensorFlow* para identificar individuos a partir de imágenes faciales, evaluando su utilización como un segundo factor de autenticación en sistemas computacionales. El objetivo general de este estudio es implementar un modelo de inteligencia artificial basado en redes neuronales convolucionales (CNN) utilizando la librería *TensorFlow* para técnicas de reconocimiento facial, con un enfoque en explorar y analizar el desempeño de estas redes dentro del campo de la visión por computador. Se adoptó un enfoque cuantitativo con un diseño experimental Cuasiexperimental. La unidad de análisis fue un conjunto de imágenes faciales, para lo cual se utilizó un clasificador de rostros pre-entrenado, *haarcascade_frontalface*. Se utilizaron métodos de regularización, ajustes en los hiperparámetros y la manipulación de variables independientes, con el fin de optimizar tanto la precisión (*precision*) como la adaptabilidad del modelo ante diferentes condiciones. El principal hallazgo revela que el modelo final, alcanzó una precisión (*precision*) del 98%, demostrando una sólida capacidad para adaptarse y responder a diferentes variaciones en los datos. Los resultados sugieren que las redes neuronales convolucionales (CNN) podrían tener un gran potencial para mejorar la seguridad en sistemas de autenticación, dada su capacidad para procesar y reconocer patrones faciales complejos. Al ajustar cuidadosamente los hiperparámetros y aplicar técnicas de regularización, el modelo desarrollado mostró una precisión (*precision*) notable y una capacidad de generalización que lo hacen prometedor para su implementación en entornos donde la seguridad es crítica. Estos hallazgos podrían abrir la puerta a futuras investigaciones y aplicaciones en sistemas de autenticación basados en redes neuronales convolucionales (CNN).

Palabras clave: reconocimiento facial, redes neuronales convolucionales, autenticación, *TensorFlow*, seguridad informática, visión por computador.

Índice

Tabla de contenido

Índice de Figuras

Índice de Tablas

Introducción

En los últimos años, hemos sido testigos de un acelerado avance tecnológico, destacándose el significativo progreso en el campo del reconocimiento facial. Las redes neuronales convolucionales (CNN) se han destacado por su capacidad para identificar y clasificar patrones complejos en imágenes, lo que las ha convertido en una herramienta clave en sistemas de seguridad y autenticación. Desde su uso en teléfonos móviles hasta su implementación en sistemas de vigilancia, el reconocimiento facial basado en CNN ha demostrado ser esencial para mejorar la precisión y eficiencia en la identificación de personas.

Sin embargo, a medida que la demanda de sistemas de autenticación más seguros y efectivos crece, también surgen desafíos importantes. Aspectos como las variaciones en la iluminación, las expresiones faciales y las diferencias entre grupos demográficos pueden afectar la precisión de estos modelos. Además, encontrar un equilibrio entre evitar falsos positivos y asegurar que todos los usuarios legítimos sean correctamente identificados sigue siendo un reto significativo. Por lo tanto, es fundamental seguir investigando y perfeccionando los modelos de CNN para garantizar su solidez y fiabilidad en una variedad de situaciones.

Este estudio tiene como objetivo desarrollar y evaluar un modelo de reconocimiento facial utilizando redes neuronales convolucionales, con un enfoque particular en la optimización de parámetros clave y técnicas de preparación de datos. La meta es no solo mejorar la precisión y eficiencia del modelo, sino también establecer una base sólida que permita su aplicación en sistemas de autenticación en tiempo real, respondiendo así a las crecientes demandas de seguridad en un mundo cada vez más conectado.

Planteamiento del problema

En los últimos años, la seguridad en los sistemas informáticos ha sido una preocupación creciente debido al incremento de ataques cibernéticos y brechas de seguridad. Según un informe de *Cybersecurity Ventures*, se espera que el costo global del cibercrimen alcance los 10.5 billones de dólares anuales para 2025 (Freeze, 2022). El uso de contraseñas endebles es una práctica de seguridad deficiente. A nivel de la región ecuatoriana enfrenta diversas vulnerabilidades que ponen en riesgo la seguridad de la información de los usuarios. Según un estudio reciente, se identificó que la mayor vulnerabilidad en la banca virtual ecuatoriana es el robo de credenciales, con un índice del 31% en los sistemas de hardware y software (Montes & Israel, 2023). Esta es una tendencia alarmante que resulta de la incapacidad de los sistemas de autenticación existentes, como las contraseñas, para asegurar el acceso no autorizado de manera adecuada.

Con el propósito de reducir estas dificultades, se ha sugerido el fortalecimiento del sistema de seguridad mediante la incorporación de un segundo factor de autenticación. Hoy en día, asegurar la identidad digital se está volviendo bastante crítico, y la tecnología biométrica con la asistencia de la inteligencia artificial es la que puede ayudar en este esfuerzo. La toma de nuestras características biométricas como credenciales de entrada será bastante simple y sin esfuerzo, además agrega seguridad en todos los dominios. García (2023), escribe que "La tecnología biométrica impulsada por IA es una forma mejor, más cómoda, ágil y segura de probar que somos quienes decimos ser sin tener que depender de contraseñas". Este estudio se propone aplicar las tecnologías de Inteligencia Artificial (IA), Aprendizaje Profundo (DL) y en particular la librería *TensorFlow* para crear un modelo de Reconocimiento Facial que pueda ser utilizado como segundo método de autenticación. Este planteamiento responde a la pregunta de investigación ¿Cómo puede la inteligencia artificial basada en técnicas de reconocimiento facial ayudar a fortalecer la seguridad y eficiencia como un segundo factor de autenticación en sistemas informáticos?

Justificación

La investigación sobre el Reconocimiento Facial como Segundo Método de Autenticación responde a la necesidad de fortalecer la seguridad en sistemas informáticos. En efecto, cada vez más se hace imperativo la seguridad de la información y la protección de datos. Con el aumento de las amenazas y el acceso no autorizado a sistemas informáticos, se hace necesario explorar y fortalecer los métodos de autenticación utilizados en diferentes plataformas.

La capacidad de analizar y verificar la identidad a través de características faciales únicas ofrece ventajas como velocidad y comodidad, aspectos clave en la experiencia del usuario. Al utilizar el reconocimiento facial como segundo método de autenticación, se pretende fortalecer la seguridad de los sistemas, reducir los riesgos de acceso no autorizado y mejorar la experiencia del usuario al eliminar la necesidad de recordar contraseñas complejas.

Finalmente, esta investigación tiene como objetivo general implementar y evaluar un modelo de redes neuronales convolucionales (*CNN*) para el reconocimiento facial. Al centrarse en este objetivo, se busca proporcionar una línea base comprensible para aquellos que inician en la ciencia de datos, especialmente en el área de visión por computadora. Metodológicamente, representa un paso fundamental en la exploración de las capacidades de las redes neuronales convolucionales (*CNN)*. El enfoque cuantitativo permitirá una comprensión profunda de los componentes clave, la estructura, el funcionamiento y las técnicas de optimización que hacen que las redes neuronales convolucionales (*CNN*) sean efectivas en el reconocimiento de patrones complejos en imágenes faciales.

Objetivos

Objetivo general

Desarrollar un modelo de inteligencia artificial para el reconocimiento facial utilizando redes neuronales convolucionales y la librería *TensorFlow*.

Objetivos específicos

- Implementar un modelo de redes neuronales convolucionales para el reconocimiento facial.
- Entrenar el modelo de reconocimiento facial utilizando un conjunto de datos preprocesado y estándar.
- Evaluar la precisión y eficiencia del modelo de reconocimiento facial mediante pruebas con un conjunto de datos de validación.
- Optimizar el modelo ajustando hiperparámetros clave como la tasa de aprendizaje, el número de capas y la cantidad de neuronas en cada capa.

Hipótesis

El uso de técnicas avanzadas de *deep learning*, específicamente redes neuronales convolucionales (*CNN*), en el reconocimiento facial como segundo factor de autenticación en sistemas informáticos, mejora significativamente la seguridad y la eficiencia del proceso de autenticación. Este enfoque moderno en visión por computadora reduce la tasa de acceso no autorizado y optimiza el tiempo de autenticación en comparación con métodos tradicionales, proporcionando una solución más robusta y rápida para la protección de sistemas.

Variables del Estudio

1. Imágenes de Personas

En el contexto del reconocimiento facial, estas imágenes se utilizan como entradas para los modelos de *deep learning* para identificar y verificar a los individuos (Zhao et al., 2003).

Tipo de Variable: Variable independiente.

2. Modelo CNN (Red Neuronal Convolucional)

Un modelo de red neuronal convolucional (*CNN*) es una arquitectura de red neuronal profunda diseñada para procesar datos con una estructura de cuadrícula, como imágenes. Las *CNNs* son eficaces en el reconocimiento y clasificación de patrones complejos en imágenes mediante el uso de capas convolucionales, de agrupamiento y de *fully connected* (LeCun et al., 2015).

Tipo de Variable: Variable independiente.

3. Identificación de Individuo

La identificación de un individuo se refiere al proceso de reconocer y verificar la identidad de una persona a partir de sus características faciales en una imagen. En el contexto de reconocimiento facial, esto implica la capacidad del sistema para asociar una imagen facial con un individuo conocido en una base de datos (Zhang et al., 2016).

Tipo de Variable: Variable dependiente.

Fundamentación bibliográfica o marco teórico

Antecedentes

Primero, es fundamental comprender cómo ha progresado el reconocimiento facial a lo largo del tiempo. En sus inicios, las técnicas convencionales dependían de la identificación manual de características, como el análisis de puntos clave. Aunque estos métodos fueron innovadores, enfrentaban importantes limitaciones en cuanto a precisión y capacidad de escalar (Turk y Pentland, 1991). Sin embargo, con el desarrollo de las redes neuronales convolucionales (CNN) y los avances en el

procesamiento de imágenes, este campo ha experimentado una transformación significativa (LeCun et al., 1998).

Por otro lado, los estudios pioneros de la década de 1990, como la introducción de *Eigenfaces* por Turk y Pentland, sentaron las bases del reconocimiento facial automatizado. Estos trabajos demostraron la viabilidad de la tecnología, pero fue el surgimiento del aprendizaje profundo en los años 2010 lo que permitió el desarrollo de modelos más precisos y complejos. En particular, la popularidad de las CNN se disparó con el éxito de AlexNet en la competencia ImageNet en 2012, marcando un antes y un después en el procesamiento de imágenes y el reconocimiento de patrones (Krizhevsky, Sutskever, & Hinton, 2012).

En definitiva, los avances en el reconocimiento facial han sido impulsados por el desarrollo de redes neuronales avanzadas y técnicas de procesamiento de imágenes. Hoy en día, las CNN son fundamentales en los sistemas de reconocimiento facial modernos, ofreciendo una precisión sin precedentes. Este progreso ha abierto nuevas oportunidades en aplicaciones de seguridad, vigilancia y biometría, consolidando el reconocimiento facial como una tecnología esencial en la inteligencia artificial (Taigman et al., 2014).

Marco teórico

Las Redes Neuronales Convolucionales *CNN* son especialmente efectivas para el análisis de imágenes, ya que capturan características espaciales y patrones visuales de manera jerárquica. Este tipo de redes ha revolucionado el procesamiento de imágenes y ha demostrado su eficacia en tareas de reconocimiento de patrones (LeCun et al., 1998).

En términos generales, las Redes Neuronales Convolucionales *CNN* estándar está compuesta por varias capas, que incluyen capas convolucionales, de agrupamiento (*pooling*), y completamente conectadas (*fully connected*). Las capas convolucionales aplican filtros a la imagen de entrada para extraer características locales, mientras que las capas de agrupamiento disminuyen la dimensionalidad y retienen

las características más importantes. Por otro lado, las capas completamente conectadas integran estas características para llevar a cabo la clasificación final. Además, técnicas como la normalización por lotes (*batch normalization*) y la regularización (*dropout*) contribuyen a mejorar la precisión y a evitar el sobreajuste (Ioffe & Szegedy, 2015; Srivastava et al., 2014).

Marco conceptual

En el marco conceptual, es fundamental definir los conceptos clave que sirven de mapa para este estudio que se centrada en la implementación de un modelo de inteligencia artificial basado en redes neuronales convolucionales (*CNN*) utilizando *TensorFlow* para el reconocimiento facial, es crucial cubrir una serie de temas fundamentales. A continuación, se detallan los temas que se deben tratar, organizados de manera que construyan un camino claro y didáctico para el estudio.

La Inteligencia Artificial (IA)

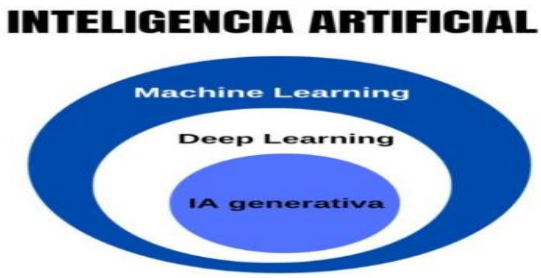
Es una rama de la informática que se dedica a desarrollar sistemas capaces de llevar a cabo tareas que suelen requerir inteligencia humana. Es una tecnología que permite a los ordenadores y máquinas imitar la inteligencia humana y su habilidad para resolver problemas (¿Qué es la Inteligencia Artificial (IA)?, 2021).

Ramas de la Inteligencia Artificial (IA)

La Inteligencia Artificial (IA) se divide en varias subáreas que permiten abordar diferentes tipos de problemas y aplicaciones. Las ramas más destacadas de la IA son el *Machine Learning*, el *Deep Learnign* y la Inteligencia Artificial Generativa.

Figura 1

Ramas de la IA



Fuente: (Abalde, 2023)

La inteligencia artificial (IA) está teniendo un impacto significativo en diversas industrias, transformando la vida cotidiana en formas que antes parecían ciencia ficción.

