



UNIVERSIDAD NACIONAL DEL ALTIPLANO
FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA,
ELECTRÓNICA Y SISTEMAS
ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA



**“DISEÑO DE UN SISTEMA DE CLASIFICACIÓN DE LIMONES
USANDO UNA RED NEURONAL CONVOLUCIONAL EN EL
MERCADO UNIÓN Y DIGNIDAD DE LA CIUDAD DE PUNO –
2022”**

TESIS

PRESENTADA POR:

JHAROL IVÁN CENTENO PALOMINO

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PUNO – PERÚ

2023



Reporte de similitud

NOMBRE DEL TRABAJO

DISEÑO DE UN SISTEMA DE CLASIFICACIÓN DE LIMONES USANDO UNA RED NEURONAL CONVOLUCIONAL EN EL MERCADO UNIÓN Y DIGNIDAD DE LA CIUDAD DE PUNO - 2022

AUTOR

JHAROL IVÁN CENTENO PALOMINO

RECuento DE PALABRAS

21008 Words

RECuento DE CARACTERES

119719 Characters

RECuento DE PÁGINAS

119 Pages

TAMAÑO DEL ARCHIVO

3.0MB

FECHA DE ENTREGA

May 4, 2023 11:11 PM GMT-5

FECHA DEL INFORME

May 4, 2023 11:13 PM GMT-5

● 15% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base

- 12% Base de datos de Internet
- 5% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossref
- 10% Base de datos de trabajos entregados

● Excluir del Reporte de Similitud

- Material bibliográfico
- Material citado


Christian Augusto Pomero Gozquez
INGENIERO ELECTRÓNICO
CIP 133003


JAMES ROLANDO PAREDONDO MAMANI
CIP 122404
SUB DIRECTOR DE INVESTIGACIÓN
E.P. INGENIERÍA ELECTRÓNICA

Resumen



DEDICATORIA

A mis padres, quienes siempre me apoyaron, y a todas las personas que me prestaron ayuda durante este tiempo.

Jharol Iván



AGRADECIMIENTOS

En primer lugar, agradecer a Dios, por permitirme seguir adelante cada día.

De forma muy especial agradezco a mi asesor M. Sc. Christian Augusto Romero Goyzuela por su apoyo incondicional en todo el desarrollo de esta investigación.



ÍNDICE GENERAL

DEDICATORIA

AGRADECIMIENTOS

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ACRÓNIMOS

RESUMEN 14

ABSTRACT..... 15

CAPITULO I

INTRODUCCIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA 17

1.2. HIPÓTESIS 17

1.3. OBJETIVO GENERAL..... 17

1.4. OBJETIVOS ESPECÍFICOS..... 17

CAPITULO II

REVISIÓN DE LITERATURA

2.1. ANTECEDENTES DE LA INVESTIGACIÓN 18

2.2. LIMÓN 22

2.3. PRINCIPALES PRODUCTORES DE LIMÓN EN EL PERÚ 22

**2.4. PROCESO DE PRODUCCIÓN PARA LA DISTRIBUCIÓN NACIONAL Y
EXTRANJERA..... 22**

2.4.1. Etapa agrícola 22

2.4.2. Etapa agroindustrial 23

2.4.3. Operación logística 23



2.5. DISPOSICIONES CON RELACIÓN A LA CALIDAD DEL LIMÓN	23
2.6. CONCEPTOS COMPLEMENTARIOS	24
2.6.1. Redes neuronales artificiales	24
2.6.2. Tensor	26
2.6.3. Red neuronal convolucional	27
2.6.4. Capas de una red neuronal convolucional	28
2.6.5. Función de activación	31
2.6.6. Pre - procesamiento de una imagen	35
2.6.7. Entrenamiento de la red neuronal	37
2.7. PYTHON.....	40
2.8. LIBRERÍAS	41
2.8.1. Tensorflow	41
2.8.2. Keras	43
2.8.3. Numpy	43
2.8.4. Python imaging library (PIL)	44
2.8.5. Path lib	45
2.8.6. Mathplotlib	45
2.8.7. Os.....	46
2.8.8. Csv	48
2.8.9. Google colab.....	48
2.8.10. Google drive	49
2.9. RASPBERRY PI.....	50
2.9.1. Cámara pi.....	52
2.9.2. Sistema operativo raspbian.....	53
2.9.3. Cli raspberry	54



2.9.4. Protocolo ssh	56
2.9.5. Vnc viewer.....	57
2.10. ENRUTADOR	58
2.11. SERVIDOR.....	62
2.11.1. Samba	64
2.12. DATASET	64
2.13. ANACONDA NAVIGATOR.....	65

CAPITULO III

MATERIALES Y MÉTODOS

3.1. MATERIALES	69
3.1.1. Hardware	69
3.1.2. Software.....	70
3.1.3. Caja de adquisición de datos	70
3.2. DISEÑO Y NIVEL DE LA INVESTIGACIÓN	71
3.2.1. Diseño de la investigación.....	71
3.2.2. Nivel de la investigación	72
3.3. POBLACIÓN Y MUESTRA DE LA INVESTIGACIÓN	72
3.4. UBICACIÓN DE LA INVESTIGACIÓN	73
3.5. TÉCNICAS E INSTRUMENTOS PARA LA RECOLECCION DE DATOS	74
3.5.1. Técnicas	74
3.5.2. Instrumentos	75

CAPITULO IV

RESULTADOS Y DISCUSIÓN

4.1. DATASET	76
4.2. IMPORTACIÓN DE LIBRERÍAS	78



4.3. EXPLORACIÓN DE LOS DATOS	82
4.4. DATOS CARGADOS CON KERAS	85
4.5. VISUALIZACIÓN DE LOS DATOS.....	89
4.6. CONFIGURACIÓN DE LOS DATOS PARA EL RENDIMIENTO.....	94
4.7. ESTANDARIZACIÓN DE LOS DATOS.....	95
4.8. CREACIÓN DEL MODELO	95
4.9. COMPILACIÓN DEL MODELO	98
4.10. RESUMEN DEL MODELO	99
4.11. ENTRENAMIENTO DEL MODELO.....	100
4.12. VISUALIZACIÓN DE LOS RESULTADOS DEL ENTRENAMIENTO ...	101
4.13. PREDICCIÓN DE NUEVOS DATOS.....	104
V. CONCLUSIONES.....	107
VI. RECOMENDACIONES	108
VII. REFERENCIAS BIBLIOGRÁFICAS.....	109
ANEXOS.....	115

Área: Telecomunicaciones y Telemática

Tema: Redes Neuronales

FECHA DE SUSTENTACIÓN: 23 mayo de 2023



ÍNDICE DE FIGURAS

Figura 1: Redes Neuronales Artificiales.....	25
Figura 2: Etapas de una red neuronal convolucional.....	28
Figura 3: Operación de convolución	29
Figura 4: Max Pooling.....	30
Figura 5: Average Pooling.....	31
Figura 6: Función de activación	31
Figura 7: Función sigmoide.....	32
Figura 8: Función tangente Hiperbólica	33
Figura 9: Función de activación RELU	34
Figura 10: Imagen dividida en cuadrículas.....	36
Figura 11: Imagen dividida en cuadrículas.....	38
Figura 12: Taza de aprendizaje.....	39
Figura 13: Manejo de rutas de un sistema operativo	45
Figura 14: Raspberry Pi.....	50
Figura 15: Cámara Pi.....	52
Figura 16: Interfaz de línea de comandos de Raspberry Pi	55
Figura 17: Funcionamiento del protocolo de comunicación SSH.....	57
Figura 18: Router TL-WR940N	59
Figura 19: Servidor.....	63
Figura 20: Caja de adquisición de datos, con medidas de 35 x 35 x 35 <i>cm</i> ³	70
Figura 21: Iluminación con luces led de 12V.....	71
Figura 22: Mercado Unión y Dignidad de la ciudad de Puno	74
Figura 23: Código de Python para el recorte de imágenes	77



Figura 24: Los resultados del recorte de limones de la categoría “Limones en mal estado”.....	77
Figura 25: Los resultados del recorte de limones de la categoría “Limones en buen estado”.....	78
Figura 26: Importación de librerías de Python	81
Figura 27: Versión de TensorFlow instalado	82
Figura 28: Importamos el drive al Google colab.....	82
Figura 29: Se lista el contenido de la carpeta Train	83
Figura 30: Asignación de la ruta de la carpeta train a la variable data_dir	84
Figura 31: Se muestran la cantidad de Limones Buenos y Limones Malos.....	85
Figura 32: Asignación de un batch size de 32 al modelo de la red.	86
Figura 33: Preprocesamiento de las imágenes de Train	88
Figura 34: Preprocesamiento de las imágenes de Val.	88
Figura 35: Impresión de las clases de los limones	89
Figura 36: Parámetros de figura de matplotlib	89
Figura 37: Código para mostrar 9 imágenes del quinto grupo de datos.....	92
Figura 38: Limones etiquetados del quinto grupo.....	92
Figura 39: Código para mostrar 9 imágenes del sexto grupo de datos.....	93
Figura 40: Limones etiquetados del sexto grupo.....	93
Figura 41: Mostramos en tensor de (32,180,180,3).....	94
Figura 42: Caché y Prefetch	94
Figura 43: Estandarización de los datos que están de 0 a 1.....	95
Figura 44: Aplicación mediante Dataset.map	95
Figura 45: Creación del modelo de red neuronal	98
Figura 46: Compilación del modelo de red neuronal convolucional	99



Figura 47: Resumen del modelo de la red neuronal	99
Figura 48: Entrenamiento del modelo	101
Figura 49: Código para la visualización de resultados	101
Figura 50: Precisión del modelo de red neuronal	102
Figura 51: Disminución de error de la red neuronal.....	103
Figura 52: Importación de nuevos datos y la respectiva predicción.....	104



ÍNDICE DE TABLAS

Tabla 1: Características de Raspberry Pi.....	50
Tabla 2: Características de la Cámara Pi.....	53
Tabla 3: Características de sistema operativo Raspbian.....	54
Tabla 4: Características de Hardware de Router Tp-Link TL-WR940N	59
Tabla 5: Características de Router Tp-Link	60
Tabla 6: Características de Software de Router Tp-Link TL-WR940N.....	61
Tabla 7: Resultados de clasificación con imágenes nuevas.	105
Tabla 8: Resultados de clasificación con imágenes nuevas.	106



ÍNDICE DE ACRÓNIMOS

CNN	: Red neuronal Convolutacional
COLAB	: Colaboratory
SENASA	: Servicio de Sanidad Agraria del Perú
DP	: Deep Learning
SO	: Sistema Operativo
SSH	: Secure Shell (Interprete de órdenes seguro)
LAN	: Red de área local (Local Área Network)
CPU	: Unidad Central de Procesamiento (Central Processing Unit)
GPU	: Unidad de Procesamiento Gráfico (Graphics Processing Unit)
TPU	: Unidad de Procesamiento Tensorial (Tensor Processing Unit)



RESUMEN

En esta investigación, se explora el uso de las redes neuronales convolucionales (CNN) para la clasificación de limones en el mercado unión y dignidad de la ciudad de Puno, con fecha de ejecución 07/06/2022 – 06/12/2022. En la actualidad los sistemas de clasificación mejoran en gran medida el problema de la velocidad y la eficiencia de las personas. La presente investigación se centra en la creación de un modelo de clasificación de imágenes de limones en dos categorías: limones en buen estado y limones en mal estado. Las redes neuronales necesitan entrenarse, es por ello que se genera un conjunto de 1000 imágenes para cada categoría. Finalmente se desarrolla el modelo en Google Colaboratory (COLAB) y los resultados de precisión que se obtienen en el modelo se calcula dividiendo el número de casos favorables sobre el número de casos totales, obteniendo como resultado un 90% de precisión del modelo, esto demuestra la efectividad del sistema logrado, el sistema puede ser mejorado lo que conlleva a una serie de recomendaciones para mejoras del modelo.

Palabras clave: Red neuronal convolucional, código, drive, modelo, clasificación.



ABSTRACT

In this research, we explore the use of convolutional neural networks (CNN) for the classification of lemons in the Union and Dignity market of the city of Puno, with execution dates from 07/06/2022 to 06/12/2022. Currently, classification systems greatly improve the problem of speed and efficiency of people. This research focuses on creating a lemon image classification model into two categories: lemons in good condition and lemons in bad condition. Neural networks need to be trained, which is why a set of 1000 images are generated for each category. Finally, the model is developed in Google Colaboratory (COLAB), and the accuracy results obtained are calculated by dividing the number of favorable cases by the total number of cases, resulting in a 90% model accuracy, which demonstrates the effectiveness of the achieved system. The system can be improved, leading to a series of recommendations for model improvements.

Keywords: Convolutional neural network, code, drive, model, classification.



CAPITULO I

INTRODUCCIÓN

La inteligencia artificial ha cambiado la forma en que las personas viven, trabajan y se comunican. Estas tecnologías que surgen día a día tienen el potencial de ayudar a millones de personas a vivir una vida más feliz, saludable y productiva.

La inteligencia artificial tiene dentro de ella varias ramas y una de ellas son las redes neuronales. El termino red neuronal es un método de la inteligencia artificial que enseña a una computadora a procesar datos de una manera que está inspirada en el cerebro humano.

La identificación de imágenes u objetos es uno de los problemas de la inteligencia artificial. Inicialmente la identificación de imágenes se realizaba con redes neuronales comunes, en donde era suficiente para identificar imágenes, pero el rendimiento disminuía cuando se aumentaba el tamaño de dichas imágenes, la solución al problema se da con las redes neuronales convolucionales (CNN) que es un algoritmo de aprendizaje profundo que puede tomar una imagen de entrada, asignar pesos y sesgos a varios aspectos/objetos en la imagen y ser capaz de diferenciar uno de otro.

Las redes neuronales convolucionales (CNN) nos ayudan a categorizar grandes cantidades de imágenes, para entender su funcionamiento debemos tener en cuenta que tiene tres capas: capa de convolución, capa de agrupación de datos y las capas totalmente conectadas.

En esta investigación se diseña un sistema de clasificación de limones usando una red neuronal convolucional (CNN), adicionalmente se genera un conjunto de datos que consiste en imágenes de limones etiquetadas como: limones buenos y limones malos.



1.1. PLANTEAMIENTO DEL PROBLEMA

¿cómo realizar el diseño de un sistema de clasificación de limones usando una red neuronal convolucional en el mercado unión y dignidad de la ciudad de Puno?

1.2. HIPÓTESIS

El diseño de un sistema de clasificación de limones usando una red neuronal convolucional permitirá la clasificación de limones en el mercado unión y dignidad de la ciudad de Puno.

1.3. OBJETIVO GENERAL

Diseñar un sistema de clasificación de limones usando una red neuronal convolucional para el mercado unión y dignidad de la ciudad de Puno.

1.4. OBJETIVOS ESPECÍFICOS

- Generar un grupo de datos que consiste en imágenes de limones del mercado unión y dignidad de la ciudad de Puno.
- Diseñar un sistema de clasificación de imágenes de limones usando una red neuronal convolucional.



CAPITULO II

REVISIÓN DE LITERATURA

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Según la investigación “Sistema de visión artificial basado en redes neuronales convolucionales para la selección de arándanos según estándares de exportación”, realizado por Willy Aldo Narciso Horna y Edgas Andree Manzano Ramos. En su investigación diseñaron un sistema de visión artificial a través de redes neuronales convolucionales (CNN) en el cual se aplica el modelo R-CNN con el propósito de desarrollar una alternativa para la automatización de la selección de arándanos para la exportación. En esta investigación el entrenamiento de la red neuronal demoró 5 horas donde entrenó 41400 pasos y logro un rango de precisión del 24% al 40.5% para cinco defectos de arándanos considerados (Narciso Horna & Manzano Ramos, 2021).

Según la investigación “clasificación automática de citrus aurantifolia usando visión artificial”, realizado por Fredy Daniel Alcarazo Ibàñez, en la universidad de señor de Sipán, en el año 2021. En esta investigación la caracterización de las clases de citrus aurantifolia se realizó gracias a consultas a expertos en la materia. Esto permitió obtener una matriz con el color RGB promedio de las clases maduro, pintón y verde, con base a esa información realizar la clasificación mediante técnicas de visión artificial. Para la obtención de las imágenes sin excesos de brillos se utilizó técnicas que logran ese objetivo, como la altura y la minimización de reflejos de luz. Finalmente se usa la técnica de clasificación SVM y KNN, estas lograron un grado de precisión alto en la clasificación de citrus aurantifolia. En el entrenamiento se utilizó 80% de las características extraídas y se utilizó un 20% para evaluar el rendimiento de las técnicas mediante indicadores de la sensibilidad, especificad, exactitud y precisión (Alcarazo Ibañez, 2021).



Según la investigación “sistema de control de calidad utilizando redes neuronales para la clasificación del estado de la granadilla”, realizado por Bryan Cristhian Martin Cuya Marzal y Martin Guillermo Ramos Lugo, desarrollado en la universidad de Lima, en el año 2020. Se discutió la problemática actual de la propagación de enfermedades altamente contagiosas y se propuso una solución potencial para mejorar el proceso de verificación de calidad posterior a la recolección en el sector agrícola. Esta solución implica el uso de tecnología renovadora, como el Internet de las cosas, que conecta una cámara con un ordenador para capturar imágenes de las granadillas y enviarlas sin conexión física para su tratamiento en una red neuronal convolucional que separa la producción como válida o inválida. La implementación de esta solución sería beneficiosa para los negocios agrícolas que han sido afectados por las medidas gubernamentales y requeriría una solución de bajo costo. El pensamiento de diseño es una metodología efectiva para abordar este tipo de problemas y tiene una perspectiva centrada en los usuarios. El modelo de clasificación alcanzó una precisión del 97,97%, con mejores resultados utilizando un filtro pequeño. Se recomienda que las imágenes utilizadas para la formación de redes tengan el enfoque, iluminación y entorno adecuados. (Cuya Marzal & Ramos Lugo, 2020).

Según la investigación “Clasificación de frutas basadas en redes neuronales convolucionales”, realizado por Jonathan Victor Aguilar-Alvarado y Milton Alfredo Campoverde-Molina, realizado en el año 2020. Según este estudio, es importante utilizar imágenes de alta resolución para obtener un buen promedio de eficiencia en un modelo de red neuronal. En la investigación, se logró una eficiencia del 87%, y se concluyó que reducir las imágenes a 224 * 224 px disminuye la eficacia de conocimiento. Además, en casos de frutas con similitudes, se necesitan numerosas imágenes de alta resolución. Afortunadamente, no se necesitan recursos computacionales altos para procesar El



adiestramiento de una red neuronal convolucional. Para que el modelo CNN de MobileNetV1 sea eficiente, es necesario alcanzar una efectividad de al menos el 86% durante el entrenamiento. (Aguilar-Alvarado & Campoverde-Molina, 2020).

Según la investigación “Clasificación de mamografías mediante redes neuronales convolucionales”, realizado por Mayra Cristina Berrones Reyes, realizado en la universidad autónoma de nuevo león – Mexico, en el año 2019. Logró ver que Adagrad no resultó ser un buen optimizador, ya que era recomendado por muchos artículos. Se identificaron múltiples soluciones a este problema, una de las cuales sugiere que los parámetros utilizados en el algoritmo pueden no ser adecuados para la información que se está procesando. A través de la revisión de varios libros de investigación, se encontró que aumentar la cantidad de conjuntos de datos empleados en el proceso de adiestramiento puede mejorar la eficacia del modelo, pero también puede afectar negativamente la precisión. Se descubrió que el optimizador SGD produce los mejores resultados, aunque su uso puede prolongar el tiempo de entrenamiento, pero proporciona una mayor confiabilidad. (Berrones Reyes, 2019).

Según la investigación “Sistema de clasificación de Frutas de mango usando red neuronal y visión por computadora”, realizado por Emny Harna Yossy, Jhonny Pratana, Tommy Wijaya, Heri Hermawan, Widodo Budiharto, en el año 2017. Concluyó que el modelado de este sistema de clasificación de frutos de mango utilizará la red neuronal y la técnica de visión por computadora para clasificar los mangos correctamente con una alta precisión que alcanza el 94 %. dos aspectos que verifican la velocidad y precisión del recorte del mango son encontrar el rango óptimo de capas ocultas, mientras que, para entrenar la tasa de aprendizaje, el valor más bajo puede desarrollar el método de aprendizaje más lento, pero la precisión puede aumentar; de lo contrario, cuanto mayor



sea la tasa de aprendizaje, más rápido podrá construir el método de entrenamiento; sin embargo, resultará un poco de precisión (Yossy et al., 2017).

Según la investigación “Clasificación de manzanas utilizando visión artificial y redes neuronales artificiales”, realizado por Canek Mota Delfin, Carlos Juárez González y Juan Carlos Olguin Rojas, en la universidad autónoma de Chapingo en la ciudad de México, en el año fiscal 2018. Los resultados obtenidos indican que este método de clasificación de manzanas es efectivo, pero se puede mejorar aún más mediante la ampliación de nuestro dataset de entrenamiento en términos de resolución de imagen e iluminación, ya que estos factores tienen un gran impacto en la exactitud de la red. Es importante enfatizar que las pruebas del modelo se realizaron en un entorno ruidoso en términos de fondo de imagen y luz, por lo que la aplicación se realiza en un entorno controlado para aumentar la precisión. Las especificaciones del proyecto proporcionan pautas para el procesamiento posterior, no solo para la clasificación de manzanas, sino también para considerar otros aspectos como enfermedades de la manzana, úlceras, etc. (Mota-Delfin et al., 2018).

Según la investigación “Identificación de variedades de azufaifo indio cultivadas en Arabia Saudita utilizando una red neuronal artificial”, realizado por Adel M. Al-Saif, Mahmoud Abdel-Sattar, Abdulwahed M. Aboukarima, Dalia H. Eshra, en el año 2021. La identificación de la calidad de los cultivares de azufaifo de la India tiene un papel importante que desempeñar en la prevención de la adulteración comercial de este valioso producto. En esta investigación, se ha presentado un se ha desarrollado un nuevo enfoque basado en la técnica de redes neuronales artificiales para identificar once cultivares de azufaifo indio, que se cultivaron en Arabia Saudita. El clasificador de red neuronal artificial desarrollado fue entrenado para identificar cultivares de azufaifo indio utilizando características morfológicas de la fruta como el diámetro medio aritmético, el porcentaje



de esfericidad y los descriptores de área superficial y color. Los resultados indican que la tasa general de identificación correcta fue del 97,56 % durante la fase de prueba. El método propuesto se puede extender a la identificación de cultivares de azufaifo indio en tiempo real mediante la captura de las características de las frutas mediante imágenes digitales (Al-Saif et al., 2021).

