REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería Mecánica, Mecánica-Eléctrica y Mecatrónica

Clasificador por visión artificial y redes neuronales aplicado para análisis de calidad en frutas, Arequipa 2024

Tesis presentada por el Bachiller:

Loyola Canales, Gaston Alonso

ORCID: 0009-0002-1647-1088

Para optar el Título Profesional de Ingeniero Mecatrónico

Asesor:

Dr. Collado Oporto, Christiam Guillermo

ORCID: 0000-0002-0529-738X

Arequipa – Perú

2025

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA MECANICA, MECANICA-ELECTRICA Y MECATRONICA

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 23 de Mayo del 2025

Dictamen: 012984-C-EPIMMEM-2025

Visto el borrador del expediente 012984, presentado por:

2015701391 - LOYOLA CANALES GASTON ALONSO

Titulado:

CLASIFICADOR POR VISIÓN ARTIFICIAL Y REDES NEURONALES APLICADO PARA ANÁLISIS DE CALIDAD EN FRUTAS, AREQUIPA 2024

Nuestro dictamen es:

APROBADO

Titulo Profesional/Titulo de Segunda Especialidad/Grado Académico a optar:

INGENIERO MECATRÓNICO

29592341 - MESTAS RAMOS SERGIO ORLANDO
DICTAMINADOR

40302702 - QUISPE CCACHUCO MARCELO JAIME
DICTAMINADOR

29277716 - CUADROS MACHUCA JUAN CARLOS
DICTAMINADOR

Clasificador por visión artificial y redes neuronales aplicado para análisis de calidad en frutas, Arequipa 2024

INFORME DE ORIGINALIDAD

13%	11%	3%	7%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Universidad Católica de Santa María Trabajo del estudiante	2%
2	tesis.ucsm.edu.pe Fuente de Internet	1%
3	hdl.handle.net Fuente de Internet	1%
4	Submitted to Universidad Internacional de la Rioja Trabajo del estudiante	1%
5	Submitted to Pontificia Universidad Catolica del Peru Trabajo del estudiante	<1%
6	Submitted to Pontificia Universidad Catolica del Ecuador - PUCE Trabajo del estudiante	<1%
7	www.coursehero.com Fuente de Internet	<1%

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

DEDICATORIA

A mi padre, que desde el cielo guía mi camino profesional, que me enseño el valor del esfuerzo, tiempo y respeto que se pone para enfrentar cualquier reto en la vida, a mi madre que me enseño el valor de la resiliencia la cual me ayuda a jamás rendirme, buscando siempre ser más para servir mejor, gracias a ambos por todo el amor, apoyo y comprensión que me brindaron en todos estos años de formación, este logro siempre será de ustedes.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

AGRADECIMIENTOS

A mi hermana, que fue la pionera en esta generación de profesionales en mi familia, que nos llevó a todos a superarnos cada día, siempre mi admiración y mi respeto para ti, a mi familia y amistades que nunca dudaron de mí y me apoyaron en cada pequeño avance logrado en este proceso, gracias infinitamente.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

RESUMEN

El desarrollo del presente trabajo de investigación propone un sistema automatizado de clasificación por calidades en frutas basado en características visuales, con el fin de demostrar cuan efectivo logra ser un sistema de visión artificial y redes neuronales con hardware de costo accesible.

La clasificación se realizará considerando la forma, pigmentación y características típicas de manzanas tipo royal y naranjas tipo huando.

El desarrollo del clasificador se detalla en el documento en tres secciones, diseño de clasificador por visión artificial y redes neuronales, implementación en Arduino e implementación en Spyder, detallando en cada uno el código realizado, lógica utilizada y diagramas de flujo correspondientes a cada una de las etapas.

Además, se detallará las etapas de la implementación de software y hardware, así como la elección de los componentes utilizados y las modificaciones realizadas a lo largo de la implementación.

Adicionalmente se presentará resultados obtenidos en los diferentes modelos que se utilizaron para el entrenamiento de la red en términos de exactitud, error y capacidad de generalización, otorgando una justificación para el uso del modelo seleccionado.

El clasificador implementado trabajara con el modelo de red neuronal con arquitectura MobileNetV2, el cual presenta una exactitud general del 69%, y un promedio de precisión entre clases del 75%.

Palabras Clave: Clasificador, visión artificial, redes neuronales

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

ABSTRACT

The development of this research project proposes an automated fruit quality classification system based on visual characteristics, with the aim of demonstrating how effective an artificial vision system and neural networks can be when implemented with low-cost hardware.

The classification is performed by considering the shape, pigmentation, and typical characteristics of Royal apples and Huando oranges.

The development of the classifier is described in the document in three sections: the design of the classifier using computer vision and neural networks, implementation with Arduino, and implementation in Spyder, detailing in each case the code developed, the logic used, and the corresponding flowcharts for each stage.

Furthermore, the stages of software and hardware implementation are described, as well as the selection of the components used and the modifications made throughout the development process.

Additionally, the results obtained from different models used to train the neural network are presented, focusing on accuracy, error, and generalization capacity, providing justification for the selected model.

The implemented classifier works with a neural network model based on the MobileNetV2 architecture, which achieved an overall accuracy of 69% and an average class precision of 75%.

Keywords: Classifier, computer vision, neural network.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

ÍNDICE

DEDICATORIA
AGRADECIMIENTOS
RESUMEN
ABSTRACT
INTRODUCCIÓN 1
CAPÍTULO I 2
1. PLANTEAMIENTO DE LA INVESTIGACIÓN 2
1.1. Enunciado del problema 2
1.2. Identificación del problema 2
1.3. Descripción del problema 3
1.4. Área de conocimiento 4
1.4.1. Justificación social 4
1.4.2. Justificación académica 4
1.5. Objetivos de la investigación 5
1.5.1. Objetivo principal 5
1.5.2. Objetivos específicos 5
1.6. Interrogante 5
1.6.1. Interrogante principal 5
1.7. Alcances 5
1.8. Limitaciones 6
1.9. Hipótesis 6
1.9.1. Hipótesis general 6
1.10. Variables de investigación 6
1.10.1. Variable independiente 6
1.10.2. Variable dependiente 6
1.11. Antecedentes 7
1.11.1. Antecedentes Internacionales 7
1.11.2. Antecedentes Nacionales 7
CAPÍTULO II 9
2. MARCO TEÓRICO 9
2.1. Bases teóricas 9
2.1.1. Visión humana 9

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

2.1.2. Visión artificial 10
2.1.3. Aplicación de visión artificial 11
2.1.4. Procesamiento de imágenes visión artificial 14
2.1.5. Elementos de los sistemas de visión artificial 15
2.1.6. Etapas en el proceso de visión artificial 22
2.1.7. Adquisición de imagen 23
2.1.8. Preprocesamiento de imagen 24
2.1.9. Filtros para mejora de imagen 24
2.1.10. Segmentación 30
2.1.11. Extracción de características 32
2.1.12. Interpretación 33
2.2. Redes Neuronales artificiales 34
2.2.1. Introducción a las redes neuronales artificiales 34
2.2.2. Principios básicos de las redes neuronales artificiales 34
2.2.3. Tipos de redes neuronales artificiales 35
2.2.4. Algoritmos de aprendizajes en redes neuronales. 38
2.2.4.1. Funciones de activación. 58
2.2.5. Clasificación de frutas mediante inteligencia artificial 60
2.2.5.1. Métodos Clásicos de Clasificación. 60
2.2.5.2. Deep Learning (Aprendizaje Profundo) 64
CAPÍTULO III 65
3. DISEÑO DEL CLASIFICADOR POR VISION ARTIFICIAL Y REDES NEURONALES 65
3.1. Elección de entorno de desarrollo, lenguaje y controlador. 65
3.2. Datos de entrenamiento 66
3.3. Entrenamiento de la Red en Jupyter 71
3.4. Subir rutas de entrenamiento, validación y pruebas. 80
3.5. Implementación en Arduino 101
3.6. Implementación en Spyder. 104
3.7. Selección de faja 112
3.10. Selección de sensores y actuadores 116
3.11. Selección de cámara 119
3.12. Selección de actuadores 120

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

3.13. Selección de la iluminación 128
CAPÍTULO IV 132
4. PRUEBAS Y RESULTADOS DEL MODELO. 132
4.1.Pruebas y resultados. 134
4.2.Costo 144
4.3.Financiamiento 145
CONCLUSIONES 146
RECOMENDACIONES 147
5. REFERENCIAS 149
6. ANEXOS 151

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Índice de Tablas

Tabla 1 Diferencias entre filtros de paso alto. 30
Tabla 2 Tabla de diferenciación entre gradientes Batch, Estocástico y Mini Batch 41
Tabla 3 Diferencias entre las técnicas de regularización. 44
Tabla 4 Diferencia entre los diferentes tipos de arquitectura para redes neuronales convolucionales. 49
Tabla 5 Calidades de manzanas, naranjas y su estado 67
Tabla 6 Cantidad de imágenes dentro de carpetas de entrenamiento 69
Tabla 7 Cantidad de imágenes dentro de las carpetas de validación. 70
Tabla 8 Cantidad de imágenes dentro de la carpeta de pruebas 71
Tabla 9 Parámetros train accuracy vs val accuracy 89
Tabla 10 Distribución de conexión en Arduino 102
Tabla 11 Tabla de comparación de servomotores 122
Tabla 12 Consumo de corriente de towerpro mg995 124
Tabla 13 Tabla de toma de pruebas modelo. 142
Tabla 14 Presupuesto del proyecto 144

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Índice de Figuras

Figura 1 Proceso de obtención de información 9
Figura 2 Diagrama básico de la visión artificial. 10
Figura 3 Detección de defectos en fruta por pigmentación. 11
Figura 4 Proceso de metrología. 12
Figura 5 Diferencia y clasificación de diferentes frutos. 12
Figura 6 Proceso de verificación de montaje 13
Figura 7 Proceso de lectura de códigos 13
Figura 8 Aplicación Bin Picking con brazo robótico 14
Figura 9 Iluminación retroactiva difusa 16
Figura 10 Iluminación frontal 17
Figura 11 Luz direccional. 17
Figura 12 Luz Estructurada. 18
Figura 13 Lampara incandescente. 18
Figura 14 Lampara fluorescente. 19
Figura 15 Iluminación por fibra óptica. 19
Figura 16 Iluminación laser. 20
Figura 17 Iluminación led. 20
Figura 18 Diagrama de bloques de las etapas de un sistema de Visión Artificial. 23
Figura 19 Estructura de una red neuronal 35
Figura 20 Red neuronal Monocapa 36
Figura 21 Red neuronal Multicapa 36
Figura 22 Red neuronal convolucional. 37
Figura 23 Red neuronal recurrente. 38
Figura 24 Proceso de descenso de gradiente. 39
Figura 25 Capa convolucional. 45
Figura 26 Proceso de una red neuronal convolucional. 47
Figura 27 Algoritmo Support Vector Machines 61
Figura 28 Algoritmo Random Forest 62
Figura 29 Algoritmo K-Nearest Neighbors 63
Figura 30 Carpetas utilizadas para el entrenamiento de la red neuronal. 66
Figura 31 Carpetas de entrenamiento por calidades 66
Figura 32 Manzana Tipo Mq1 67

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 33 Naranja Tipo Nq1 68
Figura 34 Manzana Tipo Mq2..... 68
Figura 35 Naranja Tipo Nq2 69
Figura 36 Imágenes Dentro De Carpeta De Prueba 71
Figura 37 Creación de entorno de trabajo en anaconda navigator. 72
Figura 38 Librerías para modelo entrenado desde cero 73
Figura 39 Librerías para modelo con arquitectura mobilenetv2 73
Figura 40 Activando entorno virtual en anaconda prompt..... 74
Figura 41 Instalación de librerías en anaconda prompt. 74
Figura 42 Resumen de todas las capas, salidas y números de parámetros a entrenar de una red entrenada desde cero. 78
Figura 43 Resumen de todas las capas, salidas y números de parámetros a entrenar de una red con arquitectura MobileNetV2. 80
Figura 44 Proceso de entrenamiento por épocas para modelo entrenado desde cero. 85
Figura 45 Proceso de entrenamiento por épocas para modelo con arquitectura MobileNetV2. 85
Figura 46 Evaluación de error y accuracy general para modelo entrenado de cero..... 85
Figura 47 Evaluación de error y accuracy general para modelo con arquitectura MobileNetV2. 86
Figura 48 Gráfico de parámetros importantes para evaluación de modelo entrenado desde cero. 87
Figura 49 Gráfico de parámetros importantes para evaluación de modelo con arquitectura MobileNetV2..... 87
Figura 50 Array de predicciones tipo flotante para modelo entrenado desde cero. 91
Figura 51 Array de predicciones para modelo con arquitectura MobileNetV2. 91
Figura 52 Código y resultado de matriz tipo entero. 92
Figura 53 Muestra de data de entrenamiento con índice de clase. 93
Figura 54 Matriz de confusión de modelo entrenado desde cero..... 96
Figura 55 Matriz de confusión para modelo con arquitectura MobileNetV2. 97
Figura 56 Código y resultado del desempeño Para modelo entrenado desde cero. 98
Figura 57 Código y resultado del desempeño para modelo con arquitectura MobileNetV2. 98
Figura 58 Archivo h.5. 99

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Figura 59 Pantalla de ajuste de ROI. 132
Figura 60 Mensaje de aviso para movimiento de ROI. 132
Figura 61 Mensaje de coordenadas grabadas para ROI. 132
Figura 62 Ventana a la espera de detección para iniciar clasificación. 133
Figura 63 Fruta detectada ROI habilitado, clasificación en proceso. 133
Figura 64 Mensaje en consola después de detección. 134
Figura 65 Termino de programa a consecuencia de pulsar tecla q en teclado. 134
Figura 66 Faja original sin modificaciones. 112
Figura 67 Modificación para toma zona de toma de pruebas 113
Figura 68 Tapa extraíble para acceso a zona de toma de pruebas. 113
Figura 69 Interior de zona de toma de pruebas. 114
Figura 70 Zona de clasificación. 115
Figura 71 Montaje de paleta de clasificación 115
Figura 72 Anclaje de motorreductor 116
Figura 73 Prueba de sensor ultrasónico en faja. 117
Figura 74 Montaje de sensor infrarrojo ir1 118
Figura 75 Montaje de sensor infrarrojo ir2 118
Figura 76 Cámara web para toma de datos 119
Figura 77 Parámetros de configuración de Cámara 120
Figura 78 Posicionamiento de Cámara dentro de zona de toma de pruebas. 120
Figura 79 Selección del fruto más pesado. 121
Figura 80 Medición de distancia entre eje y paleta clasificadora 121
Figura 81 Servomotor TowerPro mg995 123
Figura 82 Fuente de alimentación para servomotor towerpro mg 995 125
Figura 83 Medición del diámetro del rodillo de la faja 125
Figura 84 Implementación de motorreductor y relé a 5v. 128
Figura 85 Imagen de prueba sin regularización de luz 129
Figura 86 Imagen de prueba iluminación perimetral 129
Figura 87 Zona de toma de datos modificada para apreciación de detalle. 130
Figura 88 Resultado final para toma de datos. 131
Figura 89 Conexión de sensores y actuadores con Arduino. 135
Figura 90 Etiquetas para orientación de conexión en cables. 135
Figura 91 Implementación física sensores y actuadores en faja. 136

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Figura 92 Manzanas y naranjas para prueba de modelo implementado. 136
Figura 93 Arduino y fuentes externas energizadas. 137
Figura 94 Entorno en Spyder para inicio de secuencia. 137
Figura 95 Roi utilizado para realización del proyecto. 138
Figura 96 Dosificación manual de manzanas en faja 138
Figura 97 Dosificación manual de naranjas en faja 139
Figura 98 Ventana de clasificación en tiempo real. 140
Figura 99 Confirmación de clasificación en consola. 140
Figura 100 Ejemplo de falla en clasificación. 141

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

INTRODUCCIÓN

En la actualidad, el proceso de clasificación de frutas aun presenta desafíos en cuanto a precisión y velocidad, aumentando el requerimiento de mano de obra para mayor tasa de producción. Muchas empresas no pueden costear sistemas avanzados de clasificación por medio de visión artificial. Por ello se propone desarrollar un sistema económico y funcional que permita clasificar frutas automáticamente (Kakani et al., 2020).

Frente a la problemática expuesta, el presente proyecto tiene como objetivo implementar y evaluar un clasificador basado en visión artificial y redes neuronales para analizar la calidad en frutas, en función de características visuales. Para el desarrollo del presente proyecto se utilizará una red neuronal convolucional basada en MobileNetV2 para lograr la clasificación de manzanas y naranjas en buen y mal estado, integrando todo con una faja transportadora y un microcontrolador Arduino para el control de sensores y actuadores los cuales comprenden a sensores infrarrojos, cámara web, motor DC y servomotor. También se observará la alternativa de solución de una red neuronal convolucional entrenada desde cero, otorgando ventajas y desventajas en su aplicación.

Este trabajo busca implementar y evaluar un Clasificador basado en visión artificial y redes neuronales para analizar la calidad en frutas, utilizando elementos de costo accesibles y con resultados aceptables, logrando otorgar una alternativa mejorable y adaptable para cualquier proceso de clasificación.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

CAPÍTULO I

1. PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1.Enunciado del problema

En la industria agroalimentaria, garantizar la calidad de las frutas es un desafío debido a la subjetividad de un proceso manual de clasificación. la falta de prototipos con visión artificial limita la eficiencia y uniformidad al evaluar la calidad, lo que afecta la competitividad de los productores en el mercado.

1.2.Identificación del problema

En la era moderna el mundo de la exportación de frutas, esta estandarizado a través de parámetros de calidad, que establecen los requisitos mínimos para poder producir cualquier tipo de fruta.

Alcanzar los estándares de calidad los cuales se basan en forma, tamaño, color, características únicas, peso, entre otras, no es una tarea sencilla debido a que, en jornadas largas de trabajo, utilizando la mano de obra humana, se entra en un nivel elevado de fatiga, el cual puede generar error, para poder solventar este problema tendrían que haber horarios rotativos lo cual generaría recurrir a más mano de obra humana, por lo tanto, un incremento en el costo de producción.

Para lograr conseguir los estándares de calidad con un funcionamiento continuo se puede utilizar en tal caso un sistema autómata de control de calidad y esto se puede conseguir utilizando estrategias como la visión artificial complementado con las redes neuronales, de esta manera se genera un sistema de control autómata que solo necesitaría de supervisión humana y en casos muy específicos la intervención de esta en el proceso, logrando así la disminución de costos, aumento en la eficiencia y reducción en la tasa de error evitando la fatiga.

Actualmente los países primermundistas, recurren a la automatización de los procesos, porque este es capaz de reducir costos, tiempo y mano de obra humana.

Si bien es cierto no es posible depender en un 100% de la automatización, la idea es requerir la menor cantidad posible de mano de obra humana, esto con el fin de utilizarla

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

únicamente para el monitoreo del proceso.

En el siguiente proyecto se identifica el siguiente problema ¿Como mejorar la exactitud, precisión y sensibilidad en la clasificación de frutas mediante el uso de visión artificial, redes neuronales y prototipo de hardware?

1.3.Descripción del problema

En la actualidad, en el mundo está pasando por una gran demanda de alimentos debido a la sobrepoblación existente, debido a ello se están ideando nuevas maneras de hacer mucho más eficiente la producción de alimentos mientras que se abaratan los costos (Kakani et al., 2020).

Solo tomando en cuenta la población peruana para el año 2010 nuestra población era de 29 millones de habitantes, para el año 2020 quedamos en el puesto siete de los países de América con mayor población, teniendo una cantidad de 32 millones de habitantes, y con una tasa de crecimiento del 1.7% y por último en el año 2023 tuvimos un incremento de la población a 33 millones de habitantes (INEI, 2024).

Un clasificador por visión artificial en concreto es un sistema diseñado para poder discriminar automáticamente objetos que se encuentren dentro del entorno de producción.

Este enfoque no solo mejorará la precisión en la evaluación de calidad, sino que también reducirá la dependencia de operadores humanos, incrementará la velocidad de procesamiento y reducirá costos a largo plazo, impactando positivamente en la competitividad de los productores de la región Arequipa.

Para maximizar los beneficios de la visión artificial se utilizan cámaras y un software especializado que se encargue de interpretar las imágenes capturadas, para de esta forma pueda tomar decisiones en tiempo real, esto permite generar una mayor eficiencia y reducir el error humano causado por cansancio en jornadas largas de trabajo.

Para todo ello, nos enfrentamos en su implementación a los siguientes desafíos.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

1. **Variabilidad de objetos**

   Debido a que el proyecto se especifica directamente al área de clasificación de frutas, se tendrá distintas formas, complicando la capacidad de distinguir de manera fluida entre uno y otro.

2. **Condiciones de iluminación**

   Como el proyecto se realiza con una contemplación de uso en un entorno controlado, se considera que el ambiente contara con una iluminación correcta en todo el periodo de uso.

3. **Velocidad de procesamiento**

   El programa deberá trabajar de tal manera que pueda distinguir entre frutas de manera rápida y eficiente.

4. **Adaptabilidad y aprendizaje**

   Se deberá crear una red neuronal para poder generar un aprendizaje adecuado, de esta manera optimizar el procesamiento (Medina Fernández, 2019).

1.4. Área de conocimiento

El proyecto de tesis se desarrolla primordialmente en el área de ciencias de la computación y la ingeniería mecatrónica, con énfasis en inteligencia artificial y visión artificial ya que se enfoca en la implementación, desarrollo y evaluación de un clasificador por visión artificial con redes neuronales para poder realizar la labor de discriminación en un sistema automatizado de clasificación de frutas.

1.4.1. Justificación social

Esta investigación contribuye al desarrollo tecnológico del sector agrícola, como Arequipa donde la agroindustria es fundamental, promoviendo el uso de tecnologías de fácil acceso para mejorar la competitividad de pequeños y medianos productores, fomentando de esa manera el desarrollo regional.

1.4.2. Justificación académica

Esta investigación representa una aplicación directa de conocimientos en inteligencia artificial y visión por computadora. Es una contribución al campo de estudio de las CNN aplicadas a la industria y puede servir de base para investigaciones futuras.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

1.5. Objetivos de la investigación

1.5.1. Objetivo principal

- Implementar y evaluar un Clasificador basado en visión artificial y redes neuronales para analizar la calidad en frutas, en función de características visuales en la región de Arequipa 2024.

1.5.2. Objetivos específicos

- Diseñar un modelo de redes neuronales para un clasificador de frutas por categorías de calidad basada en características visuales.
- Implementar un sistema de visión artificial y hardware que capture y procese imágenes en tiempo real.
- Integrar el clasificador con el prototipo de hardware.
- Evaluar el desempeño del clasificador por visión artificial y redes neuronales en términos de exactitud y precisión.

1.6. Interrogante

1.6.1. Interrogante principal

¿Un sistema basado en visión artificial y redes neuronales podrá mejorar la exactitud y eficiencia en la clasificación de frutas por calidad?

1.7. Alcances

El alcance de la presente investigación es evaluar un sistema clasificador por visión artificial y redes neuronales que integrara software y hardware para la mejora del proceso de selección de frutas, para ello se incorporarán sensores, actuadores además de lenguaje de programación Python con el objetivo de lograr superar la interrogante previamente planteada.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

1.8. Limitaciones

- El clasificador estará limitado a la detección de manzanas tipo Royal y naranjas tipo Huando.
- El clasificador trabajara con las características de color y forma, permitiendo su clasificación en buen y mal estado.
- La dosificación de la fruta será manual.
- La validación del prototipo se realizará en un entorno controlado.
- El rendimiento del sistema podría estar limitado por la calidad y diversidad de el conjunto de datos de entrenamiento además de utilizar hardware de costo moderado.

1.9.Hipótesis

1.9.1. Hipótesis general

La implementación de un sistema clasificador por visión artificial y redes neuronales mejorará la exactitud y precisión en la clasificación de la calidad de la fruta. En comparación con métodos manuales.

1.10. Variables de investigación

1.10.1. Variable independiente

Clasificador basado en visión artificial y redes neuronales

1.10.2. Variable dependiente

Calidad en la clasificación de frutas.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

1.11.Antecedentes

1.11.1. Antecedentes Internacionales

Según Heras (2017) en su artículo de investigación titulado "Clasificador de imágenes de frutas basado en inteligencia artificial", tuvo como propósito construir y diseñar e implementar un clasificador que pueda ser aplicado en muchas áreas de industria, con el fin de poder utilizar la inteligencia artificial como herramienta de clasificación, para ello realizo una serie de pasos para poder tener acceso a los datos necesarios para el análisis de las imágenes, siendo los pasos los siguientes, digitalización de imágenes, preparación de datos, extracción de características y aprendizaje supervisado, el aporte de esta investigación radica en la explicación detallada que se entrega para poder lograr el análisis de los datos entregados por la imagen en cuestión en el programa Python.

Según Carolina et al. (2009) en su artículo de investigación titulado "adquisición y reconocimiento de imágenes por medio de técnicas de visión e inteligencia artificial", tuvo como propósito diferenciar y constatar las diferencias entre las técnicas de visión e inteligencia artificial en la detección de imágenes, realizando estas pruebas con una faja transportadora controlado mediante lógica difusa que se encargaba de posicionar el elemento dentro del foco de la cámara, posteriormente utilizaba la metodología de procesamiento de ruido, segmentación por umbral y el realce de bordes para facilitar la detección de formas. El aporte de esta investigación radica en los resultados obtenidos sobre la mejor manera de poder trabajar el procesamiento de imágenes, dando resultados con porcentajes de tasa de éxito.

Según Santa María Pinedo et al. (2021) en su artículo de investigación titulado "Reconocimiento de patrones de imágenes a través de un sistema de visión artificial en MATLAB", tuvo como propósito realizar una investigación del tipo aplicada, donde utilizo un grupo experimental y un grupo de control, y aplico conocimientos algorítmicos y técnicas de filtrado digital para procesamiento de imágenes y patrones. El aporte de esta investigación radica en su logro de describir y correlacionar el reconocimiento de patrones y el sistema de visión artificial.

1.11.2. Antecedentes Nacionales

Según Medina Fernández (2019) en su tesis titulada "Diseño e implementación de un prototipo de clasificación de lúcumas aplicando visión artificial" tuvo como objetivo

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

proponer un sistema automatizado de clasificación teniendo en cuenta el tamaño y peso, para lograr disminuir el error en clasificación debido al trabajo manual, asi como tiempo y coste. El aporte de esta investigación radica en los pasos realizados para poder obtener los resultados finales, logrando este clasificar exitosamente las lúcumas estrictamente por su tamaño y peso empleando esto la visión artificial y las redes neuronales.

Para Gamonal Chanco (2020) en su tesis titulada "Diseño de un sistema por visión artificial para determinar la calidad de mandarinas" tuvo como finalidad diseñar un algoritmo que clasifique las mandarinas según las siguientes características, tamaño, forma, color y defectos, está basada en NTP, para de esta manera poder tener un control estandarizado de calidad. Para lograr el objetivo utiliza el procesamiento digital de imágenes que consiste en la adquisición de una imagen a través de una cámara web, para posteriormente realizarle un pre procesamiento, seguido de la segmentación por el método de regiones, posteriormente hacer una descripción y reconocimientos para finalmente la interpretación, el aporte de esta investigación radica en el software propuesto para esta aplicación siendo este el MATLAB, otorgando un panorama amplio sobre cómo se puede resolver la problemática fuera del software PYCHARM.

Según Mundaca Vidarte (2016) en su tesis titulada "Detección de caracteres de placas de automóviles mediante técnicas de visión artificial", tuvo como finalidad generar un sistema que permita la identificación de caracteres en las placas de automóviles, considerando que no debe importar la perspectiva de la captura, la importancia de esta investigación radica en los posfiltros utilizados para el procesamiento de imágenes y en la mejora del tiempo de respuesta computacional, además de enseñarnos el método de correlación de Pearson para el reconocimiento de caracteres.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

CAPÍTULO II

2. MARCO TEÓRICO

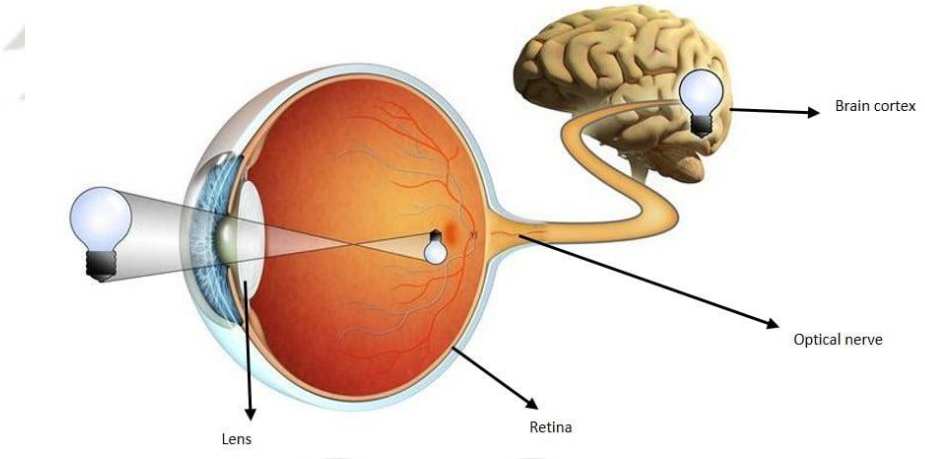
2.1. Bases teóricas

2.1.1. Visión humana

La visión humana es sin duda uno de los sentidos más importantes en el ser humano, es un proceso complejo que involucra los ojos y el cerebro, los cuales están conectados por medio de una red de neuronas, receptores y células especializadas, en gran medida, la visión humana es capaz de procesar la información automáticamente para interpretar la información del mundo exterior, mediante sistemas internos de codificación y representación, todo esto a partir de la información obtenida de las imágenes retinianas (Álvaro M. Pons Moreno & Francisco M. Martínez Verdú, 2004).

Figura 1

Proceso de obtención de información



*Nota***.** Tomado de Pucrs, 2018

Como se puede observar en la Figura 1, el proceso de obtención de la información del objeto pasa desde la detección del mismo, posteriormente la luz pasa a través del ojo, la imagen se invierte en la retina y esa información pasa al cerebro por medio del nervio óptico, y gracias al cerebro es que la imagen recobra su sentido de rotación correcto.

La interfaz cerebro visión del ser humano es capaz de procesar entre 10 y 12 imágenes por segundo de forma separada, sin embargo al superar esta cantidad de captaciones se procederá a

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

interpretar como un movimiento, sin embargo la visión a partir de los 60 fotogramas por segundo, ya no es capaz de distinguir la diferencia con facilidad en el cambio de los fotogramas por segundo, pero es necesario distinguir que aunque no seamos capaces de percibir la diferencia entre 60 FPS y 120 FPS, esta última es capaz de entregarnos una sensación de notable (Medina Fernández, 2019).

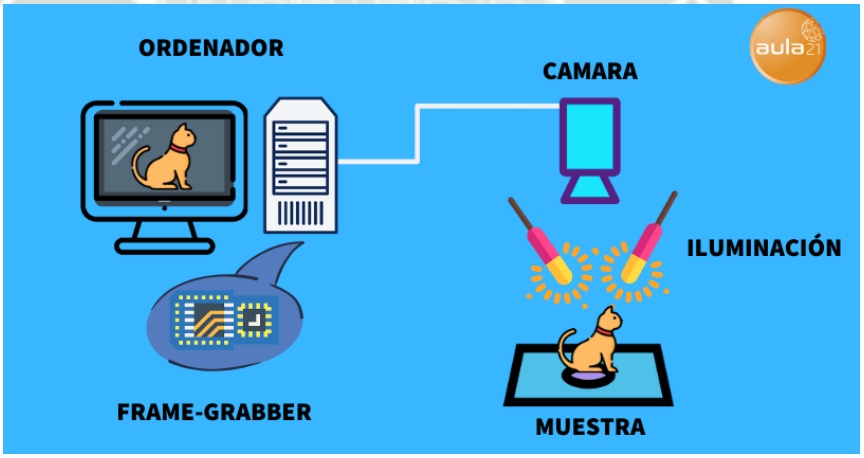
2.1.2. Visión artificial

La visión artificial proviene de un campo de la inteligencia artificial, el cual gracias al uso de redes neuronales y machine learning, logra entrenar a los sistemas para poder extraer información significativa de las imágenes, videos y otras entradas visuales, con esta información es capaz de dotar al sistema de la información necesaria para la toma de acción, ya sea por defectos o fallas en el proceso.

Realizando una analogía se podría tomar en cuenta que la IA, le permite al ordenador pensar, mientras que la visión artificial le permite al ordenador ver y comprender.

Figura 2

Diagrama básico de la visión artificial.



Nota. Tomado de centro de formación técnica para la industria aula 21, 2015

El sistema de visión artificial consta de componentes básicos para poder realizar la adquisición de imágenes:

Cámara

Su labor es la de capturar la imagen o el video que se desea procesar.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Centro de iluminación

Es el encargado de otorgarle a la cámara el ambiente necesario para reducir los errores de imagen, como lo son la sombra y el ruido, y otorgarle la posibilidad de detectar de manera eficiente los defectos que se presenten en el objeto.

Software de análisis

Es el encargado de dotar al sistema de la capacidad de calculó y recepción de las imágenes, además de darle al sistema las instrucciones posteriores a la captura de la imagen.

Como cualquier sistema la visión artificial posee desventajas como lo son la falla en la captura, o la falla en el procesamiento de imágenes, por eso es recomendable que el sistema sea monitoreado por un especialista (Centro de formación técnica para la industria, s/f).

2.1.3. Aplicación de visión artificial

La visión artificial en la actualidad esta aplicada en los siguientes campos.

1. **Detección de defectos**

   En esta aplicación de la visión artificial se centra en hallar los defectos presentados en los objetos a analizar, estos pueden ser grietas en los metales, inicio de corrosión, defectos de pintura, putrefacción en frutas.

Figura 3

Detección de defectos en fruta por pigmentación.



Nota. Tomado de ATRIA, 2019

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

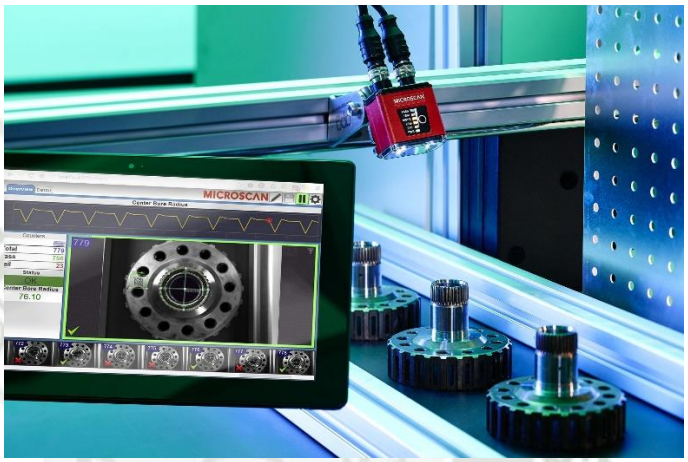
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

2. Metrología

Esta aplicación se centra en la capacidad de poder medir de forma eficiente un área de objeto, y de allí poder definir el tamaño de piezas, su rectitud o paralelismo, se requiere un ajuste de referencia óptimo.

Figura 4

Proceso de metrología.



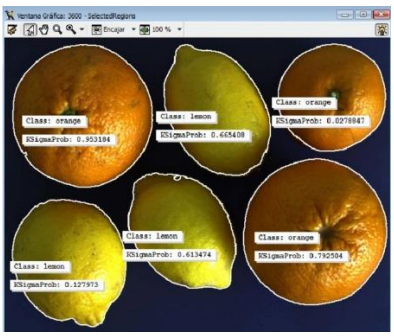
Nota. Tomado de Calvek, 2019

3. Detección de intrusos

Esta aplicación determina si en un grupo de control existe algún elemento que no pertenezca a este, es decir que, si se trata de una planta de acopio de fruta, en caso de existir la presencia de una piedra, el sistema sería capaz de detectarlo y proceder a una acción de rectificación.

Figura 5

Diferencia y clasificación de diferentes frutos.



Nota. Tomado de Salvador Giró, 2017

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

4. Verificación de montaje

En esta rama, se encarga de comprobar el correcto ensamble de las piezas en un proceso, también puede detectar la correcta aplicación de una etiqueta, de encontrar alguna variación de los estándares programados, se procedería a eliminar la pieza.

Figura 6

Proceso de verificación de montaje



Nota. Tomado de Monga Amit, 2019

5. Lector de caracteres

Esta aplicación es utilizada cuando a simple vista no es posible obtener información de un objeto, ya sea porque a nuestros ojos por el desgaste sea complicada la lectura, esta aplicación hace uso de filtros y patrones de reconocimiento para poder detectar los caracteres existentes.

Figura 7

Proceso de lectura de códigos



Nota. Tomado de Bcn Vision, 2020

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

6. **Bin picking**

   Se trata de una aplicación orientada a robótica en la cual gracias a la visión artificial el robot es capaz de hacer el recojo de piezas y colocarlas de manera que puedan seguir el proceso sin quedar atascadas, o bien para que los procesos de reconocimientos siguientes tengan el ángulo requerido para su análisis.

Figura 8

Aplicación Bin Picking con brazo robótico



Nota. Tomado de Blumenbecker, 2016

Cabe resaltar que estas aplicaciones son una de muchas otras existentes, debido a que la visión artificial es flexible a cualquier requerimiento en industria (ATRIA, s/f).

2.1.4. Procesamiento de imágenes visión artificial

El procesamiento de imágenes con visión artificial hace referencia al uso de técnicas y algoritmos computacionales que se utilizan para analizar, interpretar y comprender las imágenes digitales o videos, esto con el fin de lograr extraer la información útil y posteriormente ser utilizadas para poder automatizar procesos que normalmente requieren de la intervención humana. En el procesamiento de imágenes por visión artificial se combina la visión por computadora, la inteligencia artificial y el procesamiento digital de imágenes.

Para lograr el procesamiento de imágenes por visión artificial se utilizan los siguientes aspectos:

1. **Captura y adquisición de imágenes**

   Se trata del primer paso necesario para lograr el procesamiento de imágenes y necesita de cámaras, escáneres entre otros, para lograr la adquisición de datos que posteriormente serán procesados.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

2. **Procesamiento**

   En este paso se realiza la mejora de la calidad de los datos adquiridos (imágenes) realizando la mejora del contraste, la eliminación del ruido, la normalización de colores, entre otros procesos.

   El objetivo que realiza el procesamiento es dar el formato adecuado a cada imagen para su análisis.

3. **Segmentación**

   En esta etapa, se divide la imagen y se logran identificar los objetos dentro de ella. Los algoritmos son los encargados de diferenciar las áreas segmentadas de la imagen para facilitar el análisis.

4. **Extracción de características**

   Esta etapa ocurre posterior al proceso de segmentado, aquí se realiza la extracción de las características importantes, como bordes, formas, texturas y patrones, los cuales son necesarios para permitir la comprensión de la imagen.

5. **Reconocimiento y clasificación**

   Una vez realizada la extracción de características, se procede en esta etapa al uso de técnicas de aprendizajes, redes neuronales entre otras técnicas relacionadas a la inteligencia artificial, el conjunto de aprendizaje y características extraídas, nos permiten el reconocimiento de los objetos, patrones y la toma de decisiones según la aplicación (Gonzalez & Woods, 2018).

El procesamiento de imágenes por computadora da un amplio espectro de uso, debido a que la utilidad es dada según los requerimientos y codificación que se le pueda otorgar al sistema (D. Forsyth & J. Poince, 2003).

2.1.5. Elementos de los sistemas de visión artificial

Un sistema de visión artificial tiene la finalidad de emular la capacidad de la visión humana, básicamente le permite a la maquina ver y procesar las imágenes que capta para analizarlas y lograr tomar decisiones en un entorno automático.

Los principales elementos de los sistemas de visión artificial son los siguientes:

1. **Dispositivos de captura de imágenes**

   Es el componente inicial necesario para cualquier tipo de sistema artificial, se encarga de la toma de imágenes o videos.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Los dispositivos utilizados van desde cámaras tradicionales, cámaras infrarrojas, sensores 3D, cámaras estereoscópicas, entre otros tipos.