Visión por Computador

La visión por computador es una disciplina dentro de la inteligencia artificial (IA) que permite a las computadoras y sistemas procesar imágenes digitales, videos y otras entradas visuales para extraer información valiosa, con el fin de tomar decisiones o realizar recomendaciones basadas en esa información. Mientras que el *Machine learning* permite que las computadoras "piensen", la visión por computador les otorga la capacidad de "ver", observar y comprender su entorno.

Machine Learning o Aprendizaje Automático

El Aprendizaje Automático, conocido como *Machine Learning* en inglés, es una subdisciplina de la Inteligencia Artificial que se enfoca en la creación de algoritmos y modelos que capacitan a las máquinas para aprender y mejorar en tareas específicas mediante la experiencia y el análisis de datos, las máquinas en *Machine Learning* reciben datos que les permiten identificar patrones, realizar predicciones o tomar decisiones fundamentadas en esa información. La clave del Aprendizaje Automático reside en la habilidad de las máquinas para auto-mejorarse progresivamente a medida que procesan más datos y ganan experiencia (Abalde, 2023).

Principales técnicas del Machine Learning

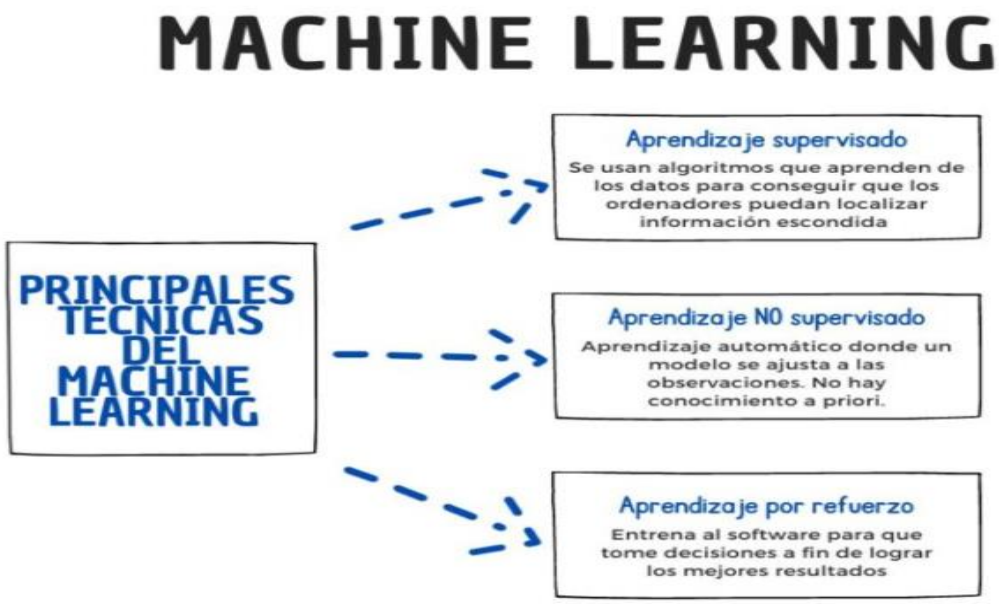
Aprendizaje Supervisado. - Es una técnica que implica entrenar un modelo con un conjunto de datos etiquetados, donde las respuestas correctas ya están disponibles. (González & Martín, 2021).

Aprendizaje No Supervisado. - Trabaja con datos no etiquetados. En este caso, el objetivo del modelo es descubrir patrones o estructuras ocultas en los datos sin tener un conjunto de salidas correctas predefinidas. (Pérez & López, 2020).

Aprendizaje por Refuerzo. - Es una estrategia donde un agente aprende a tomar decisiones interactuando con su entorno y ajustando su comportamiento en función de la retroalimentación que recibe, ya sea en forma de recompensas o castigos. (Sutton & Barto, 2018).

Figura 2

Principales técnicas del Machine Learning



Fuente: (Abalde, 2023).

Figura 3

Aplicaciones del Machine Learning

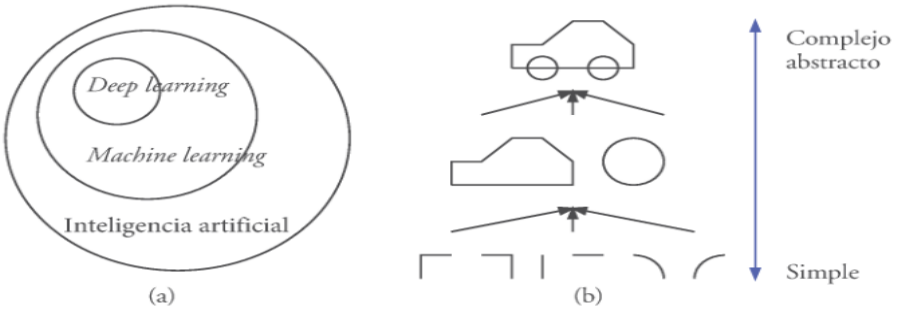
Fuente: (Abalde, 2023).

Deep Learning o Aprendizaje Profundo

El *Deep Learning* es una rama específica dentro del *Machine Learning* que se enfoca en imitar la forma en que el cerebro humano procesa datos complejos, como imágenes o texto. Utilizando redes neuronales artificiales, el *Deep Learning* es capaz de aprender y detectar patrones de manera autónoma y en un nivel profundo. La esencia del *Deep Learning* radica en observar el cerebro humano e inspirarse en su estructura y funcionamiento para replicar, mediante modelos matemáticos e informáticos, su capacidad de procesamiento y aprendizaje (Abalde, 2023).

Figura 4

Relación del deep learning con otros conceptos (a) y ejemplo de jerarquía para la generación de conceptos complejos (b)



Fuente: (Bosch et al., 2019).

El concepto de *deep learning*, o aprendizaje profundo, se refiere a una rama del aprendizaje en la que se busca crear modelos capaces de representar conceptos complejos o abstractos a partir de otros

más simples. Por ejemplo, a partir de líneas, cruces y arcos, se puede definir el concepto de coche como una combinación de estos elementos más simples. Cuando esta jerarquía de conceptos tiene múltiples capas, se habla de la "profundidad" del modelo, de donde surge el término "aprendizaje profundo" o *deep learning* **(Bosch et al., 2019).**

El Deep Learning utiliza una variedad de técnicas, siendo algunas de las más destacadas:

- Redes Neuronales Convolucionales (CNN)
- Redes neuronales recurrentes (RNN)
- Redes Neuronales Generativas Antagónicas (GANs)

El Perceptrón

El Perceptrón es un modelo básico de red neuronal y representa una de las formas más sencillas dentro del campo de la inteligencia artificial. Fue creado por Frank Rosenblatt en 1957 y desempeña un papel esencial en el aprendizaje supervisado. Este modelo emula una neurona artificial, con el propósito de clasificar entradas binarias en una de dos categorías posibles.

Aplicaciones y Limitaciones

El Perceptrón es eficaz en la resolución de problemas de clasificación que son linealmente separables, es decir, cuando las clases pueden ser divididas mediante una línea recta (en dos dimensiones) o un hiperplano (en dimensiones superiores). Sin embargo, no es capaz de manejar problemas más complejos en los que las clases no se pueden separar linealmente, como el problema XOR. Esta limitación condujo al desarrollo de modelos más avanzados, como las redes neuronales multicapa (Lara, 2020).

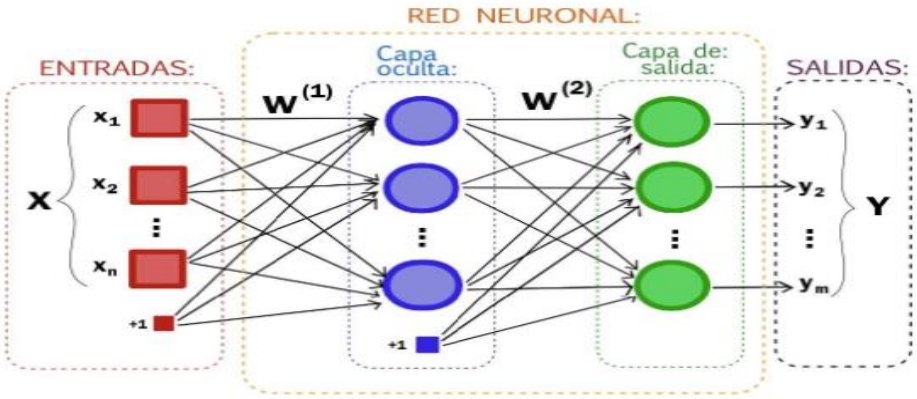
Redes Neuronales Convolucionales (*Convolutional Neural Network* CNN)

Estas redes consisten en varias capas de nodos, que incluyen una capa inicial, una o más capas intermedias, y una capa final. Los nodos dentro de la red están interconectados y se les asignan pesos y umbrales específicos. Si la salida de un nodo excede el umbral establecido, el nodo se activa y transmite

la información a la siguiente capa. Si el umbral no es superado, la información no se transfiere a la capa siguiente (IBM, 2024).

Figura 5

Arquitectura red neuronal

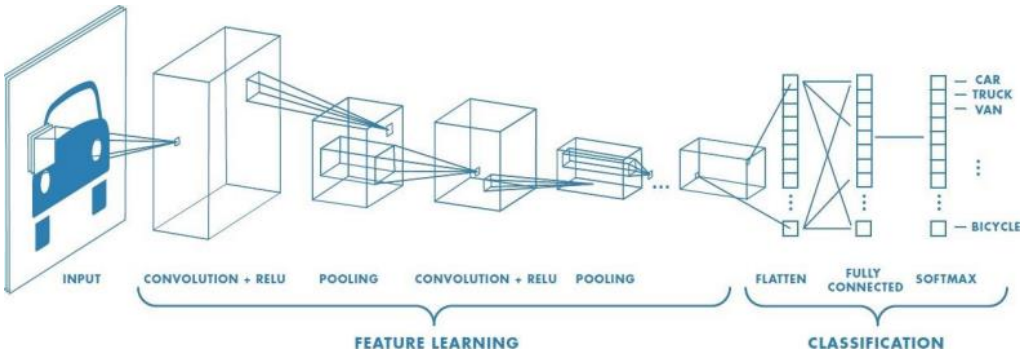


Fuente: (Ahmed, 2019).

La arquitectura básica de una CNN

Figura 6

Arquitectura básica de una CNN



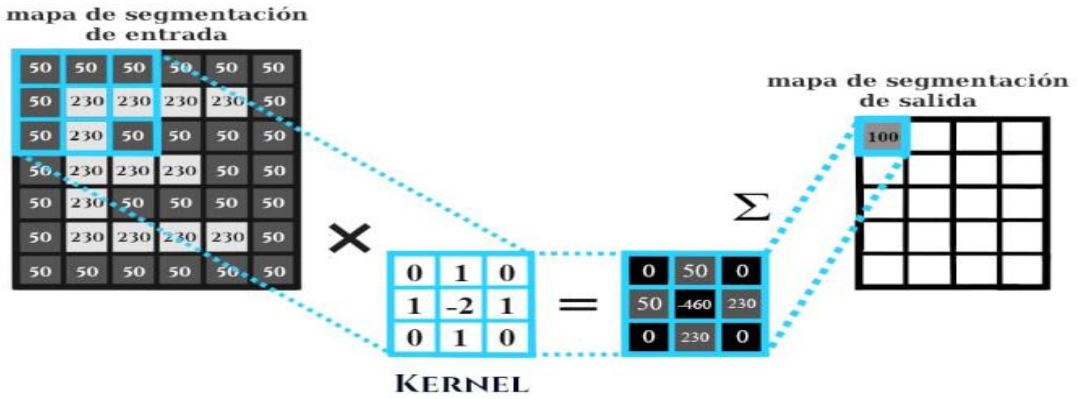
Fuente: (Pinar, 2024).

La arquitectura básica de una CNN (Red Neuronal Convolucional) está compuesta por varias capas organizadas para procesar datos visuales de manera efectiva. Estas capas incluyen:

Capa Convolucional: Es la primera capa principal de una CNN, donde se aplican filtros o *kernels* a la imagen de entrada para extraer características importantes como bordes, texturas y patrones. Cada filtro se desliza sobre la imagen y produce un mapa de características (*feature map*) que destaca las características detectadas.

Figura 7

Operación de convolución



Fuente: (Abril, 2021)

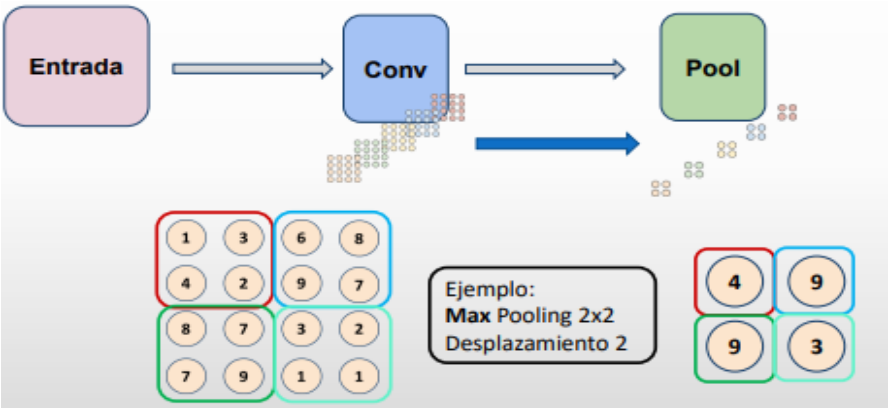
Los valores de cada píxel toman valores de 0 a 256, de más oscuro a más claro. El campo de *kernel* es de 3×3 dimensiones, lo cual resulta en un mapa de tamaño 4×5.

Al comienzo del proceso de convolución, la red neuronal se enfoca en identificar elementos básicos, como líneas y curvas. A medida que se suman más capas, la red desarrolla la capacidad de reconocer formas más complejas.