2.2. LIMÓN

La producción de limones es constante, ya que se puede cultivar y recolectar a lo largo de todo el año. La cosecha suele llevarse a cabo luego de 4 años desde la siembra. El ciclo económico del cultivo tiene una duración aproximada de 15 años. (MINAGRI, 2017)

2.3. PRINCIPALES PRODUCTORES DE LIMÓN EN EL PERÚ

La principal región productora de limón en los últimos años en términos de participación, la región de Piura tuvo un porcentaje del 54,8%. Las regiones que le siguieron fueron Lambayeque (19,1%), Tumbes (11%), Loreto (4%) y Ucayali (3,3%). (MINAGRI, 2017)

2.4. PROCESO DE PRODUCCIÓN PARA LA DISTRIBUCIÓN NACIONAL Y EXTRANJERA

2.4.1. Etapa agrícola

Esta primera etapa de esta actividad incluye el manejo agrotécnico (principalmente el cultivo, la plantación y la obtención de frutos) de los limones. Durante esta etapa, el objetivo es asegurar que al menos el 60% del producto de la cosecha cumpla con los aspectos aceptables en el mercado local y global. La elaboración en este ciclo cuenta con el apoyo de los sectores de aprovisionamiento (instrumentos, nutrientes,



plaguicidas, bactericidas, etc.) y capital humano, así como apoyo técnico especializado, servicios (agua y electricidad) e inspección por parte de la autoridad competente (SENASA, 2020).

2.4.2. Etapa agroindustrial

La siguiente fase es el procesamiento o la operación agroalimentaria de los limones, que consiste en empaquetar los limones para su venta en el extranjero y en los departamentos del Perú. Los frutos son lavados, seleccionados, empacados, y enfriados de acuerdo a sus diferentes cualidades. Esta cuenta con el apoyo de los departamentos de obtención de productos (insumos de empaque, parafina y recambios), capital humano y garantía de calidad, como también la supervisión de la autoridad competente. (SENASA, 2020).

2.4.3. Operación logística

En la tercera etapa se gestiona la movilidad y carga y/o envíos por aire y tierra hasta el destino y se lleva a cabo la realización en cadena de frío hasta donde el exportador tenga responsabilidad: puerto de origen, puerto de destino, puerto aéreo, etc. El apoyo para esta fase es proporcionado por el departamento de control de calidad y los operadores logísticos (compañías de transporte, aduanas, agentes de despacho, etc.) (SENASA, 2020).

2.5. DISPOSICIONES CON RELACIÓN A LA CALIDAD DEL LIMÓN

En todas las categorías de limones, se requiere cumplir con las disposiciones específicas asignadas a cada categoría y se admiten ciertos márgenes de tolerancia.

A fin de que los limones satisfagan los requisitos de calidad establecidos, es fundamental que estén completos, saludables y sin daños ni descomposición que puedan



impedir su consumo. Además, es necesario que estén limpios y casi completamente libres de cualquier material extraño visible, así como de plagas que puedan afectar su apariencia general o causar daños en su superficie. La humedad externa anormal debe ser evitada, excepto por la condensación que se produce al sacarlos de una cámara frigorífica. Adicionalmente, los limones deben carecer de cualquier aroma o sabor atípico, tener una textura firme y estar libres de daños causados por temperaturas bajas o golpes.

Los limones deben ser cosechados con precaución y haber logrado un nivel adecuado de crecimiento estado de maduración, según los criterios específicos de la variedad y la región de producción son los que determinan la calidad de los productos. El grado de desarrollo y la condición de los limones deben ser suficientes para permitirles:

- Resistir el traslado y la manipulación.
- Llegar al destino en buenas condiciones.

Se espera que la mayoría del fruto tenga el color característico de su variedad en al menos dos tercios de su superficie. Aunque el fruto debe ser principalmente verde, puede tener manchas amarillas que cubran hasta el 30% de su superficie. (Del Castillo Huaccha, 2018).

2.6. CONCEPTOS COMPLEMENTARIOS

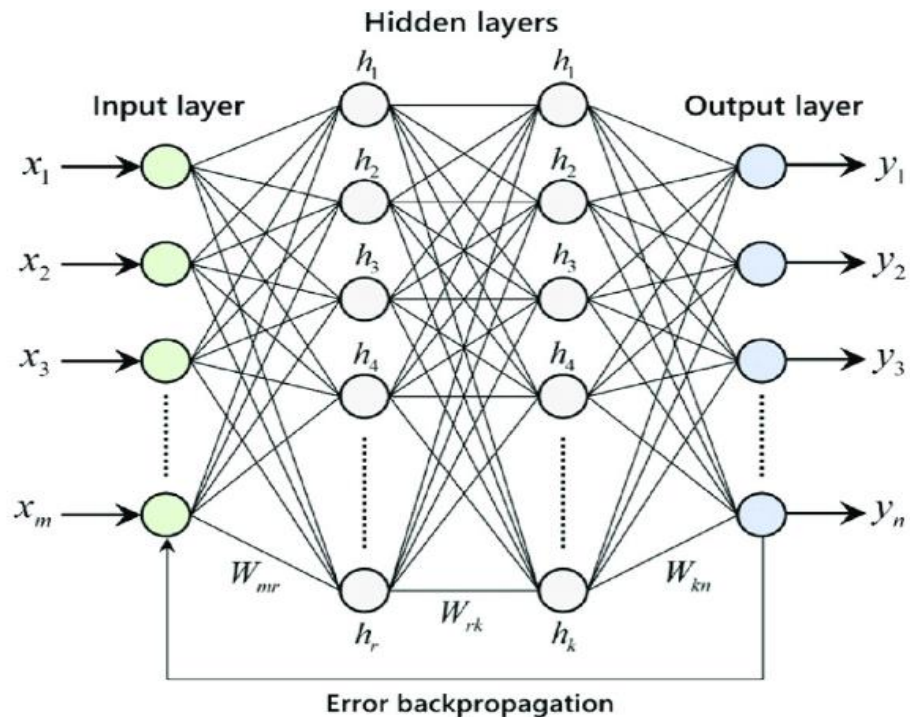
Se presenta algunas definiciones necesarias para comprender el funcionamiento de las redes neuronales convolucionales.

2.6.1. Redes neuronales artificiales

Una Red Neuronal Artificial (RNA) es un modelo que usa la matemática para procesar datos de manera similar al cerebro humano, y se compone de un gran número de elementos entrelazados llamados células nerviosas o neuronas, que colaboran en la

resolución de un problema. Las neuronas artificiales son nodos, donde las flechas direccionales y los pesos muestran la correspondencia entre las salidas y las entradas de las neuronas. Según la arquitectura, las ANN se dividen en dos grupos: feed-forward y recurrentes. (S. Tian et al., 2021)

Figura 1: Redes Neuronales Artificiales



Fuente: (S. Tian et al., 2021).

En la figura 1 el gráfico representa una red neuronal convolucional con capas de entrada, capas ocultas y capas de salida.

Cuando se utiliza más de dos capas ocultas, una red neuronal se convierte en profunda. Estas redes tienen la habilidad de solucionar problemas, lo que llevó al desarrollo del aprendizaje profundo, que también recibe el nombre de "Deep Learning" en inglés. El entrenamiento de redes neuronales con algoritmos de aprendizaje y capas profundas presenta nuevos desafíos en la tarea de procesar imágenes. Sin embargo, esto ha llevado a la exploración de redes neuronales convolucionales, las cuales se destacan en este tipo de tareas.



2.6.2. Tensor

Las entradas, salidas y transformaciones dentro de las redes neuronales se representan mediante tensores y, como resultado, la programación de redes neuronales utiliza mucho los tensores. Los tensores son la estructura de datos utilizada por los sistemas de aprendizaje automático, y conocerlos es una habilidad esencial que debe desarrollarse desde el principio.

Un tensor es un contenedor de datos numéricos. Es la forma en que almacenamos la información que usaremos dentro de nuestro sistema. Tres atributos principales definen un tensor:

- Rango
- Forma
- Tipo de datos

Los tensores se hicieron muy conocidos en IA con tensorflow precisamente porque se utiliza en el campo del aprendizaje automático de Google. Se utilizan tensores como unidad básica de cálculo. Aunque el término es el mismo, no son del todo iguales. Los tensores en programación no son lo mismo que los tensores en matemáticas. Simplemente heredan algunas de sus cualidades. De ellos toman algunas técnicas de representación. Aquí simplemente se representan como matrices de matrices o listas de listas. En el aprendizaje automático, hay muchas formas en que se puede manipular esta representación y no tienen que obedecer las estrictas leyes de transformación de coordinación establecidas por las matemáticas y la física. Por lo tanto, no puede considerarse completamente equivalente a los tensores matemáticos, pero puede considerarse que hereda algunas de esas propiedades matemáticas.



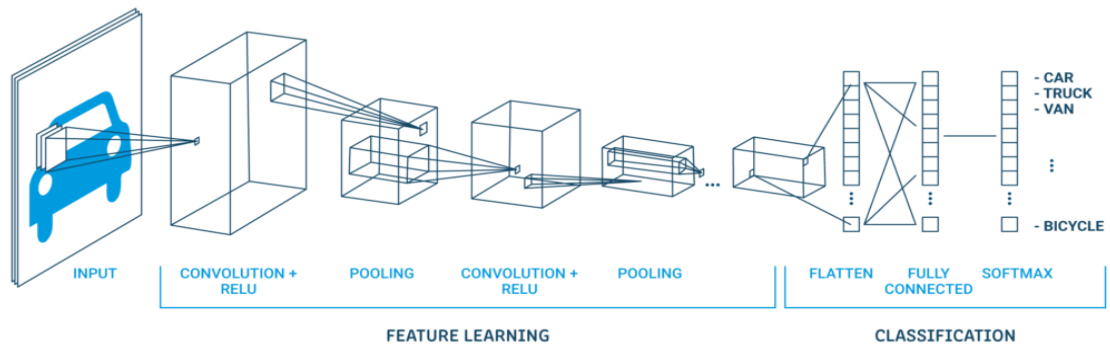
Tensorflow tiene un atributo llamado forma para representar cada tensor. Cada tensor tiene una forma (x, y) donde x es la longitud del tensor y “ y ” es la dimensión de las matrices o lista/arreglo en este caso dentro del tensor. Los datos de píxeles de las imágenes se pueden representar fácilmente en una matriz. Lo mismo puede decirse de los fotogramas de un vídeo. La representación se vuelve mucho más fácil. Entonces, lo importante es que podemos obtener una representación tan precisa de los datos que se puede considerar casi muy cercana a la representación natural de esos objetos (Silvestre, 2020).

2.6.3. Red neuronal convolucional

Una CNN es un tipo de red neuronal profunda. La Inteligencia Artificial (IA) y el desarrollo de Convolutional Neural Network (CNN) creó la capacidad de identificar imágenes y detectar objetos en una imagen. (Woan Ching et al., 2022)

Frecuentemente se emplean las redes neuronales en la tarea de clasificación, en la cual la salida es un vector que asigna una probabilidad a cada clase. Particularmente, las redes neuronales convolucionales son utilizadas para la categorización de patrones visuales. Como evolución de este tipo de aplicaciones, han surgido otras como las de etiquetado semántico o las de detección de elementos visuales, las cuales no solo identifican los objetos presentes en la imagen, sino que también proporcionan las coordenadas espaciales donde se encuentran dichos objetos.

Figura 2: Etapas de una red neuronal convolucional



Fuente: (Charu C., 2018)

La figura 2 representa el proceso por el cual debe pasar una imagen para su predicción.

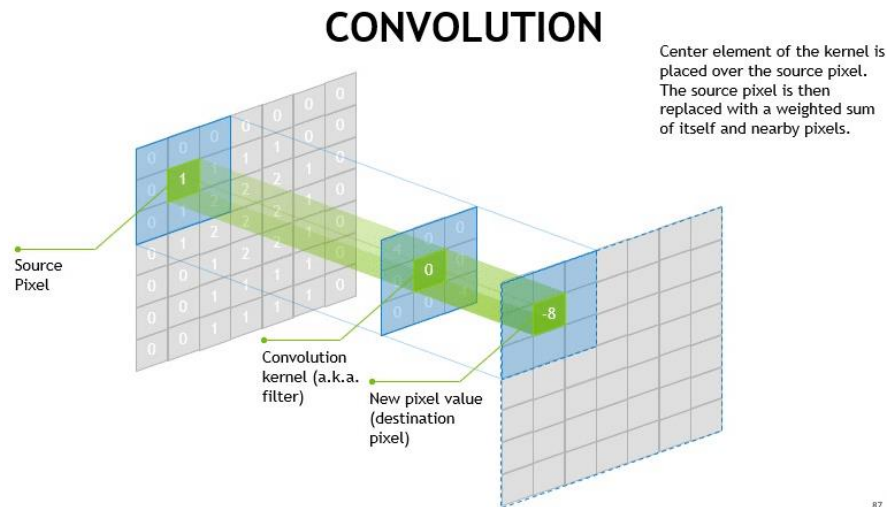
2.6.4. Capas de una red neuronal convolucional

a) Capa de convolución

La operación de convolución es uno de los procedimientos característicos de estas redes de aprendizaje profundo. Consiste en la selección de un conjunto de unidades de imagen de entrada, sobre los cuales se aplica un producto escalar con un núcleo determinado. El núcleo se desplazará a lo largo de cada una de las neuronas de entrada y generará un nuevo resultado, que se convertirá en una de las capas ocultas.

Durante este procedimiento, se utiliza una imagen y un núcleo, con el objetivo de que el núcleo se aplique a toda la imagen. En resumen, el tamaño del núcleo generalmente es más pequeño con respecto a la imagen. La convolución es un proceso donde se multiplica el núcleo con una porción de la imagen, se realiza la multiplicación tal como indica la imagen y después el núcleo hace un traslado y es por ello que se considera un proceso iterativo (Charu C., 2018).

Figura 3: Operación de convolución



Fuente: (Charu C., 2018).

El gráfico de la figura 3 representa la operación de convolución, el cual es usado para el proceso de identificación de atributos visuales en una imagen.

b) Capas de pooling

La agrupación en redes neuronales convolucionales es una técnica para generalizar características extraídas por filtros convolucionales y ayudar a la red a reconocer características independientemente de su ubicación en la imagen.

Las capas convolucionales son los componentes básicos de una red de convolución utilizada para aplicaciones con percepción artificial, como la identificación de imágenes. Una capa de convolución desliza un filtro sobre la imagen y extrae características, lo que da como resultado un grupo de características que se puede usar para alimentar a la siguiente capa de convolución para extraer características de nivel superior (Saleh, 2020).

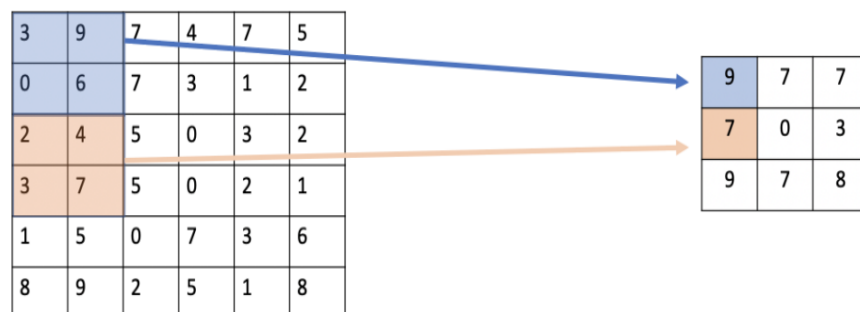
Por lo tanto, el apilamiento de múltiples capas convolucionales permite que las CNN reconozcan estructuras y objetos cada vez más complejos en una imagen.

El procedimiento básico de la agrupación es muy similar a la operación de convolución, se selecciona un filtro y lo desliza sobre el mapa de elementos del resultado de la capa convolucional pasada. El tamaño del filtro más utilizado es 2×2 y se desliza sobre la entrada con un paso de 2. Según el tipo de operación de agrupación que haya seleccionado, el filtro de agrupación calcula una salida en el campo receptivo (la parte del mapa de características debajo del filtro).

Hay varios enfoques para la puesta en común. Los enfoques más utilizados son la máxima agrupación (Max pooling) y la agrupación por media aritmética (Average Pooling).

- **Max Pooling:** En el max Pooling, el filtro con facilidad selecciona la magnitud de unidad de imagen máximo en el campo receptivo. Por ejemplo, si tiene 4 píxeles en el campo con valores 3, 9, 0 y 6, seleccione 9.

Figura 4: Max Pooling

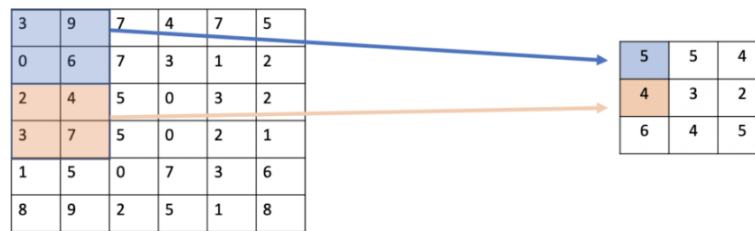


Fuente: (Saleh, 2020)

El grafico de la figura 4, se puede visualizar la operación de máxima agrupación (Max Pooling).

- **Average Pooling:** La agrupación por media aritmética funciona calculando el valor promedio de los valores de unidades de imagen en el campo visual. Dados 4 píxeles con los valores 3, 9, 0 y 6, la capa de agrupación promedio produciría una salida de 4,5. Redondeando a números completos nos da 5.

Figura 5: Average Pooling



Fuente: (Saleh, 2020)

El gráfico de la figura 5 representa la operación de agrupación promedio.

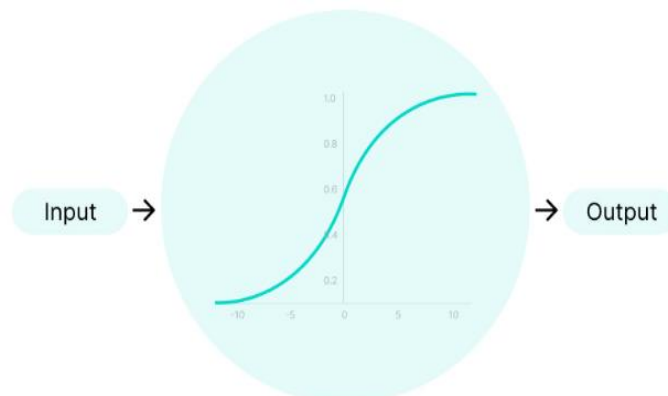
c) Capa completamente conectada

Capa completamente conectada o también conocida como fully connected layer, es la capa final de una red neuronal convolucional. Las capas de salida se construyen transformando los tensores más profundos de la red en vectores y tiene la capacidad de clasificar las imágenes (Saleh, 2020).

2.6.5. Función de activación

El objetivo de una función de activación es dotar a la red neuronal de características no lineales.

Figura 6: Función de activación



Fuente: (Yoav Goldberg, 2017)

La figura 6 representa una función de activación el cual se usa para que el modelo no realice transformaciones lineales.

Las funciones de activación introducen un paso adicional en cada capa durante la propagación directa, pero su cálculo vale la pena.

Si se supone que tenemos una red neuronal funcionando sin las funciones de activación. En ese caso, cada neurona solo realizará una transformación lineal en las entradas utilizando los pesos y sesgos. Es porque no importa cuántas capas ocultas adjuntemos en la red neuronal; las capas serán homogéneas en su comportamiento ya que la combinación de dos funciones lineales resulta en una función lineal

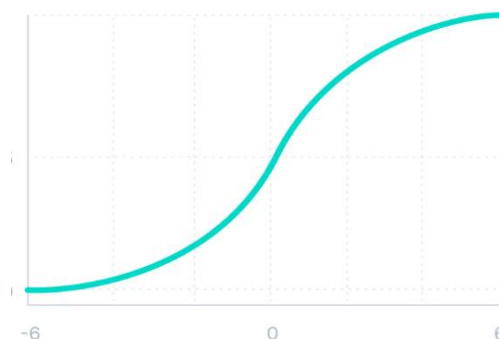
- **Función de activación Sigmoide**

La función es capaz de aceptar cualquier valor numérico y producir como salida valores que se encuentran dentro del rango de 0 a 1.

A medida que la entrada aumenta, el valor de salida se acerca cada vez más a 1, mientras que si la entrada es más pequeña (negativa), la salida se aproxima a 0, como se muestra a continuación.

Figura 7: Función sigmoide

Sigmoid / Logistic



Fuente: (Yoav Goldberg, 2017)

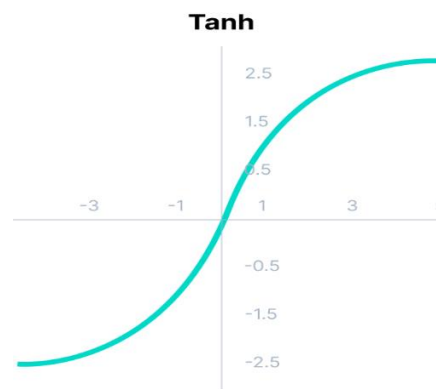
Matemáticamente puede representarse como:

$$f(X) = \frac{1}{1 + e^{-X}}$$

- **Función tangente hiperbólica:**

La función Tanh tiene cierto parecido a la función de activación exponencial logística, e incluso tiene la misma apariencia de S con una diferencia en el rango de salida de -1 a 1. En Tanh, cuanto mayor sea la entrada, más próximo será el valor de salida de 1, mientras que cuanto menor sea la entrada, más cerca estará la salida de -1.