- Cámaras: se utilizan cuando se desean capturar imágenes o videos.
- Sensores: son utilizados cuando se necesitan tomar datos que van más allá de la luz visible, unos ejemplos de estos son las imágenes térmicas y los espectros electromagnéticos (Davies, 2012).

2. **Iluminación**

La iluminación es crucial cuando se trata de la calidad de la captura, gracias a la iluminación se consigue un mejor contraste, brillo y claridad en la toma.

Existen varias técnicas de iluminación:

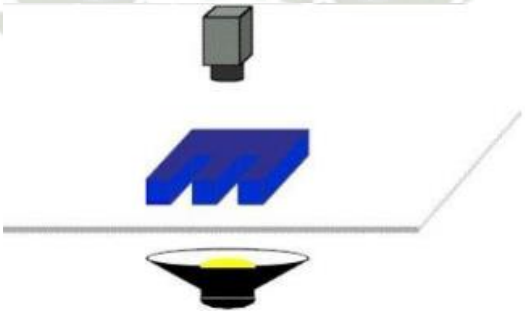
- Retroalimentación difusa:

  Esta técnica es utilizada cuando solo sea necesario resaltar el contorno de la pieza para su reconocimiento o toma de medidas, se requiere que la pieza este apoyada sobre una superficie transparente.

  Para lograr su ejecución se debe iluminar contra la cámara, situando el objeto entre la cámara y la lámpara, lo que permite un contraste de fondo y pieza elevado, esto permite que mediante la técnica de binarización se logre segmentar la imagen, se tiene que considerar que en esta técnica se pierden los detalles de la superficie del objeto (Medina Fernández, 2019).

Figura 9

Iluminación retroactiva difusa



Nota. Tomado de Medina Fernández, 2019

- **Iluminación frontal**

  Esta técnica trata de iluminar la pieza de forma frontal. Presenta el inconveniente de presentar brillos y sombras que afectan el contraste existente entre la pieza y el fondo, es utilizada en piezas poco reflectoras para reducir los brillos producidos (Medina Fernández, 2019).

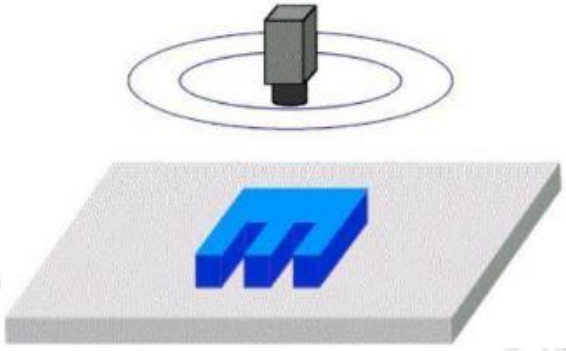
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 10

Iluminación frontal



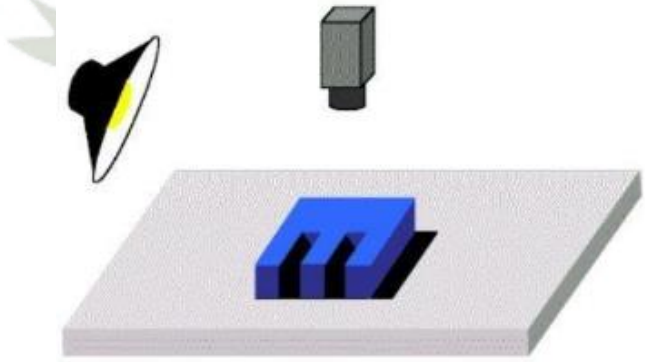
Nota. Tomado de Medina Fernández, 2019

- **Luz direccional**

  Se utiliza cuando se requiere captar una característica en específico del objeto, esto permite la creación de sombras, lo cual aumenta el contraste de partes tridimensionales y aumenta la información 3D del objeto (Medina Fernández, 2019).

Figura 11

Luz direccional.



*Nota***.** Tomado de Medina Fernández, 2019

- **Luz estructurada**

  Esta técnica de iluminación permite la proyección de patrones de luz modulados sobre la pieza, el reflejo de la luz también es modulada, logrando

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

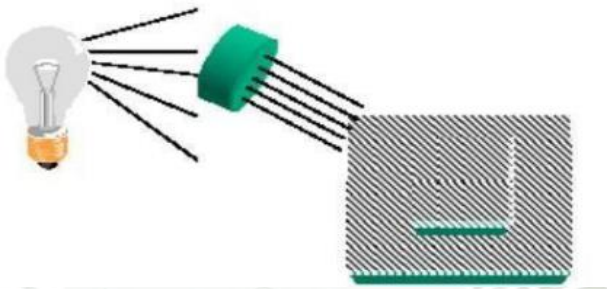
REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

obtener la información de la estructura de la superficie del objeto, que mediante la técnica de triangulación puede ser reconstruida, normalmente se utiliza iluminación laser para la aplicación de esta técnica debido a que un requerimiento es que la luz sea estructurada, es mayormente utilizada para aplicaciones de reconstrucción 3D de objetos y para conocer la forma de este (Medina Fernández, 2019).

Figura 12

Luz Estructurada.



*Nota***.** Tomado de Medina Fernández, 2019

Existen diferentes fuentes de iluminación:

- **Lámparas incandescentes**

  Es la fuente de iluminación más utilizada, consta de un filamento de tungsteno o halógeno-tungsteno. Se le encuentran con gran variedad de potencias sin embargo no mantienen la luminosidad con el tiempo.

Figura 13

Lampara incandescente.



Nota. Tomado de Asociación de Ciencias Ambientales, 2016

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

- **Tubos fluorescentes**

  Especialmente utilizadas con objetos con un índice de refracción elevado, debido a que ofrecen una luz difusa.

Figura 14

Lampara fluorescente.



Nota. Tomado de Carlos Alcaraz, 2018

- **Fibra óptica**

  Utilizada para la iluminación cuando el acceso es pequeño, sin embargo, provee una iluminación constante.

Figura 15

Iluminación por fibra óptica.



Nota. Tomado de FiberMAX, 2018

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

- **Laser**

  Se utilizan para una iluminación con luz estructurada, sin embargo, no son útiles cuando la superficie a iluminar absorbe luz.

Figura 16

Iluminación laser.



Nota. Tomado de Shutterstock, 2024

- **Leds**

  Los diodos emisores de luz (LED) son un tipo de iluminación de bajo consumo energético logrando consumir un 50-55% menos que la iluminación tradicional, es muy versátil en cuanto términos de potencias sin embargo son susceptibles a subidas de tensión, debido a que aumenta el diodo su temperatura y puede llegar a descomponerse (Medina Fernández, 2019).

Figura 17

Iluminación led.



*Nota***.** Tomado de i-Lumina, 2017

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

3. **Lentes**

   Los lentes son necesarios cuando se trata del control y el enfoque de la imagen que llega al sensor. Estos se encargan de ajustar el campo de visión, la profundidad de campo, el enfoque y distorsión.

   Existen lentes fijos y variables que son usados para controlar el detalle y la precisión con la que se capturan las imágenes, del mismo modo existen filtros ópticos que son capaces de otorgar un bloqueo de ciertas longitudes de onda o permitir el paso de la luz polarizada para mejorar la calidad de la toma (Trucco & Verri, 1998).

4. **Unidad de procesamiento**

   La unidad de procesamiento es la encargada de procesar la imagen capturada, en esta etapa se utilizan los algoritmos para la mejora el filtrado y análisis de la imagen.

   Para lograr un correcto procesamiento digital se realizan las siguientes tareas

   - Procesamiento de imagen
   - Segmentación
   - Extracción de características (Gonzalez & Woods, 2018).

5. **Software de visión artificial**

   Aquí se tienen en cuenta los siguientes apartados:

   - Algoritmos de visión: reconocimiento de patrones, clasificación de los objetos y la detección de bordes.
   - Redes neuronales y aprendizaje profundo: son utilizadas para tareas complejas de reconocimiento de imágenes.

   El software se encarga de proporcionar la plataforma de procesamiento y análisis de imágenes, utiliza todos los apartados anteriormente expuestos, para la interpretación de imágenes y la toma de decisiones (Szeliski, 2022).

6. **Interfaz y comunicación**

   Un sistema de visión artificial necesita una forma de interactuar con los demás sistemas, puede ser la comunicación con robots industriales, sistemas de control de calidad o dispositivos de alerta y es necesario para la toma de decisiones en tiempo real.

   Para ello tenemos que destacar:

   - Protocolos de comunicación: Ethernet, wi-fi, USB, etc.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

- Interfaz gráfica de usuario (GUI): Necesario para permitir la interacción humana en labores de configuración y supervisión del sistema (Sonka et al., 2007).

7. **Controlador**

   Un controlador es el cerebro del sistema, está encargado de la ejecución de los algoritmos y la gestión de la comunicación entre los componentes del sistema.

   Esta tarea puede ser realizada por los siguientes elementos:

   - Controladores embebidos

     Son sistemas diseñados para realizar ciertas funciones específicas con eficiencia y autonomía.

   - Unidades de procesamiento gráfico (GPU)

     Son utilizadas cuando se requieren procesamientos de imágenes de alta demanda (Bradski & Kaehler, 2008).

8. **Actuadores**

   Son los encargados de ejecutar las ordenes de control basados en la visión artificial, su acción es realizada en tiempo real, estos pueden ser brazos robóticos o pistones, servomotores, motores paso a paso entre otros (Siciliano et al., 2009).

2.1.6. Etapas en el proceso de visión artificial

Para comenzar con el proceso de la visión artificial, se comenzará por la adquisición de la imagen digital, para ello se hace uso de sensores que tengan la capacidad para digitalizar la señal que este logra captar, posterior a esto comienza la etapa de preprocesamiento de dicha imagen, aquí se mejora la imagen captada, con el objetivo de tener una mayor tasa de éxito en el proceso final. Una vez se preprocesa la imagen se inicia la etapa de la segmentación, su función es la de dividir la imagen en las partes o los objetos que la forman, es necesario que estos datos se conviertan de tal forma que sean apropiadas para el ordenar, para ello tenemos dos modos, utilizar la representación por frontera, la cual es apropiada cuando el objetivo tiene que ser analizado basado en sus características externas, ya sea esquinas, concavidades o convexidades y otro modo es la representación por región completa, la cual logra obtener del objetivo propiedades internas, como la textura o el esqueleto, sin embargo es posible el uso de ambas formas dependiendo del requerimiento.

Finalmente se procede al reconocimiento que es el proceso que logra asignar una etiqueta a un objeto basado en la información que proporcionan los descriptores para

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

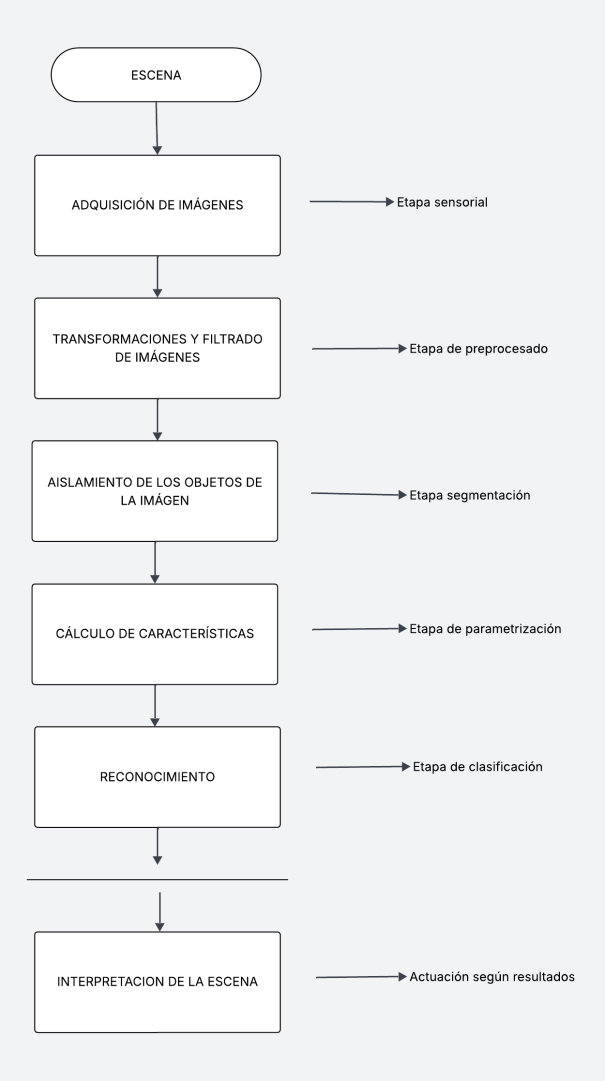
REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

después por pasar a la etapa de la interpretación la cual asigna un significado al conjunto de objetos que se reconocieron (Medina Fernández, 2019).

Estos pasos se muestran en la siguiente figura:

Figura 18

Diagrama de bloques de las etapas de un sistema de Visión Artificial.



Nota. Tomado de Medina Fernández, 2019.

2.1.7. Adquisición de imagen

Esta es la etapa inicial conocida como etapa sensorial, mediante la cual un dispositivo de entrada, como una cámara o un sensor óptico hace la captura inicial del objeto, con el fin de convertir la información del mundo físico en una representación digital.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Se debe tener en cuenta que en esta etapa se trata de conseguir la imagen más adecuada posible para los procesos posteriores, por lo tanto, se deben tomar en cuenta factores como el hardware encargado de la captura, el entorno, el posicionamiento del objeto, la iluminación, el fondo, el ruido eléctrico óptico y el posicionamiento de la cámara (Gonzalez & Woods, 2018).

2.1.8. Preprocesamiento de imagen

En esta etapa ingresa la imagen capturada pero sin ningún tratamiento, cuando la imagen es ingresada al computador esta se convierte en formato digital la cual se interpreta en forma de dígitos, cada uno de estos dígitos se convierten en pequeños puntos llamados pixeles, donde 0 es negro y 255 es blanco, cada pixel en color se representa por las siglas RGB las cuales hacen alusión a los colores rojo, verde y azul, los cuales también toman valores de 0 a 255 y esto dependerá de la intensidad del color representado.

En esta etapa se hace el tratado de la imagen la cual puede contener ruido, variaciones de iluminación entre otras imperfecciones, unas técnicas comunes en esta etapa pueden ser:

- **Filtrado**

  Se busca la eliminación del ruido por medio de filtros gaussiano, mediano o promedio.
- **Normalización**

  Se busca ajustar el brillo y el contraste de la imagen.
- **Transformaciones geométricas**

  Trabaja el escalado, alineación o rotación de la imagen (Gonzalez & Woods, 2018).

2.1.9. Filtros para mejora de imagen

Los filtros son métodos que permiten resaltar o suprimir información contenida en una imagen a diferentes escalas espaciales, esto se realiza de forma selectiva y su objetivo es resaltar características u ocultar valores anómalos, que se capturan en la imagen, los filtros más utilizados son los de paso bajo que se encargan de suavizar la imagen, los de paso alto, que su función es la de aumentar el contraste, los filtros direccionales, que se encarga de detectar estructuras en la imagen que siguen una dirección determinada y los de detección de bordes que son utilizados para identificar y asilar objetos con propiedades homogéneas

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Filtros de paso bajo

Son utilizados con el fin de suavizar la imagen, su utilidad radica cuando la imagen a procesar cuenta con mucho ruido y este se necesita eliminar, también son utilizados cuando se necesita resaltar información a una determinada escala.

- **Filtro de la media**

  Se utiliza para la reducción de ruido de la imagen, para ello reemplaza cada pixel de la imagen con la media aritmética (promedio) de los valores de intensidad de los pixeles en un vecindario definido por una ventana de tamaño fijo.

  La ventana o Kernel puede ser de diferentes tamaños ya sea 3x3 o 5x5 entre otros, esta determina el área del vecindario alrededor de cada pixel, mientras mayor sea la ventana, mayor será el suavizado que se lograra, pero se perderá detalle.

  Ventajas:

  1. Su implementación es sencilla y computacionalmente eficiente
  2. Suaviza los ruidos de alta frecuencia.

  Desventajas

  1. Puede llegar a difuminar los bordes y detalles importantes ya que no es capaz de diferenciar entre ruidos y características y trata a ambas de la misma manera.
  2. No funciona bien cuando se trata de ruido impulsivo, como por ejemplo los puntos negros y blancos que puedan aparecer en la captura, para ello es mejor el uso de filtro mediano (Gonzalez & Woods, 2018).

- **Filtro de media ponderada**

  En este filtro los elementos de la matriz de filtrado no son todos, en vez de eso este filtro asigna más peso a uno de ellos, que es generalmente el central, con el fin de lograr un resultado cercano a la imagen original y evitar que esta aparezca borrosa. Este filtro utiliza el uso de pesos para llevar un control más refinado sobre el efecto de suavizado.

  Ventajas

  1. Mayor precisión en el efecto de suavizado
  2. Logra preservar los bordes en la imagen
  3. Logra tener flexibilidad al poder ajustar el peso de los Kernel.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Desventajas

1. Mayor complejidad para su implementación
2. Poca efectividad ante ruidos impulsivos (Gonzalez & Woods, 2018).

- **Filtro de la mediana**

A diferencia de los filtros de media ponderada y el filtro de la media, este filtro es capaz de ser particularmente efectivo ante el ruido impulsivo, sin difuminar los bordes.

Para su funcionamiento este filtro selecciona una ventana cuadrada de 3x3, 5x5 u otros modelos, que se deslizara pixel por pixel sobre toda la imagen, los valores de los pixeles en la ventana se ordenan de menor a mayor, la mediana será el valor central de los pixeles ordenados, además el pixel central de la ventana se reemplazara por este valor obtenido.

Ventajas

1. Logra la reducción de ruido impulsivo.
2. Logra preservar los bordes en la imagen.

Desventajas

1. Necesita mayor tiempo de cómputo cuando las imágenes son grandes, debido al tiempo requerido en el ordenamiento de los pixeles.
2. Es únicamente efectivo cuando el ruido es impulsivo, sin embargo, para otros tipos de ruido, pierde efectividad.
3. El tamaño de la ventana debe tener un tamaño adecuado debido a que, de ser muy pequeña, no se lograra filtrar de manera adecuado el ruido y si es demasiado grande puede llegar a distorsionar detalles finos (Gonzalez & Woods, 2018).

- **Filtros adaptativos**

Son filtros más complejos que los anteriores, debido a que los coeficientes de ponderación son recalculados para cada uno de los pixeles, este en función del histograma de los ND que aparecen en la ventana.

Sin embargo, dentro de sus características principales se encuentran las siguientes:

1. Son capaces de ajustar sus parámetros de filtrado según las características locales, como intensidad o presencia de bordes.
2. Son eficientes cuando la imagen presenta ruidos variables.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

3. Son capaces de preservar bordes de estructuras importantes mientras eliminan el ruido.

Las desventajas que presenta este tipo de filtro son las siguientes:

1. Este tipo de filtro requiere más cálculos debido a la adaptación dinámica, por lo cual es más complejo computacionalmente.
2. El rendimiento está directamente afectado por la configuración de parámetros como el tamaño de ventana o los modelos de ruido.
3. Su uso en tiempo real es crítico debido a que requieren tiempo de cómputo extendido, por lo cual en sistemas que tengan restricción de tiempo no podrán ser implementados (Gonzalez & Woods, 2018).

- **Filtros gaussianos**

Estos filtros simulan una distribución gaussiana bivariante, es decir que el valor máximo aparece en el píxel central y va disminuyéndose mientras se va recorriendo hacia los extremos, es un filtro ideal cuando se requiere un efecto de suavizado suave, debido a que aplica una convolución entre la imagen y una máscara basada en la función gaussiana.

Ventajas

1. Reduce el ruido gaussiano de manera efectiva
2. Tiene poco impacto en los bordes, por lo cual su proceso de suavizado es eficiente.
3. Utilizando mascaras predefinidas se logra una facilidad de implementación.

Desventajas

1. Es posible que difumine detalles finos.
2. Con ruido impulsivo no es eficiente (Gonzalez & Woods, 2018).

Filtros de paso alto

Este tipo de filtros tiene la tarea de resaltar zonas de mayor variabilidad eliminando la componente media, que son las que afectan los filtros de paso bajo, se considera en estos filtros que la respuesta de cada pixel está contaminada por la de pixeles vecinos, debido

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

a que se toma la radiación reflejada por cada pixel como una contaminación para los otros, estos filtros solucionan este problema.

- **Filtros direccionales**

  Son filtros utilizados cuando se necesita resaltar, detectar o analizar ciertas características en una imagen, pero en determinadas direcciones, estos filtros son fundamentales para la identificación de bordes, líneas, texturas o patrones. El funcionamiento de este filtro empieza por la identificación de las variaciones en los valores de intensidad de los pixeles en una dirección especifica, puede ser vertical, horizontal o diagonal, la operación de este filtro está asociada a una máscara o Kernel que actúa como una plantilla que calcula como cambian los valores en cierta dirección.

  Ventajas

  1. Logran detectar patrones en orientaciones especificas
  2. Logran resaltar bordes y texturas complejas

  Desventajas

  1. Requieren preprocesamiento debido a que son sensibles al ruido
  2. Los filtros para direcciones especificas suelen requerir intensidad de computo
  3. No siempre son capaces de diferenciar entre ruido y características reales de la imagen (Gonzalez & Woods, 2018).

- **Filtros para la detección de bordes**

  Los filtros de detección de bordes trabajan calculando el gradiente de intensidad de una imagen. Los principales métodos utilizan mascaras o Kernels que operan en la vecindad de cada pixel. Dentro de este tipo de filtros se tiene los siguientes:

  1. **Filtro Sobel**

     Utilizado en detección de bordes en direcciones específicas, pero solo en direcciones horizontales y verticales, este filtro utiliza mascaras convolucionales para calcular su gradiente de intensidad.

     Es un filtro simple y eficiente y logra resaltar bordes bien definidos, sin embargo, es sensible al ruido.

  2. **Filtro Prewitt**

     Es similar al filtro Sobel con la diferencia que las máscaras en este son más simples. Solo sirve para direcciones horizontales y verticales.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Es un filtro más rápido en su cálculo, pero menos preciso que el filtro Sobel.

3. **Filtro de Laplace**

   Este filtro logra la detección de bordes aplicando la segunda derivada, es útil cuando existen áreas donde hay un cambio rápido de intensidad, este filtro puede trabajar en cualquier dirección, y no es necesario para su uso el cálculo de gradientes separados, sin embargo, es muy sensible al ruido por lo cual para su uso es recomendable suavizar primero la imagen.

4. **Filtro Canny**

   Es un filtro avanzado que requiere de 4 pasos para su correcto uso:

   Primer paso:

   Suavizado de la imagen con un filtro gaussiano

   Segundo paso:

   Realizar cálculo del gradiente utilizando Sobel o Prewitt

   Tercer paso:

   Se aplica supresión no máxima para afinar los bordes

   Cuarto paso:

   Detección de bordes con umbrales (histéresis)

   Este filtro es muy robusto al ruido y es capaz de producir bordes precisos, sin embargo, requiere de mucho coste computacional.

5. **Filtro Roberts**

   Este filtro se encarga de la detección de bordes al calcular diferencias entre los pixeles adyacentes en direcciones diagonales, es simple y rápido en su respuesta sin embargo al utilizar imágenes de alta resolución no funciona bien debido a que tiene sensibilidad al ruido.

6. **Filtro de Kirsch**

   El filtro Kirsch detecta bordes en 8 direcciones específicas, es capaz de resaltar los bordes con orientación clara, pero debido a su versatilidad en la detección también produce un costo computacional elevado.

7. **Filtro de Gabor**

   Este filtro combina un filtro gaussiano con una onda sinusoidal, realizando así la detección de bordes y texturas en una dirección especifica, sin embargo, para su correcta implementación se necesita de parámetros cuidadosamente ajustados (Gonzalez & Woods, 2018).

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Tabla 1

Diferencias entre filtros de paso alto.

Filtro	Ventajas	Desventajas
Sobel	Simple y eficiente.	Sensibilidad al ruido.
Prewitt	Mas rápido que filtro Sobel.	Precisión reducida a comparación de filtro Sobel.
Laplace	Detección de bordes en todas las direcciones.	Alta sensibilidad al ruido requiere de preprocesamiento.
Canny	Preciso y robusto.	Costo computacional muy elevado.
Roberts	Simple y rápido.	Poco efectivo en bordes complejos y sensibilidad al ruido.
Kirsch	Detección de bordes en 8 direcciones.	Coste computacional elevado, no es muy utilizado.
Gabor	Detecta texturas y bordes direccionales.	Los parámetros deben ser ajustados de manera precisa para garantizar buenos resultados.

Nota. Elaboración propia.

2.1.10. Segmentación

La segmentación es la etapa en la cual se procede a la separación de una imagen en sus partes constituyentes, con el objetivo de obtener solo lo que nos interesa del resto de la imagen, el nivel de esta subdivisión estará definida por los requerimientos del problema, al hacer la separación de partes constituyentes de una imagen se logran identificar los bordes o se separan está en regiones, líneas o curvas.

El objetivo de la etapa de segmentación radica en la segmentación de objetos en fondo, la división de una imagen en regiones homogéneas y preparación de los datos para análisis posteriores como el reconocimiento o la medición de características. p

Para la etapa de segmentación existen técnicas las cuales se detallan a continuación.

1. **Segmentación basada en umbrales**

   Esta técnica consiste en separar regiones en función de un nivel de intensidad, existen tres tipos, umbral simple que divide la imagen en región de objetos y región de fondos, umbral adaptativo, que divide la imagen en regiones más pequeñas, ajustando el

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

umbral localmente y Otsu, que determina automáticamente el umbral óptimo para separar las regiones.

Esta técnica es simple y de rápida respuesta computacional, sin embargo, presenta sensibilidad al ruido y a las variaciones de iluminación.

2. **Segmentación basada en bordes**

   Esta técnica detecta los bordes que delimitan las regiones en la imagen esto mediante el uso de filtros, Sobel, Prewitt, Canny o de Laplace, es adecuada cuando las imágenes que se tienen cuentan con contrastes bien definidos, sin embargo, puede llegar a generar bordes incompletos o bordes múltiples para un mismo objeto.

3. **Segmentación basada en regiones**

   Agrupa pixeles conectados que tienen propiedades similares como intensidad o textura, empieza desde un pixel semilla y expande la región agregando pixeles vecinos con características similares. También divide recursivamente la imagen en regiones más pequeñas luego las combina si cumplen un criterio de similitud.

   Es capaz de generar regiones homogéneas además de ser eficaz con imágenes de características consistentes, sin embargo, requiere de criterios bien definidos para similitud.

4. **Segmentación por agrupamiento**

   Se utiliza algoritmos de agrupamiento para clasificar pixeles en diferentes grupos basándose en sus características.

5. **Segmentación basada en modelos**

   Utiliza modelos probabilísticos o geométricos para segmentar las imágenes pueden ser tres:

   - Modelos activos. Detectan los contornos moviéndose hacia los bordes de los objetos.
   - Método nivel-set. Representan los bordes como curvas que evolucionan iterativamente.
   - Segmentación bayesiana

   Usa modelos probabilísticos para clasificar pixeles.

   Es capaz de manejar formas complejas además de lidiar bien con objetos con bordes suaves, pero es computacionalmente intensivo.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

6. **Segmentación mediante redes neuronales**

   Utiliza redes neuronales profundas para el aprendizaje de características específicas en los objetos en la imagen, estas pueden ser

   - U-Net: muy utilizada en segmentación medica
   - Mask R-CNN: Identifica objetos y sus máscaras de segmentación

   Es altamente precisa y es capaz de aprender características complejas sin embargo requiere de gran cantidad de datos etiquetados y el costo computacional es alto (Gonzalez & Woods, 2018).

2.1.11. Extracción de características

La extracción de características es el proceso mediante el cual se logra identificar y seleccionar los elementos más importantes de una imagen, para su análisis o clasificación posterior. Su funcionamiento de forma abreviada trata de evitar el procesamiento de una imagen completa, debido a que esto supondría un gasto computacional elevado, en vez de eso se aplica la extracción de atributos clave, la cual representara la información más significativa (Gonzalez & Woods, 2018).

1. **Importancia de la extracción de características**

   Esta etapa es importante porque es capaz de generar 3 aspectos significativos para el beneficio de nuestra aplicación.

   - Genera reducción de datos al solo extraer las características esenciales de las imágenes
   - Mejora el rendimiento y facilita el trabajo a los algoritmos de clasificación y reconocimiento
   - Aumenta la precisión de detección, eliminando datos irrelevantes que pueden generar errores (Gonzalez & Woods, 2018).

2. **Tipos de características extraídas**

   Existen tres tipos de características extraídas y estas dependen del nivel de abstracción.

   2.1. Características de bajo nivel (Basadas en pixeles)

   Son propiedades básicas adquiridas directamente de los pixeles que se encuentran en la imagen, pueden ser propiedades como:

   - Color: Histograma de color, espacios de color (RGB, HSV, YCbCr)
   - Textura: son métodos como matrices de Coocurrencia de nivel de gris (GLCM) o transformada de Wavelet.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

- Forma: contornos, detección de bordes, entre otros.

2.2. Características de nivel medio (Estructurales)

Este tipo de características se generan gracias a la combinación de las características de bajo nivel. En este nivel se observa:

- Puntos clave: aquí se encuentra los SIFT (Scale-Invariant Feature Transform) SURF (Speed up Robust Features) ORB (Oriented Fast and Rotated Brief)
- Segmentación de objetos: aquí se encuentra los algoritmos como Watershed o K-means.

2.3. Características de alto nivel (Semánticas)

Para lograr obtener características de alto nivel es requerido el uso de redes neuronales y aprendizaje profundo, por ejemplo, para obtener las características profundas es necesario el uso de redes neuronales convolucionales (Gonzalez & Woods, 2018).

3. Métodos utilizados para la extracción de características

3.1 Detección de bordes

Se utilizan métodos como Sobel o Canny, los cuales son capaces de identificar zonas donde la intensidad de la imagen cambiar bruscamente, ayudando a encontrar contornos y formas.

3.2 Transformada de Fourier y Wavelet

Métodos en donde se separa la imagen en sus componentes de frecuencia, útiles en análisis de texturas.

3.3 Histograma de gradientes orientados

Representa la distribución de bordes y estructuras en la imagen, muy utilizado cuando la aplicación es para reconocimiento de personas.

3.4 Redes convolucionales

En lugar de tener que definir manualmente las características, una CNN aprende automáticamente cual es la mejor representación a partir de los datos de entrenamiento que posee (Gonzalez & Woods, 2018).

2.1.12. Interpretación

Este procesamiento es de alto nivel de reconocimiento, en él se le asigna una etiqueta a un objeto que será otorgada en base a la información que le brindaron sus descriptores, en esta etapa se les da un sentido y significado a los objetos reconocidos, para poder llegar a clasificarlos de acuerdo a sus propiedades. Una técnica que se utiliza de manera común

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

está basada en clasificadores estadísticos o modelos neuronales multicapa, mediante un algoritmo de retro propagación (Medina Fernández, 2019).

2.2.Redes Neuronales artificiales

2.2.1. Introducción a las redes neuronales artificiales

Las redes neuronales artificiales son un conjunto de algoritmos que basan su funcionamiento en el cerebro humano, están diseñados para poder detectar patrones, son fundamentales para el aprendizaje automático (machine learning) y dentro de la inteligencia artificial son mayormente usados para resolver problemas de reconocimiento de voz, imágenes y el procesamiento del lenguaje natural (Haykin, 2009).

2.2.2. Principios básicos de las redes neuronales artificiales

Para una red neuronal artificial, las neuronas se conocen como nodos, y estas se organizan en capas:

a) **Capa de entrada:**

Es la primera capa dentro de la red neuronal, su función es la de recibir datos que se necesitan procesar, cada neurona existente en esta capa representa una variable del conjunto de datos de entrada.

b) **Capas ocultas:**

Las capas ocultas están situadas entre la capa de entrada y la capa de salida, la razón de su nombre se debe a que los valores que tienen no son visibles en el exterior del sistema. Esta capa transforma los datos de entrada aplicando funciones de activación no lineales, esto permite que la red logre relaciones complejas entre las variables de entrada. Mientras más capas ocultas tiene una red, más complejas son las funciones que puede aprender.

c) **Capas de salidas**

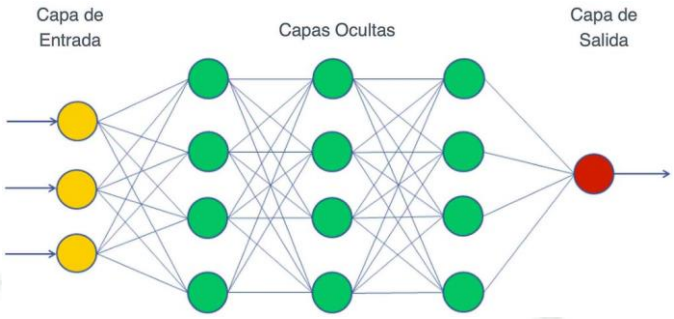
Es la última capa de la red neuronal, es donde se genera el resultado del proceso. El número de las neuronas existentes en la capa de salida está relacionado al tipo de problema que se intenta resolver. la capa de salida obtiene las representaciones de las capas ocultas y lo transforma en un formato que pueda ser interpretado (Haykin, 2009).

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 19

Estructura de una red neuronal



Nota. Tomado de Juárez Pacheco, 2023

Las RNA (Red Neuronal Artificial) poseen tres características importantes:

a) Autoorganización y Adaptabilidad:

   Las RNA emplean algoritmos de aprendizaje adaptativo y de autoorganización para lograr labores de procesado robusto.

b) Procesado no lineal:

   Otorga a la red mayor capacidad de clasificación de patrones, la aproximación de funciones y el aumento de su inmunidad frente al ruido.

c) Procesado paralelo:

   Se trata del uso a gran escala de nodos de procesado con un alto nivel de interconectividad.

Los nodos reciben una entrada desde la salida de otros nodos anteriores o del exterior, cada entrada que llega al nodo cuenta con un peso específico asociado "w", y este valor fluctuara en el proceso del aprendizaje. Cada nodo aplica una función f de la suma de las entradas ponderadas mediante los pesos. (Medina Fernández, 2019)

$$y_i = \sum_j w_{ij*y_j}$$

2.2.3. Tipos de redes neuronales artificiales

Existen diferentes tipos de RNA, para poder solucionar cada tipo de problema:

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

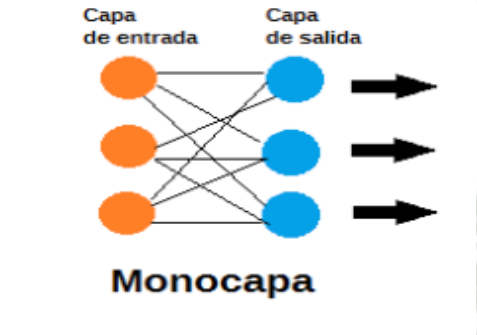
REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

1. Red Neuronal Monocapa

Se trata de la red neuronal más simple, esta consta de una única capa de neuronas. está orientado a la clasificación lineal y solo es capaz de aprender funciones simples, la desventaja del perceptrón yace en la falta de capacidad para resolver problemas no lineales (Bishop, 2006).

Figura 20

Red neuronal Monocapa



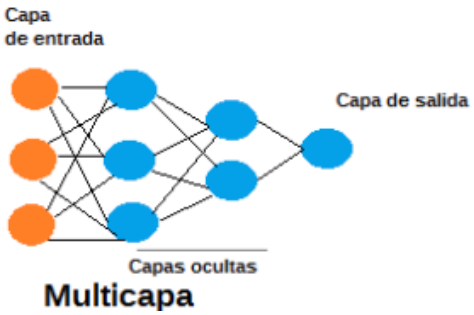
Nota. Tomado de Juárez Pacheco, 2023

2. Red neuronal multicapa (MLP)

Las redes neuronales multicapa constan de una capa de entrada, una o más capas ocultas y una capa de salida, por lo cual es capaz de afrontar retos con funciones no lineales, su aprendizaje se realiza mediante la retro propagación del error, un algoritmo que ajusta los pesos sinápticos mediante el cálculo de los gradientes (Bishop, 2006).

Figura 21

Red neuronal Multicapa



Nota. Tomado de Juárez Pacheco, 2023

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

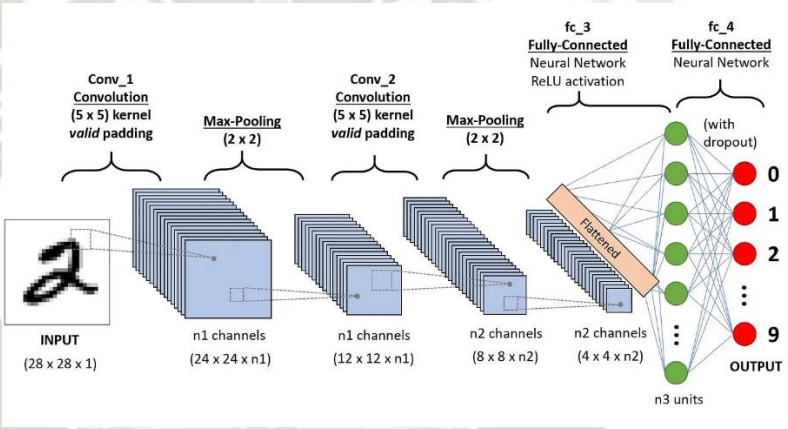
3. Redes neuronales convolucionales (CNN)

Las CNN (Convolutional Neural Networks) están específicamente diseñadas para el procesamiento de datos con estructura espacial, como lo son imágenes. Estas redes aplican filtros o convoluciones a las imágenes con el fin de extraer características relevantes ya sean bordes, texturas o formas.

Estas redes son necesarias cuando existen aplicaciones como visión artificial, reconocimiento de imágenes o procesamiento de videos (Bishop, 2006).

Figura 22

Red neuronal convolucional.



*Nota***.** Tomado de Sumana, 2022

4. Redes neuronales recurrentes (RNN)

Las RNN (Recurrent Neural Networks) son redes aplicadas para realizar datos secuenciales como series temporales o procesamiento de lenguaje natural.

En este tipo de red las conexiones entre las neuronas forman bucles, gracias a esto la red posee memoria y es capaz de utilizar la información almacenada para influir en las predicciones posteriores (Bishop, 2006).

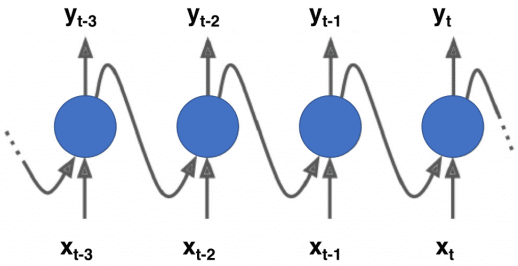
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 23

Red neuronal recurrente.



*Nota***.** Tomado de Jordi Torres, 2019

2.2.4. Algoritmos de aprendizajes en redes neuronales.

Los algoritmos de aprendizaje en las redes neuronales son métodos que le permiten a las redes detectar y aprender patrones y gracias a ello poder realizar predicciones a partir de los datos los principales algoritmos de aprendizaje son los siguientes:

1. **Retropropagación (Backpropagation)**

   Es el algoritmo más común para el entrenamiento de redes neuronales, basa su funcionamiento en la regla de descenso de gradiente.

   Funcionamiento:

   Para calcular la salida de la red se realiza una propagación hacia adelante, posteriormente se calcula el error comparando la salida predicha con la salida real, posteriormente el error se propaga hacia atrás a través de la red para actualizar los pesos mediante el cálculo del gradiente (Goodfellow et al., 2017).

2. **Algoritmos de optimización**

   En los algoritmos de optimización encontramos:

2.1. Descenso de gradiente:

Se encarga de actualizar los pesos en la dirección opuesta al gradiente de error con la finalidad de reducir la función de error.