Capa de *Pooling*: Una capa de *pooling* reduce la dimensionalidad de los mapas de características generados. El *pooling* puede ser de tipo máximo (*max pooling*) o promedio (*average pooling*), y su objetivo es reducir el tamaño de los mapas, manteniendo las características más importantes y reduciendo el riesgo de sobreajuste. La operación más común en esta capa es el *max-pooling*, que segmenta la imagen de entrada en varias áreas rectangulares y, de cada una de estas subregiones, se extrae el valor máximo.

Figura 8

Función de agrupación por subregiones

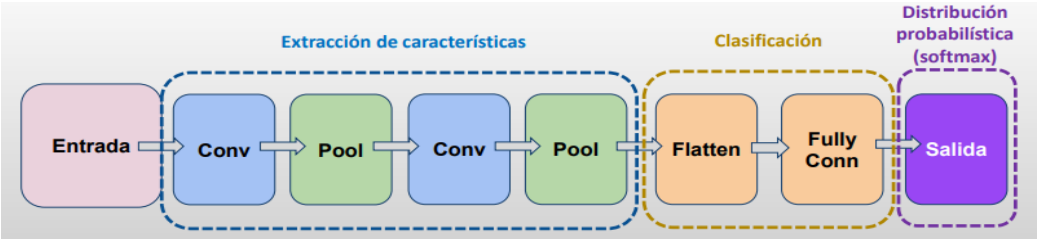


Fuente: (Pinar, 2024).

Al aplicar una ventana de *pooling* de 2x2 con un desplazamiento de 2, se logra eliminar el 75% de los parámetros.

Figura 9

Capas pooling en una CNN

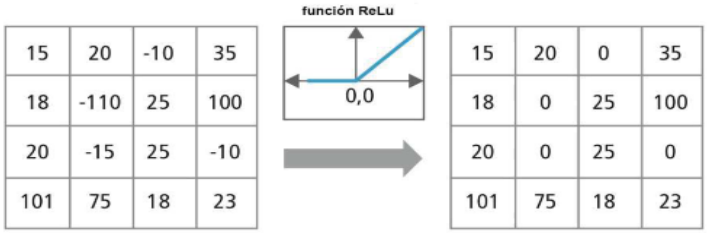


Fuente: (Pinar, 2024).

Capa de Activación: Generalmente, después de cada capa convolucional, se aplica una función de activación como ReLU (*Rectified Linear Unit*) para introducir no linealidad en el modelo, lo que permite a la red aprender patrones más complejos. La función aplica un umbral a cada elemento, estableciendo en cero cualquier valor de entrada que sea menor a cero.

Figura 10

Operación función ReLu



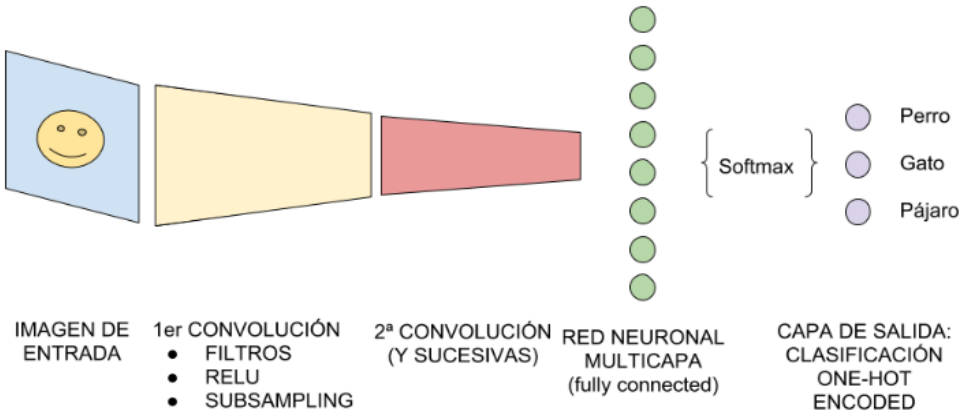
Fuente: (Galárraga Cañizares, 2017).

Capa Completamente Conectada (*Fully Connected Layer*): Al final de la red, las características extraídas y procesadas por las capas anteriores se aplanan (*flattening*) y se pasan a través de una o más capas completamente conectadas.

Capa de Salida: La última capa de la *CNN* es una capa completamente conectada que produce la salida final, que puede ser una probabilidad de clase en tareas de clasificación o un valor continuo en tareas de regresión (Espinosa, 2018) .

Figura 11

Capa Salida



Fuente: (Bagnato, 2018).

En el ejemplo ilustrado en la Figura 12, se utiliza una función conocida como *Softmax*, la cual se conecta a la capa de salida final de la red neuronal. El número de neuronas en esta capa de salida corresponderá a la cantidad de clases que se están clasificando.

Retropropagación: Su función principal es ajustar los pesos internos de la red durante el proceso de aprendizaje, con el fin de reducir al máximo la diferencia entre las predicciones de la red y los resultados esperados. Es un procedimiento matemático que se lleva a cabo automáticamente durante el entrenamiento de la red, implementado a través de *frameworks* como *TensorFlow*, *PyTorch*, o *Keras*.

Comparación entre Redes Neuronales Convencionales y Redes Neuronales Convolucionales

Tabla 1

Comparación de Redes Neuronales

Aspecto	**Red Neuronal Convencional**	**Red Neuronal Convolucional (CNN)**
Datos de entrada en la capa inicial	Características específicas como ancho, alto, grosor, etc.	Pixeles de una imagen. Si es en color, se usan tres canales para rojo, verde, azul.
Capas ocultas	Cantidad determinada de neuronas en las capas ocultas.	Incluyen tipos como convolución (configurada con tamaño de *kernel* y número de filtros) y *subsampling*.
Capa de salida	Número de neuronas que corresponde a la cantidad de clases a clasificar. Ej: 'comprar' o 'alquilar' = 2 neuronas.	La última convolución se aplana y se conecta con capas tradicionales. Luego se usa SoftMax para la clasificación final.
Aprendizaje	Supervisado	Supervisado
Interconexiones	Todas las neuronas de una capa están conectadas con todas las neuronas de la siguiente capa.	Se requieren menos conexiones, ya que se ajustan los pesos de los filtros o *kernels*.
Significado de la cantidad de capas ocultas	El número de capas ocultas no tiene un significado específico y su cantidad es experimental.	Las capas ocultas son mapas jerárquicos que detectan características de la imagen, desde líneas hasta formas complejas.
Retropropagación (*Backpropagation*)	Ajusta los pesos de todas las interconexiones entre capas.	Ajusta los pesos de los filtros o *kernels* aplicados en las operaciones de convolución.

Fuente: (Bagnato, 2018).

Métricas de evaluación

La Matriz de confusión: Esta matriz desglosa las predicciones del modelo, mostrando en sus columnas el número de predicciones para cada clase y en las filas las instancias que realmente corresponden a cada clase.

Figura 12

Matriz de confusión

		PREDICCIÓN	
		Positiva	**Negativa**
VALOR REAL	**Positivo**	Verdadero positivo (TP)	Falso negativo (FN)
	Negativo	Falso positivo (FP)	Verdadero Negativo (TN)

Fuente: Elaboración propia.

Verdadero positivo (TP): El valor real es positivo y la predicción indica que el resultado era positivo. O bien una persona está enferma y la predicción así lo demuestra.

Verdadero negativo (TN): El valor real es negativo y la predicción indica que el resultado era negativo. O bien la persona no está enferma y la predicción así lo demuestra.

Falso negativo (FN): El valor real es positivo, y la predicción indica que el resultado es negativo. La persona está enferma, pero la predicción dice de manera incorrecta que no lo está.

Falso positivo (FP): El valor real es negativo, y la predicción indica que el resultado es positivo. La persona no está enferma, pero la predicción dice de manera incorrecta que si lo está.

Estas definiciones son fundamentales para entender las métricas que se derivan de la matriz de confusión, tales como la Exactitud, Precisión, Sensibilidad y Especificidad (Arce, 2019).

Matriz de Confusión Métrica

***Accuracy* (Exactitud):** La exactitud es una métrica que mide la fracción de predicciones correctas respecto al total de las instancias. Es una medida sencilla y ampliamente utilizada para evaluar el rendimiento de un modelo de clasificación, especialmente cuando las clases están equilibradas.

La fórmula es:

$$Accuracy = \frac{TP + TN}{P + N} = \frac{TP\ +\ TN}{TP\ +\ TN\ +\ FP\ +\ FN}$$

No obstante, dicha métrica presenta un problema para conjuntos de datos en los que la proporción de instancias de cada clase sea desbalanceada. Es por este motivo, que se usarán otras métricas cuya información sea más confiable.

Precision **(Precisión):** La precisión es una métrica que mide la fracción de positivos reales entre los ejemplos que se predicen como positivos. Responde a la pregunta ¿Qué porcentaje de las predicciones positivas fue acertado en realidad?

La fórmula es:

$$Precision = \frac{TP}{TP\ +\ FP}$$

Recall **(Sensibilidad):** La sensibilidad indica la fracción de instancias positivas correctamente clasificadas. Por este motivo también puede llamarse *true positive rate (TPR)*. Responde a la pregunta ¿Cuál es el porcentaje de casos positivos reales que fueron correctamente identificados?

La fórmula es:

$$Recall\ = \frac{TP}{P} = \frac{TP}{TP + FN}$$

Specifity (Especificidad): La especificidad indica la fracción de instancia negativas cuyo estado real es también negativo. Es lo opuesto a la sensibilidad. La fórmula es:

La fórmula es:

$$Specifity\ = \frac{TN}{N} = \frac{TN}{TN\ +\ FP}$$

F1-Score **(Puntuación F1):** El F1-Score es una métrica que combina la precisión y la sensibilidad (recall) en una única puntuación armónica. Se utiliza especialmente cuando hay un desequilibrio entre las clases en los datos. El F1-Score es útil porque ofrece un balance entre las dos métricas, destacando el

rendimiento del modelo en términos de la relación entre los verdaderos positivos, los falsos positivos y los falsos negativos.

La fórmula es:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Todas estas métricas parten de los valores obtenidos en las métricas TP, TN, FP y FN (Ahmed, 2019).

Generalización del Conocimiento

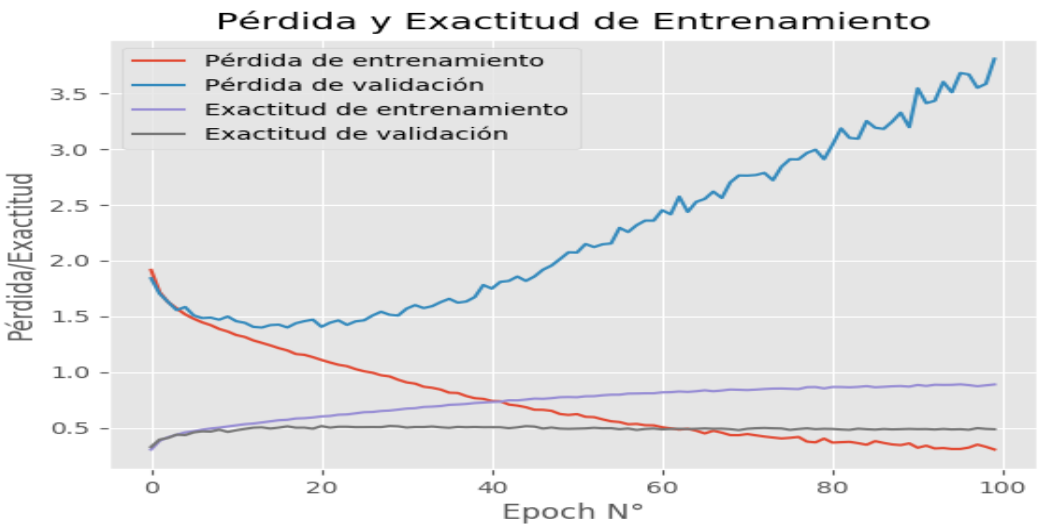
La generalización del conocimiento es crucial en el aprendizaje automático, ya que se refiere a la capacidad de un modelo para aplicar lo aprendido a nuevos datos que no se utilizaron durante su entrenamiento.

***Overfitting* (Sobreajuste) en Redes Neuronales Convolucionales (*CNN*)**

El *overfitting* ocurre cuando un modelo de CNN se adapta excesivamente a los datos de entrenamiento. Esto significa que el modelo no solo aprende los patrones generales de los datos, sino también las peculiaridades y el ruido específico de los mismos. Como resultado, el modelo funciona bien en el conjunto de datos de entrenamiento, pero su rendimiento se desploma cuando se le presentan nuevos datos, es decir, no generaliza bien (Bagnato, 2017).

Figura 13

Pérdida y Exactitud de Entrenamiento



Fuente: (Martínez, 2019).

Cuando hay una brecha significativa entre el rendimiento en los datos de entrenamiento y los de validación, es un indicio de que nuestro modelo está "sobreajustando".

Para evitar el *overfitting* en CNNs, se pueden utilizar diversas técnicas. Una de las más comunes es la regularización, que penaliza la complejidad del modelo y evita que aprenda patrones demasiado específicos. Otra estrategia es el *dropout*, donde se omiten aleatoriamente algunas conexiones neuronales durante el entrenamiento, lo que obliga al modelo a aprender patrones más robustos y menos dependientes de características específicas. Además, el aumento de datos (*data augmentation*), como rotar, escalar o reflejar imágenes, puede ayudar a que el modelo aprenda a generalizar mejor.