Figura 8: Función tangente Hiperbólica



Fuente: (Yoav Goldberg, 2017)

Matemáticamente esta función se puede representar como

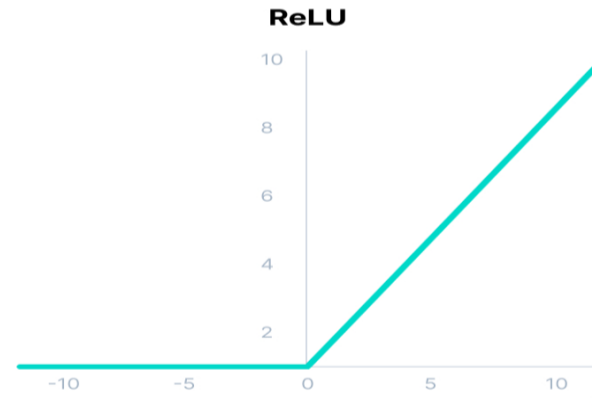
$$F(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- **Función de activación RELU:**

ReLU (Rectified Linear Unit) que en inglés significa Unidad lineal rectificada. Aunque da la impresión de una función lineal, ReLU tiene una función derivada y permite la retro propagación al mismo tiempo que lo hace computacionalmente eficiente. El problema que se presenta con la función ReLU es que no todas las

neuronas se activan al mismo tiempo. Las neuronas se desactivarán solamente si la salida de la operación matemática que realizan es un número menor que 0.

Figura 9: Función de activación RELU



Fuente: (Yoav Goldberg, 2017)

Las funciones se aplican a la salida de cada capa convolucional y de las capas completamente conectadas, excepto en la última capa. En la última capa, la función de activación más usada es la "softmax", que se utiliza en problemas de categorización.

- **Función de activación softmax**

Softmax se usa para la transformación una lista de elementos o vector de N valores numéricos reales en otro de N valores numéricos racionales que suman 1. Aunque las entradas puedan ser positivas, negativas, mayor que cero y distinto de uno, la función softmax las convierte en datos dentro del rango de 0 a 1, permitiendo interpretarlos como probabilidades. Cuando una entrada es de menor valor o con tendencia hacia los negativos, softmax la convierte en una eventualidad baja, mientras que, si la entrada es de mayor valor, se convierte en una eventualidad alta, aunque permanentemente manteniéndose en el rango de 0 a 1.

La función softmax es también conocida como softargmax o regresión logística multiclase. Esta denominación se debe a que softmax es una ampliación de la

regresión logística que se aplica para la clasificación de múltiples clases, y su ecuación es muy similar a la función anterior, llamada sigmoide, que se utiliza en la regresión binomial. Es importante tener en cuenta que la función softmax solo es apropiada para ser utilizada en un clasificador cuando las clases son mutuamente excluyentes.

En algunas redes neuronales multicapa, la capa previa a la última genera puntajes numéricos reales que no se escalan adecuadamente, lo que puede complicar su uso posterior. Para solucionar este problema, la función softmax resulta muy conveniente, ya que transforma estos puntajes en una distribución de probabilidad normalizada que se puede mostrar a los usuarios o usar como entrada para otros sistemas. Por esta razón, es común agregar una capa final con función softmax en la red neuronal. (Yoav Goldberg, 2017).

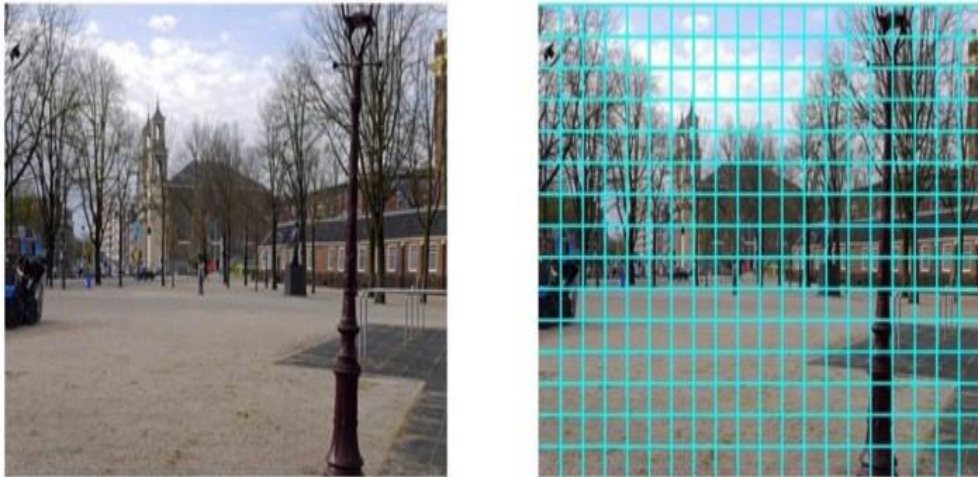
Fórmula matemática softmax:

$$\text{Softmax } \sigma(Z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

2.6.6. Pre - procesamiento de una imagen

Todas las imágenes pueden ser descritas mediante una matriz de píxeles. Básicamente, una imagen es una matriz donde cada celda contiene información del píxel correspondiente. La imagen se divide en una cuadrícula de píxeles y cada uno de ellos almacena información de la imagen, como el color o brillo. En las redes neuronales, se utilizan estas matrices para identificar bordes, colores y formas en las imágenes, lo que permite el reconocimiento de imágenes. Estas redes pueden llevar a cabo tareas como la categorización de imágenes o la identificación de objetos dentro de las mismas. (Deepika Ghai et al., 2022).

Figura 10: Imagen dividida en cuadrículas



Fuente: (Deepika Ghai et al., 2022)

En esta etapa existen algunas técnicas para concretar el preprocesamiento:

- **Relación de aspecto**

La razón o relación de aspecto de su imagen es la razón entre el ancho de su imagen y el nivel de altura de su imagen, debe escalar todas sus imágenes para que tengan una relación de aspecto uniforme. La mayor cantidad de las representaciones asumen que la conformación de su figura de entrada es cuadrada, es por ello que debe recortar sus imágenes para que todas tengan forma cuadrada, es posible que se desee realizar un recorte central para extraer la parte más importante de su imagen. Recortar una imagen para obtener una forma cuadrada de las características más importantes hace que la correspondencia de aspecto de toda la cantidad completa de las imágenes con las que está trabajando sea constante.

- **Tamaño de Imagen uniforme**

La información de la imagen se alimenta en lotes para entrenar su modelo. Un conjunto de imágenes requiere que todas las figuras de sus datos tengan igual

dimensionamiento. Debe asegurarse de que todas las imágenes alimentadas en su modelo tengan la misma altura y anchura. Las redes neuronales convolucionales usan algo llamado mapas de características para obtener la información de la figura de entrada.

- **Imágenes medias y perturbadas**

Una forma de acelerar el entrenamiento de las redes neuronales es trabajar con imágenes de entrada normalizadas. La normalización implica centrar los valores de los píxeles alrededor de una media de cero. Para ello, se resta la media aritmética de la figura de todos los datos de entrenamiento o de un lote y se divide por la desviación normalizada. De esta manera, se certifica que cada unidad de imagen o también llamada pixel tenga una distribución de datos semejante. Una vez normalizada la imagen, se puede ajustar la escala de los valores de la unidad de imagen para que estén en el intervalo de 0 a 1.

- **Reducción de dimensionalidad**

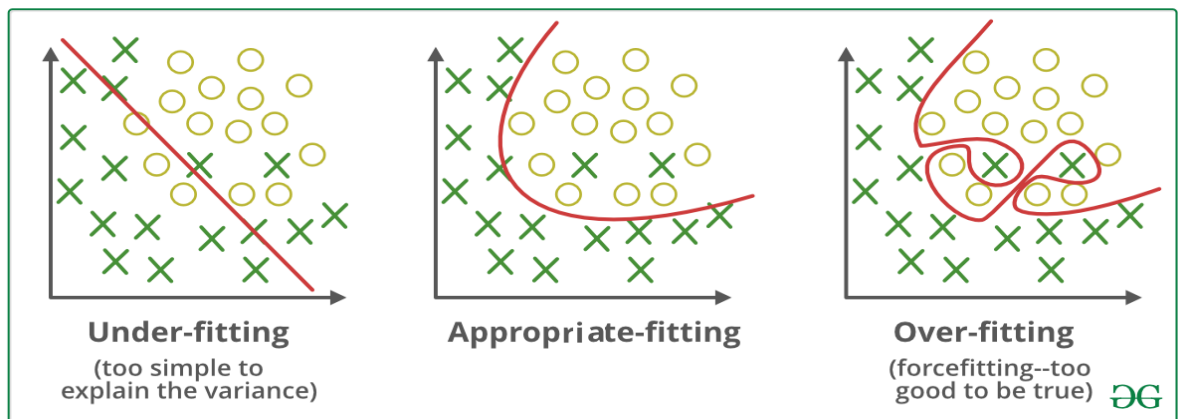
Las imágenes con las que se trabajan pueden ser muy grandes, lo que significa que se trata de características de gran dimensionalidad. Es posible que desee realizar una reducción de la dimensionalidad antes de usar imágenes para entrenar su modelo. Reducir imágenes RGB a escala de grises es una especie de reducción de dimensionalidad. La reducción de la dimensionalidad ayuda a extraer la latente de las características significativas de sus datos subyacentes para que su red neuronal no tenga que lidiar con características irrelevantes.

2.6.7. Entrenamiento de la red neuronal

La formación de una red neuronal inicia cuando se tiene listo las capas que incluyen las imágenes de los limones, posterior a ello la red neuronal se alimenta de

dataset de imágenes que se etiquetan como limones en buen estado y limones en mal estado. Existen ciertos problemas que pueden surgir en el proceso del entrenamiento, es más conocido como sobreajuste u overfitting, donde la red neuronal memoriza todos los datos y será incapaz de reconocer otros datos, lo cual no se desea; el otro problema es el underfitting que es un problema de poco ajuste (Sebastian Raschka et al., 2022).

Figura 11: Imagen dividida en cuadrículas



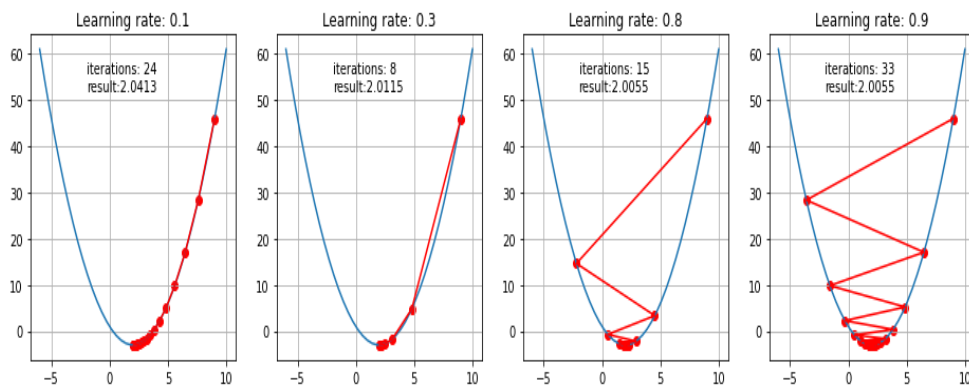
Fuente: (Sebastian Raschka et al., 2022)

La figura 11 representa el ajuste apropiado que debe tener un modelo y así evitar el sobre ajuste o el bajo ajuste.

a) Gradiente descendiente

La gradiente descendiente es una técnica para evitar el sobreajuste, quiere decir que es fundamental en Deep Learning para entrenar las redes neuronales. La gradiente descendiente es un algoritmo que emplea técnicas de optimización de primer orden y se ejecuta en forma iterativa, con el fin de encontrar un mínimo o un máximo local en una función determinada (Sebastian Raschka et al., 2022) .

Figura 12: Taza de aprendizaje



Fuente: (Sebastian Raschka et al., 2022)

b) Tamaño de lote

El tamaño del lote define la cantidad de muestras que usamos en una época para el entrenamiento una red de neuronas. Hay tres tipos de descenso de gradiente con respecto al tamaño del lote:

- Descenso de gradiente por lotes: utiliza todas las imágenes del conjunto de entrenamiento en una época.
- Descenso de gradiente estocástico: utiliza solo una muestra aleatoria del conjunto de entrenamiento en una época.
- Descenso de gradiente de mini lotes: utiliza un número determinado de muestras del conjunto de capacitación en una época.

El tamaño del lote afecta algunos indicadores, como el tiempo de entrenamiento general, el tiempo de entrenamiento por época, la calidad del modelo y similares. Por lo general, elegimos el tamaño del lote como una potencia de dos, en el rango entre 16 y 512. Pero, en general, el tamaño de 32 es una regla general y una buena elección inicial (Saleh, 2020).



2.7. PYTHON

Python fue propuesto por Guido van Rossum en 1989. Es un lenguaje informático de alto nivel interpretado y enfocado a objetos. Después de su lanzamiento público, ha sido popular entre los entusiastas del programa. Python se caracteriza por ser conciso y sintaxis clara, es fácil de leer, tiene buena escalabilidad y es un software de código abierto. Los programadores no necesitan tener un base de programación profunda. El lenguaje informático sencillo de entender y es muy adecuado para aprender programación primaria (Jiang et al., 2022).

En la actualidad, muchas instituciones nacionales y extranjeras, incluidas las escuelas, han desarrollado gradualmente capacitación y cursos relacionados con la programación en Python. Python puede implementar API enriquecidas y varias herramientas informáticas científicas, de modo que incluso los programadores que están acostumbrados a usar el lenguaje C, C y JAVA pueden comenzar y operar fácilmente e integrar sus paquetes escritos en otros lenguajes en la plataforma python. Además, el propio compilador de Python también se puede integrar en programas en otros lenguajes, de ahí el nombre de "lenguaje adhesivo".

Cada vez más investigadores utilizan Python y sus bibliotecas de extensión para procesar datos experimentales, dibujar hermosos gráficos e incluso desarrollar algunas aplicaciones. Con el fin de optimizar la eficiencia de la clasificación visual de los grandes datos de texto económico, este estudio aplica el software Python, que procesa la matriz de datos y calcula el valor numérico muy rápido.

Con Python se puede lograr lo siguiente:

- Aplicaciones web
- Grandes aplicaciones de datos



- Pruebas
- Automatización
- Ciencia de datos, aprendizaje automático e IA
- Software de escritorio
- Aplicaciones móviles

Python es multiplataforma. Los programas Python pueden ejecutarse en Windows, Linux y macOS. Se tiene una gran comunidad en la familia de Python, siempre que se necesite ayuda, tienes una comunidad activa.

2.8. LIBRERÍAS

2.8.1. Tensorflow

TensorFlow es una biblioteca de aprendizaje autónomo de software libre. Específicamente, es software, y los usuarios lo necesitan para construir modelos matemáticos programando en Python y otros lenguajes. Estos modelos se utilizan en la aplicación de inteligencia artificial. (Liu et al., 2020)

Con TensorFlow, los programadores tienen la capacidad de construir diagramas de flujo de datos que detallan el movimiento del conjunto de datos a través de una red de puntos de procesamiento. Cada punto en la red representa un cálculo matemático y los bordes entre los puntos conectan matrices de datos multidimensionales, también conocidas como tensores.

Se puede ejecutar las aplicaciones de TensorFlow en diversas plataformas, de manera local en una PC, en la nube, en aparatos móviles iOS y Android, CPU o GPU. Para una mayor aceleración, si se utiliza los servicios de la drive o nube de Google, se



puede hacer funcionar TensorFlow desde la Unidad de procesamiento de TensorFlow (TPU) de Google. Aunque los modelos creados con TensorFlow son altamente portables y se pueden implementar en una amplia variedad de dispositivos, para generar predicciones.

En octubre de 2019, se lanzó TensorFlow 2.0, que fue rediseñado considerando los comentarios de las personas para hacerlo más sencillo y eficiente de hacer código con él. Se adicionó API de Keras más sencillo para el entrenamiento de redes neuronales y una nueva API que facilita el entrenamiento distribuido. Además, la integración con TensorFlow para dispositivos móviles permite hacer modelos en más sistemas. Además, el código escrito para versiones pasadas de TensorFlow tiene que cambiarse para aprovechar de manera óptima los nuevos detalles de TensorFlow 2.0, a veces solo ligeramente y a veces significativamente.

TensorFlow ofrece al programador todas estas características a través del lenguaje informático Python. Python es un software de programación fácil de aprender y trabajar, y proporciona formas convenientes de expresar cómo se pueden acoplar las abstracciones de alto nivel. TensorFlow es compatible con versiones de Python desde la 3.7 hasta la 3.10, aunque puede funcionar en versiones anteriores de Python, no hay garantía de que lo haga.

En TensorFlow, los nodos y tensores se definen como objetos de Python y las aplicaciones de TensorFlow se escriben en Python. Sin embargo, las operaciones matemáticas no se ejecutan directamente en Python, sino que están implementadas como bibliotecas de transformación de alto rendimiento en C++. Python actúa como un intermediario entre las partes y ofrece abstracciones de programación de nivel superior para conectarlas.



El desarrollo de alto nivel en TensorFlow, que consiste en crear nodos y capas y conectarlos, se realiza a través de la biblioteca de Keras. La API de Keras parece ser sencilla; un modelo básico de tres capas puede definirse en menos de 10 líneas de código, y para entrenar dicho modelo solo se necesitan unas pocas líneas de código adicionales.

2.8.2. Keras

Keras es la opción preferida entre los profesionales que trabajan con bibliotecas de aprendizaje profundo. Esta plataforma es una interfaz de software construida encima del motor de trabajo TensorFlow, lo que permite a los usuarios desarrollar, entrenar y validar redes neuronales de manera eficiente. Keras encapsula la mayor parte de la complicación de bajo nivel que se maneja para implementar redes profundas desde lo básico, lo que facilita su uso. Esta herramienta también abstracta muchos de los aspectos complicados de TensorFlow mientras brinda opciones de personalización y usabilidad. Como resultado de su popularidad y facilidad de uso, Keras es la elección principal para aplicaciones de aprendizaje profundo y es la plataforma preferida para construir uno de los extremos de un puente de dos vías.(Ott et al., 2020)

2.8.3. Numpy

NumPy es un paquete de Python para ofrecer computación científica soporte para matrices multidimensionales y una biblioteca de algoritmos numéricos.(Arabas et al., 2014)

La biblioteca NumPy proporciona un conjunto de funciones que permiten realizar operaciones matemáticas en matrices y arreglos de datos, lo cual incluye diversas operaciones como aritméticas, estadísticas, trigonométricas y de álgebra lineal, entre otras. Gracias a su habilidad para manejar y calcular grandes conjuntos de datos de



manera eficiente, NumPy es ampliamente utilizada la exploración de datos y el aprendizaje automático (machine learning).

Además, NumPy se utiliza en combinación con otras bibliotecas de Python como Pandas, Matplotlib, Scikit-learn y TensorFlow, entre otras. Debido a su importancia en el modelado de datos y en el procesamiento de datos en general, esta herramienta es considerada esencial en el kit de herramientas de cualquier persona que trabaja en estas áreas.

2.8.4. Python imaging library (PIL)

PIL agrega soporte a Python para abrir, guardar y cambiar diferentes formatos de archivo de imagen.(Jaskolka et al., 2019)

La Biblioteca de Procesamiento de Imágenes de Python (conocida como PIL, por su nomenclatura en inglés) es tomado en cuenta como el paquete de procesamiento de imágenes por defecto para el lenguaje de programación Python. Incorpora herramientas ligeras de procesamiento de imágenes que ayudan a editar, crear y guardar imágenes. El soporte para Python Imaging Library se suspendió en 2011, pero un proyecto llamado pillow bifurcó el proyecto PIL original y le agregó soporte para Python3.x. Pillow se anunció como reemplazo de PIL para uso futuro. Pillow admite una gran cantidad de extensiones de archivo de imagen, incluidos BMP, PNG, JPEG y TIFF. La biblioteca fomenta la adición de soporte para formatos más nuevos en la biblioteca mediante la creación de nuevos decodificadores de archivos.

2.8.5. Path lib

Python incluye la biblioteca llamada pathlib que es usado para manejar rutas de sistemas de archivos de forma escéptica en cualquier entorno operativo.

Este fragmento es particularmente útil para crear y manipular rutas de archivos y directorios, así como para acceder a archivos y directorios de manera eficiente. Además, pathlib permite trabajar con rutas de manera independiente del entorno operativo que se esté usando, lo que lo hace altamente portátil. Si bien es posible usar otras herramientas para realizar este tipo de operaciones, el módulo pathlib le permite realizar estas operaciones con un alto nivel de claridad y con una cantidad de código conciso (Sebastian Raschka et al., 2022).

Figura 13: Manejo de rutas de un sistema operativo



Fuente: (Sebastian Raschka et al., 2022)

2.8.6. Mathplotlib

Matplotlib es una biblioteca muy usada en el entorno de programación Python. Es útil a la hora de visualizar los resultados con gráficos (Z. Tian, 2021). Matplotlib es una herramienta que nos facilita la creación de gráficos y, además, hace posible la realización



de gráficos más complejos. Con Matplotlib, es fácil crear y personalizar diferentes tipos de gráficos, incluyendo:

- Representaciones gráficas en forma de barras.
- Gráficos de frecuencia en forma de rectángulos adyacentes.
- Gráficos circulares que muestran la proporción de cada categoría.
- Gráficos que muestran la forma en que los datos están dispuestos a través de una caja y líneas que se extienden hacia afuera.
- Representaciones gráficas que muestran la forma en que están dispuestos los datos mediante una curva.
- Representaciones gráficas que muestran la relación entre dos variables mediante una gráfica de dispersión, en la que los datos se visualizan en un plano.
- Representaciones gráficas que muestran la tendencia de una variable a lo largo del tiempo mediante una línea.
- Gráficos que muestran la magnitud y el cambio en el tiempo de una o varias variables mediante una serie de áreas sombreadas.
- Representaciones gráficas que muestran los niveles de una variable a través de un vector que une los puntos de igual valor.
- Mapas que muestran la variación espacial de una variable mediante una escala de colores.

2.8.7. Os

El módulo OS en Python es parte de la biblioteca muy utilizada del entorno de programación. Cuando se importa, ayuda al programador interactuar con el entorno



operativo nativo en el que Python se está ejecutando actualmente. En términos simples, proporciona una manera fácil para que el usuario interactúe con varias funciones del sistema operativo que resultan útiles en la programación diaria. El módulo OS y los módulos `os.path` son los mismos y se pueden importar fácilmente desde la biblioteca estándar, en cualquier momento (Edureka, 2019).

