



UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

LICENCIATURA EN INGENIERÍA EN ELECTRÓNICA

PROYECTO TERMINAL

**SISTEMA CLASIFICADOR DE MANZANAS EN BUEN Y MAL ESTADO EMPLEANDO
UNA RED NEURONAL CONVOLUCIONAL (CNN)**

Para obtener el grado de
Ingeniero en Electrónica

PRESENTA

José Antonio García García

Director:

M. en C .Omar Samperio Vázquez

Codirector:

Dr Kristian Freyri Maya Gress

Pachuca de Soto, Hgo., México 17 de junio de 2026



Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

School of Engineering and Basic Sciences

Dirección

Office of the Director

Mineral de la Reforma, Hgo., a 4 de junio de 2026

Número de control: ICBI-D/1058/2026

Asunto: Autorización de impresión.

MTRA. OJUKY DEL ROCÍO ISLAS MALDONADO
DIRECTORA DE ADMINISTRACIÓN ESCOLAR DE LA UAEH

Con Título Quinto, Capítulo II, Capítulo V, Artículo 51 Fracción IX del Estatuto General de nuestra Institución, por este medio, le comunico que el Jurado asignado al egresado de la Licenciatura en Ingeniería en Electrónica **José Antonio García García**, quien presenta el trabajo de titulación "**SISTEMA CLASIFICADOR DE MANZANAS EN BUEN Y MAL ESTADO EMPLEANDO UNA RED NEURONAL CONVOLUCIONAL (CNN)**", ha decidido, después de revisar fundamento en lo dispuesto en el Título Tercero, Capítulo I, Artículo 18 Fracción IV; dicho trabajo en la reunión de sinodales, **autorizar la impresión del mismo**, una vez realizadas las correcciones acordadas.

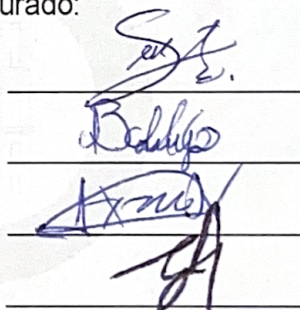
A continuación, firman de conformidad los integrantes del Jurado:

Presidente: Dr. Juan Carlos González Islas

Secretario: Ing. Benigno Martínez Gutiérrez

Vocal: M. en C Omar Samperio Vázquez

Suplente: Dr. Kristian Freyri Maya Gress



Sin otro particular por el momento, reciba un cordial saludo.

Atentamente
"Amor, Orden y Progreso"

Dr. Gabriel Vergara Rodríguez
Director del ICBI



GVR/MMM

"Amor, Orden y Progreso"



Ciudad del Conocimiento, Carretera Pachuca-Tulancingo
Km. 4.5 Colonia Carboneras, Mineral de la Reforma, Hidalgo,
México. C.P 42184

Teléfono: 52 (771) 71 720 00 Ext. 40001

direccion_icbi@uaeh.edu.mx, vergara@uaeh.edu.mx

uaeh.edu.mx



José Antonio García García

**SISTEMA CLASIFICADOR DE
MANZANAS EN BUEN Y MAL
ESTADO EMPLEANDO UNA RED
NEURONAL CONVOLUCIONAL
(CNN)**

Gracias a la

UAEH

UAEH, por la ardua formación académica que se me fue impartida durante mi estancia en la carrera, de igual modo por el fortalecimiento y la determinación adquirida por parte de los profesores, amigos de clases, y mi familia que han tocado y dejado huella a mi vida, ya que sin su apoyo todo este tiempo no hubiera sido posible la realización de la presente memoria.

Índice general

Índice de figuras	VI
Índice de cuadros	VIII
1. Introducción	1
1.1. Resumen	1
1.2. Abstract	2
1.3. Introducción	3
1.4. Antecedentes	6
1.5. Planteamiento del problema	8
1.6. Objetivos	9
1.6.1. Objetivo general	9
1.6.2. Objetivos específicos	9
1.7. Justificación	10
1.8. Hipótesis	11
1.9. Estado del arte	12
1.10. Benchmarking	24
1.10.1. Primeras apariciones	25
1.10.2. Descripción	27
2. Marco teórico	29
2.1. Marco Teórico	29
2.1.1. Automatización	29
2.1.2. Automatización Fija (Rígida)	30
2.1.3. Automatización Programable	30
2.1.4. Automatización Flexible	31
2.1.5. Visión artificial	32
2.1.6. Imágenes digitales	32
2.1.7. Resolución y píxeles	33
2.1.8. Procesamiento digital de imágenes	34
2.1.9. Inteligencia artificial	34
2.1.10. Redes neuronales artificiales	35

2.1.11. Redes neuronales convolucionales (CNN)	36
2.1.12. Sistemas embebidos	36
2.1.13. Aplicación de la visión artificial y la IA en la agricultura	37
2.2. Generalidades del sistema	38
2.2.1. Dinámica de funcionamiento de la Raspberry Pi 400	38
2.2.2. Banda transportadora	39
2.2.3. Servomotor SG90	40
2.2.4. Módulo L298N	41
2.2.5. Cámara HD WEBCAM 4K	42
2.2.6. Alimentación (Fuente de alimentación para protoboard v2 MB-102 3.3V 5V)	43
2.2.7. Pantalla LCD	44
2.2.8. Buzzer zumbador	45
3. Diseño y desarrollo del sistema	46
3.1. Marco Metodológico	46
3.1.1. Estructura	47
3.1.2. Soporte tipo arco	47
3.1.3. Soporte tipo resbaladilla	48
3.1.4. Soporte pasa-bandas	49
3.1.5. Diagrama constituido del circuito	49
3.1.6. Circuito	50
3.1.7. Pytorch	50
3.1.8. Red neuronal CNN	50
3.1.9. Entrenamiento	51
3.1.10. Época	54
3.1.11. Programación	54
3.1.12. Diagrama de flujo del funcionamiento.	55
3.1.13. Montaje final	57
3.1.14. Pruebas	59
4. Análisis y discusión de resultados	60
4.1. Resultados	60
5. Conclusiones	65
5.1. Conclusión	65
5.2. Limitaciones	65
5.3. Trabajo a futuro	66
Bibliografía	67
A. Código del clasificador de manzana	74

Índice de figuras

1.1. Implementación de diferentes tipos de automatización en la vida diaria.	3
1.2. Problemas en la industria agrícola y aumento en la población mundial.	4
1.3. Adquisición de datos por medio de visión artificial y ejemplo de estructura de una red neuronal.	4
1.4. Manzanas de entrenamiento y diseño del clasificador.	13
1.5. Clasificación y detección de manzanas.	16
1.6. Proceso de evaluación y precisión de las manzanas.	17
1.7. Tipos de manzanas y estructura de la red neuronal CNN.	18
1.8. Matriz de la confusión.	19
1.9. Diagrama de flujo del sistema.	20
1.10. Clasificación y detección de manzanas.	21
1.11. Contador principal del sistema.	21
1.12. Arquitectura del sistema y detección de la manzanas.	22
1.13. Diferencia entre manzana buena y mala	23
1.14. Origen e historia de las manzanas.	25
1.15. Clasificación manual de manzanas.	26
1.16. Ejemplo de manzana de calidad.	26
1.17. Primera banda transportadora y línea de ensamble de FORD.	27
1.18. Bandas industriales en procesos de producción.	28
2.1. Automatización fija en fábrica embotelladora.	30
2.2. Automatización programable en fábrica de electrodomésticos.	30
2.3. Automatización flexible en fábrica de alimentos de galletas.	31
2.4. Representación de imagen digital.	33
2.5. Procesamiento de una imagen de un camaleón.	34
2.6. Inteligencia artificial en aprendizaje y toma de decisiones.	35
2.7. Ejemplo de sistema embebido de un cajero automático.	37
2.8. Elementos esenciales de la Raspberry pi 400.	38
2.9. Banda transportadora a escala.	39
2.10. Servomotor SG90.	40

2.11. Módulo L298N.	41
2.12. HD WEBCAM 4K.	42
2.13. Fuente de alimentación protoboard.	43
2.14. Pantalla LCD de 16*2.	44
2.15. Buzzer zumbador de 5 volts.	45
3.1. Diagrama a bloques constituido del sistema.	46
3.2. Soporte tipo arco del clasificador.	48
3.3. Resbaladilla del clasificador de manzanas.	48
3.4. Diseño 3D del pasa-bandas.	49
3.5. Diagrama constituido del circuito.	49
3.6. Circuito soldado del sistema del clasificador de manzanas.	50
3.7. Arquitectura de una red neuronal CNN.	51
3.8. Diagrama a bloques del entrenamiento de la CNN.	52
3.9. Diagrama bloques del 1 al 4 del entrenamiento de la CNN.	53
3.10. Diagrama bloques de capa densa y capa de salida de la CNN.	53
3.11. Diagrama de flujo del sistema del clasificador de manzanas.	55
3.12. Parte de inicio del sistema.	56
3.13. Activación del sistema.	56
3.14. Ejecución del sistema.	57
3.15. Finalización total del sistema.	57
3.16. Montaje de la estructura.	58
3.17. Montaje final del sistema.	58
3.18. Manzanas de prueba.	59
4.1. Manzana buena de color verde.	60
4.2. Manzana buena de color amarillo.	61
4.3. Manzana buena de color roja.	61
4.4. Resultados de las manzanas malas.	62
4.5. Manzana buena de color roja.	63
4.6. Resultados mostrados en la pantalla LCD.	63
4.7. Resultados de la clasificación de las manzanas.	64

Índice de cuadros

1.1. Tabla de resultados 1-10.	13
1.2. Tabla de resultados 11-20.	14
1.3. Tabla de resultados 21-30.	14
1.4. Tabla de resultados 31-40.	15
2.1. Tipo de resoluciones.	33

Capítulo 1

Introducción

1.1. Resumen

En el presente proyecto se diseñó y desarrolló un sistema clasificador de manzanas automatizado usando una red neuronal convolucional (CNN), entrenada con imágenes de manzanas y usando técnicas de visión artificial, instrumentándolo en una RASPBERRY PI 400 empleando herramientas como Python, OpenCV y PyTorch.

Como resultado, se obtuvo un prototipo experimental funcional que es capaz de clasificar manzanas en buen y mal estado en tiempo real, realizando su separación automática y mostrando información en las pantallas del LCD y monitor.

El objetivo principal del proyecto es la clasificación de manzanas en buen estado y mal estado dependiendo de sus características visuales y todo esto en tiempo real, así mismo se espera que en un futuro se pueda implementar en la industria agrícola para reducir tiempos muertos y aumentar tanto la productividad como la eficiencia en pequeñas y medianas empresas.

La arquitectura del sistema está compuesta por diferentes elementos electrónicos de fácil accesibilidad y que tiene una función indispensable cada uno, estos elementos son una banda transportadora a escala de 54 cm que tiene como función transportar la manzana, un servomotor de 90 grados que es el encargado de clasificar las manzanas dependiendo de su estado, una cámara WEB que es por donde se usa la visión artificial del sistema, una fuente de voltaje para protoboard de 5 volts para alimentar los componentes, pantalla LCD de 16*2 para indicar el número de manzanas buenas y malas, 2 botones físicos para pausar y continuar el sistema

una vez ejecutado, un buzzer para indicar el inicio del sistema y la cantidad límite de manzanas requeridas, y por último un modulo L298N que controla el motor de la banda transportadora.

Palabras clave: Sistema, red neuronal, librería, visión artificial, RASPBERRY PI 400, clasificación, arquitectura, CNN, elementos electrónicos.

1.2. Abstract

In the present project, an automated apple classification system was designed and developed using a Convolutional Neural Network (CNN), trained with apple images and employing computer vision techniques. The system was implemented on a Raspberry Pi 400 using tools such as Python, OpenCV, and PyTorch.

As a result, a functional experimental prototype was obtained, capable of classifying apples in good and poor condition in real time, performing their automatic separation and displaying information on both the LCD screen and monitor.

The main objective of the project is the real-time classification of apples into good and poor condition depending on their composition. Likewise, it is expected that, in the future, this system may be implemented in the agricultural industry to reduce downtime and increase both productivity and efficiency in small and medium-sized enterprises.

The system architecture is composed of various easily accessible electronic components, each fulfilling an indispensable function. These elements include a 54 cm-scale conveyor belt whose function is to transport the apple; a 90-degree servomotor responsible for classifying apples according to their condition; a webcam through which the system performs computer vision; a 5-volt breadboard power supply used to power the components; a 16*2 LCD screen to indicate the number of good and defective apples; two physical buttons used to pause and resume the system once executed; a buzzer to indicate system startup and the required apple limit; and finally, an L298N module that controls the conveyor belt motor.

Keywords: system, neural network, library, computer vision, Raspberry Pi 400, classification, architecture, CNN, electronic components.

1.3. Introducción

Durante los últimos años, el desarrollo tecnológico ha impactado significativamente en los procesos productivos en diversos sectores industriales, siendo la automatización uno de los principales motores de cambio en la industria. La implementación de sistemas automatizados como se ilustra en la figura 1.1 ha permitido optimizar tareas repetitivas, reducir errores humanos y aumentar la eficiencia operativa, contribuyendo positivamente en áreas como la manufactura, la industria alimentaria, y de manera particular el sector agrícola[1].



(a) Automatización en la vida[2].



(b) Automatización en la industria[3].

Figura 1.1. Implementación de diferentes tipos de automatización en la vida diaria.

Actualmente, la industria agrícola, como se observa en la figura 1.2a se enfrenta a grandes desafíos derivados del crecimiento acelerado de la población mundial como representa la figura 1.2b y del aumento en la demanda de alimentos de calidad. Estos factores han obligado a la modernización de los procesos tradicionales de producción, selección y distribución de dicho sector, con el objetivo de garantizar productos seguros, uniformes y competitivos en el mercado[4]. Dentro de esta cadena productiva, la clasificación de frutas representa una etapa crítica, ya que de ella depende la aceptación del producto por parte del consumidor y el cumplimiento de estándares de calidad[5].



(a) Problemas de demanda en el campo[6].



(b) Aumento de la población mundial[7].

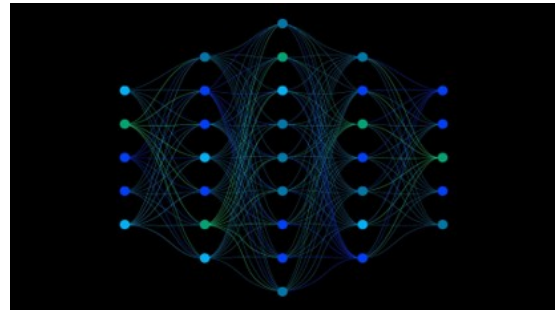
Figura 1.2. Problemas en la industria agrícola y aumento en la población mundial.

Ante este panorama, la automatización, apoyada de tecnologías emergentes como la inteligencia artificial y la visión artificial, surge como una alternativa viable y eficiente para la mejora de los procesos de clasificación[4]. La visión artificial permite la adquisición y el procesamiento de imágenes en tiempo real, mientras que la inteligencia artificial por medio del uso de redes neuronales como se logra apreciar en la figura 1.3, posibilita el análisis y la identificación de patrones complejos, los cuales serían difíciles de detectar de forma manual[8].

En particular, las redes neuronales convolucionales (CNN) demuestran un alto desempeño en tareas de reconocimiento y clasificación de imágenes, siendo ampliamente utilizadas en aplicaciones industriales y agrícolas[9].



(a) Ejemplo de visión artificial[10].



(b) Estructura de una red neuronal[11].

Figura 1.3. Adquisición de datos por medio de visión artificial y ejemplo de estructura de una red neuronal.

El uso de sistemas embebidos de bajo costo y alto rendimiento, como la Raspberry Pi, ha facilitado la implementación de soluciones inteligentes accesibles, capaces de integrar procesamiento de imágenes, control de actuadores y comunicación con dispositivos electrónicos. Esta fusión tecnológica permite desarrollar prototipos funcionales que puedan adaptarse a entornos

reales de producción, ofreciendo una alternativa económica y escalable para la automatización de procesos[8]. Uno de los procesos de la producción agrícola es la clasificación de frutas, pues de esta etapa depende la calidad del producto que llega al consumidor final[12]. Tradicionalmente, este proceso se realiza de forma manual, lo que puede generar errores, inconsistencias y pérdidas económicas debido a factores humanos como la fatiga visual o la falta de uniformidad en los criterios de evaluación[5].

En este contexto, el presente proyecto propone el diseño e implementación de un sistema automatizado que sea capaz de realizar el proceso de clasificación de manzanas en buen y mal estado mediante el uso de una red neuronal convolucional entrenada con imágenes. El sistema integra visión artificial, control electrónico y actuadores mecánicos, utilizando una Raspberry Pi 400 como plataforma principal. El objetivo es desarrollar un prototipo capaz de realizar la clasificación en tiempo real, reduciendo errores humanos, optimizando tiempos de proceso y contribuyendo a la modernización de los sistemas de selección en la industria agrícola.

Finalmente, este trabajo busca no solo aportar una solución tecnológica al problema de la clasificación de manzanas, sino también aprovechar esta área de oportunidad para aplicar y destacar los conocimientos adquiridos en la carrera de ingeniería electrónica con un enfoque en el ramo de la industria agrícola y alimentaria, demostrando el impacto que la automatización y la inteligencia artificial pueden tener en la mejora de procesos productivos reales.

1.4. Antecedentes

A lo largo de la historia antigua y moderna, la automatización pasó de ser una idea lejana a convertirse en una herramienta presente en numerosos ámbitos de la vida cotidiana. Estos sectores han facilitado y satisfecho distintas necesidades del ser humano, y se observan con claridad en industrias como la agrícola, farmacéutica, aeronáutica, automovilística, tecnológica y militar, entre otras[13].

Por ello, estudiar la historia de la automatización es de suma importancia, ya que permite reconocer los primeros fundamentos e inventos que dieron origen a los sistemas actuales y visualizar el futuro que esta puede ofrecer para mejorar aún más los sectores ya mencionados.

Los antecedentes más antiguos de la automatización registrados se ubican desde la Antigüedad y la Edad Media. En este periodo surgieron sistemas autómatas y artefactos mecánicos inventados por culturas como la griega, romana y egipcia[13].

Un ejemplo representativo es la máquina de Anticitera (100–150 a.C.), capaz de predecir eventos astronómicos mediante engranajes y considerada una de las primeras máquinas analógicas. Posteriormente, se desarrollaron turbinas hidráulicas griegas para moler trigo, marcando inicios relevantes del aprovechamiento de la energía hidráulica[14]. Durante la Edad Media también se presentaron avances, como los molinos de viento, capaces de usar energía renovable para impulsar procesos mecánicos y facilitar trabajos en aserraderos, fábricas de papel, molinos para el afilado de herramientas y otras actividades de la época. A inicios del siglo XVIII, la Revolución Industrial en Gran Bretaña transformó la economía, la sociedad y la tecnología[14]. Un ejemplo de esta transformación fue el reemplazo de la fuerza humana por máquinas en la industria textil. Más adelante, en 1785, Oliver Evans desarrolló un proceso industrial altamente automatizado mediante un molino de harina. En 1801 se introdujo el concepto de programación en una máquina de telar mediante tarjetas perforadas, gracias a Jacquard, lo cual se considera un precursor de la programación moderna. Tiempo después, Joseph Henry desarrolló el relé, un dispositivo que permitió la creación de automatismos eléctricos cableados[15].

En la década de 1940, la automatización se consolidó en la industria con el desarrollo de líneas de producción cada vez más mecanizadas. Posteriormente, en las décadas de 1960 y 1970 aparecieron nuevas tecnologías como los microprocesadores, así como lenguajes de programación que posibilitaron una comunicación más eficiente. En las décadas de 1980 y 1990, con la invención de nuevos sistemas de automatización (termostatos, sistemas de seguridad y sistemas de control), estas tecnologías se introdujeron en el uso doméstico, en casas, oficinas, hospitales y hoteles. En la década de 2000, tecnologías inalámbricas como Bluetooth e internet se volvieron más asequibles y fáciles de usar en el trabajo, la escuela y el ámbito personal. De igual manera, el surgimiento de dispositivos inteligentes permitió mejorar la conexión y la sofisticación de estos sistemas[15]. Hoy en día, la automatización se ha expandido hacia áreas como las pruebas de interfaz de software y los procesos de negocio en bancos y aseguradoras. Además, la inteligencia artificial (IA), aunque es una tecnología emergente, se ha integrado en nuevos procesos de automatización. Un ejemplo claro es el aprendizaje automático y la robótica aplicados a la conducción autónoma, así como el uso de asistentes virtuales como Alexa o el Asistente de Google en pequeñas y medianas empresas que emplean la automatización con IA como servicio[1].

1.5. Planteamiento del problema

Actualmente, el uso de la tecnología ha crecido de manera exponencial en los ámbitos más comunes de la vida humana, como lo podemos observar en la industria agrícola, ya que, durante las últimas décadas la población mundial ha crecido abruptamente, obligando a buscar nuevas alternativas para aumentar, satisfacer, y cumplir la demanda alimenticia que existe hoy en día. Un ejemplo claro de ello es el proceso de clasificación de frutas y verduras para garantizar al consumidor la calidad del producto y cumplir con los lineamientos dados por el mercado. Comúnmente, el método de la clasificación de frutas y verduras es realizado manualmente por personas basándose en la inspección visual del tamaño, color, textura y defectos que pueda tener[5]. Esto puede provocar errores, inconsistencias y pérdidas económicas de ganancias por las diversas limitaciones del personal en cuanto al tiempo que el proceso puede conllevar, porque la clasificación manual no permite mantener un control uniforme en grandes producciones, limitando así la competitividad en el sector[5].