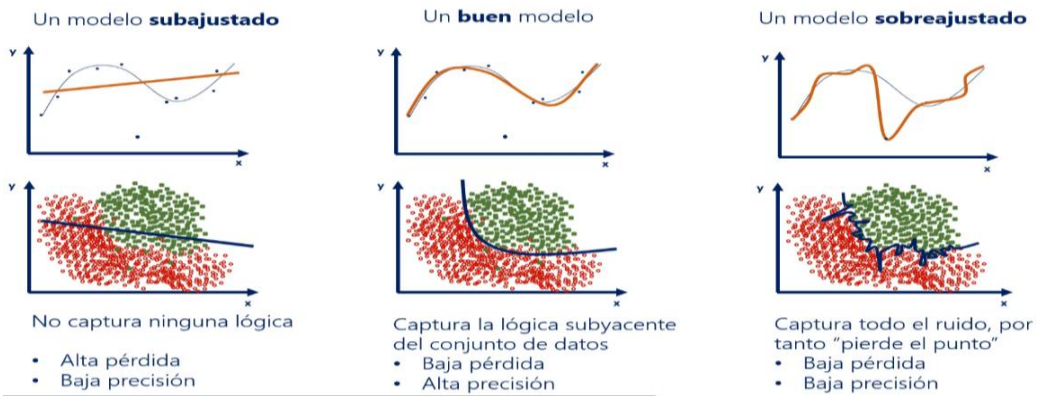
***Underfitting* (Subajuste) en Redes Neuronales Convolucionales (*CNN*)**

El subajuste en redes neuronales convolucionales (CNN) ocurre cuando el modelo es demasiado simple para captar las relaciones complejas presentes en los datos de entrenamiento. Esto puede suceder si el modelo tiene pocas capas, insuficientes neuronas, o no se entrena el tiempo necesario, lo que da como resultado un modelo que no se ajusta bien ni a los datos de entrenamiento ni a los nuevos datos (GeeksforGeeks, 2017).

Para evitar el subajuste, es esencial asegurarse de que el modelo tenga la complejidad adecuada para la tarea que se está abordando. Esto puede implicar añadir más capas o neuronas, o ajustar parámetros como la tasa de aprendizaje. También es fundamental darle al modelo suficiente tiempo de entrenamiento para que pueda aprender correctamente a partir de los datos. Además, utilizar técnicas como el ajuste de hiperparámetros puede ayudar a encontrar el equilibrio adecuado entre la capacidad del modelo y su rendimiento (GeeksforGeeks, 2017).

Figura 14

El equilibrio del Aprendizaje



Fuente: (365 Careers, 2020).

TensorFlow

TensorFlow desde su lanzamiento en 2015, se ha convertido en una herramienta fundamental para la construcción y despliegue de modelos de inteligencia artificial (IA) (Ortego, 2017).

Introducción a los Tensores

Un tensor es una estructura de datos que generaliza los conceptos de escalares, vectores y matrices a dimensiones más elevadas.

Figura 15

Tensor

Fuente: (Méndez, 2021).

En *TensorFlow*, los tensores se utilizan para representar todo tipo de datos, desde simples números hasta complejas imágenes de varias dimensiones.

Importancia de los Tensores en *TensorFlow*

Cada tensor fluye a través de un grafo computacional que describe las operaciones matemáticas necesarias para el entrenamiento y la inferencia de modelos.

Keras

Keras se distingue por su alto nivel de abstracción, facilidad de uso y flexibilidad. *Keras* es una librería de alto nivel se puede utilizar con *TensorFlow,* lo hace fácil de usar para desarrolladores de todos los niveles (López, 2024) .

Implementación de *CNN* en *TensorFlow*

TensorFlow es una herramienta para implementar redes neuronales convolucionales (*CNN*) gracias a su flexibilidad y potencia. Con la ayuda de su API de alto nivel, *Keras*, es posible crear modelos *CNN* de manera eficiente y con menos esfuerzo de codificación.

Una *CNN* básica en *TensorFlow* podría incluir los siguientes tipos de capas:

- **Capas Convolucionales**: Estas capas aplican filtros a las imágenes para identificar características locales, como bordes y texturas.
- **Capas de *Pooling***: Estas capas disminuyen la dimensionalidad de las características detectadas, lo que ayuda a conservar solo la información más importante y a reducir el riesgo de sobreajuste.

- **Capas Completamente Conectadas** (*Fully Connected*): Después de extraer las características, estas capas se encargan de clasificarlas en distintas categorías.

Estructura de un Proyecto en *TensorFlow*

Un proyecto típico en *TensorFlow* sigue una serie de pasos claramente definidos, que son esenciales para crear, entrenar y evaluar modelos de aprendizaje profundo de manera efectiva. A continuación, se describe el flujo común de un proyecto en *TensorFlow*.

1. **Preparación de los Datos**

En *TensorFlow*, los datos se manejan generalmente como tensores, que son estructuras que permiten organizar grandes cantidades de información numérica en varias dimensiones. Es crucial dividir los datos en conjuntos de entrenamiento y prueba, y en muchos casos, normalizarlos para optimizar el rendimiento del modelo (López, 2024).

Por ejemplo, cuando se trabaja con imágenes, un paso común es la normalización de los datos, que ajusta los valores de los píxeles a un rango entre 0 y 1, lo cual mejora la precisión y estabilidad del modelo durante su entrenamiento (TensorFlow, 2022).

2. **Definición del Modelo**

En *TensorFlow*, esto se realiza principalmente con la *API* de *Keras*, que facilita la construcción de modelos de manera modular y accesible. Una *CNN* típica puede incluir varias capas convolucionales seguidas de capas de *pooling*, que ayudan a reducir la cantidad de información procesada, manteniendo las características más relevantes. Estas capas se organizan en secuencia para crear una red profunda capaz de aprender patrones complejos a partir de los datos de entrada (López, 2024).

3. **Compilación del Modelo**

Este paso consiste en seleccionar un optimizador, como Adam, y definir una función de pérdida, como la entropía cruzada. También se especifican las métricas que se usarán para medir el rendimiento del modelo, tanto durante el entrenamiento como en la evaluación final (TensorFlow, 2022).

4. Entrenamiento y Evaluación del Modelo

En *TensorFlow*, este proceso se lleva a cabo en múltiples iteraciones, conocidas como épocas, en las que el modelo va mejorando progresivamente su capacidad para hacer predicciones correctas basadas en los datos que se le han proporcionado (López, 2024).

Técnicas de Mejora y Regularización

- **Aumento de Datos (*Data Augmentation*):** Consiste en aplicar transformaciones aleatorias a las imágenes de entrenamiento, como rotaciones y zoom, para aumentar la diversidad del conjunto de datos y mejorar la generalización del modelo (TensorFlow Core, 2024).
- ***Dropout*:** Es una técnica de regularización que consiste en desactivar aleatoriamente algunas neuronas durante el entrenamiento, lo que obliga al modelo a aprender características más robustas y evita que dependa demasiado de ciertas características específicas (Datacamp, 2023).

Metodología *SEMMA*

SEMMA es un marco conceptual ampliamente utilizado en el análisis de datos y modelado predictivo, desarrollado por *SAS Institute*. Su nombre es un acrónimo que describe las cinco etapas fundamentales de este enfoque. *SEMMA* está diseñado para guiar el proceso de descubrimiento de patrones y relaciones en los datos, y se utiliza comúnmente en proyectos de *data mining* y *machine learning* (Romero, 2020).

Muestreo (*Sample*). - En esta etapa, se elige una porción representativa del conjunto de datos disponible. Este subconjunto debe ser lo suficientemente grande para capturar toda la información relevante y reflejar las características principales del conjunto completo, pero lo suficientemente pequeño para que su análisis sea práctico y manejable (Romero, 2020).

Exploración (*Explore*). - Durante la fase de exploración, se examinan los datos seleccionados para identificar patrones, relaciones y posibles anomalías. Este proceso generalmente utiliza técnicas de

visualización y estadísticas descriptivas para obtener una comprensión más detallada y profunda del conjunto de datos y sus características (Romero, 2020).

Modificación (*Modify*). - La fase de modificación se enfoca en preparar y transformar los datos para su análisis final. Esto implica actividades como la limpieza de datos, normalización, transformación de variables y la creación de nuevas variables derivadas. El objetivo es mejorar la calidad del conjunto de datos para que el modelo funcione de manera óptima (Romero, 2020).

Modelado (*Model*). - En esta etapa, se aplican técnicas de modelado predictivo para descubrir patrones en los datos que puedan ser aplicables a nuevos conjuntos de datos. Se desarrollan, entrenan y ajustan modelos de aprendizaje automático, optimizando los hiperparámetros y seleccionando los algoritmos más adecuados (Romero, 2020).

Evaluación (*Assess*). - Finalmente, en la fase de evaluación, se analiza el rendimiento del modelo utilizando métricas específicas para comprobar su capacidad de generalizar a nuevos datos. Esto incluye revisar la precisión, sensibilidad, especificidad y otras métricas relevantes para determinar la efectividad del modelo (Romero, 2020).

Objetivo

El objetivo de esta investigación es implementar un modelo de inteligencia artificial basado en Redes Neuronales Convolucionales utilizando la librería *TensorFlow* para técnicas de reconocimiento facial. Esta implementación busca explorar y comprender el funcionamiento de las *CNN*, optimizar su rendimiento y evaluar su eficacia en la identificación de individuos. No se pretende crear una mejor *CNN* de las que ya existen si no saber cuál es el funcionamiento de esta.

Método

Diseño

El presente estudio se basa en un enfoque cuantitativo con un diseño experimental Cuasiexperimental, dirigido a desarrollar y evaluar un modelo de reconocimiento facial utilizando redes

neuronales convolucionales (*CNN*) como técnica principal dentro del ámbito de la visión por computador. La selección de este diseño fue considerada la más apropiada dado que el objetivo del presente estudio es implementar un modelo que sea capaz de identificar individuos a partir de imágenes faciales, considerando que se llevará a cabo una serie de pruebas controladas para evaluar el rendimiento del modelo CNN en diferentes condiciones. Estas pruebas incluirán la manipulación de variables independientes, como el conjunto de datos de imágenes y las variaciones en las expresiones faciales, para observar su efecto sobre la variable dependiente, que es la clasificación facial de una persona.

Participantes

Descripción de las Características

La unidad de análisis en esta investigación se centra en un conjunto de datos compuesto por imágenes faciales de 8 personas, donde se incluyen 4 adultos, 2 jóvenes, 1 adolescente y 1 niños. Cada imagen tiene una resolución de 160x160 píxeles y está en formato JPG a color.

Selección de Muestreo

Para este estudio, se ha elegido a un grupo de personas que representan una variedad de edades y rasgos faciales. Cada una de las 8 personas incluidas en el conjunto de datos ha aportado 650 fotografías. Con esta selección, se busca que el modelo pueda reconocer rostros de manera efectiva en un entorno controlado y con distintos tipos de rostros.

Normas Éticas

En cuanto a las normas éticas, se ha asegurado que todas las imágenes utilizadas en este estudio han sido recopiladas con el consentimiento informado de los participantes. Se han implementado medidas para garantizar que los datos se manejen de manera segura y anónima, previniendo cualquier riesgo de identificación no autorizada de los individuos involucrados. Además, el uso de imágenes de niños ha sido tratado con especial cuidado, siguiendo las directrices éticas y legales pertinentes para la protección de menores en investigaciones científicas.

Instrumentos

Laptop HP con Cámara Integrada (24GB RAM, Core i7), esta laptop se utilizará como el hardware principal para capturar imágenes faciales y ejecutar los modelos de reconocimiento facial.

OpenCV para Captura de Rostros, será la herramienta principal para la detección y captura de rostros en las imágenes. Se empleará para asegurar que las imágenes capturadas sean consistentes y adecuadas para el entrenamiento del modelo.

***TensorFlow* y *Keras*,** Estas bibliotecas se emplearán para implementar, entrenar y evaluar el modelo de redes neuronales convolucionales.

Python y Jupyter Notebook, *Python* el lenguaje de programación utilizado para desarrollar todo el proyecto, incluyendo el preprocesamiento de imágenes, y la implementación del modelo. *Jupyter* Notebook facilitará la organización y presentación del código.

Procedimiento

El marco SEMMA proporciona una estructura clara y lógica para guiar el proceso de análisis de datos y modelado predictivo. Su enfoque en etapas secuenciales asegura que cada paso se realice de manera organizada. En el contexto de la investigación "Modelo de Reconocimiento Facial con Redes Neuronales: Un enfoque de visión por Computador," SEMMA ofrece un marco útil para estructurar el desarrollo del modelo CNN.

1. **Muestreo (*Sample*)**
    1.1. Se definió el objetivo de desarrollar y evaluar un modelo de redes neuronales convolucionales (CNN) orientado al reconocimiento facial.
    1.2. Selección de la Muestra, se decidió trabajar con un grupo de personas, incluyendo 4 adultos, 2 jóvenes, 1 adolescente y 1 niño.

Cada participante fue fotografiado 650 veces bajo un entorno contralado, condiciones de luz y con distintas expresiones faciales. Las fotos fueron guardadas en formato JPG y tienen una resolución de 160x160 píxeles.

Figura 16

Rostros con distintas expresiones faciales

Además, para la detección de rostros en las imágenes, se utilizó un clasificador de rostros pre-entrenado, *haarcascade_frontalface*, proporcionado por *OpenCV*. Este clasificador permitió identificar y recortar las regiones faciales de manera automática, asegurando que los datos alimentados al modelo estuvieran enfocados únicamente en las áreas de interés.

Figura 17

Captura rostro con haarcascade_frontalface

Para mantener un orden claro y facilitar el manejo de las imágenes en etapas posteriores, se organizó el material en 8 carpetas, una por cada participante.

Figura 18

Set de fotos organizadas

2. Exploración de los Datos (Explore)

Verificación de la resolución de las imágenes, asegurando que todas estuvieran correctamente dimensionadas a 160x160 píxeles y que el formato fuera uniforme (JPG), lo que facilitó un procesamiento homogéneo.

Figura 19

Tipo y tamaño de las imágenes

El clasificador pre entrenado *haarcascade_frontalface* de *OpenCV* jugó un papel fundamental, permitiendo identificar y aislar las regiones faciales de cada imagen.

3. Modificación de los Datos (*Modify*)

3.1. División Inicial del Conjunto de Datos

En un inicio, el conjunto de datos fue dividido manualmente en dos subconjuntos:

- Conjunto de Entrenamiento: 80% de las imágenes disponibles.
- Conjunto de Pruebas: 20% de las imágenes.

