UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN

SISTEMA INTELIGENTE DE DETECCIÓN DE ESTRÉS HÍDRICO PARA DETERMINAR LA CALIDAD DE CULTIVO DE LECHUGA A TRAVÉS DE VISIÓN ARTIFICIAL

TRABAJO DE GRADO PREVIO LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE COMUNICACIÓN

AUTOR: CARLA DAYANARA RIVERA VACA

DIRECTOR: MSc. FABIÁN GEOVANNY CUZME RODRÍGUEZ

ASESOR: MSc. EDGAR ALBERTO MAYA OLALLA

IBARRA-ECUADOR

2023

UNIVERSIDAD TECNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE.

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DEL CONTACTO			
CÉDULA DE	1003584255		
APELLIDOS Y	Rivera Vaca Carla Dayanara		
DIRECCIÓN	Natabuela, Vencedores y Flores Vásquez		
E-MAIL	cdriverav@utn.edu.ec		
TELÉFONO FIJO	062535450	**TELÉFONO MÓVIL**	0998222060

DATOS DE LA OBRA	
TÍTULO	"Sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de visión artificial"
AUTOR	Rivera Vaca Carla Dayanara
FECHA	12/06/2023
PROGRAMA	Pregrado
TÍTULO	Ingeniero en Electrónica y Redes de Comunicación
DIRECTOR	MSc. Fabián Geovanny Cuzme Rodríguez

2. CONSTANCIAS.

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 12 días del mes de junio de 2023

EL AUTOR

.....

Carla Dayanara Rivera Vaca

CI: 100358425-5

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN.

MAGISTER FABIÁN CUZME RODRÍGUEZ, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN CERTIFICA:

Que, el presente trabajo de Titulación "SISTEMA INTELIGENTE DE DETECCIÓN DE ESTRÉS HÍDRICO PARA DETERMINAR LA CALIDAD DE CULTIVO DE LECHUGA A TRAVÉS DE VISIÓN ARTIFICIAL" Ha sido desarrollado por la señorita Carla Dayanara Rivera Vaca bajo mi supervisión.

Es todo en cuanto puedo certificar en honor de la verdad.

MSc. Fabián Geovanny Cuzme Rodríguez

DIRECTOR

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DEDICATORIA

Este trabajo de titulación lo dedico con todo mi amor y cariño a mi madre Consuelo por su apoyo incondicional, sus consejos para hacer de mí una mejor persona, por no dejarme rendir ante las dificultades de la vida.

A mi padre Carlos que desde el cielo me cuida y me guía para seguir adelante con mis proyectos.

A mi hermano Kevin, por su compañía y palabras de aliento.

A mi familia en general por compartir buenos y malos momentos, por sus palabras de motivación.

Carla Dayanara Rivera Vaca

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AGRADECIMIENTO

Gracias a Dios, por ser mi guía y fortaleza para seguir adelante.

A mis padres, por su amor, paciencia, por su esfuerzo y dedicación me permitieron culminar mi carrera universitaria y me brindaron su apoyo para no decaer antes las adversidades.

A la Universidad Técnica del Norte, por la oportunidad de formarme como profesional. A los docentes de la carrera de Ingeniería en Electrónica y Redes de Comunicación por sus conocimientos impartidos. A mis compañeros y amigos, por sus consejos y momentos compartidos.

Agradezco de manera especial a mi tutor de tesis, MSc. Fabian Cuzme y a la asesoría del MSc. Edgar Maya por su tiempo, consejos y correcciones para poder culminar con mi trabajo de titulación.

Carla Dayanara Rivera Vaca

RESUMEN

En el trabajo de titulación se desarrolló un sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de Visión Artificial, como un método de apoyo hacia el sector agrícola, de forma que se pueda obtener un mayor control en el desarrollo de la producción, evitar pérdidas y determinar la calidad de los cultivos mediante su clasificación.