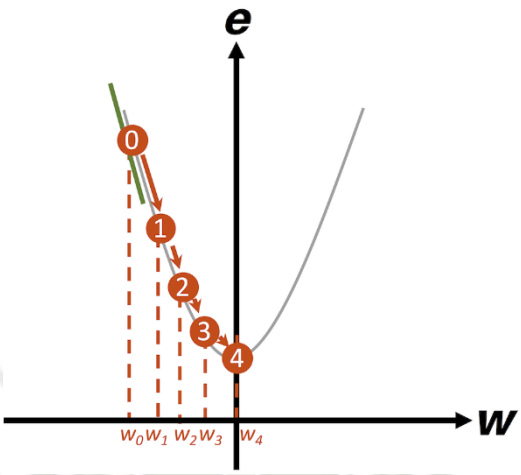
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 24

Proceso de descenso de gradiente.



Nota. Tomado de Codificando Bits, 2018

Como se puede observar en la imagen 16 el algoritmo de optimización de descenso de gradiente se cuenta con dos ejes, el eje vertical "e" que representa al error y el eje horizontal "w" que representa los pesos.

El proceso comienza en el punto 0 o W0 y cuenta con un total de 4 iteraciones, que se realizan en pasos discretos es decir que se mueve del paso 0 a 1 luego de 1 a 2 y así hasta terminar el número de iteraciones necesarias, en este punto (W0) se observa que el error es grande, sin embargo a medida que el proceso avanza, con cada iteración se va disminuyendo el error, hasta llegar al punto W4 en el cual el error es mínimo, en este punto se detiene el algoritmo puesto que la gradiente es casi cero y el peso alcanzo un valor óptimo para el modelo analizado.

Esta gradiente presenta ciertas variantes:

2.1.1. Gradiente descendente por batch

En la gradiente descendente por batch, se utiliza todo el conjunto de datos disponibles de entrenamiento para el cálculo de la gradiente y la actualización de pesos en cada iteración. Es decir que para la modificación de un peso se utilizan todos los ejemplos de entrenamiento.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

2.1.1.1. Ventajas

- Es capaz de garantizar una dirección estable hacia el error mínimo, debido a que considera el conjunto completo de datos de entrenamiento.
- No requiere muchas iteraciones debido a que el cálculo de la gradiente es más preciso.

2.1.1.2. Desventajas

- Requiere mucha memoria y tiempo cuando se cuenta con un numero de datos de entrenamiento grande.
- No es óptimo cuando se requiere de una respuesta rápida (Goodfellow et al., 2017).

2.1.2. Gradiente descendente estocástico

En la gradiente descendente estocástica se utiliza en cada iteración un ejemplo de entrenamiento aleatorio para calcular la gradiente y la actualización de los pesos, por lo cual no se espera al procesamiento de todos los ejemplos para ajustar el peso, si no que se ajusta el peso después del análisis de cada ejemplo.

2.1.2.1. Ventajas

- Es rápido en cada iteración debido a que solo necesita de un ejemplo para el cálculo de la gradiente.

2.1.2.2. Desventajas

- En ciertos casos llega a ser inestable y fluctuar alrededor del mínimo en lugar de converger de manera suave, debido a que cada actualización es ruidosa.
- El número de iteraciones es mayor para alcanzar la convergencia (Goodfellow et al., 2017).

2.1.3. Gradiente descendente por mini batch

La gradiente descendente por mini batch es un método que en lugar de tomar todos los datos de entrenamiento como en el caso de gradiente descendente por batch o utilizar cada dato de entrenamiento de forma individual para cada iteración como el caso de gradiente descendente estocástica, este método utiliza un grupo de datos de entrenamiento (mini batch), se debe tener en cuenta que cada tamaño de mini

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

batch debe ser una pequeña fracción del conjunto de datos, los valores pueden ser 32, 64 o 128 ejemplos.

2.1.3.1. Ventajas

- Buen balance entre estabilidad y velocidad de convergencia.
- Requiere menos memoria y tiempo que el método batch.
- Utilizada en redes neuronales que requiera aprendizaje profundo.

2.1.3.2. Desventajas

- Se requiere una selección correcta del tamaño del mini batch, debido a que el tamaño repercute directamente en la convergencia (Goodfellow et al., 2017).

Tabla 2

Tabla de diferenciación entre gradientes Batch, Estocástico y Mini Batch

Método	Datos de entrenamiento	Velocidad de convergencia	Requerimientos de memoria	Estabilidad
Batch	Se utiliza todos los datos de entrenamiento para cada iteración.	Lenta	Alta	Alta
Estocástico	Se utiliza un único dato aleatorio para cada dato de iteración	Alta	Baja	Baja
Mini Batch	Se utilizan grupos de datos de entrenamiento de 32, 64 o 128 datos.	Media	Media	Media

Nota. Elaboración propia.

2.2. Regularización

Se utilizan para evitar el sobreajuste, el sobreajuste ocurre en una red neuronal y se debe a que el modelo se adapta demasiado a los datos de entrenamiento, debido a esto se pierde la capacidad de generalización para datos nuevos (Goodfellow et al., 2017).

Los métodos de regularización más comunes son los siguientes:

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

2.2.1. Regularización L1 y L2 (Regularización de pesos)

Estos métodos agregan un término de penalización a la función de costo para controlar los valores de los pesos de la red.

Regularización L1 (Lasso)

Basa su funcionamiento en la suma de los valores absolutos de los pesos.

Este método tiende a reducir algunos pesos a cero con lo cual ayuda a la simplificación del modelo.

$$J(w) = Error + \lambda \sum_i |w_i|$$

$donde$:

$\boldsymbol{J(w)}$: Es la función de pérdida del modelo.

$\boldsymbol{Error}$: Es el error que posee el modelo sin regularización

$\boldsymbol{\lambda}$: Es el coeficiente de penalización, cuando es mayor que el error, se otorga mas peso a la regularización, mientras que, si es menor, el modelo se enfocara en minimizar el error.

$\sum_i |\boldsymbol{w_i}|$: Es la suma de las magnitudes absolutas de los pesos, se le conoce como regularización L1 o penalización Lasso. Incentiva al modelo a tener pesos pequeños o que algunos de ellos sean cero. (Goodfellow et al., 2017)

Regularización L2 (Ridge o Tikhonov)

Este método de regularización agrega una penalización que se basa en la suma de los cuadrados de los pesos, con lo cual logra que los pesos grandes se reduzcan y con ello permitir que el modelo sea menos complejo.

$$J(w) = Error + \frac{\lambda}{2} \sum_i {w_i}^2$$

$donde$:

$\boldsymbol{J(w)}$: Es la función de pérdida del modelo.

$\boldsymbol{Error}$: Es el error que posee el modelo sin regularización

$\frac{\boldsymbol{\lambda}}{\boldsymbol{2}}$: Es el coeficiente de penalización, cuando es mayor que el error, se otorga más peso a la regularización, mientras que, si es menor, el modelo se enfocara en minimizar el error.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

$\sum_i w_i^2$: Es la suma de los cuadrados de los pesos, conocido como regularización L2 o Penalización Ridge, esta regularización penaliza los pesos grandes, y logra que el modelo prefiera valores pequeños en los pesos, pero no los fuerza a ser cero (Goodfellow et al., 2017).

2.3. **Dropout**

La técnica Dropout se encarga de desactivar neuronas aleatoriamente en cada iteración, es decir que en cada paso del entrenamiento algunas neuronas no contribuirán a la predicción ni retro propagación de errores.

En cada iteración cada neurona posee una probabilidad "p" de ser desactivada. Durante el proceso de prueba no se aplica Dropout, pero los pesos se escalan para poder compensar la pérdida de neuronas durante el entrenamiento.

Dropout se utiliza para lograr la independencia de las neuronas debido a que estas a veces tienden a depender mucho unas de otras, esto otorga robustez en la representación de características (Goodfellow et al., 2017).

2.4. **Aumento de datos**

Esta técnica no modifica el modelo, sin embargo, se le considera una técnica debido a que al aumentar el número de datos de entrenamiento, se generan ejemplos nuevos a partir de los datos preexistentes.

Esta técnica ayuda al modelo a que realice una mejor generalización al lograr ver más variaciones de datos (Goodfellow et al., 2017).

2.5. Normalización de Batch

Esta técnica se encarga de normalizar las activaciones en cada capa para que tengan una media y varianza específicas durante el entrenamiento. Esto estabiliza y acelera el entrenamiento al mantener las activaciones controladas.

Para lograr esto, en cada minibatch las activaciones de una capa se normalizan y luego se reescalan usando parámetros de escala y desplazamientos aprendidos, esto permite al modelo ajustar los valores de activación en cada paso del entrenamiento.

La normalización batch reduce la sensibilidad de la red a los valores iniciales y le permite tener tasas de aprendizajes altas, debido a esto es capaz de mejorar la velocidad de convergencia (Goodfellow et al., 2017).

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

2.6. Parada Temprana (Early Stopping)

Esta técnica de regularización consta en detener el entrenamiento cuando el error en el conjunto de validación comienza a aumentar sin importar si el error en el conjunto de entrenamiento siga disminuyendo. De otro modo esto significaría que el modelo está comenzando a sobre ajustarse a los datos de entrenamiento.

Para lograr esta técnica se debe evaluar el error en un conjunto de validación después de cada época en el entrenamiento, cuando el error de validación deja de mejorar en cierto número de épocas, el entrenamiento se detiene (Goodfellow et al., 2017).

Tabla 3

Diferencias entre las técnicas de regularización.

Técnica	Utilidad	Ventaja	Desventaja
Regularización L1 y L2	Penalizan los pesos grandes en la función de costo.	Otorgan simplicidad y eficiencia al modelo.	El parámetro de regularización debe ser ajustado.
Dropout	Desactiva neuronas aleatoriamente en cada proceso de iteración en la etapa de entrenamiento.	Otorga eficiencia en redes neuronales profundas.	Introduce aleatoriedad al modelo.
Aumento de datos	Aumenta el conjunto de datos mediante transformaciones.	Mejora la generalización.	Se requiere transformación de datos.
Normalización batch	Normaliza las activaciones en cada una de las capas para estabilizar el aprendizaje.	Acelera la convergencia.	Vuelve más compleja la red.
Parada temprana	Detiene el entrenamiento cuando el error de validación deja de mejorar.	Evita el sobreajuste.	Requiere conjunto de validación.

Nota. Elaboración propia.

3. Redes neuronales convolucionales

Las redes neuronales convolucionales están diseñadas específicamente para el procesamiento de datos que posean una estructura cuadriculada, como las imágenes. Su uso es amplio cuando se trata del campo de la visión artificial por

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

computadora, debido a que son capaces de reconocer patrones espaciales y locales en los datos.

3.1 Arquitectura Básica de una red neuronal convolucional.

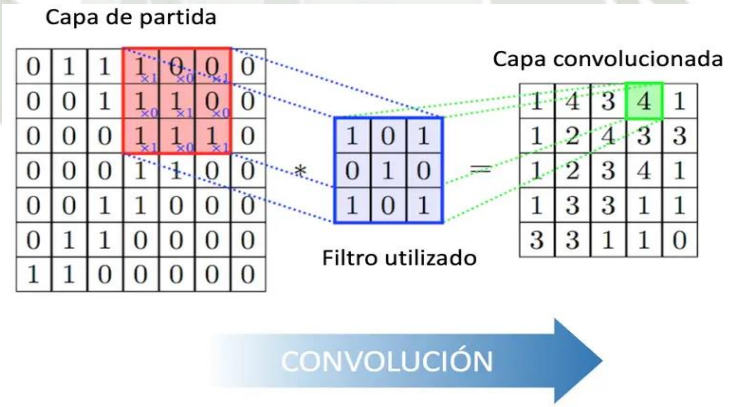
La arquitectura de la red neuronal convolucional normalmente incluye varias capas organizadas en un flujo secuencial, las capas principales son las siguientes:

- Capas Convolucionales

Las capas convolucionales son el núcleo de las redes neuronales convolucionales, estas aplican filtros (Kernels) en la capa de entrada para poder extraer características locales de los datos, las capas convolucionales otorgan la posibilidad de detección de patrones ya sean bordes y texturas en las primeras capas y características más complejas cuando se encuentra en las capas profundas.

Figura 25

Capa convolucional.



Nota. Tomado de Diego Calvo, 2019

En la figura 24 se observa el proceso inicial para crear una capa de convolución, en la capa de partida se puede observar los datos de entrada originales que están representando la imagen en una escala de grises, donde cada celda contiene un valor de intensidad, a esta imagen se le aplica un filtro (Kernel) el cual recorrerá toda la capa de partida con un paso el cual se llama stride, y será el encargado de extraer las características de esta, este filtro contiene valores aprendidos predefinidos como 1,0 y 1 en el caso de la imagen presentada, para poder realizar

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

la convolución el filtro recorrerá elemento por elemento, multiplicándolo y sumando los resultados, estos resultados irán a una matriz nueva generada que la llamaremos capa convolucionada la cual reduce el tamaño de la matriz de la capa de partida a 3x3, debido a que al aplicar la convolución los bordes de la imagen no son considerados.

- **Filtro o Kernel**

Es una matriz de pesos que se desplazan sobre la entrada y realizan la operación de convolución, cada filtro aplicado extrae una característica especifica en la entrada.

- **Mapas de Características**

Al aplicar la operación de convolución, en la salida se crea un mapa de características, que representa la presencia de una característica especifica en diferentes partes de la imagen.

- **Capas de Activación**

Después de cada operación de convolución se debe aplicar una capa de activación, normalmente ReLU, que se encarga de introducir no linealidad al modelo y ayudar en el aprendizaje de representaciones más complejas

- **Capas Pooling**

Estas capas están diseñadas para reducir la dimensionalidad de los mapas de características, disminuyendo la carga computacional y el sobreajuste, la operación más común es la Max Pooling, que toma el valor máximo de una región específica.

- **Capas Completamente Conectadas**

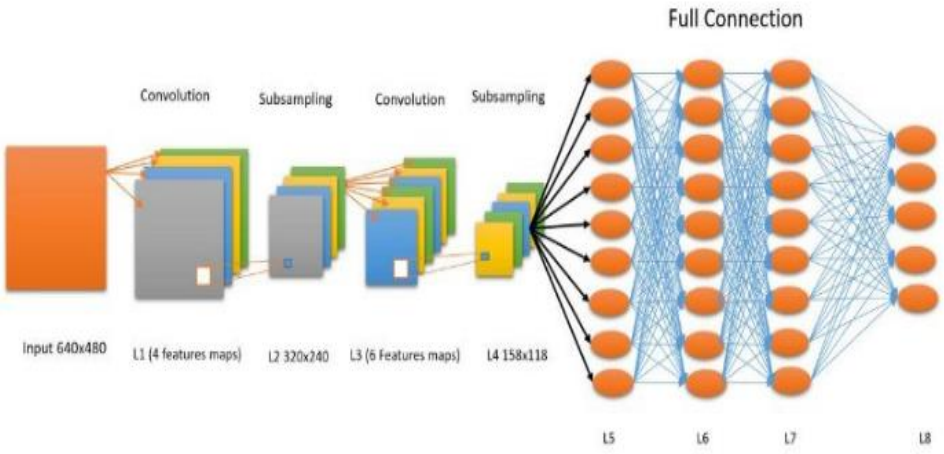
Se encuentran al final de la red, las capas totalmente conectadas o capas densas combinan las características que se aprendieron para realizar la clasificación o la regresión, cuando se trata de clasificación de imágenes normalmente la última capa es una Softmax, que genera probabilidades para cada clase (Goodfellow et al., 2017).

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 26

Proceso de una red neuronal convolucional.



Nota. Tomado de De la Rosa, 2016

3.2 Arquitecturas de las redes neuronales convolucionales

- **LeNet-5**

Se trata de una de las primeras redes convolucionales, está diseñada para el reconocimiento de dígitos escritos a mano en el dataset, fue desarrollada por Yann LeCun en el año 1998, al tratarse de una de las primeras redes convolucionales, cuenta con una estructura muy simple comparada con redes más modernas, esta estructura consta de 2 capas convolucionales seguidas de capas de Pooling y por último capas completamente conectadas.

Cuenta con un total de 7 capas, sin tener en cuenta capas de entrada y salida.

- **AlexNet**

Esta red fue propuesta por Alex Krizhevsky, utiliza varias capas convolucionales y Max Pooling, además popularizo el uso de la activación ReLU, desarrollada en el año 2012, fue una revolución debido a la integración del GPU para el entrenamiento de modelos profundos.

Esta red es similar a la red LeNet-5 sin embargo esta es más profunda y contiene más parámetros, esta red en su estructura cuenta con 5 capas convolucionales y 3 capas completamente conectas.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Esta red hace uso de las activaciones ReLU, Dropout para poder solucionar problemas como sobreajuste y normalización local de respuesta (LRN).

La red cuenta con alrededor de 60 millones de parámetros lo cual la hace ideal para la clasificación de imágenes en el dataset ImageNet, uno de sus problemas se debe al gran consumo de memoria debido a la gran cantidad de parámetros.

- **VGGNet**

Esta arquitectura fue propuesta por Karen Simonyan y Andrew Zisserman esta red introdujo el novedoso uso de pequeñas convoluciones de 3x3, enfatizando profundidad y simplicidad en el diseño, desarrollada en el año 2014.

Esta arquitectura cuenta con redes muy profundas como la VGG-16 y la VGG-19 las cuales cuentan con 16 y 19 capas respectivamente, además se aplica a cada una de las convoluciones filtros de 3x3, seguidas de capas Pooling y en su parte final se encuentra una capa completamente conectada.

En VGGNet se cuenta con hasta 138 millones de parámetros lo que hace que los requerimientos de memoria y tiempo de entrenamiento sean altos, sin embargo, es una red ideal para realizar clasificación de imágenes en ImageNet y transferencia de aprendizaje.

- **Inception**

Esta arquitectura fue propuesta por Christian Szegedy, esta arquitectura implementaba el uso de módulos Inception que se encargan de aplicar filtros de diferentes tamaños en paralelo, lo que le permite a la red capturar patrones a varias escalas, desarrollada en el año 2014.

La arquitectura Inception son de redes profundas considerando que en su primera versión ya se contaban con 22 capas sin embargo los parámetros eran menores debido al uso de convoluciones 1x1 para reducir las dimensiones, además de no contener capas completamente conectadas.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Este modelo cuenta con 4 millones de parámetros haciéndola más pequeña que la arquitectura VGGNet, normalmente se aplica la estructura Inception en tareas complejas de clasificación y localización.

- **ResNet**

Esta arquitectura fue propuesta por Kaiming He, introduce bloques residuales que le permiten a la red el aprendizaje de funciones de identidad. Lo que permite a las redes muy profundas la posibilidad de poder entrenarse sin problemas de desvanecimiento de gradiente, desarrollada en el año 2015.

Esta arquitectura posee redes extremadamente profundas que cuentan hasta con 152 capas, sin embargo, cuenta con un diseño simple el cual genera un entrenamiento facilitado, su tamaño es moderada y cuenta con menor riesgo de sobreajuste gracias a las conexiones residuales, normalmente se utiliza esta arquitectura para tareas de clasificación, segmentación, detección de objetos, una de sus limitantes es que su implementación es más complicada que AlexNet o VGG (Goodfellow et al., 2017).

Tabla 4

Diferencia entre los diferentes tipos de arquitectura para redes neuronales convolucionales.

Arquitectura	Año	Innovación	Profundidad de capas
LeNet-5	1998	Primera Red convolucional	7 capas Red poco profunda
AlexNet	2012	Utilización de la GPU en entrenamiento de red y uso de ReLU como función de activación	8 capas Red poco profunda
VGGNet	2014	Introdujo el uso de convoluciones pequeñas de 3x3 para enfatizar la profundidad y la simplicidad del diseño.	16 a 19 capas Red profundidad media
Inception	2014	Introdujo el uso de módulos Inception además de generar redes profundas, pero con menos parámetros, gracias a su convolución 1x1 que reduce las dimensiones.	22 capas Red con profundidad media
ResNet	2015	Uso de conexiones residuales.	152 a más capas. red extremadamente profunda

Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

3.3 Aplicaciones de las redes neuronales convolucionales

- Reconocimiento de imágenes
- Detección de objetos
- Segmentación de imágenes
- Reconocimiento facial
- Procesamiento de imágenes medicas
- Procesamiento de video (Goodfellow et al., 2017).

3.4 Ventajas y desventajas de las redes neuronales convolucionales.

- **Ventajas**

1. Extracción automática de características

   Las redes convolucionales aprenden de forma automática las características importantes de las imágenes sin necesidad de ingeniería manual de características.

2. Escalabilidad

   Las redes convolucionales trabajan de forma óptima con datos de gran tamaño y pueden entrenarse teniendo grandes cantidades de imágenes.

3. Capacidad de generalización

   Las redes convolucionales tienen menos probabilidades de tener sobreajuste al reducir el número de parámetros con conectividad local y compartición de pesos

- **Desventajas**

1. Necesidad de datos y poder de cómputo

   Las redes convolucionales requieren grandes cantidades de datos etiquetados y poder computacional (GPU) para entrenarse de manera efectiva.

2. Dificultad para interpretar la toma de decisiones

   Las redes convolucionales son modelos que se conocen como caja negra debido a que su toma de decisiones es de difícil interpretación.

3. Susceptibles a pequeñas perturbaciones

Las redes convolucionales son susceptibles a ejemplos adversariales, donde las pequeñas modificaciones pueden cambiar drásticamente las predicciones (Goodfellow et al., 2017).

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

4. Redes neuronales recurrentes

Las RNN están especializadas en datos secuenciales como lo son textos o series temporales, estas redes mantienen un estado interno que les permite recordar información de entradas anteriores (Goodfellow et al., 2017).

4.1. Arquitectura Básica de una red neuronal recurrente

Las redes neuronales recurrentes poseen una célula recurrente que se encarga de recibir la entrada actual, así como el estado previo, gracias a este estado es que la red obtiene la capacidad de recordar información previa de la secuencia. La red neuronal recurrente tiene la siguiente arquitectura.

- Entrada

  Se encarga de recibir las señales que se dan en el paso del tiempo, es decir llega información como palabras oraciones o datos de una serie temporal.

- Estado oculto

  Es el encargado de trabajar como la memoria de este tipo de red, la red mantiene un estado oculto h_t, este estado oculto es una función de la entrada actual como del estado oculto anterior.

- Salida

  En cada paso las redes neuronales recurrentes son capaces de generar una salida o_t, que son utilizadas en las tareas específicas, como predicción de palabras.

Las redes neuronales recurrentes se pueden definir por la siguiente expresión:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b_h)$$

Donde:

h_t: Es el estado oculto de la red en el tiempo t, este valor representa a la memoria de la red, la cual se actualiza en cada paso de tiempo.

h_{t-1} : Es el estado oculto anterior en el tiempo t-.1

σ: Es la función de activación

W_h : Es la matriz de pesos que se aplica al estado oculto anterior.

W_x: Es la matriz de pesos que se aplica a la entrada actual xt

b_h: Es el Bias o Sesgo, este permite a la red ajustar el estado oculto de forma independiente a las entradas y los estados previos (Goodfellow et al., 2017).

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

4.2. Tipos de redes neuronales recurrentes

Los tipos de redes neuronales recurrentes están definidos por la forma de conexión de sus entradas y salidas.

- **One to One**

  Es la arquitectura clásica de una red neuronal que no posee recurrencia en donde solo existe una entrada y una salida.

- **One to Many**

  En esta arquitectura solo existe una entrada la cual produce una secuencia de salidas, es aplicable en generación de texto o música a partir de una entrada única.

- **Many to One**

  En esta arquitectura todas las entradas existentes producen una única salida, es utilizada cuando se requiere realizar clasificación de secuencias.

- **Many to Many**

  En esta arquitectura cada entrada existente obtiene una salida, se utiliza en tareas como traducción automática (Goodfellow et al., 2017).

4.3. Problemas en las Redes neuronales recurrentes básicas

Los problemas de las redes neuronales recurrentes básicas se presenta cuando la tarea trabaja con secuencias muy largas de datos lo cual causa los siguientes problemas:

- Desvanecimiento de gradiente

  Cuando se trabaja con secuencias largas ocurre que los gradientes tienden a volverse extremadamente pequeños, lo cual causa dificultad en el aprendizaje de relaciones de largo alcance, esto debido a que al retro propagar los gradientes a través de muchos pasos estos gradientes se multiplican repetidamente y tienden a desaparecer.

- Explosión de gradiente

  En este caso ocurre lo contrario al problema de desvanecimiento de gradiente, dado que aquí los gradientes tienden a crecer exponencialmente durante la retro propagación, lo cual causa inestabilidad en el entrenamiento (Goodfellow et al., 2017).

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

4.4. Variantes de mejora para las redes neuronales recurrentes

Para mitigar los problemas de desvanecimiento de gradiente y explosión de gradiente se introdujeron variantes a las redes neuronales recurrentes básicas.

- **LSTM (Long Short-Term Memory)**

  En esta variante las celdas de memoria están diseñadas para recordar dependencias de largo plazo mediante una arquitectura de puertas que se encargan de controlar el flujo de información.

  Puerta de entrada:

  - Controla cuando de la nueva información que se ingresa a x_t se almacena en la celda.

  Puerta de olvido

  - Controla cuanta información permanecerá y cuanta información será eliminada

  Puerta de salida

  - Controla que información se utilizara para calcular el estado oculto h_t y la salida

- **GRU (Gated Recurrent Unit)**

  Esta variante es una simplificación de la arquitectura LSTM, debido a que combina las puertas de entrada y olvido en una sola puerta, que la llama puerta de actualización con la cual es capaz de omitir la celda de memoria, esta variante es más rápida de entrenar, sin embargo, en ciertas aplicaciones no son tan efectivas como las LSTM (Goodfellow et al., 2017).

4.5. Método de entrenamiento de las redes neuronales recurrentes

El entrenamiento de las redes neuronales recurrentes se realiza mediante el método de la retro propagación a través del tiempo (BPTT).

Este método consiste en que la red sea capaz de extiende sus estados ocultos a lo largo de la secuencia y para cada paso de tiempo calcula su gradiente, este procedimiento requiere de un cómputo intensivo y puede derivar en problemas de desvanecimiento o explosión de gradiente en secuencias muy largas (Goodfellow et al., 2017).

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

4.6. Aplicaciones de las redes neuronales recurrentes

- Procesamiento de lenguaje natural.
- Reconocimiento de voz.
- Generación de música y arte.
- Series de tiempo y predicciones.
- Análisis de Videos (Goodfellow et al., 2017).

4.7. Ventajas

- Las redes neuronales recurrentes tienen capacidad de memoria.
- Tiene una arquitectura adecuada para tareas que requieran el uso de información en el tiempo (Goodfellow et al., 2017).

4.8. Desventajas

- Problemas de desvanecimiento y explosión de gradiente
- Hacen uso intensivo de hardware para lograr su entrenamiento
- La red presenta sensibilidad a la secuencia de entrenamiento de modo que, si se cambia algún dato, puede causar variaciones (Goodfellow et al., 2017).

5. Transfer learning

Es una técnica de aprendizaje profundo que facilita la mejora de un modelo nuevo, utilizando como base un modelo previamente entrenado, se utiliza mayoritariamente cuando se tiene una cantidad limitada de datos para la tarea nueva a desarrollar.

5.1. Motivación de uso

La técnica de transfer learning debe ser utilizada cuando no exista gran cantidad de datos disponibles para realizar el entrenamiento de una red desde cero, su uso es capaz de ahorrar tiempo y recursos computacionales.

5.2. Funcionamiento

Para aplicar esta técnica existen algunas formas que su uso dependerá del tipo de dato y del dominio de la tarea.

- **Congelación de capas**

  Este método funciona congelando las primeras capas de una red neuronal preentrenada para enfocar el entrenamiento únicamente en las capas finales, es decir que se dejan las primeras capas de una red entrenada debido a que

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

estas ya reconocen bordes, patrones o texturas básicos, sin embargo, se ajusta las capas finales para darle el nuevo enfoque para la aplicación final y así lograr distinguir características más específicas.

- **Ajuste fino**

  En este método se descongelan algunas de las capas intermedias o finales de un modelo preentrenado y se vuelve a entrenar nuevamente, pero esta vez con un aprendizaje de tasa baja, este método se utiliza cuando las características que se requieren en la nueva aplicación pueden beneficiarse de ajustes en el modelo completo, pero manteniendo la estructura del conocimiento previamente aprendido.

5.3. Ventajas

- Ahorro de tiempo y recursos
- Requiere menor número de datos de entrenamiento
- Rendimiento alto en tareas relacionadas con el modelo preentrenado.

5.4. Desventajas

- En caso de que la tarea no guarde mucha relación con la red preentrenada, la tasa de efectividad para la nueva tarea es baja, debido a que esta técnica depende de datos similares.
- Es probable que sufra de sobreajuste si el modelo preentrenado se ajusta mucho a la nueva tarea que no cuenta con demasiados datos (Goodfellow et al., 2017).

6. Aprendizaje por refuerzo

Se utiliza cuando el aprendizaje se desarrolla en contacto con el entorno, el sistema recibe recompensas o penalizaciones basadas en las acciones que realiza, gracias a esto se logra el entrenamiento y se mejora las tomas de decisiones posteriores.

6.1. Conceptos clave de aprendizaje por refuerzo

- **Agente**

  Se le conoce como agente al sistema o entidad que toma acciones en un entorno con el objetivo de adquirir mayores recompensas.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

- **Entorno**

  Es el mundo con el que interactúa el agente, está definido por sus estados, recompensas y transiciones de estados.

- **Estado**

  Representa la situación actual del agente dentro del entorno.

- **Acción**

  Es una decisión que el agente puede tomar en un estado dado.

- **Recompensa**

  La adquiere el agente después de realizada la acción.

- **Política**

  Es la estrategia que tiene el agente para la elección de acciones en cada estado, puede variar entre se determinista es decir la toma de acción específica para cada estado o probabilística, que basa sus elecciones en una distribución de probabilidades.

- **Función de valor**

  Se trata de la estimación del valor esperado en un estado específico, teniendo en cuenta las recompensas que el agente pueda obtener, es necesaria para evaluar la calidad de los estados.

- **Función Q**

  Es conocida como la función de acción valor, se encarga de evaluar el valor esperado de tomar una acción a en un estado s y seguir una política optima a partir de eso.

6.2. Tipos de aprendizaje por refuerzo

1. Aprendizaje por refuerzo basado en valor

El agente es capaz de aprender a estimar el valor que tendrá cada estado. El objetivo del agente es aprender una función de valor optima y derivar una política a partir de ella. Dentro de este tipo de aprendizaje se deber tomar en cuenta las siguientes formas:

- **Q-Learning**

  El agente aprende una función Q que le servirá para estimar el valor de tomar una acción ante un estado y seguirá una política optima que se basará en esta estimación.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

- **SARSA**

  Es una variante de Q-Learning donde el agente aprende la función Q pero basándose en la acción que se tomó realmente y no solo en el valor máximo de la próxima acción (Goodfellow et al., 2017).

2. Aprendizaje por refuerzo basado en política

En este tipo de aprendizaje el agente aprende directamente de una política en lugar de aprender de los estados o acciones, es mayormente utilizado cuando los entornos en los que se desarrolla el agente son complejos y continuos.

- **Método de la gradiente por política**

  Se utiliza para optimizar la política que sigue el agente, mediante el uso de gradientes.

- **Método actor-critico**

  Aquí se combina la estimación de una política que es el actor y la estimación de un valor que será el crítico, para la mejora de la convergencia (Goodfellow et al., 2017).

3. Aprendizaje por refuerzo profundo

Es una combinación de redes neuronales profundas y de algoritmos de RL. Lo que permite al agente el manejo de entornos complejos de alta dimensionalidad como lo son imágenes, aquí se tienen los siguientes tipos.

- **DQN (Deep Q-Networks)**

  Aproxima la función Q mediante la combinación de redes neuronales con Q-Learning

- **DDPG (Deep Deterministic Policy Gradient)**

  Es un método que se basa en políticas para problemas continuos

- **A3C (Asynchronous Advantage Actor-Critic)**

  Utiliza múltiples copias de un agente que interactúan con el entorno de manera asíncrona (Goodfellow et al., 2017).

6.3. Ventajas

- El aprendizaje por refuerzo es capaz de realizar su aprendizaje sin supervisión directa, no requiere datos etiquetados.
- Es capaz de resolver problemas secuenciales complejos que dependan de la toma de decisiones (Goodfellow et al., 2017).

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

6.4. Desventajas

- Se requiere mucho tiempo y hardware para el entrenamiento.
- La configuración de hiperparámetros hace un cambio significativo en el entrenamiento (Goodfellow et al., 2017).

2.2.4.1. Funciones de activación.

Las funciones de activación en redes neuronales son componentes esenciales que permiten introducir no linealidades en el modelo, esto se utiliza para permitir a la red la capacidad de aprender relaciones complejas en los datos (Goodfellow et al., 2017).

1. **Función de activación sigmoide**

   La función de activación sigmoide cumple la función de convertir cualquier valor de entrada en un rango de salida entre 0 y 1, es útil cuando se requiere resolver problemas de clasificación binaria.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

   1.1. Ventajas

   Esta función produce una salida en un rango limitado de 0 y 1, lo que la vuelve ideal para modelos probabilísticos.

   1.2. Desventajas

   Tiene desvanecimiento del gradiente, lo que ocasiona un aprendizaje lento (Goodfellow et al., 2017)

2. **Función de activación Tanh**

   La función de activación tangente hiperbólica es una extensión de la función sigmoide sin embargo esta función transforma los valores en un rango de -1 a 1.

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

   2.1. Ventajas

   Esta función de activación mejora el problema de desvanecimiento de gradiente.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

2.2. Desventajas

Esta función aun sufre de desvanecimiento de gradiente sin embargo esto pasa en menor medida (Goodfellow et al., 2017).

3. **Función de activación ReLU**

La función de activación ReLU es la más conocida cuando se trata de redes neuronales profundas.

$$ReLU(x) = \max(0, x)$$

3.1.1. Ventajas

Es útil para resolver el problema de desvanecimiento de gradiente, además que posee una eficiencia computacional superior.

3.1.2. Desventajas

Si bien es capaz de resolver el problema de desvanecimiento de gradiente, posee un problema que ocurre cuando las entradas que se reciben son constantemente negativas, este problema se llama "neurona muerta" (Goodfellow et al., 2017).

4. **Función de activación Leaky ReLU**

Esta función de activación es una modificación a la función ReLU, su propósito es intentar minimizar el problema de neurona muerta, otorgando una pequeña pendiente en las entradas negativas.

$$f(x) = \text{x si x} > 0, \text{y } f(x) = \alpha\text{x si x} \leq 0, \text{donde } \alpha \text{ es un valor pequeño}$$

4.1. Ventajas

Logra mitigar el problema de neuronas muertas que posee la función ReLU.

4.2. Desventajas

Puede introducir un pequeño sesgo en las activaciones (Goodfellow et al., 2017).

5. **Función Softmax**

Es una función que convierte un vector de valores en una distribución de probabilidad

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

5.1. Ventajas

Es útil en la última capa de las redes neuronales para problemas de clasificación multiclase.

5.2. Desventajas

No es útil para capas ocultas, ya que su diseño está hecho para la salida final de un clasificador (Goodfellow et al., 2017).

2.2.5. Clasificación de frutas mediante inteligencia artificial

Para lograr la clasificación de las frutas o imágenes en general mediante la inteligencia artificial, se utilizan diversas técnicas, que se clasifican desde métodos tradicionales hasta enfoques basados en aprendizaje profundo.

2.2.5.1. Métodos Clásicos de Clasificación.

Estos métodos funcionan bien cuando los conjuntos de datos son pequeños y las características que se encuentran en los objetos son de fácil identificación.

Para estos métodos se utilizan la extracción de características sumado a los algoritmos de aprendizaje automático, para los descriptores de características los métodos más usados son los siguientes

1. **SIFT (Scale-Invariant Feature Transform)**

   Esta técnica consta en la extracción de puntos clave que se describirán mediante vectores de características, basa su funcionamiento en Gauss y asigna una orientación dominante en cada punto clave, es mayormente utilizada para la detección de características robustas ya sea que la imagen tenga cambio de iluminación, perspectiva o escala. Sin embargo, su utilización tiene un costo computacional elevado.

2. **SURF (Speeded-Up Robust Features)**

   Esta técnica se trata de una optimización de SIFT, donde se adquiere una mayor velocidad de detección de características, para lograr esto, cambia las diferencias de gauss por los filtros de Haar, esto con el objetivo de encontrar los puntos clave de manera más rápida, es mayormente utilizado cuando se requiere una aplicación en tiempo real, una de sus desventajas es que es sensible a los cambios de iluminación.

3. **ORB (Oriented FAST and Rotated BRIEF)**

   Esta técnica es una alternativa más rápida y eficiente a los métodos SIFT y SURF, utiliza el detector de características FAST para la detección de puntos clave y BRIEF

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

para la descripción de esas características, su utilización se justifica debido a la rapidez y eficiencia que presenta, lo cual permite el procesamiento en tiempo real, además de ser capaz de detectar esquinas y puntos de interés sin la necesidad del cálculo de escalas o derivadas como si se realiza en el caso de SIFT y Surf, además al combinarse con BRIEF que son descriptores binarios, se logra una detección de características rápida y robusta.

4. **HOG (Histogram of Oriented Gradients)**

   Este método se enfoca esencialmente en la detección de formas y texturas su funcionamiento se basa en la división de la imagen en celdas y su posterior cálculo de histogramas de gradientes orientados en cada celda, se utiliza comúnmente para la detección de objetos, su limitación radica en que no es capaz de detectar puntos clave individuales, si no que representa características globales (Gonzalez & Woods, 2018).

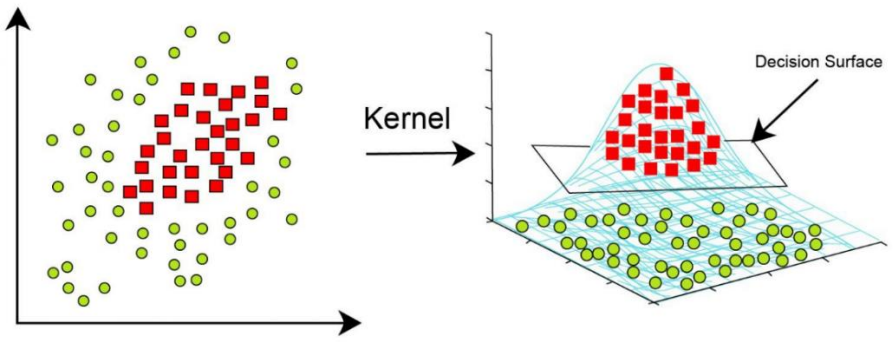
Entre los clasificadores más comunes tenemos los siguientes.

1. **SVM (Support Vector Machines)**

   El SVM es un algoritmo de clasificación y regresión supervisado, su función es la de encontrar un hiperplano optimo en donde pueda separar los datos en clases diferentes, para ello utiliza la técnica de márgenes máximos lo cual mejora la generalización, utiliza Kernels de modo que es capaz de trabajar con datos lineales y no lineales, la desventaja más resaltante de este algoritmo se encuentra cuando se trabaja con conjuntos grandes de datos, debido a que su procesamiento se vuelve lento (Gonzalez & Woods, 2018).