Tras el entrenamiento inicial del modelo con la división manual, se observó que los indicadores de rendimiento *(precisión, recall, F1-score*, etc.) alcanzaban niveles muy altos, llegando al 100% en varios

casos. Aunque este rendimiento podría parecer ideal, en contextos de aprendizaje profundo, estos resultados pueden indicar problemas de sobreajuste (*overfitting*), donde el modelo memoriza las imágenes en lugar de aprender patrones generales.

3.2. Automatización de la División y Aumento de Datos

Para abordar un posible sobreajuste (*overfitting*) y mejorar la generalización del modelo, se decidió utilizar las funcionalidades avanzadas del *framework* empleado (*TensorFlow* con *Keras*) para realizar una división automática del conjunto de datos y aplicar técnicas de aumento de datos (*Data Augmentation*).

3.2.1. **División Automática**

El *framework* se encargó de dividir automáticamente el conjunto de datos en entrenamiento y pruebas. Esta división automatizada permitió una aleatorización más adecuada, reduciendo la posibilidad de que el modelo dependiera de características específicas de la división manual inicial.

3.2.2. Aumento de Datos

Se implementaron técnicas de aumento de datos durante el entrenamiento, tales como rotaciones, cambios de escala, modificaciones en la iluminación, y otras transformaciones geométricas. Estas técnicas tienen el objetivo de simular variaciones que podrían encontrarse en escenarios reales, mejorando la capacidad del modelo para generalizar a nuevas imágenes.

```
# Crear generadores de datos con aumento de datos agresivo
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2
)
```

Este conjunto de transformaciones se aplicó de forma aleatoria en cada lote de datos, lo que aumentó la variedad de las imágenes de entrenamiento.

4. Modelado (Model)

4.1. Arquitectura

El modelo desarrollado para esta investigación sigue una arquitectura típica de redes neuronales convolucionales (CNN), la cual es especialmente adecuada para tareas de visión por computador, como el reconocimiento facial.

```
model = models.Sequential([
    layers.Rescaling(1./255, input_shape=(160, 160, 3)),

    layers.Conv2D(32, (3, 3), activation='relu', padding='same',
kernel_regularizer=regularizers.l2(0.001)),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.3),

    layers.Conv2D(64, (3, 3), activation='relu', padding='same',
kernel_regularizer=regularizers.l2(0.001)),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.4),

    layers.Conv2D(128, (3, 3), activation='relu', padding='same',
kernel_regularizer=regularizers.l2(0.001)),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.5),

    layers.Flatten(),
    layers.Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
    layers.Dropout(0.5),
    layers.Dense(len(class_names), activation='softmax')
])
```

A continuación, se describe la arquitectura del modelo implementado:

4.1.1. *Rescaling Layer*

La primera capa aplica una normalización de los valores de los píxeles, escalándolos al rango [0, 1] desde su rango original [0, 255]. Esto facilita la convergencia durante el entrenamiento.

4.1.2. Capas Convolucionales

- **Primera Capa Convolucional:** Conv2D con 32 filtros de tamaño 3x3, activación ReLU y padding 'same', lo que preserva las dimensiones de entrada. Se incluye regularización L2 para prevenir el sobreajuste.

- **Segunda Capa Convolucional:** Conv2D con 64 filtros de tamaño 3x3, también con activación ReLU, padding 'same', y regularización L2.
- **Tercera Capa Convolucional:** Conv2D con 128 filtros de tamaño 3x3, con las mismas configuraciones de activación y padding, incluyendo regularización L2.

4.1.3. Capas de *MaxPooling*

Después de cada capa convolucional, se aplica una capa de *MaxPooling* con un filtro de 2x2, que reduce las dimensiones espaciales de las características y ayuda a generalizar mejor el modelo.

4.1.4. Capas de *Dropout*

Se utilizan capas de *Dropout* con tasas de 0.3, 0.4 y 0.5 después de las capas de *pooling* y la capa densa. El *Dropout* es una técnica de regularización que desconecta aleatoriamente una fracción de neuronas durante el entrenamiento para reducir el sobreajuste del modelo a los datos de entrenamiento.

4.1.5. Capa de *Flatten*

Esta capa convierte la salida tridimensional de las capas convolucionales y de *pooling* en un vector unidimensional, preparando los datos para las capas densas subsecuentes.

4.1.6. Capas Densas

- **Primera Capa Densa:** Compuesta por 256 neuronas con activación *ReLU* y regularización L2.
- **Capa de Salida:** Una capa densa final con una función de activación *softmax* que produce una distribución de probabilidad sobre las clases disponibles, permitiendo al modelo hacer predicciones categóricas.

4.2. Entrenamiento del Modelo

A continuación, se detalla las configuraciones finales que se aplicaron en los hiperparámetros que se llevó a cabo en este proceso:

4.2.1. Optimización del Modelo

Se utilizó el optimizador ***RMSprop*** para ajustar los pesos del modelo final. *RMSprop* fue seleccionado por su capacidad para manejar problemas con alta variabilidad en las actualizaciones de gradientes, una característica común en redes neuronales profundas. La tasa de aprendizaje se configuró en **0.0001**, lo que permitió realizar ajustes en cada iteración, tratando de asegurando que el modelo convergiera de manera eficiente sin saltos bruscos que pudieran provocar un estancamiento en mínimos locales no óptimos.

4.2.2. Función de Pérdida

La función de pérdida implementada fue ***sparse_categorical_crossentropy***, adecuada para la clasificación multiclase con etiquetas enteras. Esta función mide la discrepancia entre las predicciones del modelo y las etiquetas reales, proporcionando una métrica clara para guiar el ajuste de los parámetros del modelo durante el entrenamiento.

4.2.3. *Callbacks* Utilizados

Para mejorar el proceso de entrenamiento y tratar de asegurar que el modelo alcanzara su mejor rendimiento posible, se implementaron dos *callbacks*:

4.2.3.1. ***Early Stopping***

- Se configuró para monitorear la pérdida en el conjunto de validación (*val_loss*).
- Con una paciencia de **20 épocas**, el entrenamiento se detendría automáticamente si no se observaba una mejora en la pérdida de validación durante 20 épocas consecutivas.

4.2.3.2. ***ReduceLROnPlateau***

- Este *callback* redujo la tasa de aprendizaje en **0.5** si la pérdida de validación no mejoraba durante **5 épocas** consecutivas, con un límite inferior de **0.00001** para la tasa de aprendizaje.
- Esta estrategia ayudó a que el modelo final siguiera aprendiendo de manera efectiva, incluso cuando el progreso se volvía más lento en las etapas avanzadas del entrenamiento.

4.2.3.3. **Entrenamiento del Modelo Final**

El modelo fue entrenado utilizando los generadores de datos de entrenamiento y validación, que habían sido previamente configurados para incluir aumento de datos agresivo. El entrenamiento se realizó durante un máximo de **120 épocas**. Sin embargo, gracias a la implementación de *Early Stopping*, el proceso de entrenamiento podría detenerse antes si no se observaba una mejora continua.

5. Evaluación (*Assess*)

El modelo final fue evaluado utilizando un conjunto de datos de prueba que no fue utilizado durante el entrenamiento. Se utilizaron métricas para evaluar el rendimiento del modelo como son: Precisión (*Accuracy*), Pérdida (*Loss*), Matriz de Confusión, Análisis de Sobreajuste (*Overfitting*).

Este modelo final representó el resultado de múltiples iteraciones y ajustes en su arquitectura y parámetros, tratando de garantizar que cumpliera con los objetivos de la investigación. Los resultados obtenidos durante el proceso de entrenamiento, incluyendo las métricas de pérdida y precisión, se presentan en detalle en la sección de resultados para evaluar la eficacia y la capacidad del modelo para generalizar a nuevos datos.

Resultados

El propósito principal de este estudio fue implementar y evaluar un modelo de inteligencia artificial utilizando redes neuronales convolucionales (*CNN*) con la ayuda de *TensorFlow*, enfocado en el reconocimiento facial. Entender cómo funcionan las redes neuronales convolucionales (*CNN)* y que sirva como base para aplicarse a la seguridad en sistemas informáticos.

A continuación, se presentan los resultados de forma estructurada según el orden en que se realizaron los entrenamientos. Primero, se detallan las diferentes configuraciones aplicadas al modelo, destacando los ajustes realizados en los hiperparámetros para cada corrida. Cada configuración se analiza en términos de reporte de clasificación (*Classification Report*), métricas de precisión (*accuracy*) y pérdida (*loss*), acompañados de gráficos como las Curvas de Aprendizaje y las matrices de confusión (*confusion matrices*). Finalmente, se ofrece un resumen comparativo de todas las configuraciones aplicadas,

mostrando cómo los cambios en los hiperparámetros a lo largo de las distintas corridas contribuyeron al desarrollo y refinamiento del modelo final, facilitando la comprensión del proceso de optimización.

1. **Primera corrida**

Tabla 2

Detalles Hiperparámetros del Modelo Primera Corrida

Componente	**Configuración**	**Descripción**
Capas Convolucionales	3 capas con filtros: 32, 64, 64	Extraen características clave de las imágenes.
Regularización	Ninguna aplicada	No se utiliza regularización explícita en el modelo.
Función de Activación	*ReLU* y *Softmax*	Introduce no linealidad y asigna probabilidades de clase.
Optimizador	*Adam*	Optimización adaptativa con ajuste automático del *learning* **rate**.
Pérdida y Métricas	*Sparse Categorical Crossentropy, Accuracy*	Optimiza la precisión en clasificación multiclase.
Aumento de Datos	Rescale (1./255)	Normaliza los píxeles de las imágenes.
Callbacks	Ninguno implementado	No se utilizan *callbacks* en el entrenamiento.
Entrenamiento	20 épocas, batch_size: 4	Configuración para convergencia eficiente y efectiva.

Fuente: Elaboración propia

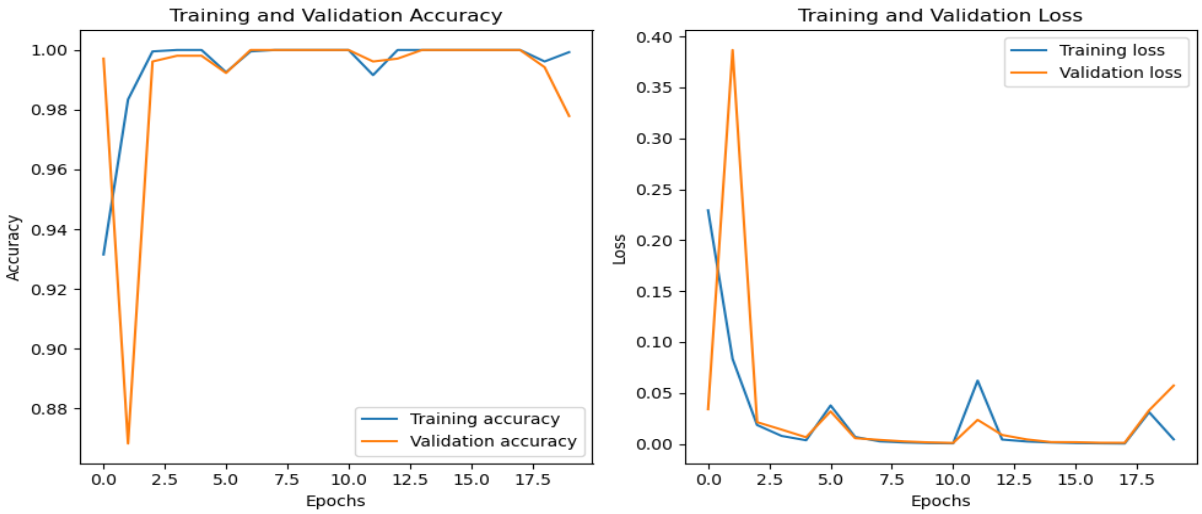
Tabla 3

Reporte de Clasificación (Classification Report) Primera corrida

	precision	recall	f1-score	support
Amparo	1.00	0.98	0.99	651
Eric	1.00	1.00	1.00	650
Luis	0.99	1.00	1.00	650
Mika	1.00	1.00	1.00	651
Miriam	1.00	1.00	1.00	650
Pamela	0.97	1.00	0.99	650
Paul	1.00	1.00	1.00	651
Robert	1.00	0.99	1.00	651
accuracy			1.00	5204
macro avg	1.00	1.00	1.00	5204
weighted avg	1.00	1.00	1.00	5204

Figura 20

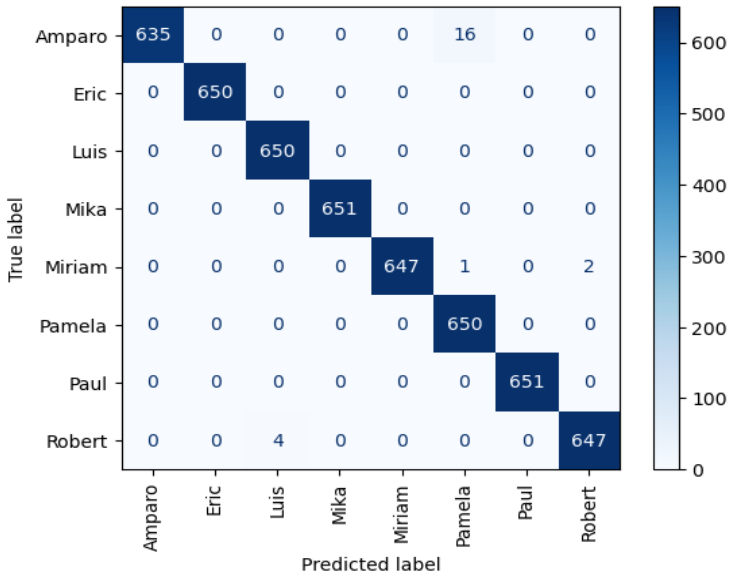
Curvas de Aprendizaje (*Learning Curves*) *Primera Corrida*



Fuente: Elaboración propia

Figura 21

Matriz de Confusión ***(Confusion Matrix)*** *Primera Corrida*



Fuente: Elaboración propia

2. Segunda corrida

Tabla 4

Detalles Hiperparámetros del Modelo Segunda corrida

Componente	Configuración	Descripción

Capas Convolucionales	3 capas con filtros: 32, 64, 64	Extraen características clave de las imágenes.
Regularización	Ninguna aplicada	No se utiliza regularización explícita en el modelo.
Función de Activación	*ReLU* y *Softmax*	*ReLU* introduce no linealidad, y *Softmax* asigna probabilidades a las clases.
Optimizador	*Adam*	Optimización adaptativa con ajuste automático del *learning rate*.
Pérdida y Métricas	*Sparse Categorical Crossentropy*, *Accuracy*	Optimiza la precisión en clasificación multiclase.
Aumento de Datos	*Rescale* (1./255)	Normaliza los píxeles de las imágenes.
Callbacks	Ninguno implementado	Los *callbacks* están configurados, pero no se implementaron en el entrenamiento final.
Entrenamiento	40 épocas, *batch_size*: 16	Configuración para una convergencia eficiente con un tamaño de lote adecuado.