Se establecieron cuatro fases: preliminar, planificación, ejecución y resultados. En la fase preliminar se realizó una investigación científica para definir algoritmos adecuados para la identificación y clasificación de objetos, seleccionando así la arquitectura de red neuronal convolucional Inception V3. Para las fases de planificación y ejecución se aplicó la metodología en Espiral en sus cuatro etapas generales: planificación, análisis, ingeniería y evaluación. En la etapa de planificación se identificaron los requisitos que debe cumplir el sistema y se determinaron los recursos disponibles. En la etapa de análisis, se propuso alternativas, prototipos y diseños, mediante un esquema de adquisición de datos usando 2 cámaras de alta resolución ubicadas en áreas estratégicas, cuyas imágenes son almacenadas en la nube de Ezviz, para luego pasar por un proceso de detección de bordes y etiquetado mediante el software LabelImg. En la etapa de ingeniería se agregó funcionalidades al prototipo propuesto, se realizó el entrenamiento de la red neuronal en Google Colab y se integró el modelo entrenado en la placa embebida Nvidia Jetson Nano.

Finalmente, se realiza una evaluación del prototipo, mediante la utilización de sistema embebido junto con una pantalla se accede a una interfaz amigable con el usuario, de forma que sea capaz de obtener información sobre la clasificación de cada lote del cultivo de lechuga, catalogándola como apta y no apta. Además, se realizó un análisis de la precisión del sistema.

ABSTRACT

In the degree work, an intelligent system for detecting water stress was developed to determine the quality of lettuce crops through Artificial Vision, as a method of support to the agricultural sector, in order to obtain greater control in the development of production, avoid losses and determine the quality of crops through their classification.

Four phases were established: preliminary, planning, execution and results. In the preliminary phase, scientific research was carried out to define suitable algorithms for object identification and classification, thus selecting the Inception V3 convolutional neural network architecture. For the planning and execution phases, the Spiral methodology was applied in its four general stages: planning, analysis, engineering and evaluation. In the planning stage, the requirements to be met by the system were identified and the available resources were determined. In the analysis stage, alternatives, prototypes and designs were proposed, through a data acquisition scheme using 2 high resolution cameras located in strategic areas, whose images are stored in the Ezviz cloud, to then go through a process of edge detection and labeling using LabelImg software. In the engineering stage, functionalities were added to the proposed prototype, the neural network was trained in Google Colab and the trained model was integrated into the Nvidia Jetson Nano embedded board.

Finally, an evaluation of the prototype is carried out, using an embedded system together with a screen to access a user-friendly interface, so that it is able to obtain information on the classification of each batch of lettuce crop, classifying it as suitable and unsuitable. In addition, an analysis of the accuracy of the system was carried out.

ÍNDICE

INDICE DE FIGURAS

INDICE DE TABLAS

Capítulo I

ANTECEDENTES

En este capítulo se presentan los antecedentes, en el cual se proporciona una breve introducción acerca del proyecto de titulación. A continuación, se presenta el tema, la problemática, los objetivos tanto general como específicos planteados, el alcance y la justificación que fundamenta este proyecto.

1.1. Tema

Sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de Visión Artificial

1.2. Problema

El sector agrícola cumple un rol fundamental en el desarrollo de la economía dentro de la población nacional y mundial, según (Ministerio de Agricultura y Ganadería, MAG, 2019) este sector además de proporcionar alimentos es una fuente principal generadora de empleo llegando a proporcionar alrededor de 2.2 millones de plazas de trabajo además aportando con el 8% al Producto Interno Bruto (PIB). Ecuador es un país que gracias a su ubicación geográfica cuenta con suelos fértiles y condiciones climáticas optimas, características que hacen una zona apta para el desarrollo de diversos tipos de cultivos; permitiendo la producción de frutas, verduras y hortalizas, como es el caso de la lechuga, el cuarto vegetal con mayor producción en provincias como: Cotopaxi, Tungurahua y Carchi, siendo este producto de mayor consumo en la sierra ecuatoriana. (El Comercio, 2011)

Actualmente, el MAG en conjunto con los gobiernos autónomos descentralizados de las siguientes parroquiales de: Angochagua, San Antonio de Ibarra, San Luis y González Suárez de los cantones de Ibarra y Otavalo respectivamente, ejecutan la

campaña “Produce tus alimentos sanos y seguros”, impulsando así la agricultura familiar campesina. Los beneficiarios según el MAG informan que reciben diversos tipos de semillas como remolacha, zanahoria, rábano y lechuga, entre otros, para que sean cultivados en las comunidades o dentro de su propiedad, aprovechando así los recursos existentes, de esta forma obtener variedad de hortalizas sanas, que aporten un beneficio a la calidad de vida de las personas gracias a una dieta equilibrada y balanceada. (Ministerio de Agricultura y Ganadería, 2020)

