Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Industrial

PROTOTIPO DE UN SISTEMA DE AUTOMATIZACIÓN Y CONTROL DE PROCESO INDUSTRIAL CON ALGORITMO DE REDES NEURONALES PARA LÍNEAS DE CONTROL DE CALIDAD

Wilson Arnoldo Velásquez López
Asesorado por el Ing. Carlos Alberto Fernando Navarro Fuentes

Guatemala, agosto de 2023

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

PROTOTIPO DE UN SISTEMA DE AUTOMATIZACIÓN Y CONTROL DE PROCESO INDUSTRIAL CON ALGORITMO DE REDES NEURONALES PARA LÍNEAS DE CONTROL DE CALIDAD

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

WILSON ARNOLDO VELÁSQUEZ LÓPEZ
ASESORADO POR EL ING. CARLOS ALBERTO FERNANDO NAVARRO FUENTES

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO MECÁNICO ELECTRICISTA

GUATEMALA, AGOSTO DE 2023

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. José Francisco Gómez Rivera (a.i.)
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Ing. Kevin Armando Cruz Armando Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANA	Inga. Aurelia Anabela Cordova Estrada
EXAMINADOR	Ing. Endor Steve Ortiz del Cid
EXAMINADOR	Ing. Edgar Yanuario Laj
EXAMINADOR	Ing. Herbert Samuel Figueroa Avendaño
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la Ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

PROTOTIPO DE UN SISTEMA DE AUTOMATIZACIÓN Y CONTROL DE PROCESO INDUSTRIAL CON ALGORITMO DE REDES NEURONALES PARA LÍNEAS DE CONTROL DE CALIDAD

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 9 de febrero de 2022.

Wilson Arnoldo Velásquez López

Guatemala 28 de febrero de 2023

Ingeniero
José Anibal Silva de los Angeles
Coordinador del Área de Electrotécnica
Escuela ingeniería Mecánica Eléctrica
Faculta de Ingeniería
Universidad de San Carlos de Guatemala

Ingeniero Silva:

Por este medio hago de su conocimiento que he concluido la revisión del trabajo de graduación del estudiante **Wilson Arnoldo Velásquez López**, con número de registro académico 201700716, y con numero CUI 2902455670101, titulado: **PROTOTIPO DE UN SISTEMA DE AUTOMATIZACIÓN Y CONTROL DE PROCESO INDUSTRIAL CON ALGORITMO DE REDES NEURONALES PARA LÍNEAS DE CONTROL DE CALIDAD**. En virtud de lo anterior y para los efectos consiguientes, someto a su consideración el presente como mi aprobación, indicando que tanto el suscrito como el estudiante Velásquez López somos responsables por el contenido del trabajo referido.

Cordialmente,

Carlos Alberto Fernando Navarro Fuentes
Ingeniero Electricista
Colegiado 8339
Asesor

ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

REF. EIME 04072023
04 de Julio de 2023

Señor Director
Ing. Armando Alonso Rivera Carrillo
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Señor Director:

Me permito dar aprobación al trabajo de Graduación titulado: **"PROTOTIPO DE UN SISTEMA DE AUTOMATIZACIÓN Y CONTROL DE PROCESO INDUSTRIAL CON ALGORITMO DE REDES NEURONALES PARA LÍNEAS DE CONTROL DE CALIDAD"**, del estudiante; Wilson Arnoldo Velásquez López, con numero de carnet 2902455670101 y registro académico 201700716 , que cumple con los requisitos establecidos para tal fin.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente,
ID Y ENSEÑAD A TODOS

JOSE ANIBAL SILVA DE LOS ANGELES
ING ELECTRONICO
COLEGIADO No 5067

Ing. José Anibal Silva de los Angeles
CC. Básicas Y Electrotecnia

ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

REF. EIME 30.2023.

El director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de área, al trabajo de Graduación del estudiante Wilson Arnoldo Velásquez López: **PROTOTIPO DE UN SISTEMA DE AUTOMATIZACIÓN Y CONTROL DE PROCESO INDUSTRIAL CON ALGORITMO DE REDES NEURONALES PARA LÍNEAS DE CONTROL DE CALIDAD**, procede a la autorización correspondiente.

Ing. Armando Alonso Rivera Carrillo

Guatemala, 5 de julio de 2023

Decanato
Facultad de Ingeniería
24189101- 24189102
secretariadecanato@ingenieria.usac.edu.gt

LNG.DECANATO.OI.608.2023

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **PROTOTIPO DE UN SISTEMA DE AUTOMATIZACIÓN Y CONTROL DE PROCESO INDUSTRIAL CON ALGORITMO DE REDES NEURONALES PARA LÍNEAS DE CONTROL DE CALIDAD,** presentado por: **Wilson Arnoldo Velásquez López,** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO a.i.
Facultad de Ingeniería

Ing. José Francisco Gómez Rivera

Decano a.i.

Guatemala, agosto de 2023

AACE/gaoc

ACTO QUE DEDICO A:

Dios	Por su amor y su guía en todo momento que mi esfuerzo sea una ofrenda de amor.
Mis padres	Por ser quienes me enseñaron a ser fuerte, a ser valiente y a soñar en grande, este triunfo es gracias a ustedes
Mis abuelos	Por ser mi historia, mi raíz, mi hogar quienes me han dado una herencia de amor, sabiduría, junto con la tradición que llevare conmigo toda mi vida.
Mis hermanos	Por estar siempre conmigo en los momentos buenos y malos.
Mis tíos	Por siempre creer en mí y alentar mi camino.
Los perros	Nuestros fieles compañeros, cuyas lecciones de lealtad, amor incondicional y alegría han dejado huella en nuestras vidas.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por haber sido una influencia importante en mi carrera y por haberme dado la bienvenida al mundo profesional. Las oportunidades que me ha brindado son incomparables, antes de esto, ni siquiera pensaba que serían posibles.
Facultad de Ingeniería	Por haber sido mi hogar durante estos años de estudio. Adquiriendo las herramientas, habilidades necesarias para desempeñarme en mí campo profesional gracias al compromiso y dedicación de un cuerpo docente excepcional.
Mis amigos de la Facultad	Brandon Pineda, Kevin Zacarías, Mario Orellana, Nelson Palencia y José Cecilio, Este ha sido un viaje emocionante, a menudo, desafiante, pero su compañía, apoyo han hecho que sea mucho más llevadero.
Danilo Rivera	Por su valiosa ayuda desinteresada.
Ing. Carlos Navarro	Por su asesoría valioso y orientación. Gracias a su conocimiento, experiencia y dedicación, pude llevar a cabo el presente trabajo de graduación de manera efectiva.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Actuador Dispositivo que activa el sistema de control de un proceso.

Amperio Unidad de medida de la intensidad de corriente eléctrica.

Aprendizaje automático El estudio de algoritmos que pueden aprender a partir de datos.

Aprendizaje profundo Una rama del aprendizaje automático que utiliza redes neuronales profundas para aprender a partir de datos.

Automatización La tecnología que se utiliza para realizar tareas sin intervención humana.

Clasificación Una tarea de aprendizaje automático que involucra la predicción de una etiqueta categórica, como la identificación de un objeto en una imagen.

Conjunto de datos Una colección de ejemplos utilizados para entrenar o evaluar un modelo de aprendizaje automático.

Convolución Una operación matemática utilizada en las redes neuronales convolucionales para extraer características de las imágenes.

Dirección IP	Siglas en inglés para Internet Protocolo, conjunto de números que identifica, de manera lógica y jerárquica, a una interfaz en red.
Entrenamiento	El proceso de ajustar un modelo a un conjunto de datos de entrenamiento.
Fabricación inteligente	Una fábrica que utiliza tecnologías de la industria 4.0 para mejorar la eficiencia, la calidad y la flexibilidad de los procesos de fabricación.
Función de activación	Una función utilizada para introducir no linealidad en una red neuronal, lo que permite que el modelo aprenda relaciones más complejas entre los datos de entrada y salida.
Inteligencia artificial	La simulación de la inteligencia humana en máquinas que se pueden programar para realizar tareas que normalmente requieren inteligencia humana.
Internet de las cosas (IoT)	Un sistema de dispositivos interconectados que pueden comunicarse entre sí y con otros sistemas a través de internet.
Machine Learning	El estudio de algoritmos que pueden aprender a partir de datos sin ser programados explícitamente.
Mapear	Proceso de extraer campos de datos de uno o varios archivos de origen y relacionarlos con sus campos de

	destino relacionados en el destino.
Parámetro	Conjunto de condiciones que modifican una variable de salida.
Píxel	Un píxel es la unidad más pequeña de una imagen digital, que se puede iluminar o apagar para crear una imagen en una pantalla.
Precisión	La proporción de ejemplos clasificados correctamente por un modelo.
Red neuronal	Un modelo de aprendizaje automático inspirado en la estructura del cerebro, que consiste en capas de neuronas artificiales.
Red neuronal convolucional (CNN)	Un tipo de red neuronal que utiliza convoluciones para procesar datos de entrada, típicamente imágenes.
Servicios en la nube	Un modelo de acceso a recursos informáticos que se entregan a través de internet.
Sistemas ciberfísicos	Sistemas que combinan la tecnología de la información y las comunicaciones con procesos físicos para controlar y supervisar sistemas en tiempo real.
Validación	El proceso de evaluar el rendimiento de un modelo en datos de prueba no vistos.

Validación cruzada	Un método para evaluar un modelo de aprendizaje automático utilizando múltiples subconjuntos de datos de entrenamiento y prueba.
Volt	Unidades para voltaje.

RESUMEN

El presente trabajo de grado se enfoca en la automatización y control de un proceso industrial de una línea de producción para el control de calidad de limones, utilizando técnicas de Visión Artificial y Aprendizaje Automático. Se emplean redes neuronales convolucionales (ConvNet) para la clasificación de limones según los estándares establecidos.

Los resultados obtenidos demostraron que las redes neuronales convolucionales (ConvNet) son una herramienta efectiva para los procesos industriales de control de calidad, clasificando los limones en dos categorías establecidos: excelente calidad y mala calidad, con una precisión del 98 %.

Por otra parte, se investiga el impacto y la importancia de la tecnología de Visión Artificial en la evolución hacia la Industria 4.0, se analiza cómo la Visión Artificial está revolucionando la industria además de cómo se utiliza para resolver problemas en diferentes sectores industriales. Se examinan los avances tecnológicos en Visión Artificial, la influencia en la automatización como así también la eficiencia de la industria. Se discute cómo la Visión Artificial está mejorando la calidad del control que a su vez se ve reflejado la toma de decisiones en tiempo real en la industria.

Se explora las barreras y desafíos a la implementación de la Visión Artificial en la industria guatemalteca, cómo se pueden abordar. Se examinan casos de estudio de diferentes industrias que han adoptado la Visión Artificial con éxito mejorando sus procesos.

Palabras claves: Industrial 4.0, Visión Artificial, *Deep Learning*, *Machine Learning*, Neurona, Tensorflow, *Tensorflow.JS*, Google *Colab*,

OBJETIVOS

General

Documentar, desarrollar e implementar un sistema de control de calidad automatizado usando Redes Neuronales Convolucionales, para mejorar el proceso de clasificación en el control de calidad, aplicables en la industria guatemalteca. A su vez inculcar competencias de investigación, desarrollo de sistemas de visión por computadora a futuros ingenieros.

Específicos

1. Facilitar información a los estudiantes sobre redes neuronales artificiales y cómo implementar Redes Neuronales Convolucionales para procesos de control de calidad.

2. Diseñar una arquitectura de la Red Neuronal Convolucional para el proceso de control de calidad de limones.

3. Documentar el proceso de muestreo, procesamiento de imágenes y entrenamiento de la Red Neuronal Convolucional. Como así también la validación del modelo mediante métricas de evaluación.

4. Implementar la Red Neuronal Convolucional desarrollada en un control de calidad.

INTRODUCCIÓN

El ser humano posee la capacidad de desarrollo como de adaptación ante los cambios del entorno, lo cual ha conducido a su evolución como especie llevándolo a la cima de la creación. Este progreso ha dado lugar a la tecnología, la cual ha disminuido el esfuerzo laboral. Un ejemplo de ello es la creación de dispositivos inteligentes que agilizan de manera más rápida tareas y operaciones que, para el ser humano, resultan tediosas o difíciles.

Los sistemas ciberfísicos (sistemas inteligentes) forma parte de la nueva revolución industrial, en la cual se utilizan modernos sistemas de control para la automatización de procesos industriales, conectando los medios de producción a una red de comunicación como nuevas formas de producción y optimización en tiempo real.

Si bien en las industrias el elemento humano es un factor importante para la productividad de esta, muchas buscaran un enfoque vigorizado en la eficiencia de productividad para impulsar el cambio, descartando estratégicamente ciertas tareas. Esta nueva visión evolucionara para hacer que las industrias sean más competitivas, creando el puente entre el mundo físico y digital, a través del internet de las cosas, o por sus siglas en inglés IoT (*Internet of Things*), produciendo un cambio en los sistemas de control de calidad e industrial, por medio del aprendizaje automático o *Machine Learning*. Reduciendo la necesidad de intervención humana.

1. REVOLUCIÓN INDUSTRIAL

La Revolución Industrial fue un periodo caracterizado por profundos cambios socioeconómicos y tecnológicos que se desarrolló desde la segunda mitad del siglo XVIII hasta la primera mitad del siglo XIX. Este proceso de transformación revolucionó los métodos de fabricación como de producción, generando un impacto duradero en la economía social.

La Revolución Industrial se caracterizó por una serie de innovaciones tecnológicas de mejoras en la producción que permitieron un aumento en la eficiencia y la producción en serie, siendo estas algunas de las innovaciones más importantes de esta época. Estos avances permitieron una mayor producción de bienes a un costo más bajo, lo que a su vez impulsó una mayor demanda con un crecimiento económico sostenido.

La revolución Industrial también estuvo marcada por una serie de cambios sociales y económicos importantes. La agricultura en conjunto con la industria se convirtió en las principales fuentes de empleo, la población urbana aumentó drásticamente que ocasiono una mayor demanda de trabajo en las ciudades para abastecer las necesidades que se empezaba a originar.

La Revolución Industrial fue un período de grandes cambios. Las innovaciones tecnológicas anudadas a los cambios socioeconómicos que se produjeron en la segunda mitad del siglo XVIII y la primera mitad del siglo XIX sentaron las bases para el desarrollo socioeconómicos de la actualidad.

1.1. Industria 1.0

La Industria 1.0, también conocida como la Primera Revolución Industrial se inició en Gran Bretaña, tuvo lugar a fines del siglo XVIII alrededor de 1760 y principios del siglo XIX alrededor de 1840-1860. Se caracterizó por la introducción de la máquina a vapor en conjunto con la producción en serie, lo que permitió una mayor eficiencia, productividad en la fabricación de bienes.

La máquina a vapor fue uno de los avances más importantes de la Industria 1.0, lo que permitió aumentar la producción y mejorar la eficiencia en la fabricación de bienes. Además, la producción en serie permitió fabricar productos en grandes cantidades a un costo más bajo, lo que a su vez impulsó una mayor demanda que incremento un crecimiento económico.

En la Industria 1.0, la producción de bienes se centró en la manufactura de productos de acero, textiles y maquinaria. Esta época también estuvo marcada por una serie de cambios socioeconómicos importantes, como la migración de la población rural a las áreas urbanas creando de nuevas fuentes de empleo en las ciudades.

1.2. Industria 2.0

La Industria 2.0, también conocida como la Segunda Revolución Industrial, tuvo lugar a finales del siglo XIX y principios del siglo XX (1860-1947). Se caracterizó por la introducción de la electrificación con la producción en masa de bienes a través de la aplicación de la tecnología automatizada en la producción industrial.

La electrificación permitió la creación de nuevos sistemas de producción, como la producción en masa y la producción por lotes, lo que permitió fabricar productos en grandes cantidades a un costo más bajo con mejor la eficiencia en la producción de bienes. Además, la introducción de la tecnología permitió la creación de nuevos productos como procesos, un ejemplo de ello es la producción de automóviles y la fabricación de productos químicos.

Durante la Industria 2.0, la economía se centró en la producción de bienes duraderos, como automóviles, maquinaria y electrónica, esta época al igual que la industria 1.0 fue marcada por una serie de cambios socioeconómicos importantes, la consolidación de la clase trabajadora en conjunto con el surgimiento de nuevos movimientos políticos-sindicales.

1.3. Industria 3.0

La Industria 3.0, también conocida como la Tercera Revolución Industrial, se refiere a la integración de la tecnología digital en la producción con la automatización de la industria. Se caracteriza por la utilización de tecnologías avanzadas, como la robótica, para mejorar la eficiencia, la personalización y la flexibilidad en la producción de bienes. Se considera que inicio en la década de 1950.

La Industria 3.0 es una continuación de la segunda revolución industrial, pero con un enfoque en la digitalización/automatización. Esto implica que los procesos industriales son controlados y monitoreados por sistemas digitales, lo que permite una mayor eficiencia además de un mejoramiento en la capacidad para personalizar los productos. La Industria 3.0 se caracteriza por una mayor interconexión entre los procesos de producción, lo que da una mejor coordinación con una mayor eficiencia en la producción. La robótica con la

integración de la inteligencia artificial también juega un papel importante en la Industria 3.0, permitiendo una automatización más avanzada con una mejora en la capacidad para personalizar los productos.

1.4. Industria 4.0

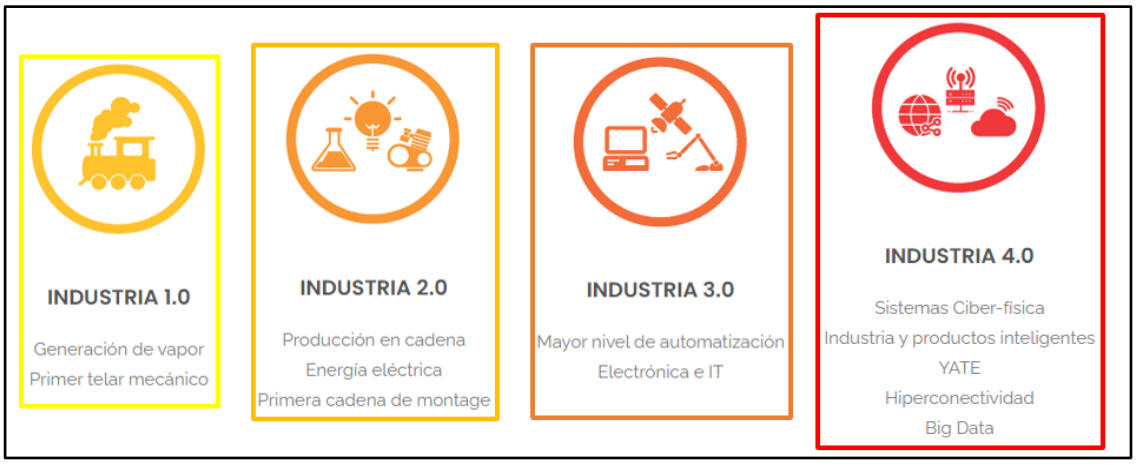
La Revolución Industrial 4.0 es un término utilizado para describir la evolución actual de la tecnología iniciando 1990 y 2000 hasta la actualidad. La industria 4.0 es un término que se refiere a la integración de tecnologías avanzadas, como la inteligencia artificial, la robótica, el internet de las cosas y la nube, volviendo la economía cada vez más digitalizada. En la producción industrial, la industria 4.0 promueve una fabricación más eficiente, flexible con opción a personalización, con mejora en la calidad, en la eficiencia de la producción.

La industria 4.0 también tiene como objetivo crear una producción más sostenible y responsable con el medio ambiente, utilizando tecnologías de energía renovable las cuales reducen el desperdicio minimizando la huella de carbono. La Revolución Industrial 4.0 está teniendo un impacto significativo en la forma en que las empresas operan y producen bienes, está cambiando la forma en la que se trabaja. Algunos ven estos cambios como desafiantes, mientras que otros los ven como una oportunidad para mejorar la eficiencia y eficacia en la producción.

La Revolución Industrial 4.0 es un desarrollo importante en la historia de la producción industrial que tendrá un impacto significativo en el futuro. Sin embargo, el impacto en los resultados finales depende de cómo se utiliza aunado a esto las regulaciones de leyes estatales y mundiales que conllevaría.

Figura 1.

Evolución Industrial



Nota: Representación gráfica de la evolución industrial. Elaboración propia, realizado con AfterShot Pro 3 RAW Photo Editor.

Los sistemas ciber-físicos están ganando impacto al permitir la integración de la tecnología digital en los procesos industriales complejos, lo que aumenta la eficiencia, la productividad mejorando la toma de decisiones, al utilizar inteligencia artificial para analizar, monitorizar y procesar datos en tiempo real, ayudara a identificar incluso solucionar problemas de manera más eficiente.

Los sistemas ciber-físicos con inteligencia artificial son una combinación de tecnologías de *hardware* y *software* que permiten la interacción e integración entre el mundo digital con el mundo físico, controlando los sistemas/dispositivos físicos, dotando de autonomía el proceso en cuestión, la eficiencia como la productividad no solo se presenta en sistemas industriales sino también la energía, la salud, la agricultura. La combinación de la visión artificial con la inteligencia artificial en conjunto con la robótica está liderando la transformación

de la industria hacia la Industria 4.0, permitiendo la creación de fábricas inteligentes con la automatización de procesos complejos.

1.5. La Inteligencia Artificial

La Inteligencia Artificial (IA) es una rama de la informática que a su vez es una rama de la ingeniería que su principal ocupación es el desarrollo de algoritmos y sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, como el reconocimiento de patrones, toma de decisiones, traducción de idiomas, entre otras. La idea es crear programas de computadora que puedan imitar la capacidad cognitiva de los seres humanos que incluso aprendan por sí mismos.

La Inteligencia Artificial es una tecnología en crecimiento que consiste en la imitación de la inteligencia humana por parte de las máquinas. La IA se utiliza en una amplia variedad de aplicaciones, desde el entretenimiento hasta la medicina. Tiene un gran impacto en la economía y en sectores clave como la seguridad, el transporte, la agricultura sumada a esto la educación. Sin embargo, su uso también plantea dilemas éticos donde la Unión Europea ha propuesto un proyecto de ley para regular su uso.

En 1997, en Nueva York, el ajedrecista Garry Kasparov se enfrentó a una supercomputadora de IBM llamada *Deep Blue* en un partido de ajedrez. Kasparov fue derrotado por la máquina, lo que se promocionó como un encuentro entre la inteligencia artificial y la humana. Desde entonces, el poder de las computadoras ha aumentado drásticamente con la disponibilidad de datos en tiempo real, lo que ha impulsado el desarrollo de tecnologías de aprendizaje de máquinas como el *Machine Learning* y el *Deep Learning*.

El término inteligencia artificial fue acuñado en 1956 en un proyecto de investigación en *Dartmouth College*. El objetivo era estudiar cómo simular funciones humanas superiores, como el lenguaje natural y el procesamiento de abstracciones. John McCarthy definió la IA como; la ciencia e ingeniería de hacer máquinas inteligentes. La Comisión Europea en 2018 definió la IA como un sistema que despliega un comportamiento inteligente que toma acciones con algún grado de autonomía para lograr objetivos específicos. Esta definición es comúnmente utilizada en la actualidad.

1.5.1. Aprendizaje automático (*Machine learnig*)

El aprendizaje automático es una rama de la inteligencia artificial que se enfoca en la construcción de algoritmos y modelos que permiten a las máquinas aprender a partir de los datos sin ser programadas explícitamente para realizar una tarea específica. El objetivo es desarrollar sistemas que puedan mejorar su rendimiento a medida que se les proporciona más información con la cual gana experiencia.

1.5.1.1. Aprendizaje supervisado

El aprendizaje supervisado es una técnica en *Machine Learning* que implica entrenar un modelo a partir de datos etiquetados previamente. En este enfoque, se proporciona al modelo una entrada junto con su etiqueta correspondiente donde el modelo aprende a mapear la entrada a la salida correcta. El objetivo del aprendizaje supervisado es encontrar una función que pueda predecir la salida correcta para una entrada dada, por lo que se utiliza comúnmente para problemas de clasificación y regresión.

Por ejemplo, en el caso de un problema de clasificación, el modelo aprende a clasificar los datos de entrada en diferentes categorías basadas en sus etiquetas previas. En cambio, en un problema de regresión, el modelo aprende a predecir un valor numérico continuo en función de los datos de entrada.

El aprendizaje supervisado se utiliza ampliamente en aplicaciones de Machine Learning, como la clasificación de imágenes, el reconocimiento de voz, la detección de fraude, la predicción de ventas.

1.5.1.2. Aprendizaje no supervisado

El aprendizaje no supervisado entrenar un modelo a partir de datos no etiquetados previamente. En este enfoque, el modelo debe encontrar patrones y relaciones ocultas en los datos sin la guía de las etiquetas de salida. El objetivo del aprendizaje no supervisado es descubrir la estructura intrínseca de los datos, identificando grupos o características ocultas.

El aprendizaje no supervisado se utiliza comúnmente en problemas de *clustering*, reducción de dimensionalidad y visualización de datos. Por ejemplo, en un problema de *clustering*, el modelo agrupa los datos similares en diferentes grupos basados en sus características, mientras que, en la reducción de dimensionalidad, el modelo reduce el número de características en los datos sin perder información importante.

El aprendizaje no supervisado también se utiliza para la detección de anomalías, la exploración de datos y la generación de datos sintéticos. A diferencia del aprendizaje supervisado, el aprendizaje no supervisado no

requiere etiquetas previas, lo que lo hace adecuado para grandes conjuntos de datos donde es costoso o difícil obtener etiquetas precisas.

1.5.1.3. Aprendizaje reforzado

El aprendizaje reforzado (o aprendizaje por refuerzo) es una rama del aprendizaje automático (*machine learning*) que se enfoca en cómo los agentes toman decisiones en un entorno dinámico para maximizar una recompensa acumulativa a lo largo del tiempo.

En el aprendizaje reforzado, un agente interactúa con un entorno y toma acciones para alcanzar ciertos objetivos. El agente recibe señales de recompensa o penalización según las acciones que tome, donde el objetivo es aprender a tomar las acciones óptimas que maximicen la recompensa acumulada a largo plazo.

El aprendizaje reforzado se basa en conceptos de teoría del control, donde se utilizan algoritmos con métodos para que el agente aprenda a través de ensayo y error, explorando diferentes acciones que son ajustados su comportamiento en función de las recompensas recibidas. Algunas técnicas populares en el aprendizaje reforzado incluyen los algoritmos de *Q-Learning*, la aproximación de funciones de valor con la planificación basada en modelos.

Este enfoque es particularmente útil en problemas donde no se dispone de datos de entrenamiento etiquetados, sino que el agente debe aprender a través de su interacción directa con el entorno y la retroalimentación de las recompensas. Algunas aplicaciones comunes del aprendizaje reforzado incluyen juegos, robótica, control de procesos como también la toma de decisiones en tiempo real.

1.5.1.4. Aprendizaje profundo (*Deep learning*)

El aprendizaje profundo o *Deep learning* es una sub-rama de la inteligencia artificial que se enfoca en el aprendizaje automático basado en redes neuronales profundas. Estas redes son estructuras computacionales que imitan la estructura como el funcionamiento del cerebro humano, permitiendo que el modelo aprenda a partir de datos para hacer predicciones sin ser programado explícitamente. El *Deep learning* es utilizado en una variedad de aplicaciones, como la visión por computadora, el procesamiento de lenguaje natural y la detección de patrones en datos.

El campo del *Deep learning* tiene varias subramas o áreas especializadas que se centran en aspectos específicos del aprendizaje profundo. Algunas de estas subramas incluyen, las siguientes.

1.5.1.5. Redes neuronales convolucionales (*Convolutional Neural Networks*, CNN)

Se utilizan principalmente en tareas de visión por computadora, como el reconocimiento de imágenes y el procesamiento de video. Las CNN están diseñadas para extraer características y patrones espaciales en datos con estructura de cuadrícula, como imágenes.

1.5.1.6. Redes neuronales recurrentes (*Recurrent Neural Networks*, RNN)

Estas redes están diseñadas para trabajar con datos secuenciales o de secuencia temporal, como el procesamiento de lenguaje natural, la traducción automática y el reconocimiento de voz. Las RNN son capaces de aprender

dependencias a largo plazo a través de la retroalimentación de las salidas anteriores.

1.5.1.7. Redes generativas adversariales (*Generative Adversarial Networks*, GAN)

Las GAN son una arquitectura compuesta por dos redes neuronales en competencia: un generador y un discriminador. Se utilizan para generar datos sintéticos realistas, como imágenes, música o texto, y han demostrado un gran potencial en el ámbito de la creación de contenido generativo.

1.5.1.8. Redes neuronales recurrentes con memoria a largo plazo (*Long Short-Term Memory*, LSTM)

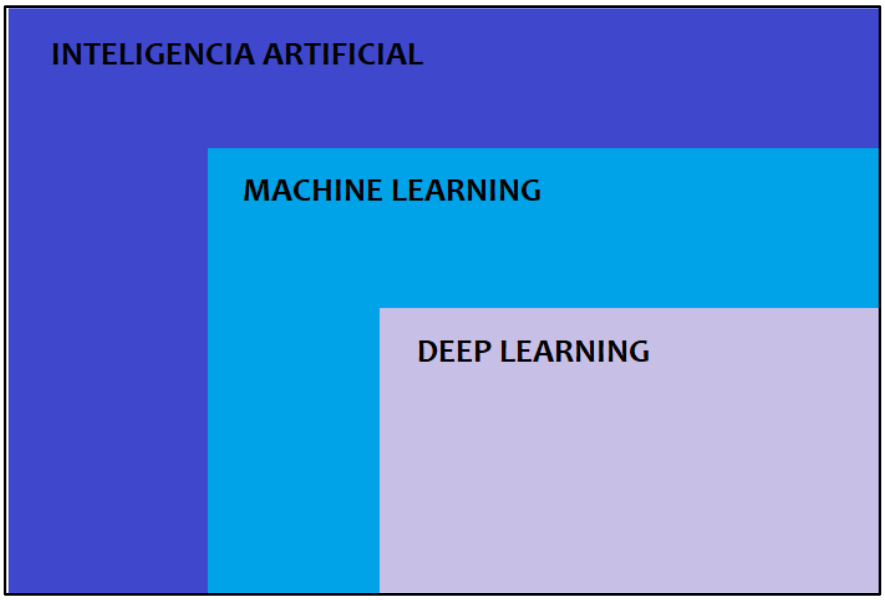
Son una variante de las RNN que se utilizan para aprender dependencias a largo plazo en datos secuenciales. Las LSTM son especialmente útiles cuando se necesita retener información relevante en secuencias largas.

1.5.1.9. Redes neuronales transformadoras (*Transformers*)

Estas arquitecturas se han vuelto populares en el procesamiento de lenguaje natural y han demostrado un gran rendimiento en tareas como la traducción automática y el modelado del lenguaje. Los *Transformers* se basan en el concepto de atención, que permite capturar relaciones entre palabras en una oración.