Fuente: Elaboración propia

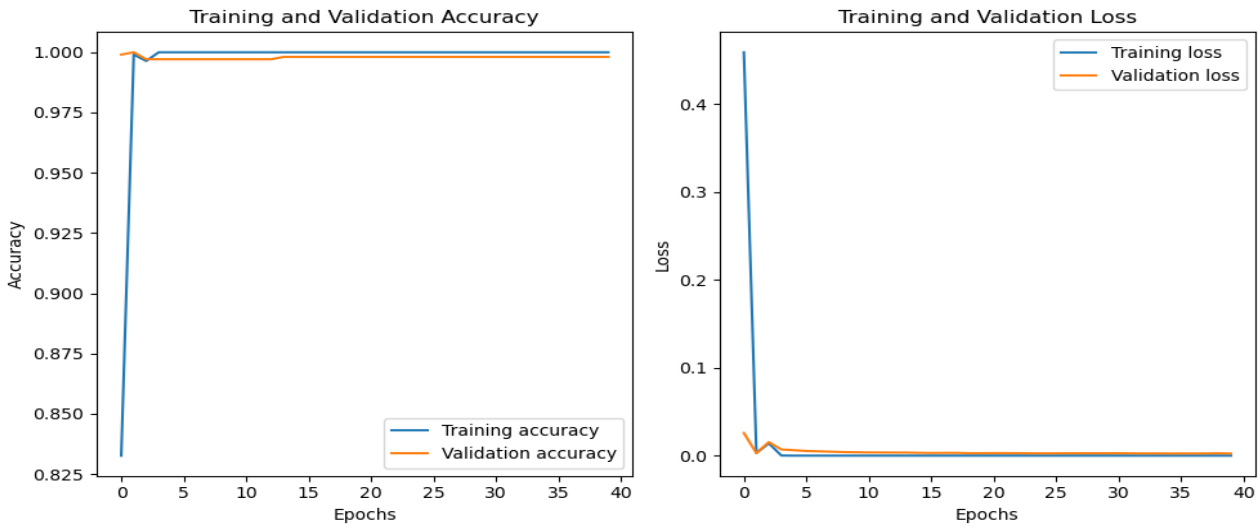
Tabla 5

Reporte de Clasificación (Classification Report) Segunda corrida

	precision	recall	f1-score	support
Amparo	1.00	1.00	1.00	651
Eric	1.00	1.00	1.00	650
Luis	1.00	1.00	1.00	650
Mika	1.00	1.00	1.00	651
Miriam	1.00	1.00	1.00	650
Pamela	1.00	1.00	1.00	650
Paul	1.00	1.00	1.00	651
Robert	1.00	1.00	1.00	651
accuracy			1.00	5204
macro avg	1.00	1.00	1.00	5204
weighted avg	1.00	1.00	1.00	5204

Figura 22

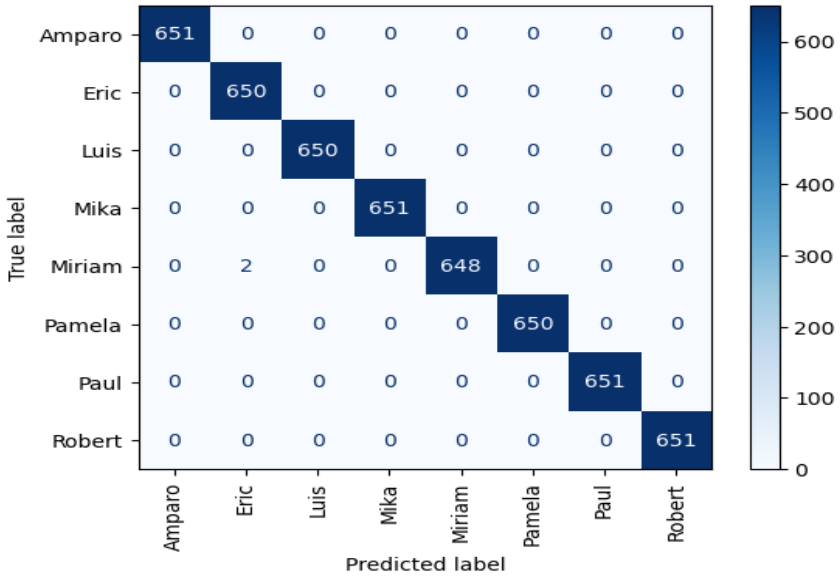
Curvas de Aprendizaje (Learning Curves) Segunda corrida



Fuente: Elaboración propia

Figura 23

Matriz de Confusión Segunda corrida



Fuente: Elaboración propia

3. Tercera corrida

Tabla 6

Detalles Hiperparámetros del Modelo Tercera corrida

Componente	Configuración	Descripción

Capas Convolucionales	3 capas con filtros: 32, 64, 64	Extraen características clave de las imágenes.
Regularización	L2 *Regularization* (0.001)	Regularización aplicada a las capas convolucionales para evitar el sobreajuste.
Función de Activación	*ReLU* y *Softmax*	*ReLU* introduce no linealidad, y *Softmax* asigna probabilidades a las clases.
Optimizador	Adam	Optimización adaptativa con ajuste automático del *learning rate*.
Pérdida y Métricas	*Sparse Categorical Crossentropy, Accuracy*	Optimiza la precisión en clasificación multiclase.
Aumento de Datos	*Rescale* (1./255), Rotación, Desplazamiento, Zoom, Volteo Horizontal	Aumenta la variabilidad de las imágenes para mejorar la robustez del modelo.
Callbacks	Ninguno implementado	Los *callbacks* fueron configurados, pero no se usaron en la ejecución final del modelo.
Entrenamiento	60 épocas, batch_size: 32	Configuración para una convergencia eficiente con un tamaño de lote adecuado.

Fuente: Elaboración propia

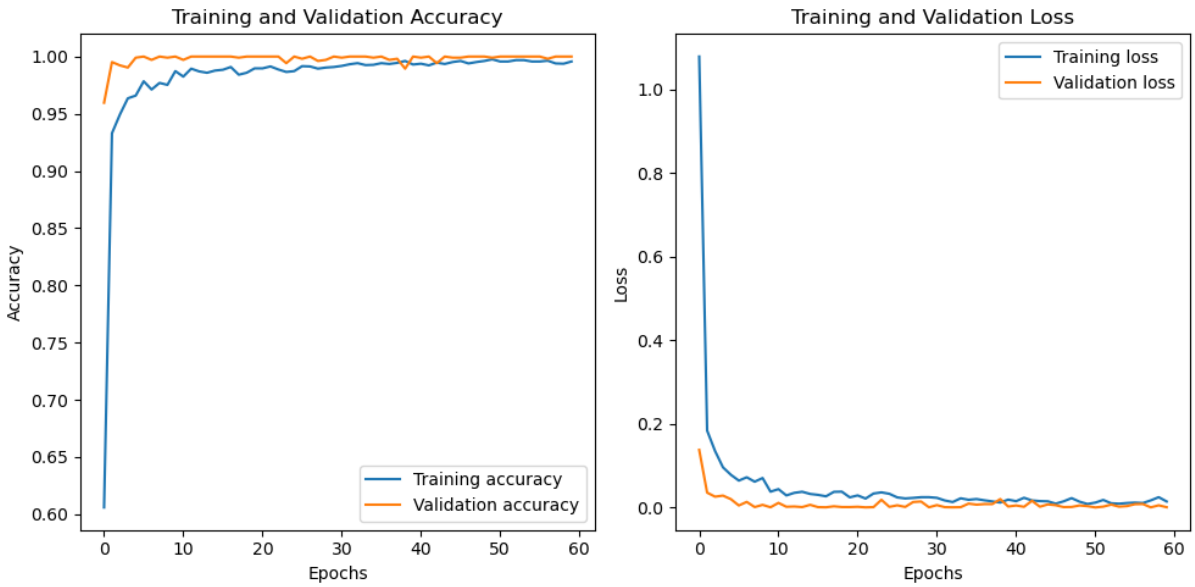
Tabla 7

Reporte de Clasificación (Classification Report) Tercera corrida

	precision	recall	f1-score	support
Amparo	1.00	1.00	1.00	651
Eric	1.00	1.00	1.00	650
Luis	1.00	1.00	1.00	650
Mika	1.00	1.00	1.00	651
Miriam	1.00	1.00	1.00	650
Pamela	1.00	1.00	1.00	650
Paul	1.00	1.00	1.00	651
Robert	1.00	1.00	1.00	651
accuracy			1.00	5204
macro avg	1.00	1.00	1.00	5204
weighted avg	1.00	1.00	1.00	5204

Figura 24

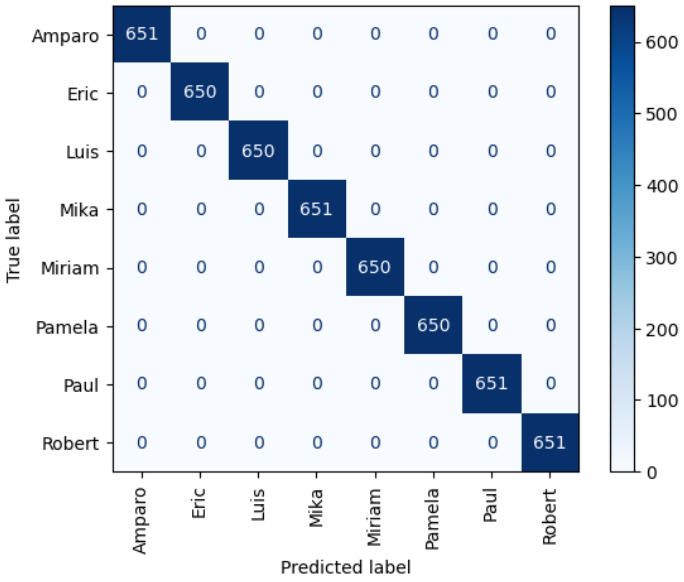
Curvas de Aprendizaje (Learning Curves) Tercera corrida



Fuente: Elaboración propia

Figura 25

Matriz de Confusión Tercera corrida



Fuente: Elaboración propia

4. Cuarta corrida

Tabla 8

Detlles Hiperparámetros del Modelo Cuarta corrida

Componente	Configuración	Descripción
Capas Convolucionales	3 capas con filtros: 32, 64, 128	Capturan características clave de las imágenes.
Regularización	L2 (0.001) y *Dropout* (0.3-0.5)	Prevención del sobreajuste para una mayor generalización.
Función de Activación	*ReLU* y *Softmax*	Introduce no linealidad y asigna probabilidades de clase.
Optimizador	*RMSprop* (LR: 0.0001)	Ajusta dinámicamente la tasa de aprendizaje.
Pérdida y Métricas	*Sparse Categorical Crossentropy*, *Accuracy*	Optimiza la precisión de clasificación multiclase.
Aumento de Datos	Rotación, Zoom, Desplazamiento, Volteo	Mejora la robustez al entrenar con variaciones de imágenes.
Callbacks	*Early Stopping*, *ReduceLROnPlateau*	Controla el sobreentrenamiento y ajusta la tasa de aprendizaje.
Entrenamiento	100 épocas, *batch_size*: 32	Configuración para convergencia óptima sin sobreentrenamiento.