Con el aumento de la población incrementa la demanda de este tipo de productos, esto hace que la producción no abastezca la cantidad de productos con los parámetros de calidad esperados, debido a condiciones climáticas o al estrés que pueden sufrir las plantas por distintas causas, ya sea por la presencia de organismos vivos como virus y bacterias o por factores ambientales como el calor, humedad y la sequía (Valladares, y otros, 2004); Esto afecta la adecuada absorción de nutrientes debido a que la planta retiene menor cantidad de agua, de la misma manera el proceso de fotosíntesis se ve interrumpido, deja de producir energía y el metabolismo de la misma se ve afectado, limitando así su crecimiento y desarrollo, observando una despigmentación de las hojas; por ser la recolección de hortalizas un proceso manual depende de la visión humana, hecho que no genera alta confiabilidad debido a la presencia de detalles que muy difícilmente pueden ser detectados por el ojo humano, como consecuencia de esto se tienen cultivos de mala calidad o a la vez cosechados antes de tiempo. (Grupo Inesta, 2018)

El proyecto pretende determinar la calidad de los cultivos mediante la captura y procesamiento de imágenes para extraer características de la planta y la posterior clasificación, de forma que se pueda tener un mayor control en el desarrollo de la

producción, y efectuar recomendaciones y medidas correctivas del lote afectado para asegurar la obtención de un producto sano y de calidad.

1.3. Objetivos

1.3.1. Objetivo General

Diseñar un sistema inteligente de detección de estrés hídrico en el cultivo de lechuga mediante aprendizaje de máquina y visión artificial para determinar la calidad del cultivo.

1.3.2. Objetivos Específicos

- Realizar un análisis bibliográfico sobre técnicas de identificación de objetos en tiempo real y métodos de procesamiento de imagen para la extracción de características de muestras basadas en los defectos del objeto a estudiar.
- Diseñar un esquema para la adquisición de datos que permita encontrar patrones adecuados para la detección de imperfecciones en el cultivo de lechuga.
- Describir al menos dos escenarios que permitan probar el sistema en un ambiente de producción controlado para identificar posibles lotes afectados.
- Ejecutar pruebas de funcionamiento y ajustes pertinentes que requiera el sistema para una correcta clasificación en dos grupos: aptos y no aptos para el consumo humano.
- Analizar la implementación del sistema basados en los costos- beneficios considerando el impacto en la calidad del producto

1.4. Alcance

El presente trabajo de titulación pretende determinar la calidad de los cultivos mediante la captura y procesamiento de imágenes para extraer características de la planta y la posterior clasificación, de forma que se pueda tener un mayor control en el

desarrollo de la producción, y efectuar recomendaciones y medidas correctivas del lote afectado para asegurar la obtención de un producto sano y de calidad.

El desarrollo de este proyecto se basa en el cumplimiento de los objetivos, para lo cual se han establecido las siguientes fases: preliminar, planificación, ejecución y resultados. En la fase preliminar se determina la metodología y algoritmos adecuados para la identificación y clasificación de objetos en tiempo real, para ello se realiza una investigación haciendo uso de repositorios digitales que abarquen artículos e investigaciones científicas, para contar con posibles alternativas tomando en cuenta los riesgos.

Para las fases de planificación y ejecución se aplica una metodología de desarrollo llamada Modelo en Espiral, el cual se basa en el análisis y evaluación de riesgos buscando la continua mejora del software permitiendo realizar modificaciones a medida que se vaya realizando el proyecto hasta eliminar todos los riesgos. (Fariño R, 2011)

La fase de desarrollo y prueba de este modelo abarcan los objetivos 2,3 y 4, para lo cual se empezará realizando un esquema de adquisición de datos mediante el uso de 2 cámaras de alta resolución ubicadas en áreas estratégicas conectadas a un módulo que permita receptar, procesar imágenes adquiridas y alojar la red neuronal para así encontrar patrones adecuados para la detección de imágenes del objeto en cuestión. Para el entrenamiento se utilizará una gran cantidad de imágenes adquiridas de Internet y por autoría propia, las cuales pasarán por un proceso de filtrado y preprocesamiento previo a ser convertidas a un formato apto para la red neuronal.