Figura 27

Algoritmo Support Vector Machines



Nota. Tomado de Van Otten, 2024

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

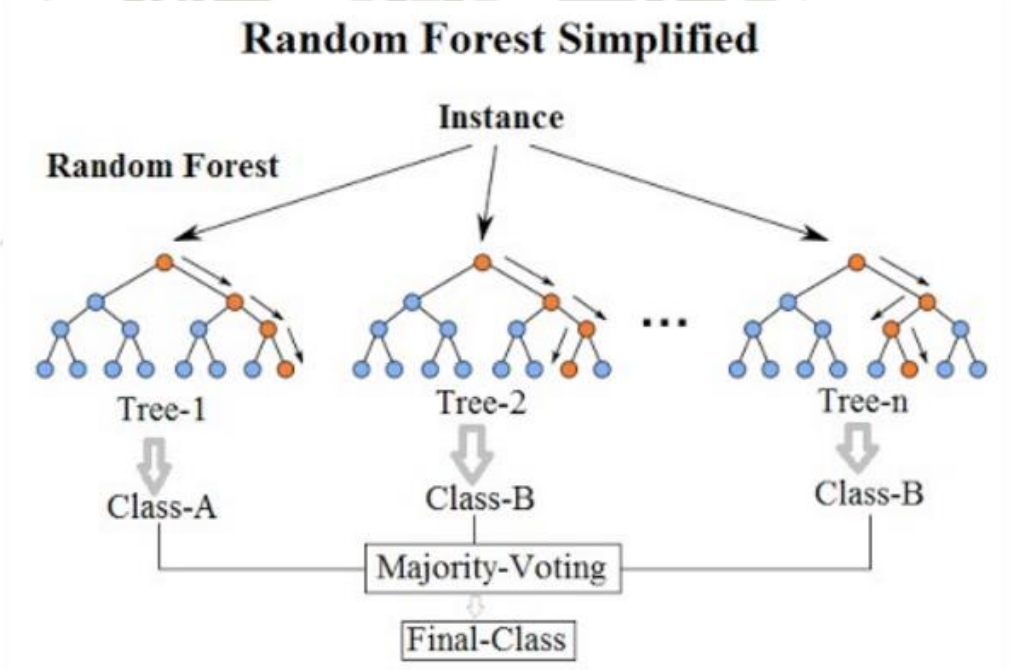
2. Random Forest

El algoritmo de Random Forest o bosques aleatorios, se trata de un algoritmo de clasificación y regresión basado en conjuntos, su funcionamiento se basa en el uso de múltiples arboles de decisión de los cuales combina sus resultados y somete a votación, para determinar la clase, mejorar la precisión y evitar el sobreajuste.

Dentro de sus características principales se encuentra su robustes ante ruidos y falta de datos además de su funcionamiento destacable en problemas de clasificación multiclase y regresión, su desventaja radica en que, al utilizar varios árboles de decisión, el proceso se vuelve lento (Gonzalez & Woods, 2018).

Figura 28

Algoritmo Random Forest



Nota. Tomado de Advanced Engineering Informatics, 2023

3. K-Nearest Neighbors (KNN)

Este algoritmo de los vecinos más cercanos es un algoritmo de clasificación y regresión basado en instancias el cual no usa un entramiento explicito, es decir que no existe un entrenamiento formal como en el algoritmo SVM y Random Forest, en este algoritmo se guardan los datos de entrenamiento y cuando se ingresa un nuevo dato, este busca un K vecino más cercanos y decide la clase basándose en estos.

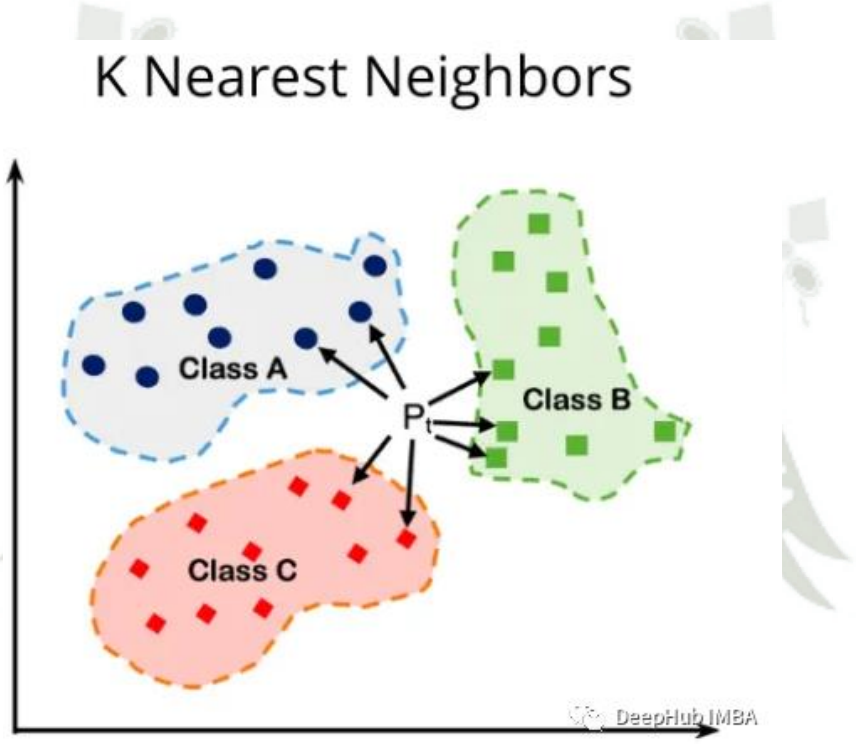
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

La mayor desventaja que tiene este algoritmo es que a diferencia de otros, este realiza la mayor parte del trabajo en la fase de predicción, además de ser más lento que otras técnicas debido a que debe calcular distancias con todos los datos almacenados en cada consulta (Gonzalez & Woods, 2018).

Figura 29

Algoritmo K-Nearest Neighbors



Nota. Tomado de DeepHub IMBA, 2025

4. **Redes Bayesianas**

Se trata de un algoritmo probabilístico que basa su funcionamiento en la teoría de Bayes, la cual representa un conjunto de variables aleatorias y sus relaciones de dependencia condicional a través de un grafo dirigido acíclico (DAG).

Es mayormente utilizada para el razonamiento bajo incertidumbre, es decir es capaz de calcular probabilidades de variables desconocidas a partir de variables observadas.

Este algoritmo requiere de un experto para su interpretación de la estructura DAG, además de requerir en su uso, métodos automáticos complejos, la inferencia para cuando se tiene redes grandes, suele ser costosa (Gonzalez & Woods, 2018).

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

2.2.5.2. Deep Learning (Aprendizaje Profundo)

Los métodos de aprendizaje profundo se basan en el uso de redes neuronales profundas y en la actualidad son los más utilizados, dentro de los más populares se encuentran los siguientes.

- **Redes neuronales convolucionales**

  Las CNN son el estándar en la clasificación de imágenes debido a su gran capacidad de aprender automáticamente las características relevantes.
- **Transfer Learning**

  En el aprendizaje por transferencia se utiliza los modelos preentrenados, los cuales se ajustan con datos nuevos, para poder llegar a la clasificación del nuevo problema.
- **Vision Transformers**

  Utiliza los mecanismos de atención en lugar de las convoluciones, los mecanismos de atención son técnicas que le permiten a la red enfocarse en las partes más relevantes de una imagen mientras ignora las regiones de no interés (Gonzalez & Woods, 2018).

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

CAPÍTULO III

3. DISEÑO DEL CLASIFICADOR POR VISION ARTIFICIAL Y REDES NEURONALES

3.1. Elección de entorno de desarrollo, lenguaje y controlador.

La elaboración del presente proyecto se realizó basando su implementación en la sencillez en términos de programación y el costo reducido.

Teniendo las premisas anteriores en mente se procede a realizar la elección tanto de software, tipo de lenguaje y el controlador a emplear.

3.1.1. Entorno de desarrollo y lenguaje de programación

Para este apartado se tomó en cuenta dos entornos en específico, Matlab y Spyder, cada cual, con un lenguaje de programación diferente, tanto Matlab con el M-code y Spyder con lenguaje Python. Sin embargo, una de las limitantes encontradas en el uso de Matlab es que para su uso se necesita pagar una membresía a comparación de Spyder que es un entorno de desarrollo de libre acceso.

Si bien Matlab puede realizar de la misma manera el proyecto, es menos flexible en términos de compatibilidad con Arduino, requiere más parámetros configurables y las codificaciones suelen ser de mayor dimensión debido a que es necesario llamar más funciones para realizar una acción.

Es por ello que se decidió por realizar el proyecto con el entorno de desarrollo Spyder el cual utiliza lenguaje Python este tipo de lenguaje brinda una sintaxis simple, además se tiene un soporte completo con Keras y TensorFlow, posee gran facilidad y flexibilidad cuando se requiere la integración con Arduino, mediante “pyserial” , además con el uso de notebook interactivo Jupyter, permite realizar el entrenamiento y visualizar los resultados con gráficos, compilando la sección requerida, sin necesidad de tener que compilar el programa entero siempre que se realice una modificación, además que de contar con alguna duda se puede tener acceso a la comunidad de GitHub donde los mismos usuarios brindan soporte sobre problemas suscitados en aplicaciones parecidas.

3.1.2. Controlador

El proyecto se realizó teniendo en mente un prototipo de aplicación fija controlado por computadora, si bien es posible realizar la programación y subirla a un microprocesador como Raspberry Pi, esto implica un costo adicional, por tal motivo se sacrificó la portabilidad que otorga esta opción.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

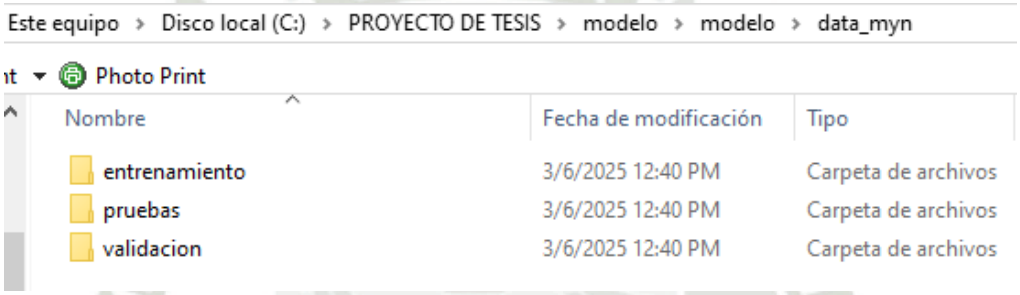
3.2. Datos de entrenamiento

Previo al desarrollo del entrenamiento del modelo se requiere hacer una recopilación de los datos que se otorgaran al modelo para su entrenamiento, en el caso de este proyecto se trabajaran con manzanas y naranjas por lo cual se deben conseguir imágenes suficientes que permitan el entrenamiento de la red de manera óptima.

Para ello se consideran 3 carpetas que serán necesarias para realizar un correcto entrenamiento:

Figura 30

Carpetas utilizadas para el entrenamiento de la red neuronal.



Nota. Elaboración propia.

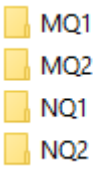
Como se puede observar en la imagen tenemos las carpetas de entrenamiento, pruebas y validación.

3.2.1. Carpeta de entrenamiento

En la carpeta de entrenamiento contaremos con subcarpetas que representaran a las calidades de las manzanas y naranjas que se utilizaran para que el modelo sea capaz de clasificar la fruta dentro de 4 calidades posibles.

Figura 31

Carpetas de entrenamiento por calidades



Nota. elaboración propia

En donde:

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Tabla 5

Calidades de manzanas, naranjas y su estado

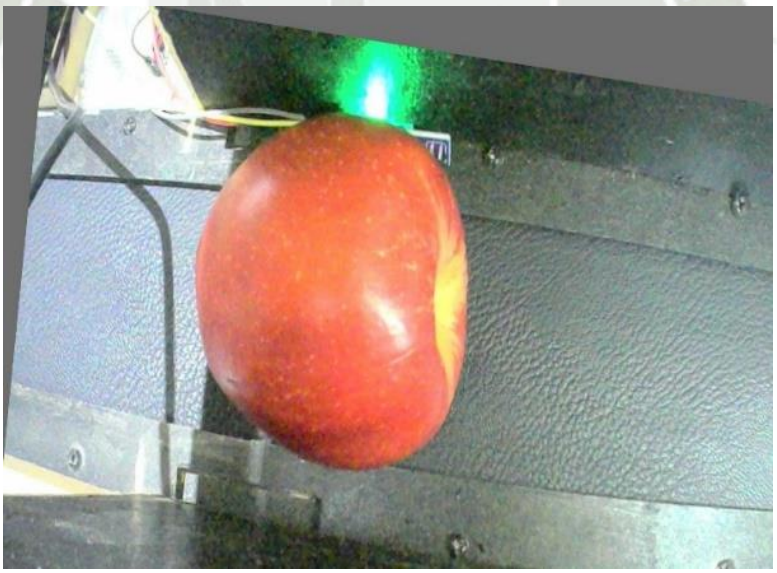
CALIDAD	ESTADO
MQ1 y NQ1	Hace referencia a un estado donde el color y la forma del fruto es óptima.
MQ2 y NQ2	Hace referencia a un estado donde la fruta tiene inicios de putrefacción, manchas o imperfecciones.

Nota. Elaboración propia.

Dentro de estas carpetas encontraremos imágenes que ayudaran al modelo a entender que exactamente es lo que se requiere en cada uno de los estados de clasificación. Dentro de la clasificación 1 se podrán encontrar imágenes como las siguientes:

Figura 32

Manzana Tipo Mq1.



Nota. Elaboración Propia.

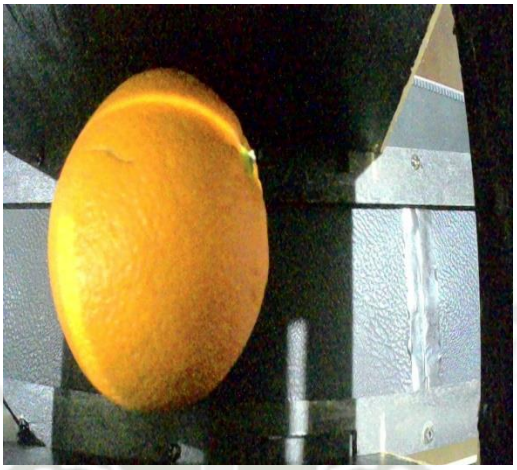
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Figura 33

Naranja Tipo Nq1



Nota. Elaboración propia.

Como se puede observar en las imágenes, aquí se otorgan datos de entrenamiento para una clasificación de un fruto en condiciones óptimas, tanto en color como en forma.

Dentro de la clasificación 2 se podrán encontrar imágenes como estas:

Figura 34

Manzana Tipo Mq2



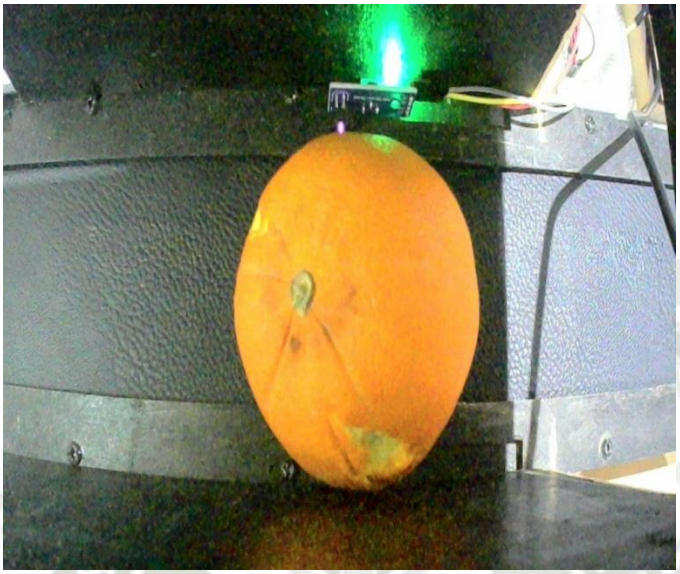
Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 35

Naranja Tipo Nq2



Nota. Elaboración propia.

En la clasificación de tipo 2 ya se encuentran signos de putrefacción o manchas en la piel del fruto que no concuerda con el estado optimo de este.

Tabla 6

Cantidad de imágenes dentro de carpetas de entrenamiento

CALIDAD	CANTIDAD IMAGENES
MQ1	603
MQ2	447
NQ1	652
NQ2	464

Nota. Elaboración propia

3.2.2. Carpeta de Validación

La carpeta de validación contiene un 20% del tamaño total de la carpeta de entrenamiento y se encarga de entregar al modelo la capacidad de realizar predicciones mientras está en proceso de aprendizaje, logra evitar que el modelo caiga en un sobre aprendizaje o memorización.

Para el modelo entrenado para este proyecto, se tuvieron los siguientes números para datos de validación

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Tabla 7

Cantidad de imágenes dentro de las carpetas de validación.

CALIDAD	CANTIDAD IMAGENES
MQ1	85
MQ2	67
NQ1	110
NQ2	83

Nota. Elaboración propia

Una consideración sumamente importante en la carpeta de validación radica en que todos los datos que se encuentren allí no deben haber formado parte del proceso de entrenamiento, en caso de que se tomen datos de entrenamiento para validación, el modelo tiende a generar sobreajuste, es decir memorizar los datos, debido a que la validación lo que hace es entregarle al modelo imágenes que representan al mundo real donde funcionara y probar su capacidad de generalización con los datos que se entrenó, pero si estos datos de validación ya los vio en su entrenamiento, simplemente los detecta por memorización.

3.2.3. Carpeta de pruebas

La carpeta de pruebas es necesaria porque será la encargada de testear el funcionamiento del modelo posterior a su entrenamiento, las imágenes que contiene esta carpeta deben ser imágenes aleatorias, y que no estén contempladas en las carpetas de entrenamiento ni en la carpeta de validación.

Esta carpeta es necesario entregarlas al modelo con las calidades en su nombre, de la siguiente manera.

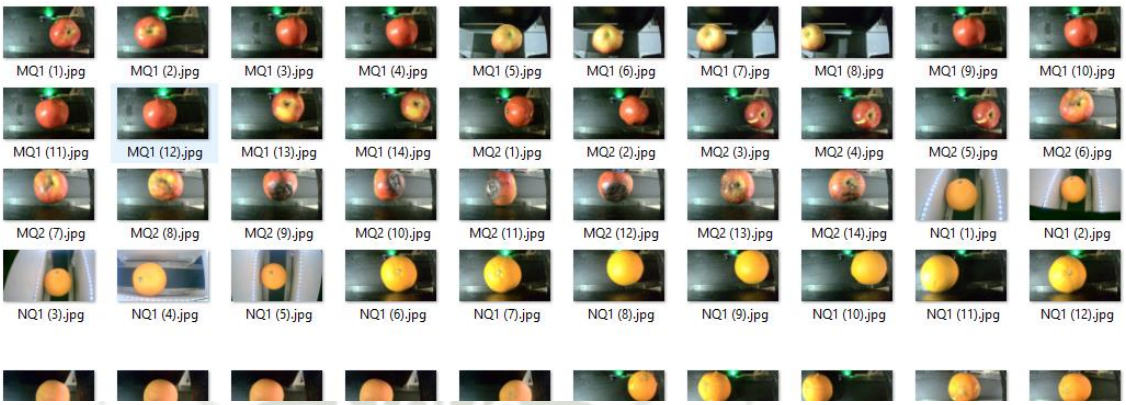
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 36

Imágenes Dentro De Carpeta De Prueba



Nota. Elaboración propia.

Esto se hace para que, al momento de probar la red, esta pueda comparar lo que clasifica con el resultado real que se espera tener.

Tabla 8

Cantidad de imágenes dentro de la carpeta de pruebas

CALIDAD	CANTIDAD IMAGENES
MQ1	14
MQ2	14
NQ1	12
NQ2	12

Nota. Elaboración Propia

3.3. Entrenamiento de la Red en Jupyter

3.3.1. Entorno de trabajo.

Mediante el programa Anaconda Navigator, creamos un entorno de trabajo que trabaje con Python 3.11.11, como se muestra en la siguiente imagen.

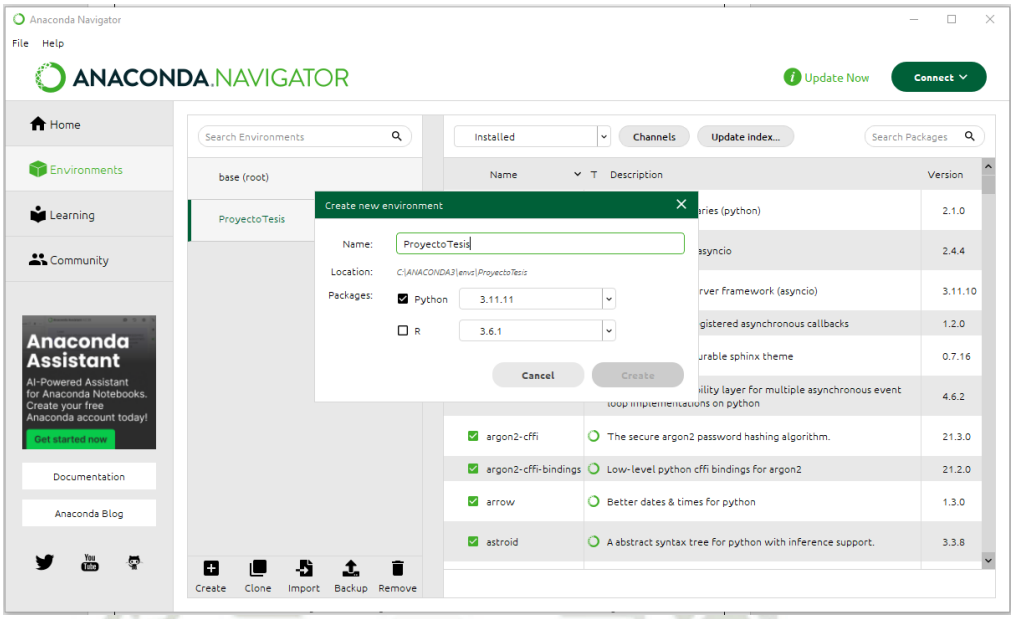
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 37

Creación de entorno de trabajo en anaconda navigator.



Nota. Elaboración propia.

3.3.2. Librerías

Para poder realizar el entrenamiento de la red, requerimos del uso de ciertas librerías que lograran facilitar la manipulación de datos, la construcción del modelo y su entrenamiento.

Para este apartado se resalta que para el modelo se realizaron varios ensayos con dos diferentes modelos de entrenamiento.

Una de ellas se genera la red entrenada con datos personales desde cero para esta red se utilizaron las siguientes librerías.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 38

Librerías para modelo entrenado desde cero

```
import pandas as pd # usada para la manipulación y el análisis de datos.
import numpy as np # utilizada para el manejo de matrices y operaciones numéricas.
import os # permite interactuar con el sistema operativo.
import matplotlib.pyplot as plt # usada para la creación de gráficos.
import itertools # proporciona funciones que crean iteradores eficientes.
import matplotlib.image as mpimg # Importa el módulo image de matplotlib, para manejar imágenes.
import tensorflow as tf # usada para el aprendizaje automático y redes neuronales.
import keras # biblioteca de alto nivel para redes neuronales, que corre sobre TensorFlow.
import cv2
from keras.utils import to_categorical# convierte vectores de clases en matrices binarias de clases.
from keras.layers import Dense, GlobalAveragePooling2D, Conv2D # Importa capas específicas de Keras para construir modelos.
from keras.layers import Dense, Dropout, Convolution2D, MaxPooling2D
from keras.layers import Flatten
from keras.preprocessing import image # Importa herramientas para la preprocesamiento de imágenes en Keras.
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras import applications # Importa aplicaciones preentrenadas de Keras.
from keras.applications.mobilenet import preprocess_input
from keras.models import Model, Sequential# para construir y configurar modelos de redes neuronales.
from keras.optimizers import Adam # Importa el optimizador Adam de Keras.
from mlxtend.evaluate import confusion_matrix # utilizada para evaluar modelos de clasificación.
from keras import backend as K # permite la compatibilidad con distintos backends (TensorFlow, Theano, etc.).
from keras.utils import plot_model # permite visualizar la estructura de un modelo de red neuronal.
import warnings
warnings.filterwarnings("ignore")
```

Nota. Elaboración propia.

Otra red fue entrenada utilizando el enfoque de transfer learning empleando la arquitectura de red preentrenada MobileNetV2, para esta red se utilizaron las siguientes librerías.

Figura 39

Librerías para modelo con arquitectura mobilenetv2

```
# Imports
import matplotlib.pyplot as plt #PERMITE GRAFICAR EL ENTRENAMIENTO DE LA RED EN LAS EPOCAS.
import numpy as np
import os
import cv2
import itertools
import math
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import confusion_matrix #NECESARIO PARA REALIZAR LA MATRIZ DE CONFUSION
```

Nota. Elaboración propia.

Para poder realizar la instalación de cada una de ellas, ingresamos al aplicativo "Anaconda Prompt" y procedemos a acceder a nuestro entorno virtual de la siguiente manera.

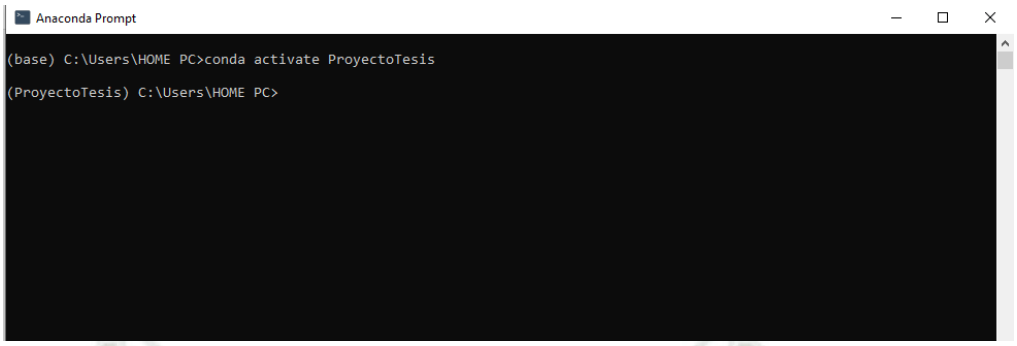
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 40

Activando entorno virtual en anaconda prompt.

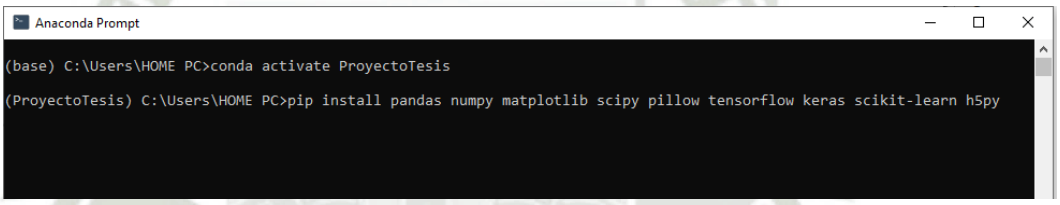


Nota. Elaboración propia.

Procedemos a instalar las librerías necesarias para el proyecto.

Figura 41

Instalación de librerías en anaconda prompt.



Nota. Elaboración propia.

Al hacer esto, nuestro entorno virtual "ProyectoTesis" contara con todas las librerías necesarias para proceder con el entrenamiento de la red.

3.3.3. Limpieza de variables de Keras y Tensorflow

Antes de empezar con el proceso de entrenamiento, se realiza la limpieza de Keras y Tensorflow de tal manera que nos aseguramos de no tener ninguna variable de entorno o algún Inter parámetro de la red pegado, para poder realizarlo se utiliza la siguiente línea de código.

" K.clear_session() #limpiar todas las variables de keras-tensorflow"

3.3.4. Extracción de características.

3.3.4.1.Para modelo entrenado desde cero.

En esta parte nosotros generaremos un modelo secuencial, el cual consta de una pila lineal de capas.

" model = Sequential() # Crea un modelo secuencial, que es una pila lineal de capas."

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Posteriormente se agregará una primera capa de convolución la cual para el caso de este modelo agregará 32 filtros, cuyo objetivo directo será otorgarle profundidad a la imagen que tenemos de entrada, extracción de características, bordes, texturas y patrones de la imagen, los datos que ingresan deben tener un tamaño estandarizado que para este modelo deberán ser de un tamaño de 160x160 y con un formato a color es decir RGB, todo ello debe estar contemplado en los parámetros de la capa de convolución.

Posterior a ello se agrega una capa de max Pooling con la cual seremos capaces de reducir el tamaño de salida de la convolución logrando así tener una mayor profundidad, pero un tamaño menor, mientras mantenemos las características más importantes, la reducción de max Pooling depende directamente del tamaño que nosotros le demos al pool, que para este modelo será un pool de (2,2).

“ # Primera capa de convolución

model.add(Convolution2D(filtrosConv1, tamano_filtro1, padding='same', input_shape=(160, 160, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=tamano_pool)) # Se añade una capa de max pooling.”

Como se puede observar en el código, la primera capa de convolución tiene diferentes parámetros “filtrosConv1, tamaño_filtro1, padding, input_Shape y activation”

Los dos primeros parámetros deberán ser previamente contemplados, tanto como para la primera convolución como la segunda y tercera, sin embargo, el parámetro input shape y la función de activación se añaden directamente en la línea de la convolución, las imágenes de entrada a la capa de convolución deberán ser de un tamaño 160x160 y el numero 3 indica que la imagen está ingresando a color es decir RGB, y en la función de activación se usa la función relu.

Del mismo modo en la capa max Pooling debemos previamente otorgar el tamaño del pool, todo ello se ve en las siguientes líneas de código.

“ #Arquitectura de la red neuronal convolucional

filtrosConv1 = 32 # numero de filtros que se aplica tras primera convolucion

tamano_filtro1 = (3, 3) #en primera convolucion

filtrosConv2 = 64 # numero de filtros que se aplica tras la segunda convolucion

tamano_filtro2 = (2, 2) #en segunda convolucion

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

filtrosConv3 = 64 # número de filtros para la tercera capa convolucional
tamano_filtro3 = (2, 2)   # en tercera convolución
tamano_pool = (2, 2) #   mejorar el avance de la convolucion"

En la imagen se puede observar que la primera convolución agregará 32 filtros, y el tamaño de los filtros será de (3,3) sin embargo el número de filtros aumenta para la segunda y tercera capa de convolución a 64, pero el tamaño del filtro baja a (2,2).

Posterior a la primera capa de convolución y su respectiva capa de max Pooling se procede a realizar el mismo proceso de convolución y max pooling para las dos etapas siguientes.

" # Segunda capa de convolución
model.add(Convolution2D(filtrosConv2, tamano_filtro2, padding='same'))
model.add(MaxPooling2D(pool_size=tamano_pool))

# Tercera capa de convolución
model.add(Convolution2D(filtrosConv3, tamano_filtro3, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=tamano_pool))"

Después de realizado el proceso de max Pooling se realiza un aplanado o "flatten" el cual tiene como finalidad convertir la salida de max Pooling que es un tensor 3D a un vector 1D, esto con el fin de volver compatible la entrada con las capas densas para su clasificación final, para ello se utiliza la siguiente línea de código.

"model.add(Flatten()) #aplanamos la imagen profunda, es decir nuestra información de la RNC solo va a tener una dimensión"

Una vez se logra obtener el vector 1D se introduce una capa densa (oculta) que trabajara con 256 neuronas, que trabajara con la función de activación ReLU, debido a que esta introduce no linealidad en la red y logra evitar los problemas de gradiente pequeño, el uso de esta capa es debido a que es capaz de aprender patrones abstractos

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

a partir de la salida convolucional, todo ello se representa en la siguiente línea de código.

“model.add(Dense(256, activation='relu')) #se añade una unica capa oculta con 256 neuronas”

Posterior a ello se agrega una capa Dropout que es una técnica de aprendizaje que ayuda a evitar el sobreajuste mediante la desactivación de neuronas de forma aleatoria, para el caso del modelo se tomara un Dropout de 0.5 (máximo recomendable), con el cual se tiene que en cada iteración del entrenamiento se apagara aleatoriamente 50% de las neuronas, esta parte se representa en la siguiente línea de código.

“model.add(Dropout(0.5)) #apagamos 50% de las neuronas en cada paso de entrenamiento, así evitamos sobre aprendizaje(overfitting)”

Luego se agregará una capa de salida con 4 neuronas, debido a que el modelo deberá ser capaz de clasificar la fruta en 4 estados de calidad distintos, siendo 2 de ellos para manzanas y 2 de ellos para naranjas, esta capa utilizara una función de activación Softmax debido a que esta convierte la salida en probabilidades de 0 a 1, donde la suma de todas las neuronas deberá ser 1, la neurona que obtenga mayor probabilidad determinara la clasificación final, todo esto se representa en la siguiente línea de código.

“model.add(Dense(4, activation='softmax')) # func. de activación de la última capa: softmax (nos indica la probabilidad
que tiene cada clase, y por ende elegimos la clasificación con mayor valor).”

Para mejorar el entendimiento de la arquitectura de la red, se grafica los pasos que realizara el modelo secuencial.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 42

Resumen de todas las capas, salidas y números de parámetros a entrenar de una red entrenada desde cero.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 160, 160, 32)	896
max_pooling2d (MaxPooling2D)	(None, 80, 80, 32)	0
conv2d_1 (Conv2D)	(None, 80, 80, 64)	8,256
max_pooling2d_1 (MaxPooling2D)	(None, 40, 40, 64)	0
conv2d_2 (Conv2D)	(None, 40, 40, 64)	16,448
max_pooling2d_2 (MaxPooling2D)	(None, 20, 20, 64)	0
flatten (Flatten)	(None, 25600)	0
dense (Dense)	(None, 256)	6,553,856
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 4)	1,028

Total params: 6,580,484 (25.10 MB)

Trainable params: 6,580,484 (25.10 MB)

Non-trainable params: 0 (0.00 B)

Nota. Elaboración propia.

En la figura 42 podemos observar los siguiente, se considera que nuestra entrada es una imagen de tamaño 160x160 a color es decir RGB, esta entrada posteriormente se le aplica una capa de convolución con 32 filtros para extracción de características, quedando la salida de la convolución con un tamaño de (160,160), debido a que en la capa de convolución no existe reducción, sin embargo en la capa siguiente de max Pooling, se realiza una reducción a (80,80) manteniendo los 32 filtros, debido a que nuestro max Pooling trabaja con una reducción de (2,2), la salida de esta ingresa a una segunda capa de convolución pero esta con 64 filtros, quedando la salida en (80,80,64) luego se aplica una segunda capa de max Pooling que deja la salida en (40,40,64), luego se llega a la tercera y última capa de convolución con también 64 filtros, que deja la salida después de pooling en (20,20,64)

Esta salida de max Pooling se multiplica todos los resultados en la capa de aplanado o flatten, dando como resultado el número 25600, número que representa un vector en 1D, el cual permite conectar con capas densas.

El vector en 1D ingresa a la capa densa que usara 256 neuronas, las cuales permitirán sacar más características importantes, luego se utilizara la técnica de Dropout para

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

permitir que la red evite el sobre aprendizaje y causar independencia en el aprendizaje de las neuronas.

Por último, en la capa de salida se designará 4 neuronas encargadas de la clasificación final del modelo.

3.3.4.2.Para modelo con arquitectura MobileNetV2.

Este modelo emplea la técnica de aprendizaje por transferencia, debido a que los datos que se tienen son muy pocos para entrenar una red robusta desde cero.

Para el modelo con arquitectura MobileNetV2, se tiene una red preentrenada con millones de parámetros, esta red ya sabe detectar bordes, tamaños, formas, texturas y patrones visuales.

Por tal caso se añade como modelo base esta red de la siguiente forma.

```
"# Modelo base
base_model = MobileNetV2(input_shape=(160, 160, 3), include_top=False, weights='imagenet')
base_model.trainable = False #Congelamos Las capas preentrenadas"
```

Como se puede observar para este modelo se congela las capas preentrenada, esto se hace porque no se quiere entrenar nuevamente la red sino aprovechar su conocimiento preentrenado, logrando así que se entrene únicamente la parte final del modelo, que es la encargada de darle la perspectiva de la tarea que se quiere realizar.

Para ejecutar el entrenamiento de la parte final del modelo se realiza un modelo secuencial que constara de lo siguiente.

```
"# Modelo final
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dropout(0.5),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(4, activation='softmax')  # MQ1, MQ2, NQ1, NQ2
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

])”

Como se puede ver contara con global average pooling, esto porque la salida de MobileNetV2 es un mapa de características 2D, con esta función se realiza la misma función que flatten, pero sin generar más parámetros. Se cuenta con dos aplicaciones de la técnica de optimización dropout al 50%, la primera se encargada de evitar que el modelo confié ciegamente en activaciones dominantes que son neuronas que se activan mucho sin importar la entrada, es como si el modelo requiriera de ellas para poder generalizar y la segunda con el fin de frenar u optimizar el riesgo de memorización. una capa densa con 256 neuronas, otro dropout al 50% y una capa de salida para clasificación de 4.

Figura 43

Resumen de todas las capas, salidas y números de parámetros a entrenar de una red con arquitectura MobileNetV2.

Model: "sequential"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_160 (Functional)	(None, 5, 5, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dropout (Dropout)	(None, 1280)	0
dense (Dense)	(None, 256)	327,936
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 4)	1,028

Total params: 2,586,948 (9.87 MB)

Trainable params: 328,964 (1.25 MB)

Non-trainable params: 2,257,984 (8.61 MB)

Nota. Elaboración Propia.

3.4. Subir rutas de entrenamiento, validación y pruebas.

Como se vio en la primera sección, previamente se realizó la creación de las carpetas necesarias para el entrenamiento, en esta etapa es el momento de subirlas al modelo para su entrenamiento. Para ello hay que agregar sus direcciones que se realizara con la siguiente línea de código.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

```
“data_entrenamiento = r"C:\PROYECTO DE TESIS\modelo\modelo\data_myn\entrenamiento" #direccion de la data de entrenamiento
data_validacion = r"C:\PROYECTO DE TESIS\modelo\modelo\data_myn\validacion" #direccion de la data de validacion”
```

Posteriormente se debe realizar un preprocesamiento de la data de entrada, es decir convertir las imágenes a vectores, para esto nos ayuda Keras, que aplicara las funciones de preprocesamiento adecuadas. Esto se realiza con la siguiente línea de código.

```
“train_datagen=ImageDataGenerator(preprocessing_function=preprocess_input) #conversion de imágenes a vectores
#aplicamos funciones de preprocesamiento de imágenes (son elegidas aleatoriamente por la librería keras)”
```

Luego procedemos a darle formato tanto a la data de entrenamiento como a la data de validación, recordando que el tamaño de entrada está estipulado como (160x160x3) lo cual quiere decir que es a color, se define un batch size que para el modelo será de 16, se recomienda que el batch size sea potencias de 2. Se define que la clasificación será “categorical” debido a que el modelo debe clasificar en categorías y además se enviarán los datos de forma desordenada, para que eso contribuya a la diversidad del entrenamiento, para ello activamos shuffle, todo ello se verá reflejado en las siguientes líneas de codificación.

Para modelo entrenado desde cero, se utilizan las siguientes líneas de código.

```
“train_generator=train_datagen.flow_from_directory(data_entrenamiento, # se da formato a la data de entrenamiento
                                                   target_size=(160,160),
                                                   color_mode='rgb',
                                                   batch_size=16,
                                                   class_mode='categorical',
                                                   shuffle=True)
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

```
validacion_datagen =
ImageDataGenerator(preprocessing_function=preprocess_input) #se da
formato a la data de validación

valid_generator = validacion_datagen.flow_from_directory(
   data_validacion,
   target_size=(160,160),
   batch_size=16,
   class_mode='categorical',
   shuffle=True)

print ("Informacion sobre las clases asignadas a cada clasificación")
print(train_generator.class_indices) “
```

Para modelo con arquitectura MobileNetV2, se utilizan las siguientes líneas de código.

```
“# Preprocesamiento de datos
train_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
val_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator = train_datagen.flow_from_directory(
   'C:/PROYECTO DE TESIS/modelo/modelo/data_myn/entrenamiento',
   target_size=(160, 160),
   batch_size=16,
   class_mode='categorical',
   shuffle=True
)

valid_generator = val_datagen.flow_from_directory(
   'C:/PROYECTO DE TESIS/modelo/modelo/data_myn/validacion',
   target_size=(160, 160),
   batch_size=16,
   class_mode='categorical',
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

```
    shuffle=False
)”
```

Después se deberá indicar cuantas rondas de entrenamiento se otorgará al modelo, para el caso de la red entrenada desde cero se tomará un total de 8 rondas y para el modelo con arquitectura MobileNetV2 un total de 10 rondas. esto se indica en la siguiente línea de código.

Para modelo entrenado desde cero se utiliza la siguiente línea de código.

```
“TOTAL_EPOCAS_TRAINING= 8 #total de rondas de entrenamiento”
```

Para modelo con arquitectura MobileNetV2, se utiliza la siguiente línea de código.

```
“# Entrenamiento
TOTAL_EPOCAS = 10”
```

Luego se contemplará en ambos casos de modelos, la técnica de entrenamiento early stopping, esto para evitar el sobreajuste, su funcionamiento es simple, si el modelo al pasar de las rondas no logra mejorar, el entrenamiento se detiene y se recupera la época con la mejor tasa de entrenamiento, en el caso de ambos el early stopping trabajara con una tolerancia de 5 rondas.