1.6. Objetivos

1.6.1. Objetivo general

Diseñar y construir un sistema clasificador de manzanas a través de un sistema automático implementado con una red neuronal convolucional (CNN), entrenada con imágenes para su identificación en línea, con el propósito de separarlas de acuerdo con su estado visual (apta o no apta para el consumo), optimizando así la eficiencia y confiabilidad del proceso de selección.

1.6.2. Objetivos específicos

- Implementar un mecanismo físico basado en una banda transportadora y un separador de manzanas, diseñado en CAD y fabricado mediante impresión 3D, que permita automatizar el flujo de clasificación.
- Configurar en Python una red neuronal convolucional (CNN) entrenada con imágenes de manzanas, orientada a optimizar la identificación automática según sus características visuales.
- Construir una plataforma experimental funcional que integre el sistema de visión computacional y el mecanismo de separación, capaz de clasificar las manzanas buenas de las malas en tiempo real.
- Instrumentar el proyecto en una Raspberry Pi 400, aprovechando sus características y ventajas para la ejecución de aplicaciones electrónicas y de inteligencia artificial en entornos compactos y de bajo consumo energético.

1.7. Justificación

El presente proyecto se justifica en la necesidad de responder a los retos actuales de la industria agrícola, donde la creciente demanda alimenticia exige procesos más eficientes, confiables y tecnológicamente avanzados. La clasificación manual de frutas, como las manzanas, presenta limitaciones significativas: errores humanos, falta de uniformidad en grandes volúmenes de producción y pérdidas económicas derivadas de inconsistencias en la inspección visual. Ante este panorama, la automatización y el uso de inteligencia artificial se convierten en herramientas estratégicas para garantizar la calidad del producto y fortalecer la competitividad del sector.

La propuesta de diseñar y construir un clasificador automatizado de manzanas a través de un sistema autómatas implementado con una red neuronal entrenada con imágenes responde directamente a esta problemática. Este enfoque permitirá optimizar el tiempo de inspección, reducir errores y asegurar un control uniforme en la selección de productos, contribuyendo a la mejora de la cadena de valor agrícola. Desde el ámbito académico, el proyecto representa una oportunidad para aplicar de manera práctica los conocimientos adquiridos en la carrera de Ingeniería Electrónica, integrando áreas como diseño CAD, impresión 3D, programación en Python y el uso de plataformas de desarrollo como la Raspberry Pi 400. De esta forma, se fomenta el desarrollo de competencias técnicas y profesionales que acercan al egresado a escenarios reales de la industria, fortaleciendo su capacidad de innovación y resolución de problemas.

Asimismo, el proyecto contribuye a la formación integral del estudiante de la Universidad Autónoma del Estado de Hidalgo, al promover la investigación aplicada y la vinculación con tendencias tecnológicas actuales como la inteligencia artificial y la automatización. Estas tecnologías han demostrado un impacto creciente en diversos sectores, y su incorporación en la agricultura abre la posibilidad de generar soluciones sostenibles, escalables y adaptadas a las necesidades del mercado. El proyecto no solo radica en su valor académico y formativo, sino también en su pertinencia social y económica, al ofrecer una alternativa tecnológica que puede mejorar la eficiencia productiva, garantizar la calidad alimentaria y contribuir al desarrollo de la industria agrícola en México y en el mundo.

1.8. Hipótesis

La implementación de un prototipo de sistema electrónico automatizado, basado en redes neuronales, para la clasificación de manzanas según su estado de conservación (buenas o defectuosas) permitirá alcanzar una precisión superior al 90 % en la identificación de características como color, textura y presencia de daños, reduciendo significativamente los errores humanos y el tiempo de inspección en comparación con el método manual.

Este sistema optimizará la eficiencia del proceso de selección, disminuirá las pérdidas económicas por clasificaciones incorrectas y garantizará un producto homogéneo y confiable para el consumidor.

1.9. Estado del arte

A continuación se presentan trabajos similares al expuesto en esta tesis, donde se describen de manera breve los procedimientos empleados y los resultados obtenidos en dichos trabajos.

CLASIFICACIÓN DE MANZANAS CON REDES NEURONALES CONVOLUCIONALES

En este artículo científico publicado y elaborado en colaboración por diferentes personas egresadas de distintas universidades de MÉXICO, describen el diseño y elaboración de un clasificador de manzanas como se logra apreciar en la figura 1.4b, usando tres arquitecturas diferentes como la CNN, LeNet5 y la VGG16 cuyos modelos fueron entrenados por medio de diferentes conjuntos de imágenes como se demuestra en la figura 1.4a. Cada arquitectura cumplió una función específica para la clasificación de las manzanas, por ejemplo la arquitectura CNN fue la encargada de ejecutar el sistema de visión, logrando la clasificación de 4 variedades de manzanas en categorías como sanas y con daños. Mientras la arquitectura LeNet5 es la encargada del conjunto de datos MNIST de reconocimiento de caracteres escritos, logrando una clasificación con exactitud del 99.2 por ciento. Ésta arquitectura constó de 2 capas CONVOLUCIONALES, 2 capas de cribado máximo, y 3 capas densas, más algunas modificaciones en ciertas capas. Y por último la arquitectura VGG16 está basada en una serie de bloques CONVOLUCIONALES que consta de una capa de agrupación y capas densas en cual se le asignaron cierta cantidad de neuronas para una debida activación[16].



(a) Imágenes de entrenamiento[16].



(b) Diseño del sistema[16].

Figura 1.4. Manzanas de entrenamiento y diseño del clasificador.

De igual manera se implementó en el diseño del proyecto una red con hiperpárametros para seleccionar la mejor red o arquitectura durante rendimiento de la clasificación de las manzanas, los factores considerados en cada tratamiento fue la arquitectura, el número de capas a entrenar, la inicialización de los pesos, el optimizador y la tasa de aprendizaje que este poseía. En la tabla 1.1 se muestran las exactitudes obtenidas en la pruebas y estas se logran encontrar en un rango de 0.61 a 0.97, y que en los tratamientos (1 al 10) de la arquitectura LeNet5 con optimizador RMSProp y tasa de aprendizaje 1×10^{-4} es la de mejor desempeño, con una exactitud de 0.97[16].

Hiperpárametros de entrenamiento					Rendimiento en clasificación		
Trat.	Optimizador	Tasa de aprendizaje	Pesos iniciales	Épocas	Exactitud entrenamiento	Exactitud validación	Exactitud prueba
1	RMSProp	1.0×10^{-3}	Random	21	0.90	0.66	0.61
2	RMSProp	1.0×10^{-4}	Random	44	0.99	0.98	0.97
3	RMSProp	1.0×10^{-5}	Random	174	0.98	0.94	0.94
4	RMSProp	1.5×10^{-5}	Random	131	0.96	0.94	0.92
5	RMSProp	1.0×10^{-6}	Random	226	0.94	0.88	0.87
6	Adam	1.0×10^{-3}	Random	15	0.70	0.58	0.62
7	Adam	1.0×10^{-4}	Random	36	0.99	0.96	0.95
8	Adam	1.0×10^{-5}	Random	65	0.96	0.90	0.89
9	Adam	1.5×10^{-5}	Random	64	0.97	0.94	0.93
10	Adam	1.0×10^{-6}	Random	247	0.95	0.91	0.92

Cuadro 1.1. Tabla de resultados 1-10[16].

En la tabla 1.2 se puede observar que la exactitud de prueba para los tratamientos 11 al 20 se encuentra en un rango de intervalo de 0.12 a 0.95, y la arquitectura VGG16 con optimizador Adam y tasa de aprendizaje 1×10^{-5} , con una exactitud de 0.95 tuvo el mejor desempeño[16].

Hiperparámetros de entrenamiento					Rendimiento en clasificación		
Trat.	Optimizador	Tasa de aprendizaje	Pesos iniciales	Épocas	Exactitud entrenamiento	Exactitud validación	Exactitud prueba
11	RMSProp	1.0×10^{-3}	Random	12	0.10	0.12	0.12
12	RMSProp	1.0×10^{-4}	Random	41	0.98	0.94	0.83
13	RMSProp	1.0×10^{-5}	Random	103	0.98	0.95	0.94
14	RMSProp	1.5×10^{-5}	Random	74	0.97	0.96	0.95
15	RMSProp	1.0×10^{-6}	Random	111	0.65	0.57	0.58
16	Adam	1.0×10^{-3}	Random	17	0.11	0.12	0.12
17	Adam	1.0×10^{-4}	Random	46	0.99	0.87	0.84
18	Adam	1.0×10^{-5}	Random	71	0.99	0.97	0.95
19	Adam	1.5×10^{-5}	Random	91	0.99	0.94	0.88
20	Adam	1.0×10^{-6}	Random	99	0.77	0.73	0.66

Cuadro 1.2. Tabla de resultados 11-20[16].

La tabla 1.3 muestra los tratamientos del 21 al 30, con una exactitud en el conjunto de prueba con un intervalo de 0.91 a 0.95, y la arquitectura VGG16 con optimizador Adam su tasa de aprendizaje fue de 1×10^{-5} con extracción de características y una exactitud del 0.95[16].

Hiperparámetros de entrenamiento					Rendimiento en clasificación		
Trat.	Optimizador	Tasa de aprendizaje	Pesos iniciales	Épocas	Exactitud entrenamiento	Exactitud validación	Exactitud prueba
21	RMSProp	1.0×10^{-3}	ImageNet	36	0.97	0.94	0.93
22	RMSProp	1.0×10^{-4}	ImageNet	28	0.98	0.93	0.91
23	RMSProp	1.0×10^{-5}	ImageNet	90	0.99	0.95	0.94
24	RMSProp	1.5×10^{-5}	ImageNet	51	1.00	0.95	0.94
25	RMSProp	1.0×10^{-6}	ImageNet	190	1.00	0.95	0.95
26	Adam	1.0×10^{-3}	ImageNet	22	1.00	0.95	0.92
27	Adam	1.0×10^{-4}	ImageNet	55	1.00	0.95	0.95
28	Adam	1.0×10^{-5}	ImageNet	83	1.00	0.96	0.95
29	Adam	1.5×10^{-5}	ImageNet	66	1.00	0.95	0.95
30	Adam	1.0×10^{-6}	ImageNet	267	1.00	0.95	0.95

Cuadro 1.3. Tabla de resultados 21-30[16].

En la tabla 1.4 se muestran los resultados de la 31 al 40 y muestra una exactitud en el conjunto de prueba con un intervalo entre 0.94 y 0.96, y la arquitectura VGG16 con optimizador Adam, tasa de aprendizaje de 1.5×10^{-5} y ajuste fino tuvo exactitud de 0.96.

Se determinó que LeNet5 con el tratamiento 2, fue la arquitectura de mejor desempeño, pues obtuvo una exactitud de 97 por ciento; por lo tanto, es la mejor arquitectura[16].

Hiperparámetros de entrenamiento					Rendimiento en clasificación		
Trat.	Optimizador	Tasa de aprendizaje	Pesos iniciales	Épocas	Exactitud entrenamiento	Exactitud validación	Exactitud prueba
31	RMSProp	1.0×10^{-3}	ImageNet	23	0.96	0.78	0.94
32	RMSProp	1.0×10^{-4}	ImageNet	25	0.98	0.94	0.96
33	RMSProp	1.0×10^{-5}	ImageNet	60	1.00	0.96	0.95
34	RMSProp	1.5×10^{-5}	ImageNet	51	1.00	0.97	0.95
35	RMSProp	1.0×10^{-6}	ImageNet	120	1.00	0.96	0.95
36	Adam	1.0×10^{-3}	ImageNet	27	1.00	0.97	0.96
37	Adam	1.0×10^{-4}	ImageNet	36	1.00	0.98	0.95
38	Adam	1.0×10^{-5}	ImageNet	65	1.00	0.97	0.96
39	Adam	1.5×10^{-5}	ImageNet	70	1.00	0.97	0.96
40	Adam	1.0×10^{-6}	ImageNet	178	1.00	0.96	0.95

Cuadro 1.4. Tabla de resultados 31-40[16].

Como conclusión, los autores determinaron que la red neuronal LeNet5 entrenada desde cero, con optimizador RMSProp y tasa de aprendizaje 1×10^{-4} logra clasificar correctamente las manzanas sanas y dañadas de los 4 tipos de variedades que ellos probaron (Red Delicious, Granny Smith, Golden Delicious y Gala), con una exactitud del 97 por ciento; para las cinco categorías restantes, el porcentaje varía entre 92 y 97 por ciento.

La arquitectura seleccionada cumple con los requisitos de la norma oficial mexicana NMXFF-061-SCFI-2003. Un hallazgo en este trabajo fue que el diseño experimental generado permite encontrar específicamente los hiperparámetros y la arquitectura con el mejor desempeño en términos de clasificación [16].

Clasificación de manzanas mediante visión por computador e inteligencia artificial para detectar anomalías

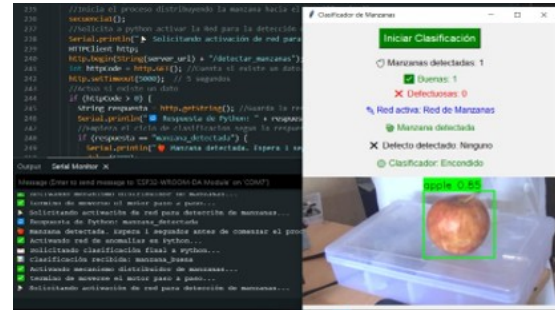
En ésta tesis de los ingenieros en mecatrónica JOSÉ LUIS TINOCO MERA y JUAN PABLO TINOCO MERA de la UNIVERSIDAD POLITÉCNICA SALESIANA DE GUAYAQUIL desarrollaron un sistema automatizado para la clasificación de manzanas mediante visión por computador e inteligencia artificial como se muestran en la figura 1.5, con el objetivo de mejorar la eficiencia y precisión en procesos agrícolas de selección de fruta de dicho país.

Los ingenieros a cargo del proyecto usaron 2 tipos de redes neuronales como la YOLOv11: uno para la detección general de manzanas y otro para la identificación de anomalías como heridas abiertas y podredumbre. La arquitectura del sistema contempla la integración de elementos electrónicos como un microcontrolador ESP32, en el diseño mecánico es integrada por una banda transportadora delimitadora impulsada por un motor paso a paso, así mismo otros elementos que requirieron durante la elaboración[17].

Uno de los objetivos primordiales de este sistema es que pueda contribuir a la automatización de tareas agrícolas utilizando tecnologías accesibles y de código abierto, fomentando su adopción en pequeñas y medianas empresas del sector.



(a) Clasificación de manzanas[17].



(b) Detección de una manzana[17].

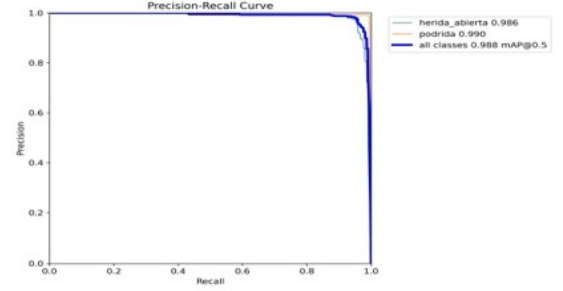
Figura 1.5. Clasificación y detección de manzanas.

Como resultado, obtuvieron una precisión de 0.994 para la clasificación de manzanas, lo cual indica que el modelo es eficaz identificando manzanas de todo tipo[17]. La precisión se evaluó mediante una gráfica cuantitativa por medio de la curva de Precisión-Recall mostrada en la figura 1.6b, y también se presentan imágenes de validación con las predicciones realizadas por la red neuronal durante el entrenamiento.

El modelo identifica y delimita correctamente las manzanas presentes en las escenas en tiempo real, evidenciando visualmente su capacidad para detectar la clase objetivo con precisión y consistencia como se observa en la figura 1.6a[17].



(a) Evaluación de las manzanas[17].



(b) Curva de precisión RECALL[17].

Figura 1.6. Proceso de evaluación y precisión de las manzanas.

La conclusión de los autores fue que el desarrollo e implementación del sistema automatizado para la clasificación de manzanas mediante la visión por computadora de los modelos entrenados presentaron un rendimiento adecuado. El modelo YOLOv11 utilizado para la detección general de manzanas logró un objetivo de precisión del 99 por ciento para manzanas de todo tipo. Por su parte, el modelo destinado a la detección de anomalías obtuvo valores superiores al 98 por ciento de precisión en ambas clases herida abierta y podrida, evidenciando un desempeño confiable y adaptado a escenarios reales[17].

Otro objetivo adicional fue la integración de los distintos componentes tanto a nivel de software como de hardware. La captura de video, el procesamiento de las imágenes en tiempo real y el manejo del mecanismo de clasificación, los cuales trabajaron conjuntamente utilizando un microcontrolador ESP32, servomotores, tornillo, y una banda transportadora presentó resultados superiores a los esperados.

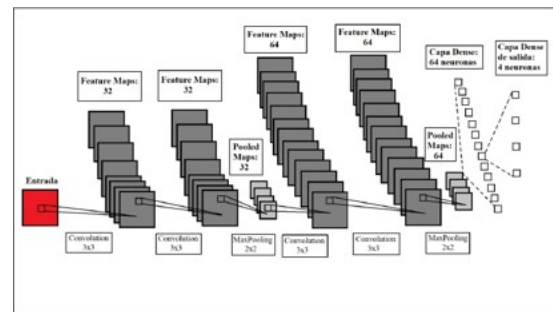
Clasificación de manzanas utilizando visión artificial y redes neuronales artificiales

En este artículo de la revista ingeniería y región de Colombia trata de igual manera sobre un clasificador de manzanas usando una red neuronal convolucional (CNN) de la figura 1.7a, la cual se desarrolló utilizando librerías de código abierto en Python con una estructura de diferentes capas CONVOLUCIONALES y MaxPooling, para un conjunto de análisis de alrededor de 2,800 imágenes de $128 * 128$ píxeles, de las cuales el 80 por ciento se utilizaron para entrenamiento y el 20 por ciento para hacer una prueba de la red, obteniendo una precisión del 98.3 por ciento y del 95.36 por ciento, respectivamente. Después del entrenamiento por imágenes que muestra la tabla de la figura 1.7b, realizaron una clasificación con un video en tiempo real, obteniendo una precisión del 92.25 por ciento. Asimismo, exploraron la posibilidad dentro de la industria con la clasificación por otras características visuales del fruto como tamaño, color, forma, etc[18].

Tabla 1. Variedad y numero de imagen de cada manzana

Variedad de manzana	Imágenes para entrenamiento	Imágenes para Prueba
Gala	560	140
Gold	560	140
Granny Smith	560	140
Red Delicious	560	140

(a) Tipos de manzanas[18].



(b) Arquitectura de una red CNN[18].

Figura 1.7. Tipos de manzanas y estructura de la red neuronal CNN.

Como resultados lograron obtener que el entrenamiento mostró una precisión del 98.3 por ciento, sin embargo, cuando midieron el rendimiento del algoritmo con las imágenes de prueba seleccionadas aleatoriamente, se clasificaron erróneamente 26 de las 560 imágenes, por lo que muestra una precisión del 95.36 por ciento debido a la matriz de confusión dada como se muestra en la figura 1.8. Por lo cual la prueba de rendimiento no asegura que el algoritmo que realizaron desempeñe la misma precisión al ser sometido a una clasificación en tiempo real y con datos diferentes a los utilizados para el entrenamiento y evaluación de la red neuronal[18].

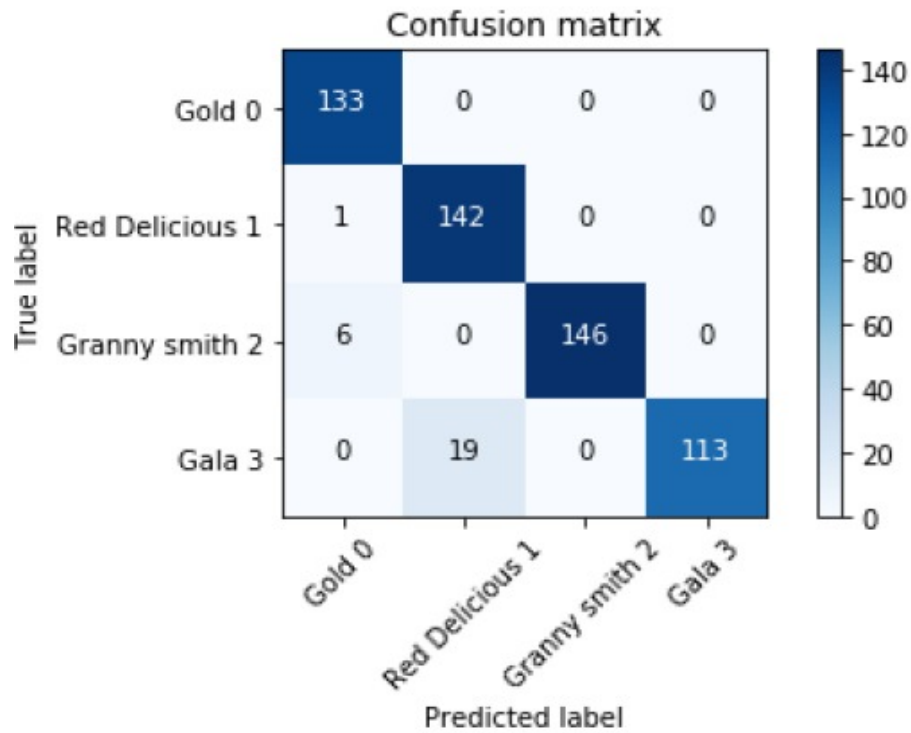


Figura 1.8. Matriz de la confusión[18].