Este módulo nos permite ejecutar funciones que se detallan a continuación:

- **os.name():** Permite conocer el nombre y las credenciales del sistema operativo actual en el que se ejecuta Python.
- **os.getcwd():** Permite conocer el directorio de trabajo actual o CWD que se ha utilizado para ejecutar su código, puede utilizar esta función. Similar a la función `os.name`, la salida de esto variará según el sistema en el que esté instalado.
- **os.error():** cada vez que utilice un módulo o una función en Python que se haya importado de la biblioteca estándar, generará un `OSError` en caso de que haya utilizado una ruta incorrecta, así como los nombres de archivo, o haya utilizado un argumento que tiene la correcta tipo pero no es aceptado por el sistema operativo que está utilizando actualmente. Esta función es un alias de la excepción `OSError` incorporada en Python.
- **os.popen():** esta función es parte de la manipulación de objetos de archivo y se usa para abrir una tubería hacia y desde un comando. El valor de retorno de esta función se puede leer o escribir dependiendo de su uso de `r` o `w`.
- **os.close():** si desea cerrar el directorio de archivos `fd`, puede utilizar esta función. Cuando se usa, un archivo debe abrirse primero con la función `open()` y luego cerrarse con la función `close()`.



- **os.rename():** si en una determinada situación necesita cambiar el nombre de un archivo de texto antiguo que ya está presente, puede hacer uso de esta función.

2.8.8. Csv

El formato CSV es muy común en la transferencia y el intercambio de datos entre diferentes aplicaciones. Debido a la prevalencia del formato, prácticamente todos sistemas de gestión de datos de potencia industrial y los marcos de procesamiento de secuencias admiten la importación de datos CSV. Sin embargo, cargar la entrada CSV cerca de la velocidad del hardware de E/S es un desafío. Los dispositivos de E/S modernos, como las NIC InfiniBand y las SSD NVMe, son capaces de soportar altas tasas de transferencia de 100 Gbit/s y superiores. Al mismo tiempo, el rendimiento del análisis de CSV está limitado por los complejos flujos de control en los que incurre su diseño semiestructurado y basado en texto (Lutz & Kumaigorodski, 2021).

2.8.9. Google colab

Colaboratory o también conocido como Colab es una herramienta desarrollada por investigación de Google que permite a cualquier usuario componer y ejecutar un código de Python en un navegador web sin necesidad de instalar software en su equipo. Esta herramienta es particularmente útil para proyectos de análisis predictivo, análisis de información o datos y en la educación, proporcionando una plataforma accesible y fácil de usar para estos fines. Desde una perspectiva técnica, Colab es un servicio para portátiles Jupyter que no requiere instalación y proporciona acceso gratuito a los recursos informáticos, incluidas las GPU. (Google, 2022).

La disponibilidad y cantidad de recursos en Colab no son asegurados y pueden variar en diferentes momentos. Esta fluctuación es esencial para ofrecer el servicio sin costo alguno. Los cuadernos de Colab pueden ser guardados en Google Drive y también



importados desde GitHub. Además, las carpetas de Colab se pueden compartir de manera similar a como se pueden compartir los documentos de Google o las hojas de cálculo.

El código que escribe se procesa en una máquina virtual que está dedicada exclusivamente a su cuenta. Si una máquina virtual ha estado inactiva por un período prolongado, se borra automáticamente. Además, el sistema de Colab establece un tiempo máximo de vida útil para las máquinas virtuales.

2.8.10. Google drive

Una herramienta proporcionada por Google que permite a los usuarios almacenar archivos en línea se llama Google Drive. Fue lanzado en 2012 y está disponible tanto en una versión gratuita como en varios planes de pago. Google Docs, fue reemplazada por Google Drive, esta herramienta cuenta con características equivalentes. Los usuarios de Google Drive pueden almacenar figuras, material audiovisual, dibujos, archivos de texto y otros tipos de archivos con diferentes extensiones. La versión gratuita del servicio permite almacenar hasta 15 gigabytes de datos sin costo alguno. Es importante tener en cuenta que Google Drive es un servicio de computación en la nube, lo que significa que los usuarios pueden iniciar sesión con su nombre de usuario y contraseña desde cualquier dispositivo (computadora, tableta o teléfono inteligente) para acceder a sus archivos. Como resultado, el acceso a los archivos no está restringido a un solo dispositivo.

Por otro lado, Google Drive simplifica el intercambio de archivos entre numerosas personas. Es posible que el propietario de la cuenta invite a otros usuarios a ver, descargar y/o editar documentos sin tener que proporcionar los archivos por correo electrónico u otro método.(Definicion.de, 2020).

2.9. RASPBERRY PI

El Raspberry Pi es un dispositivo informático pequeño y de precio razonable que tiene aproximadamente el tamaño de una tarjeta de crédito. Puede conectarse a una computadora o monitor de TV y operarse con un teclado y mouse estándar. Esta herramienta es capaz de realizar todas las funciones que se esperan de una computadora de escritorio y es útil para que personas de todas las generaciones puedan indagar el mundo del software y aprender a hacer código en lenguajes como Scratch y Python. Además, La Raspberry Pi es una placa computacional que tiene la habilidad de interactuar con el mundo real. Es utilizada en una variedad de proyectos de creación digital, como por ejemplo, en el desarrollo de puntos de medición de clima, entre otros (Simon Monk, 2021).

Figura 14: Raspberry Pi



Fuente: (Simon Monk, 2021)

Tabla 1: Características de Raspberry Pi

CARACTERÍSTICAS DE RASPBERRY PI 4 MODELO B	
PROCESADOR	Cortex-A72, four-core, Broadcom BCM2711B0
FRECUENCIA DEL CPU	1,5 GHz
PROCESADOR DE VIDEO	VideoCore VI 500 MHz



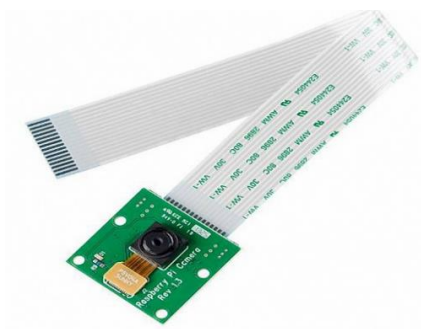
(Continuación...)	
MEMORIA RAM	4 GB LPDDR4-3200
CONEXIÓN NO FISICA	Wi-Fi 2,4 GHz - 5GHz IEEE - 802.11 Bluetooth 5.0
CONECTIVIDAD DE INTERNET	Gigabit Ethernet
PUERTOS	GPIO-40 pines 2 x Micro HDMI 2 x USB versión 2.0 2 x USB versión 3.0 CSI (cámara Raspberry Pi) Tomas auriculares y vídeo compuesto Sand Disk USB-C para alimentación Power-over-Ethernet (PoE)
FECHA DE LANZAMIENTO	24/06/2019

Fuente: (Simon Monk, 2021)

2.9.1. Cámara pi

La placa de la cámara de la Raspberry Pi se puede conectar de manera sencilla y directa al conector CSI de la placa Raspberry Pi. Este dispositivo es capaz de mostrar una imagen clara con una resolución de 5 MP o grabar videos de alta definición de 1080p a 30 fps, siendo la versión más reciente la 1.3. creada y desarrollada por la sociedad Raspberry Pi con sede en el Reino Unido, la placa de La cámara está equipada con un sensor Omnivision 5647 de 5MP (2592-1944 píxeles) en un módulo de enfoque predefinido. Este módulo se conecta a la Raspberry Pi mediante un cable de cinta de 15 pines que se conecta a una interfaz serial de cámara (CSI) MIPI de 15 pines especialmente diseñada para cámaras. La interconexión de datos CSI puede transmitir datos a velocidades muy altas y lleva solamente datos de unidades de imagen al procesador de raspberry. La placa en sí no es de gran tamaño, con unas dimensiones de alrededor de 25 milímetros x 20 milímetros x 9 milímetros y pesa un algo más de 3 g, lo que la hace ideal para terminales móviles y otras aplicaciones donde el tamaño y el peso son importantes. El sensor tiene una resolución nativa de 5 megapíxeles y una lente de enfoque fijo integrada. En cuanto a las imágenes fijas, la cámara puede capturar imágenes de 2592 x 1944 píxeles. La cámara es compatible con la versión más nueva de Raspbian, el sistema informático preferido de Raspberry Pi (Simon Monk, 2021).

Figura 15: Cámara Pi



Fuente: (Simon Monk, 2021)

Tabla 2: Características de la Cámara Pi

Características de la cámara pi de 5MP	
Compatibilidad	Tiene compatibilidad con el modelo A y el modelo B
Cámara	Omnivision 5647 de 5MP
Resolución de imagen estática	2592 x 1944
Video	Tiene soporte 1080p @ 30fps, 720p @ 60fps y 640x480p Grabación 60/90
Bus	Serial de cámara MIPI de 15 pines que se conecta directamente a la placa Raspberry Pi
Tamaño	20x25x9mm
Peso	3g

Fuente: (Simon Monk, 2021)

2.9.2. Sistema operativo raspbian

Raspberry Pi OS es un sistema informático de costo cero que utiliza como base Debian y ha sido desarrollado para explotar al máximo el hardware de la Raspberry Pi, siendo el sistema operativo más adecuado para un uso general en esta plataforma. El sistema operativo cuenta con una amplia variedad de más de treinta y cinco mil lotes de módulos precompilados, los cuales se encuentran disponibles en un formato que facilita su integración en la Raspberry Pi. El sistema informático Raspberry Pi se encuentra en

desarrollo constante, con el afán en mejorar el equilibrio y el desempeño de varios lotes Debian como sea posible en la placa de la Raspberry Pi (Agus Kurniawan, 2018).

Tabla 3: Características de sistema operativo Raspbian

Información del Sistema Operativo Raspbian	
Equipo desarrollador	Comunidad Raspbian
Modelo de desarrollo	Código libre y abierto
Núcleo	Linux
Plataformas admitidas	Raspberry Pi

Fuente: (Agus Kurniawan, 2018)

2.9.3. Cli raspberry

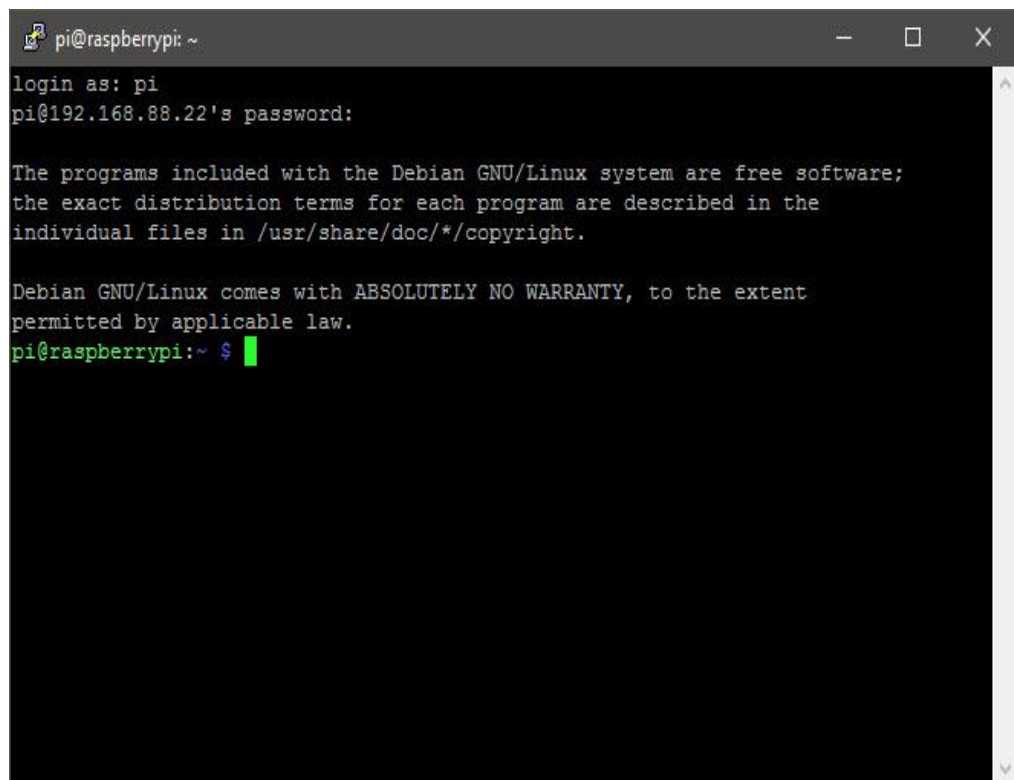
En realidad, Linux hace referencia al núcleo o kernel del sistema operativo, que es el elemento central que controla todas las acciones realizadas en el sistema informático. Al hablar sobre el hecho de que X "ejecuta Linux", se hace referencia normalmente al kernel del sistema y al conjunto de herramientas que se utilizan en conjunto con él.

El núcleo Linux tiene la capacidad de ejecutar diversos tipos de software en una variedad de plataformas de hardware. Una computadora puede desempeñar funciones de servidor, que consiste en procesar datos en nombre de otros usuarios, o de computadora de escritorio, donde el usuario interactúa directamente con ella. También puede ser utilizado como una herramienta para desarrollar software. Linux no está limitado a una función específica en el sistema y puede realizar múltiples tareas dependiendo de cómo se configuren las aplicaciones y su ejecución.

La Interfaz de Línea de Comandos (CLI) es una manera comunicarse con un sistema de cómputo mediante la introducción de comandos de texto en lugar de hacer clic en opciones de menú en un entorno visual de usuario (GUI). La Terminal es la aplicación o plataforma que ofrece este tipo de interfaz al usuario.

El interfaz de consola (CLI) recibe las ordenes escritos por el usuario y los envía a un shell. El shell interpreta las instrucciones escritas por el usuario y las ejecuta en el sistema operativo (SO). Si el comando específico produce una salida, se muestra en la terminal. En caso de encontrar algún problema con los comandos, se muestra una advertencia de equivocidad en la pantalla. (Agus Kurniawan, 2018).

Figura 16: Interfaz de línea de comandos de Raspberry Pi



```
pi@raspberrypi: ~
login as: pi
pi@192.168.88.22's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
pi@raspberrypi:~ $
```

Fuente: (Agus Kurniawan, 2018)

La figura 16 muestra la consola del sistema operativo Raspbian, la línea de comandos nos permite navegar y hacer muchas operaciones.

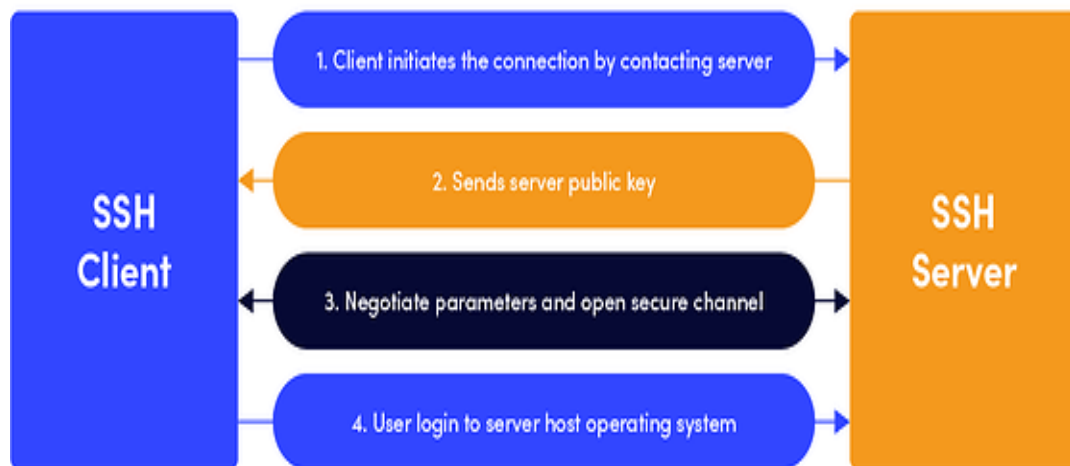


2.9.4. Protocolo ssh

El Servicio de Shell Seguro (SSH, por sus iniciales en el idioma inglés), este tiene la tarea de proporcionar un acceso seguro al intérprete de comandos. Su finalidad es garantizar que la información transmitida entre cliente y servidor se mantengan confidenciales e íntegros, incluso en redes inseguras. SSH fue inventado para mejorar la gestión remota de los aparatos, dado que otros sistemas que usaban protocolos como Telnet, no ofrecían un nivel de seguridad adecuado. Debido a sus funciones criptográficas, SSH ofrece un encriptado robusto, técnicas de autenticación criptográfica y fuerte protección contra violaciones de integridad. Es importante señalar que, si bien el servicio inicialmente ofrece gestión y administración remota del terminal, sus implementaciones pueden ir mucho más allá, asegurando en ocasiones la comunicación de otros servicios y empaquetando diferentes tipos de implementaciones de protocolo. (León Rodríguez, 2018).

El proceso de conexión SSH sigue el modelo de cliente y servidor, con el cliente iniciando la comunicación con el servidor. El cliente SSH es responsable de configurar la conexión y autenticar el servidor SSH utilizando la criptografía de clave pública. Una vez que se completa la fase de configuración, el protocolo SSH utiliza un fuerte algoritmo de cifrado simétrico y algoritmos hash para asegurar que los datos transferidos entre el cliente y el servidor estén protegidos tanto en términos de privacidad como de integridad.

Figura 17: Funcionamiento del protocolo de comunicación SSH



Fuente: (Michael W Lucas, 2018)

La Figura 17 ilustra la comunicación entre un cliente y un servidor SSH.

2.9.5. Vnc viewer

Virtual Network Computing (VNC) es un sistema de espacio de trabajo gráfico compartido que permite el control remoto de la interfaz de escritorio de una computadora desde otra computadora o dispositivo móvil mediante la ejecución de un servidor VNC. (Das et al., 2021).

Remote Framebuffer, o RFB, es el protocolo que regula el formato de los datos que se pasan entre el cliente y el servidor en el sistema VNC. Esto es lo que permite a un cliente ver y controlar otra computadora de forma remota. Es aplicable a todas las aplicaciones y sistemas de ventanas, lo que significa que funciona en plataformas como Windows, macOS, Linux y otros sistemas operativos populares.

El lugar donde se sienta el usuario, con las capacidades de pantalla, mouse y teclado, se denomina cliente o visor RFB. El lugar donde se originan los cambios del framebuffer (como en el sistema de ventanas) se llama servidor RFB. Remote Framebuffer está diseñado para que los clientes puedan ejecutarse en la más amplia gama



de hardware y para que la implementación de un cliente sea lo más simple posible, con muy pocos requisitos necesarios por parte del cliente. RFB comenzó como un protocolo muy simple, pero se ha mejorado para incluir funciones como la transferencia de archivos, una compresión más refinada y medidas de seguridad más sólidas a medida que se ha ido desarrollando.

2.10. ENRUTADOR

Un enrutador es un aparato que permite la comunicación de red que conecta muchas redes de área local y redes de área amplia en el internet. Tiene las funciones de interconexión de redes, procesamiento de datos y administración de redes y desempeña un papel vital en Internet (Wang et al., 2022).

El TL-WR940N es un enrutador de red de conexión inalámbrica y por cable que se ha diseñado especialmente para pequeñas empresas y redes domésticas. Su tecnología MIMO permite un excelente y avanzado rendimiento inalámbrico, excelente para la transferencia de video de alta definición, juegos online y VoIP. Su elegante diseño exterior cuenta con un botón WPS que asegura WPA2, lo que previene intrusiones externas en la red.

El TL-WR940N tiene la especificación del IEEE 802.11n, lo que permite establecer una red sin conexión física que alcanza velocidades hasta quince veces superiores y una cobertura 5 veces mayor que los productos 11g comunes. Además, con velocidades de transmisión de hasta 450 Mbps, lo que demuestra capacidades aún mejores para evitar la pérdida de datos en largas distancias al atravesar obstrucciones en una oficina pequeña, un apartamento grande e incluso un edificio de hierro. (Tp-link, 2020).

Figura 18: Router TL-WR940N



Fuente: (Tp-link, 2020)

Tabla 4: Características de Hardware de Router Tp-Link TL-WR940N

CARACTERÍSTICAS DEL HARDWARE	
Interfaz	4 PUERTOS DE RED LAN 10/100Mbps 1 PUERTO DE RED WAN10/100Mbps
Pulsador	Botón sin conexión física on/ off, Botón WPS-Reset, Botón de on / off
Fuente de alimentación del router	12Voltios / 1Amperio
Reglas Inalámbricas	Los estándares IEEE 802.11n, 802.11g y 802.11b
Medidas	9.1 x 5.7 x 1.4 pulg. (230 x 144 x 35mm)
Antenas	3 emisores desacopables omnidireccionales de 5 dBi (RP-SMA)

Fuente: (Tp-link, 2020)



Tabla 5: Características de Router Tp-Link

CARACTERÍSTICAS INALÁMBRICAS	
Frecuencia de trabajo	2.4 a 2.4835GHz
Velocidad de datos	11n: logra 450Mbps(dinámica) 11g: logra 54Mbps(dinámica) 11b: logra 11Mbps(dinámica)
Sensibilidad de entrada de señal	270M: -68dBm@10% PER 130M: -68dBm@10% PER 108M: -68dBm@10% PER 6M: -88dBm@10% PER 1M: -90dBm@8% PER
Potencia de emisión de señal	CE: <20dBm(2.4GHz) FCC: <30dBm
Modos sin conexión física	Router Mode, Range Extender (for V5 and later), Access Point Mode (for V5 and later)



(Continuación...) Funciones sin conexión física	Habilitar / Deshabilitar Radio Inalámbrico, Puente WDS, WMM, Estadísticas Inalámbricas
Protección Inalámbrica	64/128/152-bit WEP / WPA / WPA2, WPA-PSK / WPA2-PSK

Fuente: (Tp-link, 2020)

Tabla 6: Características de Software de Router Tp-Link TL-WR940N

CARACTERÍSTICAS DE SOFTWARE	
Eficiencia del servicio	WMM, Control de Banda Ancha
WAN	IP Dinámica/IP Estática/PPPoE/ PPTP (Acceso Dual) /L2TP (Acceso Dual) /BigPond
Gestión	Reglamento de Acceso Gestión Local Gestión Remota
DHCP	Servidor, cliente, relación de cliente DHCP
Reenvío de puertos	Servidor virtual, encendido de puertos, UPnP, DMZ
DNS dinámico	DynDns, Comexe, NO-IP



(Continuación...)	
Pase por VPN	PPTP, L2TP, IPSec (ESP Head)
Verificación de Acceso	Control Parental, control de administración de manera local, lista de host, horario de Acceso, administración de reglas
Seguridad de firewall	DoS, SPI Firewall IP Address Filter/MAC Address Filter/Domain Filter IP and MAC Address Binding
Protocolos	Soporta IP versión 4 e IP versión 6
Red de invitados	2.4GHz

Fuente: (Tp-link, 2020)

2.11. SERVIDOR

Un servidor es un sistema o una computadora que provee a otras computadoras, llamadas clientes, recursos, datos, programas o servicios a través de una red. Las computadoras se consideran servidores cuando comparten recursos con las máquinas cliente. Hay varios tipos de servidores, incluidos servidores web, servidores de correo electrónico y servidores en la nube.