Fuente: Elaboración propia

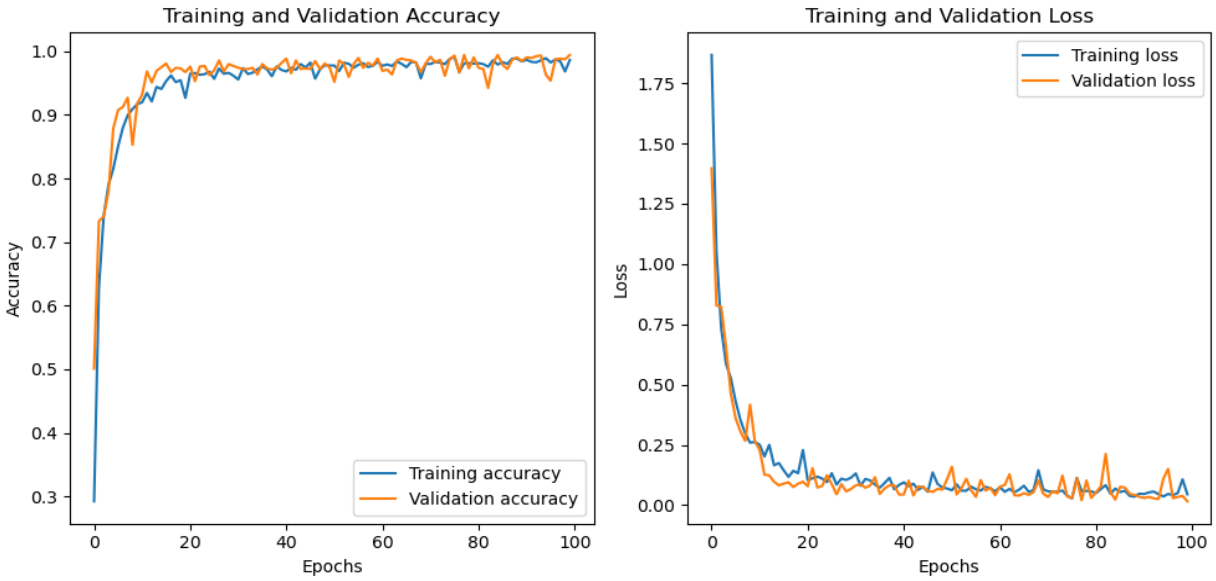
Tabla 9

Reporte de Clasificación (Classification Report) Cuarta corrida

	precision	recall	f1-score	support
Amparo	1.00	1.00	1.00	651
Eric	1.00	1.00	1.00	650
Luis	1.00	1.00	1.00	650
Mika	1.00	1.00	1.00	651
Miriam	1.00	1.00	1.00	650
Pamela	1.00	1.00	1.00	650
Paul	1.00	1.00	1.00	651
Robert	1.00	1.00	1.00	651
accuracy			1.00	5204
macro avg	1.00	1.00	1.00	5204
weighted avg	1.00	1.00	1.00	5204

Figura 26

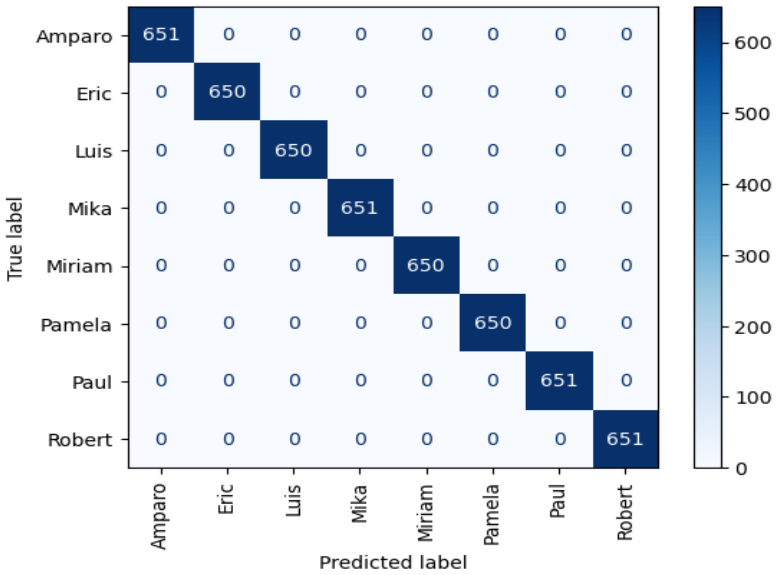
Curvas de Aprendizaje (Learning Curves) Cuarta corrida



Fuente: Elaboración propia

Figura 27

Matriz de Confusión Cuarta corrida



Fuente: Elaboración propia

5. Resultado corrida Modelo final

Tabla 10

Detalles Hiperparámetros del Modelo Final

Componente	Configuración	Descripción
Capas Convolucionales	3 capas con filtros: 32, 64, 128	Extraen características clave de las imágenes.
Regularización	L2 *Regularization* (0.001)	Regularización aplicada a las capas convolucionales y densas para evitar el sobreajuste.
Función de Activación	*ReLU* y *Softmax*	*ReLU* introduce no linealidad, y *Softmax* asigna probabilidades a las clases.
Optimizador	*RMSprop* (*learning_rate*=0.0001)	Optimización adaptativa con un learning rate ajustado.
Pérdida y Métricas	*Sparse Categorical Crossentropy*, *Accuracy*	Optimiza la precisión en clasificación multiclase.
Aumento de Datos	*Rescale* (1./255), Rotación, Desplazamiento, Zoom, Volteo Horizontal	Aumenta la variabilidad de las imágenes para mejorar la robustez del modelo.
Callbacks	*Early Stopping*, *ReduceLROnPlateau*	Se implementaron para evitar el sobreajuste y ajustar dinámicamente el *learning rate*.
Entrenamiento	120 épocas, *batch_size*: 32	Configuración para una convergencia eficiente con un tamaño de lote adecuado.

Fuente: Elaboración propia

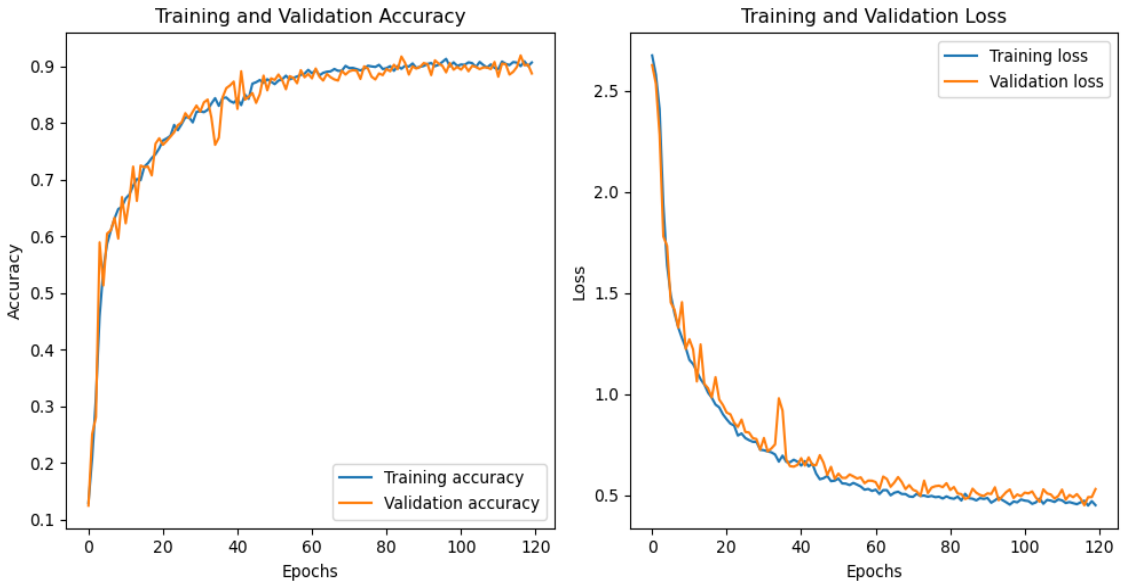
Tabla 11

Reporte de Clasificación (Classification Report) Modelo final

	precision	recall	f1-score	support
Amparo	0.99	0.94	0.97	651
Eric	1.00	1.00	1.00	650
Luis	1.00	0.99	1.00	650
Mika	0.94	0.94	0.94	651
Miriam	0.99	1.00	1.00	650
Pamela	1.00	1.00	1.00	650
Paul	1.00	1.00	1.00	651
Robert	0.93	0.98	0.95	651
accuracy			0.98	5204
macro avg	0.98	0.98	0.98	5204
weighted avg	0.98	0.98	0.98	5204

Figura 28

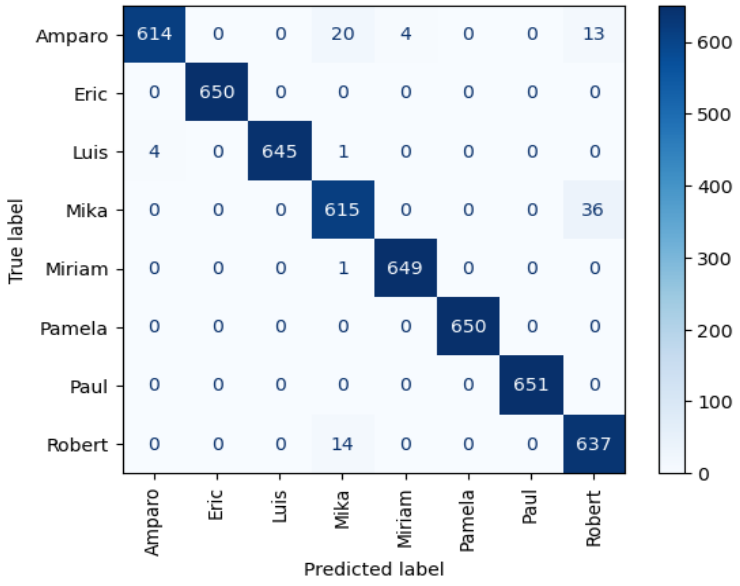
Curvas de Aprendizaje (Learning Curves) Modelo Final



Fuente: Elaboración propia

Figura 29

Matriz de Confusión Modelo Final



Fuente: Elaboración propia

Comparación de Configuraciones Aplicadas al Modelo

Tabla 12

Comparación de Configuraciones Aplicadas al Modelo en cada corrida

Componente	Config. 1	Config. 2	Config. 3	Config. 4	Modelo Final Config.
Capas Convolucionales	3 capas con filtros: 32, 64, 64	3 capas con filtros: 32, 64, 64	3 capas con filtros: 32, 64, 64	3 capas con filtros: 32, 64, 128	3 capas con filtros: 32, 64, 128
Regularización	Ninguna aplicada	Ninguna aplicada	L2 Regularization (0.001)	L2 (0.001) y Dropout (0.3-0.5)	L2 Regularization (0.001)
Función de Activación	ReLU y Softmax	ReLU y Softmax	ReLU y Softmax	ReLU y Softmax	ReLU y Softmax
Optimizador	Adam	Adam	Adam	RMSprop (LR: 0.0001)	RMSprop (learning_rate =0.0001)
Pérdida y Métricas	Sparse Categorical Crossentropy, Accuracy	Sparse Categorical Crossentropy, Accuracy	Sparse Categorical Crossentropy, Accuracy	Sparse Categorical Crossentropy, Accuracy	Sparse Categorical Crossentropy, Accuracy
Aumento de Datos	Rescale (1./255)	Rescale (1./255)	Rescale (1./255), Rotación, Desplazamiento, Zoom, Volteo Horizontal	Rotación, Zoom, Desplazamiento, Volteo	Rescale (1./255), Rotación, Desplazamiento, Zoom, Volteo Horizontal
Callbacks	Ninguno implementado	Ninguno implementado	Ninguno implementado	Early Stopping, ReduceLROnPlateau	Early Stopping, ReduceLROnPlateau
Entrenamiento	20 épocas, batch_size: 4	40 épocas, batch_size: 16	60 épocas, batch_size: 32	100 épocas, batch_size: 32	120 épocas, batch_size: 32
Resultados de Métricas	Accuracy: 100%, F1-Score: 1.00	Accuracy: 100%, F1-Score: 1.00	Accuracy: 100%, F1-Score: 1.00	Accuracy: 100%, F1-Score: 1.00	**Accuracy: 98%, F1-Score: 0.98**

Fuente: Elaboración propia

En síntesis, los resultados obtenidos a partir de las diferentes configuraciones del modelo revelan hallazgos importantes sobre la optimización del rendimiento. Las configuraciones iniciales (1 a 3), que no aplicaron regularización significativa y mantuvieron configuraciones más simples, lograron una precisión y F1-Score perfectos del 100%. Sin embargo, al avanzar hacia configuraciones más complejas, como en la Configuración 4 y el Modelo Final, se implementaron estrategias adicionales, incluyendo la regularización

L2, Dropout, y un cambio al optimizador RMSprop, acompañado de técnicas de aumento de datos más rigurosas. Aunque esto resultó en una ligera disminución de la precisión a un 98% en el Modelo Final, estos ajustes son cruciales para mejorar la robustez y la capacidad del modelo para generalizar, subrayando la importancia de equilibrar la complejidad y la eficacia para desarrollar modelos confiables en aplicaciones prácticas.

Discusión y conclusiones

Discusión de Resultados

Los resultados obtenidos en este estudio están alineados con la pregunta de investigación: **"¿Cómo puede la inteligencia artificial basada en técnicas de reconocimiento facial ayudar a fortalecer la seguridad y eficiencia como un segundo factor de autenticación en sistemas informáticos?"** Los resultados demuestran que las redes neuronales convolucionales (CNN) pueden contribuir significativamente a mejorar la seguridad y eficiencia en sistemas de autenticación. A lo largo de las diferentes corridas, incluyendo el modelo final, se observó una evolución notable en la capacidad del modelo, desde configuraciones iniciales hasta una versión optimizada.

- **Primera Corrida**: Los resultados de la **Tabla 2** ("Detalles Hiperparámetros del Modelo Primera Corrida") y el **Reporte de Clasificación** en la **Tabla 3** ("Reporte de Clasificación Primera Corrida") muestran una precisión (precision) casi perfecta, lo que sugiere que el modelo capturó eficazmente las características del conjunto de datos de entrenamiento. Sin embargo, las **Curvas de Aprendizaje** en la **Figura 21** indican un posible sobreajuste, dado que no se implementaron técnicas de regularización. Esto limita la capacidad del modelo para generalizar y aplicarse a datos nuevos, lo cual es esencial para la autenticación en sistemas reales.
- **Segunda Corrida**: En esta fase, descrita en la **Tabla 4** ("Detalles Hiperparámetros del Modelo Segunda Corrida"), se mantuvieron los mismos parámetros, pero se incrementaron las épocas y el tamaño del lote. Los resultados reflejados en la **Tabla 5** ("Reporte de Clasificación Segunda Corrida") y la **Figura**

23 muestran que el modelo sigue manteniendo altos niveles de precisión (*precision*). Sin embargo, la persistencia de una precisión del 100% sugiere que el modelo aún podría estar demasiado adaptado a los datos de entrenamiento. Las **Curvas de Aprendizaje** revelan un proceso de aprendizaje más exhaustivo, pero también refuerzan la idea de un posible sobreajuste, lo que podría afectar la eficiencia del modelo en un entorno de autenticación más variado.