Como conclusión, los autores se basaron en los resultados obtenidos en la clasificación de manzanas, de los cuales estos dieron muy buenos rendimientos, por otra parte los autores mencionan que es posible mejorar el proyecto ampliando la base de datos para el entrenamiento en cuanto a la calidad de imagen y de iluminación, ya que esos aspectos generalmente afectan la precisión de la red. También mencionan que la prueba del modelo lo realizaron en un ambiente con mucho ruido, en cuanto al fondo de la imagen e iluminación, lo cual provocó que la aplicación se realizara en un ambiente controlado pudiendo mejorar su precisión[18].

Clasificador automático de frutas y verduras

En este proyecto de la UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO se muestra y desarrolla un sistema dedicado a la clasificación automática de frutas y verduras por medio de la adquisición de imágenes que son procesadas y con la implementación de un sistema embebido. El objetivo de dicho proyecto fue el desarrollo de un sistema electromagnético para la clasificación mediante la adquisición de imágenes y su procesamiento con un sistema embebido, cuyo diagrama se muestra en la figura 1.9.

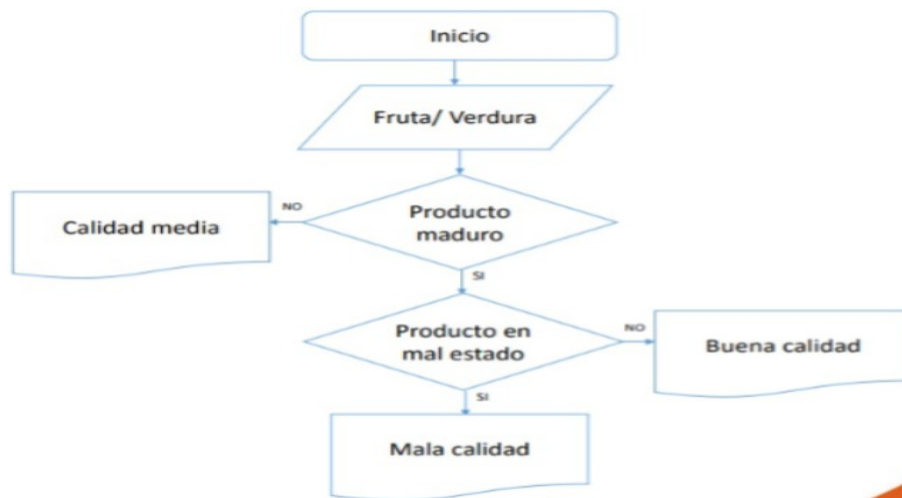
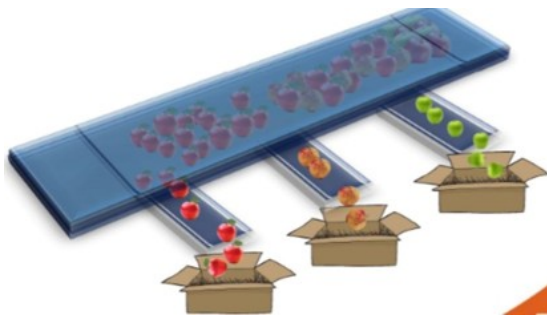
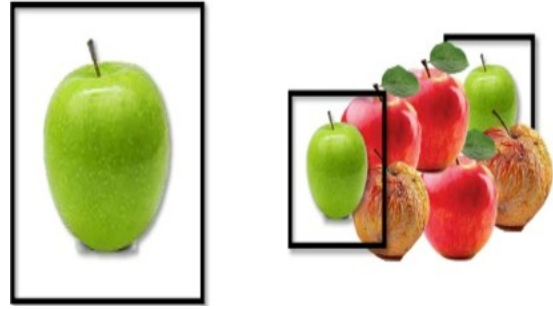


Figura 1.9. Diagrama de flujo del sistema[19].

En el diseño de la estructura física de la máquina se buscó que se clasificaran las frutas y verduras en grandes masas como ejemplifica la figura 1.10b, y que se desplazara el producto por medio de una banda principal como se muestra en la figura 1.10a. Durante el recorrido se implementaron 3 áreas de clasificación, en el momento en que se detecte un producto que no esté maduro, se activará un brazo para que desvíe el producto a una primera clasificación, la segunda clasificación deriva de una mala calidad, y la tercera clasificación es una buena calidad del producto. Cada banda de clasificación contó con un contador, el cual permitió un mejor control de la cantidad de producto en las cajas [19].



(a) Clasificación de manzanas[19].



(b) Detección de manzanas[19].

Figura 1.10. Clasificación y detección de manzanas.

El contador fue un sensor infrarrojo que trabajó conjuntamente con un dispositivo de visualización para facilitar la interacción del usuario con el sistema como se ilustra en la figura 1.11 [19].

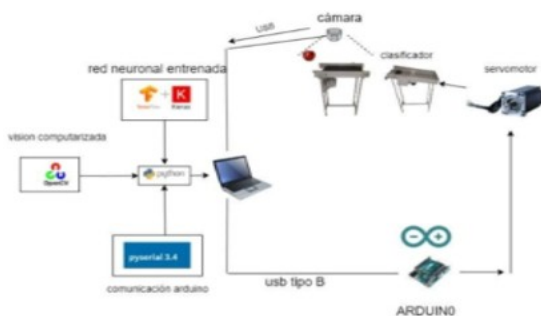


Figura 1.11. Contador principal del sistema[19].

Como conclusión de los autores fue que el clasificador automático identificó la calidad del producto de las imágenes capturadas que fueron comparadas con imágenes estándares para su clasificación y detección de la ubicación del producto [19].

DISEÑO DE UN PROTOTIPO DE SISTEMA DE CLASIFICACIÓN DE MANZANAS SEGÚN SU DEFECTO MEDIANTE TÉCNICAS DE VISIÓN ARTIFICIAL PARA LA INSPECCIÓN DE CALIDAD EN PROCESOS INDUSTRIALES

En este trabajo de tesis de la UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA elaborada por el autor PAÚL SUÁREZ RICARDO, propuso y diseño un prototipo de clasificadora a través de sistemas artificiales para la selección de manzanas. Su sistema de implementación de la figura 1.12a está compuesta por una etapa de transportación donde utilizó un motor reductor AC que está conectado por unas poleas que moviliziza la banda transportadora hacia la zona de visión artificial y ésta captura video en tiempo real para posteriormente pase a la etapa de clasificación donde cuenta con un servomotor conectado a una paleta en forma horizontal, donde dependiendo del estado visual de la manzana, ésta será dirigida del lado derecho o izquierdo de las manzanas ya sea en buen o mal estado como se aprecia en la figura 1.12b[20].



(a) Arquitectura del sistema[20].



(b) Detección de manzanas[20].

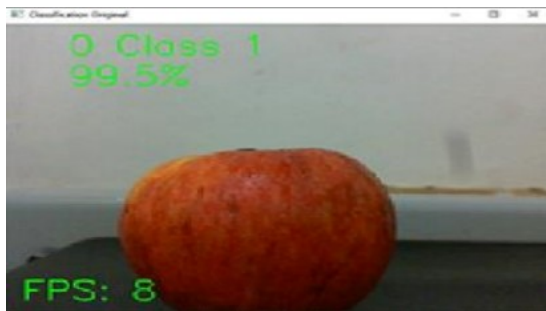
Figura 1.12. Arquitectura del sistema y detección de la manzanas.

Los resultados del autor fueron que el sistema diseñado en Python haciendo uso de las bibliotecas de Tensorflow, Teachable Machine y Open CV para realizar la captura de video en tiempo real, ésta predijo el tipo de clase de manzana a través del modelo entrenado con lo proyectado en la cámara. La manzana buena de la figura 1.13a obtuvo un 99.5 por ciento de efectividad, el caso de manzana mala de la figura 1.13b se obtuvo un 99,97 por ciento, y para el caso de estado vacío se obtuvo un 99,98 por ciento al momento de ser clasificada, resultando óptimo en el sistema de visión artificial de la red neuronal[20].

La estructura del proyecto del entrenamiento fue establecida para 50 épocas, un tamaño de lote aproximado de 16 y una tasa de aprendizaje de 0,001 en donde ésta arrojó diversos resultados:

- Precisión por época al final del entrenamiento de 0,999.
- Pérdida por época al final del entrenamiento de 0.00083630.
- Aproximadamente cero, considerada prácticamente nula

Lo cual indicó que las predicciones para cada clase superan el 98 por ciento de efectividad durante el proceso.



(a) Manzana buena[20].



(b) Manzana mala[20].

Figura 1.13. Diferencia entre manzana buena y mala

La conclusión del autor fue que el uso de una red neuronal artificial utilizando las librerías de TensorFlow y KERAS en la aplicación web Teachable Machine logró clasificar manzanas según su defecto sean estas buenas o malas de tal manera que el modelo se puede descargar y usarlo en un sistema genérico para pequeñas y medianas empresas.

También hace énfasis en la interfaz de usuario de nivel básico mediante la herramienta de software Qtdesigner cuyas librerías son compatibles con el sistema de OPEN CV desarrollado en Python para el monitoreo del sistema[20].

1.10. Benchmarking

En los años recientes se ha adquirido una gran relevancia en la automatización mediante el control basado en inteligencia artificial en la ingeniería electrónica, debido a que en la vida cotidiana se interactúa con sistemas de cualquier índole, desde los más fáciles hasta los más complejos, dado a que existen sistemas que funcionan de manera autónoma, sin la necesidad de que exista una intervención humana. También es cierto que sin la acción que realiza el ser humano para que los sistemas se comporten como uno desee, dichos sistemas resultarían inútiles; a todas aquellas acciones y decisiones que se toman con base en referencias, patrones y comportamientos se le llama automatización[21].

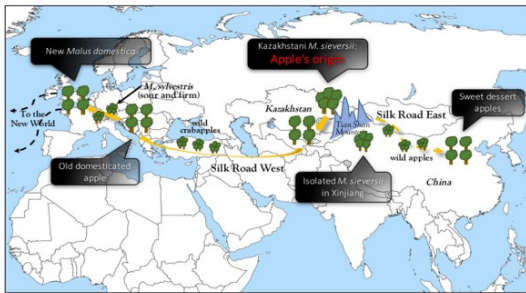
La automatización es la eliminación de la necesidad de intervención humana en tareas repetitivas o complejas que tienen como función la ejecución de procesos, control de maquinaria, y gestión de datos de forma automática[22].

Como objetivos primordiales de la automatización, se pueden destacar los siguientes puntos:

- Incrementar la productividad y eficiencia.
- Reducir costos.
- Eliminar errores humanos.
- Optimizar tiempos de proceso.
- Reemplazar tareas repetitivas y tediosas.

1.10.1. Primeras apariciones

Como se observa en las figuras 1.14a, la historia de la manzana tiene su origen en Asia central y zonas cercanas al Mar Caspio y al Mar Negro cerca de unos 10,000 años atrás; donde los humanos de aquella época iniciaron el proceso de domesticación y clasificación de manzanas cerca de las montañas de Tien Shan, se piensa o se cree que las manzanas que conocemos hoy en día en Norteamérica descienden de la clasificación de la especie *MALUS SIEVERSII*, que crecía en las laderas de aquella región. Siglos después, los primeros colonos ingleses y españoles de América introdujeron las manzanas en los años 1620, plantando, cultivando, y clasificando las manzanas como en la figura 1.14b en los primeros huertos de la bahía de Massachusetts (Boston), Virginia, Nuevo México, Pensilvania, Ohio, Indiana, Illinois, Virginia Occidental y Ontario Canadá [23].



(a) Distribución de las manzanas[24].



(b) Agricultura en Norte América[25].

Figura 1.14. Origen e historia de las manzanas.

La producción de manzanas siguió creciendo hasta tal punto de ser una de las principales frutas más consumidas por el ser humano, por ende aumento la demanda en un momento donde no se contaba con la infraestructura necesaria para cubrir dicha demanda, no es hasta que en la mitad del siglo XX, la tecnología llega a la industria agrícola con vehículos autónomos como los tractores en las décadas de los 50's y 60's[26].

La invención de tecnologías modernas ayudaron en gran parte a satisfacer la demanda de producción de manzanas, ejemplo de esta tecnología es la banda transportadora automatizada, que al principio solamente era usada en la minería para el traslado de carbón paso a ser una pieza fundamental en industrias, sobre todo en la industria agrícola [26].

La banda transportadora industrial ha facilitado la tarea de trasladar la manzana de un punto a otro como se puede apreciar en la figura 1.15, mientras que estas son seleccionadas manualmente por personal mediante la calidad que estas poseen[27].



Figura 1.15. Clasificación manual de manzanas[28].

Hoy en día con la incorporación de nuevas energías y tecnologías digitales como la GPS, IA, sensores de precisión para implementar sistemas de agricultura, invernaderos automatizados y demás elementos, permiten la realización de tareas más eficientes en el riego, cosecha y clasificación de verduras y frutas en la que destaca la manzana como en la figura 1.16[29].



Figura 1.16. Ejemplo de manzana de calidad[30].

1.10.2. Descripción

Las bandas transportadoras tradicionalmente estaban fabricadas por un cinturón de cuero o lona que viajaba sobre una cama plana de madera, y eran operadas manualmente para mover grano y carbón como se visualiza en la figura 1.17a. No es hasta que 1870, la banda transportadora se instaló en un matadero, iniciando la producción en cadena mecanizada. Tiempo después la empresa sueca SANDVIK en 1901 creó las bandas transportadoras de acero, facilitando la transportación de materiales pesados como grava y carbón a mayores distancias. Con el avance de nuevas tecnologías que surgieron del siglo XX, se fueron mejorando estas bandas transportadoras con nuevos materiales duraderos como el sintético y haciéndolas más seguras para los operadores por las leyes de estándares de seguridad de la asociación estadounidense[26]. Esto provocó que se expandiera en industrias de proceso, una de esta industria fue la de automóviles de HENRY FORD como en la figura 1.17b, el mismo implementó bandas transportadoras en el sistema de la línea de montaje de su empresa, reduciendo drásticamente el tiempo de ensamblaje de automóviles y popularizándolo a nivel mundial [15].



(a) 1era banda transportadora[26].



(b) Línea de ensamblaje FORD[31].

Figura 1.17. Primera banda transportadora y línea de ensamblaje de FORD.

Ya entre las décadas de los años 70s y 80s los diseños de bandas transportadoras eran menos ruidosos, además de desarrollar rodillos transportadores de accionamiento interno y poleas motorizadas para eliminar un costoso mantenimiento[26]. En los años recientes, las bandas transportadoras empezaron a implementar elementos más modernos como sensores de proximidad y temperatura, pantallas para mostrar mensajes del proceso, brazos robóticos para la clasificación y cámaras de monitoreo las 24 horas del día[27].

Las bandas transportadoras son una parte fundamental en los procesos industriales de todo el mundo, debido a que estas facilitan y optimizan los procesos repetitivos que se viven día con día en industrias como se muestra en la figura 1.18, sin embargo, el desarrollo de nueva tecnología como la IA puede mejorar procesos, como la selección de objetos de consumo de calidad [26]. Un ejemplo claro es la clasificación de manzanas, debido a que estas pasan por un proceso de selección manual por medio de bandas transportadoras, lo cual puede derivar en una mala selección de calidad o causar pérdidas de tiempo por parte del personal, retrasando la distribución del producto en tiendas de abarrotes y servicios [27].



(a) Bandas industriales[32].



(b) Traslado de manzanas[33].

Figura 1.18. Bandas industriales en procesos de producción.

Capítulo 2

Marco teórico

2.1. Marco Teórico

Partiendo de la problemática inicial de la clasificación manual de manzanas y sus diversas limitaciones por el factor humano, desde el punto de vista del investigador en la búsqueda de una solución viable y automatizada, es indispensable la comprensión y definición de fundamentos conceptuales y técnicos que sustentan el desarrollo del proyecto en cuestión, abordando así, los principios de automatización, visión artificial, inteligencia artificial y las redes neuronales, así como su aplicación en sistemas embebidos orientados a procesos agrícolas. Estos conceptos permitirán el entendimiento del sistema propuesto y justifican la selección de tecnologías empleadas.

2.1.1. Automatización

La automatización se define como el uso de sistemas tecnológicos capaces de ejecutar tareas o procesos con mínima o nula intervención humana. Su principal objetivo es mejorar la eficiencia, precisión y repetibilidad de los procesos productivos, reduciendo errores y optimizando recursos. En el ámbito industrial, la automatización ha permitido la sustitución de tareas manuales repetitivas por sistemas mecánicos y electrónicos capaces de operar de manera continua y controlada[29].

Existen diferentes tipos de automatización, cada una con sus propias características y ventajas, de las cuales las más destacables son las siguientes:

2.1.2. Automatización Fija (Rígida)

Este tipo de automatización se caracteriza por la producción masiva y específica, diseñada exclusivamente para un producto en específico y de gran volumen para la producción en masa continua, ejemplo de ello son las embotelladoras en la figura 2.1 [29].

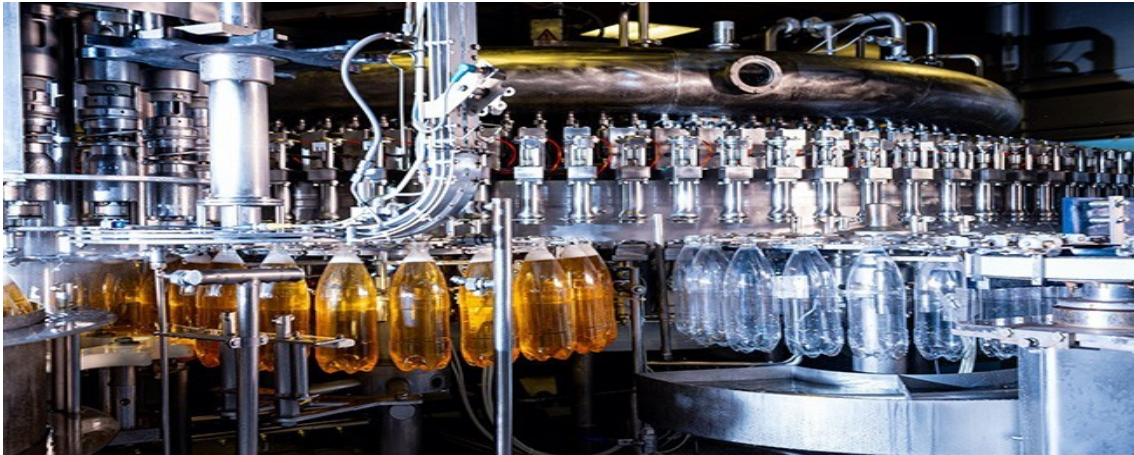


Figura 2.1. Automatización fija en fábrica embotelladora[34].

2.1.3. Automatización Programable

Esta automatización es la que permite el cambio de configuración para producir diferentes productos en lotes como en la figura 2.2. Es ideal en fábricas que permite re-programar los productos como los electrodomésticos[29].



Figura 2.2. Automatización programable en fábrica de electrodomésticos[35].

2.1.4. Automatización Flexible

La automatización flexible combina lo mejor de la fija y programable, permitiendo ajustes automáticos y rápidos entre productos sin detener la línea. Es ideal para la producción de pequeños lotes o mezclas de productos con cambios frecuentes, por ejemplo, las plantas de alimento como en la figura 2.3[29].



Figura 2.3. Automatización flexible en fábrica de alimentos de galletas[36].

Existen otros tipos de automatización como:

- **Automatización integrada:** Une sistemas automatizados para que funcionen como un solo equipo, incorporando control de calidad y gestión operativa.
- **Automatización Robótica de Procesos (RPA):** Empleo de bots de software para automatizar tareas repetitivas en oficinas, como la entrada de datos, replicando acciones humanas.
- **Automatización de Procesos de Negocio (BPA):** Automatiza procesos de trabajo integrales dentro de una organización.
- **Automatización del Hogar (Domótica):** Control de dispositivos domésticos como luces, alarmas y riego.

En la industria agrícola, la automatización ha cobrado especial relevancia debido a la necesidad de incrementar la productividad y garantizar la calidad de los productos. Procesos como el riego, la cosecha, el transporte y la clasificación de frutas han sido beneficiados por la incorporación de bandas transportadoras, sensores y sistemas de control automático. La automatización de la clasificación de productos agrícolas permite estandarizar criterios de selección y reducir la dependencia de la evaluación humana[4].

2.1.5. Visión artificial

La visión artificial es una rama de la ingeniería que permite a las máquinas interpretar información visual proveniente del entorno mediante el uso de cámaras y algoritmos de procesamiento digital de imágenes. Su objetivo es emular, en cierta medida, la capacidad del sistema visual humano para identificar objetos, formas y características relevantes dentro de una imagen. Un sistema de visión artificial generalmente se compone de una etapa de adquisición de imágenes, seguida de un proceso de reconocimiento, análisis y toma de decisiones. La adquisición se realiza mediante cámaras digitales que capturan imágenes o video en tiempo real. Posteriormente, las imágenes pueden ser sometidas a procesos como conversión a escala de grises, filtrado, normalización y segmentación, con el fin de mejorar la calidad de la información visual y facilitar su análisis[5].

En aplicaciones industriales y agrícolas, la visión artificial es utilizada para la inspección de calidad, detección de defectos, conteo de productos y clasificación, permitiendo una evaluación objetiva y consistente de los productos analizados[5].

2.1.6. Imágenes digitales

Las imágenes digitales son representaciones bidimensionales de escenas visuales, compuestas por píxeles, donde cada píxel almacena información de intensidad o color. Pueden ser estáticas o dinámicas, con tipos como raster (mapas de bits, basados en píxeles) o vectoriales (escalables sin pérdida de calidad). Su calidad depende de factores como la resolución (píxeles por pulgada), profundidad de color y tamaño, permitiendo manipulación en computadoras mediante software como representa la figura 2.4[37].



Figura 2.4. Representación de imagen digital [38].