```
“# Early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)”
```

Posterior a ello y para el caso del entrenamiento desde cero, se utiliza el optimizador Adam para lograr la reducción del error, esta ajusta los valores de gradiente de forma automática, en la función de perdida utilizamos categorical_crossentropy debido a que el modelo deberá clasificar, y se colocara en el apartado métricas, accuracy, para que en el momento del entrenamiento se muestre la exactitud que se va generando en cada época del entrenamiento, todo ello se muestra en la siguiente línea de código.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

“model.compile(optimizer='Adam',loss='categorical_crossentropy',metrics=['accuracy'])
#categorical_crossentropy porque clasificamos CLASES de imagenes #ADAM el mejor optimizador, el mas estable.”

Para el caso del modelo con MobileNetV2, no es necesario agregar el optimizador, debido a que la red ya se encuentra previamente entrenada.

Para ambos modelos y para poder asegurar que el entrenamiento genere pasos seguros se realiza una división entre el tamaño de la data de entrenamiento y el tamaño de la data de validación, dividido entre el batch size, con esto nos aseguramos de que no haya perdida de datos es decir que todas las imágenes se entrenen las veces que sea necesarias para mejorar su aprendizaje. Todo esto reflejado en las siguientes líneas de código.

“#tamaños de pasos por epoca.
step_size_train=train_generator.n//train_generator.batch_size
step_size_valid=valid_generator.n//valid_generator.batch_size”

Luego se ejecutará el algoritmo backpropagation para el entrenamiento, para ello se entrega ciertos parámetros que ya están previamente contemplados, sin embargo, se puede ver cuáles son estos en las siguientes líneas de código.

“#se ejecuta el algoritmo de entrenamiento backpropagation
H = model.fit(train_generator,
steps_per_epoch=step_size_train,
epochs=TOTAL_EPOCAS_TRAINING,
validation_data=valid_generator,
validation_steps=step_size_valid)”

Posteriormente se muestra el inicio del entrenamiento, sus épocas, perdidas, exactitud entre otros datos.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 44

Proceso de entrenamiento por épocas para modelo entrenado desde cero.

```
Epoch 1/8
135/135 ━━━━━━━━━━━━━━━━━━━━ 61s 443ms/step - accuracy: 0.6696 - loss: 0.8422 - val_accuracy: 0.5387 - val_loss: 0.9443
Epoch 2/8
135/135 ━━━━━━━━━━━━━━━━━━━━ 5s 33ms/step - accuracy: 1.0000 - loss: 0.1302 - val_accuracy: 0.5595 - val_loss: 0.9238
Epoch 3/8
135/135 ━━━━━━━━━━━━━━━━━━━━ 53s 394ms/step - accuracy: 0.9140 - loss: 0.2471 - val_accuracy: 0.5893 - val_loss: 0.8317
Epoch 4/8
135/135 ━━━━━━━━━━━━━━━━━━━━ 4s 31ms/step - accuracy: 0.9375 - loss: 0.1095 - val_accuracy: 0.5893 - val_loss: 0.8085
Epoch 5/8
135/135 ━━━━━━━━━━━━━━━━━━━━ 49s 362ms/step - accuracy: 0.9450 - loss: 0.1489 - val_accuracy: 0.5655 - val_loss: 0.8906
Epoch 6/8
135/135 ━━━━━━━━━━━━━━━━━━━━ 4s 30ms/step - accuracy: 1.0000 - loss: 0.0842 - val_accuracy: 0.5655 - val_loss: 0.9163
Epoch 7/8
135/135 ━━━━━━━━━━━━━━━━━━━━ 49s 366ms/step - accuracy: 0.9608 - loss: 0.1105 - val_accuracy: 0.5893 - val_loss: 1.2522
Epoch 8/8
135/135 ━━━━━━━━━━━━━━━━━━━━ 4s 30ms/step - accuracy: 1.0000 - loss: 0.0420 - val_accuracy: 0.5893 - val_loss: 1.2848
```

```
model.evaluate(train_generator, steps=train_generator.n // train_generator.batch_size)
#evaluamos el modelo de manera general, obtenemos: (error_general,accuracy_general)
```

```
135/135 ━━━━━━━━━━━━━━━━━━━━ 25s 184ms/step - accuracy: 0.9865 - loss: 0.0361
[0.03583914041519165, 0.9888888597488403]
```

Nota. Elaboración propia.

Figura 45

Proceso de entrenamiento por épocas para modelo con arquitectura MobileNetV2.

```
Epoch 1/10
136/136 ━━━━━━━━━━━━━━━━━━━━ 60s 407ms/step - accuracy: 0.7148 - loss: 0.9045 - val_accuracy: 0.5797 - val_loss: 0.8165
Epoch 2/10
136/136 ━━━━━━━━━━━━━━━━━━━━ 48s 353ms/step - accuracy: 0.8867 - loss: 0.2538 - val_accuracy: 0.6435 - val_loss: 0.7029
Epoch 3/10
136/136 ━━━━━━━━━━━━━━━━━━━━ 48s 351ms/step - accuracy: 0.9255 - loss: 0.1868 - val_accuracy: 0.6551 - val_loss: 0.6886
Epoch 4/10
136/136 ━━━━━━━━━━━━━━━━━━━━ 49s 362ms/step - accuracy: 0.9334 - loss: 0.1738 - val_accuracy: 0.6928 - val_loss: 0.6687
Epoch 5/10
136/136 ━━━━━━━━━━━━━━━━━━━━ 48s 353ms/step - accuracy: 0.9451 - loss: 0.1433 - val_accuracy: 0.6522 - val_loss: 0.7247
Epoch 6/10
136/136 ━━━━━━━━━━━━━━━━━━━━ 55s 407ms/step - accuracy: 0.9580 - loss: 0.1125 - val_accuracy: 0.6609 - val_loss: 0.9151
Epoch 7/10
136/136 ━━━━━━━━━━━━━━━━━━━━ 60s 444ms/step - accuracy: 0.9543 - loss: 0.1141 - val_accuracy: 0.6928 - val_loss: 0.8029
Epoch 8/10
136/136 ━━━━━━━━━━━━━━━━━━━━ 58s 425ms/step - accuracy: 0.9555 - loss: 0.1124 - val_accuracy: 0.7043 - val_loss: 0.7599
Epoch 9/10
136/136 ━━━━━━━━━━━━━━━━━━━━ 53s 389ms/step - accuracy: 0.9625 - loss: 0.1094 - val_accuracy: 0.6667 - val_loss: 0.9362
```

```
# Evaluación final
model.evaluate(train_generator, steps=step_train)
```

```
136/136 ━━━━━━━━━━━━━━━━━━━━ 50s 369ms/step - accuracy: 0.9753 - loss: 0.0694
[0.06559791415929794, 0.978733241558075]
```

Nota. Elaboración propia.

Luego se evalúa el modelo de forma general con esto logramos obtener el error general y el accuracy general.

Figura 46

Evaluación de error y accuracy general para modelo entrenado de cero.

```
model.evaluate(train_generator, steps=train_generator.n // train_generator.batch_size)
#evaluamos el modelo de manera general, obtenemos: (error_general,accuracy_general)
```

```
135/135 ━━━━━━━━━━━━━━━━━━━━ 25s 184ms/step - accuracy: 0.9865 - loss: 0.0361
[0.03583914041519165, 0.9888888597488403]
```

Nota. Elaboración Propia

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 47

Evaluación de error y accuracy general para modelo con arquitectura MobileNetV2.

```
# Evaluación final
model.evaluate(train_generator, steps=step_train)
```

```
136/136 ━━━━━━━━━━━━━━━━━━━━ 50s 369ms/step - accuracy: 0.9753 - loss: 0.0694
[0.06559791415929794, 0.978733241558075]
```

Nota. Elaboración propia.

Como se puede observar, en ambos casos el error general fue bajo y el accuracy general fue alto.

Esto indica que el modelo tiene una precisión elevada y un error general bajo, sin embargo, es necesario graficar el error y la exactitud, y los datos de perdida, para ver si el modelo esta sobre aprendiendo, para ello utilizamos la siguiente línea de código.

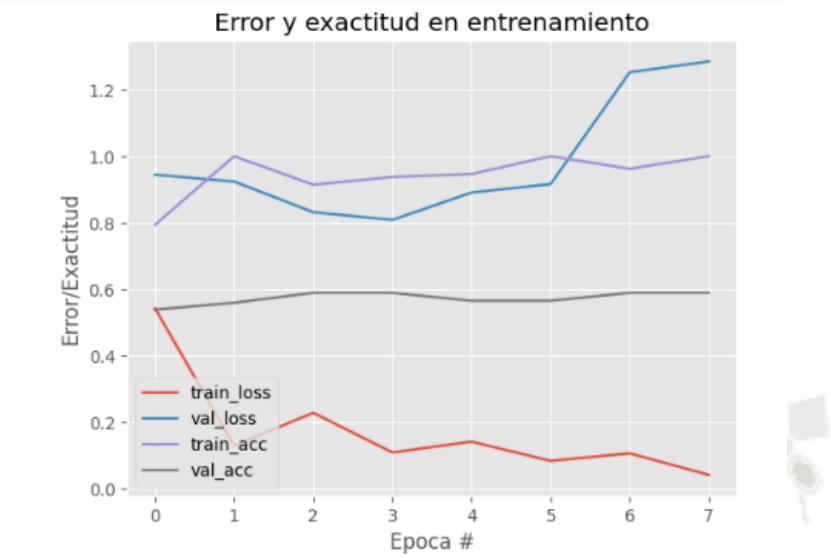
“N = TOTAL_EPOCAS
plt.style.use("ggplot")
plt.figure()
plt.plot(H.history["loss"], label="train_loss")
plt.plot(H.history["val_loss"], label="val_loss")
plt.plot(H.history["accuracy"], label="train_acc")
plt.plot(H.history["val_accuracy"], label="val_acc")
plt.title("Error y exactitud en entrenamiento")
plt.xlabel("Epoca #")
plt.ylabel("Error/Exactitud")
plt.legend(loc="lower left")
plt.savefig("C:/PROYECTO DE TESIS/modelo/modelo/nativa/myn/cnn_estadisticas_de_la_red_mobilenetv2.png")
”

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 48

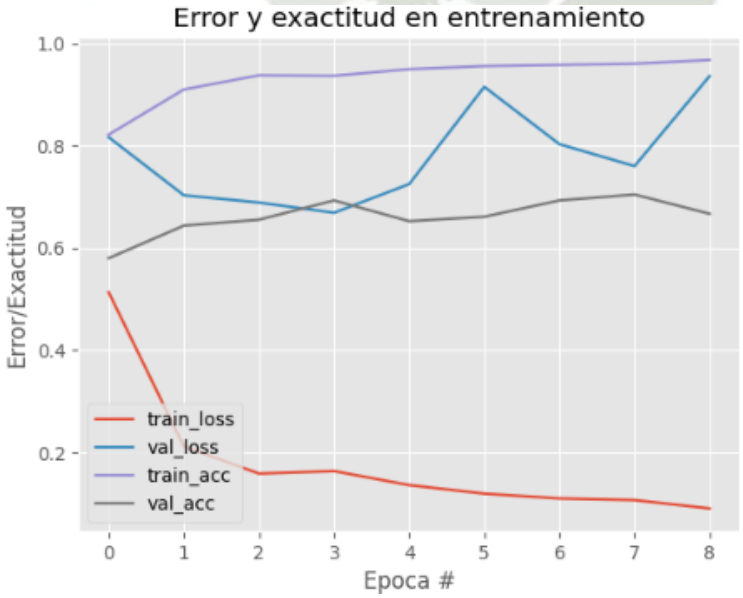
Gráfico de parámetros importantes para evaluación de modelo entrenado desde cero.



Nota. Elaboración propia.

Figura 49

Gráfico de parámetros importantes para evaluación de modelo con arquitectura MobileNetV2.



Nota. Elaboración propia.

En las imágenes se puede observar el eje "x" que contempla el número de épocas para el entrenamiento y en el eje "y" el error y la exactitud.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Como se puede ver en la línea roja que representa la función de perdida en los datos de entrenamiento (train_loss), en la época 0, para ambos modelos, el error es alto debido a que la red no aprendió nada aun, sin embargo, a lo largo del entrenamiento esto va disminuyendo, quedando la función en valores bajos, representando así que el modelo está aprendiendo bien los datos de entrenamiento, sin embargo existe una clara diferencia en ambos modelos, en el modelo desde cero, tiene un aprendizaje muy abrupto, a diferencia del aprendizaje a lo largo de las épocas del modelo con MobileNetV2, lo cual nos indica que el modelo entrenado desde cero está adaptándose muy rápido a los datos de entrenamiento, lo cual puede ser una señal de memorización, y esto tiene más sentido debido a que esto suele suceder cuando se hace un entrenamiento para una zona fija, es decir los datos no varían muchos unos de otros, las frutas de un mismo tipo casi siempre muestran rasgos similares, las formas y el fondo casi siempre se repiten. En contraparte, el modelo con arquitectura MobileNetV2 al tener un train_loss más controlado, nos indica que el modelo le cuesta más el proceso de adaptación y aprendizaje, lo cual podría indicar que existe una mejor generalización, esta ventaja se la otorga el entrenamiento previo con el que viene la red MobileNetV2, debido a que solo ajusta lo necesario y no tiende a memorizar.

La línea azul hace representación a la función de perdida en los datos de validación (val_loss), se toma de referencia val_loss para determinar cómo va la habilidad de generalización del modelo al pasar de las épocas, lo ideal para val_loss es que se mantenga bajo, un claro indicativo de sobreajuste se da cuando se ve que train_loss baja pero val_loss sube, porque esto indica que el modelo se vuelve muy preciso con los datos que ya conoce, pero cuando ve datos nuevos no es capaz de diferenciarlos ni clasificarlos de manera óptima. Como se puede ver en ambos gráficos, el modelo con MobileNetV2, si bien también llega a generar un sobreajuste a partir de la época 3, se mantiene con valores más bajos que los del modelo entrenado desde cero.

Los parámetros train accuracy y val accuracy necesitan ser evaluados de la mano, estos en conjunto nos pueden otorgar información importante para el modelo.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Tabla 9

Parámetros train accuracy vs val accuracy

Train_Acc	Val_Acc	Que representa	diagnostico
Alto	Alto	Modelo generaliza y aprende bien	Modelo optimo
Alto	Bajo	Modelo memoriza y no logra generalizar	Overfitting.
Baja	Baja	Modelo sin aprendizaje	Underfitting.

Nota. Elaboración propia.

Una consideración importante es que si se tiene un dato de val_acc aceptable pero un val_loss alto, significa que el modelo acierta en sus predicciones, pero con muy poca confianza.

Observando la gráfica en ambos casos se puede concluir que, para el modelo desde cero, se tiene un modelo de aprendizaje rápido, pero con un val_Acc estancado, es decir el modelo memoriza, pero no logra generalizar bien, cae en sobreajuste.

Para la red con MobileNetV2, se tiene un modelo que tiene un aprendizaje lento, pero más seguro, un val_Acc mejor al del modelo desde cero y con menos rondas de entrenamiento, lo cual nos indica que el modelo logra generalizar mejor.

Una vez realizado el entrenamiento en ambos modelos, se debe probar la capacidad de generalización, para ello se hace uso de la subcarpeta “test_images” dentro de la carpeta “pruebas” que se creó previamente.

Se debe tener en cuenta que el formato de ingreso de las imágenes debe ser igual a las presentadas en entrenamiento y validación es decir 160x160 para ello previo a cargarlas se debe preprocesar estas imágenes antes.

Para ello incluimos las siguientes líneas de código.

```
“#UNA VEZ ENTRENADA LA RED SE PRUEBA LA EFECTIVIDAD DE ESTA
# damos formato a la data de pruebas
test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
test_generator = test_datagen.flow_from_directory(
    directory="C:/PROYECTO DE TESIS/modelo/modelo/data_myn/pruebas/",
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
    target_size=(160, 160),
    color_mode="rgb",
    batch_size=1,
    class_mode=None,
    shuffle=False,
    seed=42
)"
```

En esta sección del código damos el formato de entrada necesario a las imágenes de prueba, se indica que las imágenes ingresaran una por una además de indicar que se ingresaran en orden.

Posteriormente estas imágenes deben pasar a las pruebas del modelo una imagen a la vez.

Es necesario recalcar que las imágenes que se encuentren en esta carpeta deben ser imágenes que el modelo no haya visto ni en su data de entrenamiento ni en la data de validación, debido a que en esta parte probaremos la capacidad de generalización del modelo frente al área donde realizara su función.

```
"STEP_SIZE_TEST=test_generator.n//test_generator.batch_size
test_generator.reset()
#llamamos a la función de predicción, aplicamos elemento a elemento, se envía data de 1 en 1.
pred = model.predict(test_generator, steps=STEP_SIZE_TEST, verbose=1)
#calculamos el array de predicción"
```

Se realiza la predicción del modelo con los datos de prueba, para los resultados es recomendable visualizar su predicción en tipo flotante, porque de esta manera podremos observar decimales que serán importantes para detectar si el modelo duda en sus predicciones o está seguro de lo que ve.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 50

Array de predicciones tipo flotante para modelo entrenado desde cero.

```
pred.round(2) #array de predicciones-tipo flotante
```

```
array([[0.08, 0.92, 0.  , 0.  ],
       [0.69, 0.31, 0.  , 0.  ],
       [0.9 , 0.1 , 0.  , 0.  ],
       [0.97, 0.03, 0.  , 0.  ],
       [0.1 , 0.35, 0.42, 0.13],
       [0.18, 0.81, 0.01, 0.  ],
       [0.88, 0.12, 0.  , 0.  ],
       [0.66, 0.34, 0.  , 0.  ],
       [0.67, 0.33, 0.  , 0.  ],
       [0.5 , 0.5 , 0.  , 0.  ],
       [0.58, 0.42, 0.  , 0.  ],
       [0.77, 0.23, 0.  , 0.  ],
       [0.84, 0.16, 0.  , 0.  ],
       [0.69, 0.31, 0.  , 0.  ],
       [0.56, 0.44, 0.  , 0.  ],
       [0.44, 0.56, 0.  , 0.  ],
       [0.02, 0.98, 0.  , 0.  ],
       [0.03, 0.97, 0.  , 0.  ],
       [0.03, 0.97, 0.  , 0.  ],
       [0.  , 0.  , 0.99, 0.01],
       [0.  , 0.  , 0.98, 0.02],
       [0.  , 0.  , 1.  , 0.  ],
       [0.  , 0.  , 0.98, 0.02],
       [0.  , 0.03, 0.23, 0.74],
       [0.01, 0.  , 0.97, 0.02],
       [0.97, 0.03, 0.  , 0.  ],
       [0.54, 0.01, 0.41, 0.03],
       [0.  , 0.  , 0.78, 0.22],
       [0.  , 0.  , 0.99, 0.01],
       [0.  , 0.  , 0.9 , 0.1 ],
       [0.  , 0.  , 0.93, 0.07],
       [0.  , 0.08, 0.08, 0.84],
       [0.  , 0.02, 0.  , 0.98],
       [0.  , 0.  , 0.  , 1.  ],
       [0.  , 0.  , 0.  , 1.  ],
       [0.  , 0.  , 0.01, 0.99],
       [0.  , 0.  , 0.96, 0.04],
       [0.  , 0.  , 1.  , 0.  ],
       [0.  , 0.  , 0.98, 0.02]], dtype=float32)
```

Nota. Elaboración propia.

Figura 51

Array de predicciones para modelo con arquitectura MobileNetV2.

```
pred.round(2) #array de predicciones-tipo flotante
```

```
array([[0.02, 0.98, 0.  , 0.  ],
       [0.57, 0.43, 0.  , 0.  ],
       [0.9 , 0.1 , 0.  , 0.  ],
       [0.95, 0.05, 0.  , 0.  ],
       [0.08, 0.92, 0.  , 0.  ],
       [0.01, 0.99, 0.  , 0.  ],
       [0.2 , 0.8 , 0.  , 0.  ],
       [0.73, 0.27, 0.  , 0.  ],
       [0.83, 0.17, 0.  , 0.  ],
       [0.47, 0.53, 0.  , 0.  ],
       [0.57, 0.43, 0.  , 0.  ],
       [0.69, 0.31, 0.  , 0.  ],
       [0.69, 0.31, 0.  , 0.  ],
       [0.83, 0.17, 0.  , 0.  ],
       [0.41, 0.59, 0.  , 0.  ],
       [0.33, 0.67, 0.  , 0.  ],
       [0.  , 1.  , 0.  , 0.  ],
       [0.01, 0.99, 0.  , 0.  ],
       [0.01, 0.99, 0.  , 0.  ],
       [0.02, 0.03, 0.58, 0.37],
       [0.  , 0.  , 0.94, 0.06],
       [0.  , 0.  , 0.93, 0.07],
       [0.  , 0.  , 0.58, 0.42],
       [0.09, 0.19, 0.01, 0.71],
       [0.07, 0.58, 0.04, 0.32],
       [0.  , 0.  , 0.  , 1.  ],
       [0.01, 0.02, 0.27, 0.7 ],
       [0.  , 0.  , 0.35, 0.65],
       [0.  , 0.  , 0.12, 0.88],
       [0.  , 0.  , 0.94, 0.06],
       [0.  , 0.  , 0.97, 0.03],
       [0.  , 0.  , 0.11, 0.89],
       [0.  , 0.01, 0.21, 0.79],
       [0.  , 0.  , 0.27, 0.73],
       [0.  , 0.  , 0.06, 0.93],
       [0.  , 0.  , 0.03, 0.97],
       [0.  , 0.  , 0.92, 0.08],
       [0.  , 0.  , 0.37, 0.63],
       [0.  , 0.  , 0.36, 0.64]], dtype=float32)
```

Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Aquí se puede hacer una comparativa del desempeño de ambos modelos, en donde se ve que el modelo entrenado desde cero presenta un porcentaje más elevado de seguridad en sus predicciones, sin embargo, es necesario observar la matriz de confusión para poder dar el diagnostico completo de lo que se ve.

Para poder determinar la capacidad de generalización del testeo, es necesario que mostremos los resultados de la clasificación en tipo entero.

Figura 52

Código y resultado de matriz tipo entero.

```
predicted_class_indices=np.argmax(pred,axis=1) #lista de predicciones en tipo entero
```

```
print (predicted_class_indices) #mostramos lista de predicciones
print(type(predicted_class_indices))
```

```
[1 0 0 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 2 2 2 2 3 1 3 3 3 3 2 2 3 3 3 3 3 2
 3 3]
```

Nota. Elaboración propia.

Una vez obtenida la matriz de predicción de tipo entero, para lograr un entendimiento mayor de lo que sucede, se cargara las imágenes que entraron como dataset de pruebas, con sus predicciones, para ello utilizamos las siguientes líneas de código.

```
"# Obtenemos los índices de las clases del generador de datos de entrenamiento (train_generator).
labels = (train_generator.class_indices)
labels = dict((v, k) for k, v in labels.items())
predictions = [labels[k] for k in predicted_class_indices]
#mostramos las imágenes de prueba con su predicción
folder=r"C:\PROYECTODETESIS\modelo\modelo\data_myn\pruebas\test_images"
# Obtener la lista de archivos
filenames = os.listdir(folder)
# Configurar la figura
fig = plt.figure(figsize=(30, 30))
fig.subplots_adjust(hspace=0.1, wspace=0.1)
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

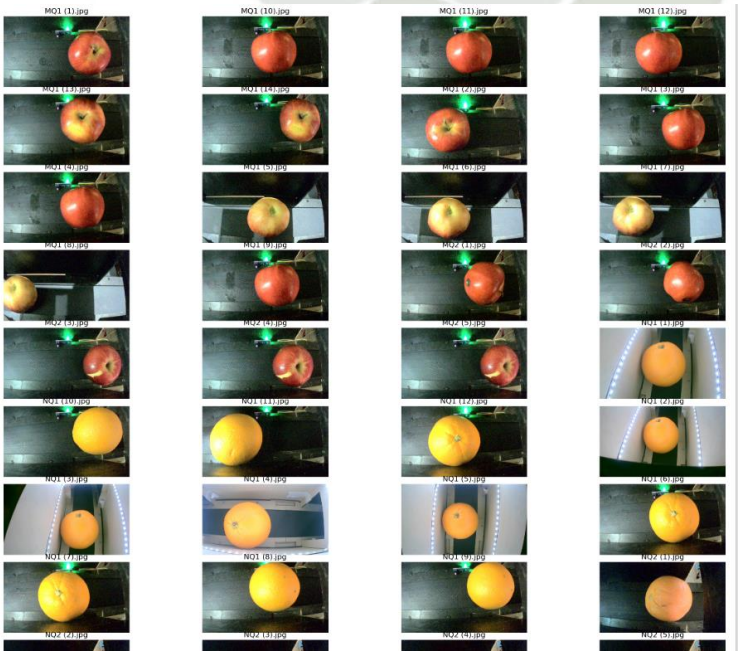
```
rows = 10
cols = len(filenames) // rows
if len(filenames) % rows > 0:
    cols += 1

# Mostrar imágenes
for i, filename in enumerate(filenames):
    img_path = os.path.join(folder, filename)  # Construir ruta completa
    img = cv2.imread(img_path)

    if img is not None:
        ax = fig.add_subplot(rows, cols, i + 1)
        ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))  # Convertir de BGR a RGB
        ax.axis("off")
        ax.set_title(filename, fontsize=16)”
```

Figura 53

Muestra de data de entrenamiento con índice de clase.



Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

En ellas podremos observar que es lo que debería haber clasificado el modelo y se comparara con la matriz de predicciones tipo entero.

De manera que de cada imagen cargada se extraerá la clase deseada.

```
“my_dict = {
    'MQ1': 0,
    'MQ2': 1,
    'NQ1': 2,
    'NQ2': 3,

}
real_class_indices=[]
for filename in filenames:
    # Obtenemos el nombre del archivo
    basename = filename.split(os.sep)[-1].lower()

    found_class = None

    # Recorremos cada estado y su índice
    for key, val in my_dict.items():
        if key.lower() in basename:
            found_class = val
            break  # No hace falta seguir buscando

    # Guardamos la clase que encontramos
    if found_class is not None:
        real_class_indices.append(found_class)
    else:
        # si alguna predicción no hace match se le pondrá -1
        real_class_indices.append(-1)

print(real_class_indices)
real_class_indices = np.array(real_class_indices)”
```

Figura 53.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Array de clasificación esperada según data de prueba.

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3]
```

Nota. Elaboración propia.

Figura 54.

Array de clasificación obtenida después de procesamiento de data de prueba.

```
predicted_class_indices
```

```
array([1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2,
       2, 3, 1, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 2, 3, 3])
```

Nota. Elaboración propia.

Estos datos son necesarios porque son los que ingresaran para realizar la matriz de confusión, para ello se realiza las siguientes líneas de código.

```
“cm = confusion_matrix(real_class_indices, predicted_class_indices)
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Matriz de confusión normalizada")
    else:
        print('Matriz de confusión, sin normalización')

    print(cm)

    thresh = cm.max() / 2.
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

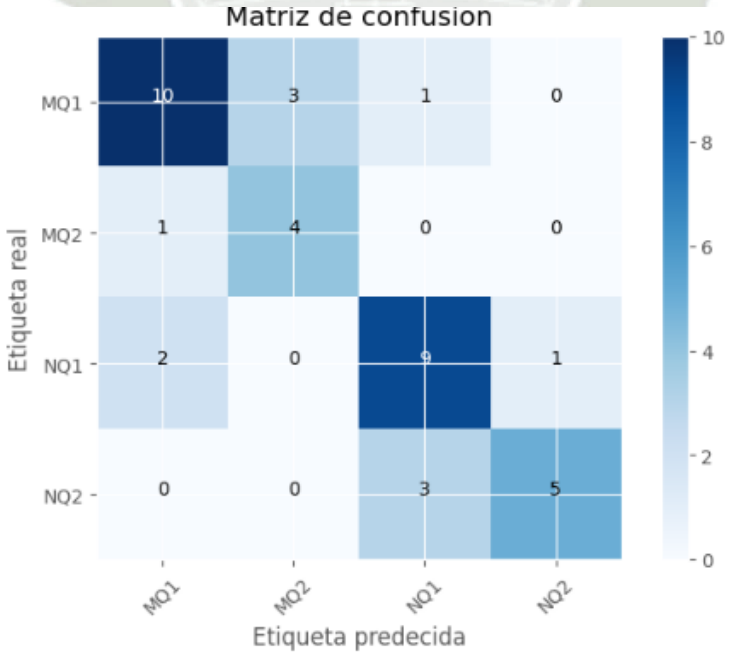
```
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j],
             horizontalalignment="center",
             color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()
plt.ylabel('Etiqueta real')
plt.xlabel('Etiqueta predecida')
cm_plot_labels = train_generator.class_indices
print (cm_plot_labels)
key_list = list(cm_plot_labels.keys())
val_list = list(cm_plot_labels.values())
plot_confusion_matrix(cm, cm_plot_labels, title= 'Matriz de confusion')"
```

Los resultados obtenidos para ambos casos fueron los siguientes.

Figura 54

Matriz de confusión de modelo entrenado desde cero.



Nota. Elaboración propia.

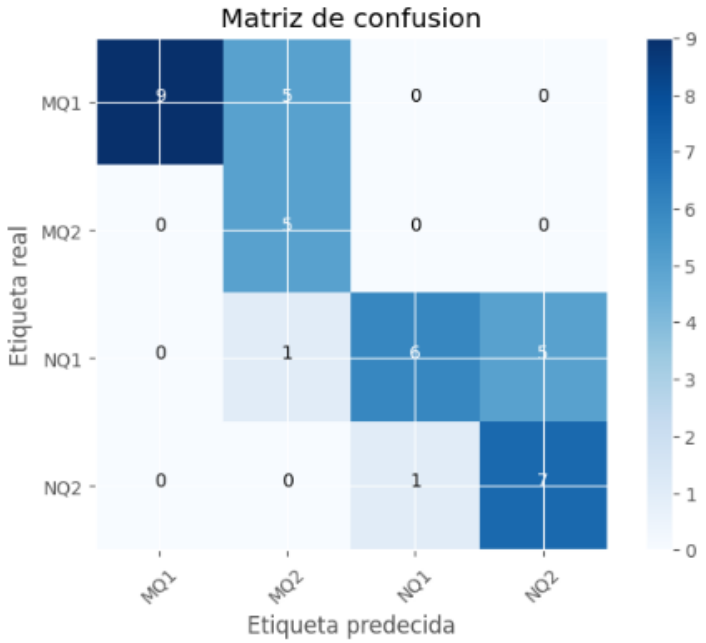
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 55

Matriz de confusión para modelo con arquitectura MobileNetV2.



Nota. Elaboración propia.

Como se vio en la matriz con datos tipo flotante, el modelo entrenado desde cero, tenía más tasa de certeza en sus predicciones, sin embargo posee muchos erros como confusión de calidad de frutas, pero lo grave es que se equivoca con mucha seguridad cuando clasifica naranjas, y fue en 2 casos diferentes en las que aseguro de que la manzana que está viendo era una naranja, esto descarta la red entrenada desde cero para este tipo de aplicaciones en la que los datos son reducidos, debido a que el modelo requiere de más cantidad de datos y más calidad de imágenes para realizar una generalización correcta.

La decisión de usar el modelo entrenado con la arquitectura MobileNetV2, fue debido a que el modelo si bien presenta fallos en la clasificación, solo confundió una sola fruta de todas las que entraron en su data de prueba y si bien su taza de seguridad no era muy alta, su tasa de error era muy baja cuando se trataba de clasificar el tipo de fruta que veía y para este tipo de aplicación un error muy grave es confundir una manzana con una naranja en más de una ocasión.

Para saber con certeza el desempeño del modelo se realiza el cálculo de su error y exactitud, para ello utilizamos la función scoring y la librería sklearn. metrics para obtener un reporte más detallado de la clasificación.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 56

Código y resultado del desempeño Para modelo entrenado desde cero.

```
[27]: from mlxtend.evaluate import scoring #incluye diversas funciones para evaluar el rendimiento de los modelos.

      res = scoring(y_target=real_class_indices, y_predicted=predicted_class_indices, metric='error')

      print('Error:',np.round(res * 100,2),"%")

      res = scoring(y_target=real_class_indices, y_predicted=predicted_class_indices, metric='accuracy')
      print ("Exactitud:",np.round(res*100,2),"%")

Error: 28.21 %
Exactitud: 71.79 %

[28]: from sklearn.metrics import classification_report #generar reporte que muestra las principales métricas de clasificación
      print(classification_report(predicted_class_indices,real_class_indices))

              precision    recall  f1-score   support

           0       0.71      0.77      0.74        13
           1       0.80      0.57      0.67         7
           2       0.75      0.69      0.72        13
           3       0.62      0.83      0.71         6

    accuracy                           0.72        39
   macro avg       0.72      0.72      0.71        39
weighted avg       0.73      0.72      0.72        39
```

Nota. Elaboración propia.

Figura 57

Código y resultado del desempeño para modelo con arquitectura MobileNetV2.

```
[27]: from mlxtend.evaluate import scoring #incluye diversas funciones para evaluar el rendimiento de los modelos.

      res = scoring(y_target=real_class_indices, y_predicted=predicted_class_indices, metric='error')

      print('Error:',np.round(res * 100,2),"%")

      res = scoring(y_target=real_class_indices, y_predicted=predicted_class_indices, metric='accuracy')
      print ("Exactitud:",np.round(res*100,2),"%")

Error: 30.77 %
Exactitud: 69.23 %

[28]: from sklearn.metrics import classification_report #generar reporte que muestra las principales métricas de clasificac
      print(classification_report(predicted_class_indices,real_class_indices))

              precision    recall  f1-score   support

           0       0.64      1.00      0.78         9
           1       1.00      0.45      0.62        11
           2       0.50      0.86      0.63         7
           3       0.88      0.58      0.70        12

    accuracy                           0.69        39
   macro avg       0.75      0.72      0.68        39
weighted avg       0.79      0.69      0.69        39
```

Nota. Elaboración propia.

En las métricas podemos darnos cuenta de que el modelo entrenado desde 0 tiene mejores puntuaciones con respecto al modelo con arquitectura MobileNetV2, y de hecho se ve que en la clase 0 que son manzanas en buen estado, tuvo una precisión mayor.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

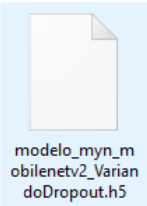
Sin embargo, para este tipo de aplicaciones lo que se requiere es que la generalización sea lo menos errática posible, en un ejemplo práctico donde el proceso está viendo una manzana, es preferible que el modelo tenga 60% de certeza de que está viendo una manzana a tener 90% de certeza de que viendo la manzana diga que es una naranja.

Finalmente, para ambos modelos, se guarda el modelo .h5 necesario para la implementación del modelo con la parte física, para ello se utiliza la siguiente línea de código.

"model.save(r'C:\PROYECTODETESIS\modelo\modelo\nativa\myn\modelo_myn_mobilenetv2_VariandoDropout.h5')"

Figura 58

Archivo h.5.



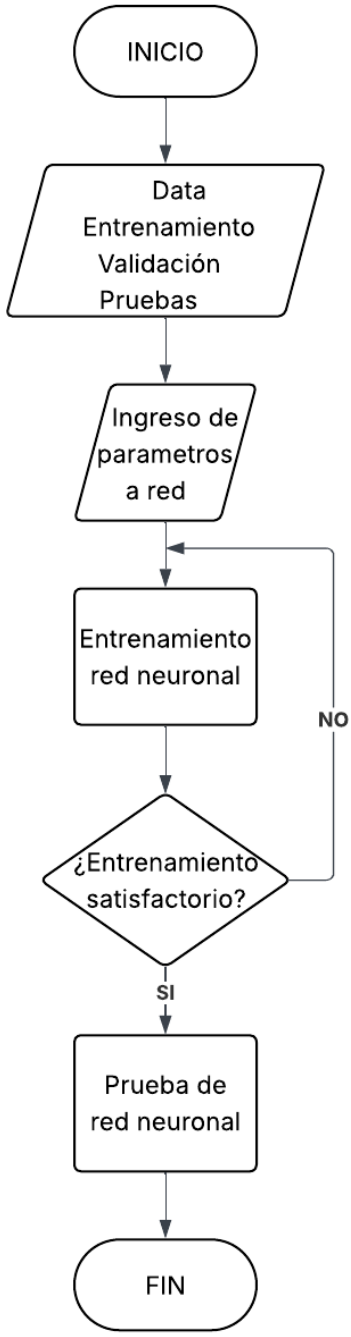
Nota. elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 59

Diagrama de flujo de entrenamiento de la red neuronal



Nota. Elaboración Propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

3.5. Implementación en Arduino

Para la sección de codificación en Arduino, nos guiaremos con el diagrama de conexión (anexo 5), para tener una mayor idea de lo que se está realizando.

3.5.1. Lógica de control

Para el inicio del sistema se activan los pines de los sensores infrarrojos, el servomotor se lleva a la posición 70° y se activa el motor eléctrico para el movimiento de la faja.

El proceso recibirá de forma manual la fruta, una vez esta llegue al sensor IR1, el proceso se conectará con el modelo y la programación en spyder para realizar la clasificación, IR1 mandará una señal Cam_on la cual activará la clasificación. Una vez realizado este proceso, el sensor IR1 quedara inhabilitado además de enviar una señal de Cam_off al modelo en spyder, estado que permanecerá hasta que se cumpla cualquiera de las siguientes dos condiciones.

Si el modelo llegara a clasificar frutas en buen estado, el proceso seguirá con la activación de la faja, la fruta al pasar por el sensor IR2 activara este para que la señal que envíe vuelva a habilitar la detección de frutas.

Si el modelo clasifica frutas en mal estado, el proceso seguirá hasta llegar al IR2, el cual, al detectar la presencia, mandará una señal de pare de faja, que la detendrá en 0.5 segundos después de la detección, en este punto el servomotor se moverá de la posición 70° hacia los 180° para lograr realizar la eliminación de la fruta en mal estado del proceso, luego de ello volverá a los 70° para esperar la siguiente clasificación de frutas en mal estado.

El sensor IR2 siempre será el encargado de habilitar el sensor IR1.

3.5.2. Pines de conexión

Para el modelo implementado, se utilizó un Arduino mega 2560 para el cual se utilizaron las siguientes salidas.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Tabla 10

Distribución de conexión en Arduino

Conexión en Arduino	Destino
Digital 31	Relé de 5v para control On/Off del motor eléctrico.
Digital 53	Señal desde sensor infrarrojo IR1
Digital 52	Señal desde sensor infrarrojo IR2
PWM 04	Control de Servomotor.

Nota. Elaboración propia.

3.5.3. Codificación Arduino IDE

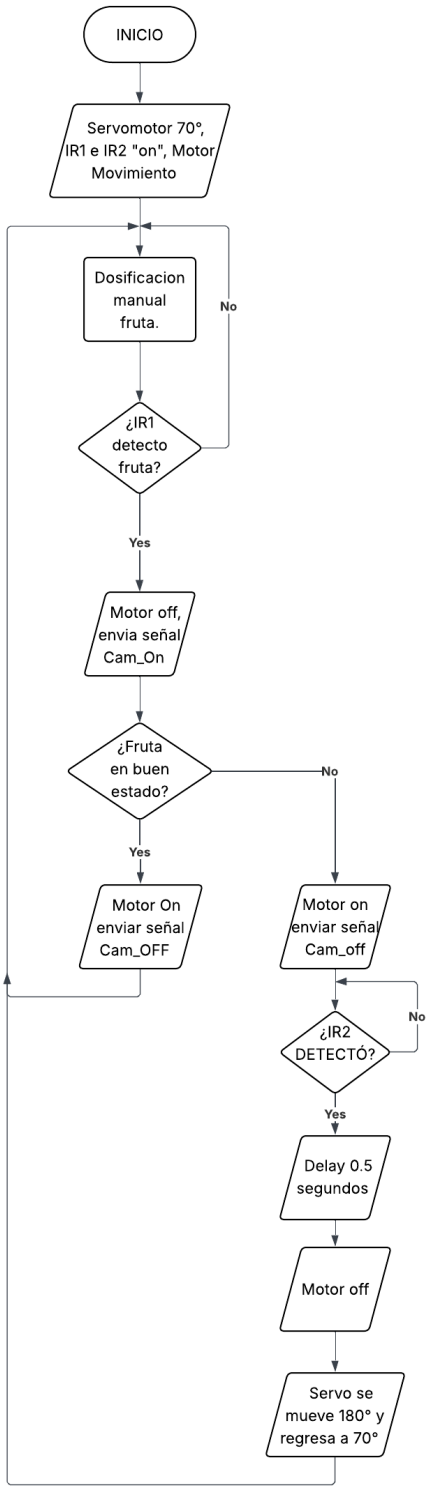
Para poder observar la codificación en Arduino IDE, se revisará el ANEXO 07.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 60

Diagrama de Flujo Codificación Arduino



Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

3.6. Implementación en Spyder.

Para poder realizar la prueba del modelo entrenado, se realizará la codificación en Spyder correspondiente.