El propósito de un dispositivo como servidor es para responder a las solicitudes de los clientes a través de una conexión de red, lo que implica la configuración para escuchar dichas solicitudes. Esta funcionalidad puede estar incorporada en el sistema

operativo, en aplicaciones instaladas, en roles específicos o en una integración de estos elementos. Por ejemplo, el sistema informático Windows Server de la empresa Microsoft permite la recepción y respuesta de peticiones de los clientes, mientras que las funciones y actividades adicionales instaurados en el servidor amplían su capacidad de respuesta. Otro ejemplo sería el servidor web Apache, que utiliza una aplicación separada del sistema operativo para atender las solicitudes del navegador web. Cuando un cliente necesita acceder a datos o funcionalidades del equipo servidor, envía una petición a través de la red de internet y el equipo servidor la recibe y responde adecuadamente. Este modelo de intercambio entre cliente y servidor es conocido como solicitud y respuesta, o también como modelo de petición-respuesta.

Con frecuencia, un proveedor de servicios llevará a cabo numerosas tareas adicionales como parte de una única solicitud y respuesta, como verificar la identidad del solicitante, asegurarse de que el cliente tenga permiso para acceder a los datos o recursos solicitados en el formato apropiado, o devolver la respuesta solicitada. en el plazo previsto (Paessler, 2020).

Figura 19: Servidor



Fuente: (Christopher Negus, 2020)



2.11.1. Samba

Samba es una herramienta de libre código que implementa el protocolo de compartición de ficheros de Microsoft Windows, anteriormente conocido como SMB y ahora conocido como CIFS, en sistemas computacionales tipo UNIX. Esto permite que las computadoras que ejecutan sistemas informáticos como GNU - Linux, MacOS o Unix, puedan actuar como clientes o servidores en redes de Windows. (Banda Fernández, 2021).

Samba se basa en el protocolo común cliente - servidor de Bloque de mensajes del servidor (SMB) y Sistema de archivos de Internet común (CIFS). Mediante el uso de software de cliente que también es compatible con SMB/CIFS (por ejemplo, la mayoría de los productos de Microsoft Windows), un usuario final envía una serie de solicitudes de cliente al servidor Samba en otra computadora para abrir los archivos de esa computadora, acceder a una impresora compartida o acceder a otros recursos. El servidor Samba en la otra computadora responde a cada solicitud del cliente, otorgando o denegando el acceso a sus archivos y recursos compartidos (Dirk Deimeke et al., 2021).

2.12. DATASET

Se puede denominar conjunto de datos o dataset a cualquier grupo de registros con nombre. Estas colecciones pueden incluir varios tipos de información, como registros médicos., registros de seguros, grupos de imágenes, entre otros, que son utilizados por programas que se ejecutan en el sistema. Además, las recopilaciones de datos se utilizan para almacenar información que necesitan las aplicaciones o el sistema informático, como el código fuente, las bibliotecas de macros o variables y los parámetros del sistema. Para conjuntos de datos que contienen texto legible, puede imprimirlos o mostrarlos en una consola (muchos conjuntos de datos contienen módulos de carga u otros datos binarios



que no son realmente imprimibles). Los conjuntos de datos se pueden catalogar, lo que permite hacer referencia al conjunto de datos por su nombre sin especificar dónde se almacena (Magnus Ekman, 2021).

2.13. ANACONDA NAVIGATOR

Anaconda Navigator es una herramienta de línea de comandos visual incluida en la distribución de Anaconda que permite la gestión de paquetes, entornos y canales dentro de Anaconda sin necesidad de utilizar la interfaz de línea de comandos. Los paquetes se pueden encontrar en Anaconda.org o utilizando Navigator para acceder a un repositorio local de Anaconda. Funciona con Windows, Mac OS X y Linux. Numerosos paquetes científicos requieren versiones específicas de otros programas para funcionar. (Dr. Poornima G. Naik, 2019).

Anaconda es una herramienta para administrar paquetes y entornos que se usa principalmente desde la línea de comandos, lo que permite a los científicos de datos garantizar que cada versión de cada paquete tenga todas las dependencias necesarias y funcione según lo previsto. Por otro lado, Anaconda Navigator es una interfaz de usuario que elimina la necesidad de escribir comandos en la terminal y permite buscar, instalar, ejecutar y actualizar paquetes de forma intuitiva sin necesidad de hacerlo.

- El paquete de software de código abierto conocido como Anaconda incluye una serie de herramientas para el procesamiento y análisis de datos a gran escala, así como para la computación científica, incluidos Jupyter y Spyder. Los lenguajes de programación R y Python también son compatibles con Anaconda.
- Python se implementa usando Spyder. En spyder, podemos usar opencv para python. El sistema de administración de paquetes de anaconda realiza un



seguimiento de las versiones de los paquetes. Anaconda debe instalarse antes de poder usarse.

- Anaconda es una herramienta del lenguaje de programación de alto nivel Python y R destinada a facilitar la administración y la implementación de datos en la informática científica, como la disciplina que se especializa en el estudio y análisis de datos, el uso de técnicas de modelado automático, el manejo y procesamiento de grandes conjuntos de datos, así como la creación de modelos para la predicción de resultados futuros.
- Para Windows, Linux y MacOS, la distribución ofrece paquetes de ciencia de datos. Anaconda Distribution o Anaconda Individual Edition son dos productos de Anaconda, Inc., mientras que Anaconda Team Edition y Anaconda Enterprise Edition, que no son gratuitos, son otros dos productos de Anaconda, Inc.
- El sistema de administración de paquetes de Anaconda realiza un seguimiento de las versiones de los paquetes. Este administrador de paquetes se convirtió en un paquete de código abierto distinto porque resultó ser valioso en sí mismo, no solo para Python.
- Miniconda es una versión mínima de Anaconda con arranque que solo incluye conda, Python, sus dependencias y algunos otros paquetes.
- La distribución de Anaconda viene con más de 250 paquetes preinstalados, y hay disponibles más de 7500 paquetes de código abierto adicionales en PyPI, junto con el administrador de entornos virtuales y el paquete conda.



- La principal diferencia entre conda y pip es cómo se mantienen las dependencias del paquete, lo cual es una dificultad clave para el aprendizaje automático de Python y la base de la existencia de conda.
- Conda realiza un análisis del entorno actual, incluyendo todos los elementos instalados, y determina cómo instalar un conjunto de dependencias compatibles, teniendo en cuenta las limitaciones de versión especificadas. Si no es posible, se emite una advertencia.
- Con el comando de instalación de conda, los usuarios pueden instalar paquetes de código abierto individualmente desde el repositorio de Anaconda, Anaconda Cloud o su propio repositorio o espejo privado.
- Anaconda, Inc. compila y crea los paquetes en el repositorio de Anaconda y distribuye archivos binarios para Windows de 32-64 bits, Linux de 64 bits y MacOS de 64 bits.
- Los paquetes personalizados se pueden crear con el comando conda build y compartir con otros a través de Anaconda Cloud, PyPI u otros repositorios.
- Anaconda2 viene con Python 2.7 por defecto, mientras que Anaconda3 viene con Python 3.7 por defecto. Sin embargo, cualquier versión de Python suministrada con anaconda se puede usar para construir nuevos entornos.
- El navegador Anaconda contiene casi todas las bibliotecas necesarias. Es sencillo hacer un cuaderno de proyecto de ciencia de datos.
- Anaconda Navigator es lo suficientemente adaptable para funcionar en una variedad de contextos de Python para satisfacer nuestras necesidades. Las actividades del foro están fácilmente disponibles en la comunidad.



- La interfaz de usuario es sencilla y fácil de operar. Hacer un cuaderno Jupyter es fantástico, ya que es una herramienta fantástica para aprender scripts de Python ejecutándolos línea por línea.
- Anaconda Navigator nos permite acceder rápidamente a documentos y carpetas. Las sugerencias automáticas de código de Spyder son fantásticas.
- Jupyter, Spider, R y otras herramientas esenciales de programación y ciencia de datos están disponibles en Anaconda.
- El símbolo del sistema de Anaconda permitió un fácil uso e instalación de numerosas bibliotecas de Python.
- El navegador Anaconda hace que software como Jupyter, Spyder, R y QT Console sean accesibles. Anaconda es fácil de configurar, incluso si no tenemos muchas habilidades técnicas. Jupyter simplifica el desplazamiento por los archivos y la adición de nuevas bibliotecas.



CAPITULO III

MATERIALES Y MÉTODOS

3.1. MATERIALES

3.1.1. Hardware

Computador:

- Ejemplar: HP Pavilion 17 Notebook PC
- Procesador: AMD A10-5745M APU with Radeon(tm) HD Graphics, 2100 MHz
- Memoria instalada (RAM): 8.00GB de RAM
- Sistema: Sistema Operativo de 64 bits con Windows Home

Raspberry Pi:

- Modelo: RaspberryPi 4 - B
- Procesador: Broadcom BCM2711, Quad-Core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz
- Memoria (RAM): 4 GB LPDDR4-2400 SDRAM
- Almacenamiento: 16GB memoria SD
- Características adicionales: 2.4 GHz y 5.0 GHz IEEE 802.11B/g/n/ac Wireless LAN, Bluetooth 5.0, Gigabit Ethernet de doble realidad

Cámara Pi:

- Lente de enfoque: OV5647 de 5 megapíxeles

- Resolución: 2592 x 1944

3.1.2. Software

- Sistema Operativo de 64 bits Windows 10 home
- Sistema operativo Raspbian
- Microsoft Word 2016 profesional
- Google Colab Python
- Anaconda Navigator – Spyder

3.1.3. Caja de adquisición de datos

Figura 20: Caja de adquisición de datos, con medidas de 35 x 35 x 35 cm^3



Elaboración propia

Figura 21: Iluminación con luces led de 12V



Elaboración Propia

3.2. DISEÑO Y NIVEL DE LA INVESTIGACIÓN

3.2.1. Diseño de la investigación

El diseño se refiere al plan o método que se elabora con el propósito de recopilar la información necesaria para una investigación y dar respuesta a la pregunta o problema planteado, existen dos tipos de diseños de investigación, los cuales son los experimentales y no experimentales. El término "experimento" tiene al menos dos acreditaciones, una genérica y otra específica. El general se refiere a "hacer una elección" y luego "observar las consecuencias", y el específico se refiere a un estudio en el que una o más variables independientes se cambian a propósito (causas previstas), para examinar los efectos del cambio en uno o más variables dependientes (consecuencias previstas), mientras el investigador se encuentra en un entorno controlado. (Sampieri Hernández, Roberto Méndez Valencia & Mendoza Torres, 2018). Esta investigación es no experimental y se

realizará en el ambiente de negocios unión y dignidad de la ciudad de Puno, que se ubica en la avenida Simón Bolívar – 21001.

3.2.2. Nivel de la investigación

El nivel de profundidad con el que se investigan ciertos fenómenos o eventos de la realidad social en una investigación se conoce como nivel de investigación. El nivel de investigación en este caso es exploratorio, lo que significa que se busca descubrir nuevos horizontes en el desarrollo del conocimiento humano mediante la exploración de un tema en particular e indagan desde una perspectiva innovadora (Sampieri Hernández, Roberto Méndez Valencia & Mendoza Torres, 2018). La actual investigación usa técnicas de inteligencia artificial para la clasificación de imágenes.

3.3. POBLACIÓN Y MUESTRA DE LA INVESTIGACIÓN

La población de esta investigación es finita, en su composición tiene imágenes de limones, de dimensiones de 670 x 630 en píxeles.

El universo acotado o población está conformada por la cantidad de imágenes, que en este caso es 40000. Para detectar un error del 5% con un nivel de confianza del 90%, se busca obtener muestras con una fórmula que permita calcular el tamaño adecuado de la población, considerando que la probabilidad de éxito es del 90%, mientras que la probabilidad de falla es del 10%. se calcula de la siguiente manera:

$$n = \frac{N \cdot Z_{\alpha}^2 \cdot p \cdot q}{d^2 \cdot (N - 1) + Z_{\alpha}^2 \cdot p \cdot q}$$

En donde:

N = tamaño de la población



Z_{α} = nivel de confianza

p = probabilidad de éxito proporción esperada

q = probabilidad de fracaso

d = precisión (Error máximo admisible en términos de proporción)

Si tenemos los valores $N= 2000$, $Z_{\alpha} = 1.645$ (Nivel de confianza 90%), $p = 0.9$, $q=0.1$ y $d=0.05$.

Los resultados se muestran con un valor de 92.93, que representa aproximadamente el 5%, lo que indica que para una población de 40000 se utilizará aproximadamente 2000 imágenes que representa nuestra muestra total, esto permite a la investigación obtener resultados deseados.

3.4. UBICACIÓN DE LA INVESTIGACIÓN

La investigación científica tendrá lugar en la localidad de Puno, que se encuentra en la provincia y el departamento del mismo nombre. La investigación tomará como referencia la zona comercial conocida como unión y dignidad, el cual es uno de los principales centros de abastos de la ciudad.

Figura 22: Mercado Unión y Dignidad de la ciudad de Puno



Elaboración propia

3.5. TÉCNICAS E INSTRUMENTOS PARA LA RECOLECCION DE DATOS

3.5.1. Técnicas

Existen varias estrategias de recolección, pero en este caso se usará la observación, este nos permite conocer el comportamiento del objeto de estudio de forma directa, porque se tiene entendido que en la actualidad la observación es una de las técnicas más confiables de recolección de datos, donde es necesario la utilización de dispositivos electrónicos para lograr su estudio con éxito. Los datos tienen que ser verificados en el proceso investigativo, en base a ello se hará el sistema de clasificación de imágenes de limones.



3.5.2. Instrumentos

Existen instrumentos mecánicos o electrónicos que se utilizan para recolectar datos específicos que tengan validez, confiabilidad y objetividad, estos son pasos que debe seguir el investigador para llegar a obtener datos confiables o llevar a cabo un estudio, es por ello que los medios que se utilizan para la recolección de datos son hardware y software que permita el almacenamiento de los datos como también la manipulación de estos, adicionalmente se verifican datos de eventos ocurridos en el procedimiento.



CAPITULO IV

RESULTADOS Y DISCUSIÓN

4.1. DATASET

Las imágenes adquiridas fueron clasificadas por el estado que se encuentran. En esta investigación las imágenes tienen una cantidad de 2000 imágenes en total, esto quiere decir que 1000 son limones en buen estado y 1000 son limones en mal estado, estas imágenes fueron redimensionadas a 670 x 630 píxeles, para lo cual se usó código de recorte en spyder.

La librería PIL (Python Image Library) de Python se utiliza para la manipulación de imágenes, existe el módulo Image de la librería PIL, que permite el recorte o redimensionamiento por medio del método Crop() (McKinney, 2022), el cual usa coordenadas o parámetros de izquierda, superior, derecha, inferior, las coordenadas son las siguientes: 1000, 600, 1670, 1230.

El módulo os se usa para manipular rutas, en el programa se usa la ruta para llegar a la carpeta de imágenes y lograr redimensionarlas. La manera de recortarlas muy eficientemente es con un bucle for que permite recorrer elementos de una carpeta u objeto iterable.

Para guardar los archivos que fueron recortados, se reemplaza con el mismo nombre y queda solo la imagen recortada en la carpeta y para ello se utiliza la función save(). Las imágenes que son recortadas serán usadas para en aprendizaje de la red neuronal.

Figura 23: Código de Python para el recorte de imágenes

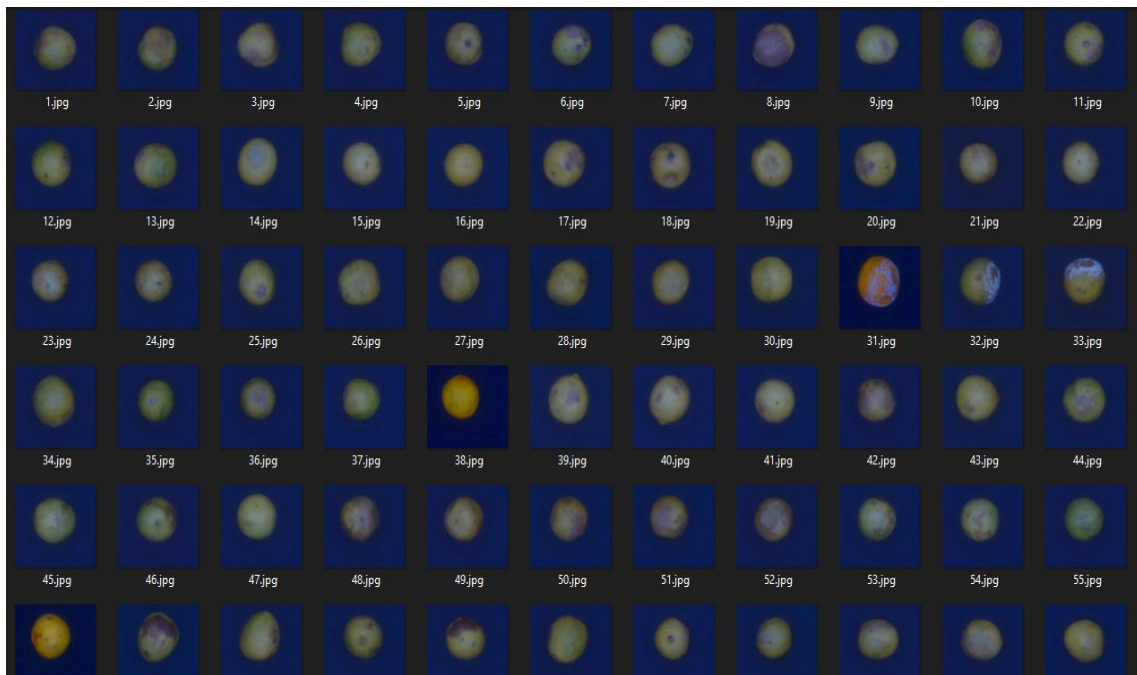
```
5 @author: Jharol
6 """
7 from PIL import Image
8 import os
9
10
11 for i in os.listdir('E:/Prueba'):
12     file = f"{'E:/Prueba'}\\{i}"
13     im = Image.open(file)
14     im = im.crop((1000,600,1670,1230))
15     im.save(file)
16
```

Elaboración propia

La figura

23 muestra el código utilizado para recortar las imágenes previamente fotografiadas con la cámara pi.

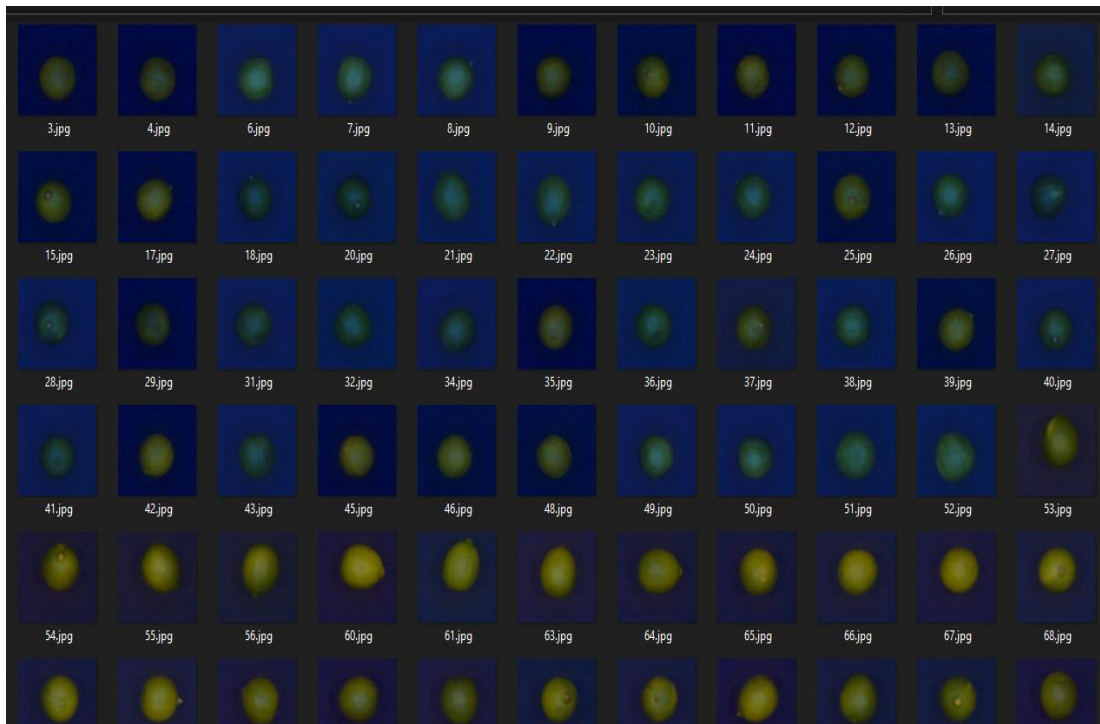
Figura 24: Los resultados del recorte de limones de la categoría “Limones en mal estado”.



Elaboración propia.

La figura 24 muestra las imágenes recortadas de la categoría “Limones en mal estado”.

Figura 25: Los resultados del recorte de limones de la categoría “Limones en buen estado”.



Elaboración propia.

La figura 25 muestra las imágenes recortadas de la categoría “Limones en buen estado”.