2.1.7. Resolución y píxeles

La resolución se refiere al número total de píxeles que componen una imagen o pantalla, calculado multiplicando el ancho por el alto (por ejemplo, 1920 x 1080). Un píxel es la unidad básica de color, y resoluciones más altas ofrecen mayor nitidez al aumentar la densidad de píxeles por pulgada (PPI o PPP) como se observa en la tabla 2.1 [39].

Nombre	Resolución
True 8K	8192 x 4320
8K UHD	7680 x 4320
True 4K	4096 x 2160
4K UHD	3840 x 2160
2K	2048 x 1080
Full HD	1920 x 1080
HD	1280 x 720

Cuadro 2.1. Tipo de resoluciones[40].

2.1.8. Procesamiento digital de imágenes

El procesamiento digital es la aplicación de técnicas para mejorar, analizar y transformar imágenes digitales mediante algoritmos computacionales. Una imagen digital está formada por una matriz de píxeles, donde cada píxel representa un valor de intensidad o color. Entre las operaciones más comunes en el procesamiento de imágenes se encuentran el mejoramiento de imagen a través de la mejora de brillo y contraste, la adquisición de datos, y la segmentación de regiones de interés como se demuestra en la figura 2.5 [41]. Estas técnicas permiten resaltar características importantes de la imagen que pueden ser utilizadas posteriormente por sistemas de inteligencia artificial para realizar tareas de clasificación o reconocimiento.



Figura 2.5. Procesamiento de una imagen de un camaleón[42].

En el presente proyecto, el procesamiento digital de imágenes es fundamental para preparar las imágenes capturadas por la cámara antes de ser analizadas por la red neuronal, garantizando una mejor precisión en la clasificación de las manzanas.

2.1.9. Inteligencia artificial

La inteligencia artificial (IA) es la capacidad de las computadoras y de las máquinas que usan algoritmos y datos para el desarrollo de sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, tales como el aprendizaje, la toma de decisiones y el reconocimiento de patrones como se pretende representar en la figura 2.6. Dentro de la inteligencia artificial se encuentra el aprendizaje automático o machine learning, el cual consiste en el entrenamiento

de modelos matemáticos utilizando datos de entrada para que puedan realizar predicciones o clasificaciones sin necesidad de ser programados explícitamente para cada caso[43].

Este tipo de aprendizaje es ampliamente utilizado en aplicaciones de visión artificial y análisis de datos.

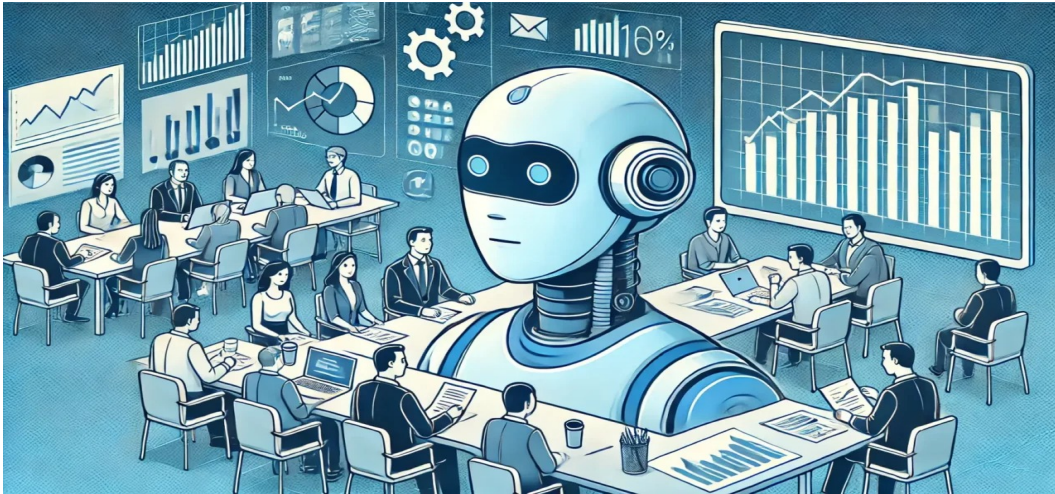


Figura 2.6. Inteligencia artificial en aprendizaje y toma de decisiones.[44].

2.1.10. Redes neuronales artificiales

Las redes neuronales artificiales son modelos o esquemas de computación inspirados en el funcionamiento del sistema nervioso humano. Están formadas por un conjunto de neuronas artificiales interconectadas por múltiples elementos procesadores elementales, organizadas en capas, que procesan información mediante operaciones matemáticas[8].

Una red neuronal típica consta de una capa de entrada, una o varias capas ocultas y una capa de salida. Cada neurona recibe señales de entrada, las pondera mediante pesos sinápticos y las procesa a través de una función de activación, generando una salida que es transmitida a la siguiente capa. Durante el entrenamiento, los pesos de la red se ajustan para minimizar el error entre la salida obtenida y la salida esperada. Las redes neuronales son ampliamente utilizadas en problemas de clasificación, reconocimiento de patrones y predicción, debido a su capacidad para aprender relaciones complejas a partir de datos[8].

2.1.11. Redes neuronales convolucionales (CNN)

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son un tipo especializado de red neuronal artificial que permiten el procesamiento de imágenes. A diferencia de las redes neuronales tradicionales, las CNN utilizan operaciones de convolución que permiten extraer automáticamente características relevantes como bordes, texturas y formas. Una CNN está compuesta principalmente por capas convolucionales, capas de agrupamiento (pooling) y capas densas. Las capas convolucionales tienen una estructura jerárquica en la que aplican filtros sobre la imagen de entrada para detectar patrones locales, mientras que las capas de agrupamiento reducen la dimensionalidad de la información, disminuyendo el costo computacional y evitando el sobre-ajuste. Finalmente, las capas densas se encargan de realizar la clasificación final de las imágenes en las categorías definidas[9].

Gracias a su alta capacidad de aprendizaje y precisión, las CNN son ampliamente utilizadas en aplicaciones de visión artificial, incluyendo la clasificación de frutas y productos agrícolas, donde es necesario analizar grandes cantidades de imágenes con variaciones de iluminación, forma y color[9].

2.1.12. Sistemas embebidos

Un sistema embebido es un dispositivo computacional que integra un sistema electrónico en el que realiza una función específica dentro de un sistema mayor o discreto. A diferencia de una computadora de propósito general, los sistemas embebidos están optimizados para tareas concretas y suelen integrarse con sensores, actuadores y dispositivos electrónicos como un cajero automático de la figura 2.7[45].

La Raspberry Pi es un ejemplo de sistema embebido de bajo costo y alto rendimiento, capaz de ejecutar sistemas operativos, procesar imágenes y controlar dispositivos externos a través de sus pines GPIO.

Su versatilidad la convierte en una plataforma ideal para el desarrollo de proyectos de automatización y visión artificial, especialmente en entornos educativos e industriales a pequeña escala[46].



Figura 2.7. Ejemplo de sistema embebido de un cajero automático[47].

2.1.13. Aplicación de la visión artificial y la IA en la agricultura

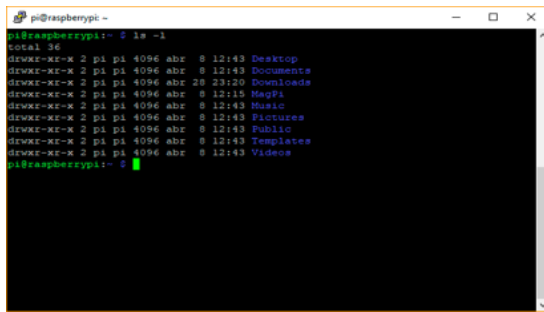
La aplicación de la visión artificial y la inteligencia artificial en la agricultura ha permitido el desarrollo de sistemas inteligentes capaces de mejorar la eficiencia y calidad de los procesos productivos. Estas tecnologías se utilizan en tareas como la detección de enfermedades en cultivos, el monitoreo del crecimiento de plantas y la clasificación automática de frutas[5]. En particular, la integración de redes neuronales convolucionales con sistemas embebidos ha facilitado la creación de soluciones accesibles para pequeñas y medianas empresas, permitiendo automatizar procesos que anteriormente dependían exclusivamente del trabajo manual[45].

El presente proyecto se fundamenta en estos principios, integrando visión artificial, inteligencia artificial y control electrónico para el desarrollo de un sistema automatizado de clasificación de manzanas.

2.2. Generalidades del sistema

2.2.1. Dinámica de funcionamiento de la Raspberry Pi 400

La Raspberry Pi 400 tiene la capacidad de hacer múltiples tareas de un ordenador personal, que van desde configurar, actualizar, y descargar un sistema por medio de la barra de comandos como el de la figura 2.8a, al igual que posee integrado unos puertos tipo USB, Ethernet, HDMI y los GPIO de la figura 2.8b que sirven para la entrada y salida de señales digitales y analógicas de componentes electrónicos externos que pueden interactuar como actuadores[46].



(a) Barra de comandos[46].



(b) Pines GPIO[46].

Figura 2.8. Elementos esenciales de la Raspberry pi 400.

La Raspberry pi 400 funciona como la unidad de control principal del sistema del proyecto, ya que realiza múltiples tareas de programación y activación de los demás elementos electrónicos que lo conforman gracias a la conexión hacia puertos GPIO [46].

A continuación estas son algunas características que conforman la Raspberry Pi 400:

1. Procesador de cuatro núcleos de 64 bits.
2. 4 GB de RAM.
3. Conexión inalámbrica.
4. Salida para dos pantallas.
5. Un conector GPIO de 40 pines que permiten conectar hardware externo.

2.2.2. Banda transportadora

El movimiento de la banda transportadora es simple y muy fácil de manejar, ya que ésta posee un motor tipo TT mostrado en la figura 2.9, que está incrustado a un rodillo principal de la banda transportadora, encargado de tensar la banda de lona y otros elementos de ésta misma. El motor al ser DC, puede invertir el sentido de la polaridad, lo cual indica que puede ir en ambas direcciones dependiendo de cómo se desee usar, el motor puede ser alimentado desde los 5 volts hasta los 12 volts[48].



Figura 2.9. Banda transportadora a escala[48].

La principal función de la banda transportadora dentro del sistema es transportar las manzanas de buen o mal estado para su posterior clasificación[48].

Características principales y especificaciones de la banda transportadora a escala:

- **Dimensiones:** Aproximadamente 54 cm de largo, 10 cm de ancho de banda y 11.5 cm de ancho total.
- **Material:** Cuerpo principal impreso en 3D con plástico PLA y sensible al calor ($>60^{\circ}\text{C}$).
- **Motor:** Motor reductor de corriente continua (DC) de 3 a 6 Volts con engranajes metálicos para un torque adecuado.
- **Componentes:** Banda plástica blanca, tornillos para ensamble y bases tipo clip para sensores (no incluidos).
- **Funcionalidad:** Simular el transporte de materiales a pequeña escala.

2.2.3. Servomotor SG90

El servomotor SG90 funciona recurriendo a un sistema de control de bucle cerrado, mediante la señal de PWM (Modulación por Ancho de Pulso) para mover libremente su eje a un ángulo que va desde los 0 grados a 180 grados. En su interior está conformado por un pequeño motor CC que mueve un conjunto de engranajes y un potenciómetro que informa al circuito el control de la posición actual como se observa en la figura 2.10, permitiendo que se pueda ajustar constantemente por el usuario[49].

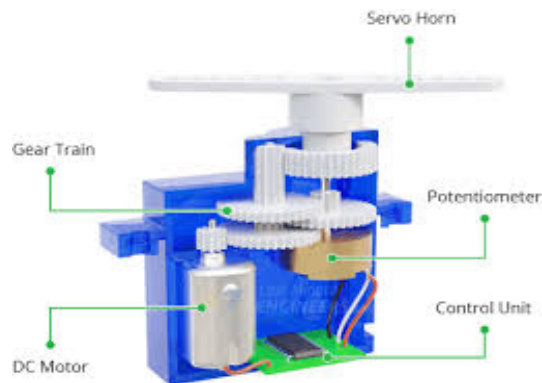


Figura 2.10. Servomotor SG90[49].

La principal función dentro del sistema es el actuador encargado de realizar el movimiento para clasificar las manzanas dependiendo del estado que posean[49].

Principales características y especificaciones del SG90:

- **Modelo:** Modelo SG90.
- **Color:** Azul.
- **Tamaño:** 22.8mm x 12.3mm x 22.5mm.
- **Peso:** 13 g.
- **Grados / Angulo de Rotación Máximo:** 0 grados a 180 grados.
- **Engranajes:** Nylon.
- **Temperatura de trabajo:** -30 a +60 centígrados.

2.2.4. Módulo L298N

El módulo L298N como se observa en la figura 2.11 o comúnmente llamado puente H tiene dos canales de puentes H, haciendo posible controlar 2 motores DC o un motor Paso a Paso, controlando su sentido de giro, la velocidad, y deteniendo la banda transportadora. Su entrada de alimentación puede ser desde los 5 hasta los 12 volts, dependiendo del tipo de motor usado y que función tiene dentro de un sistema[50].

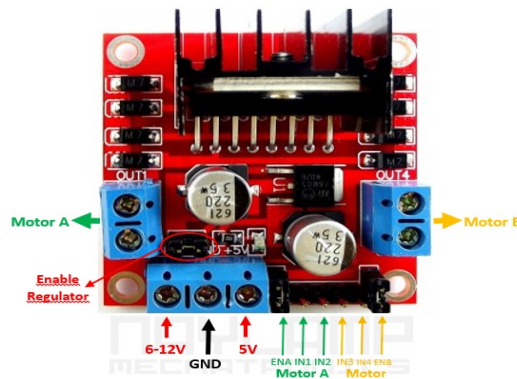


Figura 2.11. Módulo L298N[50].

Su función principal del puente H L298N dentro del sistema es de controlar el motor de la banda transportadora de manzanas, ya sea deteniendo el sistema o iniciándolo por medio de dos pulsadores[50].

Principales características y especificaciones del módulo L298N:

- **Chip:** L298N.
- **Voltajes de alimentación:** 5 V, lógico 5 V, operación 5 V a 35 V.
- **Consumo de corriente digital:** 0 a 36 mA.
- **Capacidad de corriente:** 2A (picos de hasta 3A).
- **Potencia máxima:** 25 Watts.
- **Canales:** 2 (soporta 2 motores DC o 1 motor paso a paso).
- **Dimensiones:** 43 mm x 43 mm x 27 mm.

2.2.5. Cámara HD WEBCAM 4K

La cámara HD WEBCAM de la figura 2.12, funciona capturando la luz con una lente y un sensor de imagen que posee, convirtiendo la luz en datos digitales que un procesador comprime y envía a una computadora decodificando y mostrando el video en tiempo real, permitiendo video llamadas y transmisiones, incluyendo audio gracias a un micrófono integrado[51].

La función principal dentro del sistema es que permite hacer uso de la visión artificial por medio de la librería OpenCV, encargada de monitorear el estado de composición de la manzana para su debida clasificación.



Figura 2.12. HD WEBCAM 4K[51].

Características principales y especificaciones de la cámara WEB:

- **Resolución de Video:** 4K ultra HD de fluidez total para imágenes nítidas y videos.
- **Velocidad de Fotogramas(FPS):** 30 FPS (fotogramas por segundo) para un video fluido.
- **Micrófono:** Incorporado con tecnología de cancelación de ruido para audio claro.
- **Compatibilidad:** Funciona con diversas aplicaciones de videoconferencia como Zoom, Skype, etc.
- **Calidad de Imagen:** Incluye corrección de luz baja automática para mejorar la imagen en condiciones de poca luz.
- **Lente:** Generalmente de gran angular para capturar un campo de visión amplio.

2.2.6. Alimentación (Fuente de alimentación para protoboard v2 MB-102 3.3V 5V)

La alimentación del sistema es bastante importante para que todos los componentes realicen su funcionamiento de manera adecuada, sin embargo, es importante tener en cuenta que los componentes operan con un voltaje mínimo de 5 a 7 V en DC[52].

Es por ello que la MB-102 de la figura 2.13, es una fuente de alimentación modular diseñada específicamente para alimentar de forma práctica y segura. Convierte una entrada de voltaje externo, típicamente de 6.5V a 12V DC (por conector jack o USB), en dos salidas seleccionadas de 3.3V o 5V, lo que la hace ideal para alimentar el resto de los componentes electrónicos. Su funcionalidad se basa en estabilizar el voltaje de entrada mediante el uso de reguladores lineales como el AMS1117, proporcionando así una alimentación constante y adecuada para prototipos electrónicos que requieren múltiples niveles de tensión[52].

La MB-102 Fuente de Alimentación de 3.3V/5V es ampliamente utilizada en el desarrollo de prototipos electrónicos, ya que permite alimentar circuitos de manera sencilla. Sus usos más comunes incluyen proyectos con microcontroladores, pruebas de sensores, sistemas embebidos, y experimentación con módulos de comunicación o actuadores.

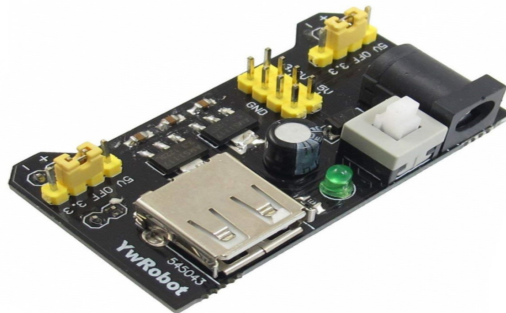


Figura 2.13. Fuente de alimentación protoboard[52].

2.2.7. Pantalla LCD

Display LCD 16*2 figura 2.14, es una pantalla de cristal líquido donde muestra información como texto, números, símbolos y pequeñas figuras. Cuenta con 16 pines para realizar su configuración básica de funcionamiento, con la retroiluminación de color azul y letras blancas o fondo amarillo y letras negras. Es compatible con diversas tarjetas de desarrollo como Arduino, PICt's, STM, AVR, NodeMCU, ESP32, Raspberry Pi, entre otras. Ideal para crear interfaz humano-máquina (HMI) para el control o visualización de información [53].



Figura 2.14. Pantalla LCD de 16*2[53].

Principales características y especificaciones de la pantalla LCD:

- **Tipo:** Pantalla LCD Monocromática
- **Voltaje de alimentación:** 5V DC.
- **Interfaz:** Paralelo 4 y 8 bits (6800 paralelo de 4 bits, y 6800 paralelo de 8 bits.)
- **Controlador:** AIP31066, HD44780, KS0066, SPLC780, ST7066.
- **Color:** Fondo azul y texto blanco Fondo amarillo y texto negro.
- **Número de pines:** 16 pines.
- **Corriente máxima:** 25mA.

2.2.8. Buzzer zumbador

El Buzzer Zumbador pasivo 5V de la figura 2.15, es un piezoeléctrico que se encuentra adherido a un disco cerámico sobre una placa vibrante. La tensión aplicada causa una contracción o expansión del disco cerámico generando sonidos. Utilizado comúnmente para generar alarmas sonoras en tarjetas electrónicas, computadoras, multímetros, en las series de luces de navideñas, mecanismo de señalización, alarma en automóviles u hogar, electrodomésticos y su uso más común es como despertador[54].



Figura 2.15. Buzzer zumbador de 5 volts[54].

Principales características y especificaciones buzzer:

- **Frecuencia de Trabajo:** 100Hz a 10KHz.
- **Corriente de Consumo:** 30 mA (aproximada)
- **Impedancia:** 16 ohms.

Capítulo 3

Diseño y desarrollo del sistema

3.1. Marco Metodológico

En este apartado se describe la estructura y el ensamble del sistema, mediante el cual se verán montados todos los elementos sobre el prototipo de prueba. De igual manera, en el siguiente diagrama de bloques de la figura 3.1 se muestra de como esta constituido el sistema del clasificador automatizado de manzanas en buen y mal estado empleando una red neuronal convolucional (CNN) y su funcionamiento interno.

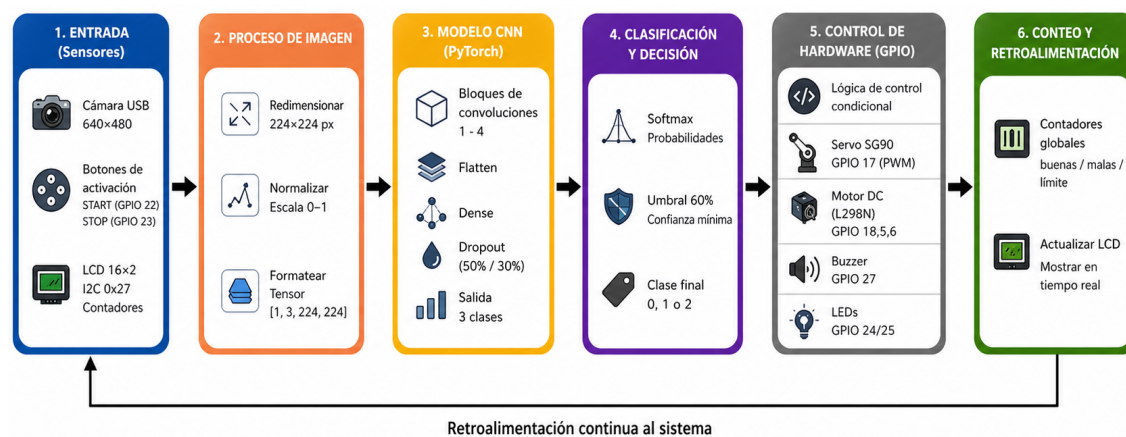


Figura 3.1. Diagrama a bloques constituido del sistema.