Además, en esta etapa se realizará una interfaz amigable con el usuario que permita recibir información sobre la clasificación de cada lote del cultivo de lechuga.

La implementación se efectuará en un huerto de lechugas ubicado en el sector de Natabuela, se contará con dos escenarios que permitan probar el sistema de detección de estrés hídrico en un ambiente de producción, uno de los escenarios estará en óptimas condiciones, el mismo que tendrá un terreno al aire libre, preparado y listo para su producción, cada hortaliza dentro de este lote será sembrado con una distancia adecuada, regado periódicamente con el método de riego por goteo El segundo escenario tendrá un regado esporádico, condiciones no aptas para el desarrollo de la hortaliza. Para la adquisición de imágenes las cámaras serán ubicadas de forma fijas que permitan obtener una visión extensa del área, además se implementará pruebas de ensayo y error dentro del proyecto de forma que la programación de algoritmos sea un proceso rápido y eficaz.

La fase de verificación del sistema se ejecutarán pruebas de funcionamiento comparando la correcta clasificación del objeto tomando en cuenta características como color, tamaño y defectos de la lechuga. Realizando las pruebas necesarias para verificar posibles errores y continuar con su proceso de tal forma si existiera se busca nuevas soluciones o alternativas

Finalmente se realizará un análisis de la implementación del sistema basados en los costos- beneficios considerando el impacto en la calidad del producto.

1.5. Justificación

En el Ecuador, según (La Hora, 2021) la agricultura es uno de los principales sectores que aportan a la economía del país, tanto en la generación de empleo como al PIB, siendo posicionado en noveno lugar a nivel de América Latina.

La mayoría de la producción agrícola proviene de pequeños productores, los cuales en el sector rural se dedican al cultivo y crianza de animales especialmente para

el consumo doméstico, ocasionalmente los productos excedentes son vendidos en mercados locales, limitando así el poder adquisitivo a los consumidores externos. (Viteri Vera & Tapia Toral , 2018). Además, la producción de cultivos depende de los altos costos de materia prima como: semillas, fertilizantes, plaguicidas, maquinaria; condiciones climáticas, presencia de plagas, falta de capacitación y producción mediante técnicas tradicionales. (FAO, 2021)

Con el presente proyecto se pretende aportar y ayudar a continuar con el Plan Nacional del Buen Vivir 2017-2021 apoyándose específicamente en el segundo eje que dice la economía al servicio de la sociedad, apoyado del objetivo 5, el cual busca ampliar la productividad, generar empleo digno, además busca garantizar la soberanía alimentaria y el desarrollo rural integral. (Senplades, 2017)

La implementación de métodos tecnológicos e innovadores incentiva a dejar atrás técnicas tradicionales de cultivo, a través de la agricultura de precisión se acelera el proceso de producción, mejora la eficacia de la producción y garantizar la calidad del producto, aportando un beneficio al estilo de vida de las personas con productos sanos, sin fertilizantes, beneficiándonos así de todos los nutrientes de la tierra siendo estos: agua y minerales reduciendo de esta manera el impacto ambiental a largo plazo. (Hinojosa Pinto , 2019)

Conjuntamente, con la aplicación de otro tipo de tecnologías modernas como el uso de dispositivos electrónicos, sistemas de posicionamiento global y técnicas de visión artificial y el lenguaje de aprendizaje de máquina para la recolección, interpretación y análisis de datos es posible gestionar de manera eficiente la producción agrícola, facilitando el manejo de tierras y optimizando el uso de recursos. (InfoAgro, 2020)

Además, favorece tanto al impacto social como económico, especialmente al sector agrícola tanto al productor comercial como interno ya que le permite al agricultor evaluar la calidad del cultivo de lechuga durante el proceso de mantenimiento y cosecha, garantizando así la producción en su totalidad y que el producto cuente con los nutrientes adecuados aportando una mejor calidad de vida; con esto no solo se reducen costos de producción, maquinaria, personal, sino se asegura el buen uso de los recursos naturales y ayuda a combatir dificultades actuales como la malnutrición, aumento de población y el cambio climático. (Axpuac Yoc, 2020)

Capítulo II

FUNDAMENTACIÓN TEÓRICA

En el segundo capítulo se presenta una recopilación sobre temas que se aplican al desarrollo de este proyecto, se definen conceptos relacionados a la agricultura de precisión y sus diferentes campos de aplicación, se describe información referente al cultivo, parámetros que influyen en el desarrollo y producción de la siembra. Finalmente, se trata sobre conceptos de Visión Artificial y aprendizaje automático.