4.2. IMPORTACIÓN DE LIBRERÍAS

Para el desarrollo de cualquier modelo de aprendizaje profundo se deben seguir los siguientes pasos basados en tensorflow y keras:

- La creación de un modelo usando la función sequential de Keras.
- La compilación del modelo, donde se logra definir la función error y el optimizador que minimizará la función.
- El entrenamiento del modelo con un 80% del total de las imágenes.
- La predicción, donde se usa el modelo ya entrenado para procesar datos de la prueba que representa un 20% del total.



En el código se importa tensorflow que es uno de los primordiales requisitos para lograr un modelo de red neuronal, debido a que es el framework más utilizado que permite ejecutar operaciones matemáticas de una forma optimizada en una CPU, GPU, TPU (Kapoor et al., 2022) .

Layers o capas son los componentes básicos de las redes neuronales. Una capa consta de cálculo de tensor de entrada y de salida. Las capas mantienen su estado y se actualizan cuando recibe datos durante su entrenamiento y se almacenan.

Keras tiene el objetivo de desarrollar las redes neuronales más rápidamente porque reduce significativamente la cantidad de código para implementar una red neuronal, es una API. En nuestra red neuronal se usa sequential de keras, esta función crea una especie de contenedor vacío, la cual nos permite agregar secuencialmente diferentes capas del modelo o elementos que definirán las características del modelo (Sridhar & Adari, 2019).

La biblioteca Numpy se utiliza para trabajar con vectores y matrices de gran tamaño y para realizar operaciones matemáticas complejas. Su función principal es proporcionar una estructura de datos denominada ndarray, que permite trabajar con matrices de varias dimensiones. Esta biblioteca también ofrece una amplia variedad de funciones matemáticas de alto nivel.

Tenemos OS y PIL que permiten manipulación de las rutas e imágenes respectivamente en nuestro sistema operativo.

Se puede imprimir un mensaje en el monitor u otro dispositivo de salida estándar utilizando la función print(). El mensaje puede estar en forma de cadena o cualquier otro objeto, el cual será convertido en cadena antes de ser mostrado en la pantalla. Para ver que versión tiene nuestro tensorflow usamos print(tf.__version__).

Su sintaxis está dada por los siguientes parámetros:



- Object(s): Puedes imprimir cualquier cantidad de objetos que desees, y antes de imprimirlos, se convertirán a formato de cadena de caracteres.
- Sep: Especifica cómo separar los objetos, si hay más de uno. El valor predeterminado es ' '.
- End: Especifica qué imprimir al final. El valor predeterminado es '\n' (salto de línea).
- File: Un objeto con un método de escritura. El valor predeterminado es sys.stdout.
- Flush: Un valor booleano que especifica si la salida se vacía (verdadero) o se almacena en búfer (falso). El valor predeterminado es falso.

Pathlib es un módulo de Python creado para facilitar el trabajo con rutas en un sistema de archivos. Este módulo debutó en Python 3.4 y fue propuesto por el PEP 428. Proporciona una abstracción de alto nivel que funciona bien en sistemas POSIX, como Linux y Windows. Abstrae la representación del camino y proporciona las operaciones como métodos (McKinney, 2022).

Matplotlib.pyplot es un grupo de funciones que permite a Matplotlib trabajar de manera semejante a MATLAB. Debemos tener en cuenta que cada función de pyplot modifica una figura de alguna manera, como crear un nuevo gráfico, crear un área de trazado en la figura, dibujar líneas en el área de trazado, agregar etiquetas al gráfico, etc. Matplotlib.pyplot mantiene varios estados a través de las llamadas a las funciones, lo que significa que rastrea la figura actual, el área de trazado y las funciones de trazado se aplican a los ejes actuales. (se debe tener en cuenta que los "ejes" en matplotlib y en la mayoría de los lugares en la documentación se refiere a los ejes parte de una figura y no el término matemático eje) (Poladi, 2018).

La biblioteca csv proporciona funcionalidad tanto para leer como para escribir en archivos CSV. Diseñado para funcionar de manera inmediata con archivos CSV generados por Excel, se adapta fácilmente para trabajar con una variedad de formatos CSV. La biblioteca csv contiene objetos y otro código para leer, escribir y procesar datos desde y hacia archivos CSV (Kong et al., 2020).

Es necesario tener en cuenta que se puede usar el módulo google.colab para montar todo el Google drive. Luego de montar el drive en el Google colab, se puede interactuar con el Google drive como si fuera una carpeta en el entorno de colab. Uno de los puntos más importantes a tener en cuenta es que cualquier cambio que se realice en la carpeta se reflejará directamente en el Google drive.

Figura 26: Importación de librerías de Python

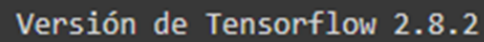
```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import numpy as np
import os
import PIL
import PIL.Image
import tensorflow as tf
#import tensorflow_datasets as tfds
print(tf.__version__)
import pathlib
import matplotlib.pyplot as plt
import csv
# Mount Google Drive
from google.colab import drive
```

Elaboración propia

La figura 26 muestra librerías que se utilizarán para la creación de la red neuronal convolucional y para graficar la precisión del modelo.

Luego de ejecutar la anterior celda obtenemos la versión de Tensorflow que se usa para el modelo.

Figura 27: Versión de TensorFlow instalado



```
Versión de Tensorflow 2.8.2
```

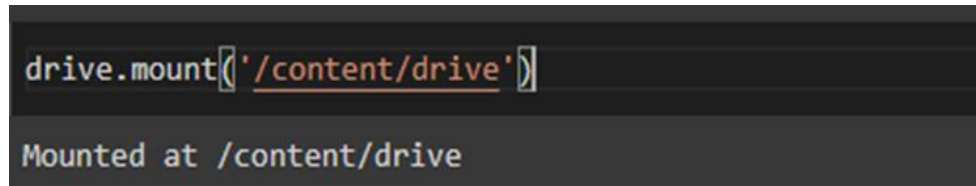
Elaboración Propia

4.3. EXPLORACIÓN DE LOS DATOS

Necesitamos trabajar con el conjunto de datos, el cual se encuentra en las carpetas del drive, para cargarlo al Google Colab se usan los siguientes pasos:

- En primer lugar, se carga las imágenes al drive
- Se ejecuta el script en el colab Shell “drive.mount('/content/drive')”
- Posteriormente tenemos el drive montado en nuestro Google colab

Figura 28: Importamos el drive al Google colab



```
drive.mount('/content/drive')  
Mounted at /content/drive
```

Elaboración propia.

La figura 28 nos muestra la celda de código que importa el Google drive a Google colab.

Al cargar nuestro drive, podemos listar los archivos con el comando “ls” el cual se utiliza para listar archivos o directorios en colab. Este comando es igual que navegar por el explorador de archivos con una interfaz gráfica de usuario, y se puede interactuar con ellos a través de comandos. Para nuestro caso queremos ver que las carpetas de limones buenos y limones malos están en el drive que hemos cargado, sabemos que este

se encuentra dentro de la ruta “/CNN-Lemon/data/train”, se muestra a continuación el resultado.

Figura 29: Se lista el contenido de la carpeta Train

```
!ls -l drive/MyDrive/ColabNotebooks/dataLemon/dataLemon/train

total 8
drwx----- 2 root root 4096 Sep 29 22:58 BadLemons
drwx----- 2 root root 4096 Sep 29 22:58 GoodLemons
```

Elaboración propia.

La figura 29 muestra el comando para listar la ruta especificada.

Para manejos eficientes de los directorios de limones buenos y malos, le asignaremos una variable llamada `data_dir`. Una variable es una manera fácil de identificar un dato almacenado en la memoria del ordenador. Es como un recipiente que guarda un dato y su valor puede cambiar a medida que el programa avanza. Las variables facilitan el acceso y manipulación del dato almacenado.

Asignar un valor a una variable es muy sencillo, solo se debe utilizar el operador “=”, la operación debe tener 3 componentes o tres partes:

- El nombre de la variable
- El operador “=”
- Un literal, una llamada a una función o simplemente como en nuestro caso asignaremos carpetas del drive

En seguida reemplazamos en la misma variable lo siguiente “`pathlib.Path(data_dir)`”, con ello le indicamos al programa que `data_dir` es una clase Path, es decir, una ruta.

Figura 30: Asignación de la ruta de la carpeta train a la variable data_dir

```
[ ] #data_dir = 'data/train/'  
    data_dir = 'drive/MyDrive/Colab Notebooks/CNN-Lemon/data/train/'  
    data_dir = pathlib.Path(data_dir)
```

Elaboración propia.

Len es una función integrada en python. Puede usar len() para obtener la longitud de la cadena, matriz, lista, tupla, diccionario, etc. dados. Se puede usar la función “len” para optimizar el rendimiento del programa. El número de elementos almacenados en el objeto nunca se calcula, por lo que len ayuda a proporcionar el número de elementos.

La sintaxis que usa es “len(value)”, al ejecutar esto nos devolverá un valor entero, es decir, la longitud de la cadena, matriz, y para nuestra red neuronal listamos la cantidad de imágenes que existen en las carpetas de train y val.

Las listas en el lenguaje de programación python son un tipo de estructura que guardan de forma ordenada un conjunto de datos que puede ser del mismo tipo o como también no puede serlo. En los archivos que tenemos, todos tienen la misma extensión, y a todas las imágenes las listamos (McKinney, 2022).

Aunque la interfaz del módulo glob es simple, su funcionalidad es muy amplia y potente. Este módulo es útil en cualquier situación en la que un programa necesite buscar en el sistema de archivos una lista de archivos que tengan nombres que coincidan con un patrón determinado. Para crear una lista de nombres de archivos que tengan una cierta extensión, prefijo, o cualquier cadena común en el medio, usar glob en lugar de escribir código personalizado para escanear los contenidos del directorio nos permite más eficiencia y menos código de Python JPG (Kong et al., 2020) . En nuestra red neuronal lo que hacemos es traer en una lista los nombres del subdirectorio GoodLemons y BadLemons con la extensión

Se muestra la cantidad de los limones, 800 de limones buenos y 800 de limones malos, para hacer las pruebas respectivas se reserva un 20% del total, es decir, 400 limones.

Figura 31: Se muestran la cantidad de Limones Buenos y Limones Malos



```
[12] image_count = len(list(data_dir.glob('GoodLemons/*.jpg')))
print(image_count)

800

[13] image_count = len(list(data_dir.glob('BadLemons/*.jpg')))
print(image_count)

800
```

Elaboración propia.

La figura 31 nos muestra el código que hace la cuenta de las imágenes de la carpeta de GoodLemons y BadLemons.

4.4. DATOS CARGADOS CON KERAS

El tamaño del lote es el número de puntos de datos en cada iteración del ciclo, también conocido como época. Esto es importante porque la red neuronal actualiza los pesos y el sesgo con mayor frecuencia. Debemos tener en cuenta que cuando hay grandes cantidades de datos, se requieren computadoras con mayor uso de memoria y las redes neuronales tardan más en ejecutar cada ciclo. Para abordar esto, debemos dividir los ciclos en iteraciones más pequeñas con menos puntos de datos; al hacerlo, no es necesario cargar todos los datos en la memoria a la vez y, como resultado, las redes neuronales se entrenan de manera muy rápida y efectiva. (Kapoor et al., 2022). Para nuestra red neuronal se usa un batch size de 32.

Las dimensiones de nuestras imágenes serán de 180 x 180 para la red neurona, las variables a la que se asignan se muestran a continuación. Dado que la canalización procesa lotes de imágenes que deben tener el mismo tamaño, debe proporcionarse.

Figura 32: Asignación de un batch size de 32 al modelo de la red.

```
batch_size = 32  
img_height = 180  
img_width = 180
```

Elaboración propia.

La figura 32 muestra las variables que posteriormente serán usadas para el preprocesamiento de imágenes.

Llamamos a la función “`tf.keras.preprocessing.image_dataset_from_directory()`”, este devuelve un `tf.data.Dataset` que permite producir lotes de imágenes de los subdirectorios `class_a` y `class_b`, junto con etiquetas de 0 y 1, donde 0 corresponde a la `class_a` y 1 corresponde a `class_b` (Tensorflow, 2022).

La “`tf.data.Dataset()`” `Dataset` proporciona una forma eficaz y descriptiva de escribir tuberías de entrada. Se sigue un patrón común al usar esta herramienta:

- Se genera un conjunto de datos de origen utilizando los datos de entrada.
- Se aplican transformaciones de conjunto de datos para realizar el preprocesamiento de los datos.
- Se itera sobre el conjunto de datos y se procesan los elementos.

La iteración ocurre en forma de transmisión, por lo que no es necesario que el conjunto de datos completo quepa en la memoria.

La función “`tf.keras.preprocessing.image_dataset_from_directory()`”, tiene varios parámetros que de detallan a continuación:

Tiene a “`data_dir`” que es el directorio donde se encuentran las imágenes.

- `Validation_split`: Es un número que puede tomar valores entre cero y uno, este determinará el porcentaje de los datos de entrenamiento, sabemos que el 20% de los datos deben ser de validación, es por ello que se debe el valor es 0.2.
- `Subset`: Es un subconjunto de los datos a devolver, uno de entrenamiento, validación, o ambos. Se usa siempre que esté configurado.
- `Seed`: Es un método, se utiliza para inicializar el generador de números sin premeditación. El generador de números sin premeditación necesita un numero con el cual empezar al que se le conoce como semilla, para generar un numero aleatorio. Esta semilla es opcional para barajar y hacer transformaciones, en nuestra red la usamos con el valor entero de 123
- `Imagen_size`: Es una variable donde se le da valores anteriormente ya definidos, el alto y el ancho de nuestras imágenes.
- `Batch_size`: Es una variable donde se ha definido anteriormente el cual tiene un valor de 32.
- `Labels`: Son etiquetas que se generan a partir de la estructura del directorio, todas tienen que tener etiquetas. Se puede generar también una lista o tupla de etiquetas enteras del mismo tamaño que la cantidad de archivos de las imágenes que se encuentran en el fichero. Estas etiquetas deben ordenarse de acuerdo con el orden alfanumérico de las rutas de los archivos de imagen.

Todo lo que se ha detallado anteriormente se le asigna una variable, tanto para validación y para test.

Figura 33: Preprocesamiento de las imágenes de Train

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(  
    data_dir,  
    validation_split=0.2,  
    #validation_split=None,  
    subset="training",  
    seed=123,  
    image_size=(img_height, img_width),  
    batch_size=batch_size,  
    #color_mode="grayscale",  
    labels = 'inferred')
```

Found 1600 files belonging to 2 classes.
Using 1280 files for training.

Elaboración propia

La figura 33 muestra el código que hace el preprocesamiento de las imágenes de la carpeta de Train que representa un 80% de los datos.

Figura 34: Preprocesamiento de las imágenes de Val.

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(  
    data_dir,  
    validation_split=0.2,  
    subset="validation",  
    seed=123,  
    image_size=(img_height, img_width),  
    batch_size=batch_size,  
    labels = 'inferred')
```

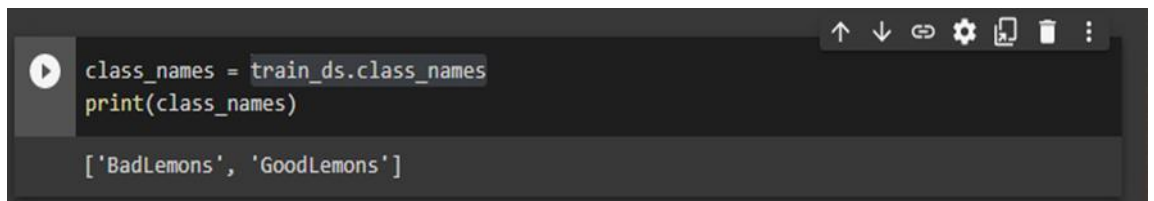
Found 1600 files belonging to 2 classes.
Using 320 files for validation.

Elaboración propia

La figura 34 muestra el código que hace el preprocesamiento de las imágenes de la carpeta de validación que representa un 20% de los datos.

Las clases en Python son una plantilla básica de un objeto. La clase nos proporciona elementos iniciales de estado e implementaciones del comportamiento. Para saber cuáles son las clases que se encuentran en nuestra carpeta de train, aplicamos la línea de código “train_ds.class_names” y luego las imprimimos. Veremos que tenemos Badlemons y GoodLemons.

Figura 35: Impresión de las clases de los limones



```
class_names = train_ds.class_names
print(class_names)

['BadLemons', 'GoodLemons']
```

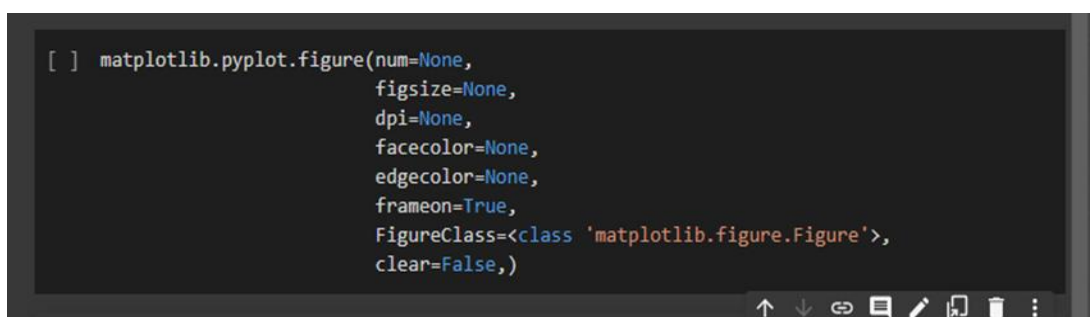
Elaboración propia

4.5. VISUALIZACIÓN DE LOS DATOS

La unidad de tamaño de figura nativa en Matplotlib es pulgadas, derivada de los estándares de la industria de la impresión. Sin embargo, es posible que los usuarios deban especificar sus cifras en otras unidades, como centímetros o píxeles. Para nuestro caso especificaremos el tamaño en pulgadas.

Usamos `matplotlib.pyplot.figure()` para cambiar o establecer las diversas propiedades de una figura Matplotlib. Los parámetros que usa se muestran a continuación.

Figura 36: Parámetros de figura de matplotlib



```
[ ] matplotlib.pyplot.figure(num=None,
                             figsize=None,
                             dpi=None,
                             facecolor=None,
                             edgecolor=None,
                             frameon=True,
                             FigureClass=<class 'matplotlib.figure.Figure'>,
                             clear=False,)
```

Elaboración propia.



El bucle for en el lenguaje Python se utiliza para iterar sobre los elementos de un objeto iterable, como una lista, tupla, conjunto o diccionario. Es importante tener en cuenta que cada vez que hace una iteración, solo tiene en cuenta un solo elemento de la carpeta u objeto iterable. Se pueden hacer muchas operaciones a la variable.

Para usar el bucle for basado en una secuencia numérica Python usa range, el cual devuelve un iterable donde los valores van desde 0 hasta el número indicado, sabemos que si inicia en 0 el número final tendrá que restarse en una unidad.

La función `plt.subplot` crea una figura y una cuadrícula con una sola llamada. Esta función la asignamos a la variable `ax` (`AxisSubplot`) que hace referencia a un objeto (Poladi, 2018).

La función `imshow()` en el módulo `pyplot` de la biblioteca `matplotlib` se usa para mostrar datos como una imagen; es decir, en un ráster regular 2D (McKinney, 2022). Esta función devuelve `AxisImage`. Los parámetros que tiene este método se describen a continuación:

- `cmap`: es un mapa de colores que se utiliza para mostrar la imagen. Algunos de los mapas de colores comunes incluyen 'gray', 'viridis', 'plasma', 'inferno', entre otros.
- `vmin` y `vmax`: estos parámetros establecen los valores menor y mayor para la escala de color. Si no se especifican, se utilizarán los valores menor y mayor de los datos de la imagen.
- `interpolation`: este parámetro determina cómo se interpolará la imagen si se redimensiona. Algunos de los valores comunes son 'nearest', 'bilinear', 'bicubic', entre otros.



- origin: este parámetro establece la ubicación del punto más bajo y a la izquierda de un objeto o área determinada.
- de la imagen. Los valores comunes son 'upper' para la esquina superior izquierda y 'lower' para la esquina inferior izquierda.
- aspect: este parámetro establece la relación de aspecto de la imagen. El valor 'auto' ajustará la relación de aspecto para que coincida con el tamaño de la figura.
- extent: este parámetro se utiliza para establecer los límites de la imagen en el eje x y el eje y. El valor debe ser una tupla en el orden (xmin, xmax, ymin, ymax).

La función `matplotlib.pyplot.title()` establece títulos, tiene títulos disponibles sobre el eje del centro, alineados con el borde izquierdo y alineados con el borde derecho. Los parámetros que tiene son: `etiqueta`, `fontdict=None`, `loc='centro'`, `pad=None`, `**kwargs`.

- Etiqueta: texto que se usa para el título
- Fontdict: un diccionario que controla la apariencia del texto del título.
- Loc: se refiere a la posición, tenemos el centro, izquierda, derecha
- Pad: el desplazamiento del título desde la parte superior de los ejes, en puntos.
- kwargs: son otros argumentos que pueden acompañar, por ejemplo, texto con propiedades válidas.

La función `matplotlib.pyplot.axis()`, esta función se utiliza para establecer propiedades en el eje del gráfico. Los parámetros que usa son “xmin, xmax, ymin, ymax”, si no se tiene en cuenta no podríamos visualizar los valores en los ejes del plano. En

nuestro programa no lo necesitamos, entonces lo que se hace es poner el argumento “off”, con ello evitamos los ejes (Poladi, 2018).

La función `Matplotlib.pyplot.draw()` se usa para volver a dibujar la figura actual, es decir, se usa para actualizar una figura alterada.

Los datos que se muestran a continuación son de `train_ds`.