3.1.1. Estructura

Para entender la estructura del prototipo se requiere que este sistema por sí solo clasifique las manzanas, por ende debe de tener 3 soportes ideales que cada uno se encarguen de ejecutar una tarea específica dentro del sistema. El primer soporte tipo arco de la figura 3.2, tiene como función sostener la cámara ya que como se ha mencionado con anterioridad es nuestro dispositivo de visión artificial encargada de determinar la composición de las manzanas, el segundo soporte que se debe tener es un tipo resbaladilla como se aprecia en la figura 3.3, con 2 salidas que permitan clasificar las manzanas buenas de las malas y que este soporte tenga implementado un servomotor en medio de este, para que así sirva como un actuador físico encargado de la clasificación. Por último, el tercer soporte de la banda transportadora de la figura 3.4, se basa en el peso y tamaño del objeto que se transporta en este caso el de la manzana, este soporte es el más fundamental ya que tiene incluido un motor tipo TT que al igual que el servomotor del soporte 2 sirve como actuador para que mueva la banda del soporte 3. El diseño de este sistema está compuesto de 2 tipos de materiales que es la madera MDF para el soporte del arco y el soporte tipo resbaladilla, y el otro soporte de la banda transportadora está hecho de impresión 3D con plástico PLA.

3.1.2. Soporte tipo arco

Uno de los principales elementos que integran el sistema es la cámara WEB, por lo que ésta debe ir a una parte fija superior para que pueda monitorear la calidad de las manzanas todo el tiempo. Derivado de este requisito para el sistema, se desarrolló una estructura tipo arco como soporte de la cámara WEB, este soporte está hecho de madera MDF y de residuos de madera reutilizada como se muestran en la figura 3.2.



(a) Soporte tipo arco.



(b) Soporte tipo arco con LEDS

Figura 3.2. Soporte tipo arco del clasificador.

3.1.3. Soporte tipo resbaladilla

El soporte tipo resbaladilla de la figura 3.3 al igual que el soporte tipo arco se realizó de madera MDF, y para su diseño se imaginó que fuera una forma tipo V para separar las manzanas buenas de las malas, de igual manera se incrustó en medio de este soporte el servomotor SG90, ya que como se ha mencionado antes este es el encargado de clasificar las manzanas.



(a) Soporte tipo resbaladilla.



(b) Soporte tipo resbaladilla de costado.

Figura 3.3. Resbaladilla del clasificador de manzanas.

3.1.4. Soporte pasa-bandas

El soporte del pasa-bandas está hecho de impresión 3D de plástico PLA como se visualiza en la figura 3.4, este soporte se adquirió de una tienda oficial de electrónica para proyectos, y toda la estructura fue diseñada por el fabricante en modelo 3D.

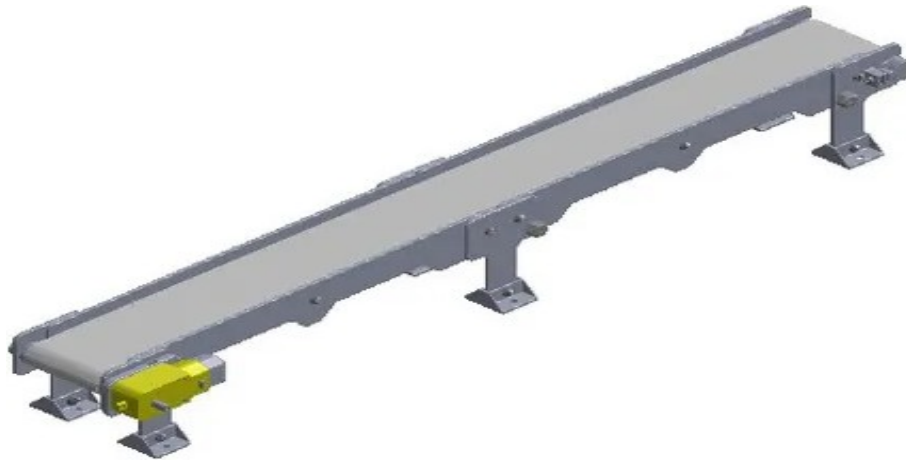


Figura 3.4. Diseño 3D del pasa-bandas[48].

3.1.5. Diagrama constituido del circuito

A continuación, se muestra el diagrama de la figura 3.5 la conexión del circuito del sistema con los componentes electrónicos que lo conforman.

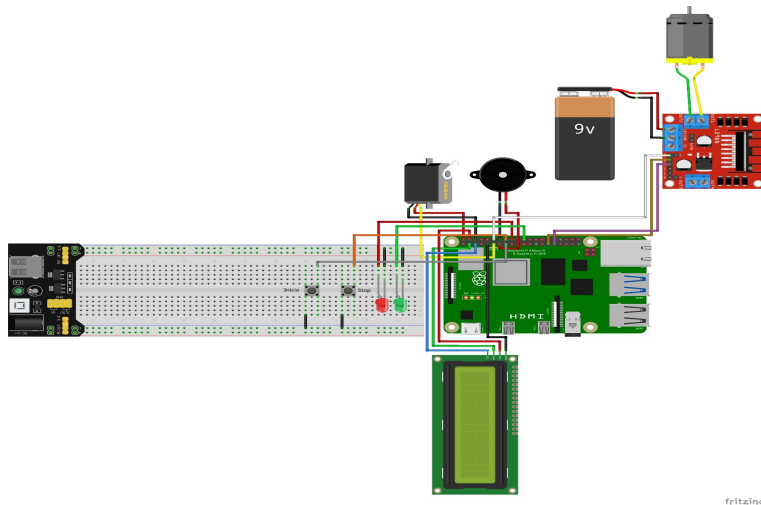
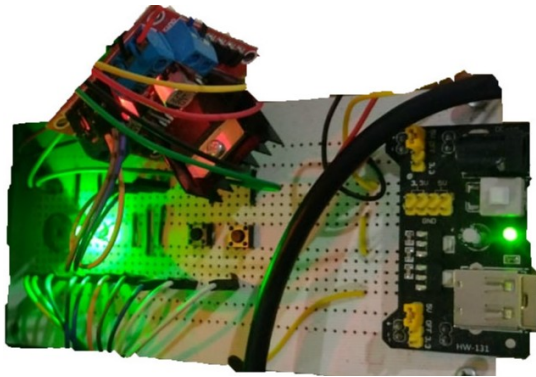


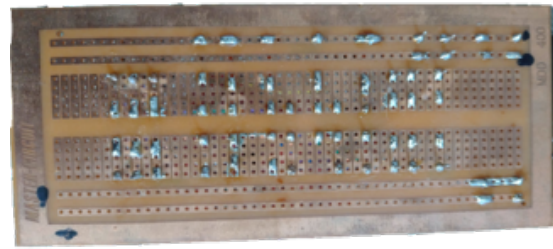
Figura 3.5. Diagrama constituido del circuito[48].

3.1.6. Circuito

El circuito del sistema está realizado en una placa fenólica tipo protoboard, ya que esta placa facilita el soldado de los componentes requeridos y es de bajo costo. El circuito tiene los componentes electrónicos soldados como los son jumpers, buzzer, LEDs, botones, cables para soldar y fuente de voltaje para protoboard de 5 volts como se observan en la figura 3.6a y la figura 3.6b.



(a) Circuito con los componentes físicos.



(b) Circuito soldado.

Figura 3.6. Circuito soldado del sistema del clasificador de manzanas.

3.1.7. Pytorch

El PyTorch es un código de biblioteca de aprendizaje profundo basado en Python, ésta librería permite el desarrollo de redes neuronales profundas, entrena modelos IA, así como el uso visión por computadora, y procesamiento de lenguaje natural [55]. El pytorch en este sistema activa un entorno virtual (interacción de datos y objetos en tiempo real) facilitando la interacción de usuario-máquina, de igual manera genera scripts para realizar tareas específicas de nuestra red neuronal CNN como el entrenamiento, la captura de imágenes, etcétera.

3.1.8. Red neuronal CNN

El sistema utiliza una Red Neuronal Convolutiva (CNN), ésta red específicamente diseñada para la clasificación de imágenes y su arquitectura está conformada mediante bloques como en la figura 3.7, cada bloque es capaz de realizar múltiples tareas.

Ejemplo de ello es la extracción de características visuales de un objeto, en este caso las manzanas son los objetos donde se le extrae los caracteres más simples como líneas y texturas hasta lo más complejo como la identificación completa de la manzana.

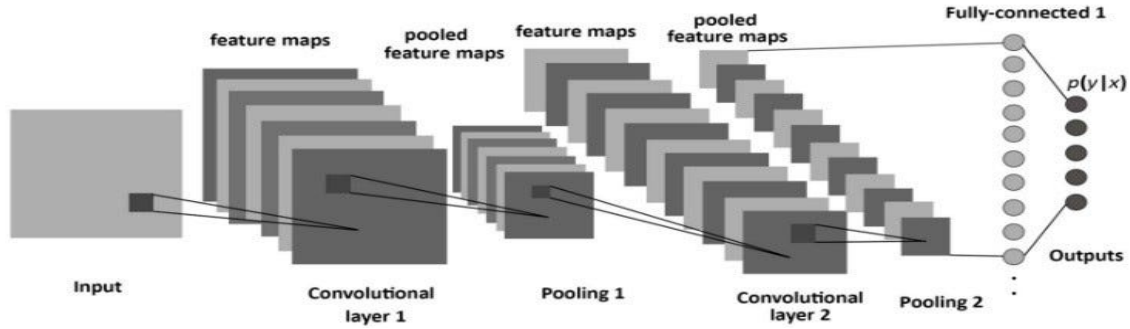


Figura 3.7. Arquitectura de una red neuronal CNN[56].

3.1.9. Entrenamiento

La red neuronal funciona a través de 7 bloques convolucionales progresivos, cada bloque tiene una función en extraer caracteres por una cierta cantidad de filtros cada uno [9]. Como se muestra en la figura 3.8 el bloque 1 posee 32 filtros, bloque 2 posee 64 filtros, bloque 3 y 4 posee la misma cantidad de 128 filtros.

Estos bloques pasan por un proceso interno de convolución, normalización por lote, activación de ReLU, y Max Pooling.

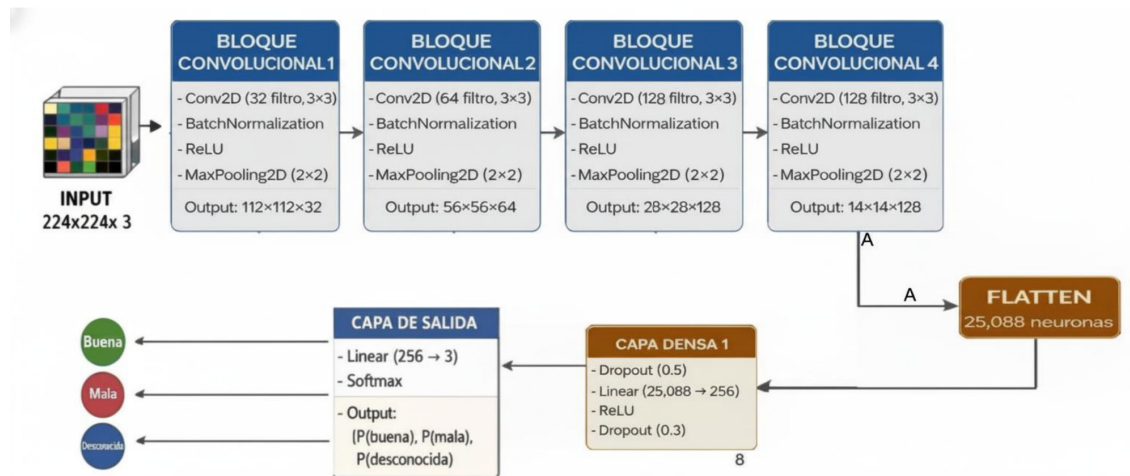


Figura 3.8. Diagrama a bloques del entrenamiento de la CNN.

Posterior a los 4 bloques convolucionales pasa por otros 2 bloques restantes el cual es el flatten (capa densa 1) y una capa de salida. Como en la figura 3.9 , el input (entrada de datos) tenemos una entrada de 224 píxeles* 224 píxeles* 3 canales de colores de las características visuales de entrada extraídas de las manzanas mediante imágenes tomadas por la cámara WEB, una vez pasado el proceso de la red neuronal tendrá 3 resultados de salida para la clasificación de la manzana el cual son buena, mala, y desconocida.

Del bloque 1 al bloque 4 realiza procesos internos como Conv2D (32, 64, y 128 filtros 3*3), BatchNormalization, ReLU, MaxPooling2D (2*2), el cual hace posible detectar bordes simples, líneas, y texturas básicas de un objeto, las salidas convolucionales de dichos bloques son de 112*112*32, 56*56*64, 28*28*128, y 14*14*128.

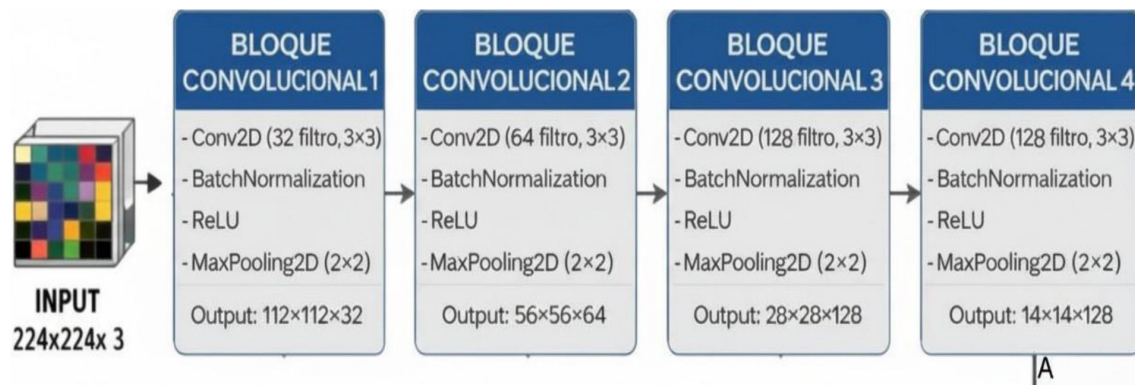


Figura 3.9. Diagrama bloques del 1 al 4 del entrenamiento de la CNN.

Posteriormente el bloque de flatten realiza el proceso de combinar las características aprendidas de las neuronas de salida generadas por los 4 bloques anteriores, por último bloque de capa de salida da como resultado la salida de 3 neuronas finales para la clasificación de las manzanas como se observa en la figura 3.10.

- **Neurona 0:** Manzana buena.
- **Neurona 1:** Manzana mala.
- **Neurona 2:** Objeto desconocido.

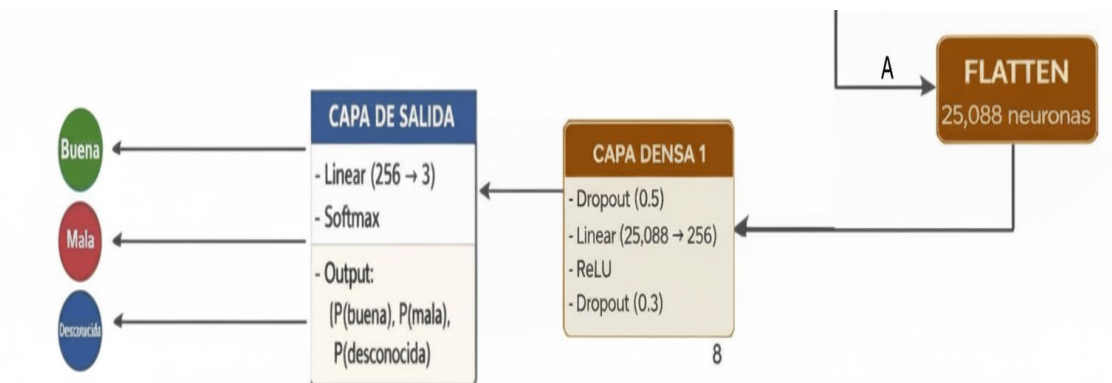


Figura 3.10. Diagrama bloques de capa densa y capa de salida de la CNN.

Los procesos internos de cada bloque son esenciales, debido a que normalizan las activaciones entre capas (batch normalization), reducen las dimensiones espaciales 2*2 (Max Pooling), desactiva aleatoriamente las neuronas durante el entrenamiento (dropout), combina características aprendidas (capas densas), y activan la función más utilizada en redes neuronales.

3.1.10. Época

Todo este proceso de entrenamiento se hace por época, la cuál no es más que un ciclo completo donde nuestro modelo revisará y analizará las imágenes una sola vez de cada imagen de las manzanas del entrenamiento de la red neuronal, en este caso son de 25 épocas para el sistema del clasificador de manzanas, lo que significa que el entrenamiento revisará y analizará 25 veces las 722 imágenes tomadas de manzanas buenas, así como de las 722 imágenes de manzanas malas, y más 100 imágenes de objetos que no son manzanas.

3.1.11. Programación

La programación del clasificador de manzanas está basada en el lenguaje Python debido a su versatilidad, fácil manejo, su amplia variedad de bibliotecas y demás propiedades que llega ofrecer el lenguaje. Para la programación se importó librerías para las funciones de hardware y software que se requirió para la funcionalidad del sistema. Las librerías importadas son las siguientes:

- **Import cv2:** Captura frames de la cámara, re-dimensiona, dibujos en pantalla del monitor.
- **Import numpy as np:** Procesa arrays y operaciones matemáticas.
- **Import torch:** Modelo de IA.
- **Import torch.nn as nn :** Redes neuronales.
- **From torchvision import transforms:** Procesamiento de imágenes.
- **Import time:** Temporizadores y delays.
- **Import os:** Sistema de archivos.
- **Import RPi.GPIO as GPIO:**Control de pines GPIO.
- **from RPLCD.i2c import CharLCD:**Display LCD I2C.

3.1.12. Diagrama de flujo del funcionamiento.

Como se ha mencionado con anterioridad, el sistema realizará múltiples tareas en plena ejecución para la clasificación de manzanas. A continuación se representa en la siguiente figura 3.11 un diagrama de flujo del sistema donde muestra genéricamente el funcionamiento más importante de cada proceso.

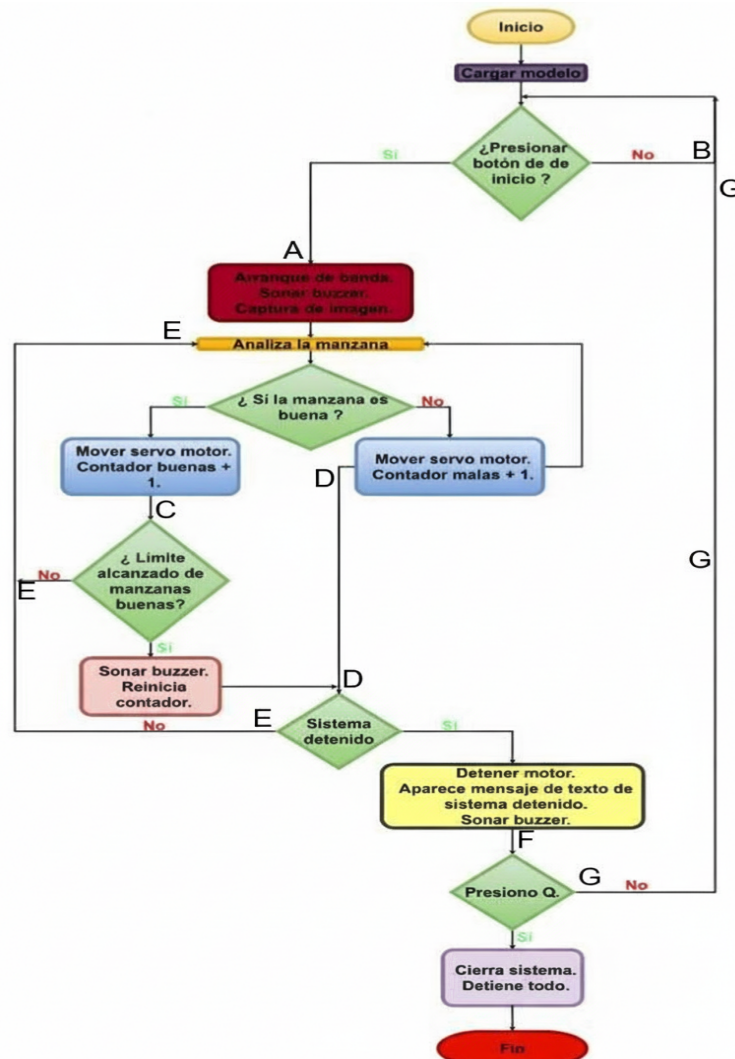


Figura 3.11. Diagrama de flujo del sistema del clasificador de manzanas.

Como se puede apreciar en la figura 3.12, el sistema del clasificador de manzanas inicia con la carga del modelo que a su vez en ese momento incluye la configuración de las partes del hardware como los puertos GPIO, servomotor SG90, buzzer, motor, etcétera.

Posteriormente el sistema se activará presionando el botón de inicio.



Figura 3.12. Parte de inicio del sistema.

En la figura 3.13 muestra si en caso de presionar el botón de inicio, este activará el sistema iniciando con la captura de imagen, el arranque de la banda, el zumbido de buzzer y seguido se analizará la manzana en tiempo real para su debida clasificación. Y en caso de que no sea presionado, solo se tendrá cargado el modelo.

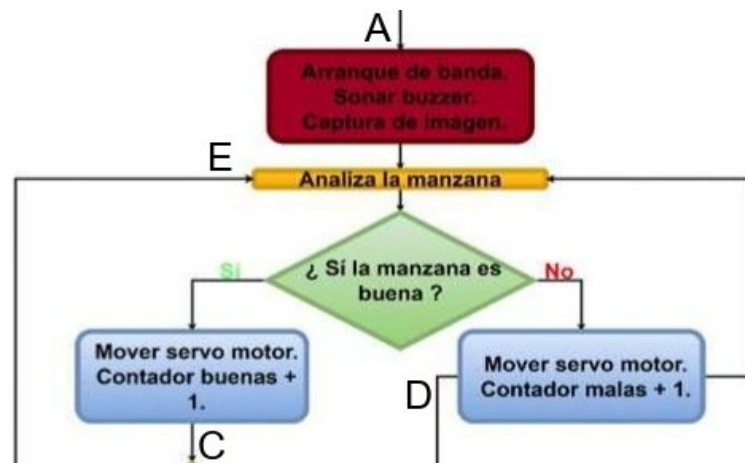


Figura 3.13. Activación del sistema.