Figura 2.

Ramas de la inteligencia artificial



Nota: Ramas específicas de la Inteligencia Artificial. Elaboración propia, realizado con AfterShot Pro 3 RAW Photo Editor.

La visión artificial ha sido una herramienta valiosa en la industria para automatizar procesos y mejorar la eficiencia. Sin embargo, con el avance de la inteligencia artificial, ha surgido un modelo computacional llamado las redes neuronales artificiales como una evolución más avanzada de la visión artificial, permitiendo una comprensión más profunda como precisa de la información visual.

Las redes neuronales se están utilizando en aplicaciones industriales para mejorar la toma de decisiones que a su vez optimiza procesos complejos, estas redes imitan la estructura del funcionamiento del cerebro humano, lo que les permite aprender incluso la dota de cierta autonómica para tomar decisiones

basadas en la información que reciben. Algunos de los usos más comunes de las redes neuronales en la industria incluyen:

- Predicción de mantenimiento: las redes neuronales pueden utilizarse para predecir fallos en los equipos y sistemas industriales, lo que permite a los usuarios realizar el mantenimiento preventivo reduciendo costos.
- Control de calidad: las redes neuronales pueden utilizarse para inspeccionar productos y detectar defectos, lo que aumenta la eficiencia, precisión del control de calidad.
- Análisis de datos: las redes neuronales pueden utilizarse para analizar grandes cantidades de datos y extraer información valiosa que puede utilizarse para mejorar la toma de decisiones que a su vez optimiza los procesos industriales.
- Automatización de procesos: las redes neuronales pueden utilizarse para automatizar procesos complejos, como la manipulación de objetos o la optimización de la producción.

Para comprender el funcionamiento de la inteligencia artificial, es fundamental tener conocimientos sobre los fundamentos matemáticos en los que se basa.

1.6. Matemáticas de la Inteligencia Artificial

Las matemáticas son fundamentales para la inteligencia artificial (AI), en particular para el aprendizaje automático (*Machine Learning*), aprendizaje profundo (*Deep Learning*). En AI, los algoritmos utilizados para procesar los datos que a su vez tomar decisiones se basan en conceptos matemáticos como el álgebra lineal, la teoría de la probabilidad, el cálculo, la estadística y la optimización.

El álgebra lineal es utilizada en la definición de vectores y matrices, que son fundamentales para el procesamiento de datos en AI. La teoría de la probabilidad es esencial en el modelado como en la inferencia estadística en AI, así como en la creación de modelos de aprendizaje automático probabilísticos. El cálculo es utilizado para optimizar los parámetros de los modelos de AI, donde la estadística es esencial para evaluar la calidad de los modelos que hacen inferencias.

La optimización es otra área importante de las matemáticas en AI. Los algoritmos de aprendizaje automático y aprendizaje profundo se basan en la optimización para ajustar los parámetros de los modelos que mejoran su rendimiento. Los algoritmos de optimización utilizados en AI incluyen el descenso de gradiente, el algoritmo de Newton, la programación lineal, entre otros.

Las matemáticas son una parte integral de la inteligencia artificial y son utilizadas en todos los aspectos de la construcción como así también en aplicaciones de los modelos de aprendizaje automático, aprendizaje profundo. Sin las matemáticas, no sería posible diseñar o entrenar modelos de AI precisos como eficientes para procesar grandes cantidades de datos que tomen decisiones precisas.

1.6.1. Modelo de regresión lineal

El modelo de Regresión Lineal es una de las bases de Machine Learning y Análisis Estadísticos. Utilizada para predecir una variable continúa a partir de una o más variables de entrada (llamadas características o atributos). Se utiliza principalmente para problemas de regresión, para predecir un valor numérico continuo como la edad, el precio de una vivienda, el ingreso, entre otros.

Modelo regresión lineal simple:

$$Y = W_0 + W_1 x$$

Dónde:

Y= variable

W_0= coeficiente constante no varía de valor (Intercepto)

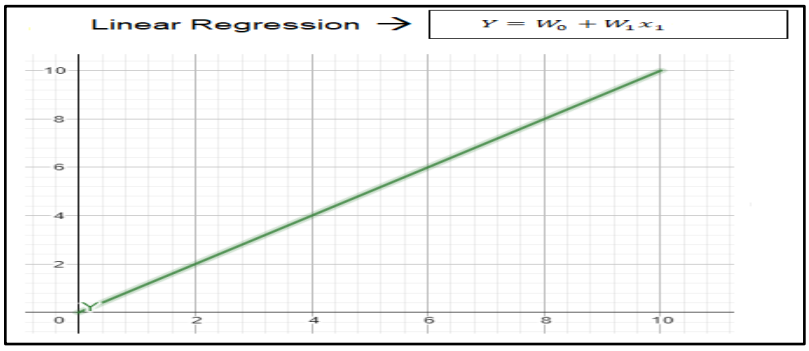
W_1= coeficiente constante no varía de valor (Pendiente)

x= variable

Tomando como ejemplo el precio de una vivienda en la ciudad, su costo dependerá de la zona en la que se encuentre. En general, las áreas más concurridas tienden a tener precios más altos que las zonas adyacentes a los suburbios. Al graficar los valores de los precios en función de las distintas zonas de la ciudad, se obtendría un resultado similar al siguiente:

Figura 3.

Modelo de regresión lineal simple



Nota: Modelo de Regresión Lineal Simple en costo de vivienda Vs Zona de la ciudad. Elaboración propia, realizado con WolframAlpha.

Con este modelo de regresión lineal simple se podría predecir el valor de la vivienda dependiendo de la zona de la ciudad, dado que sigue un patrón, entre más cerca se encuentra la vivienda de la ciudad mayor será su precio, claro, existirán excepciones en las que una vivienda alejada de la ciudad tenga un costo mayor que una ubicada dentro de la misma.

Esto se debe a factores como el número de habitaciones o el tamaño en metros cuadrados, en el anterior ejemplo de la vivienda de la ciudad solo existe una variable de entrada, ¿pero ¿qué sucede si ahora se analiza con esta nueva variable de entrada, si la casa tiene más habitaciones o metros cuadrados y esta no se encuentra dentro de la ciudad? El modelo dejaría de ser de regresión lineal simple, convirtiéndose en un modelo de regresión lineal múltiple, donde la ecuación sería similar a la de la regresión lineal simple, pero con la incorporación de más variables de entrada. A continuación, se muestra cómo se vería la ecuación ampliada:

Modelo regresión lineal múltiple:

$$Y = W_0 + W_1 x_1 + W_2 x_2$$

Dónde:

Y= variable

W_0= coeficiente constante no varía de valor (Intercepto)

W_1= coeficiente constante no varía de valor (Pendiente)

W_2= coeficiente constante no varía de valor (Pendiente)

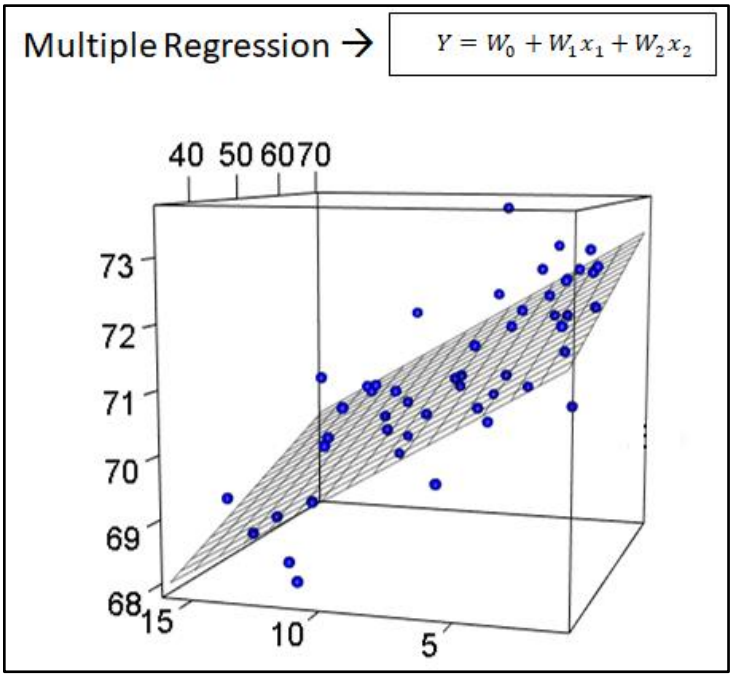
x_1= variable

x_2= variable

Ahora, el modelo de Regresión Lineal ha pasado de ser bidimensional a incluir hiperplanos en espacios multidimensionales debido a la adición de una nueva variable, como el número de habitaciones o metros cuadrados, y al considerar viviendas que se encuentran fuera de la ciudad.

Figura 4.

Regresión Lineal con Hiperplanos en espacios Multidimensional



Nota: Regresión Lineal con Hiperplanos en espacios Multidimensional Costo de vivienda Vs Zona de la Ciudad. Elaboración propia, realizado con WolframAlpha.

Cada una de las dimensiones representa una característica de la realidad (número de habitaciones, metros cuadrados, peligrosidad del área, entre otros) y la forma de representar cada una de combinación lineal de los datos es de forma Vectorial. Reemplazando las ecuaciones que se muestran.

$$Y_1 = W_0 + W_{11}x_{11} + W_{12}x_{12} + W_{13}x_{13} + \cdots$$
$$Y_2 = W_0 + W_{21}x_{21} + W_{22}x_{22} + W_{23}x_{23} + \cdots$$
$$Y_3 = W_0 + W_{31}x_{31} + W_{32}x_{32} + W_{33}x_{33} + \cdots$$
$$Y_4 = W_0 + W_{41}x_{41} + W_{42}x_{42} + W_{43}x_{43} + \cdots$$
$$Y_5 = W_0 + W_{51}x_{51} + W_{52}x_{52} + W_{53}x_{53} + \cdots$$

En una matriz de características de datos de entrada X, donde cada columna represente número de habitaciones, metros cuadrados, peligrosidad del área de la casa, entre otros., y cada fila represente las mediciones de conjuntos de datos:

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{43} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{bmatrix}$$

De igual forma se realiza con la variable Y la cual es denominado como un vector de elementos:

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \end{bmatrix}$$

De la misma forma se procede con los parámetros para crear un Vector de paramentos:

$$W = [W_0 \quad W_1 \quad W_2 \quad W_3]$$

El cual se resumen en una Ecuación Vectorial:

$$Y = XW$$

Por lo cual con esta ecuación permitirá a los modelos entrenarse más eficientemente pues la GPU se especializan en trabajar con matrices.

En resumen, el modelo de Regresión Lineal consiste en ajustar una línea recta a los datos de entrenamiento que se ajuste mejor a los valores observados de, la variable objetivo. Esto se logra minimizando la suma de los errores cuadrados entre los valores observados y los valores predichos por la línea recta.

Una vez que el modelo está ajustado a los datos de entrenamiento, se puede utilizar para hacer predicciones sobre nuevos datos. Para ello, se introducen los valores de las características del nuevo dato en el modelo y se obtiene una predicción del valor de la variable objetivo.

El modelo de Regresión Lineal tiene muchas variantes, incluyendo la Regresión Lineal Múltiple (cuando hay varias características de entrada) y la Regresión Lineal Regularizada (para evitar problemas de sobreajuste o *underfitting*).

1.6.2. Algoritmo del descenso del gradiente

El descenso del gradiente es un algoritmo de optimización utilizado en el aprendizaje de redes neuronales como en otros problemas de aprendizaje automático. El objetivo del descenso del gradiente es minimizar la función de costo (J) o pérdida de una red neuronal ajustando los pesos (W_i) y sesgos (b) de las neuronas en la red.

En el descenso del gradiente, se calcula la tasa de cambio (derivada parcial) de la función de costo con respecto a los pesos y sesgos de la red,

utilizando el gradiente de la función de costo. El gradiente indica la dirección de mayor aumento de la función de costo, por lo que se sigue la dirección opuesta (negativa) para minimizar la función de costo.

El algoritmo ajusta los pesos y sesgos de la red en pequeños incrementos, multiplicando el gradiente por una tasa de aprendizaje, que es un hiperparámetro que se ajusta para controlar la rapidez del aprendizaje. El proceso se repite iterativamente hasta que se alcanza un mínimo local de la función de costo, lo que indica que los pesos y sesgos de la red han sido ajustados para producir la salida deseada para las entradas dadas.

Como las funciones para el descenso del gradiente son multidimensionales se tendrá que calcular derivadas parciales para cada uno de los parámetros (valores obtenidos durante el entrenamiento), y cada uno de estos resultados dará las pendientes para cada uno de los parámetros.

$$\frac{\partial\ error}{\partial \theta_1} \quad ; \quad \frac{\partial\ error}{\partial \theta_2}$$

Y todas las derivadas conformarán un vector donde la pendiente asciende:

$$\begin{bmatrix} \frac{\partial\ error}{\partial \theta_1} \\ \frac{\partial\ error}{\partial \theta_2} \end{bmatrix} = \Delta f$$

Dónde:

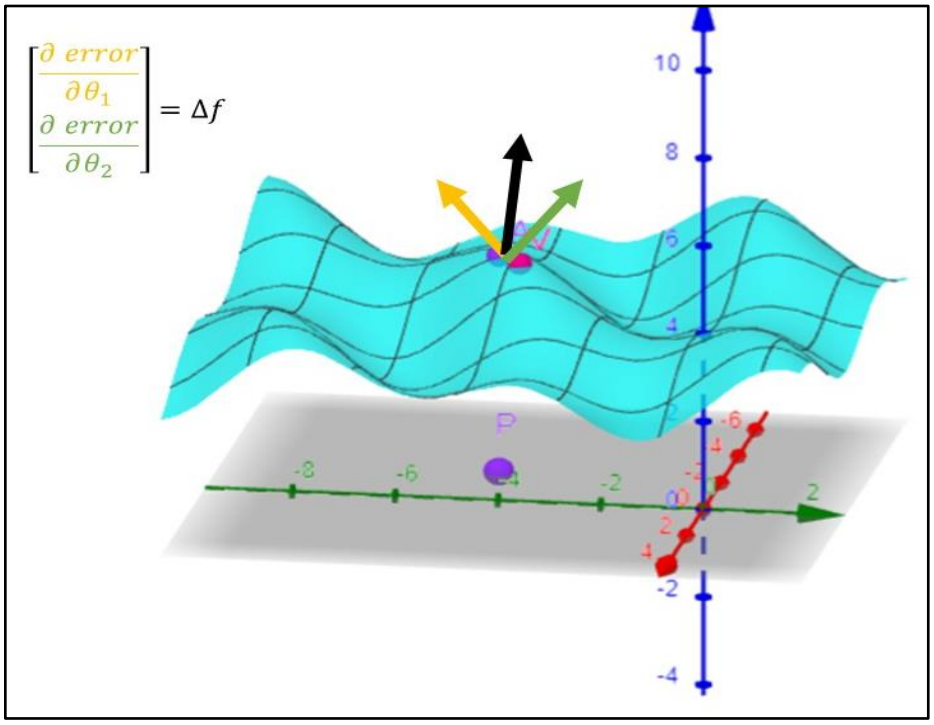
Δf= gradiente

$\frac{\partial\ error}{\partial \theta_1}$= derivada parcial parámetro 1

$\frac{\partial\ error}{\partial \theta_2}$= derivada parcial parámetro 2

Figura 5.

El gradiente



Nota: Visualización del gradiente. Elaboración propia, realizado con WolframAlpha.

En la figura 5 se muestra que el Gradiente de los parámetros ascienden, no obstante, se busca es el sentido puesto, el descenso, basta con restar el parámetro.

$$\theta := \theta - \nabla f$$

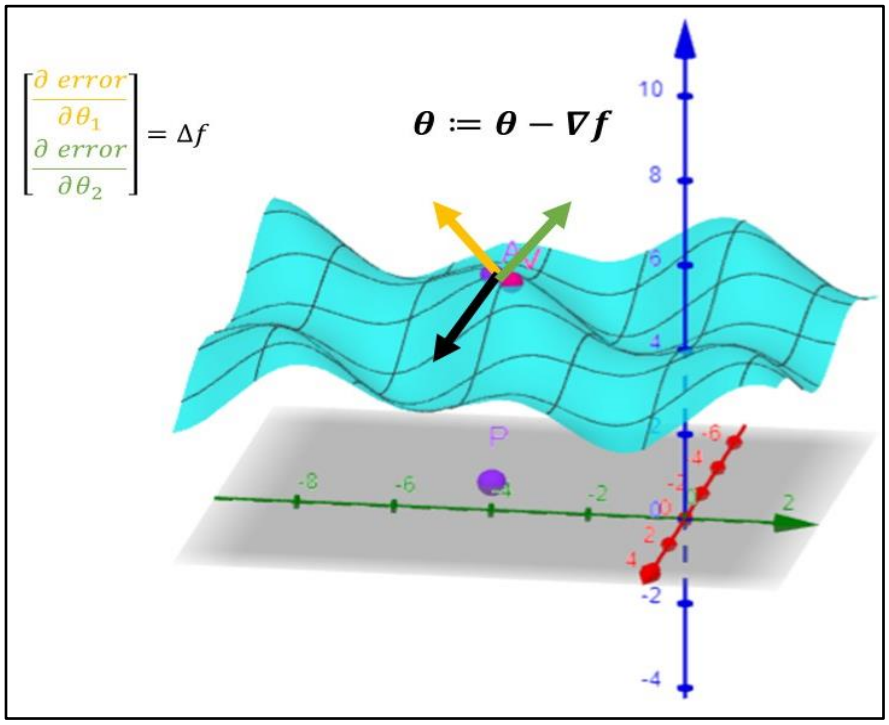
Dónde:

θ= parámetros

Δf = gradiente

Figura 6.

Descenso del gradiente

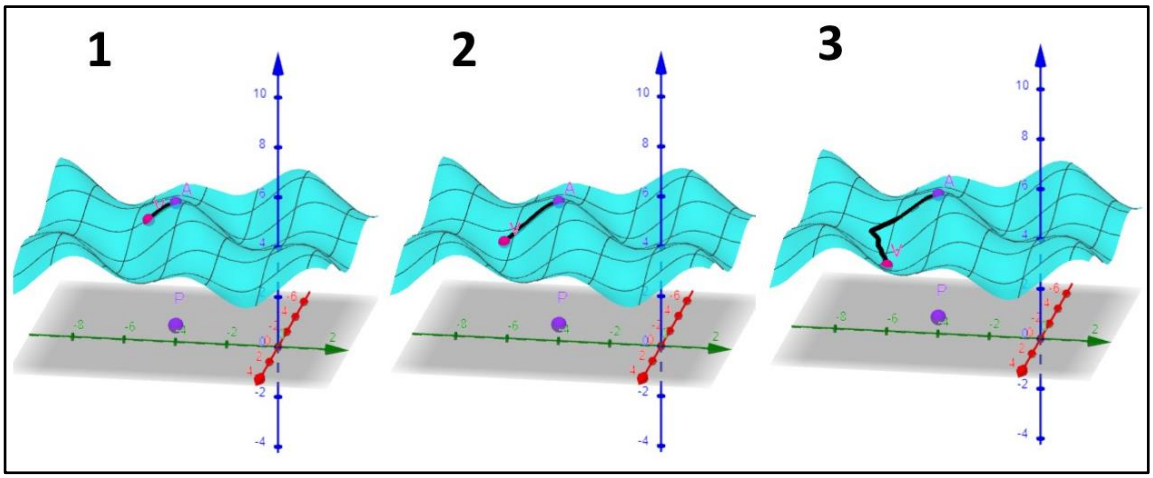


Nota: Visualización del descenso del gradiente. Elaboración propia, realizado con WolframAlpha.

Durante el descenso del gradiente se obtendrá con un conjunto de parámetros el cual resultará en un nuevo punto de la función donde se repetirá múltiples veces el proceso de descenso hasta llegar a una zona donde será el punto de mínimo local y el cual minimiza el coste del modelo.

Figura 7.

Descenso del gradiente para mínimo coste del modelo



Nota: Visualización del punto mínimo local. Elaboración propia, realizado con WolframAlpha.

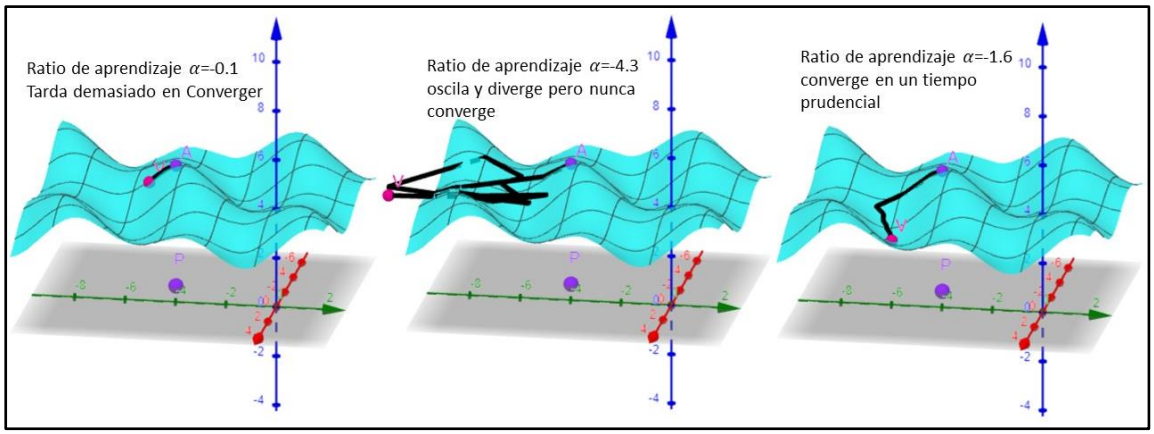
Para completar la ecuación hay que añadir un parámetro más y el cual se denomina Ratio de Aprendizaje α.

El Ratio de Aprendizaje determina la magnitud del paso que se toma en cada iteración del algoritmo para actualizar los parámetros del modelo. Si el ratio de aprendizaje es muy pequeña, el algoritmo tardará mucho en converger y si es muy grande, puede saltar por encima del mínimo global en el que se encuentra la función de coste.

Es importante elegir un valor adecuado para el Ratio de aprendizaje. Si es demasiado bajo, el algoritmo puede tardar mucho tiempo en converger, mientras que, si es demasiado alto, el algoritmo puede oscilar o divergir en lugar de converger.

Figura 8.

Importancia de elegir un valor adecuado de Ratio de Aprendizaje



Nota: Importancia de elegir un valor adecuado de Ratio de Aprendizaje. Elaboración propia, realizado con WolframAlpha.

El valor del ratio de aprendizaje puede ajustarse mediante métodos como la validación cruzada o el uso de técnicas de búsqueda de hiperparámetros. El objetivo es encontrar un valor óptimo que permita al algoritmo de descenso del gradiente converger rápidamente y obtener un modelo preciso.

Dicho lo anterior se muestra la ecuación del algoritmo del descenso de gradiente.

$$\theta := \theta - \alpha \nabla f$$

Dónde:

θ= parámetros

α=ratio de aprendizaje

Δf= gradiente

1.6.3. Modelo de regresión polinomial lineal

Es un tipo de modelo de aprendizaje automático que se utiliza en problemas de regresión para modelar relaciones no lineales entre una variable de entrada y una variable de salida. En lugar de ajustar una línea recta como en la regresión lineal, el modelo de regresión polinomial ajusta una curva de grado mayor que 1.

En general, un modelo de regresión polinomial puede ser definido por la ecuación:

$$y = w0 + w1x1 + w2x1^2 + ... + wn*x1^k + e$$

Dónde:
y= variable dependiente
x1= variable dependiente
w0=w1, w2..., wk son los coeficientes del modelo
k= exponente del modelo
e= término de error.

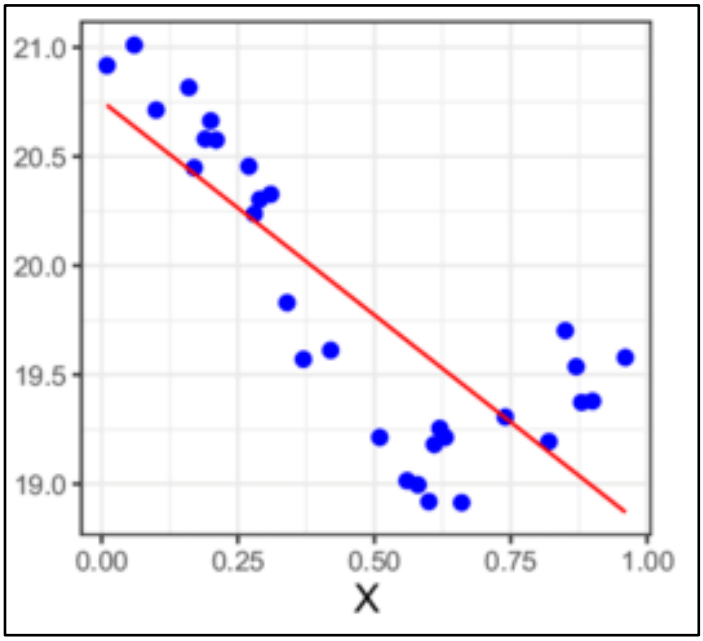
El grado de la curva se puede ajustar aumentando o disminuyendo el valor del exponente del modelo k. Cuanto mayor sea el grado de la curva, mayor será la complejidad del modelo. Por lo tanto, es importante equilibrar la complejidad del modelo con su capacidad para ajustarse a los datos.

Para ejemplificar lo anteriormente mencionado en la figura 9, se muestra una dispersión de datos, si se utiliza una ecuación lineal, se encontraría que la hipótesis de que los datos siguen la regla de aproximación lineal es falsa, puesto que la recta no es capaz de adaptarse a la naturaleza de los datos. Por

lo tanto, se tendrá una ecuación mal ajustada o lo que en un modelo de aprendizaje automático se denomina Mal ajustado o por su término en inglés se denomina como *Underfitting*.

Figura 9.

Gráfica polinomial de grado 1

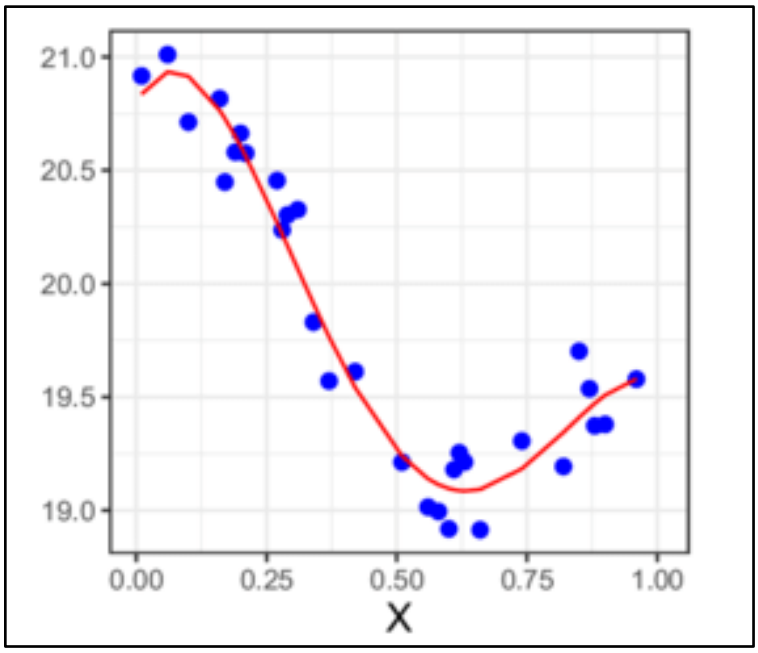


Nota: Gráfica polinomial de grado 1. Elaboración propia, realizado en Qtiplot.

Al asumir la ecuación como una recta no existe una flexibilidad que permite adaptar la gráfica con respecto a la dispersión de datos, para adaptar la ecuación a la dispersión de datos se debe aumentar el grado del polinomio elevando el exponente (k) de la variable dependiente (x), el nuevo grado de la ecuación polinomial determinará el número de puntos críticos de la curva, entre más grados polinomiales se tenga más flexible será la curva.

Figura 10.

Gráfica polinomial de grado 4

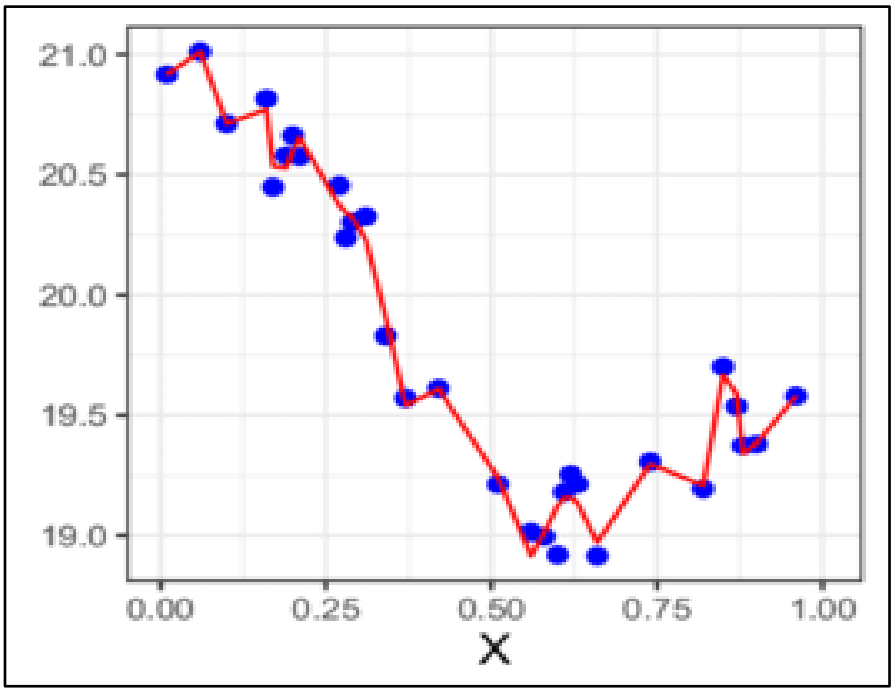


Nota: Gráfica polinomial de grado 4. Elaboración propia, realizado con WolframAlpha.

Quizá hasta este punto se piense que, entre más flexibilidad posea la ecuación o dicho de otra manera entre mayor sea el grado del polinomio mejor se adaptara a los datos. Y efectivamente la curva, ver figura 11 se ajusta mejor a la curva de la figura 10, pero como se verá más adelante, en modelos de aprendizaje automático entre mayor sea el grado de la ecuación polinomial, mayor será la distorsión del patrón real lo cual ocasionara un error en la predicción de datos a esto se denomina como sobre ajuste o por sus siglas en inglés *Overfitting*.

Figura 11.

Gráfica polinomial de grado 20



Nota: Gráfica polinomial de grado 20. Elaboración propia, realizado con WolframAlpha.