- **Tercera Corrida**: Con la introducción de la regularización L2 y técnicas de aumento de datos, como se detalla en la **Tabla 6** ("Detalles Hiperparámetros del Modelo Tercera Corrida"), el modelo mostró mejoras en la capacidad de generalización. Las **Curvas de Aprendizaje** en la **Figura 25** evidencian una separación más clara entre las curvas de entrenamiento y validación, lo que indica que el modelo está aprendiendo de manera más equilibrada y reduciendo el riesgo de sobreajuste. Esto es crucial para asegurar que el modelo pueda manejar con éxito nuevas instancias de datos, cumpliendo con el objetivo de fortalecer la seguridad y eficiencia en la autenticación.
- **Cuarta Corrida**: En la cuarta corrida, se implementaron técnicas adicionales de regularización y se ajustó el optimizador a RMSprop, como se muestra en la **Tabla 8** ("Detalles Hiperparámetros del Modelo Cuarta Corrida"). Las **Curvas de Aprendizaje** en la **Figura 27** sugieren un proceso de aprendizaje más controlado, con una curva de validación que no coincide completamente con la de entrenamiento, lo cual es positivo para evitar el sobreajuste. Este ajuste resultó en un modelo más robusto y adaptable, capaz de mantener altos niveles de precisión (*precision*) mientras mejora su capacidad de generalización, cumpliendo con los objetivos específicos del estudio.
- **Modelo Final**: El modelo final, detallado en la **Tabla 10** ("Detalles Hiperparámetros del Modelo Final") y evaluado en la **Figura 29** (Curvas de Aprendizaje Modelo Final) y la **Figura 30** (Matriz de Confusión Modelo Final), representa la culminación de todos los ajustes y mejoras realizadas a lo largo del estudio. A pesar de las ligeras disminuciones en la precisión (*precision*) para algunos sujetos, el modelo

final muestra una capacidad significativamente mejorada para generalizar y manejar variaciones en los datos, lo que es fundamental para su implementación en escenarios de autenticación reales.

- **Interpretación de la Matriz de Confusión**: La **Figura 30** (Matriz de Confusión del Modelo Final) revela información clave sobre el rendimiento del modelo en cada una de las clases de reconocimiento facial. Aunque en general se mantiene un alto nivel de precisión (*precision*) y *recall*, se observan pequeñas discrepancias en la clasificación de algunos sujetos, como "Amparo" y "Robert", que presentan valores de *recall* menores en comparación con otros sujetos. Este comportamiento sugiere que, aunque el modelo es robusto en general, hay ciertos casos donde la variabilidad en las imágenes o características específicas de estos individuos hacen que el modelo tenga dificultades para clasificarlos correctamente en todos los casos. Estas observaciones son cruciales, ya que destacan la necesidad de continuar optimizando el modelo o de considerar técnicas adicionales de preprocesamiento de datos o aumento de datos para mitigar estos efectos.
- **Análisis de las Métricas**: El **Reporte de Clasificación** en la **Tabla 11** ("Reporte de Clasificación Modelo Final") proporciona un desglose detallado de las métricas de precisión (*precision*), recall, f1-score, specificity y accuracy para cada clase:
    - **Precisión (*Precision*)**: La precisión del modelo se mantuvo alta, con un promedio general de 0.98, lo que indica que la mayoría de las predicciones positivas fueron correctas. Esto es crucial en un sistema de autenticación, ya que una alta precisión reduce la posibilidad de falsos positivos, minimizando el riesgo de permitir acceso a usuarios no autorizados.
    - **Recall**: El recall también fue alto, con un promedio de 0.98, lo que significa que el modelo identificó correctamente la mayoría de las instancias positivas reales. Sin embargo, en clases como "Amparo" y "Robert", el *recall* fue ligeramente menor, lo que sugiere que el modelo podría haber omitido algunas instancias correctas, lo que podría traducirse en falsos negativos en un contexto de autenticación.

- **F1-Score**: El *f1-score*, que es la media armónica entre precisión (*precision*) y *recall*, reflejó un buen equilibrio entre estas dos métricas, con un promedio de 0.98. Esto indica que el modelo mantiene un buen rendimiento tanto en términos de detectar correctamente instancias positivas como en evitar predicciones incorrectas.
- ***Specificity***: La *specificity*, que mide la proporción de verdaderos negativos correctamente identificados, fue también alta, lo que sugiere que el modelo es eficaz no solo en reconocer correctamente a los individuos autorizados, sino también en evitar falsas alarmas con individuos no autorizados.
- **Exactitud (*Accuracy*)**: La exactitud general del modelo fue del 98%, lo que confirma que la mayoría de las predicciones, tanto positivas como negativas, fueron correctas. Aunque la exactitud es una métrica importante, es esencial complementarla con *precision*, *recall*, *specificity* y *f1-score* para obtener una visión completa del rendimiento del modelo, especialmente en un sistema de autenticación donde los errores pueden tener consecuencias significativas.

Las curvas de aprendizaje del modelo final confirman un proceso de aprendizaje equilibrado y controlado, minimizando el riesgo de sobreajuste y maximizando la adaptabilidad del modelo.

Interpretación y Vinculación con el Marco Teórico

Cada fase de experimentación, incluyendo el análisis del modelo final, fue clave para explorar y validar la teoría detrás del uso de redes neuronales convolucionales (*CNNs*) en el reconocimiento facial. El progreso desde una precisión inicial perfecta hasta un modelo más complejo y probado en diversas situaciones refleja la importancia de equilibrar la capacidad de aprendizaje con la habilidad de generalizar a partir de datos no vistos. Este equilibrio es esencial para que las redes neuronales convolucionales (*CNNs*) sean efectivas en la práctica, especialmente como un segundo factor de autenticación en sistemas informáticos, tal como se plantea en el marco teórico del estudio. Las **Curvas de Aprendizaje**

proporcionaron información valiosa sobre el comportamiento del modelo y cómo los ajustes en los hiperparámetros afectaron su capacidad para generalizar, alineándose con los principios teóricos de aprendizaje profundo y visión por computadora.

Resultados Inesperados y Consecuencias

Los resultados inesperados de precisión perfecta en las primeras corridas levantaron preocupaciones sobre el sobreajuste, lo que llevó a realizar ajustes en las siguientes corridas, como se evidenció en la Cuarta Corrida y el Modelo Final. La introducción de más capas y técnicas de regularización, junto con ligeras disminuciones en algunas métricas de precisión (*precision*), reflejan un avance hacia un modelo que puede funcionar efectivamente en una variedad más amplia de situaciones reales. Las **Curvas de Aprendizaje** en las últimas corridas y en el modelo final demostraron una mejora en la capacidad del modelo para evitar el sobreajuste, resultando en un modelo más equilibrado y robusto, capaz de cumplir con los objetivos específicos del estudio de manera efectiva.

Limitaciones del Estudio

Una de las principales limitaciones del estudio fue la homogeneidad del conjunto de datos utilizado. A pesar de las mejoras en la configuración del modelo, es necesario probarlo en un conjunto de datos más representativo y diverso para evaluar completamente su capacidad de generalización y eficiencia en un entorno de autenticación real. Además, aunque las **Curvas de Aprendizaje** proporcionaron información valiosa, la falta de diversidad en los datos podría haber ocultado posibles problemas de generalización que podrían surgir en aplicaciones del mundo real, limitando así la aplicabilidad práctica del modelo.

Conclusión y Cumplimiento de los Objetivos

Este estudio ha logrado cumplir con el objetivo general de implementar un modelo de inteligencia artificial basado en redes neuronales convolucionales (*CNN*), utilizando la librería *TensorFlow* para técnicas de reconocimiento facial. A continuación, se detalla cómo se cumplieron los objetivos específicos:

1. **Implementación de un modelo de redes neuronales convolucionales para el reconocimiento facial**

Se implementó exitosamente un modelo redes neuronales convolucionales (*CNN*) con múltiples capas convolucionales, *max-pooling*, y capas densas, ajustadas para extraer características clave de las imágenes faciales, cumpliendo con el primer objetivo específico.

2. **Entrenamiento del modelo utilizando un conjunto de datos preprocesado y estándar**

El modelo fue entrenado con un conjunto de datos preprocesado y estándar, aplicando técnicas de normalización y aumento de datos, lo que permitió que el modelo se entrenara de manera efectiva para reconocer patrones faciales en diversas condiciones, cumpliendo el segundo objetivo específico.

3. **Evaluación de la precisión y eficiencia del modelo mediante pruebas con un conjunto de datos de validación**

El tercer objetivo específico se cumplió mediante la evaluación detallada del modelo en términos de precisión (*precision*), *recall*, *f1-score*, *specificity* y exactitud (*accuracy*). Las pruebas con el conjunto de datos de validación demostraron que el modelo final alcanzó un rendimiento aceptable en la clasificación de imágenes faciales, con pequeños ajustes necesarios en el *recall* para ciertas clases.

4. **Optimización del modelo ajustando hiperparámetros clave como la tasa de aprendizaje, el número de capas y la cantidad de neuronas en cada capa**

A través de un proceso iterativo de ajuste de hiperparámetros, como la tasa de aprendizaje, el número de capas y la cantidad de neuronas, se logró optimizar el modelo, mejorando su capacidad de generalización y reduciendo el sobreajuste, tal como se planteó en el cuarto objetivo específico.

Futuros estudios deberían centrarse en aumentar la diversidad del conjunto de datos y probar el modelo en escenarios de autenticación en tiempo real para evaluar su efectividad en entornos de seguridad informática dinámicos. Esto no solo validará la robustez del modelo en condiciones más diversas, sino que también demostrará su utilidad práctica en la protección de sistemas informáticos.

Referencias

365 Careers. (2020). *Entrenamiento completo en Data Science con Python*. Udemy. https://www.udemy.com/course/programa-completo-de-data-science-todo-para-los-datos/

Abalde, N. (2023, diciembre 1). Descubriendo las Ramas de la Inteligencia Artificial. *Smartmind*. https://www.smartmind.net/blog/ramas-ia/

Abril, R. R. (2021, marzo 21). Redes Convolucionales • Un artículo de La Máquina Oráculo. *La Máquina Oráculo*. https://lamaquinaoraculo.com/deep-learning/redes-neuronales-convolucionales/

Ahmed, S. (2019). *Reconocimiento de caracteres en imágenes mediante el uso de CNNs*. ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

Arce, J. I. B. (2019, julio 26). La matriz de confusión y sus métricas – Inteligencia Artificial –. *Juan Barrios*. https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

Bagnato, J. (2017). *Qué es overfitting y underfitting y cómo solucionarlo | Aprende Machine Learning*. https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/

Bagnato, J. (2018). *Convolutional Neural Networks: La Teoría explicada en Español | Aprende Machine Learning*. https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

Bosch, A., Casas, J., & Lozano, T. (2019). *Deep learning: Principios y fundamentos*. Editorial UOC. https://elibro.puce.elogim.com/es/ereader/puce/126167

Datacamp. (2023). *Python Convolutional Neural Networks (CNN) with TensorFlow Tutorial | DataCamp*. https://www.datacamp.com/tutorial/cnn-tensorflow-python

Espinosa, C. (2018). *Clasificacion de especies de flores usando ´ tecnicas de deep learning*. https://core.ac.uk/download/pdf/189878177.pdf

Freeze, D. (2022, octubre 13). Cybercrime To Cost The World 8 Trillion Annually In 2023. *Cybercrime Magazine*. https://cybersecurityventures.com/cybercrime-to-cost-the-world-8-trillion-annually-in-2023/

Galárraga Cañizares, J. L. (2017). *Clasificador de hojas mediante Deep Learning* [Masters, E.T.S. de Ingenieros Informáticos (UPM)]. https://oa.upm.es/47784/

GeeksforGeeks. (2017, noviembre 23). *ML | Underfitting and Overfitting*. GeeksforGeeks. https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

IBM. (2024). *¿Qué son las redes neuronales convolucionales? | IBM*. https://www.ibm.com/es-es/topics/convolutional-neural-networks

Lara, R. (2020). *FUNDAMENTOS DE REDES NEURONALES ARTIFICIALES*. https://conceptos.sociales.unam.mx/conceptos_final/598trabajo.pdf

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444. https://doi.org/10.1038/nature14539

López, L. (2024). *Introducción a TensorFlow y Keras: Fundamentos y ejemplos | OpenWebinars*. OpenWebinars.net. https://openwebinars.net/blog/tensorflow-keras-fundamentos/

Martínez, J. (2019, diciembre 17). ¿Qué es Overfitting y Cómo lo Detectamos? *DataSmarts Español*. https://datasmarts.net/es/que-es-overfitting-y-como-lo-detectamos/

Méndez, R. (2021, octubre 30). Álgebra lineal (escalares, vectores, matrices y tensores) en python. *Medium*. https://medium.com/@rubenmndez/%C3%A1lgebra-lineal-escalares-vectores-matrices-y-tensores-en-python-be7353da782d

Montes, M., & Israel, D. (2023). *Análisis de los ataques cibernéticos en la banca ecuatoriana: Mapeo sistemático*.

Ortego, D. (2017). *¿Qué es Tensorflow? | OpenWebinars*. OpenWebinars.net. https://openwebinars.net/blog/que-es-tensorflow/

Pinar, I. (2024). *Máster Deep Learning—Inteligencia Artificial con Python*. Udemy. https://www.udemy.com/course/master-redes-neuronales-deep-learning-con-tensorflow-2021/

Romero, J. (2020, abril 27). *Metodologías de Minería de Datos – Jorge Romero*. https://jorgeromero.net/metodologias-de-mineria-de-datos

TensorFlow. (2022). *Guia inicial de TensorFlow 2.0 para principiantes | TensorFlow Core*. https://www.tensorflow.org/tutorials/quickstart/beginner?hl=es-419

TensorFlow Core. (2024). *Clasificación de imágenes | TensorFlow Core*. TensorFlow. https://www.tensorflow.org/tutorials/images/classification?hl=es-419

Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM Comput. Surv.*, *35*(4), 399-458. https://doi.org/10.1145/954339.954342