Cuando se detecta las manzanas buenas y malas estas poseen un límite cada una establecidas, cuando este límite es alcanzado se reinicia el contador de cada una, si en caso de que el sistema quiere ser detenido se presionará un botón de pausa donde se detendrá la ejecución inmediatamente y aparecerá un mensaje que diga sistema detenido, como se indica en la figura 3.14.

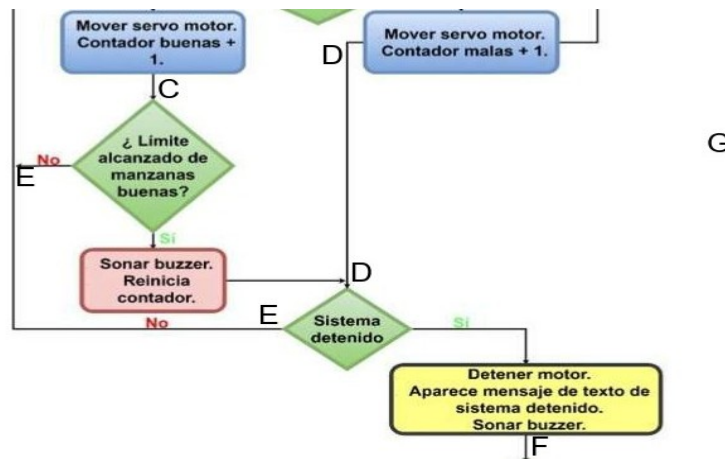


Figura 3.14. Ejecución del sistema.

Si se desea finalizar por completo la ejecución del sistema se presionará la tecla Q, la cual inmediatamente cerrará la ventana de visión artificial y apagará el sistema. Pero en caso contrario solo se tendrá que presionar otra vez el botón de inicio para seguir continuando con el sistema como se muestra en la figura 3.15.



Figura 3.15. Finalización total del sistema.

3.1.13. Montaje final

Una vez terminada la programación del entrenamiento y la detección de manzanas, se montó el sistema de la estructura y la conexión del circuito a la RASPBERRY PI 400.



Figura 3.16. Montaje de la estructura.

Uniendo las 3 estructuras principales del sistema, se conectó a una pantalla para ver los resultados finales del clasificador.

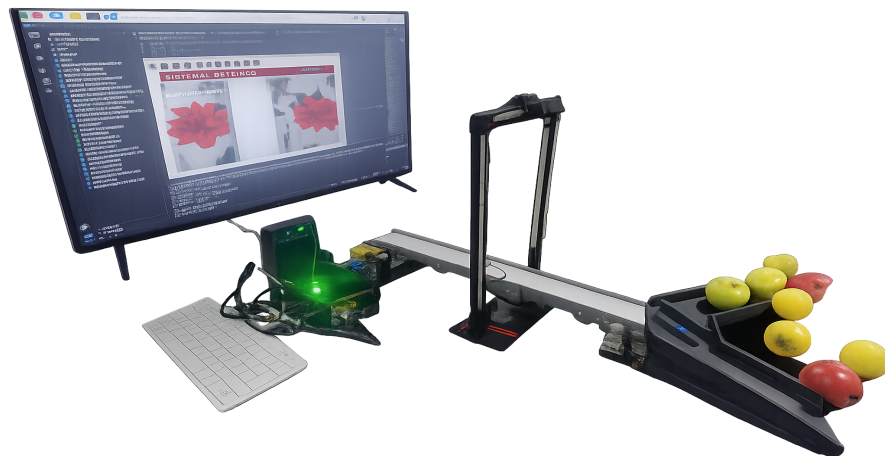


Figura 3.17. Montaje final del sistema.

En la figura 3.17, se observa la conexión correctamente realizada, al igual que la captación de imagen en tiempo real mostrada en la pantalla mediante la cámara WEB.

3.1.14. Pruebas

Concluida la etapa de montaje, se dedicó a hacer algunas pruebas para comprobar la efectividad de la clasificación de las manzanas del sistema. Las pruebas se hicieron con 3 tipos de variedad de manzana de diferentes colores las cuales fueron rojo, amarillo, y verde en buen estado y mal estado como en la figura 3.18. Cabe mencionar que estas pruebas se hicieron decenas de veces en diferentes partes del día, por lo cual arrojaron diferentes resultados durante el proceso de clasificación, así como el nivel de confianza y contador mostrado en el monitor y la pantalla LCD.



Figura 3.18. Manzanas de prueba.[57].

Capítulo 4

Análisis y discusión de resultados

4.1. Resultados

Los resultados obtenidos en las pruebas se observaron que algunas de las manzanas en buen estado si fueron detectadas como buenas manzanas por el sistema del clasificador, por ejemplo las manzanas verdes como las de la figura 4.1, las amarillas de la figura 4.2 y rojas si fueron detectadas correctamente.

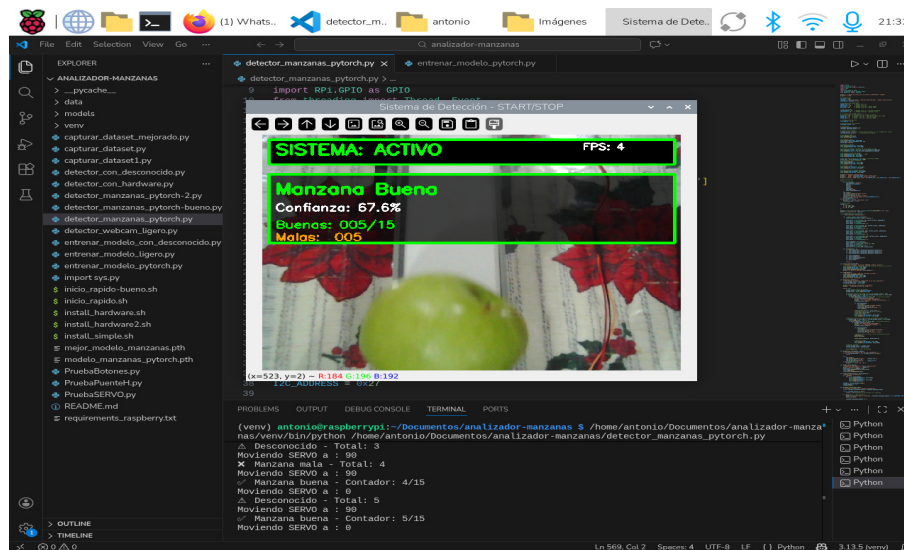


Figura 4.1. Manzana buena de color verde.

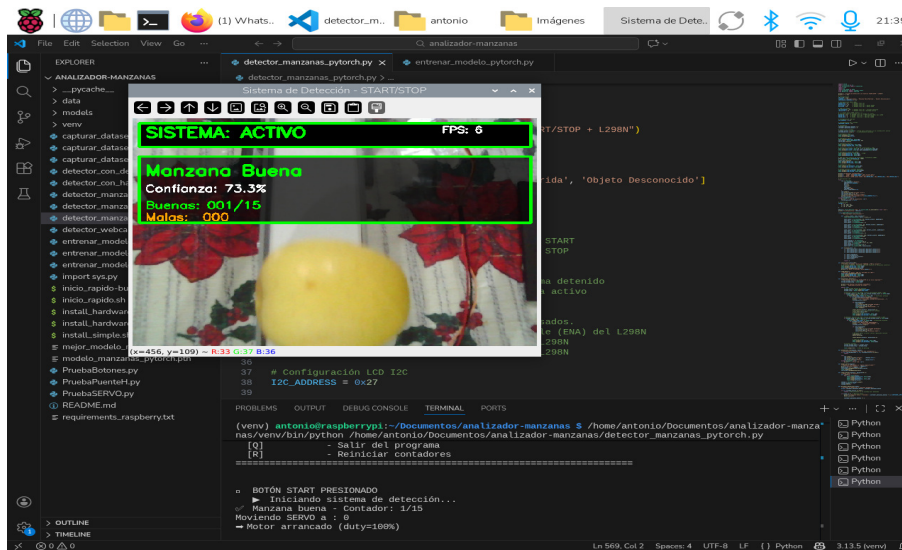


Figura 4.2. Manzana buena de color amarillo.

En algunos casos las manzanas de color rojo y amarillo en buen estado se detectaron como manzanas en mal estado o objeto no reconocido como se observa en la figura 4.3, este puede deberse a ciertos factores como el tipo de imágenes que se usaron durante el entrenamiento, ya sea por la luz, las fotos, los fondos, o el tipo de cámara en cuanto a su resolución que ésta posee.

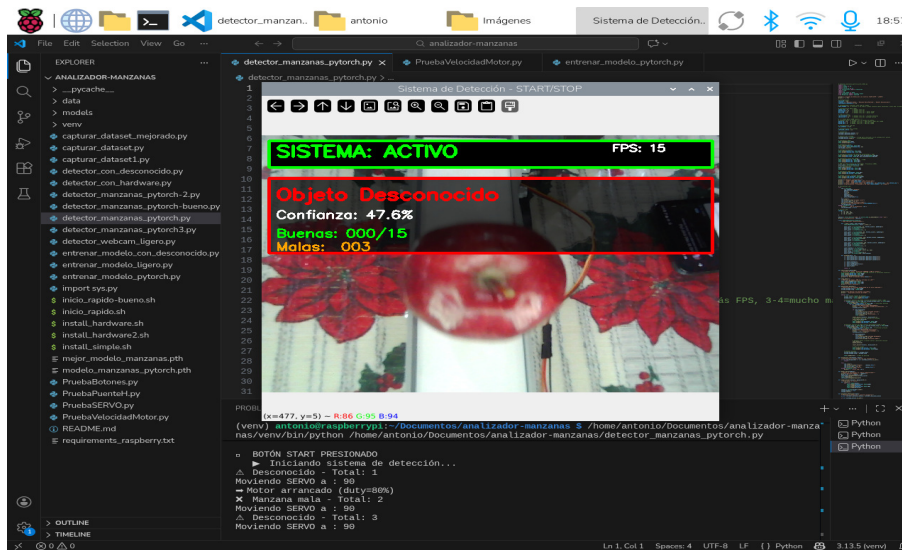


Figura 4.3. Manzana buena de color roja.

Posterior a la prueba de manzanas de calidad se hizo otra prueba con manzanas en mal estado, los resultados fueron alentadores debido a que logró detectar las manzanas malas clasificándolas del lado izquierdo del soporte de la resbaladilla del sistema.

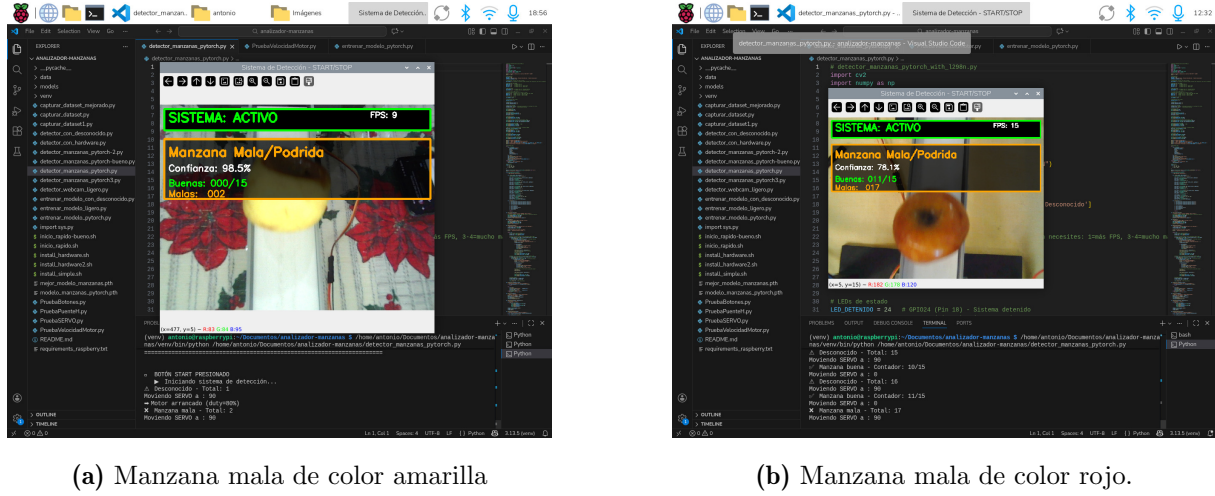


Figura 4.4. Resultados de las manzanas malas.

Como se aprecia en la figura 4.4a y la figura 4.4b, las manzanas que claramente están en mal estado son detectadas con la leyenda de (mala/podrida) seguido del nivel de confianza detectado y generado automáticamente.

Lo que demuestra que la arquitectura de nuestra red neuronal es efectiva a pesar de que llega a cometer algunos errores en cuestión de las manzanas de calidad de color rojo y amarillo. Ante este pequeño problema se reentrenó la red neuronal, además al soporte de la cámara se le colocó una tira LED de color blanco, esto para que iluminara durante la transportación por medio de la banda para ver si era cuestión de la luz lo que afectaba nuestro sistema.

Como se observa en la figura 4.5 la manzana roja de calidad se detectó como manzana buena, indicando que la luz afecta la clasificación del sistema, lo que sugiere que el sistema puede detectar erróneamente una manzana ya sea de buena calidad o mala calidad.

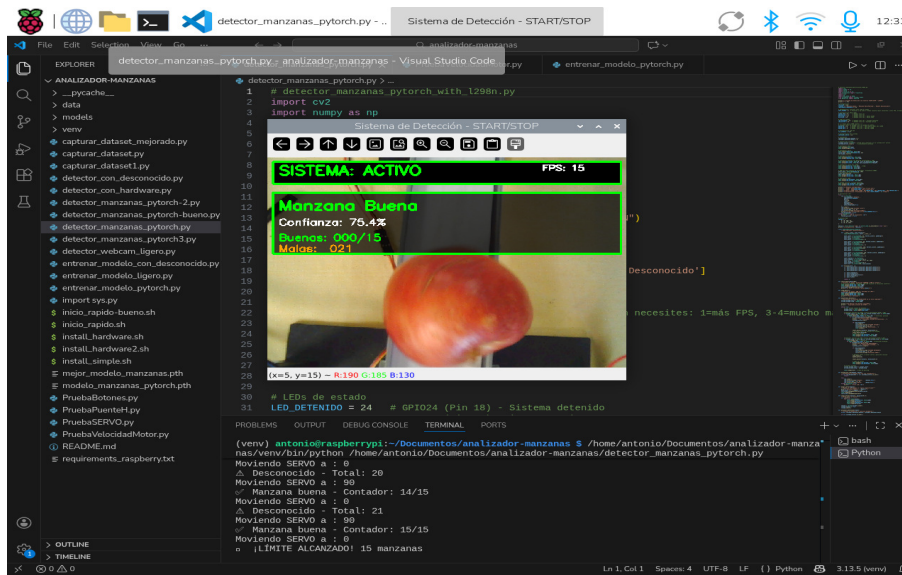


Figura 4.5. Manzana buena de color roja.

También se verificó si los mensajes mostrados por la pantalla LCD concuerdan con los mostrados en la pantalla del monitor, el resultado fue que efectivamente el sistema mostraba en la pantalla LCD los mismos resultados observados en el monitor en tiempo real como se ilustran en la figura 4.6.



(a) Pantalla LCD del clasificador.



(b) Contador en la pantalla LCD.

Figura 4.6. Resultados mostrados en la pantalla LCD.

Por último se comprobó que el sistema funcionó correctamente indicado por un LED de color verde en el circuito, en donde este se ilumina en el momento que el sistema está siendo ejecutado como en las figuras 4.7.



(a) Sistema transportando la manzana.



(b) Clasificación de las manzanas.

Figura 4.7. Resultados de la clasificación de las manzanas.

Capítulo 5

Conclusiones

5.1. Conclusión

La importancia de la seguridad en frutas y verduras hoy en día es indispensable para la salud y la calidad del consumidor como del proveedor de dicho producto alimenticio. No obstante este proceso de selección de calidad sigue siendo de manera manual por personas que llegan a cometer más errores de los esperados, el desarrollo y la implementación de tecnologías como la automatización y la IA son una alternativa a estos y otros problemas presentados en el sector agrícola. El desarrollo de proyectos como este sistema clasificador automatizado de manzanas en buen y mal estado empleando una red neuronal convolucional (CNN), a pesar de ser un proyecto de tesis, estos proyectos tienen potencial para contribuir al desarrollo del sector, y en cuanto al sistema se determinó que es funcional en el proceso de la clasificación de las manzanas de prueba verde, amarilla y roja.

5.2. Limitaciones

Sin embargo surgieron algunos errores en la clasificación de manzanas a causa de factores que rodean el entorno del sistema como lo es la luz y el movimiento que detecta la cámara, estos 2 factores cambian la percepción del resultado de la manzana. Asimismo la librería Pytorch que fue usada para diseñar y entrenar la red neuronal CNN a pesar de cumplir con su función ésta era la segunda opción en usar, ya que primeramente se optó por usar TensorFlow, no obstante las limitaciones de capacidad de la Raspberry Pi 400 no fue posible emplear dicha librería.

5.3. Trabajo a futuro

Como se ha mencionado la implementación de tecnología automatizada y la IA no solamente será el futuro de esta industria, sino que de otras más para facilitar aquellos problemas que han existido desde inicios de la humanidad. Por esa misma razón este proyecto a futuro, se propone a ampliar el conjunto de datos de entrenamiento con más imágenes reales y variables, de igual manera implementar un análisis estadístico de los resultados de falsos positivos y falsos negativos con el fin de determinar y mejorar la precisión del modelo de la manzana.

También con la implementación de arquitecturas más avanzadas en redes neuronales mediante diferentes técnicas de aprendizaje, finalmente se pretende extender la funcionalidad del sistema, no solo para clasificar manzanas, sino también la adaptación a otros productos agrícolas.

Bibliografía

- [1] Datatechnic. Antecedentes de la automatización industrial. <https://datatechnic.com.mx/antecedentes-de-la-automatizacion-industrial/>, October 2024. [En línea; accedido: 18-abr-2026].
- [2] Learning Heroes. Ventajas y desventajas de la tecnología: la bella y cruda verdad. <https://www.learningheroes.com/blog/aprende-sobre-tecnologias-disruptivas/ventajas-y-desventajas-de-las-tecnologias-explorando-su-impacto-en-la-sociedad-contemporanea>, June 2025. [En línea; accedido: 18-abr-2026].
- [3] David Alexander Matos. Expoindustrial 2025: la automatización impulsa la transformación de la industria en el suroccidente colombiano. <https://www.elpais.com.co/informacion-comercial/expoindustrial-2025-la-automatizacion-impulsa-la-transformacion-de-la-industria-en-el-suroccidente-colombiano-0701.html>, May 2025. [En línea; accedido: 18-abr-2026].
- [4] Autonomación, automatización, inteligencia artificial e industria-agricultura 5.0 en la producción agroecológica sostenible de alimentos. https://www.iade.org.ar/noticias/autonomacion-automatizacion-inteligencia-artificial-e-industria-agricultura-50-en-la?utm_source=chatgpt.com, September 2024. Accessed: 2026-2-17.
- [5] Interempresas. Visión artificial en la industria alimentaria. <https://www.interempresas.net/Robotica/180292-Vision-artificial-en-la-industria-alimentaria.html>, 2026. [En línea; accedido: 18-abr-2026].
- [6] Gobierno de México. Productores deben aprovechar programas de apoyo de sagarpa y firco. <https://www.gob.mx/agricultura/campeche/articulos/productores-deben->

- [aprovechar-programas-de-apoyo-de-sagarpa-y-firco](#), 2025. [En línea; accedido: 18-abr-2026].
- [7] Gaceta UNAM. Se lentifica el crecimiento de la población mundial. <https://www.gaceta.unam.mx/se-lentifica-el-crecimiento-de-la-poblacion-mundial/>, February 2023. [En línea; accedido: 18-abr-2026].
- [8] Rodrigo Salas. Redes neuronales artificiales. *Universidad de Valparaíso. Departamento de Computación*, 1(1):1–7, 2004.
- [9] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, et al. Deep convolutional neural networks for computer-aided detection. *arXiv preprint arXiv:1602.03409*, 2016.
- [10] Wario Duckerman. Qué es la visión por computadora: descripción general, aplicaciones y beneficios. <https://brita.mx/que-es-la-vision-por-computadora-descripcion-general-aplicaciones-beneficios-y-como-aprenderla/>, November 2021. [En línea; accedido: 18-abr-2026].
- [11] Futuro Eléctrico. Redes neuronales: ¿la panacea de la inteligencia artificial? <https://futuroelectrico.com/redes-neuronales-inteligencia-artificial/>, April 2022. [En línea; accedido: 18-abr-2026].
- [12] Reynaldo Sucari León, Yolanda Aroquipa Durán, Edgardo Quispe Yapo, Anibal Sucari León, Luz Delia Quina Quina, and Fredy Abel Huanca Torres. VISIÓN ARTIFICIAL EN RECONOCIMIENTO DE PATRONES PARA CLASIFICACIÓN DE FRUTAS EN AGRONEGOCIOS. *Puriq*, 2(2):109–118, 2020.
- [13] Robbie Dickson. ¿Cuál es la evolución y la historia de la automatización?, Jan 2023.
- [14] Autoelec. EVOLUCION DE LA AUTOMATIZACION | hasta la actualidad.
- [15] Manufactura Latam. Evolución de la automatización industrial: de la máquina de vapor a la industria 4.0. <https://www.manufactura-latam.com/noticias/es/evolucion-de-la-automatizacion-industrial>, July 2024. [En línea; accedido: 18-abr-2026].
- [16] Juan C Olguín-Rojas, Juan I Vasquez-Gomez, Gilberto de J López-Canteñs, and Juan C Herrera-Lozada. CLASIFICACIÓN DE MANZANAS CON REDES NEURONALES CONVOLUCIONALES. *Rev. Fitotec. Mex.*, 45(3):369, September 2022.