Esta teoría de Modelo de Regresión Polinomio se utilizará en el modelo de regresión lineal simple cuando la gráfica no se ajuste a los datos y al utilizar el modelo de descenso del gradiente ajusta la curva a los datos para así mejorar la taza de aprendizaje.

2. FUNDAMENTO DE LAS REDES NEURONALES ARTIFICIALES

2.1. Red neuronal humana

El cerebro es considerado la estructura más compleja del cuerpo humano, siendo una estructura única y altamente sofisticada en el reino animal que presenta aptitudes fascinantes, con facultades que va desde, memorísticas, asociativas a hechos vividos, adaptativa en entornos cambiantes, imaginativos, experimentales como de razonamiento, estas características que presenta han llevado a querer comprender como funciona.

Pero ¿quién realmente trata de comprender al cerebro? se supone que la mente humana es un cerebro y no una sustancia inmaterial que flota sobre la cabeza, entonces ¿es el cerebro quien intenta explicarse a sí mismo?, quizá se piense que la respuesta se esté desplazando de una especie inmaterial que se denomina alma o mente al órgano que parece albergarla, esta comprensión de la mente humana que antiguamente fue dominio de la filosofía, está en manos de lo que se denomina como, la neurociencia una ciencia multidisciplinaria centrada en el estudio del cerebro como las funciones cognitivas.

Que hasta el momento de redacción de este tema de investigación no se ha encontrado aún la respuesta concreta a la pregunta planteada, pero lo que, si se ha descubierto gracias a los estudios interdisciplinarios, es la existencia de una célula con características únicas que se encuentra el cerebro la cual recibe la información del exterior que la transporta por medio de impulso nervioso

siendo parte fundamental del razonamiento como de otras aptitudes la cual es llamada neurona.

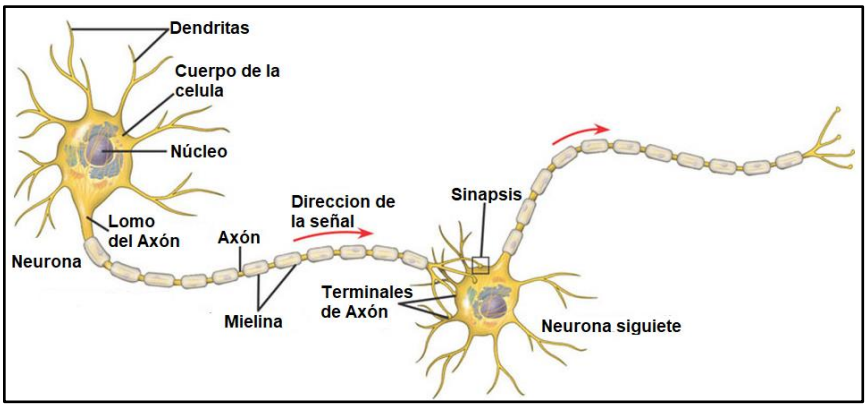
El cerebro funciona formando redes por medio de neuronas, en la cual cada una de las neuronas se comunican entre sí, esta comunicación crea largas ramificaciones, que a su vez crean un bosque neuronal de alta complejidad, comprendidas de 10^{11} neuronas celulares fundamentales, en comunicación con miles u millares de otras por medios de conexiones neuronales denominados nodos.

Las capacidades que presenta el cerebro humano han despertado a diferentes campos disciplinarios como, la ingeniería, psicología, fisiología y filosofía a derivar varios estudios en colaboración con las neurociencias para crear modelos computacionales que emulen funciones básicas del cerebro que ha resultado en lo que se denomina como, redes neuronales artificiales.

Para comprender las bases de la red neuronal artificial es necesario comprender como funciona las neuronas biológicas del cerebro humano y sistema nervioso. Si bien existen distintos tipos de neuronas, se presenta un esquema generalizado en la figura 12, de lo que básicamente es una neurona.

Figura 12.

Morfología de una neurona biológica



Nota: Morfología de una neurona biológica. Elaboración propia, realizado con AfterShot Pro 3 RAW Photo Editor.

La comunicación de las neuronas se logra por medio de un proceso de comunicación intercelular denominado sinapsis, ya sea con neuronas o tejidos efectores (músculos y glándulas), se clasifican según la transmisión del impulso nervioso, que pueden ser eléctricos o químicos. El siguiente trabajo de investigación, no busca profundizar en los tipos de sinapsis según la transmisión del impulso nervioso, puesto que el objetivo es comprender el funcionamiento biológico sin indagar en el tema, por ello de manera resumida se expresará como impulsos electroquímicos.

Como se observa en la figura 12; del cuerpo de la neurona se ramifica fibras llamadas dendritas con una fibra más larga llamada axón, el cual transmite las señales, las dendritas permiten la recepción, para lograr una transmisión de datos se necesita un potencial de acción, denomina como impulso nervioso, en otras palabras la neurona recibe una estimulación través de sus entradas, es decir, dendritas, cuando alcanza un cierto umbral, la

neurona se activa, pasando una señal hacia el axón (salida de la neurona), esto logrado por medio de impulsos electroquímicos que transmite la información a la siguiente neurona.

2.2. Red neuronal artificial

Las redes neuronales artificiales o por sus siglas en inglés: Artificial Neuronal Networks ANN. Son modelos computacionales que emulan características de un cerebro humano, las ANN tratan de capturar la esencia del sistema neuronal biológico imitando la capacidad de asociar hechos, de memorizar que incluso aprenden a través de experiencia acumulada, emulando un bosque neuronal biológico que simula la actividad eléctrica y sistema nervioso del cerebro para así poder interactuar con el mundo real, de la forma que lo hace el sistema nervioso biológico.

2.2.1. Características de las redes neuronales artificiales

Las redes neuronales artificiales son modelos computacionales inspirados en el cerebro humano que se utilizan para el procesamiento de información y el aprendizaje automático. Estas redes están compuestas por nodos interconectados llamados neuronas artificiales y son capaces de aprender patrones y realizar tareas de reconocimiento, clasificación y predicción a partir de datos.

2.2.1.1. Capacidad de aprendizaje

Presentan la capacidad de aprender por medio de ejercicios y experiencias, con capacidad de adaptar su comportamiento en función de su entorno tanto aditivo como de visión.

2.2.1.2. Consideraciones de entorno

Trata de manera general la reducción de ruidos o distorsiones del entorno, teniendo entradas con distorsión, pero presentando resultados correctos. Debido a la capacidad de modificación de sus conexiones entre neuronas, de esta manera puede aprender de experiencia y generaliza conceptos.

2.2.1.3. Capacidad de separación de cualidades

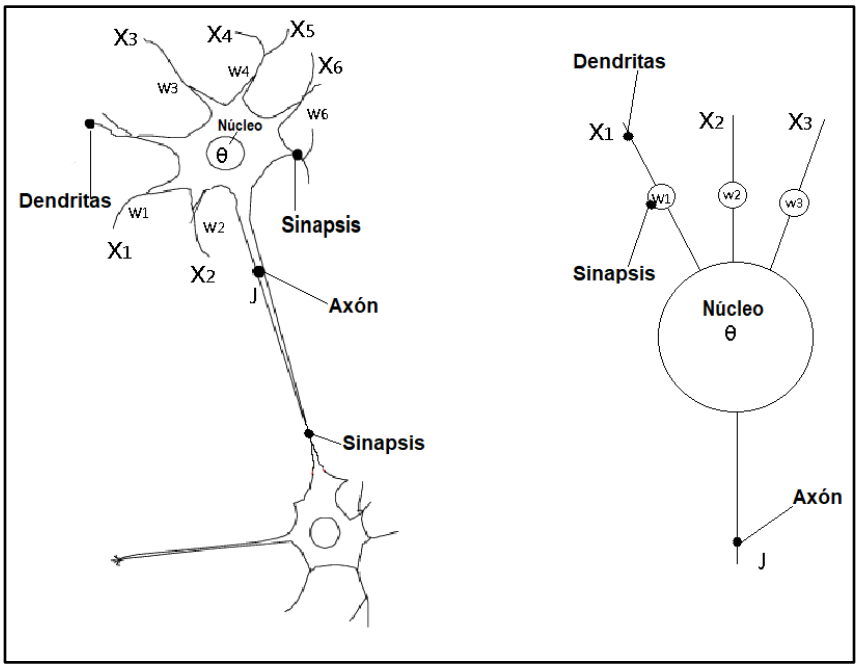
Aísla cualidades específicas del objeto, con capacidades de abstraer la esencia como podría ser contornos, formas, colores o pixeles específicos.

2.3. Estructura básica de las redes neuronales artificiales

Las neuronas artificiales al igual que las neuronas biológicas serán estimuladas con señales de entrada, a diferencia de las neuronas biológicas estas no son estimuladas con actividades sinápticas, al ser un modelo computacional recibirán estímulos de tipo vectoriales a la entrada (Ver figura 13) con estos vectores entrantes la neurona artificial realizara un cálculo interno y generara un valor de salida, vista desde una perspectiva matemática, la neurona artificial se comporta como una función f(x).

Figura 13.

Comparativa de una neurona biológica (izq.) y una neurona artificial (derecha)



Nota: Comparativa de una neurona biológica (izquierda) y una neurona artificial (Derecha). Elaboración propia, realizado con AfterShot Pro 3 RAW Photo Editor.

Internamente la neurona artificial realiza una suma ponderada de todos sus vectores de entrada (x_n) para así obtener una salida (z), esta suma ponderada es logrado gracias a un valor conocido como peso sináptico (w_i) que será asignada en la entrada, estos pesos sinápticos estipulan la prioridad a cada entrada para ser procesadas en la sumatoria. También es necesario agregar una señal de entrada permanentemente activada denominada sesgo, (b) el cual es un valor que dispara la función de activación para garantizar un aprendizaje exitoso con: uno (1) neurona activada y cero (0) neurona desactivada.

En el modelo la salida z de la neurona está dada por:

$$z = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + \cdots + x_i * w_i + b$$

Si se extrae la ecuación anterior a cualquier número de dimensiones se expresa como:

$$z = b + \sum_i w_i x_i$$

Dónde:

z = salida de la neurona

b = sesgo de activación constante

w_i = peso sinóptico

x_i = vector entrada

Si se vuelve a las bases de las redes neuronales artificiales, las neuronas biológicas, para transmitir la información de una neurona a otra, lo hacen mediante un potencial de acción los cuales son impulsos nerviosos que dispara la neurona, en la figura 12 se mostró como dirección de señal. Este potencial de acción es homologo a lo que en redes neuronales artificiales se denomina, funciones de activación.

2.3.1. Funciones de activación

En las neuronas biológicas, se pueden encontrar en dos estados: excitada o no excitada, es decir, tienen un estado de activación, de la misma forma sucede con las neuronas artificiales, también tiene estado de activación, esto indicara si el proceso es importante o no, en función de su activación comprendido en dos rangos que se encuentra entre (0,1) o de (-1,1), esto es

debido a que la neurona puede estar completamente inactiva (-1,0) y totalmente activa (1).

La función de activación transmite la información generada por la suma ponderada de las entradas y así definir la salida pudiendo predecir la información, es por ello por lo que la elección de la función de activación impacta tanto en la capacidad como en el rendimiento de la neurona, dado que con una mala elección se necesitara grandes cantidades de nodos para calcular problemas no triviales conllevando a usar más recursos de los necesarios.

Hay que considerar la función de activación como un modelo matemático la cual es no lineal, debido a que debe adaptarse a cantidades enormes de datos.

Para entender las funciones de activación es necesario definir tres tipos de capas que conforman la neurona artificial y como un conjunto de capas secuenciales formara una arquitectura de neuronas.

2.3.1.1. Capa de entrada

Recibe directamente la información sin procesar el dominio, el número de nodos es directamente proporcional a los vectores de entrada.

2.3.1.2. Capas ocultas

No tiene contacto directo con el entorno exterior, el número de niveles ocultos puede variar, pudiendo ser cero capas y un número elevado de capas, entre más capas ocultas mejor se logra el aprendizaje, aunque esto puede variar, según el modelo o a lo que se quiere orientar, estas capas ocultas filtran

la información importante, no producen salidas del modelo. Su comportamiento es metafóricamente una caja negra.

2.3.1.3. Capa de salida

Transfieren la información procesada hacia el exterior, combinando los datos de las capas anteriores para así producir la salida.

Hay tres tipos de funciones de activación que por lo general se utilizar en las capas ocultas, no son todas las que existen, pero son las más utilizadas:

- Activación Lineal Rectificada (ReLu)
- Logística (Sigmoide)
- Tangente Hiperbólica (Tanh)

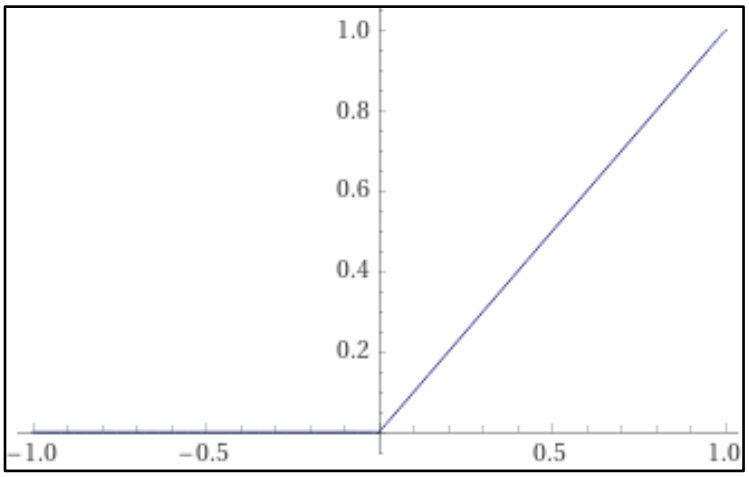
2.3.1.4. Función de activación Lineal Rectificada (ReLu)

Es una de las funciones más utilizadas en capas ocultas, debido a que permite un aprendizaje rápido es simple de implementar como efectiva en comparación a otras funciones como la logística y la tangente hiperbólica. Esta función es utilizada recurrentemente en red neuronal convolucional como en perceptor multicapa.

$$R(x) = \max(0, x)\,, x = [-1,1]$$

Figura 14.

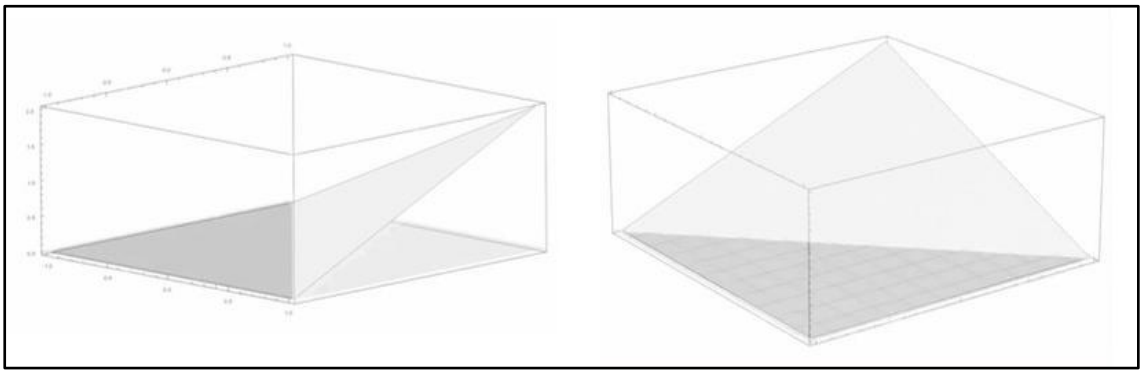
Función de activación Lineal Rectificada (ReLu)



Nota: Función de activación Lineal Rectificada. Elaboración propia, realizado con WolframAlpha.

Figura 15.

Función de activación Lineal Rectificada (ReLu) con vista Geométrica



Nota: Función de activación Lineal Rectificada (ReLu) con vista Geométrica. Elaboración propia, realizado con WolframAlpha.

2.3.1.5. Función de activación Logística (*Sigmoide*) y Tangente hiperbólica (Tanh)

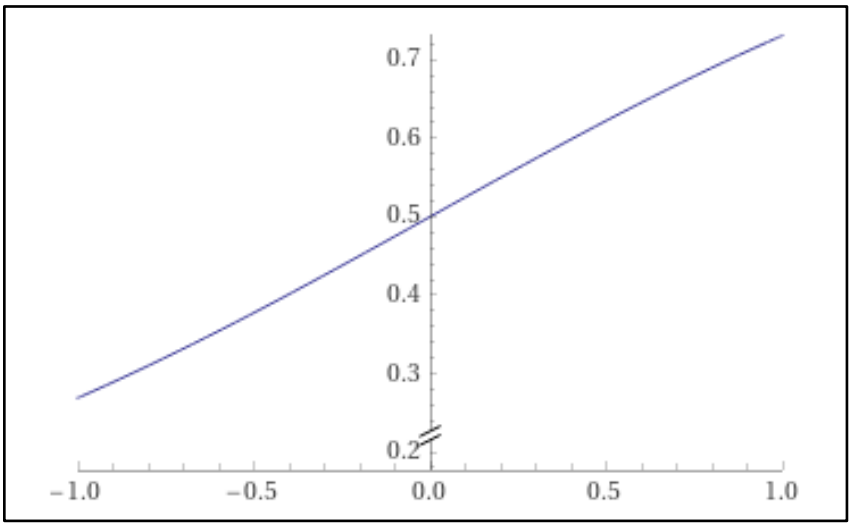
Estas funciones hacen el modelo más susceptible en los entrenamientos suelen ser utilizados en arquitecturas de modelos de memoria a largo o corto plazo, la función logística para conexiones recurrentes y la tangente hiperbólica para activación de salidas.

- Función Logística (Sigmoide)

$$R(x) = \frac{1}{1 + e^{-x}} \, , x = [-1,1]$$

Figura 16.

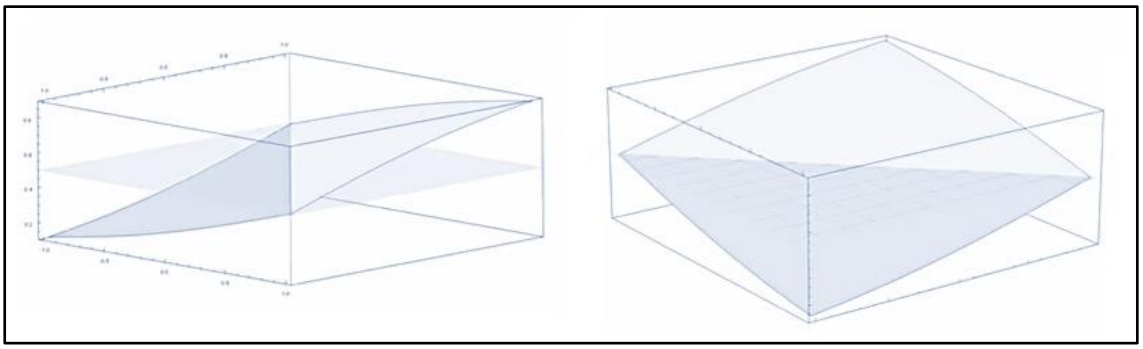
Función logística (sigmoide)



Nota: Función logística. Elaboración propia, realizado con WolframAlpha.

Figura 17.

Función de activación Sigmoide con vista Geométrica



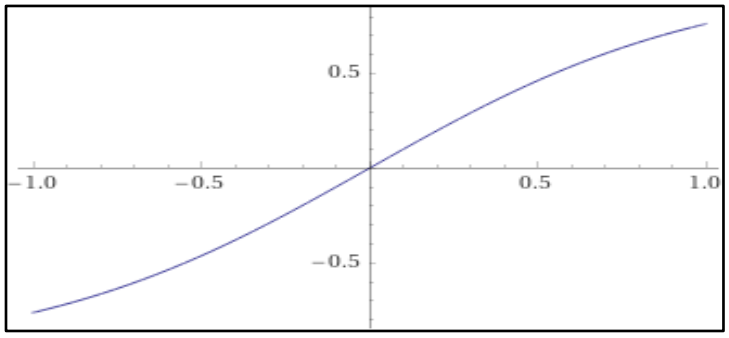
Nota: Función de activación Logística (Sigmoide) con vista Geométrica Elaboración propia, realizado con WolframAlpha.

- Función Tangente Hiperbólica (Tanh)

$$R(x) = \frac{e^x - e^{-x}}{e^x + e^x}, x[-1,1]$$

Figura 18.

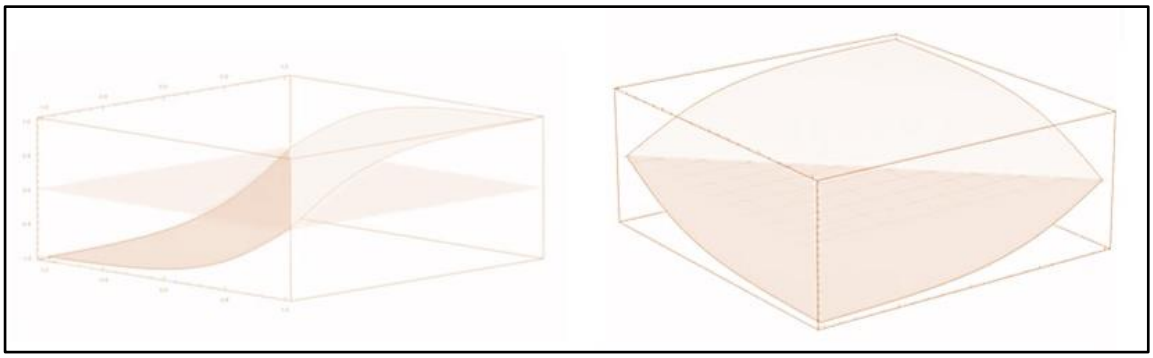
Función Tangente Hiperbólica (Tanh)



Nota: Función Tangente Hiperbólica (Tanh). Elaboración propia, realizado con WolframAlpha.

Figura 19.

Función de activación Tangente Hiperbólica (Tanh) con vista Geométrica



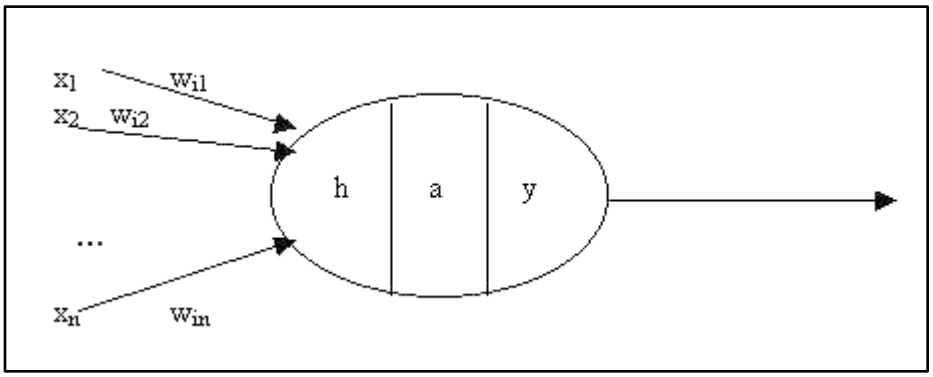
Nota: Función de activación Tangente Hiperbólica (Tanh) con vista Geométrica Elaboración propia, realizado con WolframAlpha.

Como se ha descrito anteriormente las neuronas artificiales se organizan en capas esto forma una arquitectura de redes neuronales, dependiendo al problema a solucionar así será su arquitectura, pero en su gran mayoría todas las redes neuronales usan el mismo principio.

Por medio coeficientes de peso sináptico ayudan a clasificar la prioridad de la entrada (Denominado también predictores) para posteriormente realizar una suma pondera con los vectores de entrada esto representara la fortaleza de la conexión y luego activa un umbral (función de activación), al activar la función de activación, transforma el valor de la suma ponderada entrante en una función no lineal para así obtener una salida, la cual ayudara a la predicción. Con esto en cuenta se muestra la neurona artificial.

Figura 20.

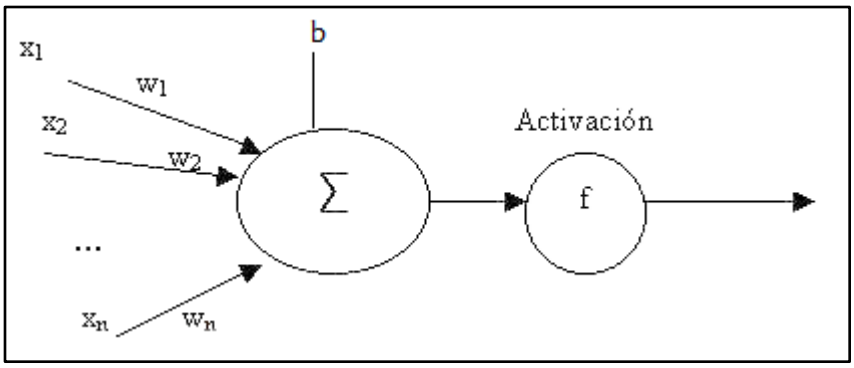
Modelo neuronal estándar



Nota: Modelo neuronal estándar. Elaboración propia, realizado con AfterShot Pro 3 RAW Photo Editor.

Figura 21.

Modelo neuronal simplificado



Nota: Modelo neuronal simplificado. Elaboración propia, realizado con AfterShot Pro 3 RAW Photo Editor.

- Modelo matemático de la salida de neuronal y:

$$y = f(\sum_{i=1} x_i w_i + b)$$

Dónde:

y = Salida del modelo neuronal

f = función de activación.

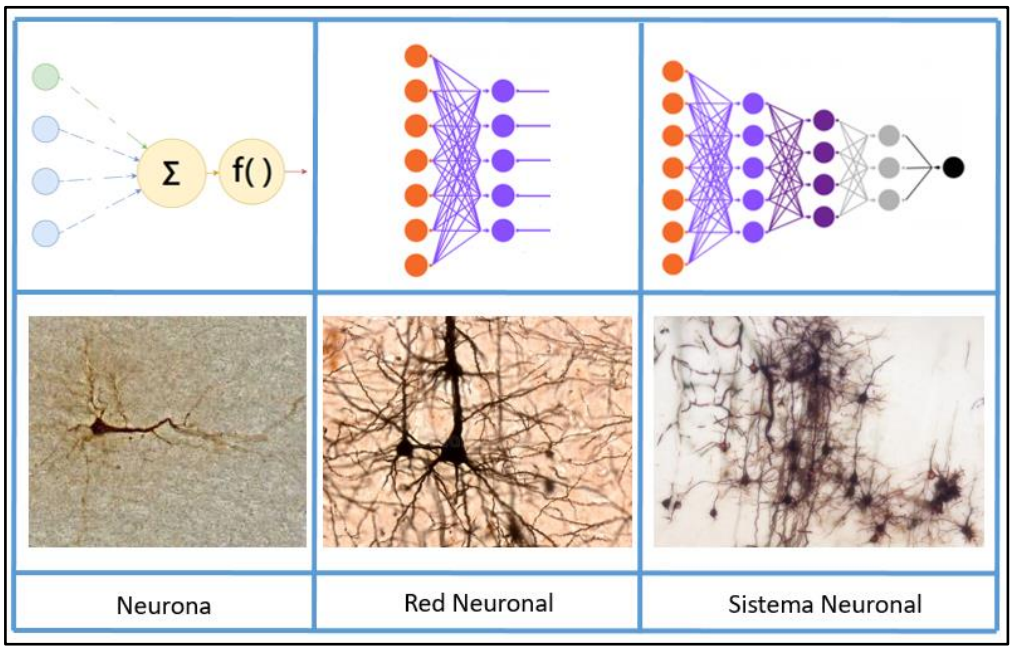
b = Sesgo de activación constante

w_i = Peso sinóptico

x_i = Vector entra

Figura 22.

Estructura Jerárquica de un sistema de redes neuronales artificiales y biológicas



Nota: Comparación sistema neuronal sistema biológico y artificial. Elaboración propia, realizado con Krita art.

2.4. Aplicaciones de las redes neuronales artificiales

Una vez desarrollado el modelo neuronal artificial con las bases teórico explicativo, es necesario profundizar en las características y el potencial que presentan las redes neuronales, los cambios que genera en la tecnología como las nuevas tecnologías latente que nacen a partir de ella.

En la tabla 1 se presentan aplicaciones de las redes neuronales en las diversas ramas tecnológicas.

Tabla 1.

Aplicaciones en la industria

Aeroespacial	Detectores y simulaciones de fallas de componentes de aeronaves, sistemas de control de aeronaves, pilotaje automático de alto rendimiento, simulaciones de trayectoria de vuelo.
Automotriz	Sistemas de guía mejorados, desarrollo de trenes de potencia, sensores virtuales, analizadores de actividad de garantía, autos completamente autónomos.
Electrónica	Análisis de fallas de chips, diseños de chips de circuitos, visión artificial, modelado no lineal, predicción de la secuencia de código, control de procesos y síntesis de voz.
Fabricación	análisis de diseño de productos químicos, modelado dinámico de sistemas de procesos químicos, control de procesos, diagnóstico de procesos y máquinas, diseño, análisis de productos, predicción de la calidad de productos, licitación de proyectos, planificación como gestión, análisis de calidad de chips informáticos, sistemas de inspección de calidad visual, análisis de calidad de soldadura.

Continuación de la tabla 1.

Mecánica	Monitoreo de condición, modelado de sistemas y control.
Robótica	Robots montacargas, controladores de manipuladores, control de trayectoria y sistemas de visión.
Telecomunicaciones	Control de red de cajeros automáticos, servicios-de-información automatizados, sistemas de procesamiento de pagos de clientes, compresión de datos, ecualizadores, gestión de fallas, reconocimiento de escritura a mano, diseño, gestión, enrutamiento y control de redes, monitoreo de redes, traducción en tiempo real del lenguaje hablado, reconocimiento de patrones (caras, objetos, huellas dactilares, análisis semántico, corrector ortográfico, procesamiento de señales y reconocimiento de voz).
Banca	Desgaste de tarjetas de crédito, evaluación de solicitudes de crédito y préstamo, evaluación de fraude, morosidad de préstamos.
Business Analytics	Modelado del comportamiento del cliente, segmentación de clientes, propensión al fraude, investigación de mercado, mezcla de mercado, estructura de mercado y modelos de deserción, incumplimiento, compra con renovaciones.
Defensa	Contraterrorismo, reconocimiento facial, extracción de funciones, supresión de ruido, discriminación de objetos, sensores, sonda, procesamiento de señales de imagen y radar, identificación de señal/imagen, seguimiento de objetivos dirección de armas.
Educación	*Software* de aprendizaje adaptativo, pronóstico dinámico, análisis y pronóstico del sistema educativo, modelado de desempeño, para los estudiantes como creación de perfiles de personalidad.

Continuación de la tabla 1.