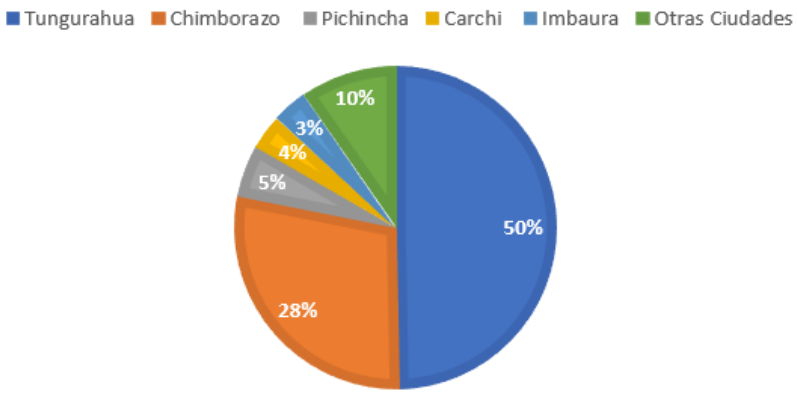
2.1. Producción nacional de lechuga

Según (La Hora, 2021), la agricultura es uno de los principales sectores que aportan a la economía del país, tanto en la generación de empleo como al PIB, posicionándose en noveno lugar a nivel de América Latina.

Una de las hortalizas con mayor demanda es la lechuga debido a su valor nutricional, aportando niveles de vitaminas y minerales aptos para el consumo humano. En el Ecuador, de acuerdo con los datos de (FAOSTAT, 2022) durante el año 2019 se alcanzó una cosecha cerca de 18.238 toneladas métricas cultivadas en un área de tres mil hectáreas siendo mayores productores las provincias de Tungurahua, Chimborazo, Pichincha, Carchi e Imbabura (figura 1). Además, el rendimiento promedio por hectárea en el país es de 7.2 toneladas métricas según la Encuesta de Superficie y Producción Agropecuaria Continua (ESPAC) realizada por el Instituto Nacional de Estadística y Censos (INEC, 2018).

Figura 1

Porcentaje de producción de lechuga por ciudades



Fuente: Autoría Propia

La mayoría de la producción de este vegetal proviene de pequeños productores, en donde un 83% de la producción nacional es para el consumo interno del país, especialmente para el consumo doméstico, ocasionalmente los productos excedentes son vendidos en mercados locales, limitando así el poder adquisitivo a los consumidores externos (Viteri Vera & Tapia Toral , 2018).

2.2. Cultivo de lechuga

La lechuga o por su nombre científico Lactuca sativa L, originaria del Mediterráneo es una planta herbácea cuyo tiempo de cultivo es de 3 a 4 meses para las variedades tempranas y de 70-80 días para las tardías, en general alcanza una altura de entre los 10 y 20 centímetros; las hojas son grandes y numerosas en forma de roseta, con una forma oval, oblongas, brillantes y opacas dependiendo la variedad de la lechuga. Tiene una superficie plana y rugosa, y suelen tener un color amarillento, verde claro, verde oscuro, rojizas y púrpuras (Saavedra Del R., 2017)

2.2.1. Variedades de lechuga

Según (Saavedra, y otros, 2017) se pueden encontrar muchas variedades de lechuga y debido a esto se ha optado por agruparlas ya que algunas comparten las mismas características, pero no los mismos nombres. Entre las variedades conocidas comercialmente se encuentran:

Romanas. - También llamadas Lactuca sativa var. longifolia. Tienen hojas alargadas y erectas, además no forman un cogollo y su nervio central es ancho. En este grupo tenemos: Romana y Baby.

Acogolladas. - Lactuca sativa var. capitata. Estas lechugas suelen formar un cogollo apretado con sus hojas, además sus hojas son más anchas. Tenemos: Batavia, Mantecosa o Trocadero e Iceberg.