- [17] Juan Pablo Tinoco Mera and José Luis Tinoco Mera. *Clasificación de manzanas mediante visión por computador e inteligencia artificial para detectar anomalías*. PhD thesis, Institution Name, 2025.
- [18] C Mota-Delfin, C Juárez-González, and J C Olguín-Rojas. Clasificación de manzanas utilizando visión artificial y redes neuronales artificiales. *Ing. Reg.*, 20:52–57, December 2018.
- [19] Ismael Domínguez Jiménez, Alejandro Mendoza, Gamiño Melecio, and Sánchez Ruiz. Clasificador automático de frutas y verduras. <https://dspace.uaeh.edu.mx/server/api/core/bitstreams/9af20b4c-820b-4f9d-80eb-0b66b94b4d4d/content>, July 2020. [En línea; accedido: 18-abr-2026].
- [20] Ricardo Paúl. Diseño de un prototipo de sistema de clasificación de manzanas según su defecto mediante técnicas de visión artificial para la inspección de calidad en procesos industriales. <https://repositorio.upse.edu.ec/items/c4452acf-820d-41a8-9542-0f9124c029fb>, April 2024. [En línea; accedido: 18-abr-2026].
- [21] Meirav Nachmanovitch Lazar. Industrial automation: embracing the future of industry 5.0. <https://matics.live/blog/embracing-the-future-industry-5-0-and-industrial-automation/>, December 2023. [En línea; accedido: 18-abr-2026].
- [22] STI Corporate. The role of automation in achieving sustainable manufacturing goals. <http://sti-corporate.com/news/the-role-of-automation-in-achieving-sustainable-manufacturing-goals>, April 2025. [En línea; accedido: 18-abr-2026].
- [23] Jerry James Stone. History of apples: What am i even eating? <https://jerryjamesstone.com/how-to/history-of-apples-what-am-i-even-eating/>, January 2026. [En línea; accedido: 18-abr-2026].
- [24] AgroAvances. Estudio genómico revela cómo la manzana se originó en asia y se extendió por el mundo. <https://agroavances.com/noticias-detalle.php?idNot=943>, 2017. [En línea; accedido: 18-abr-2026].
- [25] Antología de Letras. Agricultura, minería y manufactura. <https://antologiadeletras.wordpress.com/acerca-de/>, July 2017. [En línea; accedido: 18-abr-2026].

- [26] AyJ Transmisiones. Historia de las bandas transportadoras. <https://www.ajtransmisiones.com/blog/historia-de-las-bandas-transportadoras>, July 2021. [En línea; accedido: 18-abr-2026].
- [27] Maquisur. Beneficios de las cintas transportadoras de fruta. <https://maquinariamaquisur.com/cintas-transportadoras-frutas-beneficios/>, February 2025. [En línea; accedido: 18-abr-2026].
- [28] Meprosa. Báscula camionera: un elemento básico para las operaciones logísticas. <https://meprosa.mx/bascula-camionera-un-elemento-basico-para-las-operaciones-logisticas/>, August 2022. [En línea; accedido: 18-abr-2026].
- [29] Meirav Nachmanovitch Lazar. Industrial automation: Embracing the future of industry 5.0. <https://matics.live/blog/embracing-the-future-industry-5-0-and-industrial-automation/>, December 2023. Accessed: 2026-2-9.
- [30] PNGTree. Imágenes de manzanas png. <https://es.pngtree.com/free-png-vectors/manzanas>, 2018. [En línea; accedido: 18-abr-2026].
- [31] Excelencias del Motor. 100 años de ford en cadena. <https://www.excelenciasdelmotor.com/noticia/100-anos-de-ford-en-cadena-o-cuando-ford-reinvento-la-industria-del-automovil>, 2016. [En línea; accedido: 18-abr-2026].
- [32] Maquinados Industriales. Fábrica y reparación de bandas de transporte. <https://www.maquinados-industriales-cdmx.com.mx/bandas-de-transporte/>, July 2023. [En línea; accedido: 18-abr-2026].
- [33] Dmitri Zelenevski. Manzanas en transportador (imagen). <https://es.vecteezy.com/foto/46967008-otono-manzana-cosecha-procesando-fresco-manzanas-en-transportador-cinturon-con-otono-hojas>, 2026. [En línea; accedido: 18-abr-2026].
- [34] CatSensors. Esterilización de máquinas embotelladoras. <https://www.catsensors.com/es/catsensors/catnews/esterilizacion-de-maquinas-embotelladoras>, 2022. [En línea; accedido: 18-abr-2026].

- [35] Bosch. Bosch inaugura su primera fábrica de electrodomésticos en México. <https://www.bosch.com.mx/noticias-e-historias/bosch-inaugura-su-primera-fabrica-de-electrodomesticos-en-mexico/>, 2022. [En línea; accedido: 18-abr-2026].
- [36] J-Flex. Materiales de grado alimenticio. <https://www.j-flex.com/es/productos/productos-y-laminas-de-caucho-de-silicona-de-grado-alimenticio-y-seguros-para-uso-alimentario/>, December 2023. [En línea; accedido: 18-abr-2026].
- [37] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2008.
- [38] Delego. El futuro de la logística. <https://www.delego.ai/blog/el-futuro-de-la-logistica-mas-alla-de-la-ruta-hacia-la-inteligencia>, 2025. [En línea; accedido: 18-abr-2026].
- [39] JJ Esqueda Elizondo and L Palafox Maestre. Fundamentos de procesamiento de imágenes. *Universidad Autónoma de Baja California, México*, page a1, 2005.
- [40] Bytelix. Entender las características de una pantalla, September 2017. Consultado el 25 de abril de 2026.
- [41] Alejandro Domínguez Torres. *Procesamiento digital de imágenes*. Universidad Nacional Autónoma de México, 1996.
- [42] Vistronica. Procesamiento de imágenes. <https://www.vistronica.com/blog/post/procesamiento-de-imagenes>, September 2020. [En línea; accedido: 18-abr-2026].
- [43] Lasse Rouhiainen. Inteligencia artificial. *Madrid: Alienta Editorial*, pages 20–21, 2018.
- [44] Chequeado. Cómo puede ayudarte la ia para trabajar con bases de datos, October 2024. Consultado en abril de 2026.
- [45] D Pérez. Sistemas embebidos y sistemas operativos embebidos. *Lecturas en ciencias de la computación. Universidad Central de Venezuela, Vols. % i de % 2lSSN*, pages 1316–6239, 2009.
- [46] Raspberry Pi Ltd. Buy a raspberry pi 400 unit. <https://www.raspberrypi.com/products/raspberry-pi-400-unit/>, 2026. [En línea; accedido: 16-feb-2026].

- [47] Carlos Silva. Cancelación de cargos no reconocidos por retiros del cajero automático. <https://mejorabogado.mx/2017/09/05/cancelacion-de-cargos-no-reconocidos-por-disposicion-de-efectivo-en-cajeros-automaticos/>, September 2017. [En línea; accedido: 18-abr-2026].
- [48] Mercado Libre. Banda transportadora a escala de 54 cm para estudiantes. <https://www.mercadolibre.com.mx/banda-transportadora-a-escala-54-cm-programarestudiantes/up/MLMU521842456>, 2024. [En línea; accedido: 18-abr-2026].
- [49] UNIT Electronics. Servomotor sg90 rc 9g micro servo para arduino. <https://uelectronics.com/producto/servomotor-sg90-rc-9g/>, April 2026. [En línea; accedido: 18-abr-2026].
- [50] UNIT Electronics. Módulo driver l298n para motor a pasos. <https://uelectronics.com/producto/l298n-modulo-driver-motor-a-pasos/>, April 2026. [En línea; accedido: 18-abr-2026].
- [51] Mercado Libre. Cámara web 4k uhd ourlife con micrófono y enfoque automático usb con trípode extensible. <https://www.mercadolibre.com.mx/p/MLM62575134>, 2024. [En línea; accedido: 18-abr-2026].
- [52] UNIT Electronics. Fuente de alimentación para protoboard mb-102 3.3v / 5v. <https://uelectronics.com/producto/mb-102-fuente-de-alimentacion-para-protoboard-3-3v-5v/>, April 2026. [En línea; accedido: 18-abr-2026].
- [53] UNIT Electronics. Display lcd 16x2 fondo azul/amarillo. <https://uelectronics.com/producto/display-lcd-16x2-fondo-azul-amarillo/>, April 2026. [En línea; accedido: 18-abr-2026].
- [54] UNIT Electronics. Buzzer (zumbador) pasivo de 5v. <https://uelectronics.com/producto/buzzer-zumbador-5v-pasivo/>, April 2026. [En línea; accedido: 18-abr-2026].
- [55] David García Fernández. Aprendizaje a gran escala mediante procesos gaussianos usando pytorch y gpus, 2021.
- [56] Medium. Clasificación de imágenes con redes profundas. <https://medium.com/@a01639618/clasificacion-de-imagenes-con-redes-profundas-d2181f790dda>, December 2022. [En línea; accedido: 18-abr-2026].

- [57] Shutterstock. Imágenes de manzanas. <https://www.shutterstock.com/es/search/red-yellow-green-apples>, 2025. [En línea; accedido: 18-abr-2026].

Apéndice A

Código del clasificador de manzana

Listing A.1. Código principal

```
1 # detector_manzanas_pytorch_with_l298n.py
2 import cv2
3 import numpy as np
4 import torch
5 import torch.nn as nn
6 from torchvision import transforms
7 import time
8 import os
9 import RPi.GPIO as GPIO
10 from threading import Thread, Event
11 from RPLCD.i2c import CharLCD
12
13 print("■■ Sistema de Detección con Control START/STOP + L298N")
14 print("=" * 70)
15
16 # Configuración
17 IMG_SIZE = 224
18 CLASSES = ['Manzana Buena', 'Manzana Mala/Podrida', 'Objeto Desconocido']
19
20 CONFIDENCE_THRESHOLD = 0.6
```



```

21 # Configuraci3n GPIO (BCM)
22 SERVO_PIN = 17      # GPIO17 (Pin 11)
23 BUZZER_PIN = 27      # GPIO27 (Pin 13)
24 BTN_START = 22       # GPIO22 (Pin 15) - Bot3n START
25 BTN_STOP = 23        # GPIO23 (Pin 16) - Bot3n STOP
26
27 # LEDs de estado
28 LED_DETENIDO = 24    # GPIO24 (Pin 18) - Sistema detenido
29 LED_ACTIVO = 25      # GPIO25 (Pin 22) -Sistema activo
30
31 # --- Pines para L298N (motor) ---
32 # Elegidos para no solapar con los pines ya usados.
33 MOTOR_EN = 18        # GPIO18 (Pin 12) - PWM enable (ENA) del L298N
34 MOTOR_IN1 = 5         # GPIO5  (Pin 29) - IN1 del L298N
35 MOTOR_IN2 = 6         # GPIO6  (Pin 31) - IN2 del L298N
36
37 # Configuraci3n LCD I2C
38 I2C_ADDRESS = 0x27
39
40 # Configuraci3n del sistema
41 CONTADOR_LIMITE = 15
42
43 # Contadores
44 contador_manzanas_buenas = 0
45 contador_manzanas_malas = 0
46
47 # Control del sistema
48 sistema_activo = Event() # Flag para controlar si el sistema est3a
    activo
49 sistema_activo.clear()   # Inicialmente detenido
50
51 # Configurar GPIO
52 GPIO.setmode(GPIO.BCM)
53 GPIO.setwarnings(False)
54

```

```

55 # Configurar Servo
56 GPIO.setup(SERVO_PIN, GPIO.OUT)
57 servo_pwm = GPIO.PWM(SERVO_PIN, 50)
58 servo_pwm.start(0)
59
60 # Configurar Buzzer
61 GPIO.setup(BUZZER_PIN, GPIO.OUT)
62 GPIO.output(BUZZER_PIN, GPIO.LOW)
63
64 # Configurar Botones con PULL-UP (presionado = LOW)
65 GPIO.setup(BTN_START, GPIO.IN, pull_up_down=GPIO.PUD_UP)
66 GPIO.setup(BTN_STOP, GPIO.IN, pull_up_down=GPIO.PUD_UP)
67
68 # Configurar pines del L298N
69 GPIO.setup(MOTOR_EN, GPIO.OUT) # PWM
70 GPIO.setup(MOTOR_IN1, GPIO.OUT)
71 GPIO.setup(MOTOR_IN2, GPIO.OUT)
72
73 # PWM para EN (control de velocidad). Frecuencia 100Hz.
74 motor_pwm = GPIO.PWM(MOTOR_EN, 1000)
75 # Inicialmente motor detenido
76 motor_pwm.start(0)
77 GPIO.output(MOTOR_IN1, GPIO.LOW)
78 GPIO.output(MOTOR_IN2, GPIO.LOW)
79
80 # Configurar LEDs
81 GPIO.setup(LED_DETENIDO, GPIO.OUT)
82 GPIO.setup(LED_ACTIVO, GPIO.OUT)
83
84 # Estado inicial: sistema detenido
85 GPIO.output(LED_DETENIDO, GPIO.HIGH)
86 GPIO.output(LED_ACTIVO, GPIO.LOW)
87
88 print(f"■ Botones configurados:")
89 print(f"    START: GPIO{BTN_START} (Pin físico 15)")

```

```

90 print(f"    STOP:  GPIO{BTN_STOP} (Pin físico 16)")
91 print(f"■ḡ Motor L298N pines (BCM): EN={MOTOR_EN}, IN1={MOTOR_IN1}, IN2
    =={MOTOR_IN2}")
92
93 # Inicializar LCD
94 try:
95     lcd = CharLCD(
96         i2c_expander='PCF8574',
97         address=I2C_ADDRESS,
98         port=1,
99         cols=16,
100        rows=2,
101        dotsize=8,
102        charmap='A00',
103        auto_linebreaks=True
104    )
105    lcd.clear()
106    lcd.write_string("Sistema Listo")
107    lcd.cursor_pos = (1, 0)
108    lcd.write_string("Presiona START")
109    print(f"â LCD inicializado en 0x{I2C_ADDRESS:02x}")
110    lcd_disponible = True
111 except Exception as e:
112     print(f"â ïŸŔ LCD no disponible: {e}")
113     lcd_disponible = False
114
115 # Colores BGR
116 COLORS = {
117     0: (0, 255, 0),
118     1: (0, 165, 255),
119     2: (0, 0, 255)
120 }
121
122 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
123 print(f"■ś Dispositivo: {device}")

```

```

124
125 class ModeloManzanas(nn.Module):
126     """Red neuronal convolucional"""
127
128     def __init__(self, num_classes=3):
129         super(ModeloManzanas, self).__init__()
130
131         self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1)
132         self.bn1 = nn.BatchNorm2d(32)
133         self.relu1 = nn.ReLU()
134         self.pool1 = nn.MaxPool2d(2, 2)
135
136         self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
137         self.bn2 = nn.BatchNorm2d(64)
138         self.relu2 = nn.ReLU()
139         self.pool2 = nn.MaxPool2d(2, 2)
140
141         self.conv3 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
142         self.bn3 = nn.BatchNorm2d(128)
143         self.relu3 = nn.ReLU()
144         self.pool3 = nn.MaxPool2d(2, 2)
145
146         self.conv4 = nn.Conv2d(128, 128, kernel_size=3, padding=1)
147         self.bn4 = nn.BatchNorm2d(128)
148         self.relu4 = nn.ReLU()
149         self.pool4 = nn.MaxPool2d(2, 2)
150
151         self.flatten = nn.Flatten()
152         self.dropout1 = nn.Dropout(0.5)
153         self.fc1 = nn.Linear(128 * 14 * 14, 256)
154         self.relu5 = nn.ReLU()
155         self.dropout2 = nn.Dropout(0.3)
156         self.fc2 = nn.Linear(256, num_classes)
157
158     def forward(self, x):

```

```

159         x = self.pool1(self.relu1(self.bn1(self.conv1(x))))
160         x = self.pool2(self.relu2(self.bn2(self.conv2(x))))
161         x = self.pool3(self.relu3(self.bn3(self.conv3(x))))
162         x = self.pool4(self.relu4(self.bn4(self.conv4(x))))
163
164         x = self.flatten(x)
165         x = self.dropout1(x)
166         x = self.relu5(self.fc1(x))
167         x = self.dropout2(x)
168         x = self.fc2(x)
169
170         return x
171
172 def start_motor(duty=100):
173     """Arranca el motor: direcci3n FORWARD y PWM al duty%"""
174     # Direcci3n FORWARD: IN1=HIGH, IN2=LOW (ajusta si necesitas
175         invertir)
176     GPIO.output(MOTOR_IN1, GPIO.HIGH)
177     GPIO.output(MOTOR_IN2, GPIO.LOW)
178     motor_pwm.ChangeDutyCycle(duty)
179     print("3n Motor arrancado (duty={duty}%)")
180
181 def stop_motor():
182     """Detiene el motor: PWM 0 y entradas en LOW"""
183     motor_pwm.ChangeDutyCycle(0)
184     GPIO.output(MOTOR_IN1, GPIO.LOW)
185     GPIO.output(MOTOR_IN2, GPIO.LOW)
186     print("3n Motor detenido")
187
188 def monitorear_botones():
189     """Monitorea los botones START/STOP en un hilo separado"""
190     ultimo_estado_start = GPIO.HIGH
191     ultimo_estado_stop = GPIO.HIGH
192
193     print("\n■ Monitor de botones iniciado")

```

```

193     print("    Esperando presiÃ³n de START...")
194
195     while True:
196         # Leer estado actual de botones
197         estado_start = GPIO.input(BTN_START)
198         estado_stop = GPIO.input(BTN_STOP)
199
200         # Detectar presiÃ³n de START (flanco descendente HIGH a LOW)
201         if estado_start == GPIO.LOW and ultimo_estado_start == GPIO.HIGH
202             :
203                 time.sleep(0.05) # Debounce
204                 if GPIO.input(BTN_START) == GPIO.LOW: # Confirmar presiÃ³n
205                     if not sistema_activo.is_set():
206                         print("\n■ BOTÃN START PRESIONADO")
207                         print("    ■ Iniciando sistema de detecciÃ³n..."
208                             )
209                         sistema_activo.set()
210
211                     if lcd_disponible:
212                         lcd.clear()
213                         lcd.write_string("SISTEMA ACTIVO")
214                         lcd.cursor_pos = (1, 0)
215                         lcd.write_string("Detectando...")
216                         time.sleep(1)
217
218                     sonar_buzzer(veces=2, duracion=0.1)
219                     # Arrancar motor al iniciar el sistema
220                     start_motor(duty=100)
221                     # LEDs de estado
222                     GPIO.output(LED_DETENIDO, GPIO.LOW)
223                     GPIO.output(LED_ACTIVO, GPIO.HIGH)
224
225         # Detectar presiÃ³n de STOP (flanco descendente HIGH a LOW)
226         if estado_stop == GPIO.LOW and ultimo_estado_stop == GPIO.HIGH:
227             time.sleep(0.05) # Debounce

```

```

226         if GPIO.input(BTN_STOP) == GPIO.LOW: # Confirmar presiÃ³n
227             if sistema_activo.is_set():
228                 print("\n\t' BOTÃN STOP PRESIONADO")
229                 print("      ââââ Deteniendo sistema...")
230                 sistema_activo.clear()
231
232             if lcd_disponible:
233                 lcd.clear()
234                 lcd.write_string("SISTEMA DETENIDO")
235                 lcd.cursor_pos = (1, 0)
236                 lcd.write_string("Presiona START")
237
238             # Detener motor y volver servo a posiciÃ³n inicial
239             stop_motor()
240             mover_servo(0)
241
242             sonar_buzzer(veces=1, duracion=0.3)
243
244             # LEDs de estado
245             GPIO.output(LED_ACTIVO, GPIO.LOW)
246             GPIO.output(LED_DETENIDO, GPIO.HIGH)
247
248             # Guardar estados
249             ultimo_estado_start = estado_start
250             ultimo_estado_stop = estado_stop
251
252             time.sleep(0.01) # 10ms de polling
253
254 def actualizar_lcd(buenas, malas):
255     """Actualiza el display LCD con los contadores"""
256     if not lcd_disponible or not sistema_activo.is_set():
257         return
258
259     try:
260         lcd.clear()

```

```

261         lcd.write_string(f"Buenas:      {buenas:03d}")
262         lcd.cursor_pos = (1, 0)
263         lcd.write_string(f"Malas:      {malas:03d}")
264     except Exception as e:
265         print(f"⚠ Error LCD: {e}")
266
267 def mover_servo(angulo):
268     """Mueve el servomotor al ángulo especificado"""
269     print(f"Moviendo SERVO a : {angulo}")
270     duty_cycle = 2 + (angulo / 18)
271     servo_pwm.ChangeDutyCycle(duty_cycle)
272     time.sleep(0.3)
273     servo_pwm.ChangeDutyCycle(0)
274
275 def sonar_buzzer(veces=3, duracion=0.2):
276     """Hace sonar el buzzer"""
277     def _sonar():
278         for _ in range(veces):
279             GPIO.output(BUZZER_PIN, GPIO.HIGH)
280             time.sleep(duracion)
281             GPIO.output(BUZZER_PIN, GPIO.LOW)
282             time.sleep(duracion)
283
284     thread = Thread(target=_sonar)
285     thread.daemon = True
286     thread.start()
287
288 def procesar_deteccion(clase):
289     """Procesa la detección y controla hardware"""
290     global contador_manzanas_buenas, contador_manzanas_malas
291
292     if not sistema_activo.is_set():
293         return # No procesar si el sistema está detenido
294
295     if clase == 0: # Manzana Buena