Financiero	Calificaciones de bonos corporativos, análisis financiero corporativo, análisis de uso de líneas de crédito, predicción de precios de divisas, asesoramiento sobre préstamos, evaluación de hipotecas, tasación de bienes inmuebles y negociación de carteras.
Medico	Análisis de células cancerosas, análisis de ECG y EEG, asesoramiento sobre pruebas en salas de emergencia, reducción de gastos, mejora de la calidad para sistemas hospitalarios, optimización de procesos de trasplante, diseño de prótesis.
Valores	Calificación automática de bonos, análisis de mercado y sistemas de asesoramiento de negociación de acciones.
Transporte	Sistemas de enrutamiento, sistemas de diagnóstico de frenos de camiones y programación de vehículos.

Nota: Aplicaciones de la Inteligencia Artificial. Obtenido de Peréz, C. (2020). Unsupervised learning techniques: cluster analysis. examples with matlab. (p.55.)

Las redes neuronales se están utilizando en el sector eléctrico para resolver una variedad de problemas. Algunos de los usos más comunes de las redes neuronales en el sector eléctrico incluyen:

- Predicción de fallos: las redes neuronales se pueden entrenar con datos históricos sobre el rendimiento de los equipos eléctricos para predecir cuándo un equipo es más probable que falle. Esto puede ayudar a la industria eléctrica a planificar mejor la mantención preventiva y a reducir el tiempo de inactividad.

- Diagnóstico de equipos: las redes neuronales pueden analizar datos de sensores en tiempo real para detectar problemas en el rendimiento de los equipos eléctricos. Esto puede ayudar a los técnicos a identificar más rápidamente los problemas y a reducir el tiempo de reparación.

- Optimización del suministro de energía: las redes neuronales se pueden utilizar para analizar datos sobre el suministro y la demanda de energía para ayudar a optimizar el suministro como reducir los costos.

- Control de sistemas eléctricos: las redes neuronales se pueden utilizar para controlar sistemas eléctricos, como sistemas de generación, distribución y consumo de energía, para mejorar la eficiencia del sistema.

- En general, las redes neuronales están ayudando a la industria eléctrica a solucionar problemas complejos y a mejorar la eficiencia, la seguridad en la gestión de la energía.

2.5. Ventajas y desventajas de las Redes Neuronales Artificiales

Como toda la tecnología presenta ventajas como desventajas, la implementación de redes neuronales no es la excepción, una de las desventajas que presenta, están relacionadas al desarrollo computacional como dependencias de *hardware*, pero al ser una ciencia en desarrollo estas desventajas desaparecerán y sus ventajas aumentarán con el tiempo convirtiéndose así, en una parte indispensable de la vida.

2.5.1. Ventajas

Una de las ventajas de usar redes neuronales es las enormes cantidades de datos que pueden llegan a manejar y llegando a mejorar a medida que se ingresan más datos debido a la estructura convolucional que presentan algunos modelos.

Las redes neuronales aprenden a diferenciar patrones mediante entrenamientos, no necesitan la elaboración de modelos probabilísticos para resolver un problema, debido a que ellos generan sus propios pesos sinápticos en los enlaces en el aprendizaje.

Los sistemas neurológicos no aplican principios de circuitos lógicos o digitales.

El poder computacional disponible, permite procesar más datos, permitiendo el desarrollo de algoritmos con una ejecución mucho más rápida que antes, lo que hace posible utilizar más y más datos.

En la implementación de automatización de procesos permite desarrollar tareas repetitivas de manera autómata, sin intervención humana reduciendo el error en los procesos de clasificación y producción indetectables por el ojo humano.

El uso de redes neuronales en conjunto con visión artificial en implementación de procesos ya existentes es de fácil inserción además de ser no invasivas.

2.5.2. Desventajas

Como ya se explicó anteriormente las redes neuronales contiene capas ocultas donde se realiza gran parte de la predicción, pero todo este proceso hasta llegar a la solución es inexplicable hasta cierto punto debido al comportamiento similar a una caja negra, no existe explicación concreta del proceso. Siendo datos interpretables, esta es la razón por la que muchos bancos y ramas de la medicina no usan redes neuronales dado que se necesita una explicación precisa del porque se llegó a esa respuesta, se necesita una explicación concreta de los resultados obtenidos, para dar un veredicto.

Lo mismo ocurre con sitios web. Si un algoritmo de aprendizaje automático decidiera eliminar la cuenta de un usuario, al usuario en cuestión se le debería una explicación del porqué de la eliminación de su cuenta.

Además, no existe una regla específica para determinar la estructura de red adecuada, todo se logra mediante el ensayo experiencia y error. Con el tiempo las redes neuronales serán indispensables para la vida y procesos industriales esto llevara a la redistribución de personal en peor de los casos despidos masivos. Para realizar tareas en paralelo depende mucho de potencia de procesamiento. Las desventajas que se encuentran a nivel industrial es la falta de profesionales sobre el tema para la adopción de visón artificial por poca experiencia o desconocimiento y/o costos de implementación.

2.6. Mal ajuste Underfitting

El mal ajuste *underfitting* es un problema común en el aprendizaje automático que se produce cuando el modelo no puede capturar suficientemente la relación entre los datos de entrada y de salida. En otras

palabras, el modelo no está lo suficientemente ajustado a los datos de entrenamiento, por lo tanto, no puede hacer predicciones precisas en nuevos datos.

El mal ajuste ocurre cuando el modelo es demasiado simple para capturar la complejidad de los datos o cuando el conjunto de datos de entrenamiento es demasiado pequeño para entrenar el modelo de manera adecuada. En general, el mal ajuste se puede identificar si el error en el conjunto de datos de entrenamiento y el conjunto de datos de prueba son altos.

Hay varias formas de abordar el mal ajuste, como aumentar la complejidad del modelo (por ejemplo, aumentar el número de capas en una red neuronal), aumentar el tamaño del conjunto de datos de entrenamiento o reducir la regularización en el modelo. Sin embargo, es importante tener en cuenta que también es posible que el modelo esté sobre ajustando (*overfitting*) a los datos de entrenamiento, por lo tanto, hay que tener cuidado al ajustar los parámetros del modelo.

2.7. Sobre ajuste (*Overfitting*)

El Sobre ajuste o (*overfitting*), es un problema común en el aprendizaje automático que se produce cuando un modelo se ajusta demasiado a los datos de entrenamiento, pierde su capacidad de generalización para nuevos datos.

En otras palabras, el modelo aprende tanto de los datos de entrenamiento que memoriza el ruido o la variabilidad aleatoria presente en los datos de entrenamiento, lo que lleva a un rendimiento deficiente en los datos nuevos o de prueba. Esto se debe a que el modelo se vuelve demasiado

complejo y específico para los datos de entrenamiento, en lugar de aprender patrones generales que se pueden aplicar a nuevos datos.

Para evitar el *overfitting*, es importante utilizar técnicas como la validación cruzada, la regularización y la selección de características adecuadas para reducir la complejidad del modelo para mejorar su capacidad de generalización.

Figura 23.

Mal ajuste y sobre ajuste en una red neuronal

Nota: Que es overfitting y underfitting y cómo solucionarlo. Obtenido de Na, &Na. (2017). *El underfitting de los datos* (https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/), Consultado el 15 de abril de 2022. De dominio público.

2.8. Red neuronal convolucional

Las redes neuronales convolucionales o *Convolutional Neuronal Networks* CNN, nace en la necesidad de procesar tareas de clasificación efectivas e eficientes de imágenes con visión artificial, se aplican también en procesamiento de texto como en procesamiento de señales, pero como se ha mencionado su razón de existir es para clasificación de imágenes. Las redes neuronales convolucionales utilizan los principios del álgebra lineal, específicamente la multiplicación de matrices para identifica los patrones dentro de una imagen y operaciones matemáticas de convolución, realizando un tratamiento matricial de la señal de entrante con respecto de una función de transferencia que se encuentra a la salida de la neurona.

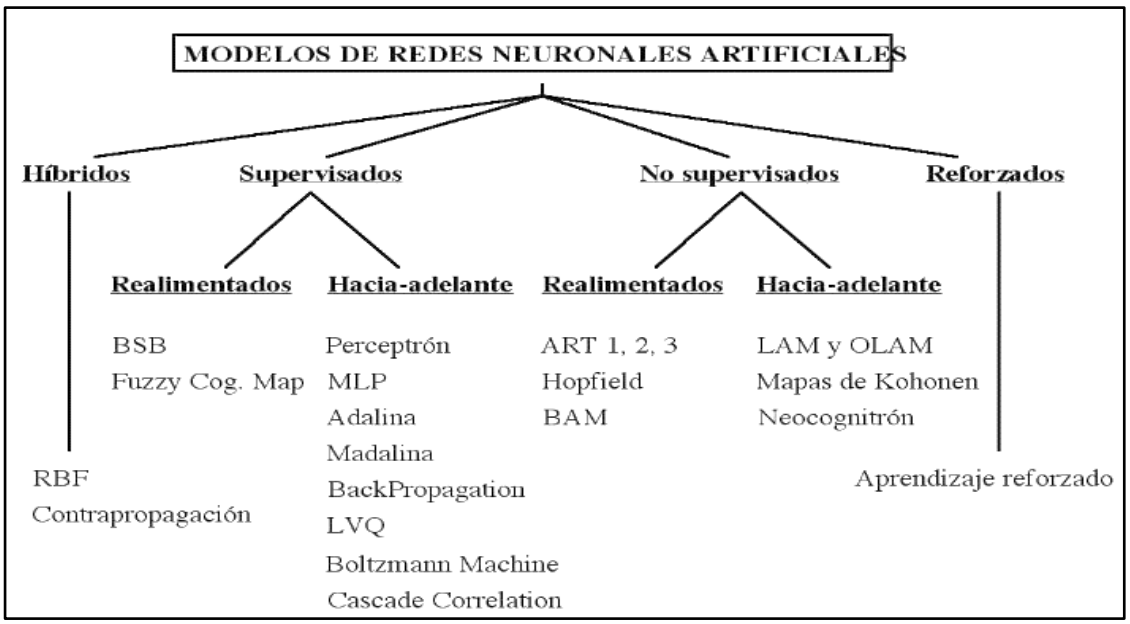
Este tipo de red neuronal se especializa en proceso de datos con topología *time-series data* e *imagen data* lo cual considera una dimensión para análisis de señales y dos dimensiones para análisis de imágenes. Estas redes neuronales convolucionales, para poder clasificar objetos o detectarlos en caso de procesamiento de imágenes, necesitan de un entrenamiento previo, para ellos se utilizan bases de datos con grandes cantidades de imágenes, con las cuales se entrenan para diferenciar o reconocer patrones inclusive patrones que nunca haya visto.

Si se requiere diferencia un gato de un perro, se entrena la red neuronal para poder deducir la especie, para ello se crean dos sets de datos, la primera con perros y la segunda como gatos, dentro de ese set se toma cierta cantidad de imágenes como datos de entrenamiento, el restante de imágenes se tomará para los datos de prueba esto se realiza para verificar la eficacia del modelo en cuanto a predicciones a esto se le denomina aprendizaje supervisado.

A lo largo de capítulo se ha explicado lo que son las neuronas artificiales (*Deep Learning*) su funcionamiento y como un conjunto de ellas emula hasta cierto punto la rede neuronal del cerebro humano, esto es la base de la Inteligencia Artificial, dentro de la inteligencia artificial esta lo que se denomina como aprendizaje automático o *maching learning*, la cual desarrolla las arquitecturas de redes neuronales para así dotar al modelo la posibilidad de aprenderá e identificar patrones que elaboran las predicciones, dotando de autonomía a la computadora, existen varios modelos neuronales, en la figura 24 se muestra la clasificación por el tipo de aprendizaje.

Figura 24.

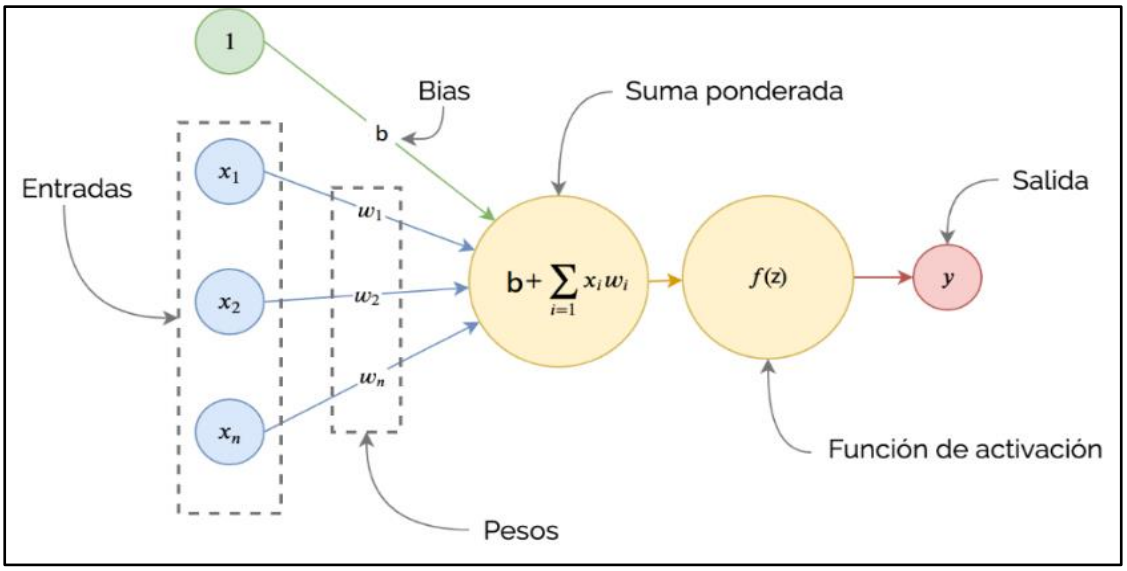
Clasificación de las Redes neuronales artificiales por tipo de aprendizaje



Nota: Clasificación de los modelos de redes neuronales artificiales. Adaptado de Martin del Brío, Serrano, C. (1995). Fundamentos de las redes neuronales artificiales: *hardware* y *software* p.118.

Figura 25.

Arquitectura de un perceptrón



Nota: Arquitectura de un perceptrón. Elaboración propia, realizado con AfterShot Pro 3 RAW Photo Editor.

2.9. Retropropagación (Backporpagation)

Es un algoritmo utilizado en el entrenamiento de redes neuronales artificiales supervisadas. Su objetivo es ajustar los pesos de las conexiones de la red para minimizar el error de predicción en los datos de entrenamiento.

El algoritmo de backpropagation trabaja en dos fases. En la primera fase, conocida como fase de propagación hacia adelante, se propaga la entrada de la red a través de las capas ocultas hasta la capa de salida, calculando la predicción de la red para cada instancia de entrenamiento.

En la segunda fase, conocida como fase de retropropagación del error, se calcula el error de predicción de la red para cada instancia de entrenamiento y se propaga hacia atrás a través de la red. Durante este proceso, se ajustan los pesos de las conexiones en cada capa para reducir el error de predicción. Este ajuste se realiza utilizando el gradiente descendente, un algoritmo de optimización que utiliza la derivada de la función de error con respecto a los pesos de las conexiones para actualizar los pesos de la red en cada iteración.

El algoritmo de backpropagation se repite durante varias épocas de entrenamiento hasta que se alcanza un punto de convergencia en el que el error de predicción de la red se ha minimizado. En general, el backpropagation es un algoritmo eficaz para entrenar redes neuronales artificiales y se ha utilizado en una amplia variedad de aplicaciones de aprendizaje automático.

2.9.1. Retropropagación en Redes Neuronales Convolucionales (ConvNet)

La retropropagación en Redes Neuronales Convolucionales sigue el mismo principio básico que en las redes neuronales artificiales estándar. La diferencia es que, en las ConvNet, las capas ocultas están compuestas por capas convolucionales y capas de *pooling*, en lugar de capas completamente conectadas.

En las ConvNet, la fase de propagación hacia adelante se realiza mediante la convolución de los filtros de la capa convolucional con la entrada de la red, seguida de la aplicación de la función de activación. Luego, la salida de la capa convolucional se pasa a través de la capa de *pooling*, que reduce la resolución espacial de la salida.

Durante la fase de retropropagación del error en una ConvNet, el error se propaga hacia atrás a través de la red utilizando los mismos principios que en las redes neuronales estándar. Sin embargo, debido a la estructura convolucional de la red, los pesos que se actualizan en cada capa también están organizados en filtros, en lugar de conexiones completamente conectadas.

La retropropagación en las CNN sigue el principio general del gradiente descendente, donde se utilizan las derivadas parciales de la función de error con respecto a los pesos de la red para actualizarlos y minimizar el error. Este proceso se repite durante varias épocas de entrenamiento hasta que se alcanza un punto de convergencia en el que se ha minimizado el error de predicción de la red.

Las redes neuronales convolucionales al ser de tipo de aprendizaje supervisado utilizan la propagación hacia delante, pero esta propagación se presenta solamente cuando la red neuronal ya se encuentra entrenada, durante el entrenamiento realiza propagación hacia atrás con el objetivo de ajustar el peso sinóptico y hacia delante como se ha mencionado anteriormente para calcular el vector de salida con respecto a la entrada.

Para comenzar con el desarrollo de un caso de clasificación, hay que entender los principios básicos de los procesos de visión humana y la visión por computadora, si al ojo humano se le muestra una imagen, si se observa un perro, el ojo humano realizará un escaneo detectando patrones conocidos, he inmediatamente se percibirá que es un perro, pero ¿cómo es que lo hace?

¿Detecta la forma, el color, el olor o todas esas características? Entonces, ¿cualquier cosa que tenga forma, color u olor sería un perro? Como

es que el cerebro interpreta cada objeto que ve, por ejemplo: los dulces como las flores tienen forma, color y olor. En otras palabras, ¿cómo es capaz el cerebro humano de procesar la imagen de un perro y entenderla como un perro o diferenciarlo de otro objeto o ser vivo?

La visión por computadora, por lo tanto, no es simplemente duplicar la visión humana, sino también la capacidad de procesar e interpretar la imagen. Para ello primero se debe entender cómo el ojo humano captura y procesa las imágenes. La visión es uno de los sentidos humanos más avanzados donde los ojos son responsables de la visión como de la comunicación no verbal.

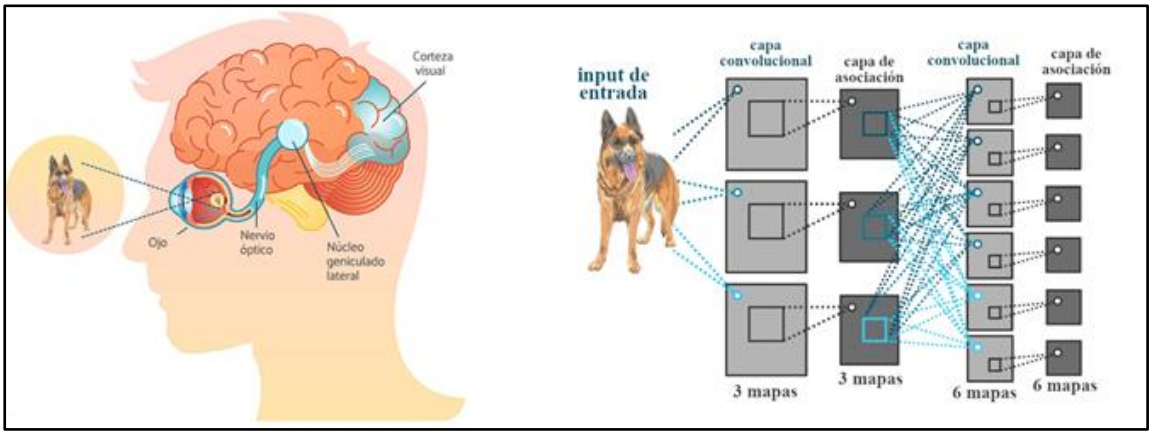
El ojo humano usa una sola lente para enfocar las imágenes en una membrana sensible a la luz llamada retina la imagen obtenida se envía al cerebro para su procesamiento, el cerebro humano cuando percibe el mundo que lo rodea hace una reconstrucción de la escena por medio de procesamientos (estímulos), pero también trata de completar esta escena, con ayuda de lo que se ha aprendido en el pasado para dar una interpretación a lo que se está percibiendo en el presente, abstrayendo patrones conocidos, pero también ha aprendido las diferentes sensaciones de los objetos por medio de las texturas, el olor o el sabor, esto debido a los 5 sentidos que caracteriza a los humanos animales.

En la visión por computadora funciona bajo el mismo principio, pero la visión por computadora carece de los sentidos de tacto de olfato y de gusto, por lo cual no puede reconocer texturas u olores ni sabores, pero si posee la capacidad de ver por medio de una cámara, como así también aprende a oír, pero no a escuchar por medio de un micrófono,

Entonces para poder reconocer patrones y contrastes, se aplican filtros de una o más capas convolucionales a la imagen de entrada, buscando diferentes patrones conocidos como no conocidos para así dar una respuesta de lo que se está observando, ahora si se requiere escuchar se aplican tratamientos de con Laplace o con transformada de Fourier.

Figura 26.

Procesamiento de imágenes por el ojo humano y rede neuronal convolucional



Nota: Comparación del ojo humano y red neuronal convolucional. Elaboración propia, realizado en AfterShot Pro 3 RAW Photo Editor.

Cuando se recibe una imagen se refleja en la retina en las cuales existen unas terminales nerviosas que reaccionan a diferentes niveles de luz o intensidades entre puntos adyacentes, estas provocan reacciones debido a derivaciones de luminosidad en la entrada y se ven reflejadas en la retina, posteriormente es enviada al tálamo donde realiza una filtración de la información para luego pasar a la corteza visual también llamada *córtex*.

La capacidad del *córtex* visual, se representan como un procesamiento en cascada, en donde identifica patrones básicos generales en posteriores capas se combina para generar patrones más complejos. Esto es la fuente de inspiración para las redes neuronales convolucionales, y las cuales se conforman por tres tipos principales de capas, que son:

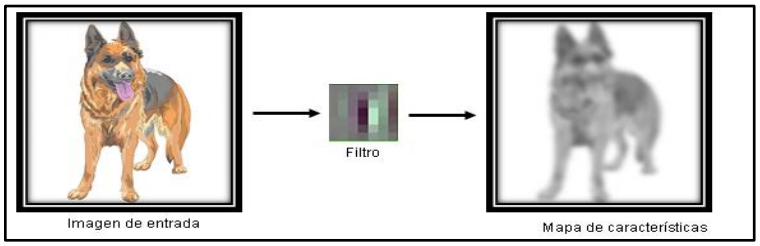
- Capa convolucional
- Capa de agrupación
- Capa totalmente conectada (*full conect*)

2.9.2. Capa convolucional

La capa convolucional hace una reducción de dimensionalidad de las características de la imagen de entrada junto a una extracción de características a través de operaciones de convolución los cuales son matrices que crean filtros denominados como Kernel que tiene un campo receptivo que captura la información local y detallada de la imagen, con lo cual extrae líneas o formas simples. Esto permite a la red aprender patrones con características esenciales en los datos.

Figura 27.

Capa convolucional de desenfoque aplicada a la imagen de un perro

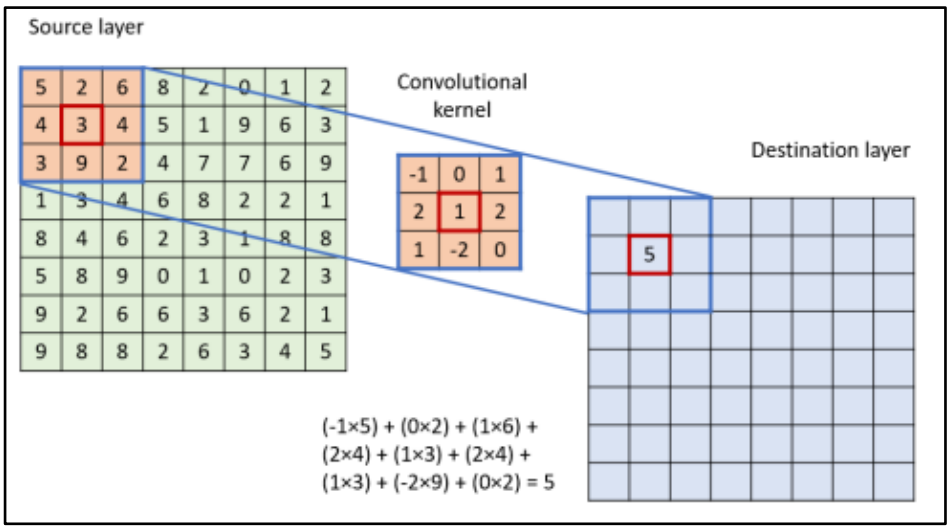


Nota: Capa convolucional. Elaboración propia, realizado con Python 3.9.6.

El kernel es una matriz pequeña de números que se desliza sobre los datos de entrada en una capa convolucional de una red neuronal. Cada aplicación del kernel es una operación matemática que se realiza sobre una porción de los datos de entrada, produciendo una nueva representación de estos. La idea detrás del uso de un kernel es que al realizar estas operaciones repetidamente en diferentes porciones de los datos, se pueden extraer características relevantes reduciendo la dimensión de los datos de entrada. Los valores en el kernel son escalares y aprendidos por la red neuronal durante el proceso de entrenamiento.

Figura 28.

Kernel convolucional en procesamiento de imágenes



Nota: Best Practice Guide - Deep Learning. Obtenido de Researchgate (2019). *Deep Learning* (https://prace-ri.eu/wp-content/uploads/Best-Practice-Guide_Deep-Learning.pdf), consultado el 30 de abril de 2022. De dominio público.

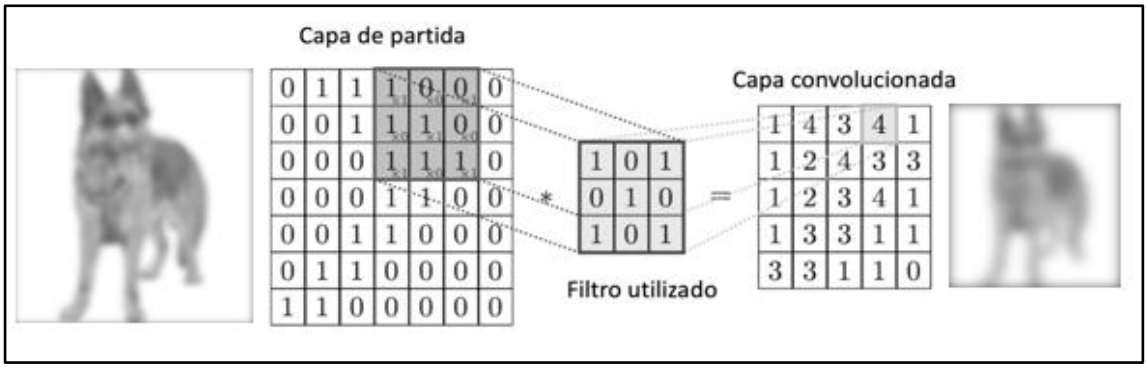
2.9.3. Capa de agrupación

La capa de agrupación se utiliza para reducir la resolución espacial de los datos que fluyen a través de la red. Esto se logra mediante la agregación de los valores de los pixeles adyacentes en una porción de los datos de entrada, produciendo una representación más compacta como resumida de la información original. Hay dos tipos de agrupación comúnmente utilizados en las redes neuronales convolucionales: la agrupación por promedio (también conocida como agrupación por pooling) y la agrupación por máximo.

La agrupación por promedio toma el promedio de los valores de los pixeles en una porción de los datos, mientras que la agrupación por máximo toma el valor máximo. La agrupación ayuda a la red a ser más resistente a las transformaciones locales en los datos de entrada y a mejorar la capacidad de la red para detectar características esenciales en los datos.

Figura 29.

Capa de agrupación aplicada a la capa convolucional



Nota: Capa de agrupación o subsampling pooling. Elaboración propia, realizado con Python 3.9.6.

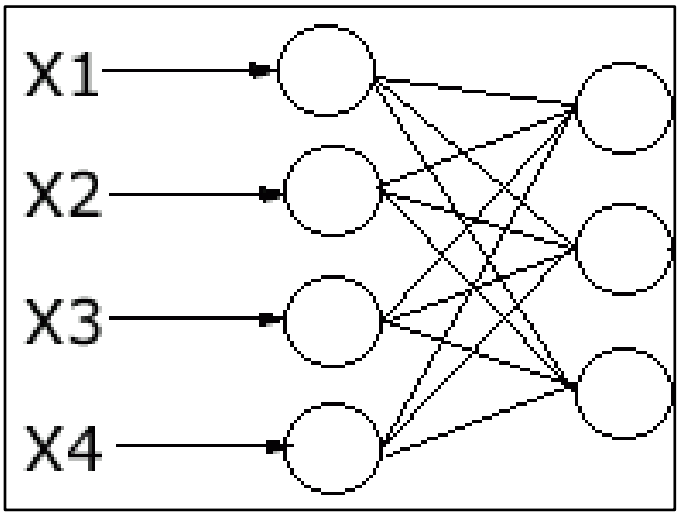
2.9.4. Capa totalmente conectada (FC)

Una capa completamente conectada en una red neuronal convolucional es una capa en la que cada neurona en la capa está conectada a todas las neuronas de la capa anterior. Esta capa se utiliza comúnmente como capa final en una red neuronal convolucional, después de que los datos han pasado a través de las capas convolucionales que a su vez pasa por las de agrupación.

La capa completamente conectada realiza una combinación lineal de los valores resultantes de las capas anteriores, luego se aplica una función de activación no lineal. La salida de esta capa es la predicción de la red neuronal. La capa completamente conectada es importante porque permite a la red aprender relaciones complejas no lineales entre las características extraídas en las capas anteriores y la salida deseada.

Figura 30.

Representación gráfica de una capa totalmente conectada

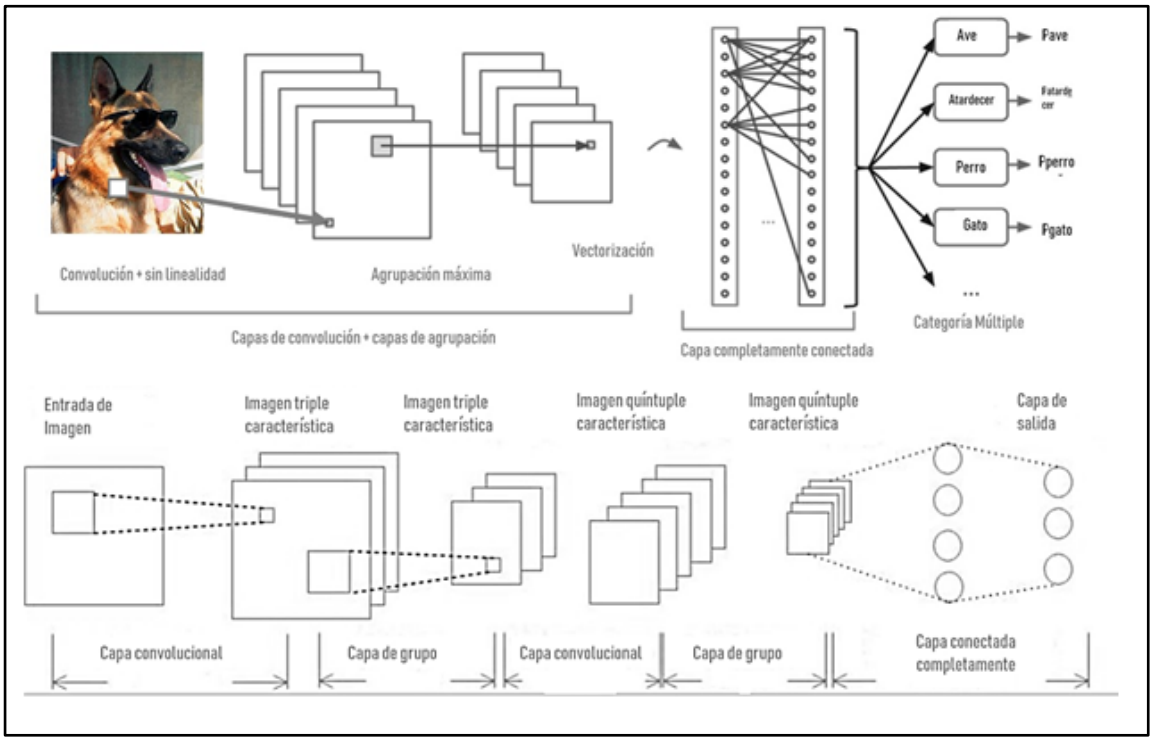


Nota: Capa totalmente conectada. Elaboración propia, realizado con Paint.Net.