3.6.1. Importe de librerías

```
“import cv2
import numpy as np
from keras.models import load_model
from keras.applications.mobilenet import preprocess_input
import time
import serial
import os”
```

Utilizando:

- Cv2: Para poder realizar la visión artificial
- Numpy: Nos permite lograr el procesamiento numérico.
- Serial: Permite la comunicación de Spyder con Arduino mediante el puerto serial.
- Time: Para manejo de tiempo
- Keras y mobilenet: Nos permite la carga de modelos y el procesamiento de las imágenes.

3.6.2. Configuración Inicial

Una vez importadas las librerías comenzamos con la configuración inicial, en donde se establecerá la comunicación con el Arduino, se cargará el modelo entrenado, se determinará las clases y el con que se pintará el “roi” en cada una de las clasificaciones, todo ello se representa en las siguientes líneas de código.

```
“# ------------------ CONFIGURACIÓN INICIAL ------------------
arduino = serial.Serial('COM3', 9600, timeout=1)
time.sleep(2)

ruta_modelo=r'C:\PROYECTODETESIS\modelo\modelo\nativa\myn\modelo_myn_
mobilenetv2_VariandoDropout.h5'
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

```
modelo = load_model(ruta_modelo)

clases = ['MQ1', 'MQ2', 'NQ1', 'NQ2']
colores = {
    'MQ1': (255, 0, 0), 'MQ2': (0, 255, 0),
    'NQ1': (255, 255, 0), 'NQ2': (0, 255, 255)
}
roi_path = "roi_config.txt""
```

3.6.3. Funciones

En la etapa de funciones, se realizará todo lo concerniente al roi, los fotogramas mostrados, el recorte de la imagen en base al roi que se defina.

Para poder guardar el roi que se determine previo al inicio del programa se utiliza las siguientes líneas de código.

```
"# ----------------- FUNCIONES -----------------
def guardar_roi(roi):
    with open(roi_path, 'w') as f:
        f.write(','.join(map(str, roi)))
    print("ROI guardado:", roi)"
```

luego se procede a cargar el "roi" guardado si los valores que se cargan están mal o no existe la carpeta donde se guardaron, el programa deberá ser capaz de restablecer un roi por defecto.

Todo ello se refleja en las siguientes líneas de código.

```
"def cargar_roi():
    if os.path.exists(roi_path):
        with open(roi_path, 'r') as f:
            valores = f.read().strip().split(',')
            if len(valores) == 4:
                return list(map(int, valores))
    return [250, 500, 150, 650]  # valor por defecto"
```

Una vez determinada la función del roi, se entra al procesamiento de la imagen que se hará con el roi, si la fruta esta fuera de la región de interés, la detección no se realizara.

```
"def preprocesar_imagen(fotograma, roi):
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

```
    fotograma_roi = fotograma[roi[0]:roi[1], roi[2]:roi[3]]
    if fotograma_roi.size == 0:
        print("ROI vacío, revisa coordenadas:", roi)
        return None
    imagen_redimensionada = cv2.resize(fotograma_roi, (160, 160))
    array_imagen = np.expand_dims(np.array(imagen_redimensionada,
dtype=np.float32), axis=0)
    return preprocess_input(array_imagen)”
```

Posterior a ello se entra en la etapa de carga del video en tiempo real y el ajuste del roi.

3.6.4. Carga de video y Roi

```
“# ---------- CARGA DE VIDEO Y ROI ----------
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error al abrir la cámara.")
    exit()

roi = cargar_roi()
ajustando_roi = True

print("🎮 Ajuste de ROI (WASD para mover, IJKL para escalar, ENTER para
guardar)")

while ajustando_roi:
    ret, frame = cap.read()
    if not ret:
        break

    frame_copia = frame.copy()
    cv2.rectangle(frame_copia, (roi[2], roi[0]), (roi[3], roi[1]), (0, 255, 255), 2)
    mostrar_texto(frame_copia, "Ajustando ROI...", (10, 30), 1, (0, 255, 255), 2)
    cv2.imshow("Ajuste de ROI", frame_copia)

    key = cv2.waitKey(0) & 0xFF
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
if key == ord('w'): roi[0] -= 5; roi[1] -= 5
elif key == ord('s'): roi[0] += 5; roi[1] += 5
elif key == ord('a'): roi[2] -= 5; roi[3] -= 5
elif key == ord('d'): roi[2] += 5; roi[3] += 5
elif key == ord('i'): roi[1] += 5
elif key == ord('k'): roi[1] -= 5
elif key == ord('j'): roi[3] += 5
elif key == ord('l'): roi[3] -= 5
elif key == 13:  # ENTER
    guardar_roi(roi)
    ajustando_roi = False
elif key == ord('q'):
    print("Cancelado por el usuario.")
    cap.release()
    cv2.destroyAllWindows()
    exit()”
```

Como se puede ver en las líneas de código previas, primero procedemos a cargar la imagen entregada por la cámara web, para nuestro caso es el (0). En caso no se encuentre la cámara web en ese destino, el código enviara error al abrir la cámara.

Para poder lograr un correcto posicionamiento del roi se realizó el ajuste por teclas, (a,s,w,d) para poder modificar el posicionamiento del roi e (i,k,j,l) para modificar el tamaño del roi.

Se le agrega confirmación por enter para iniciar el programa en tiempo real y “q” para terminar el programa.

Una vez se presiona enter, el roi quedara guardad para la siguiente vez que se corra el programa.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

3.6.5. Ciclo principal

```
"# ------------------ CICLO PRINCIPAL ------------------
mostrar = False
print("Sistema iniciado. Esperando CAM_ON...")
try:
    while True:
        ret, fotograma = cap.read()
        if not ret:
            break
        # Leer señal desde Arduino
        if arduino.in_waiting:
            linea = arduino.readline().decode().strip()
            if linea == "CAM_ON":
                mostrar = True
                print("Clasificación activada.")
                tiempo_inicio = time.time()
                enviado = False
                confirmaciones = 0
                clase_anterior = ""
                while time.time() - tiempo_inicio < 3.0:
                    ret, fotograma = cap.read()
                    if not ret:
                        continue
                    imagen_preprocesada = preprocesar_imagen(fotograma, roi)
                    if imagen_preprocesada is None:
                        continue
                    predicciones = modelo.predict(imagen_preprocesada)
                    clase_predicha = np.argmax(predicciones)
                    probabilidad = predicciones[0][clase_predicha] * 100
                    etiqueta = clases[clase_predicha]
                    color = colores.get(etiqueta, (0, 255, 0))
                    if etiqueta == clase_anterior and probabilidad > 70:
                        confirmaciones += 1
                    else:
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

```
                confirmaciones = 1
                clase_anterior = etiqueta
            if confirmaciones >= 3:
                mensaje = etiqueta + '\n'
                arduino.write(mensaje.encode())
                print(f"Enviado al Arduino: {mensaje.strip()}")
                enviado = True
                break

            if mostrar:
                cv2.rectangle(fotograma, (roi[2], roi[0]), (roi[3], roi[1]), color, 2)
                mostrar_texto(fotograma, f'Clase: {etiqueta}', (10, 30), 1, color,2)
                mostrar_texto(fotograma, f'Probabilidad: {probabilidad:.2f}%', (10,
70), 1, color, 2)
            cv2.imshow("Clasificador en Tiempo Real", fotograma)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                raise KeyboardInterrupt
        if not enviado:
            print("No se detectó fruta válida. Se omite acción.")
            arduino.write(b"MQ2\n")  # Asumir como fruta en mal estado si nada se
clasificó
    elif linea == "CAM_OFF":
        mostrar = False
        print("Clasificación terminada. Ocultando ROI.")
    # Mostrar fuera de CAM_ON si se permite
    if mostrar:
        imagen_preprocesada = preprocesar_imagen(fotograma, roi)
        if imagen_preprocesada is not None:
            predicciones = modelo.predict(imagen_preprocesada)
            clase_predicha = np.argmax(predicciones)
            probabilidad = predicciones[0][clase_predicha] * 100
            etiqueta = clases[clase_predicha]
            color = colores.get(etiqueta, (0, 255, 0))
            cv2.rectangle(fotograma, (roi[2], roi[0]), (roi[3], roi[1]), color, 2)
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

```
                mostrar_texto(fotograma, f'Clase: {etiqueta}', (10, 30), 1, color, 2)
                mostrar_texto(fotograma, f'Probabilidad: {probabilidad:.2f}%', (10, 70), 1,
color, 2)
        cv2.imshow("Clasificador en Tiempo Real", fotograma)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
except KeyboardInterrupt:
    print("Proceso interrumpido por el usuario.")
finally:
    cap.release()
    arduino.close()
    cv2.destroyAllWindows()
    print("✅ Recursos liberados. Fin del programa.") #include <Servo.h>"
```

Una vez iniciado el ciclo, la consola deberá confirmar el inicio del programa, debido a que el programa trabaja directamente con el Arduino, el programa únicamente activa el modo detección cuando el sensor infrarrojo 1 (IR1) detecte la presencia de la fruta dentro de la zona de clasificación, esto se hace con el fin de no tener clasificaciones cuando no hay nada dentro del roi.

Una vez activado el modo de clasificación, se le da un tiempo de 3 segundos para la toma de datos, en caso de que la fruta se mueva o no se logre detectar se detendrá la clasificación y la fruta que haya estado en la muestra será tomada como fruta en mal estado y se desechara.

Para lograr tener un control mayor al momento de la detección se agrega un contador que requerirá de 3 confirmaciones con más del 70% de probabilidad para poder clasificar, una vez se determine cual es la clasificación recién se le envía la señal al Arduino para continuar el proceso.

Durante la detección se deberá mostrar el roi, la clase detectada y la probabilidad de predicción, esto será controlado por la variable mostrar, con ella prenderemos y apagaremos el roi según la lógica de control.

La variable “enviado” será la que determine cuando la detección fue exitosa o no, en caso de no serlo la fruta será tomada como una en mal estado, en pocas palabras es la variable que permite el flujo de datos para continuar con el ciclo.

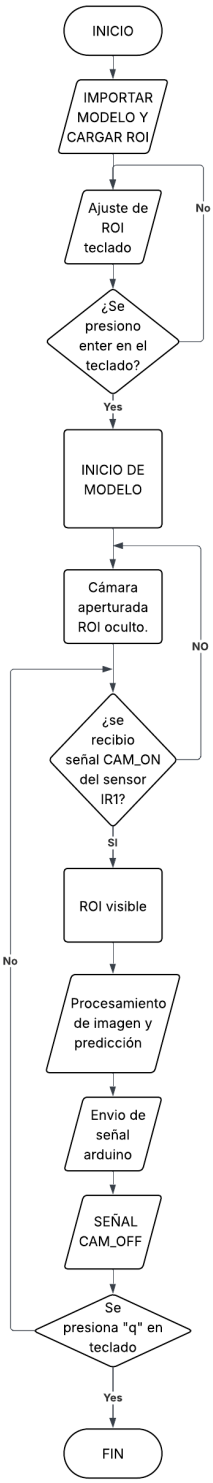
El roi deberá ocultarse cuando se termine el proceso de clasificación, esto se determinará por la señal “cam_off” que será enviada posterior a la clasificación.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 61

Diagrama de Flujo codificación en Spyder.



Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

3.7. Selección de faja

El presente proyecto no tenía el enfoque de construcción de una faja, así que se optó por el modelo más grande que se encuentro en el mercado de material MDF.

Las características de esta son las siguientes, largo 60 cm, ancho 6 cm y si se quitan los bordes que son guía para la faja llega a 10cm.

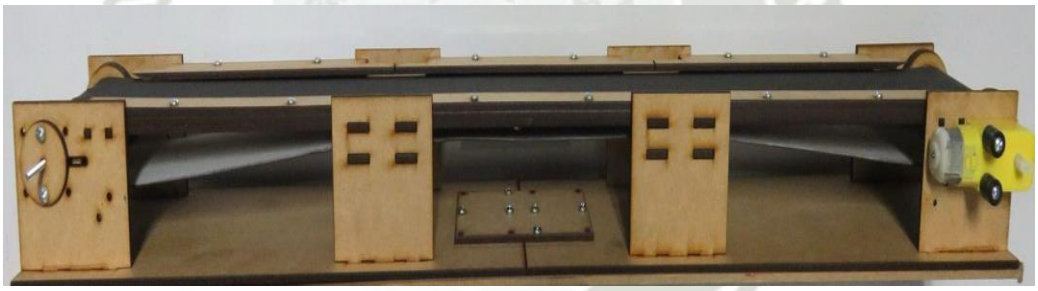
Se determino por esta debido a que la fruta más grande en diámetro necesita de 8 cm de ancho.

Además, para la clasificación final se necesita de un espacio libre para evitar el atascamiento en los bordes en el momento de la clasificación.

La faja incluía un motorreductor tipo TT, que no cumplía con los requisitos para la realización del proyecto, así que este fue reemplazado, posteriormente se en la sección de selección de actuadores se analizará la selección del nuevo motor.

Figura 62

Faja original sin modificaciones.



Nota. Elaboración propia.

3.7.1. Modificación de la faja

Para el presente proyecto se necesitaba una sección de toma de pruebas y una sección de clasificación, la faja tuvo que ser modificada de tal manera que cumpla con los requisitos para poder realizar estas tareas.

3.7.1.1. Zona de toma de pruebas

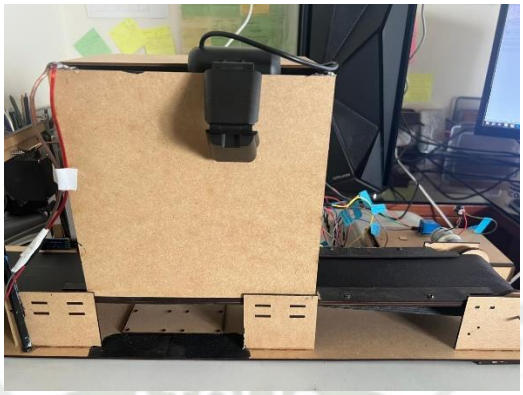
Para la zona de toma de pruebas se realizó un ambiente en MDF cerrado con tapa voladiza.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 63

Modificación para toma zona de toma de pruebas

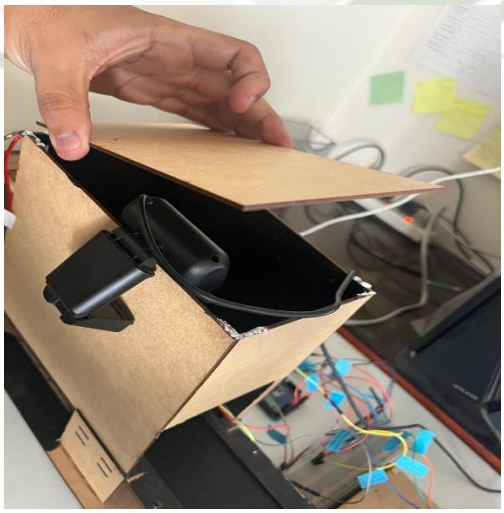


Nota. Elaboración propia.

Las medidas utilizadas en esta zona fueron (dos caras de 21cm x 20 cm que conforman las paredes exteriores de la zona de pruebas, dos caras de 10 cm x 11 cm que realizan la función de puerta de ingreso de la fruta y una tapa superior de 21cm x 10.5cm que se utiliza como techo voladizo para acceso al interior de la faja y para posicionamiento de la cámara.)

Figura 64

Tapa extraíble para acceso a zona de toma de pruebas.



Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

para realizar la unión de estas caras en MDF se utilizó pegamento SuperGlue, en la parte interior del ambiente de toma de datos se pintaron de negro las caras internas, para evitar el rebote de la luz utilizada para la iluminación del fruto, esto es importante debido a que la cámara utilizada no cuenta con enfoque manual ni forma de controlar el brillo, todo lo hace de forma automática y de no realizar este paso, el brillo rebota y quema la imagen tomada.

Figura 65

Interior de zona de toma de pruebas.



Nota. Elaboración propia.

3.7.1.2. Zona de clasificación

Para la zona de clasificación y debido a que un limitante de esta faja es el largo, se determinó por realizar la clasificación mediante un puente elevado, este del mismo modo que la zona de toma de pruebas se realizó con material MDF, pero se incluyó refuerzos en las patas que se fijan a la base de la faja, para evitar rupturas en caso de un toque excesivo del servomotor.

Las medidas de esta sección fueron las siguientes 20cm de alto x 10 de ancho con un espesor de 2mm en las patas pegadas a la base de la faja, los soportes de la faja en color negro son necesarios para evitar que un torque por atascamiento del servomotor parta esa zona.

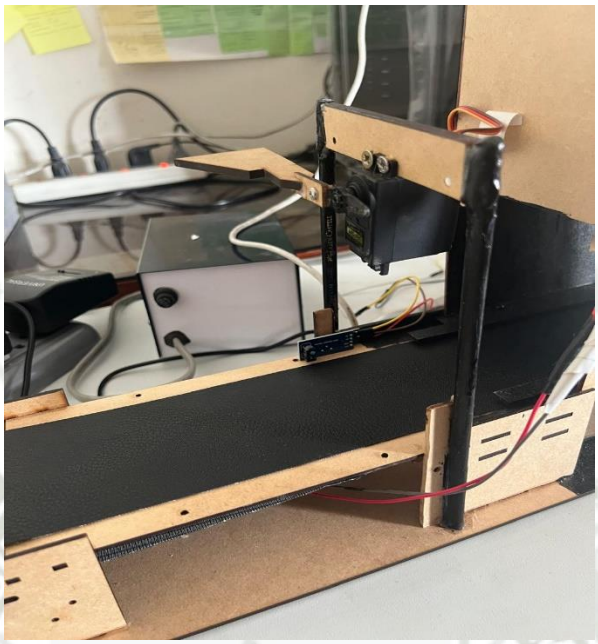
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 66

Zona de clasificación.



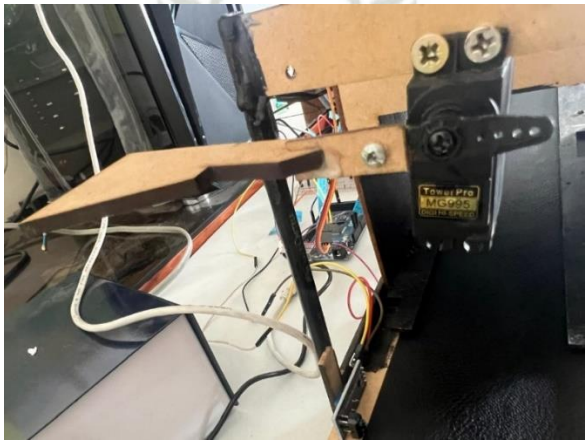
Nota. Elaboración propia.

3.8. Paleta de clasificación.

La paleta de clasificación se realizó con material MDF, con unión empernada y pegamento (super glue) para evitar que por vibraciones se desajuste la paleta.

Figura 67

Montaje de paleta de clasificación



Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

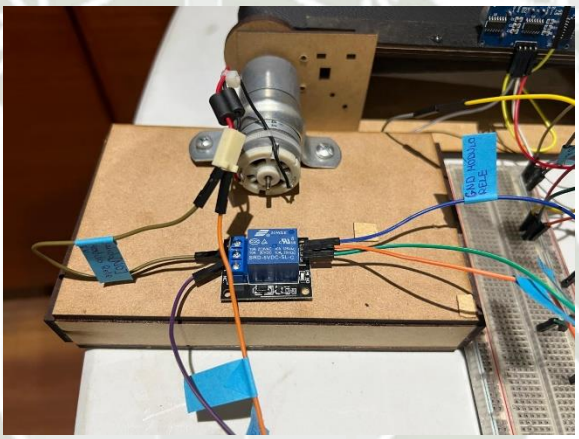
REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

3.9. Base para nuevo motorreductor DC

La faja cuando tenía el motorreductor TT tenía un acople por pernos que estaban fijados a la cara lateral de la faja, sin embargo, al cambiar el motor, esa fijación ya no era posible debido a que el nuevo motor no venía con un modo de acople horizontal, observando esto se determinó crear una base también material MDF, que permitiera la fijación del motor utilizando un abrazadera tipo U fijada con pernos. y además diera espacio para poder agregar el módulo de relé de 5 Voltios, necesario para el control On/Off de la faja.

Figura 68

Anclaje de motorreductor



Nota. Elaboración propia.

3.10. Selección de sensores y actuadores

Para la selección de sensores vamos a dividir la faja en 2 áreas, la primera es la zona de dosificación y la segunda la zona de toma de datos.

3.10.1. Selección de sensores

3.10.1.1. Selección de sensores de proximidad

El sensor de proximidad en este proyecto tiene la tarea de realizar la detección del objeto y avisar al proceso que está próximo a recibir una fruta dentro del área de toma de pruebas, además de avisar cuando la fruta en mal estado se encuentre justo debajo del servomotor, que eliminara estas del proceso.

Para esta tarea hay que tener ciertas consideraciones, el sensor trabajara con una distancia de como máximo 1 cm del objeto, además no se encontrara expuesto a luz potente directa.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Para esta tarea se tuvieron en mente dos tipos de sensores, los cuales fueron implementados y probados.

El movimiento de la faja será regulado por voltaje, pero se estima que un movimiento adecuado para evitar movimientos abruptos será de 5RPM mejorable según el comportamiento del conjunto.

Tipos de dispositivos.

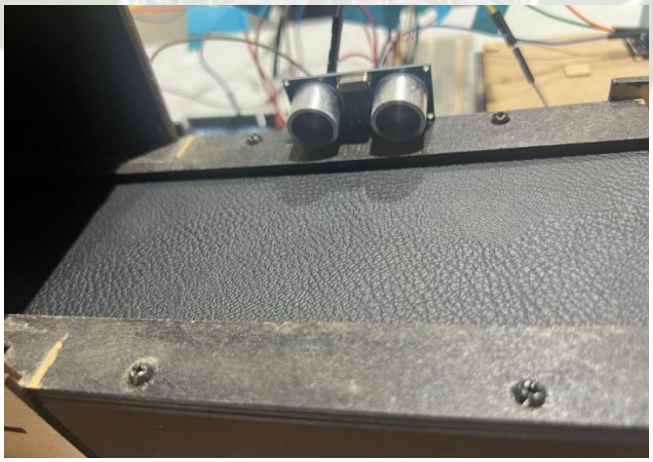
- **Sensor ultrasónico**

  El sensor ultrasónico no cumple con los requisitos de detección por cercanía, presenta un comportamiento errático debido a que este principalmente trabaja a distancias mínimas de 2cm y a una distancia máxima de 4m, además el tiempo de respuesta de este sensor es de 60 ms, tiene una precisión de 0.3 y requiere de objetos que pasen a una velocidad lenta, para tener una lectura precisa.

  Este sensor se logró implementar en la faja, dando resultados negativos como se esperaba.

Figura 69

Prueba de sensor ultrasónico en faja.



Nota. Elaboración propia.

- **Sensor infrarrojo reflexivo**

  El sensor infrarrojo es un sensor con tiempo de respuesta elevado, de 1 a 2 ms, trabaja a una distancia de 0.5 cm o menos, regulable por el

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

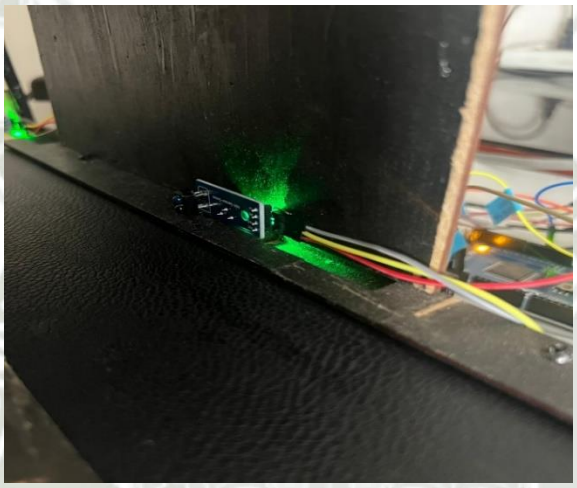
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

potenciómetro que se encuentra en su parte posterior, además de ser muy sensible al paso de objetos.

Se opto por la implementación de dos sensores infrarrojos uno destinado al aviso de la fruta cuando llega al área de toma de datos y otro destinado a dar aviso al servomotor que la fruta en mal estado se encuentra lista para su descarte.

Figura 70

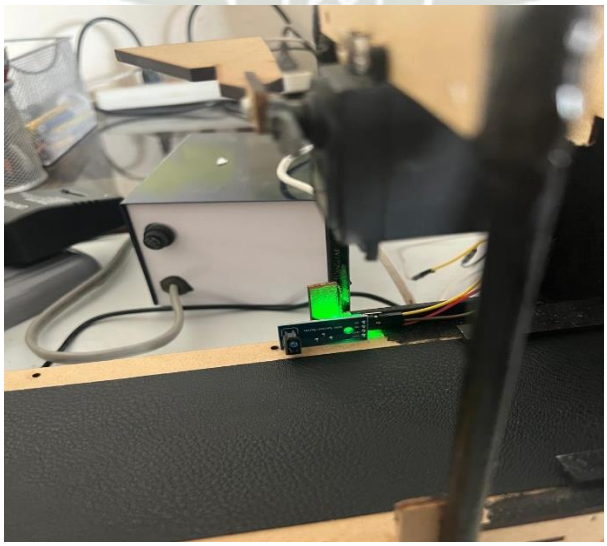
Montaje de sensor infrarrojo ir1



Nota. Elaboración propia.

Figura 71

Montaje de sensor infrarrojo ir2



Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

3.11. Selección de cámara

Debido a que uno de los limitantes del proyecto es mantenernos en un costo bajo de implementación, se decidió contar con una cámara web genérica GeneralPlus USB2.0 UVC WebCam - PID 2247 que es capaz de grabar videos en full HD hasta en 30fps, además de contar con enfoque fijo manual, regulable desde la perilla de la lente, además cuenta con una distancia focal óptima para poder realizar la captura de las imágenes en cercanía, debido a que es un proyecto con espacio reducido, esto es un punto a favor de la cámara.

sin embargo, esta cámara cuenta con limitantes propios que resultan contraproducentes para el modelo, el autoajuste de brillo y contraste, esto causa que, si se tienen pozos de luz muy incandescentes, la cámara intenta absorber la mayor cantidad y quema la imagen. la calidad de la toma de datos se ve limitada por los dos megapíxeles que brinda la cámara, es decir menor detalle en la imagen y por consiguiente menores detalles que le llegan al entrenamiento a la red.

Estos parámetros deben ser tratados con cuidado para obtener los mejores resultados posibles.

Esta cámara con un costo accesible sirve para poder realizar un proyecto decente sin tener que recurrir a cámaras de mayor costo.

Figura 72

Cámara web para toma de datos



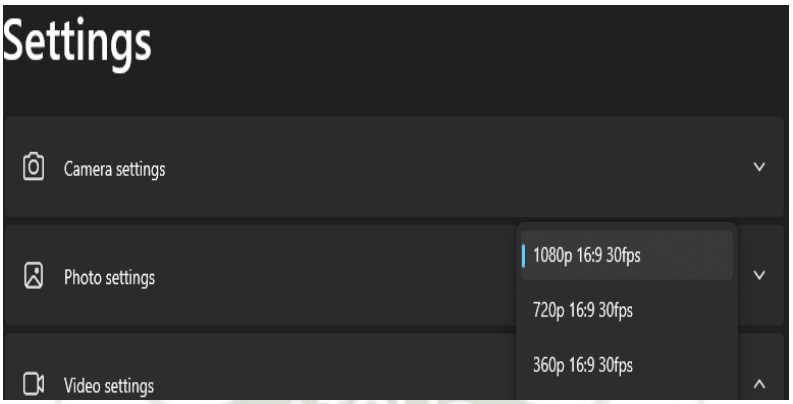
Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 73

Parámetros de configuración de Cámara.



Nota. Elaboración propia.

Figura 74

Posicionamiento de Cámara dentro de zona de toma de pruebas.



Nota. Elaboración propia.

3.12. Selección de actuadores

3.12.1. Selección de servomotor

Para hacer la selección del servomotor, se tomó datos del peso de las frutas, en el caso de este proyecto la fruta más pesada resulta ser una naranja de 271 gramos, la cual se usará como base para poder calcular el torque que se necesitará para el servomotor.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 75

Selección del fruto más pesado.

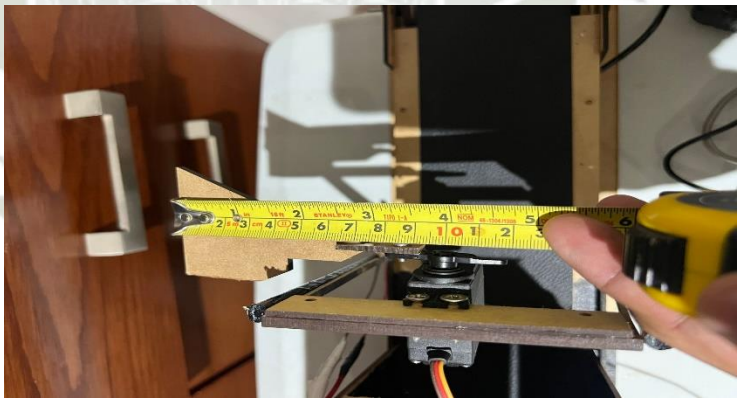


Nota. Elaboración propia.

Una vez teniendo el peso del fruto, se procede a medir la distancia que hay del eje del servomotor hasta la punta de la paleta clasificadora, que para este modelo es de 10 cm.

Figura 76

Medición de distancia entre eje y paleta clasificadora.



Nota. Elaboración propia.

3.12.1.1. Hallando fuerza.

Datos que se tienen:

$naranja\ mas\ pesada = 271\ gr = 0.271\ kg$

$distancia\ eje\ de\ servo\ y\ extremo\ de\ paleta = 10\ cm = 0.1\ m$

$gravedad = 9.81\ m/s^2$

Aplicando formula:

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

$$Fuerza = masa\ x\ gravedad$$

$$Fuerza = \ 0.271\ kg\ x\ 9.81 \frac{m}{s^2}$$

$$Fuerza = 2.66\ N$$

3.12.1.2. Hallando torque.

$$Torque = Fuerza\ x\ Distancia$$

$$Torque = 2.66\ N\ x\ 0.1\ m$$

$$Torque = 0.266\ Nm$$

Transformando torque a Kg x cm

$$1Nm = 10.1972\ kg\ x\ cm$$

$$Torque = 0.266\ Nm\ x\ \frac{10.1972\ kg\ x\ cm}{1Nm}$$

$$Torque = 2.712\ kg.cm$$

3.12.1.3. Dispositivos para elegir.

Una vez conociendo el torque necesario que deberá tener nuestro servomotor, se procede a ver que tipos existen en el mercado, de forma comercial.

Se consideraron 3 servomotores para elección estos son SG90, MG90 y TowerPro MG995, a continuación, se separan los 3 y se expone sus características.

Tabla 11

Tabla de comparación de servomotores

Características	**SG90**	**MG90**	**TowerPro MG995**
Tipo engranajes	Plásticos	Metálicos	Metálicos
Voltaje	3 – 7.2 voltios	4.8 - 6 voltios	4.8 – 7.2 voltios
Torque	1.8 – 2.5 Kg x cm	2.5 – 3 kg x cm.	9.4 – 11 kg x cm

Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Teniendo las características más importantes de los 3 tipos de servomotores nos damos cuenta de que el servomotor SG90 no cumple con el requisito debido a que el torque necesario es mayor que su torque de estancamiento.

El servomotor MG90 si cumple con el requisito de torque, sin embargo, de elegir este nos enfrentamos al primer problema el cual es que trabajara al límite y esto podría causar recalentamiento, además que en caso de estancamiento no le quedaría mucho torque disponible para lograr superar el problema.

Por descarte nuestra elección sería el TowerPro MG995, si bien el torque va de sobra para mover tanto el fruto más pesado como el más liviano, ese torque extra nos permite solventar problemas de atascamiento, además de tener al servomotor siempre trabajando por debajo de su torque mínimo.

Figura 77

Servomotor TowerPro mg995



Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Una vez seleccionado el servomotor, hay que escoger que alimentación utilizara, para ello hay que determinar el consumo del MG995 bajo la carga de la fruta y bajo una supuesta carga de estancamiento.

Según datasheet que se encuentra en el Anexo 1, se tiene lo siguiente.

Tabla 12

Consumo de corriente de towerpro mg995

Estado de Servomotor	Consumo de corriente
Sin carga y sin moverse (idle)	170 mA
moviéndose, pero sin carga (no load current)	170 mA
Corriente de bloqueo (stall current)	1200 mA

Nota. Elaboración propia.

Realizamos una regla de tres simple para poder determinar cuál será el consumo de corriente bajo el torque hallado utilizando como base la corriente de bloqueo:

$$Consumo\ de\ corriente\ para\ torque\ hallado = \frac{2.7\ kgxcm}{11kgxcm}\ x\ 1.2A$$

$$Consumo\ de\ corriente\ para\ torque\ hallado = \ 0.295A = 295mA$$

Una vez determinado el consumo de corriente, y teniendo en cuenta la corriente de bloqueo, se necesita energizar el servomotor con una fuente externa, para ello se determino utilizar una fuente de 5 voltios a 2 amperios.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 78

Fuente de alimentación para servomotor towerpro mg 995



Nota. Elaboración propia.

3.12.2. Elección de motorreductor.

para poder realizar la selección del motorreductor es necesario determinar el torque que se necesitara para mover la fruta más pesada desde parado, para ello tenemos que determinar el radio del rodillo de la faja, debido a que con ese dato recién seremos capaces de hallar el torque.

Figura 79

Medición del diámetro del rodillo de la faja



Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Se determina que el diámetro es 4 cm por consiguiente el radio es de 2 cm, con la fuerza que fue previamente hallada para la selección del servomotor y este dato se procede a reemplazar en la ecuación de torque.

$$Torque = Fuerza\ x\ Distancia$$

$$Torque = 2.66N\ x\ 0.02m$$

$$Torque = 0.0532Nm$$

Transformando todo a kg x cm

$$Torque = 0.54\ kg\ x\ cm$$

Este torque, cualquier motor es capaz de moverlo a su voltaje y velocidad nominal, sin embargo, no requerimos de 200 RPM en una faja tan corta, hay ciertas condiciones que se tomaron en cuenta, una de ellas es que la fruta debe viajar por la faja sin estar moviéndose para todo lado, es decir que el movimiento sea controlado, debido a que ya se cuenta con una fuente regulable, para ello se propone trabajar con una velocidad mínima de 5 RPM.

Con esta premisa, la pregunta seria, que motor es capaz de mover el torque hallado, bajando el voltaje de entrada, para ello se toma las características de voltaje y revoluciones de los dos prospectos, el motorreductor TT y un motorreductor DC a 30 RPM.

Motorreductor TT cuenta con una salida de 200 RPM a 6v y nosotros necesitamos saber a cuanto deberíamos bajar el ingreso de voltaje para llegar a 5RPM, para ello aplicamos una regla de tres simple obteniendo:

$$voltaje\ a\ 5Rpm\ Motorreductor\ TT = \frac{6\ Volt\ x\ 5\ RPM}{200\ RPM}$$

$$voltaje\ a\ 5Rpm = 0.15\ voltios.$$

Con este resultado el motorreductor TT queda totalmente descartado, debido a que con ese voltaje no llegaría siquiera a moverse.

Ahora aplicamos el mismo ejercicio con el motorreductor DC a 9V y 30RPM.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

$$voltaje\ a\ 5Rpm\ Motorreductor\ 30RPM = \frac{9\ Volt\ x\ 5\ RPM}{30\ RPM}$$

$$voltaje\ a\ 5Rpm\ Motorreductor\ 30RPM = 1.5\ Voltios$$

Con este resultado y probado de forma práctica el motor logra moverse, pero es necesario calcular si sería capaz de mover el torque hallado con este voltaje.

Para ello se toma datos de motores similares, debido a que este motor se consiguió en desarme y no hay manera de saber el código de motor para buscar su información, sin embargo, la gran mayoría de motorreductores que trabajan a 9 voltios y 30 RPM trabajan en un rango muy parecido de torque, entonces se tiene que estos motores trabajan entre un rango de 15 a 20 kg x cm como torque máximo, nosotros tomaremos el torque menor para los cálculos, de tal manera que nuestra estimación no este sobreevaluada.

Entonces tenemos una regla de tres simple que consta de las siguientes premisas, si a 9 voltios consigo un torque de 15 kg x cm, a 1.5 voltios cuanto torque soy capaz de generar.

$$Torque\ generado\ a\ 1.5\ voltios = \frac{15\ kgcm\ x\ 1.5\ voltios}{9\ voltios}$$

$$Torque\ generado\ a\ 1.5\ voltios = 2.5\ kg\ x\ cm$$

Con este dato nos damos cuenta de que el torque a ese voltaje es suficiente para mover el peso de la fruta desde parado, lo cual fue corroborado y testeado en la práctica, cabe resaltar que este torque lo consigue este motor, por la relación de transmisión, a tensión nominal genera pocas vueltas, además de contar con engranajes metálicos.

Una vez determinado el motor, debemos ver el modo de control que se utilizara, debido a que esta faja ya está regulada su velocidad por voltaje directo de fuente, lo único que nos hace falta es un control On/Off del motor, para ello se implementó un módulo de relé a 5V, el cual permitirá el encendido y apagado del motor dependiendo de la lógica otorgada.

Hay que destacar que la velocidad de la faja puede ser mayor dependiendo de las pruebas realizadas en la implementación.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 80

Implementación de motorreductor y relé a 5v.



Nota. Elaboración propia.

3.13. Selección de la iluminación

Para esta sección de iluminación se tuvieron que realizar pruebas diferentes, para determinar cuál era la mejor manera de iluminar los objetos presentados en la zona de toma de pruebas.

Un punto crítico al momento del entrenamiento es la toma de datos, es recomendable que las imágenes utilizadas para el entrenamiento, validación y prueba se encuentren dentro de la zona donde trabajara la cámara, es decir que, si se quiere entrenar una red eficientemente, debemos considerar sus datos como información donde esta red realizara su función, en este caso dentro de la zona de toma de datos.

Con esto en mente, las imágenes se capturan directamente desde el punto de vista de la cámara web y sus datos saldrán de tomas realizadas con la misma cámara web. Debido a que no es recomendable hacer un entrenamiento con calidad de imágenes de celular que no serán útiles al momento de realizar la detección debido a que estas muestran más detalle del que la cámara web puede captar.

Para este apartado justamente una limitante es la calidad de la cámara web, para el proyecto se utilizó una cámara genérica de costo bajo, que posee un problema en específico, esta cámara cuenta con iluminación automática y no existe manera de poder cambiar este apartado, por lo que cuando la cámara se encuentra en la zona de toma de datos, suele quemar la imagen, debido a que intenta aclarar lo más posible todo lo que ve, quedando las imágenes de esta manera.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 81

Imagen de prueba sin regularización de luz



Nota. Elaboración propia.

Este tipo de tomas generan una data inservible para aprendizaje, debido a que no se logran observar los detalles de la fruta, y no es posible realizar un entrenamiento para clasificación, ya que todas las imágenes perderán sus características.

La manera más eficiente que se encontró de poder iluminar la fruta fue mediante tira led, sin embargo, la posición de la tira led es muy importante al momento de la toma de datos, posteriormente se explicara el porqué.