```



```

296     print(f"â Manzana buena - Contador: {contador_manzanas_buenas +
297           1}/{CONTADOR_LIMITE}")
298
299     mover_servo(0)
300
301     contador_manzanas_buenas += 1
302     actualizar_lcd(contador_manzanas_buenas, contador_manzanas_malas
303                   )
304
305     if contador_manzanas_buenas >= CONTADOR_LIMITE:
306         print(f"■ ÂqLÃ■MITE ALCANZADO! {CONTADOR_LIMITE} manzanas")
307
308         if lcd_disponible:
309             lcd.clear()
310             lcd.write_string("LIMITE ALCANZADO")
311             lcd.cursor_pos = (1, 0)
312             lcd.write_string(f"{CONTADOR_LIMITE} manzanas!")
313             time.sleep(2)
314
315             sonar_buzzer(vec=5, duracion=0.15)
316             contador_manzanas_buenas = 0
317             actualizar_lcd(contador_manzanas_buenas,
318                           contador_manzanas_malas)
319
320     else: # Manzana Mala o Desconocido
321         if clase == 1:
322             print(f"â■ Manzana mala - Total: {contador_manzanas_malas +
323                 1}")
324         else:
325             print(f"â ïÿŔ Desconocido - Total: {contador_manzanas_malas
326                 + 1}")
327
328     mover_servo(90)
329     contador_manzanas_malas += 1
330     actualizar_lcd(contador_manzanas_buenas, contador_manzanas_malas
331                   )

```

```

325
326 def cargar_modelo():
327     """Carga el modelo entrenado"""
328     modelo_mejor = 'mejor_modelo_manzanas.pth'
329     modelo_final = 'modelo_manzanas_pytorch.pth'
330
331     if os.path.exists(modelo_mejor):
332         print(f"â Cargando: {modelo_mejor}")
333         model = ModeloManzanas(num_classes=len(CLASSES))
334         model.load_state_dict(torch.load(modelo_mejor, map_location=
            device))
335     elif os.path.exists(modelo_final):
336         print(f"â Cargando: {modelo_final}")
337         checkpoint = torch.load(modelo_final, map_location=device)
338         model = ModeloManzanas(num_classes=len(CLASSES))
339         model.load_state_dict(checkpoint['model_state_dict'])
340     else:
341         raise FileNotFoundError("â❏ Modelo no encontrado")
342
343     model.to(device)
344     model.eval()
345     return model
346
347 def preprocesar_frame(frame):
348     """Preprocesa el frame"""
349     img = cv2.resize(frame, (IMG_SIZE, IMG_SIZE))
350     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
351
352     transform = transforms.Compose([
353         transforms.ToTensor(),
354         transforms.Normalize(mean=[0.485, 0.456, 0.406],
355                                std=[0.229, 0.224, 0.225])
356     ])
357
358     img_tensor = transform(img)

```

```

359     return img_tensor.unsqueeze(0).to(device)
360
361 def predecir(model, frame):
362     """Realiza la predicci3n"""
363     with torch.no_grad():
364         img_tensor = preprocesar_frame(frame)
365         outputs = model(img_tensor)
366         probabilities = torch.nn.functional.softmax(outputs, dim=1)
367         return probabilities.cpu().numpy()[0]
368
369 def dibujar_resultado(frame, clase, confianza, activo):
370     """Dibuja el resultado en el frame"""
371     height, width = frame.shape[:2]
372
373     # Estado del sistema
374     if activo:
375         color_estado_sistema = (0, 255, 0)
376         texto_estado = "SISTEMA: ACTIVO"
377     else:
378         color_estado_sistema = (0, 0, 255)
379         texto_estado = "SISTEMA: DETENIDO"
380
381     # Rect3ngulo de estado
382     cv2.rectangle(frame, (10, 10), (width - 10, 60), (0, 0, 0), -1)
383     cv2.rectangle(frame, (10, 10), (width - 10, 60),
384                   color_estado_sistema, 3)
385     cv2.putText(frame, texto_estado, (20, 40),
386                 cv2.FONT_HERSHEY_SIMPLEX, 1.0, color_estado_sistema, 3)
387
388     if activo:
389         # Informaci3n de detecci3n
390         color = COLORS[clase]
391         nombre_clase = CLASES[clase]
392
393         overlay = frame.copy()

```

```

393         cv2.rectangle(overlay, (10, 80), (width - 10, 220), (0, 0, 0),
394                       -1)
395         cv2.addWeighted(overlay, 0.7, frame, 0.3, 0, frame)
396         cv2.rectangle(frame, (10, 80), (width - 10, 220), color, 4)
397
398         cv2.putText(frame, nombre_clase, (20, 120),
399                    cv2.FONT_HERSHEY_SIMPLEX, 1.0, color, 3)
400         cv2.putText(frame, f"Confianza: {confianza*100:.1f}%", (20, 155)
401                    ,
402                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
403
404         # Contadores
405         cv2.putText(frame, f"Buenas: {contador_manzanas_buenas:03d}/{
406                       CONTADOR_LIMITE}",
407                    (20, 190), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255,
408                       0), 2)
409         cv2.putText(frame, f"Malas: {contador_manzanas_malas:03d}",
410                    (20, 215), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 165,
411                       255), 2)
412
413     else:
414         # Mensaje de espera
415         cv2.putText(frame, "Presiona START para iniciar", (20, 120),
416                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2)
417         cv2.putText(frame, "o Q para salir", (20, 155),
418                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (200, 200, 200), 2)
419
420     return frame
421
422 def main():
423     global contador_manzanas_buenas, contador_manzanas_malas
424
425     # Cargar modelo
426     try:
427         model = cargar_modelo()
428     except Exception as e:

```

```

423         print(f"â Error: {e}")
424         GPIO.cleanup()
425         return
426
427     # Iniciar monitor de botones en hilo separado
428     thread_botones = Thread(target=monitorear_botones, daemon=True)
429     thread_botones.start()
430
431     # Servo a posiciÃn 0
432     mover_servo(0)
433     time.sleep(1)
434
435     # Detener motor inicialmente (asegurar)
436     stop_motor()
437     time.sleep(0.1)
438
439     # Destruir ventanas previas
440     cv2.destroyAllWindows()
441     time.sleep(0.1)
442
443     # CÃmara
444     print("\nâ Inicializando cÃmara...")
445     cap = cv2.VideoCapture(0)
446
447     if not cap.isOpened():
448         print("â Error: CÃmara no disponible")
449         GPIO.cleanup()
450         return
451
452     cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
453     cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
454
455     # Nombre Ãnico de ventana (usar como constante)
456     WINDOW_NAME = 'Sistema de DetecciÃn - START/STOP'
457

```

```

458 # Crear ventana una sola vez
459 cv2.namedWindow(WINDOW_NAME, cv2.WINDOW_NORMAL)
460
461 print("¡ Cámara lista")
462 print("\n" + "=" * 70)
463 print("■ SISTEMA LISTO")
464 print("=" * 70)
465 print("Controles:")
466 print(" [BOTÓN START] - Iniciar detección (arranca motor)")
467 print(" [BOTÓN STOP] - Detener detección (detiene motor)")
468 print(" [Q] - Salir del programa")
469 print(" [R] - Reiniciar contadores")
470 print("=" * 70 + "\n")
471
472 # Variables
473 fps_counter = 0
474 fps_start_time = time.time()
475 fps_display = 0
476
477 clase_anterior = None
478 tiempo_ultima_deteccion = 0
479 DEBOUNCE_TIME = 1.0
480
481 try:
482     while True:
483         ret, frame = cap.read()
484         if not ret:
485             break
486
487         activo = sistema_activo.is_set()
488
489         if activo:
490             # Realizar predicción solo si el sistema está activo
491             probabilities = predecir(model, frame)
492             clase_predicha = np.argmax(probabilities)

```

```

493         confianza = probabilities[clase_predicha]
494
495         if confianza < CONFIDENCE_THRESHOLD:
496             clase_predicha = 2
497             confianza = 1.0 - np.max(probabilities)
498
499         # Procesar detección con debounce
500         tiempo_actual = time.time()
501         if (clase_predicha != clase_anterior and
502             tiempo_actual - tiempo_ultima_deteccion >
503             DEBOUNCE_TIME):
504             procesar_deteccion(clase_predicha)
505             clase_anterior = clase_predicha
506             tiempo_ultima_deteccion = tiempo_actual
507         else:
508             clase_predicha = 0
509             confianza = 0
510
511         # Dibujar resultado
512         frame = dibujar_resultado(frame, clase_predicha, confianza,
513                                   activo)
514
515         # FPS
516         fps_counter += 1
517         if time.time() - fps_start_time > 1:
518             fps_display = fps_counter
519             fps_counter = 0
520             fps_start_time = time.time()
521
522         cv2.putText(frame, f"FPS: {fps_display}", (frame.shape[1] -
523             150, 30),
524                     cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255),
525                     2)
526
527         cv2.imshow(WINDOW_NAME, frame)

```

```

524
525         key = cv2.waitKey(1) & 0xFF
526
527         if key == ord('q'):
528             print("\n■ Saliendo...")
529             break
530         elif key == ord('r') and activo:
531             print("■ Reiniciando contadores")
532             contador_manzanas_buenas = 0
533             contador_manzanas_malas = 0
534             actualizar_lcd(0, 0)
535
536     except KeyboardInterrupt:
537         print("\nâ ĩÿŘ InterrupciÃşn")
538
539     finally:
540         print("\n■ğ Limpiando...")
541
542         # Mostrar mensaje en LCD y cerrar
543         if lcd_disponible:
544             try:
545                 lcd.clear()
546                 lcd.write_string("Sistema apagado")
547                 time.sleep(1)
548                 lcd.clear()
549                 lcd.close(clear=True)
550             except:
551                 pass
552
553         # Detener motor, servo y limpiar GPIO
554         stop_motor()
555         mover_servo(0)
556         time.sleep(0.5)
557         servo_pwm.stop()
558         motor_pwm.stop()

```



```
559     GPIO.output(BUZZER_PIN, GPIO.LOW)
560     GPIO.cleanup()
561     cap.release()
562     cv2.destroyAllWindows()
563
564     print("â Sistema cerrado")
565     print(f"■ Total: {contador_manzanas_buenas} buenas, {
        contador_manzanas_malas} malas")
566
567 if __name__ == "__main__":
568     main()
```

Apéndice B

Código del entrenamiento del clasificador de manzana

Listing B.1. Código principal

```
1 import os
2 import numpy as np
3 import cv2
4 import torch
5 import torch.nn as nn
6 import torch.optim as optim
7 from torch.utils.data import Dataset, DataLoader
8 from torchvision import transforms
9 from sklearn.model_selection import train_test_split
10 import time
11
12 print("■ Entrenamiento con PyTorch - Sistema de Clasificación de
13     Manzanas")
14
15 print("=" * 70)
16
17 # Configuración
18 IMG_SIZE = 224
19 BATCH_SIZE = 8 # Más pequeño para Raspberry Pi
20 EPOCHS = 25
```

```

19 CLASES = ['manzana_buena', 'manzana_mala', 'no_manzana']
20 LEARNING_RATE = 0.001
21
22 # Verificar si hay GPU disponible (probablemente no en Raspberry Pi)
23 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
24 print(f"■ Dispositivo: {device}")
25
26 class ManzanasDataset(Dataset):
27     """Dataset personalizado para manzanas"""
28
29     def __init__(self, images, labels, transform=None):
30         self.images = images
31         self.labels = labels
32         self.transform = transform
33
34     def __len__(self):
35         return len(self.images)
36
37     def __getitem__(self, idx):
38         image = self.images[idx]
39         label = self.labels[idx]
40
41         if self.transform:
42             image = self.transform(image)
43
44         return image, label
45
46 class ModeloManzanas(nn.Module):
47     """Red neuronal convolucional optimizada para Raspberry Pi"""
48
49     def __init__(self, num_classes=3):
50         super(ModeloManzanas, self).__init__()
51
52         # Bloque 1
53         self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1)

```

```

54     self.bn1 = nn.BatchNorm2d(32)
55     self.relu1 = nn.ReLU()
56     self.pool1 = nn.MaxPool2d(2, 2)
57
58     # Bloque 2
59     self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
60     self.bn2 = nn.BatchNorm2d(64)
61     self.relu2 = nn.ReLU()
62     self.pool2 = nn.MaxPool2d(2, 2)
63
64     # Bloque 3
65     self.conv3 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
66     self.bn3 = nn.BatchNorm2d(128)
67     self.relu3 = nn.ReLU()
68     self.pool3 = nn.MaxPool2d(2, 2)
69
70     # Bloque 4
71     self.conv4 = nn.Conv2d(128, 128, kernel_size=3, padding=1)
72     self.bn4 = nn.BatchNorm2d(128)
73     self.relu4 = nn.ReLU()
74     self.pool4 = nn.MaxPool2d(2, 2)
75
76     # Capas densas
77     self.flatten = nn.Flatten()
78     self.dropout1 = nn.Dropout(0.5)
79     self.fc1 = nn.Linear(128 * 14 * 14, 256)
80     self.relu5 = nn.ReLU()
81     self.dropout2 = nn.Dropout(0.3)
82     self.fc2 = nn.Linear(256, num_classes)
83
84     def forward(self, x):
85         # Bloques convolucionales
86         x = self.pool1(self.relu1(self.bn1(self.conv1(x))))
87         x = self.pool2(self.relu2(self.bn2(self.conv2(x))))
88         x = self.pool3(self.relu3(self.bn3(self.conv3(x))))

```

```

89         x = self.pool4(self.relu4(self.bn4(self.conv4(x))))
90
91         # Capas densas
92         x = self.flatten(x)
93         x = self.dropout1(x)
94         x = self.relu5(self.fc1(x))
95         x = self.dropout2(x)
96         x = self.fc2(x)
97
98         return x
99
100 def cargar_imagenes_desde_carpetas(data_dir):
101     """Carga imágenes desde las carpetas organizadas"""
102     images = []
103     labels = []
104
105     for idx, class_name in enumerate(CLASSES):
106         class_path = os.path.join(data_dir, class_name)
107
108         if not os.path.exists(class_path):
109             print(f"⚠ Advertencia: No existe la carpeta {class_path}")
110             continue
111
112         print(f"■ Procesando clase: {class_name}")
113         count = 0
114
115         for img_name in os.listdir(class_path):
116             if not img_name.lower().endswith(('.jpg', '.jpeg', '.png')):
117                 continue
118
119             img_path = os.path.join(class_path, img_name)
120
121             # Leer imagen
122             img = cv2.imread(img_path)

```

```

123         if img is None:
124             continue
125
126         # Redimensionar
127         img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
128
129         # Convertir BGR a RGB
130         img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
131
132         images.append(img)
133         labels.append(idx)
134         count += 1
135
136         print(f"      â Cargadas {count} imágenes")
137
138     return np.array(images), np.array(labels)
139
140 def entrenar_modelo(model, train_loader, val_loader, criterion,
141                     optimizer, epochs):
142     """Entrena el modelo"""
143
144     best_val_acc = 0.0
145     history = {'train_loss': [], 'train_acc': [], 'val_loss': [], '
146               val_acc': []}
147
148     for epoch in range(epochs):
149         print(f"\n{'='*70}")
150         print(f"Época {epoch+1}/{epochs}")
151         print(f"{'='*70}")
152
153         # ENTRENAMIENTO
154         model.train()
155         train_loss = 0.0
156         train_correct = 0
157         train_total = 0

```

```

156
157     start_time = time.time()
158
159     for batch_idx, (images, labels) in enumerate(train_loader):
160         images, labels = images.to(device), labels.to(device)
161
162         # Forward
163         optimizer.zero_grad()
164         outputs = model(images)
165         loss = criterion(outputs, labels)
166
167         # Backward
168         loss.backward()
169         optimizer.step()
170
171         # Estadísticas
172         train_loss += loss.item()
173         _, predicted = outputs.max(1)
174         train_total += labels.size(0)
175         train_correct += predicted.eq(labels).sum().item()
176
177         # Mostrar progreso
178         if (batch_idx + 1) % 5 == 0:
179             print(f"    Batch {batch_idx+1}/{len(train_loader)} - "
180                   f"Loss: {loss.item():.4f} - "
181                   f"Acc: {100.*train_correct/train_total:.2f}%")
182
183     train_loss /= len(train_loader)
184     train_acc = 100. * train_correct / train_total
185
186     # VALIDACIÓN
187     model.eval()
188     val_loss = 0.0
189     val_correct = 0
190     val_total = 0

```

```

191
192     with torch.no_grad():
193         for images, labels in val_loader:
194             images, labels = images.to(device), labels.to(device)
195
196             outputs = model(images)
197             loss = criterion(outputs, labels)
198
199             val_loss += loss.item()
200             _, predicted = outputs.max(1)
201             val_total += labels.size(0)
202             val_correct += predicted.eq(labels).sum().item()
203
204     val_loss /= len(val_loader)
205     val_acc = 100. * val_correct / val_total
206
207     # Guardar historial
208     history['train_loss'].append(train_loss)
209     history['train_acc'].append(train_acc)
210     history['val_loss'].append(val_loss)
211     history['val_acc'].append(val_acc)
212
213     epoch_time = time.time() - start_time
214
215     print(f"\n■ Resultados Época {epoch+1}:")
216     print(f"    Entrenamiento - Loss: {train_loss:.4f} | Acc: {train_acc:.2f}%")
217     print(f"    Validación - Loss: {val_loss:.4f} | Acc: {val_acc:.2f}%")
218     print(f"    Tiempo: {epoch_time:.1f}s")
219
220     # Guardar mejor modelo
221     if val_acc > best_val_acc:
222         best_val_acc = val_acc
223         torch.save(model.state_dict(), 'mejor_modelo_manzanas.pth')

```



```

224         print(f"      â Ĥ Mejor modelo guardado! (Val Acc: {val_acc:.2f
           }%)" )
225
226     return history
227
228 def main():
229     # Verificar carpeta de datos
230     if not os.path.exists('data'):
231         print("\nâ ERROR: No existe la carpeta 'data'")
232         print("\nCrea la siguiente estructura:")
233         print("data/")
234         print("    âââ manzana_buena/")
235         print("    âââ manzana_mala/")
236         print("    âââ no_manzana/")
237         print("\nY coloca al menos 50-100 imÃągenes en cada carpeta.")
238         return
239
240     # Cargar imÃągenes
241     print("\nâ Cargando imÃągenes del dataset...")
242     X, y = cargar_imagenes_desde_carpetas('data')
243
244     if len(X) == 0:
245         print("\nâ No se encontraron imÃągenes en las carpetas")
246         return
247
248     print(f"\nâ Total de imÃągenes: {len(X)}")
249     print(f"\nâ DistribuciÃąn por clase:")
250     for idx, class_name in enumerate(CLASSES):
251         count = np.sum(y == idx)
252         print(f"    {class_name}: {count} imÃągenes")
253
254     # Dividir datos
255     X_train, X_val, y_train, y_val = train_test_split(
256         X, y, test_size=0.2, random_state=42, stratify=y
257     )

```

```

258
259 print(f"\n■ Conjunto de entrenamiento: {len(X_train)} imágenes")
260 print(f"■ Conjunto de validación: {len(X_val)} imágenes")
261
262 # Transformaciones (data augmentation)
263 transform = transforms.Compose([
264     transforms.ToTensor(),
265     transforms.Normalize(mean=[0.485, 0.456, 0.406],
266                           std=[0.229, 0.224, 0.225])
267 ])
268
269 # Crear datasets
270 train_dataset = ManzanasDataset(X_train, y_train, transform=
    transform)
271 val_dataset = ManzanasDataset(X_val, y_val, transform=transform)
272
273 # Crear dataloaders
274 train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE,
275                           shuffle=True, num_workers=0)
276 val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE,
277                        shuffle=False, num_workers=0)
278
279 # Crear modelo
280 print("\n■ Creando modelo...")
281 model = ModeloManzanas(num_classes=len(CLASSES)).to(device)
282
283 # Contar parámetros
284 total_params = sum(p.numel() for p in model.parameters())
285 trainable_params = sum(p.numel() for p in model.parameters() if p.
    requires_grad)
286 print(f"    Total de parámetros: {total_params:,}")
287 print(f"    Parámetros entrenables: {trainable_params:,}")
288
289 # Definir pérdida y optimizador
290 criterion = nn.CrossEntropyLoss()

```

```

291     optimizer = optim.Adam(model.parameters(), lr=LEARNING_RATE)
292
293     # Entrenar
294     print("\n■ Iniciando entrenamiento...\n")
295     history = entrenar_modelo(model, train_loader, val_loader,
296                               criterion, optimizer, EPOCHS)
297
298     # Guardar modelo final
299     torch.save({
300         'model_state_dict': model.state_dict(),
301         'classes': CLASSES,
302         'img_size': IMG_SIZE
303     }, 'modelo_manzanas_pytorch.pth')
304
305     print("\n" + "="*70)
306     print("■ ENTRENAMIENTO COMPLETADO")
307     print("="*70)
308     print(f"â Modelo guardado como: modelo_manzanas_pytorch.pth")
309     print(f"â Ñ Mejor modelo guardado como: mejor_modelo_manzanas.pth")
310     print(f"■ Mejor precisiÃ³n de validaciÃ³n: {max(history['val_acc'])
311           :.2f}%")
312
313     print("\n■â Ahora ejecuta: python detector_manzanas_pytorch.py")
314
315 if __name__ == "__main__":
316     main()

```