Por los atributos que presenta una red convolucional la hacen el algoritmo ideal para reconocimiento de imágenes, al manejar grandes cantidades de pesos hace una comparación con el patrón aprendido con respecto a la imagen de entrada, La convolución se utiliza para extraer características relevantes de los datos de entrada donde la forma en que se realiza la convolución puede variar dependiendo del tipo de red y del problema que se está tratando de resolver.

Figura 31.

Arquitectura de una Red Neuronal Convolucional (CovNet)



Nota: Arquitectura de una Red Neuronal. Elaboración propia, realizado con Adobe Lightroom.

3. VISIÓN ARTIFICIAL EN PROCESOS DE PRODUCCIÓN Y CONTROL DE CALIDAD

3.1. El COVID-19 como impulsor de inteligencia artificial en procesos de producción

La tecnología es indispensable en cualquier sector, se ha vuelto una palanca para el cambio en cuanto mejoras en el mundo y durante la pandemia no ha sido la excepción permitiendo una gran cantidad de desarrollos tecnológicos en el campo de la inteligencia artificial (IA). El COVID-19 vino a replantear la forma de los negocios en las empresas.

Debido a la reciente pandemia algunas empresas adoptaron la modalidad home office con el objetivo de mantener las operaciones de la empresa y la salud de sus trabajadores, esta nueva forma beneficiaba a algunas áreas, en específico las administrativas, pero ¿qué sucede con las áreas operativas? ¿Qué sucedía con el personal responsable del control de calidad o de las personas que tenían contacto directo con el producto si estos son bienes de consumo? más aún si una persona se contagiaba, no podían exponer al personal como los productos a proliferaciones, las empresas necesitaban ser resilientes ante la situación crítica con ello venia la necesidad de actuar con velocidad, para implementar soluciones para no caer al borde del colapso debiendo cumplir con los productos o servicios de forma rápida, más aún si son artículos de primera necesidad.

A lo mencionado anteriormente se suma la situación política debida que ahora las empresas se enfrentan a decisiones que puedan tomar los gobiernos

como medida de acción y prevención ante la pandemia. Esta necesidad obligó a las empresas a implementar tecnologías para minimizar riesgos, una de ellas fue la inteligencia artificial.

La implementación de inteligencia artificial en las cadenas de suministro automatizo la toma de decisiones, al realizar tareas repetitivas y por medio de aprendizaje automático (ML), proporcionando al sistema la capacidad de aprender el proceso a partir de la experiencia. Esto significó un cambio radical, debido a que con esta nueva implementación se pudo trabajar hasta en cierta medida mejor que con seres humanos porque estos no presentaban fatiga a lo largo de la jornada de trabajo pudiendo trabajar 24 hrs del día sin perder precisión eliminando la ineficiencia e ineficacia que pueden presentar el ser humano con respecto a ciertas actividades.

McKinsey & Company, compartido un análisis en la cual indica que la implementación de la inteligencia artificial presento una disminución del 61 % de los costos de fabricación y un 53 % en aumento de los ingresos como resultado directo de la introducción de Inteligencia artificial (IA) en la cadena de suministro. Además, más de un tercio sugirió un rebote total de ingresos de más del 5 %. Algunas de las áreas de alto impacto en la gestión de la cadena de suministro incluyen planificación, programación, previsión, análisis de gastos, optimización de la red logística, inteligencia artificial (IA) en la cadena de-suministro (Pakmail, 2022).

Esto demuestra que el camino de la inteligencia artificial puede mejorar una gran variedad de indicadores de desempeño, la tecnología cada vez es más empoderada, en la actualidad se puede observar desde automóviles autónomos, traductores automáticos, hasta *chatbot* para la atención al cliente y todo esto funciona con implementación de la inteligencia artificial.

3.1.1. Uso de la inteligencia artificial en Guatemala

El control de calidad en los procesos de fabricación es fundamental para garantizar que los consumidores reciban productos con la funcionalidad, confiabilidades adecuadas. Los productos defectuosos pueden generar costos adicionales para el fabricante y dañar la confianza en una marca.

En Guatemala la agroindustria es uno de los sectores con más crecimiento del país, constituyendo una actividad de manufactura importante por sus aportaciones al producto y al empleo, donde la logística en conjunto con el apoyo a las cadenas de suministro es fuerte, con alta disponibilidad de materia prima debido a la diversidad de los productos agrícolas cultivados en el país. Pero también existen limitaciones evidenciadas en dos grandes grupos, uno de ellos con proyección exportadora la otra conformada por pequeños productores que abastecen el mercado interno; es la utilización de poca tecnología, lo que los hace vulnerables a la eficiencia precisión como automatización.

El implementar tecnología de inteligencia artificial en la agroindustria podría traer grandes beneficios, la implementación de robótica en la agricultura ayudaría en trabajo en el campo como, por ejemplo: monitoreo de suelos, controlar plagas indicador de fertilizantes como estimación del rendimiento de las cosechas, los autos autónomos ya son un hecho y esa misma tecnología implementarla en tractores para automatizar la labranza de tierras. En general la implementación de inteligencia artificial traería mejoras no solo en la agroindustria, sino que en áreas generales de producción importación sin olvidar la distribución, minimizando los riesgos, aumentando el nivel de confiabilidad, transparencia, capacidad de respuesta de las agroindustrias para

evitar que las importaciones de alto riesgo ingresen a los mercados nacionales e internacionales.

El propósito de implantar este tipo de tecnología considera así también, eliminar tareas repetitivas como laboriosas, automatizando los procesos artesanales de inspección y control de calidad, pudiendo trabajar 24 horas al día, 7 días a la semana sin fatiga. Si un ser humano inspecciona productos, el producto en 10 inspecciones estará sujeto a cambios en muchos aspectos. Lo que en día de trabajo podría parecer defectuoso, al día siguiente podría estar bien, lo cual está toma de decisiones está sujeto a cambios de la psicología humana.

Viéndose afectados por factores como: una mala alimentación, un descanso nocturno no apropiado, cambios de humor o por trabajo constantemente repetitivo. Si el producto en inspección requiere alta precisión al detalle, como podría ser el etiquetado de código de barras, fugas de productos envasados, entre otros. Si es de manera artesanal se inspecciona el producto, pero al realizar estas tareas una persona, como se ha mencionado eventualmente durante la jornada tendría fatiga cognitiva, lo que ocasionaría una baja eficiencia en la clasificación de productividad.

3.1.2. Viabilidad de implementar procesos autónomos en la industria de Guatemala

Según Global de Innovación, Guatemala ocupa el lugar 110, publicado en el año 2022, lo cual demuestra un nivel bajo de competitividad donde la innovación puede ser transcendental para fomentar el crecimiento económico y acceder a mayores oportunidades, por lo cual la inteligencia artificial se

presenta como una de las tantas oportunidades que existen para el desarrollo de Guatemala.

La presente tesis tiene como objetivo que la población general como los empresarios guatemaltecos descubran las oportunidades que existen a nivel industrial, nivel académico y que se desarrollen investigaciones que permitan aprovechar la Inteligencia Artificial de tal manera que sea una ayuda para crear un país más competitivo, las oportunidades son muy amplias aunque para Guatemala hay un largo camino por recorrer se puede lograr si, si se estructura como lo hacen México desarrollando proyectos de Inteligencia artificial y fomentando a la investigación, la cual designan un presupuesto una política en la que definen el marco integral de trabajo empresarial, gobierno-academia.

A nivel de empresarial capacitar e identificar individuos con potencial para los roles relacionados con Inteligencia Artificial, comenzar a construir datos, explorar nuevas habilidades requeridas en IA y mantener este círculo virtuoso para ir perfeccionando las capacidades (MIT SMR Connections, 2020).

A nivel general como gobierno: hacer un plan de acción, establecer una organización que a nivel nacional tenga presupuesto que a su vez la responsabilidad de desarrollar las aplicaciones de IA, crear centros de desarrollo de IA, crear y ejecutar un plan de capacitación en desarrollo de habilidades de IA en la fuerza laboral, alianzas o sociedades entre el gobierno con el sector privado.

3.2. Inspección visual

Todos los procesos de producción son evaluados por la calidad que presenta el producto final y la percepción del consumidor ante el producto

considerando si este es de mayor o menor calidad en función de los factores asociados al producto. La calidad se trata de cuidar midiendo a través de variables cuantificables establecidos por instrumentos de medición en las distintas etapas de producción como consecuencia de un buen control de producción aumentar las ventajas competitivas de la empresa.

La inspección visual humana es la forma más sencilla y antigua, posee la capacidad de adaptarse a situaciones nuevas rápidamente ha demostrado ser un método efectivo para la detección de anomalías a nivel de superficie, pero como se sabe los humanos poseen limitantes, una de esas es la fatiga, la monotonía del trabajo, por lo cual hace que el trabajo de inspección sea inconsistente con tendencia a poco seguro cuando se presenten estos factores.

La inspección de un objeto o producto manufacturado involucra variables de comparar dimensiones, características (radios, longitudes, empaquetado y envasado). Medir las interrelaciones que guardar como podría ser ángulos entre planos o distancia entre centro de gravedad.

Figura 32.

Control de calidad humana a nivel industrial

Nota: La agroindustria citrícola: consideraciones. Obtenido de citricos.com (2022). *La agroindustria citrícola: consideraciones*. (https://citricos.com/el-proceso-de-la-agroindustria-en-los-citricos), Consultado el 03 de mayo de 2022. Dominio público.

En la actualidad las empresas del mundo moderno manejan *software* que aprovecha la inteligencia artificial (IA) que no solo detecta defectos o fallas, sino enseñan a un sistema a leer imágenes y determinar los estándares a cumplir con base en ejemplos previos, con lo cual ahorran tiempo y mejoran la precisión, estas nuevas técnicas de inspección apunta a ser 100 % automático los controles de calidad en un futuro.

3.2.1. Inspección visual automática

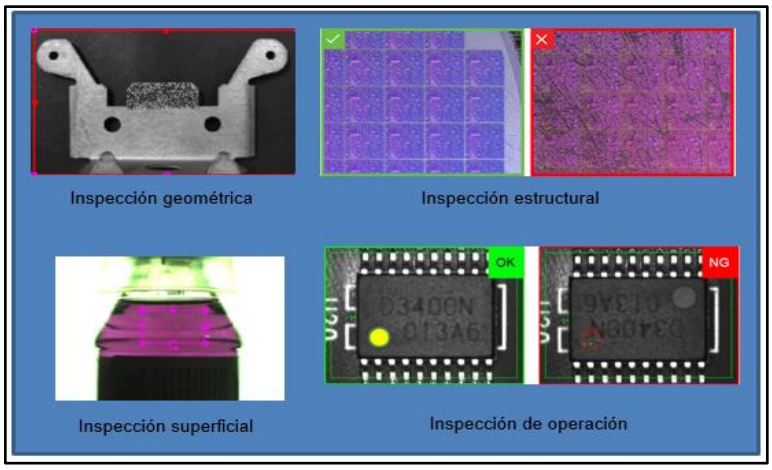
Con el avance tecnológico fueron naciendo nuevos métodos de inspección una de ellas fue la inspección automática la cual consiste en la recopilación de imágenes y videos en tiempo real con ayuda de protocolos de comunicación con el uso del internet permiten la monitorización del producto desde una ubicación remoto.

A la velocidad que avanza la tecnología no era de extrañarse una pronta evolución de la inspección automática a la cual se le denomina inspección visual inteligente, realizando las tareas aún más rápidas con una precisión del 99.9 %, presentado una gran rentabilidad en diversos entornos industriales, puesto que reduce costos de producción, es un sistema no invasor la cual permite mantener al personal alejado de entornos peligrosos y lugares confinados.

Esto es logrado juntamente con la integración de inteligencia artificial (IA) en los sistemas de inspección, permitiendo la evaluación de los sistemas como de sus componentes sin cambios permanentes.

Figura 33.

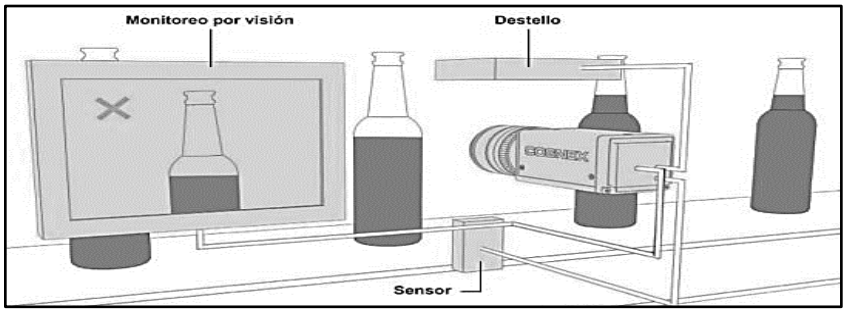
Clasificación de los diferentes tipos de inspección a nivel industrial



Nota: Clasificación de los diferentes tipos de inspección a nivel industrial. Elaboración propia, realizado con Cognex simulation.

Figura 34.

Esquema de un sistema de visión artificial



Nota: Componentes de la visión artificial. Obtenido de Grupo Grupo bcnvision (2017). *Componentes de la visión artificial para aplicaciones.* (https://www.bcnvision.es/blog-vision-artificial/componentes-vision-artificial/), Consultado el 07 de mayo de 2022.De dominio público.

Los componentes básicos que conforma un sistema de visión arterial son:

- Iluminación especial
- Cámara y lentes ópticos
- Sensores de imagen y actuadores
- Protocolo de comunicación
- Procesadores de visión

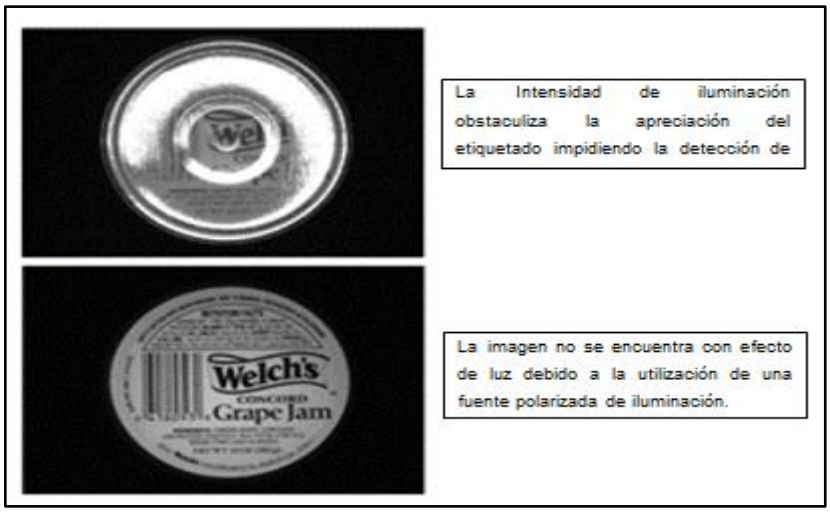
3.2.1.1. Iluminación especial

La iluminación juega un papel importante en el sistema de visión, una correcta iluminación depende el éxito de la aplicación. Las cámaras capturan la luz reflejada en la superficie u objeto obteniendo la información lumínica necesaria para el procesamiento para así controlar la forma que la cámara ve el objeto en cuestión, pero esto no significa que cualquier tipo de iluminación se utilizar en la inspección de un objeto ovalado, cuadrado o alguna pieza cromada, para ello se analiza y estudia la situación, tomando en cuenta si la implementación de la cámara será a color o monocromática, si la aplicación será de alta velocidad o baja velocidad, el campo de visión a iluminar, la superficie presentara reflejos, la características de inspección a resaltar, la duración de la iluminación, requisitos mecánicos, entre otros.

Las imágenes presentadas en la figura 35 muestran lo importante del efecto de la luz en el entorno. En la imagen superior, debido a la intensidad del brillo no se apreciaría si tuviera un defecto de impresión y lo cual obstaculizaría el proceso de control. En la imagen inferior, no se encuentra el efecto de la luz en la imagen adquirida debido a la utilización de una fuente polarizada de iluminación.

Figura 35.

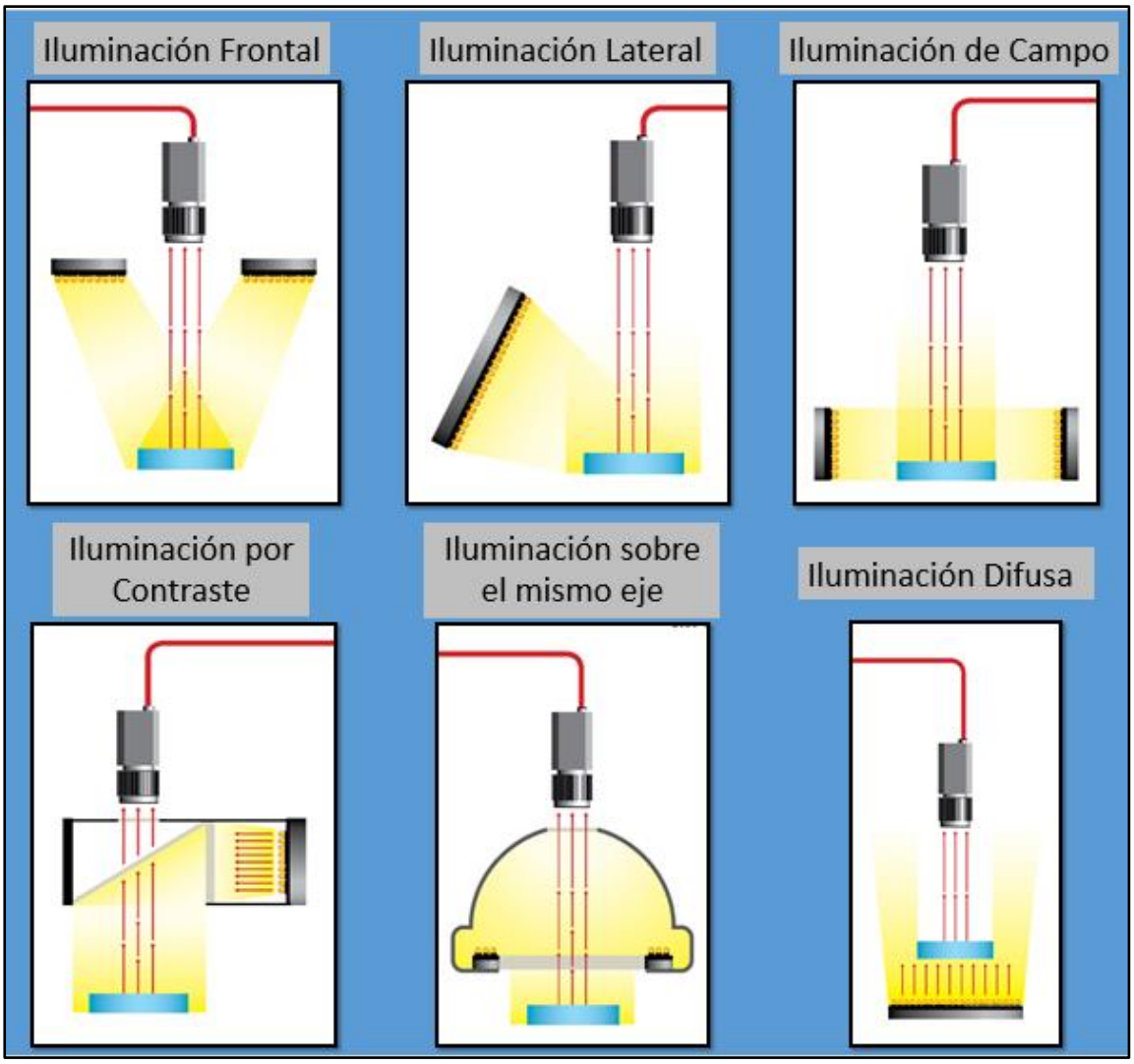
Efecto de la iluminación en la adquisición de datos



Nota: Sistemas de iluminación para aplicaciones de visión artificial [Introducción]. Obtenido de Grupo Grupo bcnvision (2017). *Efecto de la iluminación en la adquisición de datos*. (https://www.bcnvision.es/blog-vision-artificial/iluminacion-vision-artificial/), Consultado el 12 de mayo de 2022.De dominio público.

Figura 36.

Técnicas de iluminación para aplicación de visión artificial



Nota: Técnicas de iluminación para aplicación de visión artificial. Elaboración propia, realizado con Adobe Photoshop.

3.2.1.2. Tipos de óptica para la visión artificial

Muchas veces la óptica se desprecia debido a que se considera meramente como un accesorio más de la cámara, lo cual es equivocado pensar, porque forma una parte esencial del sistema de visión, la óptica transmite la luz a la cámara de manera controlada con un alto contraste y la menor distorsión geométrica posible para conseguir la mejor imagen que permita realizar mediciones correctas.

Existen factores de importancia para la óptica de la visión artificial y se deben considerar parámetros que permitan el muestro de imágenes adecuadas.

Figura 37.

Variables de influencia en calidad de imagen

Nota: Variables de influencia. Elaboración propia, realizado con Paint.net.

Por lo tanto, las lentes son importantes, las cámaras actuales realizan una disminución del tamaño de píxel. Por ende, la elección correcta de la lente juega un papel importante, para que los sistemas ópticos realicen mediciones precisas únicamente permitiendo el paso de los rayos paralelos y evitar la distorsión.

Las cámaras telecéntricos eliminan la limitación del sistema con lentes convencionales, dado que con una lente convencional si el objeto se mueve dentro del campo de visión, existiría un cambio de aumento o disminución del objeto para calibrar este error se debía usar otra cámara adicional, por lo tanto la lente telecéntrica proporciona imágenes superiores con baja distorsión y no presenta problema con los cambios del objeto dentro el campo de visión, pero estas lentes también presenta desventaja siendo está el diámetro de la lente tiende a ser proporcional a la pieza a visualizar lo cual las vuelve de mayor costo a comparación de ópticas estándares.

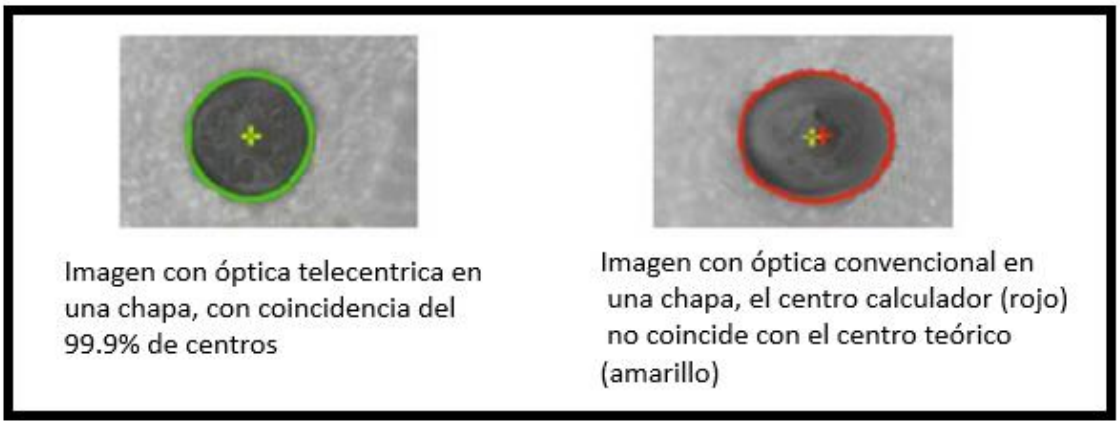
Figura 38.

Cámara telecéntrica vs. cámara convencional

Nota: Comparación de cámara telecéntrica vs. cámara convencional. Elaboración propia, realizado con Paint.net.

Figura 39.

Inspección con cámara convencional y telecéntrica



Nota: Visión artificial en procesos industriales. Obtenido de Bruned, M. (2008). *Measure control Cámaras de visión artificial* (https://measurecontrol.com/vision-artificial-en-procesos-industriales-parte-ii/) Consultado el 18 de mayo de 2022. De dominio público.

4. PROBLEMA

4.1. Antecedentes

La producción de cítricos se ha convertido en una actividad importante en la economía socioeconómica del mundo y esta importancia radica en el valor nutricional, medicinal como la cantidad de subproductos que de estos se deriva.

Analizando el sector de los cítricos a nivel nacional, especialmente la de Limón ha constituido en mediano y largo plazo una fuente importante de ingresos, en el 2012, Guatemala exportó 154,3 toneladas de limones a la Unión Europea, muy por detrás de otros países como Honduras, que exportó más de 446 toneladas en el mismo período. Los principales destinos de las exportaciones de Guatemala han sido Francia con 94,7 Tm y Holanda con 59,6 Tm (Ministerio de Agricultura, Ganadería y Alimentación, 2014).

En Guatemala las variedades de cítricos son de gran importancia, el cultivo Limón persa, es decir, Citrus latifolia Tanaka es de las más productivas del país en los departamentos Escuintla, representando el 32 % de la superficie total del territorio, Suchitepéquez el 15 %, el resto el 53 %, en sectores de Santa Rosa, Alta Verapaz, Petén, Sacatepéquez, Chimaltenango, Jutiapa, El Progreso, Retalhuleu y Zacapa (Cabrera, 2005).

4.2. Planteamiento del problema

El control de calidad en los procesos es fundamental porque garantiza a los consumidores el producto, el no optimizar de manera eficiente las etapas de

calidad, afecta la productividad con productos inconsistentes. Si bien el limón no es un pasaboca común, si es una fruta muy utilizada, en cocina, aromaterapia, medicina y dermatología.

Una mala optimización de control de calidad y pruebas de madurez aumenta un impacto de los rechazos que a su vez se ve reflejado en las renegociaciones de precio, afectando a toda la cadena de valor que esta implica, desde los agricultores, recolectores, almacenes, empresas comercializadoras hasta consumidores, pues el producto en cuestión no cumple con las normas de calidad requeridas.

Entre los defectos más comunes y con mayor probabilidad de afectar la calidad de la cadena de suministro de limones se tiene:

- Daño por rociado, produce quemaduras y dañan las cascara por lo cual es importante pensar en los aditivos nutritivos o pesticidas que se deben y utilizar al cultivo.
- Las Magulladuras suceden en cualquier momento, ya sea por la manipulación incorrecta durante la recolecta, almacenamiento o transporte reduciendo el coste del contenedor.
- El mantener por periodos prolongados congelados el cítrico.
- Picaduras producen lesiones en la casaca de los limones y es generada por almacenamiento a temperaturas altas.
- Mosquito de hongos es debido al exceso riego en el cultivo de limón, es uno de los principales rechazos de cultivos de limón y cítricos.
- El moho Fumagina, es resultado de las secreciones de ligamaza que produce los insectos entre los cuales se encuentran: cochinillas, produciendo moho en las hojas y partes del limonar.
- Los ácaros en los cítricos afectan las hojas del limonar tornando a los

limones con una apariencia plateada o con manchas.

También existen criterios de selección que no basta que esté libre de contaminantes superficiales y se valúan con base en:

- Diámetro
- Relación azúcar/ácido
- Figura
- Textura

Y a su vez se evalúa por su color, el cual se tiene cuatro categorías:

  - Extra
  - Límite permitido primera
  - Límite permitido segunda
  - Desverdizar

Esto demuestra la cálida del producto final depende tanto de la forma de cultivar como de su optimización de calidad.

4.3. Justificación

En el *Innovation & Technology* Expo realizada en el 2019, se apoyó la impulsión de aplicaciones de tecnología de punta en Guatemala, paralelamente se incentivó a crear interés de inversión en aplicaciones y desarrollos tecnológicos locales.

“Actualmente se tiene la convicción que la tecnología e innovación son elementos indispensables para mejorar la Industria nacional y por ende incidir

en la generación de empleos, el crecimiento económico como desarrollo social del país” (Zepeda, 2019, p. 61).

La visión artificial es fundamental en sistemas de automatización en la Industria 4.0. Los avances en capacidades de análisis de datos y el gran volumen de *datasets* para *maching learning* que actualmente son accesibles para implementación en visión artificial son esenciales, lo cual permite una intervención rápida como eficaz en la fábrica de la Industria 4.0.

La Inteligencia artificial puede ayudar a los productores a mantenerse competitivos en un mercado cada vez más exigente, mejorando la calidad y la consistencia de sus productos, reduciendo los costos en la producción, ya sea a través de la automatización de tareas repetitivas o la identificación temprana de problemas de calidad. Garantizando que los productos cumplan con los estándares como regulaciones aplicables. Es una buena forma de fomentar el desarrollo económico en Guatemala

4.4. Alcance

El propósito de este proyecto es el control automático y análisis de producción de limones eliminando el trabajo manual de calidad implementando inteligencia artificial para la detección de contaminantes en la superficie de la fruta cítrica (limón), como podrían: ser hongos, coloración de la piel, pudriciones, magulladuras, diámetros, colores del limón, esta propuesta puede tener un alcance amplio y transformador, incluyendo:

- Mejora en la eficiencia y eficacia en la identificación como en resolución de problemas de calidad.
- Análisis predictivo y detección temprana de posibles problemas de

calidad.

- Automatización de tareas repetitivas y reducción de errores humanos
- Aumento de la rapidez en la toma de decisiones y mejora en la respuesta a los clientes.
- Optimización de los recursos y reducción de costos. reducir el desperdicio por una mala planificación de calidad.