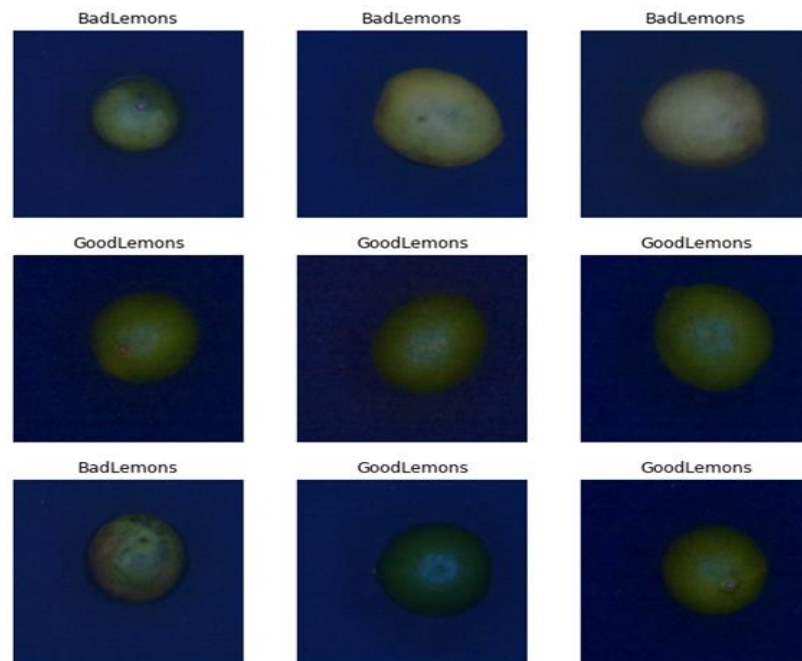
Figura 37: Código para mostrar 9 imágenes del quinto grupo de datos.

```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(5):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
    plt.draw()
```

Elaboración propia.

Una vez ejecutamos la celda, nos muestra 9 limones, con su respectiva etiqueta.

Figura 38: Limones etiquetados del quinto grupo.



Elaboración propia.

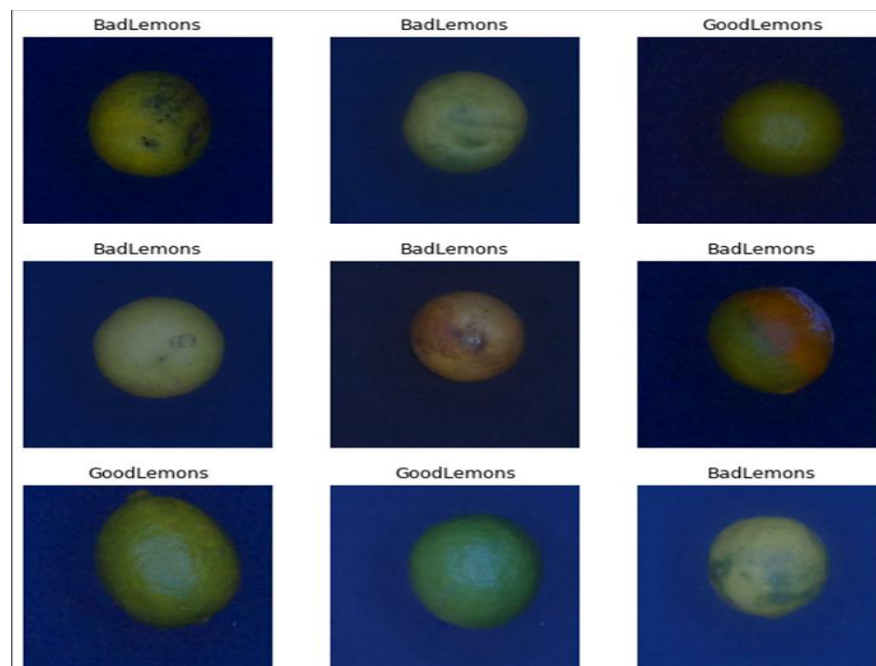
La figura 38 nos muestra los limones de la carpeta train, anteriormente se ha dividido las imágenes de train en grupos de 32 (Batch_size), en este caso estamos mostrando el grupo 5.

Figura 39: Código para mostrar 9 imágenes del sexto grupo de datos.

```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(6):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
    plt.draw()
```

Elaboración propia

Figura 40: Limones etiquetados del sexto grupo.



Elaboración propia.

La figura 41 muestra a `image_batch` que es un tensor de la forma (32, 180, 180, 3). Este es un lote de 32 imágenes de forma 180*180*3 (la última extensión se refiere a las rutas de color RGB). El `label_batch` es un objeto tensorial de la forma (32,), estas son las etiquetas correspondientes a las 32 imágenes. Mostramos a continuación la ejecución del código.

Figura 41: Mostramos en tensor de (32,180,180,3)

```
for image_batch, labels_batch in train_ds:  
    print(image_batch.shape)  
    print(labels_batch.shape)  
    break  
  
(32, 180, 180, 3)  
(32,)
```

Elaboración propia

4.6. CONFIGURACIÓN DE LOS DATOS PARA EL RENDIMIENTO

Nos debemos asegurar de utilizar la captación previa almacenada en búfer para que pueda obtener datos del disco sin que la E/S se convierta en un bloqueo, el código que lo hace posible es la figura 42. Estos son dos métodos importantes que debe usar al cargar datos:

- **Dataset.cache:** El objetivo es mantener las imágenes cargadas en la memoria después de ser leídas desde el disco durante la primera iteración, con el objetivo de evitar que el conjunto de datos obstaculice el entrenamiento del modelo, se pueden tomar algunas medidas preventivas. Esta técnica es especialmente útil para conjuntos de datos grandes que no caben en la memoria, ya que permite crear un caché en disco de alto rendimiento.
- **Dataset.prefetch:** Integra el preprocesamiento de datos y la ejecución del modelo durante el entrenamiento, superponiéndolos para mejorar la eficiencia del proceso.

Figura 42: Caché y Prefetch

```
AUTOTUNE = tf.data.AUTOTUNE  
  
train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)  
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

Elaboración propia

4.7. ESTANDARIZACIÓN DE LOS DATOS

El intervalo de valores del canal RGB es de 0 a 255, lo cual puede no ser óptimo para el aprendizaje de una red neuronal, ya que se recomienda que los valores de entrada sean pequeños en general. Existen dos formas de usar esta capa, el primero es aplicarlo al conjunto de datos llamando a `Dataset.map` y el segundo es incluir una capa dentro de la definición de su modelo, lo que permite simplificar la implementación. en este caso se usa el segundo enfoque.

Figura 43: Estandarización de los datos que están de 0 a 1

```
normalization_layer = tf.keras.layers.experimental.preprocessing.Rescaling(1./255)
```

Elaboración propia

La figura 43 muestra el código que hace posible que las imágenes que están en el rango de 0 – 255 ahora estén en el rango de 0 – 1.

Figura 44: Aplicación mediante `Dataset.map`

```
normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
print(np.min(first_image), np.max(first_image))

0.0 0.55675375
```

Elaboración propia

4.8. CREACIÓN DEL MODELO

`Layers.Conv2D`, en TensorFlow es una capa de convolución conocida como 2D que se utiliza en el modelado de redes neuronales. Esta capa aplica una convolución 2D a los datos de entrada utilizando un grupo de filtros para producir un tensor de salida. (Tensorflow, 2022). Se muestran algunos argumentos a continuación:

- `filters`: la cantidad de filtros a utilizar en la convolución.
- `kernel_size`: el tamaño de la ventana de la convolución.



- `strides`: el tamaño del paso de la convolución.
- `padding`: la forma en que se realiza el padding en la convolución.
- `activation`: la función de activación a aplicar después de la convolución.
- `input_shape`: la forma de los datos de entrada.

`t.f. _Keras.capas _MaxPool2D` es una función de TensorFlow que crea una capa de submuestreo 2D en una red neuronal de convolución utilizando el método de máximo agrupamiento. Esta capa reduce el tamaño espacial de la salida de la capa anterior aplicando un agrupamiento máximo en cada ventana de entrada de tamaño definido, seleccionando el valor máximo de la ventana y descartando el resto. Esto reduce la cantidad de parámetros en la red y ayuda a evitar el sobreajuste, manteniendo la información más relevante en la imagen. La función tiene argumentos que permiten especificar el tamaño de la ventana de agrupación, la tasa de agrupación y el modo de relleno. Se presenta argumentos que toma esta función.

- `pool_size`: especifica las dimensiones de la ventana de pooling, se puede definir un valor único o una tupla de dos valores.
- `strides`: especifica el desplazamiento de la ventana de pooling en cada dimensión, se puede definir un valor único o una tupla de dos valores.
- `padding`: especifica la forma en que se rellenan los bordes de la imagen de entrada, puede ser "valid" (sin relleno) o "same" (se rellena la imagen de entrada para que la salida tenga la misma forma que la entrada).
- `data_format`: especifica el orden de las dimensiones de la entrada, puede ser "channels_first" (ejemplo: (batch_size, channels, height, width)) o "channels_last" (ejemplo: (batch_size, height, width, channels)).



`tf.keras.layers.Dense` es una capa de red neuronal que implementa una operación de multiplicación de matrices seguida de una operación de adición. Esta capa se utiliza comúnmente como la última capa en una red neuronal para producir la salida final del modelo. La función recibe un tensor de entrada y produce un tensor de salida aplicando una transformación lineal a la entrada, seguida de una función de activación opcional. El número de neuronas de la capa se especifica mediante un argumento de la función.

El modelo secuencial en TensorFlow es una manera sencilla de crear una red neuronal en la que las capas se apilan de forma secuencial, de modo que cada capa se conecta solo a la capa siguiente. El modelo está constituido de varias capas y cada capa realiza una transformación en los datos de entrada.

En general, este modelo secuencial de una red neuronal esta constituida de las siguientes capas:

- Capa de entrada: Define el tamaño de los datos de entrada y se especifica en la primera capa del modelo.
- Capas ocultas: Capas intermedias que realizan transformaciones en los datos de entrada para generar características más abstractas. Pueden ser capas convolucionales, capas de pooling, capas recurrentes, entre otras.
- Capa de salida: Capa final que produce la salida del modelo, que puede ser un valor numérico o una clasificación.

Cada capa puede tener parámetros personalizados, como la cantidad de neuronas o filtros, las dimensiones del núcleo, la función de activación, entre otros. (Tensorflow, 2022).

Figura 45: Creación del modelo de red neuronal

```
num_classes = 2
model = tf.keras.Sequential([
    layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_height, img_wid
layers.Conv2D(16, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Conv2D(32, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Conv2D(64, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Conv2D(16, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Conv2D(16, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Dropout(0.3),
layers.Flatten(),
layers.Dense(32, activation='relu'),
layers.Dense(num_classes, activation='softmax')
])
```

Elaboración propia

4.9. COMPILACIÓN DEL MODELO

Adam es una técnica de optimización que utiliza un descenso de gradiente estocástico para estimar los momentos de primer y segundo orden de manera adaptable. Además, tiene la ventaja de ser eficiente en términos computacionales, utilizar poca memoria y ser insensible a la escala diagonal de los gradientes. Debido a estas características, es una buena opción para problemas que involucran grandes conjuntos de datos y parámetros. (Kong et al., 2020).

Metrics es la lista de métricas que evaluará el modelo durante el entrenamiento y las pruebas. Cada uno de estos puede ser una cadena, una función o una `tf.keras.metrics.Metric` que es una instancia. Para obtener información sobre la precisión del entrenamiento y la validación en cada época, es necesario especificar el argumento 'metrics' al llamar a la función `Model.compile()`. (Charu C., 2018).

Figura 46: Compilación del modelo de red neuronal convolucional

```
model.compile(  
    optimizer='adam',  
    loss=tf.losses.SparseCategoricalCrossentropy(from_logits=True),  
    metrics=['accuracy'])
```

Elaboración propia

4.10. RESUMEN DEL MODELO

Para ver el resumen del modelo usamos el método Model.summary.

Figura 47: Resumen del modelo de la red neuronal

```
print(model.summary())
```

Model: "sequential"

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d (MaxPooling2D)	(None, 90, 90, 16)	0
conv2d_1 (Conv2D)	(None, 90, 90, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 45, 45, 32)	0
conv2d_2 (Conv2D)	(None, 45, 45, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 64)	0
conv2d_3 (Conv2D)	(None, 22, 22, 16)	9232
max_pooling2d_3 (MaxPooling2D)	(None, 11, 11, 16)	0
conv2d_4 (Conv2D)	(None, 11, 11, 16)	2320
max_pooling2d_4 (MaxPooling2D)	(None, 5, 5, 16)	0
dropout (Dropout)	(None, 5, 5, 16)	0
flatten (Flatten)	(None, 400)	0
dense (Dense)	(None, 32)	12832
dense_1 (Dense)	(None, 2)	66

```
=====  
Total params: 48,034  
Trainable params: 48,034  
Non-trainable params: 0  
=====  
None
```

Elaboración propia



4.11. ENTRENAMIENTO DEL MODELO

Una época es un entrenamiento que se le da a todo el conjunto de datos hacia adelante y hacia atrás. La época óptima solo se conoce a través de la experimentación (McKinney, 2022).

El objeto model se usa en la programación de redes artificiales para definir la arquitectura de la red artificial y compilar el modelo para su aprendizaje y evaluación. El objeto model se utiliza para especificar cómo se deben conectar las capas de la red neuronal y cómo se propagan las entradas a través de la red para producir salidas.

A medida que la pérdida disminuye, se considera que el modelo es cada vez más efectivo siempre que no haya sobreajuste. La eficacia del modelo en los conjuntos de aprendizaje y validación se determina a través del proceso de entrenamiento y validación. A diferencia de la precisión, que es un porcentaje, la pérdida es la adición de los errores que el modelo ha cometido en cada ejemplo del conjunto de entrenamiento o validación.

Cuando se menciona `validation_split` como parámetro de ajuste al ajustar el modelo Deep Learning, se divide los datos en dos partes para cada época, es decir, datos de entrenamiento y datos de validación. Entrena el modelo con datos de entrenamiento y valida el modelo con datos de validación comprobando su pérdida y precisión.

Los resultados del entrenamiento al final nos muestran una pérdida de 0.1221 y una precisión de 0.9594 y para la validación tenemos una pérdida de 0.0880 y una precisión de 0.9750. se muestra a continuación el resultado.

Figura 48: Entrenamiento del modelo

```
epochs=10
history = model.fit(
    train_ds,
    validation_data=val_ds,
    # epochs=10
    epochs=epochs
)

Epoch 1/10
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: UserWarning: "`sparse_categorical_crossentropy` received `fr
return dispatch_target(*args, **kwargs)
40/40 [=====] - 96s 2s/step - loss: 0.6391 - accuracy: 0.6242 - val_loss: 0.4430 - val_accuracy: 0.8344
Epoch 2/10
40/40 [=====] - 37s 935ms/step - loss: 0.3346 - accuracy: 0.8539 - val_loss: 0.1706 - val_accuracy: 0.9344
Epoch 3/10
40/40 [=====] - 37s 935ms/step - loss: 0.2189 - accuracy: 0.9148 - val_loss: 0.3288 - val_accuracy: 0.8281
Epoch 4/10
40/40 [=====] - 37s 934ms/step - loss: 0.1815 - accuracy: 0.9266 - val_loss: 0.1414 - val_accuracy: 0.9500
Epoch 5/10
40/40 [=====] - 37s 934ms/step - loss: 0.1644 - accuracy: 0.9375 - val_loss: 0.1073 - val_accuracy: 0.9656
Epoch 6/10
40/40 [=====] - 37s 931ms/step - loss: 0.1371 - accuracy: 0.9477 - val_loss: 0.1018 - val_accuracy: 0.9688
Epoch 7/10
40/40 [=====] - 37s 933ms/step - loss: 0.1366 - accuracy: 0.9492 - val_loss: 0.1648 - val_accuracy: 0.9219
Epoch 8/10
40/40 [=====] - 37s 930ms/step - loss: 0.1260 - accuracy: 0.9539 - val_loss: 0.1214 - val_accuracy: 0.9563
Epoch 9/10
40/40 [=====] - 37s 931ms/step - loss: 0.1216 - accuracy: 0.9586 - val_loss: 0.1048 - val_accuracy: 0.9688
Epoch 10/10
40/40 [=====] - 37s 933ms/step - loss: 0.1221 - accuracy: 0.9594 - val_loss: 0.0882 - val_accuracy: 0.9750
```

Elaboración propia

4.12. VISUALIZACIÓN DE LOS RESULTADOS DEL ENTRENAMIENTO

Después del entrenamiento creamos gráficos con los datos recopilados en el entrenamiento. Donde vemos claramente el comportamiento del modelo, es decir, nos muestra la precisión del entrenamiento y la precisión de la validación. Usamos matplotlib para los gráficos.

Figura 49: Código para la visualización de resultados

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

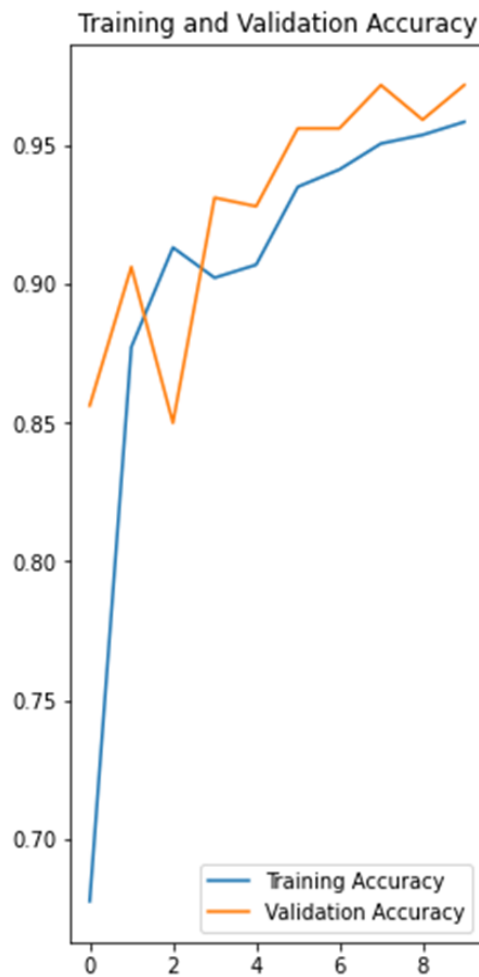
epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.draw()
```

Elaboración propia

Figura 50: Precisión del modelo de red neuronal.

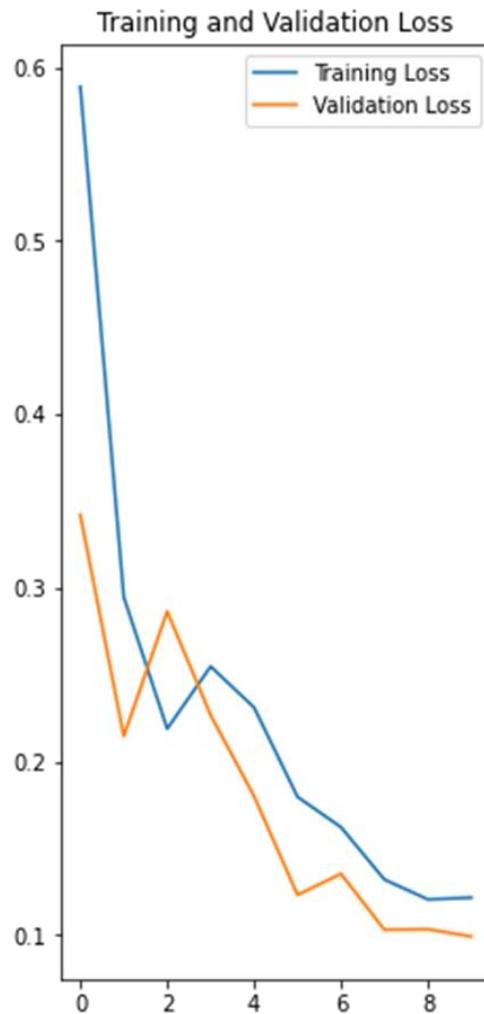


Elaboración propia

La figura 50 muestra la evolución de la precisión del modelo en el entrenamiento. La línea azul representa la precisión que se ha logrado con los datos de Train y la línea de color naranja representa la precisión que se ha logrado con los datos de Validación.

La precisión es un factor muy importante para un modelo de red neuronal, ya que de esto dependerá el correcto desempeño en la clasificación de limones, en caso de que la precisión no sea la deseada se debe volver a realizar el proceso del entrenamiento.

Figura 51: Disminución de error de la red neuronal



Elaboración propia

La figura 51 muestra la disminución del error en la red neuronal, la línea de color azul muestra el error que se ha obtenido al entrenar el modelo con los datos de Train y la línea de color naranja muestra el error obtenido con los datos de validación.

Finalmente, el modelo se usa para clasificar las imágenes de los limones que no se incluyó en los conjuntos de entrenamiento y validación.

4.13. PREDICCIÓN DE NUEVOS DATOS

Para la predicción de nuevos datos necesitamos importar el 20% de las imágenes que se han reservado y que se encuentran en la carpeta de test.

Figura 52: Importación de nuevos datos y la respectiva predicción

```
with open('out_Keras_Image_Preprocessing_Lemon.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Indice', 'state', 'confidence'])

for j in range (image_count):
    test_path = 'drive/MyDrive/ColabNotebooks/dataLemon/dataLemon/test/{}.jpg'.format(j)
    test_path = pathlib.Path(test_path)
    img = keras.preprocessing.image.load_img(
        test_path, target_size=(img_height, img_width)
    )
    img_array = keras.preprocessing.image.img_to_array(img)
    img_array = tf.expand_dims(img_array, 0)
    predictions = model.predict(img_array)
    score = tf.nn.softmax(predictions[0])
    print(
        "This image most likely belongs to {} with a {:.2f} percent confidence."
        .format(class_names[np.argmax(score)], 100 * np.max(score))
    )
    writer.writerow(["{}".format(j), "{}".format(class_names[np.argmax(score)]), "{:.2f}".format(np.max(score))])


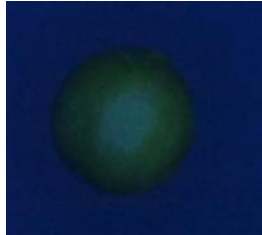

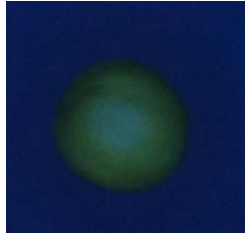


plt.show()
```

Elaboración propia

La figura 52 muestra el código que se utiliza para importar una nueva imagen del drive y mediante el modelo predecir a que categoría corresponde dicha imagen.



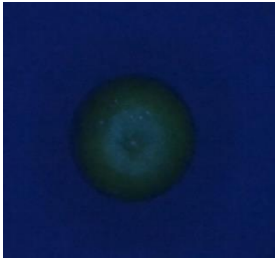



Los resultados se muestran a continuación.