Se tomo 2 maneras de posicionamiento de la tira led, en la siguiente imagen se podrá observar la toma de datos de la cámara web cuando la tira led se encuentra posicionada alrededor de toda la zona de toma de datos.

Figura 82

Imagen de prueba iluminación perimetral



Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

En este caso al estar tan cerca de la fruta la iluminación led, se crea un halo encima de ella, la cual forma parte de las características que el modelo aprende para diferenciar entre una fruta y otra, como las manzanas y las naranjas logran reflejar este halo, el modelo tiene problemas en el aprendizaje y cuando llega la prueba confunde manzanas con naranjas, también se produce el cambio de coloración de las frutas, dándole a la manzana una coloración más naranja en ciertas zonas, que luego afectaran en el reconocimiento.

Para solventar el problema del halo quemado se optó por pintar el interior de la caja de toma de pruebas de color negro, esto con el propósito de absorber la luz en vez de que esta rebote, además reposicionando la tira led, de tal manera que esta no salga en la captura de los datos y posteriormente bajándole la intensidad mediante voltaje, además de agregarle un difusor de luz hecho con papel mantequilla, que lo que permite es que la luz se distribuya de mejor manera en todo la zona y no se reflejen los puntos led en las frutas.

Figura 83

Zona de toma de datos modificada para apreciación de detalle.



Nota. Elaboración propia.

Como se puede ver en la imagen, la iluminación es homogénea y no genera iluminación por puntos, además de otorgar las siguientes imágenes para tomas de datos.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Figura 84

Resultado final para toma de datos.



Nota. Elaboración propia.

Donde se puede observar claramente las características de la manzana, con puntos quemados mínimos y mayor detalle

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

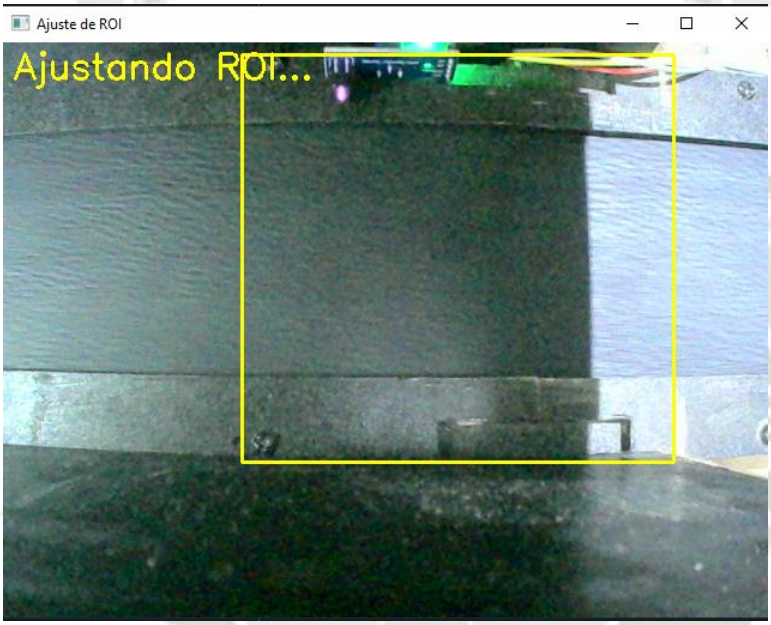
CAPÍTULO IV

4. PRUEBAS Y RESULTADOS DEL MODELO.

Al iniciar el programa, primero se despliega una pantalla para hacer el ajuste del roi.

Figura 85

Pantalla de ajuste de ROI.

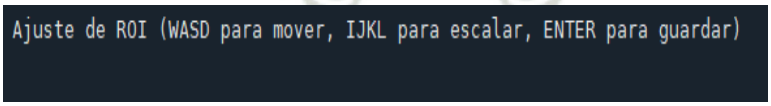


Nota. Elaboración propia.

En la consola nos enviara el siguiente mensaje.

Figura 86

Mensaje de aviso para movimiento de ROI.

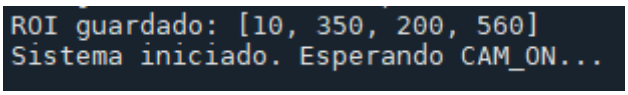


Nota. Elaboración propia.

Una vez presionado “enter” se guarda el roi y se empieza el proceso.

Figura 87

Mensaje de coordenadas grabadas para ROI.



Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

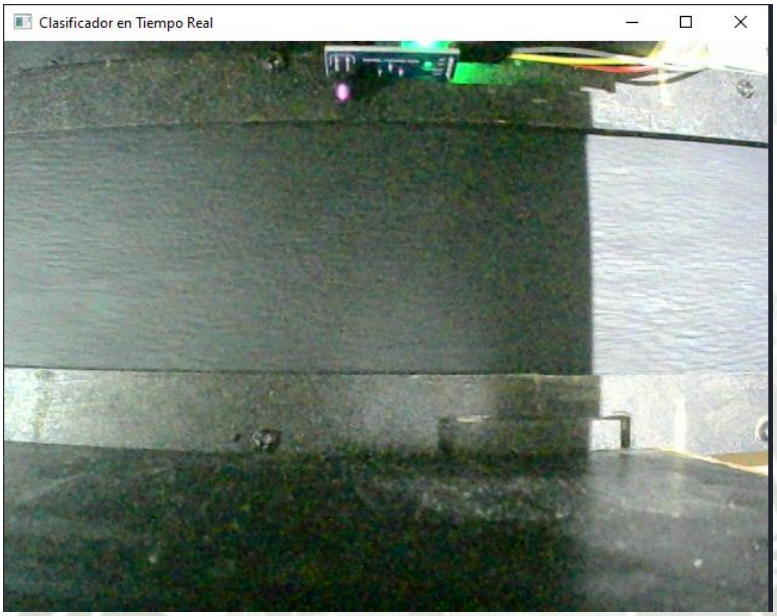
REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Nota. Elaboración propia.

El roi, la clase y la probabilidad quedan ocultas hasta que la fruta es detectada por el sensor infrarrojo (IR1).

Figura 88

Ventana a la espera de detección para iniciar clasificación.



Nota. Elaboración propia.

Una vez que el sensor IR1 detecta la fruta, recién se habilita la clasificación.

Figura 89

Fruta detectada ROI habilitado, clasificación en proceso.



Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

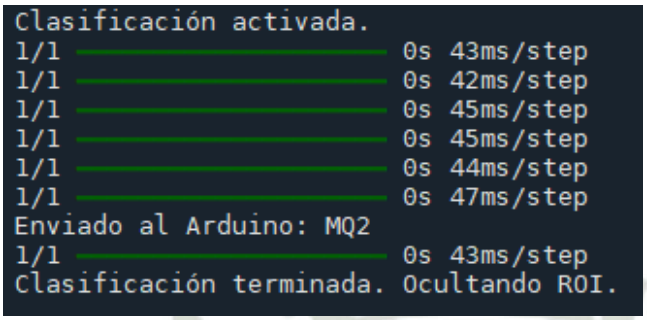
REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Una vez clasificada la fruta, se envía en la consola el tipo de clasificación que se determinó y se oculta roi.

Figura 90

Mensaje en consola después de detección.

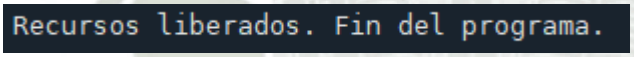


Nota. Elaboración propia.

Cuando ya se acabaron las pruebas y se presiona la letra “q” en el teclado, se acaba el proceso y se muestra lo siguiente en la consola.

Figura 91

Termino de programa a consecuencia de pulsar tecla q en teclado.



Nota. Elaboración propia.

4.1.Pruebas y resultados.

Para las pruebas se requirió el modelo completamente implementado con la parte física del proyecto.

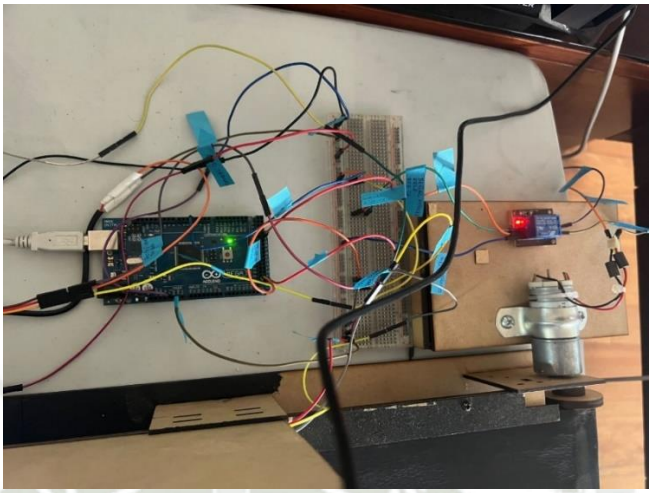
Primero se realizan las conexiones necesarias de sensores y actuadores al Arduino, para ello se hace uso de cables macho a macho, macho a hembra y hembra a hembra.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Figura 92

Conexión de sensores y actuadores con Arduino.

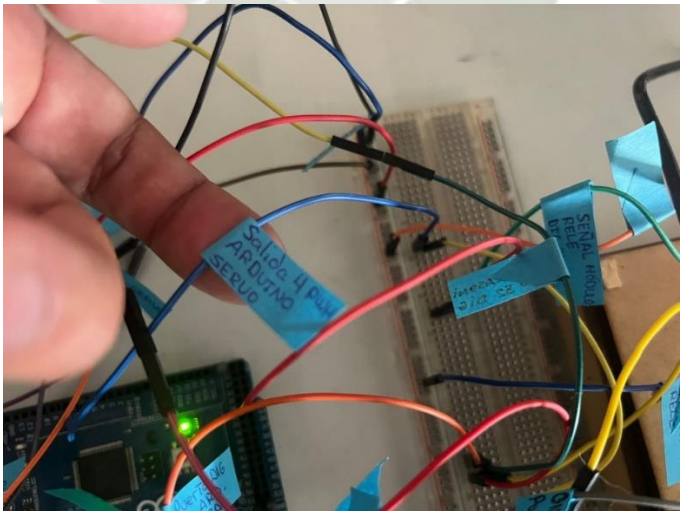


Nota. Elaboración propia.

Para facilitar el entendimiento de la conexión se agregarán etiquetas, indicando que es lo que conecta cada cable.

Figura 93

Etiquetas para orientación de conexión en cables.



Nota. Elaboración propia.

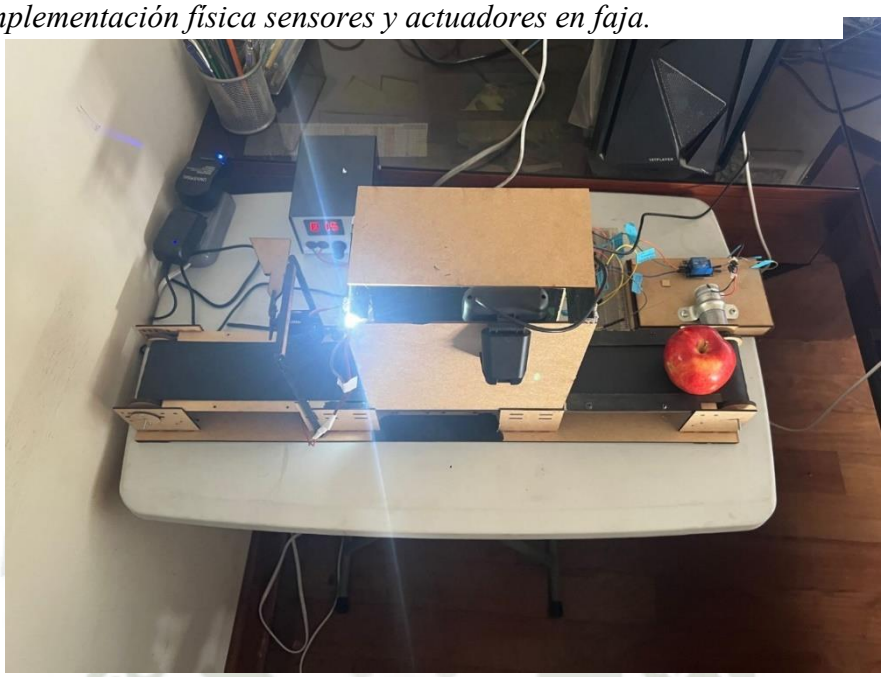
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Figura 94

Implementación física sensores y actuadores en faja.

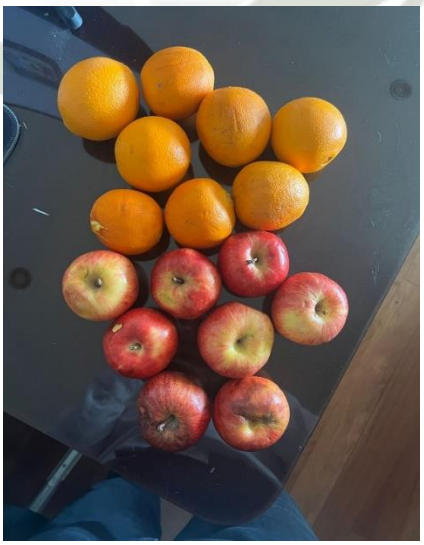


Nota. Elaboración propia.

Posterior a ello se procede a realizar seleccionar el grupo de pruebas que constara de manzanas y naranjas en buen y mal estado.

Figura 95

Manzanas y naranjas para prueba de modelo implementado.



Nota. Elaboración propia.

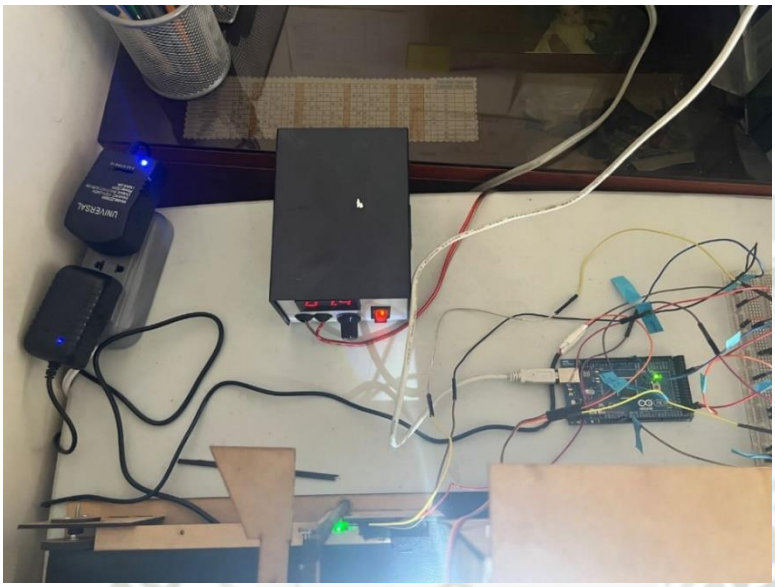
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Se procede a energizar Arduino, y conexión de fuentes externas.

Figura 96

Arduino y fuentes externas energizadas.

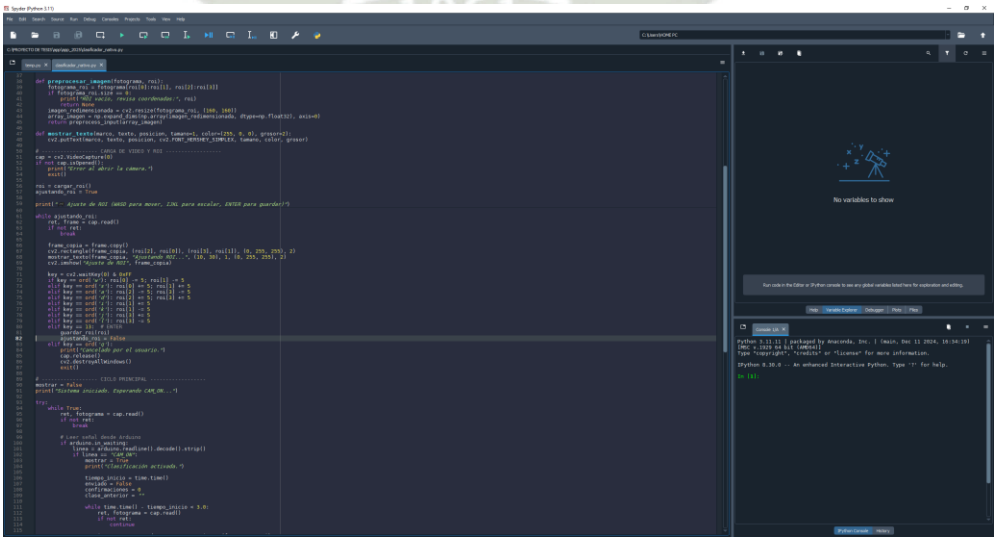


Nota. Elaboración propia.

Ingresamos a nuestro programa en spyder para iniciar con el proceso.

Figura 97

Entorno en Spyder para inicio de secuencia.



Nota. Elaboración propia.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Realizamos el ajuste de Roi que para el modelo será con las siguientes coordenadas

Figura 98

Roi utilizado para realización del proyecto.

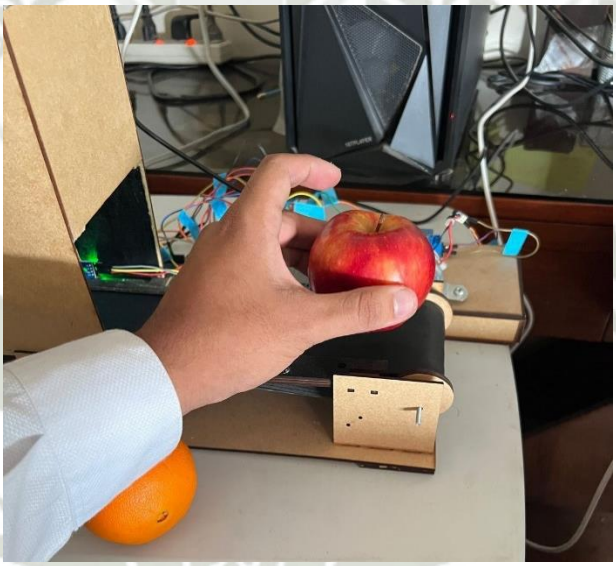
```
ROI guardado: [10, 350, 200, 560]
```

Nota. Elaboración propia.

Seguido a esto, se procede al inicio del sistema, en donde se hará la dosificación de fruta manual y se determinará la precisión del sistema en un entorno real de trabajo.

Figura 99

Dosificación manual de manzanas en faja



Nota. Elaboración propia.

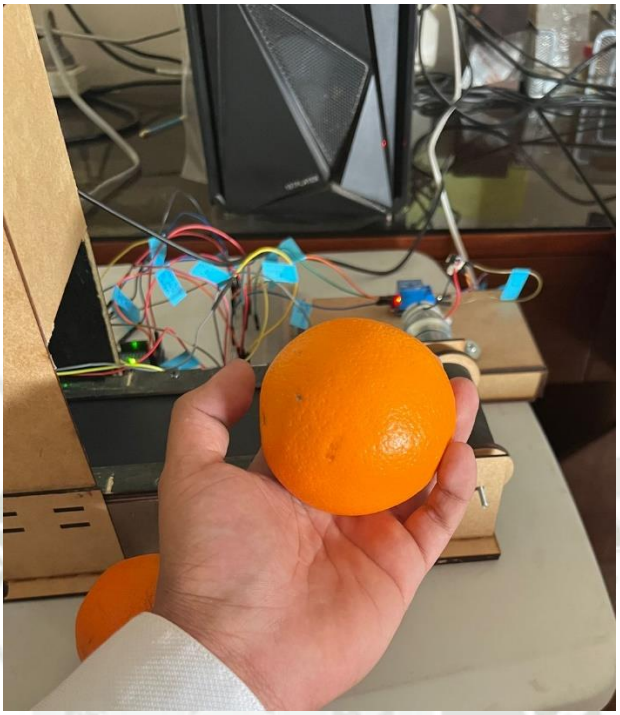
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Figura 100

Dosificación manual de naranjas en faja



Nota. Elaboración propia.

Una vez agregada la fruta, el modelo realiza la clasificación correspondiente, que se podrá observar en tiempo real gracias a la ventana de clasificación que se tiene en spyder.

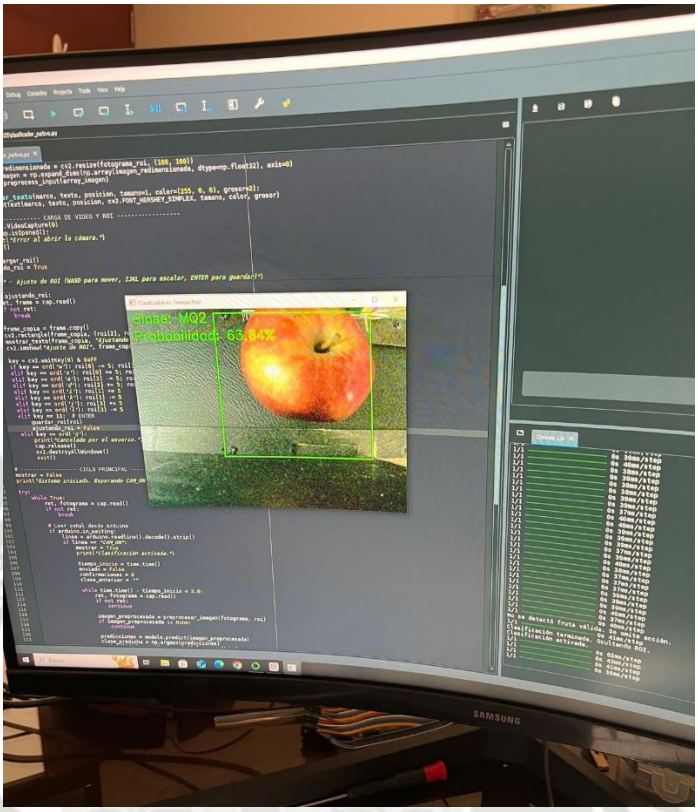
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Figura 101

Ventana de clasificación en tiempo real.



Nota. Elaboración propia.

En la consola se podrá observar que se cumplen los steps de confirmación,

Figura 102

Confirmación de clasificación en consola.

```
Clasificación activada.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Enviado al Arduino: MQ2
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
Clasificación terminada. Ocultando ROI.
```

Nota. Elaboración propia.

Para las imágenes se agregó una manzana calidad MQ2, la velocidad de respuesta del programa es de aproximadamente 177 ms, sin embargo, cabe resaltar que esto lo logra cuando la probabilidad de lo que está viendo es más del 70%.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

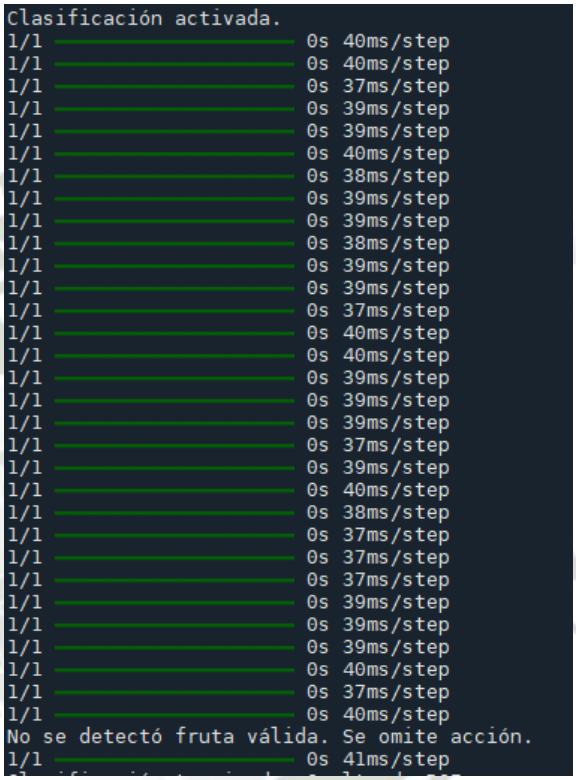
REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Para casos donde el modelo no está seguro de lo que ve, tiene una ventana de 3 segundos para hacer sus 3 confirmaciones, sin embargo, cuando clasifica 3 veces con probabilidad baja, el modelo debe priorizar la eficiencia del proceso y desechar la fruta que no puede clasificar.

Figura 103

Ejemplo de falla en clasificación.



Nota. Elaboración propia.

Para realizar el cálculo de la eficiencia del proyecto se realizó una tabla con el tipo de frutas que se utilizó para clasificación, el tiempo que demoro y la certeza que tuvo.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Tabla 13

Tabla de toma de pruebas modelo.

	CALIDAD ENTREGADA	CALIDAD CLASIFICADA	TIEMPO DE CLASIFICACION EN MS
1	NQ1	NQ1	366
2	MQ1	MQ1	1000
3	MQ2	MQ2	187
4	NQ2	FRUTA NO VALIDA	3000
5	MQ1	MQ1	289
6	MQ2	MQ1	224
7	MQ1	MQ1	212
8	NQ1	NQ1	1025
9	NQ1	NQ1	1296
10	NQ1	NQ1	165
11	NQ2	NQ2	380
12	MQ2	MQ2	169
13	MQ1	MQ1	192
14	MQ1	MQ1	203
15	MQ1	MQ2	1200
16	MQ1	MQ1	240
17	NQ1	FRUTA NO VALIDA	3000
18	NQ1	FRUTA NO VALIDA	3000
19	NQ1	NQ1	950
20	NQ1	NQ1	844
21	MQ2	MQ2	157
22	MQ2	MQ2	154
23	MQ2	MQ2	150
24	MQ2	MQ2	158
25	MQ2	MQ2	173
26	NQ2	NQ2	420
27	NQ2	NQ2	482
28	NQ2	FRUTA NO VALIDA	3000
29	NQ2	NQ2	432
30	NQ2	FRUTA NO VALIDA	3000
31	NQ2	NQ2	344
32	NQ2	FRUTA NO VALIDA	3000
33	NQ2	NQ2	480
34	NQ2	NQ2	420
35	MQ2	MQ2	169
36	MQ2	MQ2	188
37	MQ2	MQ2	194
38	MQ2	MQ2	188

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

	CALIDAD ENTREGADA	CALIDAD CLASIFICADA	TIEMPO DE CLASIFICACION EN MS
39	MQ2	MQ2	160
40	NQ1	NQ1	890
41	NQ1	NQ1	873
42	MQ1	MQ1	220
43	MQ1	FRUTA NO VALIDA	3000

Nota. Elaboración propia.

Para el modelo se realizó una prueba con 43 frutas, de diferentes tipos, entregadas a la faja de manera aleatoria, se recopilo los siguientes datos.

Exactitud del sistema:

De un total de 43 pruebas el algoritmo logro la acertar en 34.

$$Exactitud\ práctica = \left(\frac{\#\ Aciertos}{\#\ pruebas}\right) x100$$

$$Exactitud\ práctica = \left(\frac{\#\ 34}{\#\ 43}\right) x100$$

$$Exactitud\ práctica = 79.07\%$$

Porcentaje de exactitud:

El modelo está configurado para entregar porcentaje de exactitud de clasificación no menor a 70%, esto con el fin de no retardar el proceso de clasificación.

Tiempo de demora en la clasificación:

Solo se considerará el tiempo de clasificación de aquellos que fueron correctamente identificados.

Manzanas:

Para manzanas se tuvo un total de 19 clasificaciones correctas.

$$Tiempo\ de\ clasificación\ promedio\ de\ manzanas: \frac{4403\ ms}{19}$$

$$Tiempo\ de\ clasificación\ promedio\ de\ manzanas: 231.74\ ms$$

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Naranjas:

Para naranjas se tuvo 14 clasificaciones correctas.

$$Tiempo\ de\ clasificación\ promedio\ de\ naranjas: \frac{9001\ ms}{14}$$

$$Tiempo\ de\ clasificación\ promedio\ de\ naranjas: 642.93\ m$$

4.2. Costo

Tabla 14

Presupuesto del proyecto

Etapa	Cantidad	Costo
Cámara web full HD a 30 FPS	1 UND	S/ 80.00
Sensor Infrarrojo	2 UND	S/ 15.00
Servomotor MG995 TowerPro	1 UND	S/ 25.00
Placa Arduino Mega	1 UND	S/ 70.00
Faja transportadora de 60 cm	1 UND	S/ 120.00
Fuente regulable DC 24v 3Amperios.	1 UND	S/ 90
Cables de conexión I/O	30 UND	S/ 10
Iluminación para caja de captura	1 metros	S/ 8
Fuente 5 voltios y 2 Amperios	1 UND	S/ 15
Fuente regulable 3 voltios a 12 voltios	1 UND	S/ 25
MOTOR DC 9V A 30 RPM	1 UND	S/ 30
Cortes en MDF para modificación de faja.	10 Cortes	S/ 30
TOTAL		S/ 518

Nota. Elaboración propia

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

4.3.Financiamiento

El costo del proyecto será asumido completamente por el tesista, sin ningún tipo de patrocinador.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

CONCLUSIONES

1. Se logro la implementación y evaluación del clasificador basado en visión artificial y redes neuronales para analizar la calidad en manzanas y naranjas.
2. Se desarrollaron dos enfoques de entrenamiento para el modelo: una red neuronal desde cero y otra mediante el método de aprendizaje por transfer learning con MobileNetV2. La red entrenada desde cero presentó problemas de sobre entrenamiento debido a la limitada cantidad y calidad de datos. En cambio, MobileNetV2, al estar preentrenada, permitió una mejor generalización y redujo errores de clasificación, aunque con una ligera disminución en la certeza. A pesar de las limitaciones en la base de datos y la cámara, el sistema logró cumplir con el objetivo de clasificar de forma adecuada.
3. Se logro implementar el modelo de visión artificial y redes neuronales, con el hardware seleccionado, teniendo como resultado final un sistema que trabaja en conjunto para lograr una clasificación en tiempo real.
4. Se logro integrar el entrenamiento de la red con su modelo h5 en spyder, con la codificación en Arduino, logrando de esta manera generar un ecosistema de funcionamiento correcto, se realizó la modificación de la faja y la instalación de sensores y actuadores que posteriormente se utilizaron para para llevar a cabo las pruebas correspondientes del modelo en la realidad.
5. Se logro evaluar el sistema del clasificador por visión artificial y redes neuronales, obteniendo datos desde su entrenamiento en Jupyter hasta toma de datos prácticos, en los cuales se obtuvieron resultados de exactitud del 69% y precisión del 75% para el modelo entrenado y resultados de precisión utilizados para confirmaciones de estado en el modelo implementado de más del 70% y una exactitud del 79.07%, estos valores podrían ser mejorables mediante la implementación de una cámara que sea capaz de captar mayor detalle y el aumento de los datos de entrenamiento y validación.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

RECOMENDACIONES

- Para lograr tener un entrenamiento robusto, es necesario cumplir con 3 características importantes, equilibrio en la data de entrenamiento, validación y prueba, crear desbalances en la data de entrenamiento genera sesgo en el modelo, por lo cual tiende a tener preferencia de clasificación por una fruta en específico.

- El modelo presentado puede mejorar su eficiencia y generalización, cambiando la cámara a una que sea capaz de detectar más detalle en la imagen, realizar el entrenamiento con imágenes con mucho ruido y poco detalle, genera dificultad a la red para extraer características de los objetos y con ello tiende a confundir unas con otras.

- Los datos de prueba para el modelo deben ser tomados dentro de la zona donde trabajara, si se entrena con datos muy generales el modelo no será capaz de realizar predicciones correctas, en caso de tener imágenes con muchos objetos diferentes se recomienda realizar el etiquetado de las imágenes, de esta manera le mostramos a la red que es lo que queremos que aprenda.

- Las técnicas de early stopping y dropout, son necesarias cuando tenemos un modelo que tiende mucho a la memorización de datos, dropout no es recomendable utilizarlo en parámetros mayores a 0.5, debido a que un dropout muy elevado puede generar que el modelo tenga problemas para el aprendizaje, early stopping se recomiendo utilizarlo en rango de 3 a 5 rondas, de esta manera al dejar de mejorar el modelo, detendrá el entrenamiento y recuperara los datos de la época donde mejor aprendió y generalizó.

- Los datos de validación deben ser datos que el modelo nunca vio, en caso contrario el modelo pierde capacidad de generalización y empieza a memorizar los datos que observa.

- Los datos de pruebas deben ser datos que no se vieron ni en los datos de entrenamiento ni en los datos de validación, deben ser equilibrados, para tener una métrica más pareja.

- Se recomienda el uso moderado de la técnica de data augmentation sobre todo cuando se trabaja en un área fija, debido a que, aunque se generan imágenes con diferentes ángulos y aumentos, cuando se pasa de 10 copias por imágenes, el modelo puede tener datos que no generen ningún beneficio.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

- El modelo puede aumentar su eficiencia si se tratan aspectos clave como tamaño de dataset de entrenamiento y validación priorizando el uso de imágenes de mayor calidad, debido a que independientemente del número de iteraciones de entrenamiento o el tipo de red preentrenada que se utilice, el modelo no podrá extraer características precisas a partir de imágenes con distorsiones o bajo nivel de detalle.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

5. REFERENCIAS

ATRIA. (s/f). *Aplicaciones de la visión artificial*. Recuperado el 16 de octubre de 2024, de https://atriainnovation.com/blog/siete-aplicaciones-vision-artificial/

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*.

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library.*

Carolina, D., González, R., Arévalo, B. G., Escalante, E. E., Sarmiento, G. P., Manuel, J., Romero, C., & Cancino Suárez, S. (2009). *Adquisición y Reconocimiento de Imágenes por Medio de Técnicas de Visión e Inteligencia Artificial* (Vol. 6). https://hdl.handle.net/20.500.12867/3986

Centro de formación técnica para la industria. (s/f). *Vision Artificial: Todo lo que necesitas saber*. https://doi.org/https://www.cursosaula21.com/que-es-la-vision-artificial/

D. Forsyth, & J. Poince. (2003). *Computer Vision A Modern Appr.*

Davies, E. R. (2012). *Computer & Machine Vision: Theory, Algorithms, Practicalities* (4th ed.).

Gamonal Chanco, A. E. (2020). *Diseño de un sistema por visión artificial para determinar la calidad de mandarinas* [Universidad Tecnológica del Perú]. file:///C:/Users/HOME%20PC/Downloads/Antony%20Gamonal_Tesis_Titulo%20Profesional_2020.pdf

Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th ed.).

Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep Learning*.

Haykin, S. (2009). *Redes Neuronales: Un Enfoque Práctico.*

Heras, D. (2017). Clasificador de imágenes de frutas basado en inteligencia artificial. *Killkana Técnica*, *1*(2). https://doi.org/10.26871/killkana_tecnica.v1i2.79

INEI. (2024). *Estado de la población Peruana*. https://www.inei.gob.pe/media/MenuRecursivo/publicaciones_digitales/Est/Lib1743/Libro.pdf

Kakani, V., Nguyen, V. H., Kumar, B. P., Kim, H., & Pasupuleti, V. R. (2020). A critical review on computer vision and artificial intelligence in food industry. *Journal of Agriculture and Food Research*, *2*, 100033. https://doi.org/10.1016/J.JAFR.2020.100033

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Medina Fernández, J. A. (2019). *DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE CLASIFICACION DE LÚCUMAS APLICANDO VISIÓN ARTIFICIAL*. https://repositorio.ucsm.edu.pe/server/api/core/bitstreams/4d33057b-5319-414a-ad5e-ce7cc4abf088/content

Mundaca Vidarte, G. (2016). *DETECCIÓN DE CARACTERES DE PLACAS DE AUTOMÓVILES MEDIANTE TÉCNICAS DE VISIÓN ARTIFICIAL* [Universidad de Piura]. https://pirhua.udep.edu.pe/backend/api/core/bitstreams/ace60974-d9e5-4098-9ef6-67c270ab3163/content

Pons Moreno,A. & Martinez Verdú, F. (2004). *Fundamentos de visión binocular. Google Libros*. https://books.google.com.pe/books?id=LRyUBAAAQBAJ&printsec=frontcover#v=onepage&q&f=false

Santa María Pinedo, J. C., Ríos López, C. A., Rodríguez Grández, C., & García Estrella, C. W. (2021). Reconocimiento de patrones de imágenes a través de un sistema de visión artificial en MATLAB. *Revista Científica de Sistemas e Informática*, *1*(2). https://doi.org/10.51252/rcsi.v1i2.131

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*.

Sonka, M., Hlavac, V., & Boyle, R. (2007). *Image Processing, Analysis and Machine Vision* (3rd ed.).

Szeliski, R. (2022). *Computer Vision: Algorithms and Applications* (2nd ed.).

Trucco, E., & Verri, A. (1998). *Introductory Techniques for 3-D Computer Vision*.

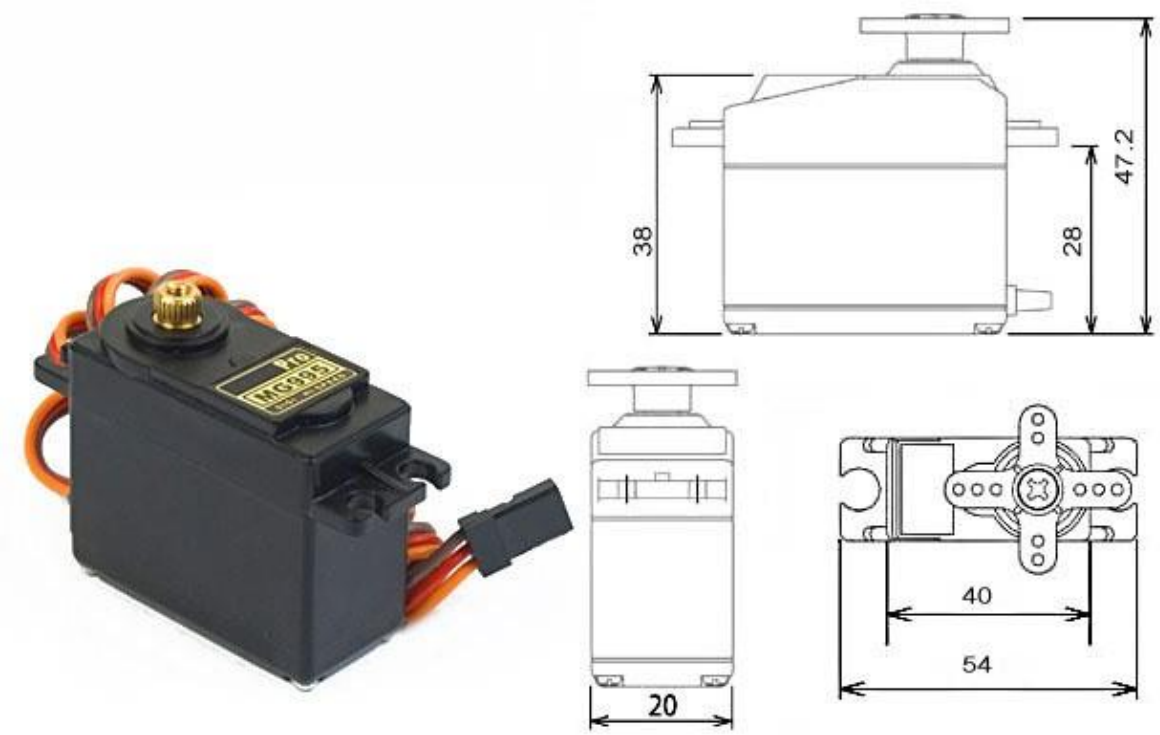
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

6. ANEXOS

Anexo 01 *Towerpro MG995.*

MG995 High Speed
Metal Gear Dual Ball Bearing Servo



The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

This high-speed standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG995 Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

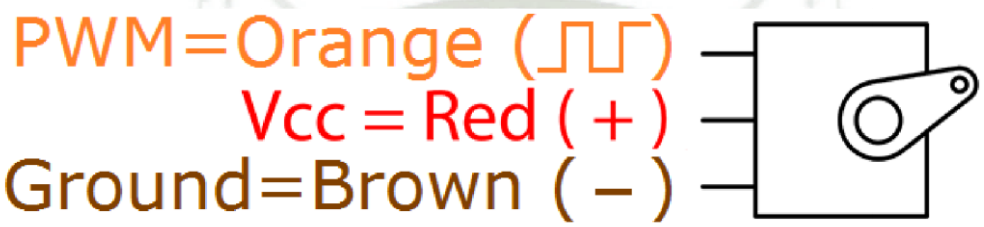
- Weight: 55 g

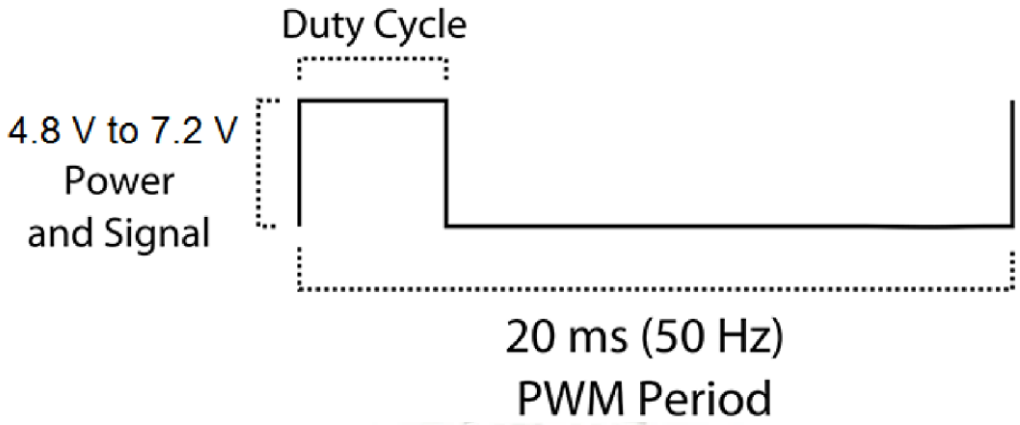
Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 8.5 kgf·cm (4.8 V), 10 kgf·cm (6 V)
- Operating speed: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Operating voltage: 4.8 V a 7.2 V
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC





Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

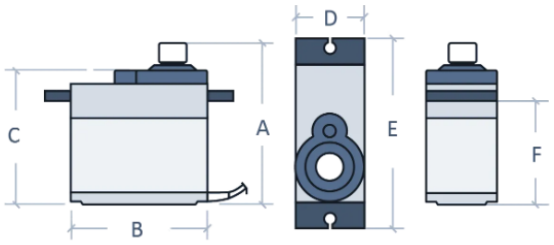
REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Overview

The **MG995 Servo Motor** is an all metal gear servo motor operating on a wide voltage range of **4.8V to 6.6V DC**, it offers exceptional torque of 11 kg/cm, ensuring performance in various applications. Its swift reaction speed (no load speed of 0.20 seconds / 60 degrees at 4.8V and 0.16 seconds / 60 degrees at 6.0V. make it a preferred choice for many applications like Robots, Robotic Arms, RC plane, RC car etc.