4.5. Limitaciones

Sin embargo, es importante tener en cuenta que la implementación de inteligencia artificial requiere una planificación cuidadosa y una comprensión profunda de los procesos de calidad como los objetivos a alcanzar al igual que existirán alcances también existirán algunos de los desafíos en la implementación de inteligencia artificial en el control de calidad de limones son:

- Datos de calidad: es importante tener una base sólida de datos de calidad para entrenar a los modelos de inteligencia artificial. Si los datos son insuficientes o de baja calidad, los resultados pueden ser poco precisos.
- Integración en procesos existentes: la implementación de inteligencia artificial en el control de calidad de limones debe ser integrada en los procesos existentes de forma suave para evitar interrupciones en el flujo de trabajo.
- Costos: la implementación de inteligencia artificial puede ser costosa, especialmente si se requiere una gran cantidad de datos y *hardware* especializado.
- Capacitación: es importante proporcionar capacitación adecuada a los trabajadores para que puedan utilizar eficazmente la tecnología de inteligencia artificial.

- Calibración y validación: es importante calibrar como validar los modelos de inteligencia artificial para garantizar su precisión evitando decisiones equivocadas en el control de calidad de limones.

5. CARACTERIZACIÓN Y DISEÑO

5.1. Requisitos de calidad para frutos cítricos

La calidad de los frutos cítricos se puede evaluar con base en varios factores, como la apariencia el cual incluye su tamaño, forma, color y estado de madurez, su calidad sabor (acidez, dulzor). La textura, firmeza, jugosidad; El contenido de los nutrientes; vitaminas, minerales como contaminantes pesticidas, microorganismos patógenos como otros contaminantes ambientales pueden afectar la calidad de los frutos cítricos que representar un riesgo para la salud humana.

Es importante evaluar la calidad de los frutos cítricos antes de su comercialización y distribución para garantizar que sean seguros para el consumo humano que a su vez cumplan con los estándares regulables aplicables.

Tabla 2.

Plagas y enfermedades comunes del limón

Plaga y/o Enfermedad	Descripción	Foto
Manchas negras	Provoca lesiones necróticas de color negro y caída prematura del fruto.	

Continuación de tabla 2.

Cancrosis	Enfermedad bacteriana que provoca caída prematura de hojas y lesiones, muerte en los frutos.	
Melanosis	Pocitos redondos con bordes amarillentos. A medida que la enfermedad avanza, las manchas se tornan protuberantes y toman una tonalidad marrón	
Sarna del naranjo dulce	Presentan lesiones corchosas y con forma de verrugas. Son de color gris o tostado	

Nota: Identificación de enfermedades emergentes de los cítricos. Adaptado de los programas educativos del Texas AgriLife Extension Servic (2020). *Guía de enfermedades de los cítricos*. (https://counties.agrilife.org/liberty/files/2020/05/Citrus-Disease-Guide-The-Quick-ID-to-Emerging-Diseases-of-Texas-Citrus-Spanish-Publ.-E-265S.pdf).Consultado 23 de mayo de 2022. De dominio público.

Tabla 3.

Calidad de colores para cítricos

Categoría	Descripción	Foto
Extra	Calidad superior y no presenta defectos marcados solo alteraciones superficiales que no afecten al aspecto general del producto ni a su calidad.	
Categoría I	Presenta características como decoloración y efectos de epidermis, cicatrizado por granizo o rozaduras, malformaciones, pero sin que afecten al aspecto general del producto ni a su calidad.	
Categoría tipo II	Cumple los requisitos mínimos, pero si entrar a categorías superiores, posee defectos como coloración, cascara rugosa, incrustaciones plateadas o quemaduras, señales de golpe de granizo, rozaduras y alteraciones epidérmicas.	

Nota: Requisitos de Calidad en Color para frutos Cítricos. Adaptado de agrícola, T.(n.d.) (2011). *Requisitos de calidad en color de frutos cítricos*. (https://www.tecnicoagricola.es/requisitos-de-calidad-en-color-para-frutos-citricos/)Consultado 25 de mayo de 2023. De dominio Público.

A continuación, se presenta los límites de color a considerar según categorías tipo I, II Y III.

Tabla 4.

Límites de color en limones

Norma tipo I	Desverdizar tipo II	Destrío tipo III
		

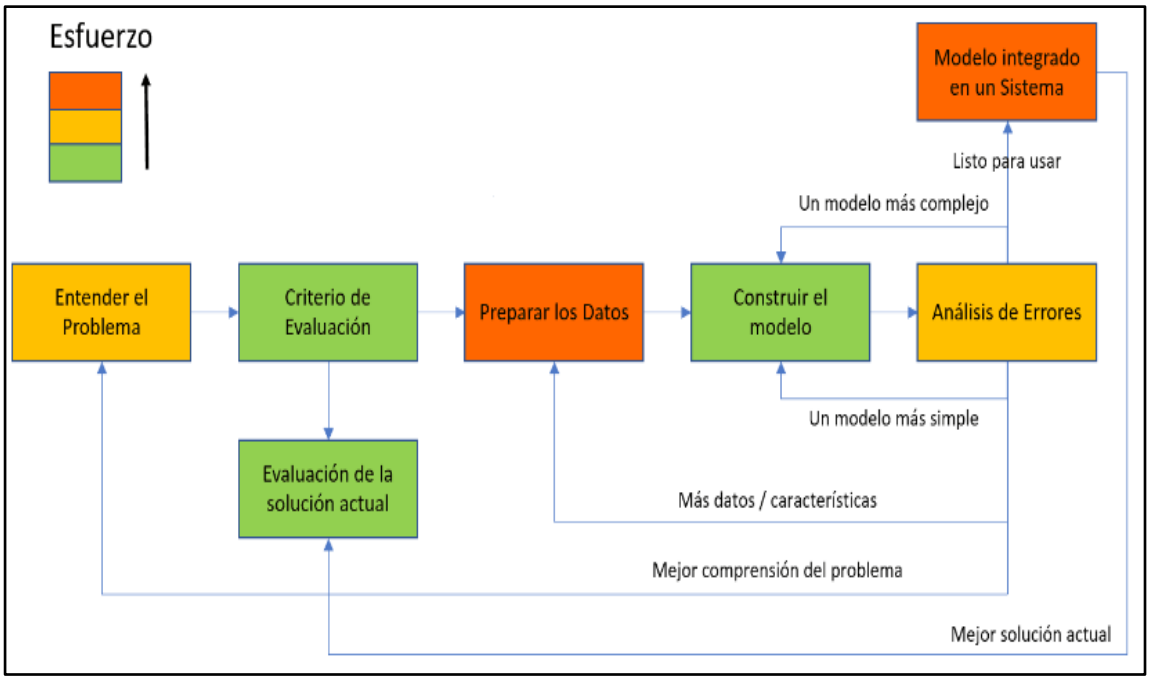
Nota: Requisitos de Calidad en Color para frutos Citricos. Adaptado de agrícola, T.(n.d.-b) (2011). *Requisitos de calidad en color de frutos cítricos*. (https://www.tecnicoagricola.es/requisitos-de-calidad-en-color-para-frutos-citricos/) Consultado 25 de mayo de 2023. De dominio Público.

5.2. Plan del diseño

Es importante considerar que la implementación de inteligencia artificial requiere una planificación cuidadosa y una comprensión profunda de los procesos de calidad los objetivos a alcanzar, en el diagrama de fases del modelo se presenta la planificación del modelo como las variables a considerar.

Figura 40.

Diagrama de fases en modelos de Machine Learning

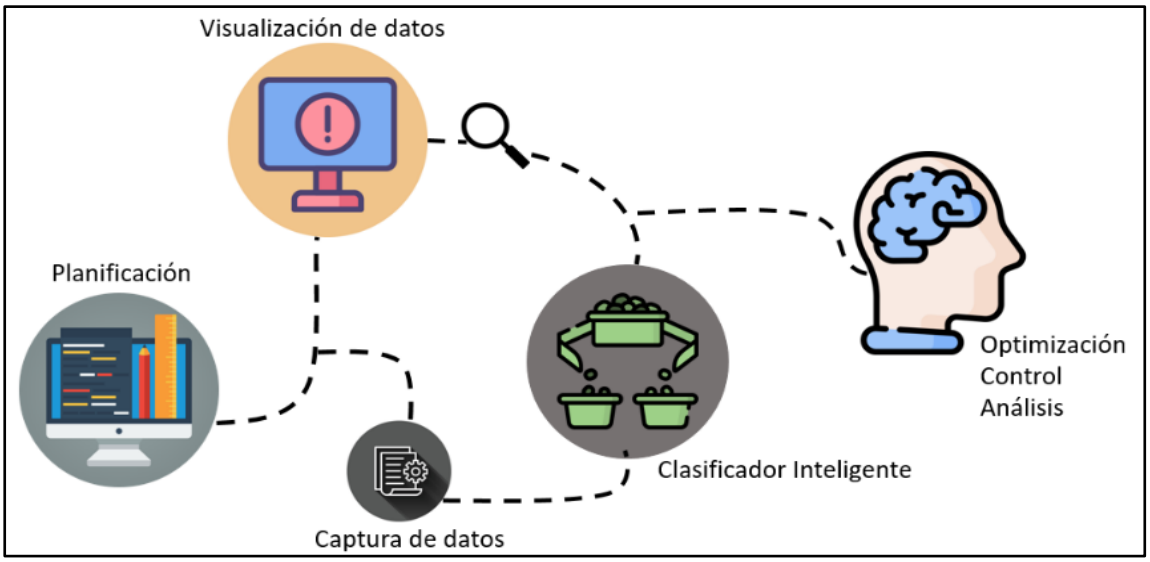


Nota: Las 7 Fases del Proceso de Machine Learning. Adaptado de Martínez Heras. J (2020). Resumen de las 7 fases del proceso de machine learning (https://www.iartificial.net/fases-del-proceso-de-machine-learning/) Consultado el 30 de mayo de 2022. De dominio público.

De la misma forma se plantea el diagrama del proceso el cual se presenta en la figura 41.

Figura 41.

Proceso productivo tecnológico



Nota: Visualización del Proceso productivo tecnológico. Elaboración propia, realizado con AfterShot Pro 3 RAW Photo Editor.

Una vez obtenido la planificación se gestiona la acción del proyecto la cual se consideraron los siguientes parámetros:

- Algoritmo: elegir el algoritmo adecuado para el problema que se está tratando.
- Hyperparámetros: valores predefinidos que se utilizan en el algoritmo para mejorar su rendimiento.
- Función de pérdida: métrica que se utiliza para evaluar la precisión del modelo.
- Métricas de evaluación: métodos para evaluar el rendimiento del modelo en términos de precisión, recall, F1, entre otros.
- Dataset: cantidad y calidad de los datos utilizados para entrenar el

modelo.

- Valoración cruzada: proceso que se utiliza para evaluar el modelo y prevenir el sobreajuste.
- Regularización: técnica que se utiliza para prevenir el sobreajuste y mejorar la generalización del modelo.
- Número de épocas: cantidad de iteraciones que se realizan para entrenar el modelo.

Estos parámetros son importantes para asegurarse de que el modelo se entrene de manera efectiva y logre un rendimiento óptimo en la tarea de control de calidad automático.

5.3. Desarrollo del modelo de aprendizaje automático

El desarrollo del modelo de aprendizaje automático implica la creación de algoritmos y técnicas que permiten a las máquinas aprender a partir de datos sin ser programadas explícitamente, mejorando su rendimiento a medida que se les proporciona más información, lo que permite tomar decisiones y realizar tareas complejas de forma automática y precisa.

5.3.1. Google *Colab*

Google *Colab* es un entorno de *jupyter* en línea gratuito proporcionado por Google. Permite ejecutar compartir código en *Python* en un entorno en la nube, con acceso a recursos informáticos potentes como GPUs y TPUs. En la cual se pueden entrenar desarrollar modelos de aprendizaje automático como otras aplicaciones de inteligencia artificial sin tener que invertir en *hardware* costoso. Además, *Colab* permite guardar, compartir fácilmente proyectos, lo que lo hace ideal para colaboraciones como trabajo en equipo. Por estas razones,

Google *Colab* se ha vuelto muy popular en la comunidad de aprendizaje automático, ciencia de datos.

5.3.1.1. Justificación del uso de Google *Colab* como entorno de desarrollo

El uso de Google *Colab* para el desarrollo de una red neuronal convolucional en la clasificación de imágenes presenta varias justificaciones:

- Acceso a recursos de cómputo: proporciona acceso a recursos de cómputo, como GPUs (Unidades de Procesamiento Gráfico) y TPUs (Unidades de Procesamiento Tensorial). Estas unidades de procesamiento aceleran el entrenamiento como la inferencia de modelos de redes neuronales convolucionales, lo que permite un desarrollo rápido, así como eficiente.

- Integración con bibliotecas con herramientas populares: es compatible con las bibliotecas y herramientas más comunes utilizadas en el desarrollo de redes neuronales convolucionales, *TensorFlow* como *Keras*. Esto facilita la implementación aparte del uso de estas bibliotecas, lo que acelera el desarrollo al igual que la experimentación con diferentes arquitecturas de red como técnicas de entrenamiento.

- Integración con Google *Drive* con otras herramientas de Google: está integrado con Google *Drive*, lo que facilita la carga, así como el almacenamiento de datos de entrenamiento y modelos. También se pueden aprovechar otras herramientas con servicios de Google, como Google *Cloud Platform*, para ampliar las capacidades del proyecto, si es necesario.

En general, el uso de Google *Colab* en el desarrollo redes neuronales convolucionales para la clasificación brinda ventajas en términos de acceso a recursos de cómputo como compatibilidad con bibliotecas populares, facilidad de colaboración y compartición de código, con otras herramientas de Google. Estas justificaciones respaldan la elección como una plataforma conveniente para el desarrollo de este proyecto.

Google *Colab* al igual que los servicios de Google ofrece beneficios que se obtiene al adquirir el servicio Pro, a continuación, se presenta una tabla comparativa entre Google *Colab* Pro, Google *Colab* Pro+ y Google *Colab* gratuito:

Tabla 5.

Tabla comparativa de tipos de Servicio de Google Colab

Características	Google *Colab* Pro	Google *Colab* Pro+	Google *Colab* gratuito
Potencia de cómputo GPU	Acceso a GPU Tesla	Acceso a GPU Tesla	Acceso a GPU K80
	K80	P100	--
Tiempo de ejecución	24 horas	24 horas	12 horas
Memoria RAM	27 GB	36 GB	12 GB
Almacenamiento	100 GB	100 GB	25 GB
Prioridad de ejecución	Alta	Alta	Baja
Colaboradores simultáneos	1	1	1

Nota: Tabla de tipos de Servicio de Google Colab. Elaboración propia, realizado con Microsoft Excel.

Google *Colab* Pro ofrece mayores recursos computacionales, incluyendo una GPU más potente, mayor tiempo de ejecución, mayor capacidad de memoria RAM y almacenamiento. Además, al utilizar *Colab* Pro se tiene mayor

prioridad en la ejecución de los cuadernillos de trabajo. Sin embargo, Google *Colab* gratuito aún proporciona un acceso valioso a recursos de computación, aunque con limitaciones en términos de tiempo de ejecución, capacidad de almacenamiento como prioridad en la cola de ejecución.

Para el desarrollo del modelo que se explica en este capítulo se optó por usar Google *Colab* Pro+, por las características mencionadas en la Tabal V.

Además, otra de las razones por lo que se hace uso de Google *Colab* Pro+ es Potencia de cálculo: Google *Colab* Pro+ proporciona acceso a GPU Tesla P100, que es una unidad de procesamiento gráfico de alto rendimiento. Estas GPUs son especialmente útiles para tareas intensivas en cómputo, como el entrenamiento de modelos de inteligencia artificial y el procesamiento de grandes conjuntos de datos. La potencia de las GPUs en *Colab* Pro+ supera a la mayoría de las GPUs de las PCs convencionales.

Las PCs pueden verse limitadas por su tiempo de ejecución y capacidad de recursos, Google *Colab* Pro+ ofrece un tiempo de ejecución de 24 horas continuas. Esto permite ejecutar tareas prolongadas sin interrupciones ni restricciones de tiempo.

También el uso de Google *Colab* Pro+ ofrece una mayor cantidad de memoria RAM y espacio de almacenamiento en comparación con las PCs convencionales. Esto es beneficioso para manejar conjuntos de datos más grandes ejecutando modelos más complejos.

5.3.2. TensorFlow

TensorFlow es una plataforma de código abierto de aprendizaje automático desarrollada por Google *Brain Team*. Se utiliza principalmente para el desarrollo y entrenamiento de modelos de aprendizaje profundo, incluyendo redes neuronales, es ampliamente utilizada en investigación, producción en el campo de la inteligencia artificial. Con *TensorFlow*, se pueden crear modelos de aprendizaje automático complejos, integrarlos en aplicaciones optimizando el rendimiento para una amplia gama de sistemas, incluidos los dispositivos móviles, servidores y supercomputadoras.

5.3.3. TensorFlow.js

TensorFlow.js es una biblioteca de aprendizaje automático de código abierto de Google, que permite ejecutar modelos de aprendizaje profundo en el lado del cliente (en el navegador web) mediante *JavaScript*. *TensorFlow.js*, permite desarrollar, entrenar modelos de aprendizaje automático en *JavaScript*, lo que permite construir aplicaciones de inteligencia artificial con capacidades de aprendizaje automático en la web sin necesidad de un servidor. Además, *TensorFlow.js* permite la importación y la reutilización de modelos pre-entrenados desarrollados con otras bibliotecas de *TensorFlow*, pudiendo aprovechar fácilmente la capacidad de aprendizaje profundo en sus aplicaciones web. Lo que simplifica el desarrollo como también la implementación de aplicaciones en páginas Web.

5.3.4. Algoritmo de la Red Neuronal

El algoritmo utilizado es una Rede Neuronal Convolucional debido a la naturaleza del proyecto, ya es una herramienta efectiva para la detección y

clasificación de patrones en imágenes, lo que las hace particularmente útil en aplicaciones de control de calidad. Al procesar los datos de imagen a través de capas convolucionales, las Redes Neuronales Convolucionales puede aprender características específicas de los patrones de interés realizar tareas de detección clasificación con alta precisión. Además, presenta resistencia a pequeñas deformaciones como a variaciones en los patrones, lo que la hace adecuada para aplicaciones en entornos realistas donde los patrones pueden estar distorsionados o sujetos a variaciones.

5.3.5. *Dataset* de entrenamiento

Una vez definido el algoritmo a utilizar, se define el *datasets* de imágenes que se utilizan como input para entrenar y evaluar el modelo de aprendizaje automático. El *dataset* proporciona al modelo una comprensión de los patrones así también las relaciones presentes en los datos, lo que permite al modelo realizar tareas de predicción, la toma de decisiones en situaciones similares a las que se le presentaron durante el entrenamiento.

El *datasets* fue cuidadosamente seleccionado para que sea representativo de la población de interés para garantizar que el modelo entrenado tenga la capacidad de generalizar bien los datos y de dotar la capacidad de predecir situaciones para las cual nunca fue entrenada. El *datasets* fue dividido en dos sets de datos: excelente calidad, Mala calidad; estos dos sets de datos forman un *dataset* de 2080 imágenes, las cuales 1,125 perteneces al set de Excelente calidad y 955 al set de Mala calidad, las imágenes fueron redimensionadas a un tamaño de 300x300 pixeles para su procesamiento.

Figura 42.

Dataset de entrenamiento para modelo convolucional

Nota: Fotos de limones Excelente calidad y Mala calidad para conformar los DataSets de entrenamiento. Elaboración propia.

1.1.1. Épocas de entrenamiento

Para el entrenamiento del modelo debe considerarse un punto importante, las épocas, las épocas del entrenamiento se refiere al número de veces que el modelo entrena con el mismo conjunto de datos. Durante cada época, el modelo recibirá los datos de entrada realizando así ajustes en los pesos y vías para mejorar su precisión. Una época completa se considera terminada cuando el modelo ha procesado todos los ejemplos de entrenamiento.

El número de épocas que se deben realizar depende de la cantidad de datos como de la complejidad del modelo, pero un número típico es de 10-50, en este modelo se utilizaron 50 épocas de entrenamiento.

5.3.6. Función de activación

Una vez definido las épocas de entrenamiento se debe tomar en cuenta la función de activación en este modelo se utilizará la *ReLU (Rectified Linear Unit*) la cual se utiliza a menudo en redes neuronales convolutionales porque proporciona una solución eficiente a dos problemas comunes en el aprendizaje automático: el problema de la saturación y el problema de la escala de los valores de entrada.

5.3.6.1. Problema de saturación

Las funciones de activación tradicionales, como la sigmoide y la tangente hiperbólica, pueden saturarse pudiendo devolver valores cercanos a 0 o 1, lo que puede obstaculizar el proceso de aprendizaje. La ReLU resuelve este problema, debido a su linealidad para valores positivos devuelve 0 para valores negativos.

5.3.6.2. Problema de escala de valores de entrada

Las funciones de activación lineales, como la ReLU, también son más eficientes en términos de cálculo, debido que requieren cálculos exponenciales costosos. Además, las funciones lineales también ayudan a mantener la escala de los valores de entrada, lo que puede ser importante para el correcto funcionamiento de la red neuronal.

En resumen, la ReLU es una función de activación eficiente como efectiva que resuelve los problemas de saturación y escala de los valores de entrada, lo que la convierte en una buena opción para su uso en redes neuronales convolucionales.

5.3.7. Función de activación de salida

Una vez obtenida la función de activación a utilizar se debe definir la función de activación de salida.

La función de activación *Softmax* se utiliza en redes neuronales convolucionales para proporcionar una probabilidad de salida para cada clase en una tarea de clasificación multi-clase. La función *Softmax* toma un vector de valores de entrada y devuelve un vector de probabilidades normalizadas, donde cada elemento del vector representa la probabilidad de que la entrada pertenezca a una determinada clase. Esto permite a la red neuronal predecir la clase más probable para una entrada dada, normalizando la suma de las probabilidades a 1, pudiendo ser 1 Excelente calidad y 0 Mala calidad la suma de estas dos probabilidades dará 1 en la neurona de salida.

5.3.8. Optimizador de la red neuronal convolucional

En la rede neuronal se utilizará el optimizador Adam, y comúnmente utilizada en modelos de clasificación debido a sus características efectivas en la optimización de funciones de pérdida complejas. Algunas de las ventajas de utilizar Adam son:

- *Adaptive learning rate*: Adam utiliza una técnica de ajuste de tasa de aprendizaje que se ajusta dinámicamente a cada parámetro, lo que permite una convergencia más rápida y estable.
- *Momentum*: Adam también utiliza una técnica de momento que ayuda a superar los mínimos locales y mejorar la convergencia.
- Computacionalmente eficiente: Adam es computacionalmente eficiente y requiere menos recursos que otros optimizadores, como SGD (*Stochastic*

Gradient Descent).

Adam es un optimizador popular y efectivo para modelos de clasificación debido a su capacidad para manejar funciones de pérdida complejas, ajustar dinámicamente la tasa de aprendizaje siendo computacionalmente eficiente.

5.3.9. Lotes del aprendizaje automático

En el aprendizaje automático, un lote es un subconjunto de datos de entrenamiento que se utiliza para actualizar los pesos y sesgos en un modelo de redes neuronales. En lugar de utilizar todos los datos de entrenamiento en una sola iteración, el modelo se entrena en pequeños lotes de datos, conocidos como mini-lotes. Cada mini-lote se utiliza para calcular el gradiente de la función de pérdida y actualizar los pesos como sesgos en la dirección opuesta al gradiente.

El tamaño del lote es un hiperparámetro que se puede ajustar con capacidad de afectar el rendimiento y la velocidad de convergencia del modelo. Un tamaño de lote pequeño puede resultar en una convergencia más rápida, pero también puede ser más sensible a las fluctuaciones en los datos. Por otro lado, un tamaño de lote grande puede ser más estable, pero puede requerir más tiempo para converger.

Los lotes son una técnica importante en el entrenamiento de redes neuronales, que permite ajustar los pesos y sesgos en pequeños pasos que mejorar la eficiencia computacional, para este modelo se utilizaron un tamaño de lote de 16.

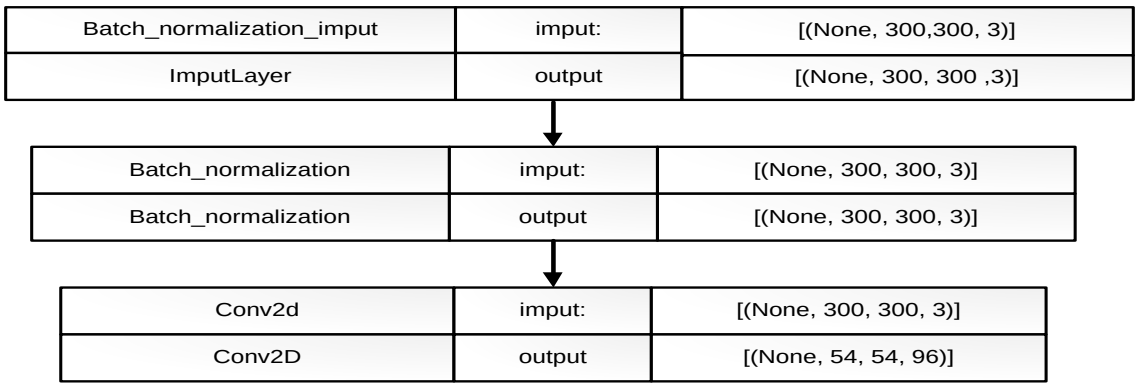
5.3.10. Tasa de aprendizaje

La tasa de aprendizaje es un hiperparámetro que controla la magnitud de la actualización de los pesos durante el entrenamiento. Es un factor que se multiplica con la gradiente para determinar el tamaño del paso en la dirección opuesta al gradiente para minimizar la función de pérdida

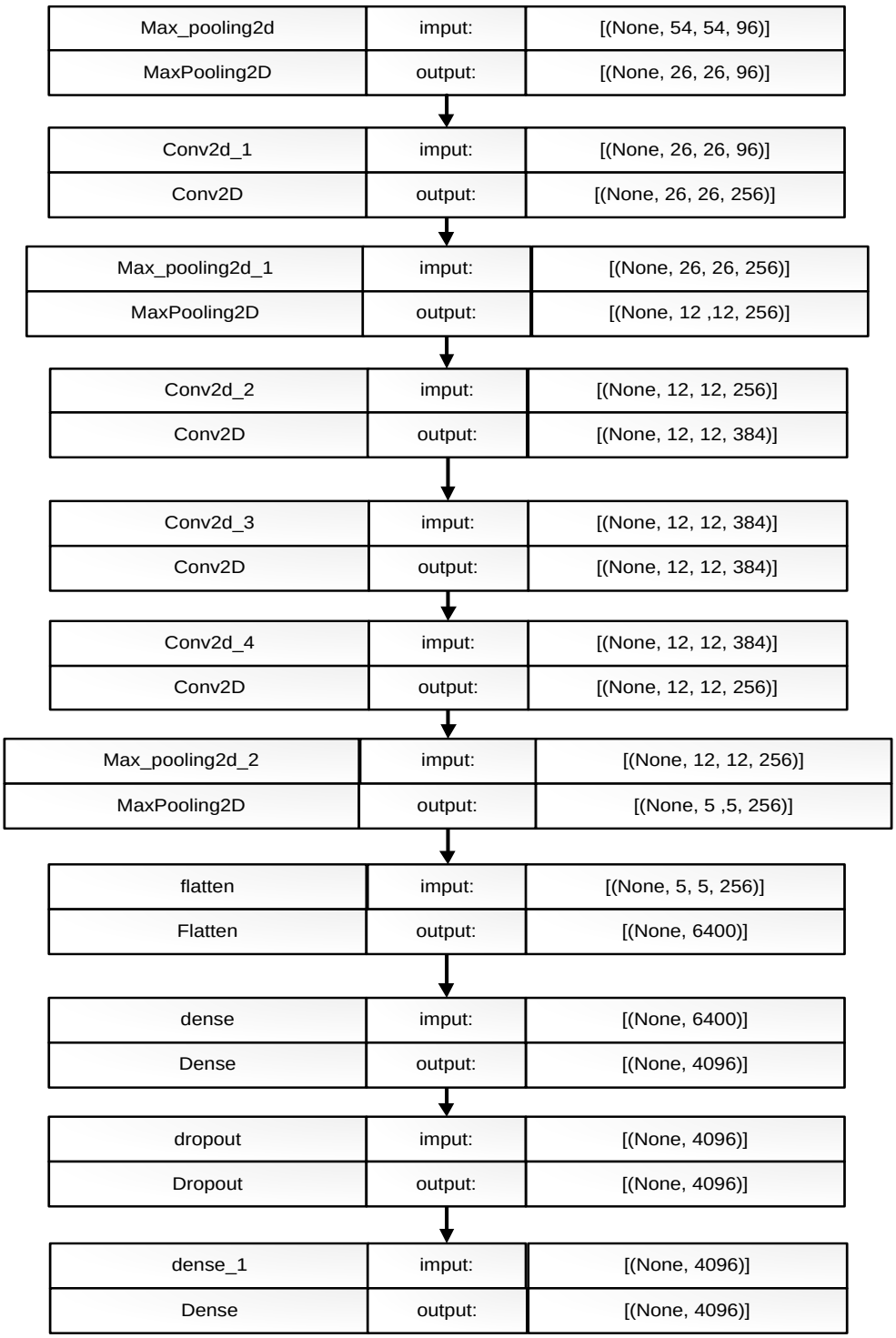
En este modelo se utilizó una taza de aprendizaje con valor de 0.001 es una tasa de aprendizaje comúnmente utilizada, siendo esta, buen punto de partida en la mayoría de los casos. Sin embargo, la tasa óptima de aprendizaje puede variar dependiendo del problema y del modelo, por lo que a menudo se recomienda probar varios valores diferentes para encontrar el que funcione mejor. Es posible que una tasa más alta o baja sea más adecuada en otro caso específico. Pero para el modelo este fue el óptimo. Luego de tener en cuenta todas las consideraciones del caso se entrena la rede neuronal convolucional y la arquitectura del modelo se presentan a continuación:

Tabla 6.

Arquitectura del modelo



Continuación de la tabla 6.



Continuación de la tabla 6

dropout_1	imput:	[(None, 4096)]
Dropout	output:	[(None, 4096)]
dense	imput:	[(None, 4096)]
Dense	output:	[(None,1)]

Nota: Arquitectura de Red Neuronal Convolucional utilizada. Elaboración propia, realizado con Google Colab, Tensorflow.