Tabla 7: Resultados de clasificación con imágenes nuevas.

1) Lo más probable es que esta imagen pertenezca a BadLemons con un 73,10 por ciento de confianza.	2) Lo más probable es que esta imagen pertenezca a GoodLemons con un 68,35 por ciento de confianza.
	
3) Lo más probable es que esta imagen pertenezca a GoodLemons con un 66,02 por ciento de confianza.	4) Lo más probable es que esta imagen pertenezca a GoodLemons con un 68,43 por ciento de confianza.
	
5) Lo más probable es que esta imagen pertenezca a BadLemons con un 73,10 por ciento de confianza.	6) Lo más probable es que esta imagen pertenezca a BadLemons con un 73,11 por ciento de confianza.
	

Elaboración propia

Tabla 8: Resultados de clasificación con imágenes nuevas.

<p>Lo más probable es que esta imagen pertenezca a GoodLemons con un 56,56 por ciento de confianza.</p>	<p>Lo más probable es que esta imagen pertenezca a BadLemons con un 73,10 por ciento de confianza.</p>
	
<p>Lo más probable es que esta imagen pertenezca a GoodLemons con un 56,74 por ciento de confianza.</p>	<p>Lo más probable es que esta imagen pertenezca a BadLemons con un 73,11 por ciento de confianza.</p>
	
<p>Lo más probable es que esta imagen pertenezca a BadLemons con un 73,11 por ciento de confianza.</p>	<p>Lo más probable es que esta imagen pertenezca a BadLemons con un 73,11 por ciento de confianza.</p>
	

Elaboración propia



V. CONCLUSIONES

Se diseñó un sistema de clasificación de limones usando una red neuronal convolucional en el mercado unión y dignidad de la ciudad de Puno, se utilizó Google Colab y Python para lograr el entrenamiento del modelo de clasificación, finalizada la prueba se puso en funcionamiento consiguiendo clasificar los limones y obteniendo un asertividad considerable en el modelo.

Se generó un grupo de imágenes de 2000 imágenes de limones, estos fueron categorizados en: limones buenos que contiene 1000 imágenes y limones malos que contiene 1000 imágenes. Estas imágenes fueron adquiridas con una caja de 35 x 35 x 35 centímetros donde se puso iluminación que posteriormente tuvo que ser manipulada para obtener imágenes donde se pueda apreciar todas las características del limón.

Se diseñó un sistema de clasificación de imágenes de limones usando una red neuronal convolucional que se tuvo que entrenar con las imágenes que previamente se tomaron. El sistema evaluó imágenes que el sistema no conoce obteniendo un resultado de precisión del 90%.



VI. RECOMENDACIONES

Para futuras investigaciones se recomienda evaluar el nivel de iluminación óptimo para la adquisición de imágenes y automatizar el proceso usando técnicas de la electrónica. Se puede usar otras cámaras que identifiquen mejor las características de los limones.

Se recomienda para posteriores investigaciones incrementar la cantidad de imágenes de limones buenos y de limones malos para mejorar la precisión del sistema al clasificar las imágenes.



VII. REFERENCIAS BIBLIOGRÁFICAS

- Aguilar-Alvarado, J. V., & Campoverde-Molina, M. A. (2020). *Clasificación de frutas basadas en redes neuronales convolucionales*. 5(01), 3–22.
- Agus Kurniawan. (2018). *Raspbian OS Programming with the Raspberry Pi: IoT Projects with Wolfram, Mathematica, and Scratch* (1st ed.). Apress.
- Al-Saif, A. M., Abdel-Sattar, M., Aboukarima, A. M., & Eshra, D. H. (2021). Identification of Indian jujube varieties cultivated in Saudi Arabia using an artificial neural network. *Saudi Journal of Biological Sciences*, 28(10), 5765–5772.
- Alcarazo Ibañez, F. D. (2021). *CLASIFICACIÓN AUTOMÁTICA DE CITRUS AURANTIFOLIA USANDO VISIÓN ARTIFICIAL*. UNIVERSIDAD SEÑOR DE SIPÁN.
- Arabas, S., Jarecka, D., Jaruga, A., & Fijałkowski, M. (2014). Formula translation in Blitz++, NumPy and modern Fortran: A case study of the language choice tradeoffs. *Scientific Programming*, 22(3), 201–222. <https://doi.org/10.3233/SPR-140379>
- Banda Fernández, M. A. (2021). *Implementación del protocolo SAMBA usando carpetas compartidas para optimizar el proceso de distribución y ejecución de aplicaciones en ubicaciones remotas en la empresa SoftwarExpress Solutions S.A.C a través de cualquier navegador*.
- Berrones Reyes, M. C. (2019). *Clasificación De Mamografías Mediante Redes Neuronales Convolucionales*. UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN.
- Charu C., A. (2018). *Neural Networks and Deep Learning* (1st ed.). Springer.



Christopher Negus. (2020). *Linux Bible* (10a edición). Wiley.

Cuya Marzal, B. C. M., & Ramos Lugo, M. G. (2020). *Sistema de control de calidad utilizando redes neuronales para la clasificación del estado de la granadilla*.

UNIVERSIDAD DE LIMA.

Das, S., Jana, B., & Mandal, S. K. (2021). Implementation of dimming controlled visible light communication using Raspberry Pi. *Optical and Quantum Electronics*, 53(12). <https://doi.org/10.1007/s11082-021-03362-4>

Deepika Ghai, Suman Lata Tripathi, Sobhit Saxena, Manash Chanda, & Mamoun Alazab. (2022). *Machine Learning Algorithms for Signal and Image Processing* (Suman Lata Tripathi, Tripathi Suman Lata, Saxena Sobhit, Chanda Manash, & Alazab Mamoun (eds.)). Wiley-IEEE Press.

Definicion.de. (2020). *GOOGLE DRIVE*.

Del Castillo Huaccha, E. (2018). Desarrollo De Un Sistema De Visión Artificial Para Realizar Una Clasificación Uniforme De Limones. In *Universidad Privada del Norte*. Universidad Privada del Norte.

Dirk Deimeke, Daniel van Soest, Stefan Kania, Peer Heinlein, & Axel Miesen. (2021). *Linux-Server: Das umfassende Handbuch. Inkl. Samba, Kerberos, Datenbanken, KVM und Docker, Ansible u.v.m.* Rheinwerk Verlag GmbH.

Dr. Poornima G. Naik. (2019). *Python with Spyder : An Experiential Learning Perspective*. Shashwat Publication.

Edureka. (2019). *OS Module in Python*.

Google. (2022). *Colaboraty*.

<https://research.google.com/colaboratory/faq.html#:~:text=Colaboratory%2C> or



“Colab” for learning%2C data analysis and education.

Jaskolka, K., Seiler, J., Beyer, F., & Kaup, A. (2019). A Python-based laboratory course for image and video signal processing on embedded systems. *Heliyon*, 5(10).

<https://doi.org/10.1016/j.heliyon.2019.e02560>

Jiang, Y., Guo, X., Ni, H., & Jiang, W. (2022). Python-Based Visual Classification Algorithm for Economic Text Big Data. *Discrete Dynamics in Nature and Society*,

2022. <https://doi.org/10.1155/2022/4616793>

Kapoor, A., Gulli, A., Pal, S., & Chollet, F. (2022). *Deep Learning with TensorFlow and Keras: Build and deploy supervised, unsupervised, deep, and reinforcement learning models* (3rd ed.). Packt Publishing.

Kong, Q., Siau, T., & Bayen, A. (2020). *Python Programming and Numerical Methods: A Guide for Engineers and Scientists* (1er edición). Academic Press.

León Rodríguez, J. D. (2018). *Acciones de hardening para mejorar la seguridad de la información cuando se usan servicios de HTTP, LDAP, SSH Y SMTP*.

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA ESCUELA DE
CIENCIAS BÁSICAS TECNOLOGÍA E INGENIERÍA ESPECIALIZACIÓN EN
SEGURIDAD INFORMÁTICA BOGOTÁ D.C.

Liu, B., Wu, Q., Zhang, Y., Cao, Q., & Xu, X. (2020). Exploiting the Relationship between Pruning Ratio and Compression Effect for Neural Network Model Based on TensorFlow. *Security and Communication Networks*, 2020.

<https://doi.org/10.1155/2020/5218612>

Lutz, C., & Kumaigorodski, A. (2021). *Fast CSV Loading Using GPUs and RDMA for In-Memory Data Processing*. *Btw*. <https://doi.org/10.18420/btw2021-01>



- Magnus Ekman. (2021). *Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, Natural Language Processing, and Transformers Using TensorFlow* (1er edició). Addison-Wesley Professional.
- McKinney, W. (2022). *Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter* (3er edició). O'Reilly Media.
- Michael W Lucas. (2018). *SSH Mastery: OpenSSH, PuTTY, Tunnels and Keys (IT Mastery)*. Tilted Windmill Press.
- MINAGRI. (2017). *Informe del Limón* (p. 6). Ministerio de agricultura y riego, Direccion general de politicas agrarias.
- Mota-Delfin, C., Juárez-González, C., & Olguín-Rojas, J. C. (2018). Clasificación de manzanas utilizando visión artificial y redes neuronales artificiales. *Ingeniería y Región*, 20, 52–57.
- Narciso Horna, W. A., & Manzano Ramos, E. A. (2021). Sistema de visión artificial basado en redes neuronales convolucionales para la selección de arándanos según estándares de exportación. *Campus*, 26(32), 155–166.
- Ott, J., Pritchard, M., Best, N., Linstead, E., Curcic, M., & Baldi, P. (2020). A Fortran-Keras Deep Learning Bridge for Scientific Computing. *Scientific Programming*, 2020. <https://doi.org/10.1155/2020/8888811>
- Paessler. (2020). *Software profesional de monitoreo de host y servidor*.
- Poladi, S. R. (2018). *Matplotlib 3.0 Cookbook: Over 150 recipes to create highly detailed interactive visualizations using Python*. Packt Publishing.
- Saleh, H. (2020). *The Deep Learning with PyTorch Workshop: Build deep neural networks and artificial intelligence applications with PyTorch* (S. Zagad (ed.)).



Packt Publishing.

Sampieri Hernández, Roberto Méndez Valencia, S., & Mendoza Torres, C. P. (2018).

METODOLOGÍA DE LA INVESTIGACIÓN. In *News.Ge* (6ta ed.). McGraw-Hill Education.

Sebastian Raschka, Yuxi (Hayden) Liu, Mirjalili Vahid, & Vahid Mirjalili. (2022).

Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python (D. Dzhulgakov (ed.)). Packt Publishing.

SENASA. (2020). *IMPLEMENTACIÓN DE BUENAS PRÁCTICAS AGRÍCOLAS (BPA)* (p. 60). Ministerio de agricultura y riego.

<https://www.gob.pe/institucion/senasa/informes-publicaciones/924871-guia-de-buenas-practicas-agricolas-para-cultivo-de-limon>

Silvestre, I. de. (2020). *Tensor Machine Learning*.

Simon Monk. (2021). *Programming the Raspberry Pi, Third Edition: Getting Started with Python* (3er edición). McGraw Hill TAB.

Sridhar, A., & Adari, S. K. (2019). *Beginning Anomaly Detection Using Python-Based Deep Learning: With Keras and PyTorch* (1st ed. ed). Apress.

Tensorflow. (2022). *Google*.

Tian, S., Arshad, N. I., Toghraie, D., Eftekhari, S. A., & Hekmatifar, M. (2021). Using perceptron feed-forward Artificial Neural Network (ANN) for predicting the thermal conductivity of graphene oxide-Al₂O₃/water-ethylene glycol hybrid nanofluid. *Case Studies in Thermal Engineering*, 26(April), 101055.
<https://doi.org/10.1016/j.csite.2021.101055>

Tian, Z. (2021). *Use Python Data Analysis to Gain Insights from Airbnb Hosts. 2021*.



Tp-link. (2020). *TL-WR940N - Router Inalámbrico N a 450 Mbps*.

Wang, L., Cao, C., Ye, J., & Zhong, W. (2022). RW-Fuzzer: A Fuzzing Method for Vulnerability Mining on Router Web Interface. *Wireless Communications and Mobile Computing*, 2022. <https://doi.org/10.1155/2022/5311295>

Woan Ching, S. L., Lai, K. W., Chuah, J. H., Hasikin, K., Khalil, A., Qian, P., Xia, K., Jiang, Y., Zhang, Y., & Dhanalakshmi, S. (2022). Multiclass Convolution Neural Network for Classification of COVID-19 CT Images. *Computational Intelligence and Neuroscience*, 2022. <https://doi.org/10.1155/2022/9167707>

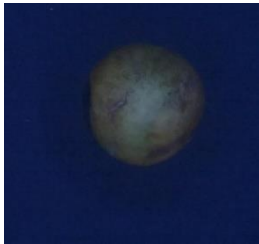
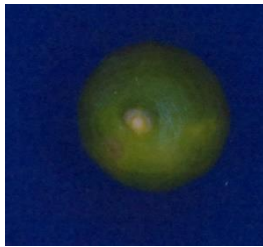


Yoav Goldberg. (2017). *Neural Network Methods in Natural Language Processing* (Graeme Hirst (ed.)). Morgan & Claypool Publishers.

Yossy, E. H., Pranata, J., Wijaya, T., Hermawan, H., & Budiharto, W. (2017). Mango Fruit Sortation System using Neural Network and Computer Vision. *Procedia Computer Science*, 116, 596–603.

ANEXOS


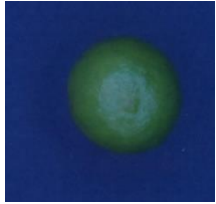




ANEXO 1

Tabla 1. 1: Resultados de la clasificación con nuevos datos.

1) Lo más probable es que esta imagen pertenezca a BadLemons con un 73,10 por ciento de confianza.	2) Lo más probable es que esta imagen pertenezca a GoodLemons con un 72,45 por ciento de confianza.
	
3) Lo más probable es que esta imagen pertenezca a GoodLemons con un 73,00 por ciento de confianza.	4) Lo más probable es que esta imagen pertenezca a BadLemons con un 73,10 por ciento de confianza.
	

Elaboración propia

Tabla 1. 2: Resultados de la clasificación con nuevos datos.

1) Lo más probable es que esta imagen pertenezca a BadLemons con un 73,10 por ciento de confianza.	2) Lo más probable es que esta imagen pertenezca a GoodLemons con un 72,87 por ciento de confianza.
	
3) Lo más probable es que esta imagen pertenezca a BadLemons con un 73,10 por ciento de confianza.	4) Lo más probable es que esta imagen pertenezca a BadLemons con un 73,10 por ciento de confianza.
	
5) Lo más probable es que esta imagen pertenezca a GoodLemons con un 56,38 por ciento de confianza.	6) Lo más probable es que esta imagen pertenezca a BadLemons con un 73,10 por ciento de confianza.
	

Elaboración propia



ANEXO 2

Código de Python

```
import pathlib

import numpy as np

from tensorflow import keras

import tensorflow as tf

from keras.models import load_model

modelo = load_model('D:/red/modelo.h5') # se carga el modelo entrenado

class_names = ['BadLemons', 'GoodLemons']#le indicamos las clases a la red neuronal

modelo.summary()#imprimimos el resumen de la arquitectura del modelo de red neuronal

data_dir = 'D:/prueba'#le indicamos la direccion de la carpeta donde se cargarán los datos

data_dir = pathlib.Path(data_dir)# hacemos que la cadena data_dir sea una ruta

image_count = len(list(data_dir.glob('* .jpg')))#buscamos todas las fotos con extensión .jpg

print(str("cantidad de limones en prueba: "),image_count)#imprimimos la cantidad

for i in range (image_count): #for se usa para iterar sobre nuestra carpeta imagen_count

    test_path = 'D:/prueba/{ } .jpg'.format(i)#format permite reempazar en las llaves

    test_path = pathlib.Path(test_path)#hacemos la conversion a ruta de la anterior variable
```



```
img = keras.preprocessing.image.load_img(test_path, target_size=(180,
180))#indicamos tamaño de imagen 180x180

img_array = keras.preprocessing.image.img_to_array(img)#preprocesamos

img_array = tf.expand_dims(img_array, 0)#expandimos la dimension

predictions = modelo.predict(img_array)# hacemos la predicción de la clase de la
imagen

score = tf.nn.softmax(predictions[0])#la predicción la llevamos a una distribución
de probabilidad

print(#imprimimos e resultado

      "Esta imagen probablemente pertenece a {} con un {:.2f} % de confiabilidad"

      .format(class_names[np.argmax(score)], 100 * np.max(score)))

estado="{}".format(class_names[np.argmax(score)])#agregamos la clase a la variable
estado

#El siguiente código es para la visualización de los leds indicadores de clases en Arduino

import serial, time

if (estado=="BadLemons"):

    arduino = serial.Serial("COM3", 9600)

    time.sleep(2)#espera dos segundos

    arduino.write(b'1')#enviamos un "1" a Arduino si el limon está en mal estado

    arduino.close()

else:
```



```
arduino = serial.Serial("COM3", 9600)
```

```
time.sleep(2)#esperar 2 segundos
```

```
arduino.write(b'0')#enviamos un "0" a Arduino si el limon está en mal estado
```

```
arduino.close()
```

#en Arduino tenemos un código que recibe los datos y hace el encendido del led correspondiente para cada caso.

ANEXO 3

Figura 1. 1: Conexión de Raspberry pi por VNC VIEWER



Elaboración propia

Figura 1. 2: Toma de imágenes de limones



Elaboración propia



DECLARACIÓN JURADA DE AUTENTICIDAD DE TESIS

Por el presente documento, Yo Jharol Iván Centeno Palomino
identificado con DNI 71459843 en mi condición de egresado de:

Escuela Profesional, Programa de Segunda Especialidad, Programa de Maestría o Doctorado
Ingeniería Electrónica

informo que he elaborado el/la Tesis o Trabajo de Investigación denominada:
"DISEÑO DE UN SISTEMA DE CLASIFICACIÓN DE LINUNES
USANDO UNA RED NEURONAL CONVOLUCIONAL EN EL
MERCADO UNION Y DIGNIDAD DE LA CIUDAD DE PUNO - 2022"

Es un tema original.

Declaro que el presente trabajo de tesis es elaborado por mi persona y no existe plagio/copia de ninguna naturaleza, en especial de otro documento de investigación (tesis, revista, texto, congreso, o similar) presentado por persona natural o jurídica alguna ante instituciones académicas, profesionales, de investigación o similares, en el país o en el extranjero.

Dejo constancia que las citas de otros autores han sido debidamente identificadas en el trabajo de investigación, por lo que no asumiré como mías las opiniones vertidas por terceros, ya sea de fuentes encontradas en medios escritos, digitales o Internet.

Asimismo, ratifico que soy plenamente consciente de todo el contenido de la tesis y asumo la responsabilidad de cualquier error o omisión en el documento, así como de las connotaciones éticas y legales involucradas.

En caso de incumplimiento de esta declaración, me someto a las disposiciones legales vigentes y a las sanciones correspondientes de igual forma me someto a las sanciones establecidas en las Directivas y otras normas internas, así como las que me alcanzan del Código Civil y Normas Legales conexas por el incumplimiento del presente compromiso.

Puno 12 de abril del 2023

FIRMA (obligatoria)



Huella



AUTORIZACIÓN PARA EL DEPÓSITO DE TESIS O TRABAJO DE INVESTIGACIÓN EN EL REPOSITORIO INSTITUCIONAL

Por el presente documento, Yo Jharel Iván Centeno Palomino identificado con DNI 71459843 en mi condición de egresado de:

Escuela Profesional, Programa de Segunda Especialidad, Programa de Maestría o Doctorado

Ingeniería Electrónica

informo que he elaborado el/la Tesis o Trabajo de Investigación denominada:

- DISEÑO DE UN SISTEMA DE CLASIFICACIÓN DE LIMONES USANDO UNA RED NEURONAL CONVOLUCIONAL EN EL TERRITORIO UNIÓN Y DIGNIDAD DE LA CIUDAD DE PUNO - 2022 -

para la obtención de Grado, Título Profesional o Segunda Especialidad.

Por medio del presente documento, afirmo y garantizo ser el legítimo, único y exclusivo titular de todos los derechos de propiedad intelectual sobre los documentos arriba mencionados, las obras, los contenidos, los productos y/o las creaciones en general (en adelante, los "Contenidos") que serán incluidos en el repositorio institucional de la Universidad Nacional del Altiplano de Puno.

También, doy seguridad de que los contenidos entregados se encuentran libres de toda contraseña, restricción o medida tecnológica de protección, con la finalidad de permitir que se puedan leer, descargar, reproducir, distribuir, imprimir, buscar y enlazar los textos completos, sin limitación alguna.

Autorizo a la Universidad Nacional del Altiplano de Puno a publicar los Contenidos en el Repositorio Institucional y, en consecuencia, en el Repositorio Nacional Digital de Ciencia, Tecnología e Innovación de Acceso Abierto, sobre la base de lo establecido en la Ley N° 30035, sus normas reglamentarias, modificatorias, sustitutivas y conexas, y de acuerdo con las políticas de acceso abierto que la Universidad aplique en relación con sus Repositorios Institucionales. Autorizo expresamente toda consulta y uso de los Contenidos, por parte de cualquier persona, por el tiempo de duración de los derechos patrimoniales de autor y derechos conexos, a título gratuito y a nivel mundial.

En consecuencia, la Universidad tendrá la posibilidad de divulgar y difundir los Contenidos, de manera total o parcial, sin limitación alguna y sin derecho a pago de contraprestación, remuneración ni regalía alguna a favor mío, en los medios, canales y plataformas que la Universidad y/o el Estado de la República del Perú determine, a nivel mundial, sin restricción geográfica alguna y de manera indefinida, pudiendo crear y/o extraer los resultados sobre los Contenidos, e incluir los Contenidos en los índices y buscadores que estimen necesarios para promover su difusión.

Autorizo que los Contenidos sean puestos a disposición del público a través de la siguiente licencia:

Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional. Para ver una copia de esta licencia, visita: <https://creativecommons.org/licenses/by-nc-sa/4.0/>

En señal de conformidad, suscribo el presente documento

Puno 12 de abril del 2023

FIRMA [obligatoria]



Huella