Dimensions



A	B	C	D	E	F
42.7	40.9	37	20	54	26.8

Specifications

Voltage	4.8V to 6.6V DC
Torque	9.4 kg/cm at 4.8V 11 kg/cm at 6.0V
Speed	0.20 seconds / 60 degrees at 4.8V 0.16 seconds / 60 degrees at 6.0V
Current	10 mA (Idle Current) 170 mA (No Load Current) 1200 mA (Stall Current)
Gear Type	Metal
Cable Length	30 cm
Plug / Connector	JR FUTABA Universal
Operating Temperature	0°C to 55°C
Weight	55g

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Anexo 02 *Sensor ultrasónico HCSR04.*

HCSR04 Ultrasonic Sensor

Elijah J. Morgan Nov. 16 2014

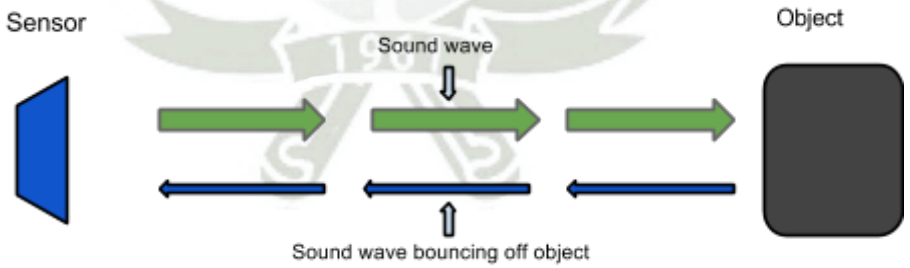
The purpose of this file is to explain how the HCSR04 works. It will give a brief explanation of how ultrasonic sensors work in general. It will also explain how to wire the sensor up to a microcontroller and how to take/interpret readings. It will also discuss some sources of errors and bad readings.



1. **How Ultrasonic Sensors Work**
2. **HCSR04 Specifications**
3. **Timing chart, Pin explanations and Taking Distance Measurements**
4. **Wiring HCSR04 with a microcontroller**
5. **Errors and Bad Readings**

1. How Ultrasonic Sensors Work

Ultrasonic sensors use sound to determine the distance between the sensor and the closest object in its path. How do ultrasonic sensors do this? Ultrasonic sensors are essentially sound sensors, but they operate at a frequency above human hearing.



The sensor sends out a sound wave at a specific frequency. It then listens for that specific sound wave to bounce off of an object and come back (Figure 1). The sensor keeps track of the time between sending the sound wave and the sound wave returning. If you know how fast something is going and how long it is traveling you can find the distance traveled with equation 1.

Equation 1. $d = v \times t$

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

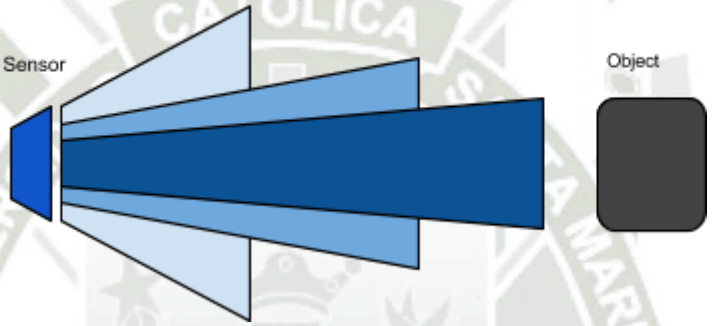
REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

The speed of sound can be calculated based on the a variety of atmospheric conditions, including temperature, humidity and pressure. Actually calculating the distance will be shown later on in this document.

It should be noted that ultrasonic sensors have a cone of detection, the angle of this cone varies with distance, Figure 2 show this relation. The ability of a sensor to detect an object also depends on the objects orientation to the sensor. If an object doesn't present a flat surface to the sensor then it is possible the sound wave will bounce off the object in a way that it does not return to the sensor.

Figure 2



2. HCSR04 Specifications

The sensor chosen for the Firefighting Drone Project was the HCSR04. This section contains the specifications and why they are important to the sensor module. The sensor modules requirements are as follows.

- Cost
- Weight
- Community of hobbyists and support
- Accuracy of object detection
- Probability of working in a smoky environment
- Ease of use

The HCSR04 Specifications are listed below. These specifications are from the Cytron Technologies HCSR04 User's Manual (source 1).

- Power Supply: +5V DC
- Quiescent Current: <2mA

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

- Working current: 15mA
- Effectual Angle: <15º
- Ranging Distance: 2400 cm
- Resolution: 0.3 cm
- Measuring Angle: 30º
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm
- Weight: approx. 10 g

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Anexo 03 *Sensor infrarrojo TCRT5000*

Reflective Optical Sensor with Transistor Output

Description

The TCRT5000(L) has a compact construction where the emitting-light source and the detector are arranged in the same direction to sense the presence of an object by using the reflective IR beam from the object.

The operating wavelength is 950 mm. The detector consists of a phototransistor.

Applications

Position sensor for shaft encoder

Detection of reflective material such as paper, IBM cards, magnetic tapes etc.

Limit switch for mechanical motions in VCR 94 9442

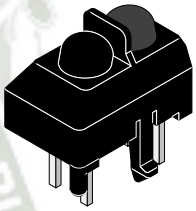
General purpose – wherever the space is limited

Features

Snap-in construction for PCB mounting

Package height: 7 mm

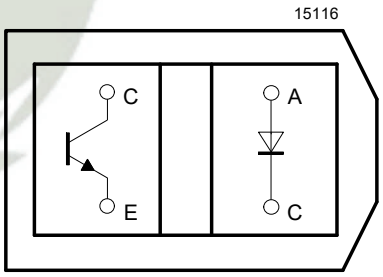
Plastic polycarbonate housing construction which prevents crosstalk



Order = long leads

Current **T**ransfer **R**atio (CTR) of typical 10% Top view

Instruction



Ordering Code	Sensing Distance	Remarks
TCRT5000	12 mm	Leads (3.5 mm)
TCRT5000(L)	12 mm	Long leads (15 mm)

Document Number 83760

Rev. A4, 03–Jul–00

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Absolute Maximum Ratings

Input (Emitter)

Parameter	Test Conditions	Symbol	Value	Unit
Reverse voltage		VR	5	V
Forward current		IF	60	mA
Forward surge current	$t_p \leq 10$ A	IFSM	3	A
Power dissipation	$T_{amb} \leq 25$ C	PV	100	mW
Junction temperature		T_j	100	C

Output (Detector)

Parameter	Test Conditions	Symbol	Value	Unit
Collector emitter voltage		VCEO	70	V
Emitter collector voltage		VECO	5	V
Collector current		IC	100	mA
Power dissipation	$T_{amb} \leq 55$ C	PV	100	mW
Junction temperature		T_j	100	C

Sensor

Parameter	Test Conditions	Symbol	Value	Unit
Total power dissipation	$T_{amb} \leq 25$ C	Ptot	200	mW
Operation temperature range		Tamb	–25 to +85	C
Storage temperature range		Tstg	–25 to +100	C
Soldering temperature	2 mm from case, t ≤ 10 s	Tsd	260	C

Electrical Characteristics (T_{amb} = 25°C)

Input (Emitter)

Parameter	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Forward voltage	I_F = 60 mA	V_F		1.25	1.5	V
Junction capacitance	V_R = 0 V, f = 1 MHz	C_j		50		pF

Output (Detector)

Parameter	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Collector emitter voltage	I_C = 1 mA	VCEO	70			V
Emitter collector voltage	I_E = 100 A	VECO	7			V
Collector dark current	V_{CE} = 20 V, I_F = 0, E = 0	ICEO		10	200	nA

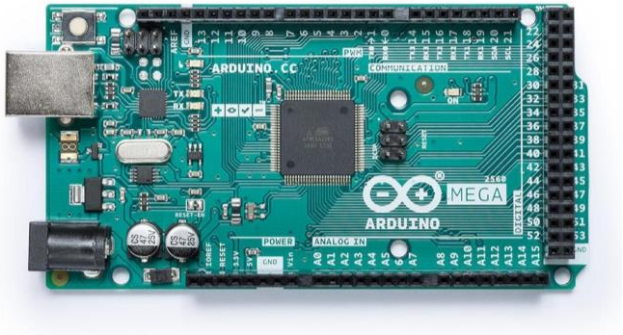
Sensor

Parameter	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Collector current	V_{CE} = 5 V, I_F = 10 mA, D = 12 mm	^{I}C 1,2)	0.5	1	2.1	mA
Collector emitter saturation voltage	I_F = 10 mA, I_C = 0.1 mA, D = 12 mm	VCEsat 1,2)			0.4	V

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Anexo 04 *Arduino mega 2560 rev3.*



Description

Arduino® Mega 2560 Rev3 is an exemplary development board dedicated for building extensive applications as compared to other maker boards by Arduino. The board accommodates the ATmega2560 microcontroller, which operates at a frequency of 16 MHz. The board contains 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a USB connection, a power jack, an ICSP header, and a reset button.

Target Areas

- 3D Printing, Robotics, Maker

Features ATmega2560 Processor

- Up to 16 MIPS Throughput at 16MHz
- 256k bytes (of which 8k is used for the bootloader)
- 4k bytes EEPROM
- 8k bytes Internal SRAM
- 32 × 8 General Purpose Working Registers
- Real Time Counter with Separate Oscillator
- Four 8-bit PWM Channels
- Four Programmable Serial USART
- Controller/Peripheral SPI Serial Interface

ATmega16U2

- Up to 16 MIPS Throughput at 16 MHz
- 16k bytes ISP Flash Memory
- 512 bytes EEPROM
- 512 bytes SRAM
- USART with SPI master only mode and hardware flow control (RTS/CTS) Master/Slave

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

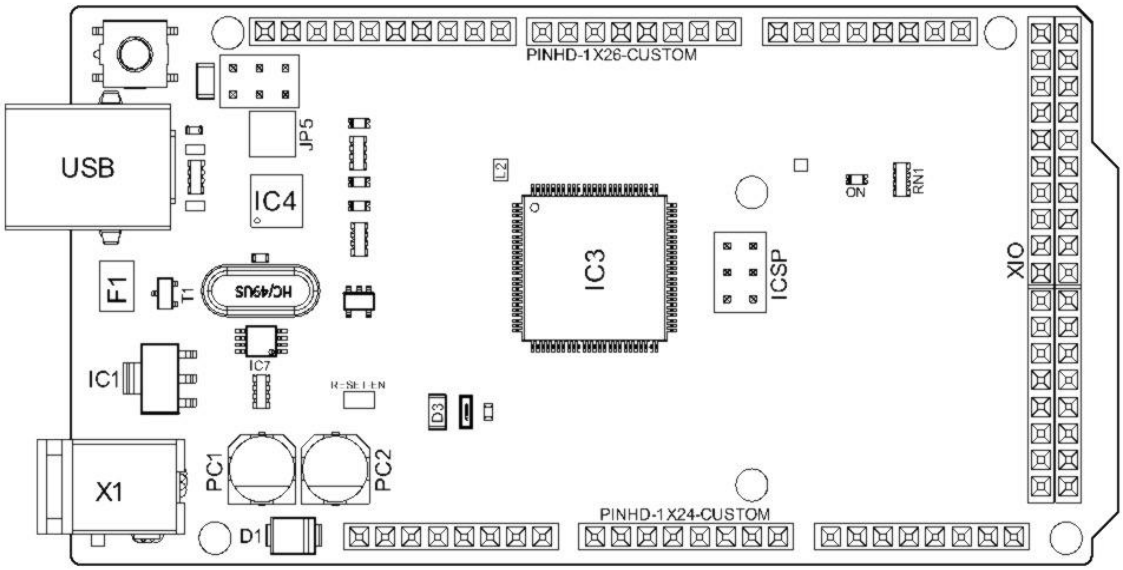
SPI Serial Interface

Sleep Modes

- Idle
- ADC Noise Reduction
- Power-save
- Power-down
- Standby
- Extended Standby

Power

- USB Connection
- External AC/DC Adapter
- 54 Digital
- 16 Analog
- 15 PWM Output



Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Arduino Mega 2560 Rev3 Top View

Ref.	Description	Ref.	Description
USB	USB B Connector	F1	Chip Capacitor
IC1	5V Linear Regulator	X1	Power Jack Connector
JP5	Plated Holes	IC4	ATmega16U2 chip
PC1	Electrolytic Alumninum Capacitor	PC2	Electrolytic Alumninum Capacitor
D1	General Purpose Rectifier	D3	General Purpose Diode
L2	Fixed Inductor	IC3	ATmega2560 chip
ICSP	Connector Header	ON	Green LED
RN1	Resistor Array	XIO	Connector

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Anexo 05 *Relay Songle Relay*

SONGLE RELAY

松乐继电器 ® SONGLE RELAY	RELAY ISO9002	**SRD**



1. MAIN FEATURES

- Switching capacity available by 10A in spite of small size design for highdensity P.C. board mounting technique.
- UL,CUL,TUV recognized.
- Selection of plastic material for high temperature and better chemical solution performance.
- Sealed types available.
- Simple relay magnetic circuit to meet low cost of mass production.

2. APPLICATIONS

- Domestic appliance, office machine, audio, equipment, automobile, etc.

  (Remote control TV receiver, monitor display, audio equipment high rushing current use application.)

3. ORDERING INFORMATION

SRD	XX VDC	S	L	C
Model of relay	Nominal coil voltage	Structure	Coil sensitivity	Contact form
SRD	03 ▼ 05 ▼ 06 ▼ 09 ▼ 12 ▼ 24 ▼ 48VDC	S:Sealed type	L:0.36W	A:1 form A
				B:1 form B
		F:Flux free type	D:0.45W	C:1 form C

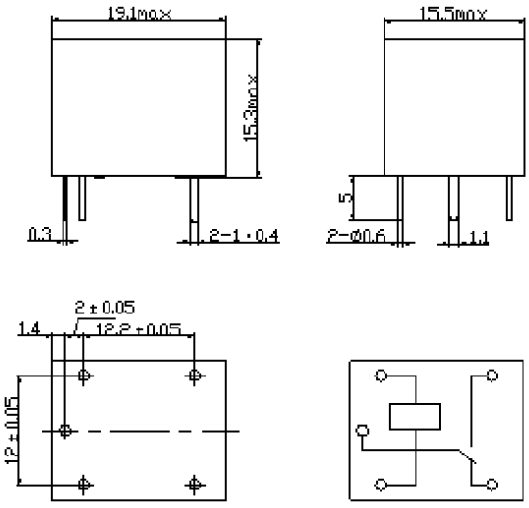
4. RATING

CCC	FILE NUMBER:	CQC03001003731	10A/250VDC
UL /CUL	FILE NUMBER:	E167996	10A/125VAC 28VDC
TUV	FILE NUMBER	R 50056114	10A/250VAC 30VDC

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

5. DIMENSION(unit:mm) DRILLING(unit:mm) WIRING DIAGRAM



COIL DATA CHART (AT20°C)

Coil Sensitivity	Coil Voltage Code	Nominal Voltage (VDC)	Nominal Current (mA)	Coil Resistance (Ω) ±10%	Power Consumption (W)	Pull-In Voltage (VDC)	Drop-Out Voltage (VDC)	Max-Allowable Voltage (VDC)
SRD (High Sensitivity)	03	03	120	25	abt. 0.36W	75%Max.	10% Min.	120%
	05	05	71.4	70				
	06	06	60	100				
	09	09	40	225				
	12	12	30	400				
	24	24	15	1600				
	48	48	7.5	6400				
SRD (Standard)	03	03	150	20	abt. 0.45W	75% Max.	10% Min.	110%
	05	05	89.3	55				
	06	06	75	80				
	09	09	50	180				
	12	12	37.5	320				
	24	24	18.7	1280				
	48	48	10	4500	abt. 0.51W			

6. CONTACT RATING

Type / Item	SRD	
	FORM C	FORM A
Contact Capacity Resistive Load (cosΦ=1)	10A 125VAC	10A 30VDC 10A 250VAC
Inductive Load (cosΦ=0.4 L/R=7msec)	3A 120VAC 3A 28VDC	5A 120VAC 5A 28VDC
Max. Allowable Voltage	250VAC/110VDC	250VAC/110VDC
Max. Allowable Power Force	800VAC/240W	1200VA/300W
Contact Material	AgCdO	AgCdO

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

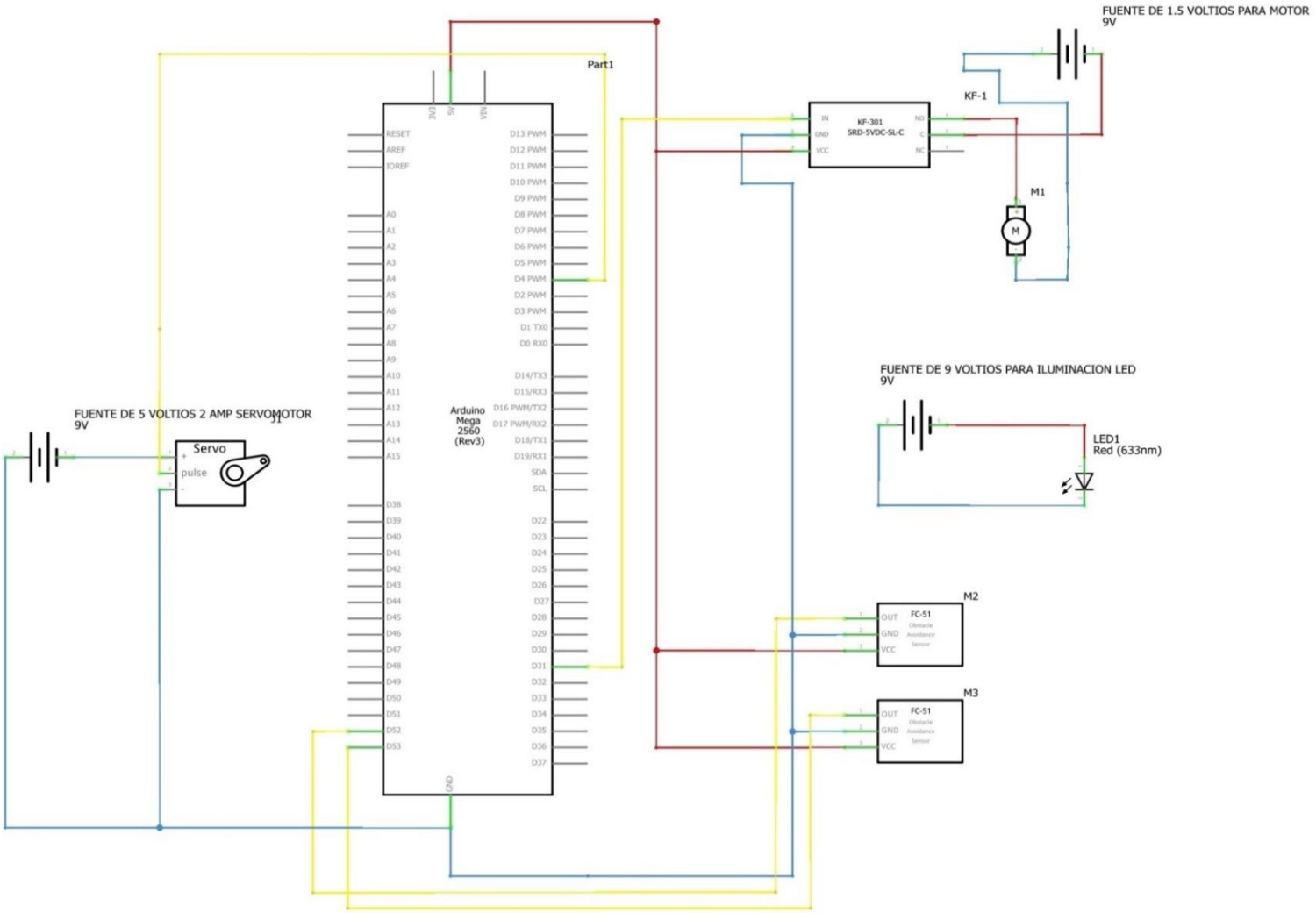
Type Item	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength Between coil & contact Between contacts	 1500VAC 50/60HZ (1 minute) 1000VAC 50/60HZ (1 minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching Mechanically Electrically	 300 operation/min 30 operation/min
Ambient Temperature	-40°C to +85°C
Operating Humidity	45 to 85% RH
Vibration Endurance Error Operation	 10 to 55Hz Double Amplitude 1.5mm 10 to 55Hz Double Amplitude 1.5mm
Shock Endurance Error Operation	 100G Min. 10G Min.
Life Expectancy Mechanically Electrically	 10^7 operations. Min. (no load) 10^5 operations. Min. (at rated coil voltage)
Weight	abt. 10grs.

Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

Anexo 06 *Diagrama de conexión arduino*



Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Anexo 07 *Codificación arduino.*

```
#include <Servo.h>

// Pines
const int releFaja = 31;
const int ir1Pin = 53;
const int ir2Pin = 52;
const int servoPin = 4;

// Estados
Servo miServo;
int anguloActual = 70;
bool esperandoClasificacion = false;
bool esperandoRespuesta = false;
bool ir1Habilitado = true;
String clase = "";

void setup() {
  Serial.begin(9600);
  pinMode(releFaja, OUTPUT);
  pinMode(ir1Pin, INPUT);
  pinMode(ir2Pin, INPUT);

  miServo.attach(servoPin);
  miServo.write(anguloActual);
  digitalWrite(releFaja, HIGH); // Faja encendida al inicio
}

void loop() {
  // --- DETECCIÓN CON IR1 ---
  if (ir1Habilitado && digitalRead(ir1Pin) == LOW && !esperandoClasificacion)
{
    digitalWrite(releFaja, LOW);       // Detiene la faja
    Serial.println("CAM_ON");          // Solicita clasificación
    esperandoClasificacion = true;
    esperandoRespuesta = true;
    ir1Habilitado = false;             // Desactiva IR1 hasta que IR2 lo
reactive
  }

  // --- RECEPCIÓN DE CLASE DESDE PYTHON ---
  if (esperandoRespuesta && Serial.available()) {
    String dato = Serial.readStringUntil('\n');
    dato.trim();

    if (dato == "MQ1" || dato == "MQ2" || dato == "NQ1" || dato == "NQ2") {
      clase = dato;
      esperandoRespuesta = false;
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
Serial.println("CAM_OFF");

// FRUTA BUENA – MQ1 / NQ1
if (clase == "MQ1" || clase == "NQ1") {
  digitalWrite(releFaja, HIGH);  // Faja avanza sin detenerse
  Serial.println("Fruta buena detectada. Solo avanza la faja.");

  // Esperar a que IR1 quede libre antes de reactivarlo
  while (digitalRead(ir1Pin) == LOW);
  delay(100);
  ir1Habilitado = true;

  clase = "";
  esperandoClasificacion = false;
}

// FRUTA MALA – MQ2 / NQ2
else if (clase == "MQ2" || clase == "NQ2") {
  digitalWrite(releFaja, HIGH);  // Faja avanza hacia el servo
  Serial.println("Fruta en mal estado. Esperando detección en IR2");

  // Esperar a que IR2 esté libre primero
  while (digitalRead(ir2Pin) == LOW);

  // Esperar a que IR2 detecte la fruta (LOW)
  while (digitalRead(ir2Pin) == HIGH);
  Serial.println("IR2 detectó fruta mala. Esperando 0.5s");
  delay(500);

  digitalWrite(releFaja, LOW);  // Detiene la faja

  // Accionar el servomotor para clasificacion mal estado 70°, 180°, 70°
  Serial.println("Servo clasificando fruta mala...");
  miServo.write(180);
  delay(1000);
  miServo.write(70);
  delay(1000);
  anguloActual = 70;

  digitalWrite(releFaja, HIGH);  // Reanudar faja
  Serial.println("Faja reactivada. Esperando que fruta pase IR2...");

  // Esperar que fruta se retire del IR2
  while (digitalRead(ir2Pin) == LOW);  // Esperar que se vaya
  delay(100);

  // Esperar que IR1 esté libre
  while (digitalRead(ir1Pin) == LOW);
  delay(100);
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
            ir1Habilitado = true;

            clase = "";
            esperandoClasificacion = false;
          }
        }
      }
    }
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Anexo 08 *Codificación spyder*

```
import cv2
import numpy as np
from keras.models import load_model
from keras.applications.mobilenet import preprocess_input
import time
import serial
import os

# ------------------ CONFIGURACIÓN INICIAL ------------------
arduino = serial.Serial('COM3', 9600, timeout=1)
time.sleep(2)

ruta_modelo = r'C:\PROYECTO DE
TESIS\modelo\modelo\nativa\myn\modelo_myn_mobilenetv2_VariandoDropout.h5'
modelo = load_model(ruta_modelo)

clases = ['MQ1', 'MQ2', 'NQ1', 'NQ2']
colores = {
    'MQ1': (255, 0, 0), 'MQ2': (0, 255, 0),
    'NQ1': (255, 255, 0), 'NQ2': (0, 255, 255)
}

roi_path = "roi_config.txt"

# ------------------ FUNCIONES ------------------
def guardar_roi(roi):
    with open(roi_path, 'w') as f:
        f.write(','.join(map(str, roi)))
    print("ROI guardado:", roi)

def cargar_roi():
    if os.path.exists(roi_path):
        with open(roi_path, 'r') as f:
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

```
            valores = f.read().strip().split(',')
            if len(valores) == 4:
                return list(map(int, valores))
    return [250, 500, 150, 650]  # valor por defecto

def preprocesar_imagen(fotograma, roi):
    fotograma_roi = fotograma[roi[0]:roi[1], roi[2]:roi[3]]
    if fotograma_roi.size == 0:
        print("ROI vacío, revisa coordenadas:", roi)
        return None
    imagen_redimensionada = cv2.resize(fotograma_roi, (160, 160))
    array_imagen = np.expand_dims(np.array(imagen_redimensionada, dtype=np.float32),
axis=0)
    return preprocess_input(array_imagen)

def mostrar_texto(marco, texto, posicion, tamano=1, color=(255, 0, 0), grosor=2):
    cv2.putText(marco, texto, posicion, cv2.FONT_HERSHEY_SIMPLEX, tamano, color,
grosor)

# ------------------ CARGA DE VIDEO Y ROI ------------------
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error al abrir la cámara.")
    exit()

roi = cargar_roi()
ajustando_roi = True

print("🎮 Ajuste de ROI (WASD para mover, IJKL para escalar, ENTER para guardar)")

while ajustando_roi:
    ret, frame = cap.read()
    if not ret:
        break
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
    frame_copia = frame.copy()
    cv2.rectangle(frame_copia, (roi[2], roi[0]), (roi[3], roi[1]), (0, 255, 255), 2)
    mostrar_texto(frame_copia, "Ajustando ROI...", (10, 30), 1, (0, 255, 255), 2)
    cv2.imshow("Ajuste de ROI", frame_copia)

    key = cv2.waitKey(0) & 0xFF
    if key == ord('w'): roi[0] -= 5; roi[1] -= 5
    elif key == ord('s'): roi[0] += 5; roi[1] += 5
    elif key == ord('a'): roi[2] -= 5; roi[3] -= 5
    elif key == ord('d'): roi[2] += 5; roi[3] += 5
    elif key == ord('i'): roi[1] += 5
    elif key == ord('k'): roi[1] -= 5
    elif key == ord('j'): roi[3] += 5
    elif key == ord('l'): roi[3] -= 5
    elif key == 13:  # ENTER
        guardar_roi(roi)
        ajustando_roi = False
    elif key == ord('q'):
        print("Cancelado por el usuario.")
        cap.release()
        cv2.destroyAllWindows()
        exit()

# ------------------ CICLO PRINCIPAL ------------------
mostrar = False
print("Sistema iniciado. Esperando CAM_ON...")

try:
    while True:
        ret, fotograma = cap.read()
        if not ret:
            break
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
# Leer señal desde Arduino
if arduino.in_waiting:
    linea = arduino.readline().decode().strip()
    if linea == "CAM_ON":
        mostrar = True
        print("Clasificación activada.")

        tiempo_inicio = time.time()
        enviado = False
        confirmaciones = 0
        clase_anterior = ""

        while time.time() - tiempo_inicio < 3.0:
            ret, fotograma = cap.read()
            if not ret:
                continue

            imagen_preprocesada = preprocesar_imagen(fotograma, roi)
            if imagen_preprocesada is None:
                continue

            predicciones = modelo.predict(imagen_preprocesada)
            clase_predicha = np.argmax(predicciones)
            probabilidad = predicciones[0][clase_predicha] * 100
            etiqueta = clases[clase_predicha]
            color = colores.get(etiqueta, (0, 255, 0))


            if etiqueta == clase_anterior and probabilidad > 70:

                confirmaciones += 1
            else:
                confirmaciones = 1
                clase_anterior = etiqueta
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

```
                if confirmaciones >= 3:
                    mensaje = etiqueta + '\n'
                    arduino.write(mensaje.encode())
                    print(f"Enviado al Arduino: {mensaje.strip()}")
                    enviado = True
                    break

                if mostrar:
                    cv2.rectangle(fotograma, (roi[2], roi[0]), (roi[3], roi[1]), color, 2)
                    mostrar_texto(fotograma, f'Clase: {etiqueta}', (10, 30), 1, color, 2)
                    mostrar_texto(fotograma, f'Probabilidad: {probabilidad:.2f}%', (10, 70), 1,
color, 2)

                cv2.imshow("Clasificador en Tiempo Real", fotograma)

                if cv2.waitKey(1) & 0xFF == ord('q'):
                    raise KeyboardInterrupt

            if not enviado:
                print("No se detectó fruta válida. Se omite acción.")
                arduino.write(b"MQ2\n")  # Asumir como fruta en mal estado si nada se clasificó

        elif linea == "CAM_OFF":
            mostrar = False
            print("Clasificación terminada. Ocultando ROI.")

    # Mostrar fuera de CAM_ON si se permite
    if mostrar:
        imagen_preprocesada = preprocesar_imagen(fotograma, roi)
        if imagen_preprocesada is not None:
            predicciones = modelo.predict(imagen_preprocesada)
            clase_predicha = np.argmax(predicciones)
            probabilidad = predicciones[0][clase_predicha] * 100
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
            etiqueta = clases[clase_predicha]
            color = colores.get(etiqueta, (0, 255, 0))

            cv2.rectangle(fotograma, (roi[2], roi[0]), (roi[3], roi[1]), color, 2)
            mostrar_texto(fotograma, f'Clase: {etiqueta}', (10, 30), 1, color, 2)
            mostrar_texto(fotograma, f'Probabilidad: {probabilidad:.2f}%', (10, 70), 1, color, 2)

        cv2.imshow("Clasificador en Tiempo Real", fotograma)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

except KeyboardInterrupt:
    print("Proceso interrumpido por el usuario.")

finally:
    cap.release()
    arduino.close()
    cv2.destroyAllWindows()
    print("✅ Recursos liberados. Fin del programa.")#include <Servo.h>
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM
UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Anexo 09 *Entrenamiento de red en Jupyter para modelo con arquitectura mobilenetv2*

```
# Imports
import matplotlib.pyplot as plt #PERMITE GRAFICAR EL ENTRENAMIENTO DE LA RED EN LAS EPOCAS.
import numpy as np
import os
import cv2
import itertools
import math
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import confusion_matrix #NECESARIO PARA REALIZAR LA MATRIZ DE CONFUSION
# Modelo base
base_model = MobileNetV2(input_shape=(160, 160, 3), include_top=False, weights='imagenet')
base_model.trainable = False #Congelamos Las capas preentrenadas
# Modelo final
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dropout(0.5),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(4, activation='softmax')  # MQ1, MQ2, NQ1, NQ2
])

# Compilamos
model.compile(
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE TESIS UCSM

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

```
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
# Mostrar la arquitectura del modelo
model.summary()
# Preprocesamiento de datos
train_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
val_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator = train_datagen.flow_from_directory(
    'C:/PROYECTO DE TESIS/modelo/modelo/data_myn/entrenamiento',
    target_size=(160, 160),
    batch_size=16,
    class_mode='categorical',
    shuffle=True
)

valid_generator = val_datagen.flow_from_directory(
    'C:/PROYECTO DE TESIS/modelo/modelo/data_myn/validacion',
    target_size=(160, 160),
    batch_size=16,
    class_mode='categorical',
    shuffle=False
)
# Early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
# Entrenamiento
TOTAL_EPOCAS = 10
step_train = math.ceil(train_generator.n / train_generator.batch_size)
step_valid = math.ceil(valid_generator.n / valid_generator.batch_size)

H = model.fit(
    train_generator,
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
    steps_per_epoch=step_train,
    epochs=TOTAL_EPOCAS,
    validation_data=valid_generator,
    validation_steps=step_valid,
    callbacks=[early_stop]
)
# Evaluación final
model.evaluate(train_generator, steps=step_train)
N = TOTAL_EPOCAS
plt.style.use("ggplot")
plt.figure()
plt.plot(H.history["loss"], label="train_loss")
plt.plot(H.history["val_loss"], label="val_loss")
plt.plot(H.history["accuracy"], label="train_acc")
plt.plot(H.history["val_accuracy"], label="val_acc")
plt.title("Error y exactitud en entrenamiento")
plt.xlabel("Epoca #")
plt.ylabel("Error/Exactitud")
plt.legend(loc="lower left")
plt.savefig("C:/PROYECTO DE
TESIS/modelo/modelo/nativa/myn/cnn_estadisticas_de_la_red_mobilenetv2.png")
#UNA VEZ ENTRENADA LA RED SE PRUEBA LA EFECTIVIDAD DE ESTA
# damos formato a la data de pruebas
test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

test_generator = test_datagen.flow_from_directory(
    directory="C:/PROYECTO DE TESIS/modelo/modelo/data_myn/pruebas/",
    target_size=(160, 160),
    color_mode="rgb",
    batch_size=1,
    class_mode=None,
    shuffle=False,
    seed=42
)
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size
test_generator.reset()
#llamamos a la funcion de prediccion, aplicamos elemento a elemento, se envia data de 1 en
1.
pred = model.predict(test_generator, steps=STEP_SIZE_TEST, verbose=1) #calculamos el
array de prediccion
pred.round(2) #array de predicciones-tipo flotante
predicted_class_indices=np.argmax(pred,axis=1) #lista de predicciones en tipo entero
print (predicted_class_indices) #mostramos lista de predicciones
print(type(predicted_class_indices))
# Obtenemos los índices de las clases del generador de datos de entrenamiento
(train_generator).
labels = (train_generator.class_indices)
labels = dict((v, k) for k, v in labels.items())
predictions = [labels[k] for k in predicted_class_indices]
#mostramos las imagenes de prueba con su prediccion
folder = r"C:\PROYECTO DE TESIS\modelo\modelo\data_myn\pruebas\test_images"

# Obtener la lista de archivos
filenames = os.listdir(folder)

# Configurar la figura
fig = plt.figure(figsize=(30, 30))
fig.subplots_adjust(hspace=0.1, wspace=0.1)

rows = 10
cols = len(filenames) // rows
if len(filenames) % rows > 0:
    cols += 1

# Mostrar imágenes
for i, filename in enumerate(filenames):
    img_path = os.path.join(folder, filename)  # Construir ruta completa
    img = cv2.imread(img_path)
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
    if img is not None:
        ax = fig.add_subplot(rows, cols, i + 1)
        ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))  # Convertir de BGR a RGB
        ax.axis("off")
        ax.set_title(filename, fontsize=16)
predicted_class_indices
my_dict = {
    'MQ1': 0,
    'MQ2': 1,
    'NQ1': 2,
    'NQ2': 3,

}
real_class_indices=[]
for filename in filenames:
    # Obtenemos el nombre del archivo
    basename = filename.split(os.sep)[-1].lower()

    found_class = None

    # Recorremos cada estado y su índice
    for key, val in my_dict.items():
        if key.lower() in basename:
            found_class = val
            break  # No hace falta seguir buscando

    # Guardamos la clase que encontramos
    if found_class is not None:
        real_class_indices.append(found_class)
    else:
        # si alguna prediccion no hace match se le pondra -1
        real_class_indices.append(-1)
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
print(real_class_indices)
real_class_indices = np.array(real_class_indices)
cm = confusion_matrix(real_class_indices, predicted_class_indices)
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Matriz de confusión normalizada")
    else:
        print('Matriz de confusión, sin normalización')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('Etiqueta real')
    plt.xlabel('Etiqueta predecida')
cm_plot_labels = train_generator.class_indices
print (cm_plot_labels)
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis

REPOSITORIO DE
TESIS UCSM

UNIVERSIDAD
CATÓLICA
DE SANTA MARÍA

```
key_list = list(cm_plot_labels.keys())
val_list = list(cm_plot_labels.values())
plot_confusion_matrix(cm, cm_plot_labels, title= 'Matriz de confusion')
from mlxtend.evaluate import scoring #incluye diversas funciones para evaluar el rendimiento
de los modelos.

res = scoring(y_target=real_class_indices, y_predicted=predicted_class_indices,
metric='error')

print('Error:',np.round(res * 100,2),"%")

res = scoring(y_target=real_class_indices, y_predicted=predicted_class_indices,
metric='accuracy')
print ("Exactitud:",np.round(res*100,2),"%")
from sklearn.metrics import classification_report #generar reporte que muestra las principales
métricas de clasificación
print(classification_report(predicted_class_indices,real_class_indices))
model.save(r'C:\PROYECTO DE
TESIS\modelo\modelo\nativa\myn\modelo_myn_mobilenetv2_VariandoDropout.h5')
```


Publicación autorizada con fines académicos e investigativos
En su investigación no olvide referenciar esta tesis