6. RESULTADOS DEL MODELO

6.1. Evaluación del modelo

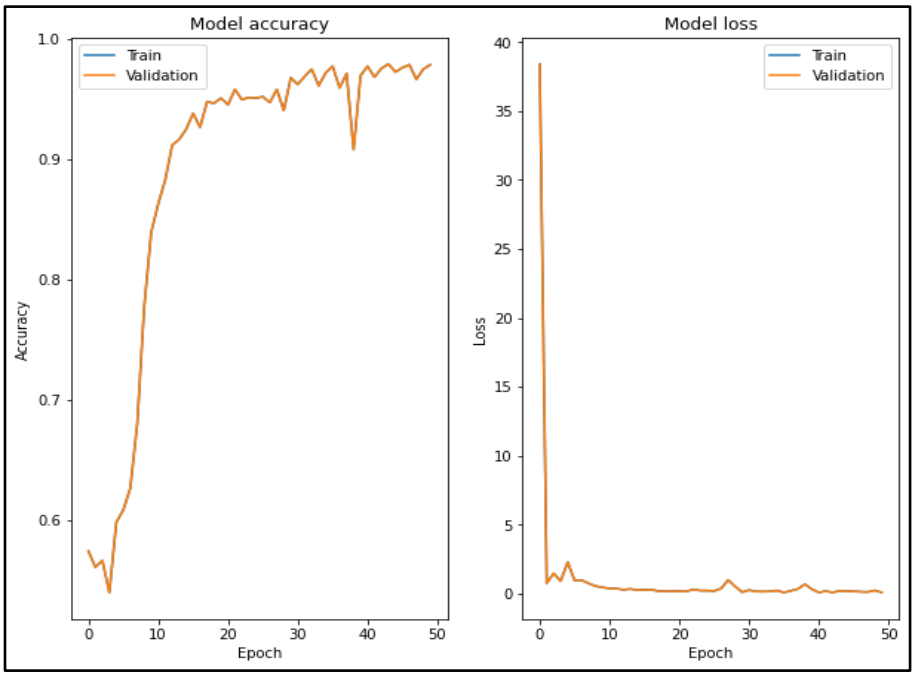
La evaluación del modelo de aprendizaje automático implica medir su rendimiento y precisión utilizando métricas específicas, como la precisión, el recall, la precisión media, la matriz de confusión, entre otras, para determinar qué tan bien generaliza y se ajusta a nuevos datos y si cumple con los objetivos y requisitos establecidos para su aplicación en un problema específico.

6.1.1. Precisión de entrenamiento

Durante el entrenamiento se tomó el 25 % para datos de validación, en las siguientes gráficas se muestra el resultado del entrenamiento tanto de perdida como de aprendizaje, presentados por épocas, el modelo ha presentado un acierto del 98 % por lo que se considera exitoso el proceso.

Figura 43.

Exactitud y pérdida del modelo durante 50 épocas de entrenamiento



Nota: Visualización de exactitud y pérdida del modelo durante 50 épocas de entrenamiento. Elaboración propia, realizado con Google *Colab*, Tensorflow.

6.1.2. Matriz de confusión

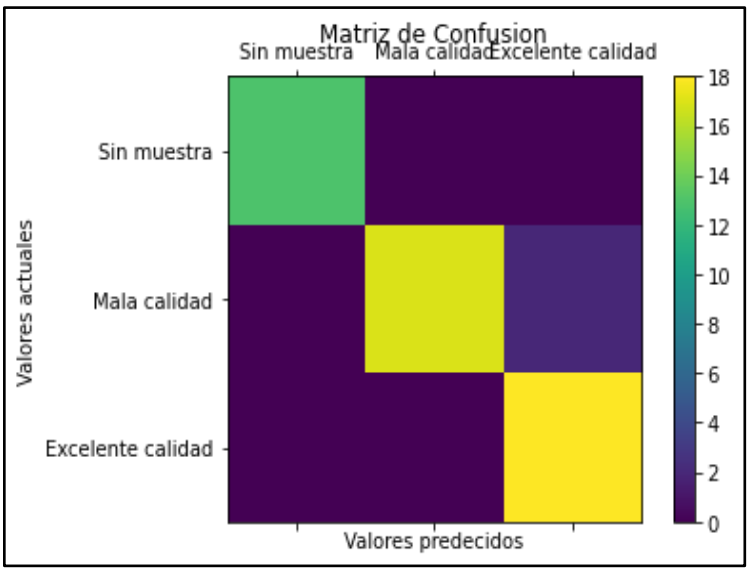
El análisis de la matriz de confusión indica un alto nivel de precisión en el rendimiento del algoritmo, con un 2 % de desacierto. Esto significa que el modelo está correctamente identificando el 98 % de las muestras en el conjunto de datos de prueba. Además, la matriz de confusión también muestra la cantidad de falsos positivos y falsos negativos que ocurren en el algoritmo. Estos valores son importantes para determinar la robustez al igual que la eficacia del modelo en situaciones reales. En general, un alto porcentaje de

acierto es una indicación positiva de que el modelo está funcionando de manera efectiva.

Sin embargo, es importante tener en cuenta que un alto porcentaje de acierto no garantiza necesariamente un modelo perfecto. Puede haber casos en los que el modelo aún tenga una tasa elevada de error debido a factores externos o a la naturaleza del problema que se está tratando de resolver. Por lo tanto, es importante continuar evaluando y mejorando el modelo para asegurarse de que tenga un rendimiento óptimo en todo momento.

Figura 44.

Matriz de confusión para el modelo clasificatorio



Nota: Visualización de matriz de confusión. Elaboración propia, realizado con Google Colab.

6.1.2.1. Análisis del modelo

Hay varios factores que pudieron afectar el rendimiento del modelo convolucional durante el entrenamiento, incluyendo:

- Calidad de los datos: la calidad de los datos de entrenamiento es crucial para el éxito de la red neuronal. Si los datos son ruidosos, incompletos o no representativos de los datos reales, la red neuronal puede tener dificultades para aprender y generalizar de manera efectiva. En este caso un problema durante la predicción pudo deberse a los datos, puesto que al no contar un *dataset* con un volumen grande de imágenes pudo haber afectado que las predicciones fallaran el 2 % debido a los datos insuficientes, el modelo pudo ajustarse a los ruidos en los datos no siendo capaz de generalizar a datos desconocidos.

- Número de épocas: el número de épocas puede tener un impacto significativo en el rendimiento del modelo. Un número insuficiente de épocas puede resultar en un modelo subóptimo, mientras que un número excesivo de épocas puede resultar en sobreajuste. En las gráficas no se muestra un sobreajuste, aunque las épocas de entrenamiento puedo ser un factor contribuyente. Pudo que el modelo se ajustara demasiado a los datos de entrenamiento y perdiera la capacidad de generalizar a datos desconocidos creando así un 2 % de inexactitud.

7. DISEÑO DE LA LÍNEA DE CONTROL DE CALIDAD

En diseño de la línea de control de calidad se consideró los siguientes casos:

- Calidad de los productos: es importante asegurarse de que la calidad de las frutas se mantenga durante todo el proceso de producción.

- Capacidad de procesamiento: la línea de producción debe tener suficiente capacidad para cumplir con la demanda de producción.

- Eficiencia de la producción: el diseño de la línea de producción debe ser eficiente y minimizar los tiempos muertos como los desperdicios.

- Requisitos sanitarios: la línea de producción debe cumplir con los requisitos sanitarios y de seguridad alimentaria para garantizar la salud de los consumidores.

- Costos: el diseño de la línea de producción debe ser cost-efectivo y minimizar los gastos de producción.

- Automatización: la automatización puede ayudar a aumentar la eficiencia y la consistencia de la producción.

- Mantenimiento: la línea de producción debe ser fácil de mantener y reparar en caso de fallas.

- Flexibilidad: la línea de producción debe ser flexible y adaptarse a diferentes tipos de frutas como así también cambios en la demanda.

7.1. Diseño de la banda transportadora

En bandas transportadoras de baja velocidad, los motores utilizados son generalmente motores eléctricos de corriente alterna (AC) o corriente continua (DC). Algunos de los motores más comúnmente utilizados en bandas transportadoras de baja velocidad incluyen:

- Motores de corriente alterna: estos motores son ampliamente utilizados en bandas transportadoras de baja velocidad debido a su fiabilidad, eficiencia y bajo costo.

- Motores de corriente continua: estos motores son adecuados para aplicaciones que requieren un control preciso de la velocidad y el par, como las bandas transportadoras de baja velocidad.

- Motores de corriente continúa sin escobillas: estos motores tienen una vida útil más larga y una mayor eficiencia que los motores de corriente continua convencionales, lo que los hace adecuados para aplicaciones de bandas transportadoras.

- Motores síncronos: estos motores son adecuados para aplicaciones que requieren una velocidad constante y un control preciso, como las bandas transportadoras de baja velocidad.

En general, la elección del motor adecuado para una banda transportadora de baja velocidad depende de los requisitos específicos de la

aplicación, como la velocidad, la capacidad de carga, el control de velocidad y la eficiencia. La intención de esta tesis es demostrar el desarrollo a una solución para una necesidad particular para así demostrar el concepto como la viabilidad del proyecto, por lo cual se diseñará y desarrollar la banda transportadora. Esto te permitirá tener un mayor control sobre las especificaciones/características del sistema.

7.1.1. Especificaciones de la banda transportadora

En el proyecto se utilizar lo que es un motor paso a paso a que es un motor adecuado para aplicaciones que requieren un control preciso de la velocidad y la posición, pueden proporcionar una excelente precisión en términos de control de velocidad como de posición se utilizara el motor paso a paso NEMA 17.

Tabla 7.

Especificaciones técnicas Motor NEMA 17, 3.2 kg/cm

Especificaciones técnicas Motor NEMA 17	
Pasos por vuelta	200 (1, 8º/paso)
Tensión	4 V
Corriente	1.2 Amperios por bobinado
Torque	3.2 kg/cm (44 oz-in)
Resistencia	3.3 Ohm por bobina
Inductancia	2.8 mH por bobina
Peso	350 gramos (13 oz)

Nota: Motor paso a paso 3.2 Kg/cm NEMA 17. Obtenido de Bricogeek (2023). Especificaciones técnicas Motor Nema 17 (https://tienda.bricogeek.com/motores-paso-a-paso/546-motor-paso-a-paso-nema-17-32kg-cm.html) Consultado el 03 de junio de 2022. De dominio público.

Tabla 8.

Especificaciones técnicas de la banda transportadora

Largo de la banda transportadora	100 cm
Altura de la banda	20 cm
Ancho de la banda	15 cm
Velocidad de la banda	100 cm/min

Nota: Detalles técnicos de la banda transportadora. Elaboración propia. realizado con Microsoft Excel.

Tabla 9.

Materiales para utilizar en banda transportadora

Materiales
Perfil de aluminio extruido 2020 ranurado en T largos
Ángulo interno en forma de L y tornillos prisioneros con cabeza allen
Cojinete de sujeción con rodamiento interno para varilla de 8mm
Tornillos cabeza hexagonal allen M5 x 8
Tuerca para ranura en T
Ejes lisos en acero de 8mm
Banda dentada GT2
Motor paso a paso NEMA 17
Soporte en angulo para motor paso a paso NEMA 17
tornillos cabeza hexagonal allen M3 x 6
Polea dentada 16 dientes GT2 para motor 5mm
Polea dentada 20 dientes GT2 para ejes 8mm
Llaves hexagonales allen de 1.5, 2 y 4 mm

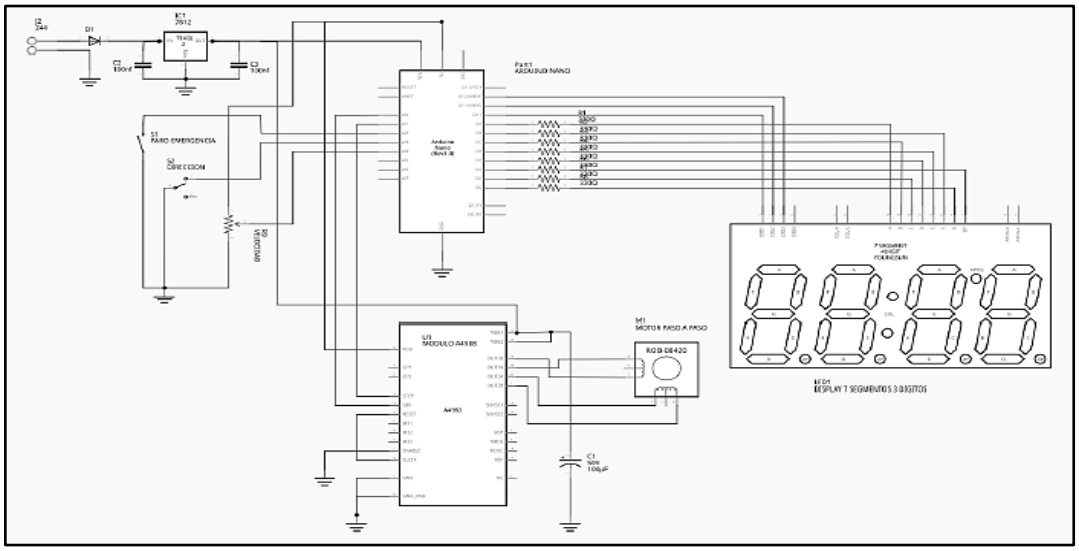
Nota: Materiales para utilizar en banda transportadora. Elaboración propia, realizado con Microsoft Excel.

Para el control de velocidad se optó por utilizar la placa de desarrollo Arduino dado que el Arduino es fácil de usar y tienen una amplia variedad de componentes disponibles, lo cual los hace ideales para la creación de prototipos

Sin embargo, es importante tener en cuenta que las placas de desarrollo como Arduino están diseñadas para prototipos, no para aplicaciones de producción. Si bien el proyecto es un prototipo exitoso y se requiere producir en grandes cantidades, es posible que sea necesario considerar un sistema de control más robusto y sofisticado que pueda manejar las demandas de un entorno de producción.

Figura 45.

Diagrama de control de la banda transportadora



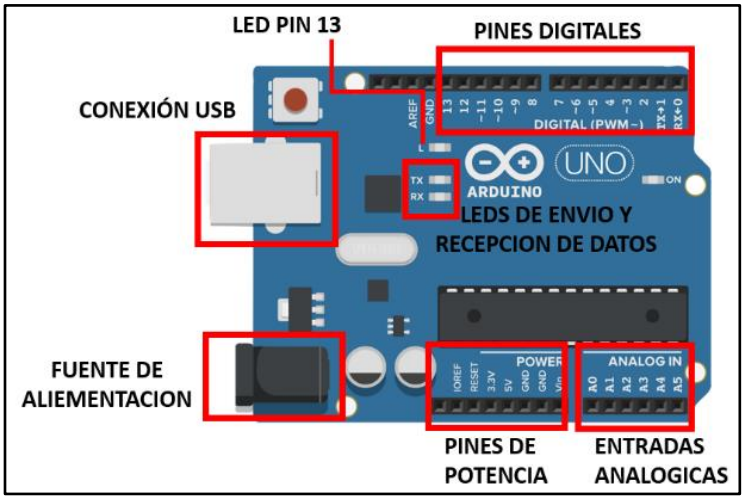
Nota: Banda transportadora desarmable y control de velocidad con Arduino. Obtenido de Automatízanos (2022). Circuito de control banda trasportadora (https://docplayer.es/77175903-Banda-transportadora-desarmable-y-control-de-velocidad-con-arduino.html)Consultado el 07 de junio de 2022. De dominio público.

7.2. Placa de desarrollo Arduino

Una placa de desarrollo Arduino es un microcontrolador programable que se utiliza para crear proyectos electrónicos interactivos. Es una plataforma de código abierto que utiliza un lenguaje de programación propio basado en C/C++. Las placas Arduino se pueden programar para controlar diversos componentes electrónicos, como luces, motores, sensores y pantallas, entre otros.

Figura 46.

Pines y entradas de Arduino uno



Nota: Visualización de pines y entradas Arduino. Elaboración propia, realizado con Paint.Net.

Es posible que se pregunte por qué se optó por utilizar Arduino en lugar de Microcontrolador para implementar una red neuronal convolucional (CNN). Esta elección depende de diversos factores y requisitos particulares del proyecto. A continuación, se presentan algunas razones que podrían respaldar la preferencia por Arduino en lugar de usar un Microcontrolador:

- Disponibilidad y comunidad: Arduino cuenta con una amplia comunidad de usuarios como así también una gran cantidad de recursos disponibles en línea. Esto facilita el acceso a documentación, tutoriales, ejemplos de código aparte de soporte técnico, lo que puede agilizar el desarrollo en solución de problemas en la implementación.

- Flexibilidad y facilidad de programación: Arduino utiliza un lenguaje de programación basado en C/C++, que es ampliamente conocido y utilizado. Esto permite una programación más accesible para aquellos familiarizados con estos lenguajes, ofrece flexibilidad para adaptar así también personalizar según las necesidades del proyecto.

- Costo y accesibilidad: Arduino es conocido por ser una plataforma asequible de bajo costo en comparación con otros Microcontroladores. Esto puede ser beneficioso especialmente en proyectos con restricciones presupuestarias, permitiendo una implementación económica de la CNN sin comprometer su rendimiento.

- Ecosistema de *hardware* y periféricos: Arduino ofrece una amplia gama de placas como de módulos complementarios que pueden ser utilizados en conjunto con la implementación de la CNN. Estos incluyen sensores, actuadores además de otros dispositivos, lo que brinda la posibilidad de crear soluciones más completas e interconectadas.

Tabla 10.

Tabla comparativa entre Arduino y Microcontrolador

Aspecto	Arduino	Microcontrolador
Facilidad de uso	Diseñado para principiantes y programación amigable	Requiere conocimientos técnicos más avanzados
Comunidad y soporte	Gran comunidad y amplia documentación disponible	Comunidad más pequeña, menos recursos disponibles
Bibliotecas y shields	Amplia variedad de bibliotecas y shields	Menor cantidad de bibliotecas y shields disponibles
Flexibilidad	Versátil y compatible con diversos periféricos	Más limitado en términos de compatibilidad
Consumo de energía	Consumo de energía relativamente mayor	Mayor eficiencia energética en aplicaciones simples
Costo	Generalmente más accesible en términos de precio	Puede ser más costoso dependiendo del modelo
Capacidad de control	Menos control de *hardware* y procesamiento	Mayor control y capacidades de procesamiento

Nota: Similitudes entre Arduino y Microcontrolador. Elaboración propia, realizado con Microsoft Excel

7.2.1. Actuador neumático para la clasificación del producto

Un actuador neumático o denominados como cilindros neumáticos, cilindros de aire, convierte el aire comprimido en movimiento lineal utilizado en aplicaciones industriales donde se requiera velocidad en aplicaciones automatizadas.

Figura 47.

Cilindro neumático redondo SMC de 16mm de diámetro, carrera de 50 mm

Nota: Cilindro neumático redondo SMC de 16mm de diámetro. Obtenido de APLITEGUA (2022). Cilindros redondos (https://aplitegua.com.gt/catalogo/subcategorias/3/87/51/CILINDROS_REDONDOS) Consultado el 10 de junio de 2022. De dominio público.

7.2.2. Cámara Web

Cámara de video digital con conexión directa a una computadora, de escritorio o portátil, con resolución Full HD 1080P (1920x1080 pixeles) con formato de imagen: BMP / JPEG / PNG. Y formato de video: AVI / WMV Tipo de lente: Trípode de5 vidrios. Compatible con Windows y MacOS.

Figura 48.

Cámara Web Full HD 1080

Nota: Cámara Web Full HD 1080P Con Micrófono Incorporado Y Aislamiento De Ruido, Cam50. Obtenido de PACIFIKP (2022). *cámaras web*. (https://www.pacifiko.com/compras-en-linea/camara-web-full-hd-1080p-con-microfono-incorporado-y-aislamiento-de-ruido-cam50argom&pid=MGE5NzA4ZG). Consultado el 12 de junio de 2022. De dominio Público.

7.2.3. Conexión Serial P5

La conexión serial P5 es un protocolo de comunicación utilizado para enviar y recibir datos entre dispositivos a través de un puerto serie. En el contexto de la programación, se utiliza para transmitir información entre una aplicación con un Arduino.

Para comunicar una aplicación Java con una placa Arduino utilizando la conexión serial P5, se pueden utilizar los sockets de Java. Los sockets son una API (interfaz de programación de aplicaciones) que permite la comunicación entre procesos en una red.

Los pasos para establecer la comunicación serial entre una aplicación Java y una placa Arduino que utiliza sockets son los siguientes:

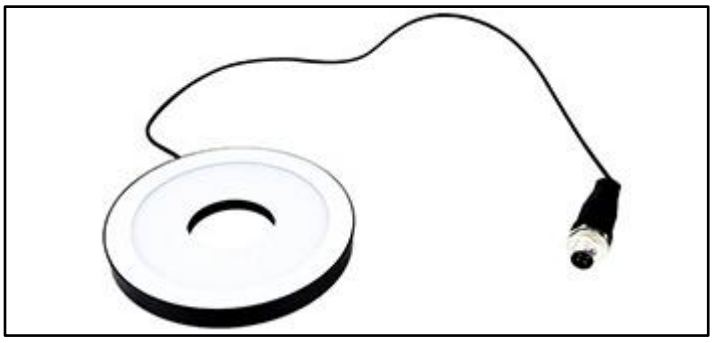
- Conectar la placa Arduino al puerto USB de la computadora.
- Cargar un sketch en la placa Arduino que establezca una conexión serial a través del puerto USB.
- En la aplicación Java, se puede utilizar la biblioteca RXTX para comunicarse con el puerto serial para enviar y recibir datos.
- Configurar los parámetros de la conexión serial, como la velocidad de transmisión de datos y el número de bits de datos, paridad de bits de parada.
- Utilizar los métodos de la biblioteca RXTX para enviar y recibir datos a través del puerto serial.
- Al utilizar sockets en Java, se pudo establecer una comunicación bidireccional entre el modelo desarrollado y la placa Arduino. Esto permitió enviar datos desde la aplicación web a la placa Arduino.

7.3. Iluminación

Para la iluminación se utilizará una iluminación anular led de dos filas con orientación de ángulo de 0 ° a 45 ° grados. Iluminación compacta, poca invasiva y de fácil integración en diferentes máquinas Tipos (robots, líneas de producción, máquinas especiales, cintas transportadoras.

Figura 49.

Iluminación anular difusa



Nota: Iluminación difusa. Obtenido de TMSRange (2022). *User manual tms-range, Version 3.1*. (https://sylvania.com.ec/wp-content/uploads/2021/01/Manual-t%C3%A9cnico-de-iluminaci%C3%B3n-Sylvania.pdf)Consultado el 15 de junio de 2022. De dominio público.

7.4. Interfaz gráfica del predictor

La interfaz gráfica del predictor es una representación visual que permite a los usuarios interactuar con el modelo de aprendizaje automático, ingresando datos de entrada y obteniendo predicciones o resultados visualmente legibles, lo que facilita su uso y comprensión sin requerir conocimientos técnicos profundos.

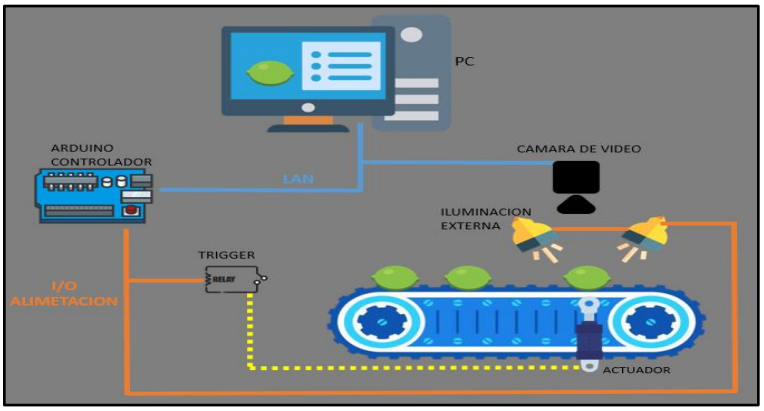
Figura 50.

Interfaz gráfica del predictor en servidor local

Nota: Predictor en servidor local. Elaboración propia.

Figura 51.

Sistema de visión artificial propuesto



Nota: Visión artificial propuesto. Elaboración propia, realizado con Krita art.

CONCLUSIONES

1. La implementación en el control de calidad de Redes Neuronales Convolucionales ha demostrado un incremento del 45 % en la eficiencia del control de calidad, lo que ha contribuido a una notable mejora en la productividad y en la reducción de errores en el proceso de clasificación.

2. Los temas desarrollados en este trabajo de graduación sobre Redes neuronales artificiales y Convolucionales permitirá adquirir habilidades en el diseño e implementación Redes Neuronales Convolucionales en control de calidad, para detectar defectos en productos utilizando esta tecnología de vanguardia.

3. El diseño de la Red Neuronal Convolucional utilizo 50 épocas de entrenamiento, lotes de 16 y una tasa de aprendizaje de 0.001, junto con el uso del 25 % del DataSet como conjunto de validación, estos parámetros optimizan el proceso de aprendizaje y garantizan la capacidad de generalización del modelo.

4. Se realizó una selección del *DataSet* representativo de la población de interés y se dividió en categorías excelente calidad y mala calidad. El análisis de métricas, como la matriz de confusión, reveló una alta precisión del algoritmo con solo un 2 % de error, y tasa de identificación correcta del 98 % en el conjunto de datos de prueba.

5. La implementación de la red neuronal convolucional en el control de calidad mejora la eficiencia y precisión al utilizar la inteligencia artificial

para detectar y clasificar defectos en los productos, optimizando así el proceso de evaluación.

RECOMENDACIONES

1. Implementar sistemas automatizados en controles de calidad utilizando Redes Neuronales Convolucionales, ya que esta tecnología ha demostrado mejorar la clasificación en el control de calidad de manera eficaz y eficiente.

2. Ofrecer a los estudiantes recursos y conocimientos sobre Redes Neuronales y Redes Neuronales Convolucionales, como aplicarlos en procesos de producción, fomentando así el aprendizaje como la creación de soluciones innovadoras para mejorar la eficiencia en la industria.

3. Realizar una investigación exhaustiva para seleccionar la arquitectura de Red Neuronal Convolucional que presente las mejores métricas de clasificación en el proceso de control de calidad.

4. Crear documentación detallada que abarque el proceso de muestreo, procesamiento de imágenes, entrenamiento de la Red Neuronal Convolucional y validación del modelo utilizando métricas de evaluación. Esta documentación asegurará la reproducibilidad, mejora continua y correcto funcionamiento del sistema en la industria.

5. Implementar Redes Neuronales Convolucionales para aprovechar las capacidades de clasificación y contribuir a mejorar el proceso de control de calidad.

REFERENCIAS

Bolaños, R. M. (21 de febrero de 2019). *¿Cómo puede el uso de internet generar más empleos en Guatemala?* Prensa Libre. https://www.prensalibre.com/economia/como-pueden-los-usuarios-de-internet-generar-mas-empleos-en-guatemala/

Cabrera, J. (2005). estudio de mercado del limón persa (citrus latifolio tanaka.), en el municipio de flores, peten. [Tesis de pregrado, Universidad de San Carlos de Guatemala]. Archivo digital. http://biblioteca.usac.edu.gt/tesis/01/01_2219.pdf

Eventos en Conferencistas internacionales llegan a Guatemala para hablar de innovación y tecnología. (n.d.). Guatemala.com. https://eventos.guatemala.com/talleres-conferencias/innovation-technology-expo-guatemala-febrero-2019.html

Ministerio de Agricultura, Ganadería y Alimentación. (septiembre de 2014). *Perfil comercial del limón*. https://www.maga.gob.gt/download/Perfil%20limon.pdf

Raschka, S., Liu, Y., Dzhulgakov, D., & Mirjalili, V. (2022) *Machine Learning with Pytorch and Scikit-Learn: Develop Machine Learning and Deep Learning Models with Python*. [Aprendizaje automático con PyTorch y Scikit-Learn: Desarrolla modelos de aprendizaje automático y aprendizaje profundo con Python] Packt Publishing.

APÉNDICE

Apéndice 1.

Video demostrativo del algoritmo desarrollado para el control de calidad.

https://www.youtube.com/playlist?list=PLjtHOGNCzJvf18xMmWE0bZQtHfhMnYi7f

Presentación del video

Nota. Visualización de imagen del viseo y Link de lista de reproducción de videos demostrativos del control de calidad. Elaboración propia, realizado con Youtube.

Apéndice 2.

Dataset de entrenamiento para red neuronal convolucional

El paso inicial para construir el modelo implica la preparación de un conjunto de datos de excelente calidad. Esto implica contar con un conjunto de ejemplos de entrenamiento que consta de indicaciones de entrada individuales y sus correspondientes salidas deseadas, también conocidas como finalización.

Para ello se ha preparado un conjunto de datos de limones para abordar el problema del control de calidad. El DataSet Contiene 2,076 imágenes de limones (300 x 300 píxeles). Las imágenes de limones se tomaron sobre una superficie blanca.

Esta *DataSet* fue pensada para clasificar tres estados:

- Limones de, excelente calidad
- Limones de, mala calidad
- No hay limones, sin muestra

El conjunto de datos contiene imágenes tanto de limones de mala calidad como de excelente calidad, bajo condiciones de iluminación ligeramente diferentes (todas durante el día) y tamaños variados de limones.

El *DataSet* fue construido utilizando una cámara de alta resolución de 12 megapíxeles con una calidad de imagen de 720p. Seguidamente las imágenes fueron redimensionadas utilizando la herramienta de *Brime* una herramienta online de redimensionamiento de imágenes popular.

Continuación del apéndice 2.

Los limones utilizados fueron limones amarillos (Citrus Medica limoná)

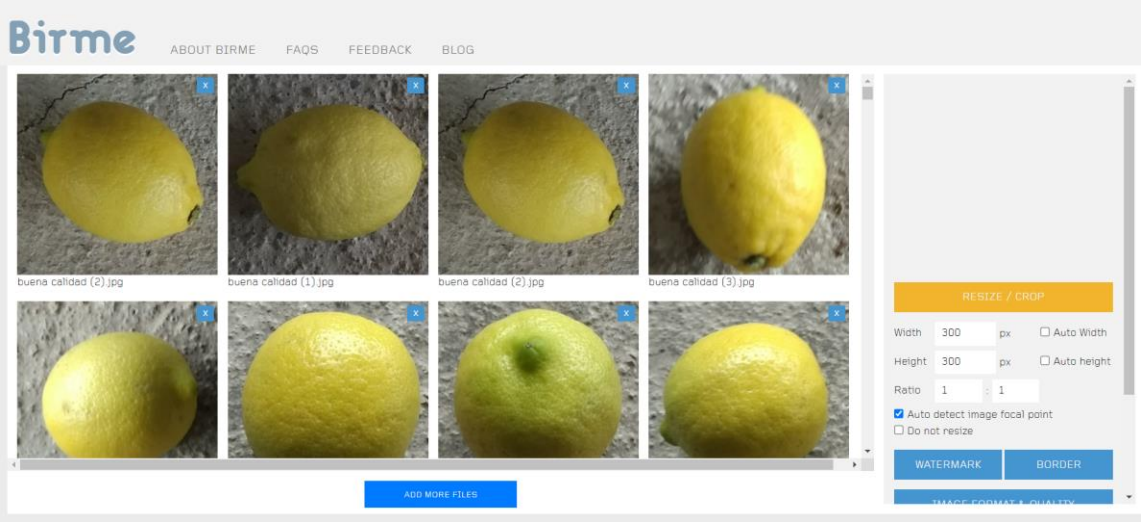
Muestra de una foto de mala calidad de limón amarillo (Citrus Medica limoná)

Muestra de una foto de excelente calidad de limón amarillo (Citrus Medica limoná)

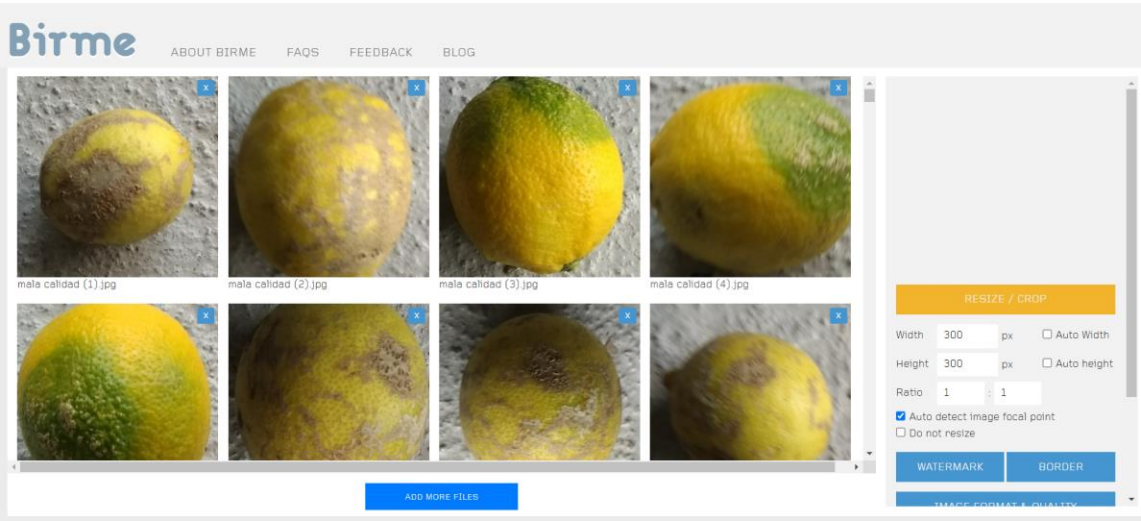
Continuación del apéndice 2.

A continuación, se muestra el redimensionamiento de las fotos a 300x300 pixeles:

Limones de excelente calidad

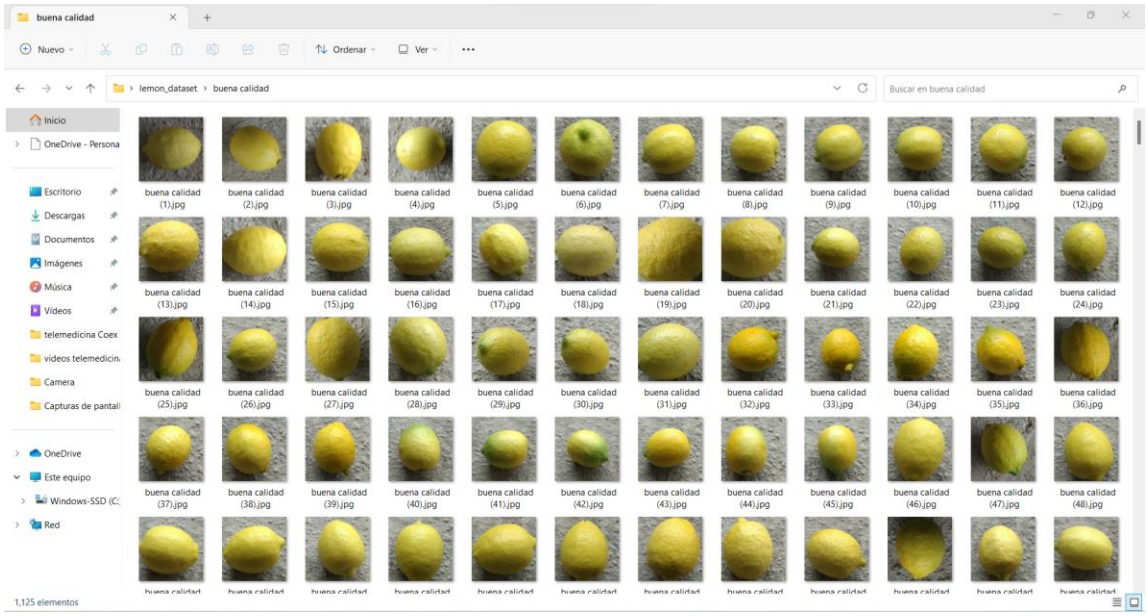


Limones de mala calidad

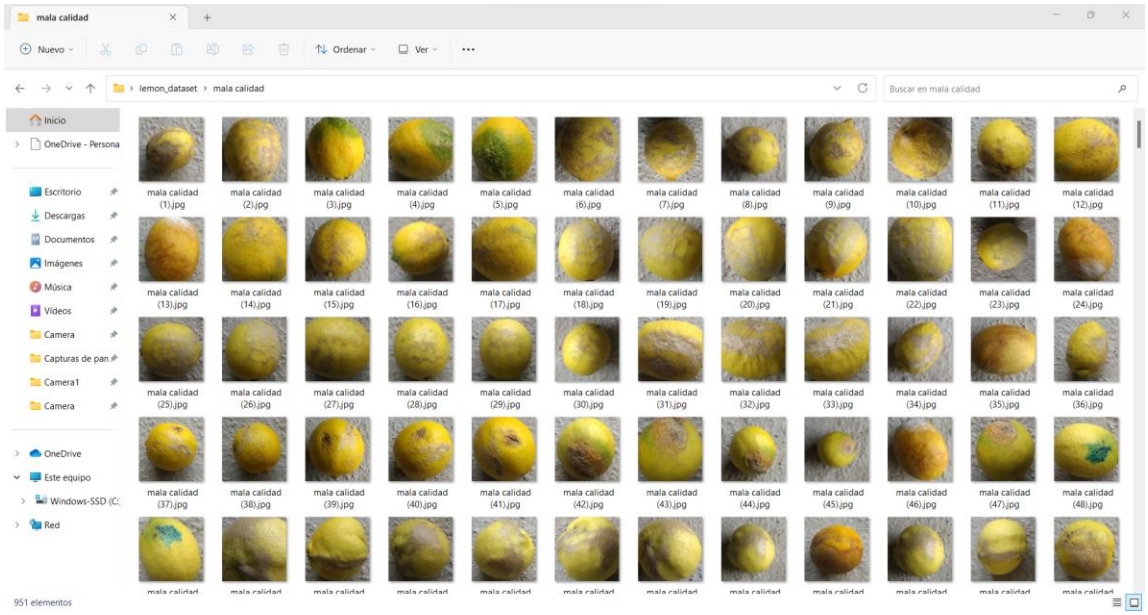


Continuación del apéndice 2.

DataSet de entrenamiento con etiquetado de excelente calidad



DataSet de entrenamiento con etiquetado de mala calidad



Continuación del apéndice 2.

Como se mencionó anteriormente, el *Dataset* contiene un total de 2,076 imágenes de limones, con 1,125 imágenes de excelente calidad y 951 imágenes de mala calidad.

Además, el etiquetado de un *dataset* se refiere al proceso de asignar etiquetas o categorías a cada una de las muestras o ejemplos en el conjunto de datos. Estas etiquetas representan la información deseada o la clasificación a la que pertenece cada muestra. El etiquetado puede realizarse de diferentes maneras, como separar las muestras en diferentes carpetas con nombres que representen las categorías o utilizando archivos de metadatos que asocien las etiquetas a cada muestra.

Nota: Diversidad de etiquetados. Elaboración propia, elaborado en Microsoft Word.

Apéndice 3

Código de red neuronal convolucional para control de calidad

Este bloque de código importa las bibliotecas necesarias para utilizar *TensorFlow* y sus componentes relacionados, como *NumPy* y *Matplotlib*. Estas bibliotecas se utilizan comúnmente para crear y entrenar modelos de aprendizaje automático con *TensorFlow*.

```
[ ]  import tensorflow as tf
     import numpy as np
     from tensorflow.keras import datasets, layers, models, Sequential
     import matplotlib.pyplot as plt

     from tensorflow import keras
     from tensorflow.keras import layers
     from tensorflow.keras.models import Sequential
```

Esta porción de código carga un conjunto de datos de imágenes desde un directorio y lo almacena en la variable *train_ds* para su uso posterior en el entrenamiento de un modelo de aprendizaje automático.

```
[ ]  train_ds = tf.keras.utils.image_dataset_from_directory(
       '/content/drive/MyDrive/IA/train')
```

Este bloque de código carga un conjunto de datos de imágenes desde un directorio y aplica algunas configuraciones adicionales, como la inferencia de etiquetas, la división de datos de entrenamiento y validación, el redimensionamiento de imágenes y el tamaño de lote.

Continuación del apéndice 3.

El conjunto de datos resultante se almacena en la variable ata para su uso posterior en el entrenamiento de un modelo de aprendizaje automático.

```
[ ]  data = tf.keras.utils.image_dataset_from_directory('/content/drive/MyDrive/IA/train', labels='inferred'
     , validation_split=0.2, subset='training', seed=123, image_size=(224, 224), batch_size=32)
```

Por tanto, el tamaño de las imágenes (224,224) en el entrenamiento de redes neuronales es un compromiso entre la eficiencia computacional, las limitaciones de *hardware*, la capacidad de generalización del modelo y la preservación de detalles relevantes. Se deben seleccionar tamaños apropiados teniendo en cuenta el contexto específico y los recursos disponibles.

Este bloque de código crea una figura de *Matplotlib* y muestra una cuadrícula de subgráficos con imágenes y etiquetas tomadas del conjunto de datos. Cada subgráfico muestra una imagen, su etiqueta como título y no tiene ejes visibles.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 10))
for images, labels in data.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(int(labels[i]))
        plt.axis("off")
```

Continuación del apéndice 3.

Esta porción de código define un modelo de CNN usando *Keras* en *TensorFlow*, con capas de normalización, convolución, submuestreo, aplanamiento y capas densas. Se usa la regularización *Dropout* y la última capa realiza una clasificación binaria.

Se especifica la forma de los datos de entrada y se muestra un resumen del modelo con la estructura de las capas y los parámetros entrenables.

Continuación del apéndice 3.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.BatchNormalization(input_shape=(224, 224, 3)),
    tf.keras.layers.Conv2D(filters=96, kernel_size=(11,11), strides=(4,4), activation='relu', input_shape=(224,224,3)),
    tf.keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    tf.keras.layers.Conv2D(filters=256, kernel_size=(5,5), strides=(1,1), activation='relu', padding="same"),
    tf.keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    tf.keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    tf.keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    tf.keras.layers.Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    tf.keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(4096, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(4096, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.build((None, 224, 224, 3))
model.summary()
```

La información Total params muestra la cantidad total de parámetros en el modelo (46,751,117), incluyendo tanto los entrenables como los no entrenables. *Trainable params* se refiere a los parámetros que pueden ser ajustados durante el entrenamiento (46, 751,111), mientras que *Non-trainable params* indica los parámetros que no se modifican durante el entrenamiento, como los de capas predefinidas o congeladas (6).

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 batch_normalization (BatchN  (None, 224, 224, 3)      12
 ormalization)

 conv2d (Conv2D)             (None, 54, 54, 96)        34944

 max_pooling2d (MaxPooling2D  (None, 26, 26, 96)       0
 )

 conv2d_1 (Conv2D)           (None, 26, 26, 256)       614656

 max_pooling2d_1 (MaxPooling  (None, 12, 12, 256)      0
 2D)

 conv2d_2 (Conv2D)           (None, 12, 12, 384)       885120

 conv2d_3 (Conv2D)           (None, 12, 12, 384)       1327488

 conv2d_4 (Conv2D)           (None, 12, 12, 256)       884992

 max_pooling2d_2 (MaxPooling  (None, 5, 5, 256)        0
 2D)

 flatten (Flatten)           (None, 6400)              0

 dense (Dense)               (None, 4096)              26218496

 dropout (Dropout)           (None, 4096)              0

 dense_1 (Dense)             (None, 4096)              16781312

 dropout_1 (Dropout)         (None, 4096)              0
```

Continuación del apéndice 3.

La función `*tf.keras.utils.plot_model()*` se usa para visualizar un modelo de red neuronal, generando un diagrama gráfico. Se especifica el modelo de *Keras* como entrada y se guarda el diagrama en un archivo llamado *complex_model.png*. Al mostrar las formas de entrada y salida de cada capa en el diagrama, se puede comprender mejor la arquitectura de la red, verificar la estructura deseada y compartir la configuración con otros.

```
tf.keras.utils.plot_model(model, to_file='complex_model.png', show_shapes=True)
```

Este bloque de código configura el modelo para utilizar la función de pérdida de entropía cruzada binaria, el optimizador RMSprop y la métrica de precisión durante el entrenamiento y la evaluación del modelo.

```
model.compile(loss = 'binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
```

Esta porción de código realiza el entrenamiento del modelo utilizando los datos de entrada durante 50 épocas, y registra los datos del entrenamiento en un directorio especificado utilizando *TensorBoard*. Esto permite visualizar y analizar el rendimiento del modelo durante el entrenamiento utilizando la interfaz de *TensorBoard*.

```
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir='tf', histogram_freq=1)
epochs = model.fit(data, epochs=50,  verbose = 1,  callbacks=[tensorboard_callback])
```

Continuación del apéndice 3.

Esta porción de código crea una figura con dos *subplots*, donde el primer *subplot* muestra la curva de precisión del modelo y el segundo *subplot* muestra la curva de pérdida del modelo durante el entrenamiento. Esto permite visualizar y analizar la evolución de la precisión y la pérdida del modelo a lo largo de las épocas.

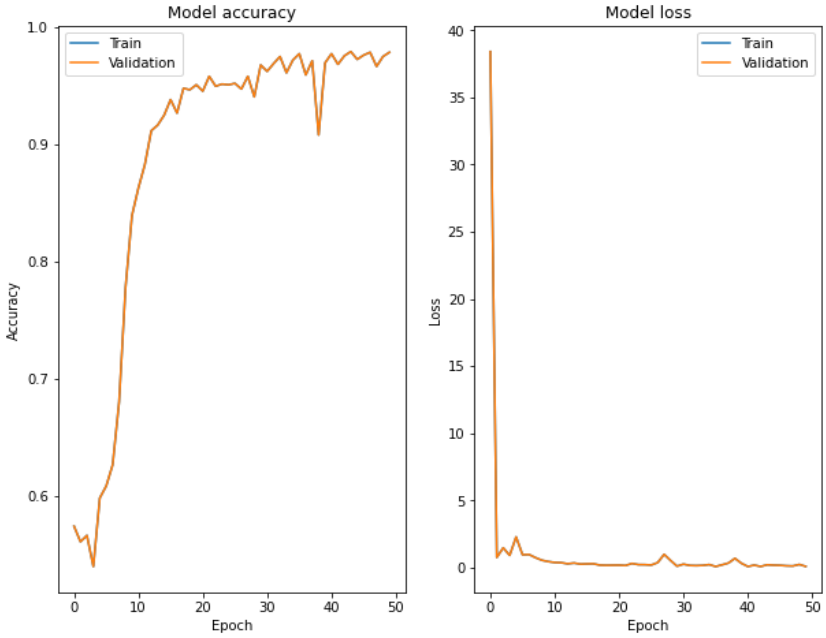
```
fig, axs = plt.subplots(1, 2, figsize=(10, 8))

# Accuracy
axs[0].plot(epochs.history['accuracy'])
axs[0].plot(epochs.history['accuracy'])
axs[0].set_title('Model accuracy')
axs[0].set_ylabel('Accuracy')
axs[0].set_xlabel('Epoch')
axs[0].legend(['Train', 'Validation'])

# Loss
axs[1].plot(epochs.history['loss'])
axs[1].plot(epochs.history['loss'])
axs[1].set_title('Model loss')
axs[1].set_ylabel('Loss')
axs[1].set_xlabel('Epoch')
axs[1].legend(['Train', 'Validation'])

# plt.show()
```

Continuación del apéndice 3.



El modelo en un archivo se puede recuperar más adelante y utilizarlo para realizar predicciones o continuar el entrenamiento desde donde se dejó.

Es una forma de persistir el modelo y compartirlo con otros o utilizarlo en diferentes entornos.

```
model.save('/content/drive/MyDrive/IA/tmp/model')
```

El formato del archivo es .h5, que es un formato comúnmente utilizado para guardar modelos de *Keras*.

```
[ ]  model.save('/content/drive/MyDrive/IA/nuevo2/cnn_clasificatoria.h5')
```

Se instala *Tensorflowjs* para poder migrar el proyecto a *JavaScript* y poder montarlo en una página web.

Continuación del apéndice 3.

```
!pip install tensorflowjs
```

Se convierte el archivo de .h5 a un formato de *JavaScript* para poder montarlo en una página web.

```
!tensorflowjs_converter --input_format=keras '/content/drive/MyDrive/IA/nuevo2/cnn_clasificatoria.h5' '/content/drive/MyDrive/IA/nuevo2/json'
```

Nota: Prototipo de un sistema de automatización y control de proceso. Elaboración propia, realizado en Microsoft Word.

Apéndice 4.

Explicación detallada por línea de código

El código utiliza *tensorflow*, una biblioteca popular para el aprendizaje automático y la inteligencia

---

En esta línea, se importa la biblioteca *TensorFlow* y se asigna el alias *tf* para facilitar su uso en el código.

```
import tensorflow as tf
```

Aquí se importa la biblioteca *NumPy* y se asigna el alias *np. NumPy* es una biblioteca para el cálculo numérico en *Python*, y se utiliza en conjunción con *TensorFlow* para manipular datos numéricos.

```
import numpy as np
```

Estas importaciones permiten acceder a diferentes componentes de *TensorFlow.keras,* que es una API de alto nivel para construir y entrenar modelos de aprendizaje automático con *TensorFlow*. Aquí se importan los módulos `*datasets*`, `*layers*`, `*models*` y `*Sequential*`.

```
from tensorflow.keras import datasets, layers, models, Sequential
```

En esta línea, se importa la biblioteca *Matplotlib* y se asigna el alias *plt*. *Matplotlib* se utiliza para visualizar datos y gráficos.

```
import matplotlib.pyplot as plt
```

Continuación del apéndice 4.

```

Estas líneas adicionales importan los mismos módulos que se importaron anteriormente, pero utilizando una sintaxis ligeramente diferente. Es posible que se hayan importado nuevamente para asegurarse de que todas las dependencias estén correctamente incluidas en el código.

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

Este bloque de código importa las bibliotecas necesarias para utilizar *TensorFlow* y sus componentes relacionados, como *NumPy* y *Matplotlib*. Estas bibliotecas se utilizan comúnmente para crear y entrenar modelos de aprendizaje automático con *TensorFlow*.

---

Utiliza la función *image_dataset_from_directory* de *TensorFlow.keras* para cargar un conjunto de datos de imágenes desde un directorio.

Aquí, *train_ds* es el nombre de la variable que se utiliza para almacenar el conjunto de datos resultante.

*'/content/drive/MyDrive/IA/train'* es la ruta al directorio que contiene las imágenes de entrenamiento. La función *image_dataset_from_directory* explorará este directorio y cargará todas las imágenes encontradas en el conjunto de datos.
```

Continuación del apéndice 4.

Utiliza la función *image_dataset_from_directory de TensorFlow.keras* para cargar un conjunto de datos de imágenes desde un directorio con opciones adicionales de configuración.

'inferred' en *labels='inferred'*: este parámetro indica que las etiquetas del conjunto de datos se infieren automáticamente a partir de los nombres de las subcarpetas en el directorio de entrenamiento. Cada subcarpeta se considera una clase distinta y se le asigna un valor de etiqueta único.

validation_split=0.2: este parámetro especifica el porcentaje de datos que se utilizarán para validación. En este caso, el 20 % de los datos se utilizarán para la validación, mientras que el 80 % restante se utilizará para el entrenamiento del modelo.

subset='training': este parámetro indica que solo se carguen los datos de entrenamiento y se excluyan los datos de validación.

seed=123: este parámetro establece una semilla para la generación de números aleatorios. Esto asegura que los datos se dividan en entrenamiento y validación de manera consistente cada vez que se ejecute el código.

image_size=(224, 224): este parámetro especifica el tamaño al que se redimensionarán las imágenes cargadas. En este caso, las imágenes se redimensionarán a un tamaño de 224x224 píxeles.

Es preciso agregar que, el tamaño de las imágenes en el entrenamiento de redes neuronales es importante debido a la eficiencia computacional, las

Continuación del apéndice 4.

Limitaciones de *hardware*, la regularización, el tamaño de las capas convolucionales y la preservación de detalles, lo que afecta la cantidad de datos, la complejidad computacional, la memoria requerida, la generalización del modelo y la capacidad de capturar características relevantes.

batch_size=32:

Este parámetro determina el tamaño de lote utilizado durante el entrenamiento del modelo. El conjunto de datos se dividirá en lotes de 32 imágenes cada uno.

Continuación del Apéndice 4.

Esta porción de código utiliza la biblioteca *Matplotlib* para visualizar un conjunto de datos de imágenes y sus etiquetas. En primer lugar, se importa la biblioteca *Matplotlib* y se asigna el alias *plt*. Luego, se crea una nueva figura con un tamaño de 10x10 pulgadas utilizando *plt.figure* (*figsize*=(10, 10)). Esto establece el tamaño de la figura en la que se mostrarán las imágenes.

import matplotlib.pyplot as plt plt.figure(figsize=(10, 10))

Este bucle itera sobre el conjunto de datos data (suponiendo que data es un conjunto de datos de imágenes y etiquetas) y toma solo el primer lote.

De imágenes y etiquetas. El tamaño del lote se determina por el número de iteraciones del bucle externo.

for images, labels in data.take(1):

Dentro del bucle interno, se realiza lo siguiente para cada imagen y etiqueta en el lote:

Continuación del apéndice 4.

Se crea un nuevo subgráfico utilizando *plt.subplot(3, 3, i + 1).* El primer argumento indica el número de filas de subgráficos, el segundo argumento indica el número de columnas y el tercer argumento indica el índice del subgráfico actual.

Se muestra la imagen utilizando *plt.imshow(images[i].numpy().astype(uint8)).* La imagen se obtiene del lote de imágenes y se convierte a un *array NumPy* antes de mostrarla.

Se establece el título del subgráfico con la etiqueta correspondiente a la imagen utilizando *plt.title(int(labels[i])).*

Se desactivan los ejes utilizando *plt.axis(off),* lo que significa que no se mostrarán los ejes x e y en el subgráfico.

Este bloque de código define un modelo de red neuronal convolucional (CNN) utilizando la API secuencial de *Keras* en *TensorFlow*. A continuación, se explica la función de cada capa en el modelo:

tf.keras.layers.BatchNormalization: esta capa realiza una normalización de lotes para normalizar los valores de entrada y estabilizar el proceso de entrenamiento.

tf.keras.layers.Conv2D: esta capa *convolucional* aplica filtros *convolucionales* a las imágenes de entrada. Se especifica el número de filtros

Continuación del apéndice 4.

(*filters*), el tamaño del *kernel* (*kernel_size*), los pasos de desplazamiento (*strides*), la función de activación (activation) y el relleno (*padding*) utilizado.

tf.keras.layers.MaxPool2D: esta capa de *max pooling* realiza un submuestreo máximo para reducir el tamaño de las características.

tf.keras.layers.Flatten: esta capa aplana la salida de la capa anterior en un vector unidimensional.

tf.keras.layers.Dense: estas capas densas son capas totalmente conectadas que aplican una transformación lineal seguida de una función de activación. Se especifica el número de unidades (*units*) y la función de activación (*activation*).

tf.keras.layers.Dropout: esta capa de *dropout* aplica regularización por abandono, descartando aleatoriamente algunas neuronas durante el entrenamiento para reducir el sobreajuste.

tf.keras.layers.Dense: la última capa densa tiene una única unidad y utiliza una función de activación sigmoide para realizar la clasificación binaria.

Después de definir el modelo, se llama al método *build* para especificar la forma de los datos de entrada. Se utiliza (*None*, 224, 224, 3) para indicar que el modelo puede aceptar lotes de imágenes de cualquier tamaño con dimensiones 224x224 y 3 canales.

Continuación del apéndice 4.

La capa *BatchNormalization* normaliza los valores de entrada para estabilizar el proceso de entrenamiento.

Las capas *Conv2D* aplican filtros convolucionales a las imágenes de entrada para extraer características.

Las capas *MaxPool2D* realizan submuestreo máximo para reducir el tamaño de las características.

La capa *Flatten* transforma la salida anterior en un vector unidimensional.

Las capas *Dense* son capas totalmente conectadas que aplican una transformación lineal seguida de una función de activación.

La capa *Dropout* aplica regularización por abandono, descartando aleatoriamente neuronas durante el entrenamiento para reducir el sobreajuste.

La última capa *Dense* tiene una única unidad y utiliza una función de activación *sigmoide* para realizar una clasificación binaria.

Después de definir el modelo, se llama al método *build* para especificar la forma de los datos de entrada, permitiendo lotes de imágenes de cualquier tamaño con dimensiones 224x224 y 3 canales. Finalmente, se muestra un resumen del modelo que muestra la estructura de las capas y el número de parámetros entrenables.

Continuación del apéndice 4.

La función `*tf.keras.utils.plot_model()*` se utiliza para visualizar un modelo de red neuronal. Toma como entrada un modelo de *Keras* y genera una representación gráfica del mismo en forma de un diagrama.

En este caso, se está utilizando la función `*plot_model*` para representar el modelo llamado `*model*`. El parámetro `*to_file='complex_model.png'*` indica que el diagrama se guardará en un archivo llamado *complex_model.png*. Y el parámetro `*show_shapes*=True` indica que se deben mostrar las formas de entrada y salida de cada capa en el diagrama.

Continuación del Apéndice 4.

La representación gráfica del modelo puede ser útil para comprender mejor la arquitectura de la red neuronal, visualizar la conexión entre las capas y verificar que la estructura del modelo sea la deseada. Además, también permite compartir y comunicar de manera efectiva la configuración del modelo con otras personas.

La porción de código *model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])* configura el proceso de compilación de un modelo de red neuronal en *Keras*.

loss='binary_crossentropy' especifica la función de pérdida que se utilizará durante el entrenamiento del modelo. En este caso, se utiliza la función de entropía cruzada binaria, que es comúnmente utilizada en problemas de clasificación binaria.

Continuación del apéndice 4.

optimizer='rmsprop' indica el algoritmo de optimización que se utilizará para ajustar los pesos del modelo durante el entrenamiento. En este caso, se utiliza el optimizador *RMSprop*, que es un algoritmo popular para la optimización de redes neuronales.

metrics=['accuracy'] define las métricas que se utilizarán para evaluar el desempeño del modelo durante el entrenamiento y la evaluación. En este caso, se utiliza la métrica de precisión (*accuracy*), que calcula la proporción de predicciones correctas en relación con el total de muestras.

La porción de código realiza el entrenamiento de un modelo de red neuronal utilizando datos de entrada `*data*` durante 50 épocas. Además, utiliza el *callback* `*TensorBoard*` para registrar los datos del entrenamiento en un directorio especificado.

- `*tf.keras.callbacks.TensorBoard(log_dir='tf', histogram_freq=1)*`*:* esta línea de código crea una instancia del *callback* `*TensorBoard*` que se utilizará durante el entrenamiento del modelo. El parámetro `*log_dir*` especifica el directorio donde se guardarán los registros de *TensorBoard*. En este caso, se utiliza el directorio *tf.*

El parámetro `*histogram_freq*` se establece en 1 para registrar histogramas de las activaciones de las capas durante el entrenamiento.

-`*epochs = model.fit(data, epochs=50, verbose=1, callbacks=[tensorboard_callback])*`: esta línea de código inicia el entrenamiento del modelo. El método `*fit*` recibe como entrada los datos `data` y se entrena

Continuación del apéndice 4.

Durante 50 épocas. El parámetro `*verbose*=1` indica que se mostrará información detallada durante el entrenamiento. El parámetro `*callbacks=[tensorboard_callback]*` se utiliza para especificar los *callbacks* a utilizar durante el entrenamiento, en este caso, se incluye el *callback* de *TensorBoard* creado anteriormente.

La porción de código realiza la visualización de la precisión y la pérdida del modelo durante el entrenamiento.

-`*fig, axs = plt.subplots(1, 2, figsize=(10, 8))*`*:* esta línea de código crea una figura y dos *subplots* en una sola fila. El parámetro `(1, 2)` indica que se crearán dos *subplots*, y `*figsize*=(10, 8)` establece el tamaño de la figura.

- `*axs[0].plot(epochs.history['accuracy'])*`*:* esta línea de código traza la precisión del modelo en el primer *subplot (`axs[0]`).* Utiliza los datos de precisión almacenados en el historial de entrenamiento *(`epochs.history['accuracy']`).*

- `*axs[0].set_title('Model accuracy')*`*:* esta línea de código establece el título del primer *subplot* como *Model accuracy.*

- `*axs[0].set_ylabel('Accuracy')*`*:* esta línea de código establece la etiqueta del eje y del primer *subplot* como *Accuracy*.

- `*axs[0].set_xlabel('Epoch')*`*:* esta línea de código establece la etiqueta del eje x del primer *subplot* como *Epoch*.

Continuación del apéndice 4.

- `*axs[0].legend(['Train', 'Validation'])*`*:* esta línea de código agrega una leyenda al primer *subplot* para indicar las curvas de entrenamiento y validación.

- `*axs[1].plot(epochs.history['loss'])*`*:* esta línea de código traza la pérdida del modelo en el segundo subplot *(`axs[1]`).* Utiliza los datos de pérdida almacenados en el historial de entrenamiento *(`epochs.history['loss']`).*

- `*axs[1].set_title('Model loss')*`*:* esta línea de código establece el título del segundo *subplot* como *Model loss.*

- `*axs[1].set_ylabel('Loss ')*`*:* esta línea de código establece la etiqueta del eje y del segundo *subplot* como *Loss*.

- `*axs[1].set_xlabel('Epoch')*`*:* esta línea de código establece la etiqueta del eje x del segundo *subplot* como *Epoch*.

- `*axs[1].legend(['Train', 'Validation'])*`*:* esta línea de código agrega una leyenda al segundo *subplot* para indicar las curvas de entrenamiento y validación.

- `*# plt.show()*`*:* esta línea de código está comentada, pero si se descomenta, mostrará la figura completa con los *subplots* y las curvas de precisión y pérdida.

---

model.save('/content/drive/MyDrive/IA/tmp/model') guarda el modelo en un archivo en el sistema de archivos. En este caso, el modelo se guarda en la ruta especificada*: /content/drive/MyDrive/IA/tmp/model.*

Continuación del apéndice 4.

---

model.save('/content/drive/MyDrive/IA/nuevo2/cnn_clasificatoria.h5')

guarda el modelo en un archivo en el sistema de archivos. En este caso, el modelo se guarda en la ruta especificada: /content/drive/MyDrive/IA/nuevo2/cnn_clasificatoria.h5.

Nota: Explicación detallada por línea de código. Elaboración propia, realizado en Microsoft Word.