De hojas sueltas. - También conocidas como Lactuca sativa var. inybacea. Estas lechugas poseen hojas sueltas y dispersas. Entre ellas están: Lollo Rossa, Red Salad Bowl y Cracarelle.

Lechuga espárrago. - Lactuca sativa var augustana. Aprovechan sus tallos para tener hojas puntiagudas y lanceoladas.

2.2.2. Valor Nutricional

Según (Almanza Vega, Zambrana, & Boris, 2016) la lechuga ofrece un gran aporte nutricional que beneficia a todos los consumidores. A continuación, se menciona el valor nutricional de la Lechuga.

Tabla 1

Valor nutricional de la lechuga

Valor nutricional de la lechuga en 100 g	
Agua (g)	95.5
Calorías (kcal)	13
Grasas (g)	0.2
Hidratos carbono (g)	2.3
Proteínas (g)	1.2
Fibra (g)	1
Potasio (mg)	257
Fósforo (mg)	23
Sodio (mg)	5
Calcio (mg)	32
Magnesio (mg)	13
Vitamina C (mg)	8
Vitamina A (UI)	970
Vitamina B6 (mg)	0.05
Ácido fólico	215

Fuente: (Editorial de Botanica-online, 2021)

2.2.3. Requerimientos del cultivo

Según (Cabrera Díaz, 2021) el cultivo de lechuga necesita cumplir con ciertas características en aspectos de clima, suelo y agua, para asegurar una producción de calidad. A continuación, se explican estos requerimientos edafoclimáticos:

Clima. - La temperatura óptima en el proceso de germinación debe ser entre 15-20°C. La lechuga es una planta con gran adaptabilidad a los distintos climas, aunque en climas no óptimos suele tener efectos, si sufre temperaturas bajas durante mucho tiempo, las hojas adoptan un color rojizo y la lechuga no emite nuevas raíces; mientras que en altas temperaturas acelera el proceso de floración y provoca que sus hojas adquieran un

sabor amargo. En la fase de crecimiento se necesitan temperaturas entre 14-18°C en el día y entre 5-8°C en la noche.

Suelo. - Los suelos en los que hay una mejor producción de lechuga son suelos ligeros con una gran cantidad de materia orgánica y que cuenten con un buen drenaje. Los cultivos de lechuga no toleran la acidez y tiene mejor adaptación a suelos ligeramente alcalinos. El pH que se considera óptimo es entre 6,7-7,4.

Agua. - La lechuga consta de un sistema radicular muy reducido en comparación a su parte aérea, debido a esto la lechuga es muy sensible a demasiada humedad haciendo que se pudra el tallo más rápido de lo habitual, y soporta muy mal los periodos de sequía con el resultado de que su tamaño es menor al habitual. La humedad óptima para la lechuga está entre el 60 y 80%.

2.3. Estrés en las plantas

El estrés es la pérdida de homeostasis, consiste en un equilibrio máximo entre organelos celulares y el potencial del trabajo que la célula puede tener. Toda circunstancia que afecte este equilibrio se considera una situación de estrés que puede ser recuperable, o totalmente irrecuperable y puede ocasionar la muerte del cultivo. (García & Jiménez, 2019)

2.3.1. Tipos

(García & Jiménez, 2019) considera que los tipos de estrés que más afecta al desarrollo de cultivos son:

Estrés medioambiental. - También conocido como estrés abiótico, abarca todos los factores físicos ambientales que puedan afectar de forma negativa el crecimiento y productividad de las plantas. Dentro de estos se puede observar la intensidad de la luz, o el déficit de los abonos inorgánicos que tienen un papel importante en pérdidas de cosechas.

Estrés mecánico. - Las plantas constan de una gran sensibilidad a cualquier cambio en el entorno, y con solo acceder a un invernadero se provoca estrés mecánico. Tan solo con el hecho de caminar por los campos o tocar seguido las plantas puede dar como resultado plantas más pequeñas o provocar lesiones en los tejidos, ocasionando diferentes enfermedades. Este tipo de estrés no se puede evitar en totalidad, pero hay que tener en cuenta en todo momento.

Estrés hídrico o producido por la sequía. - En días muy soleados, o bajo fuerte iluminación en un invernadero, las plantas llegan a marchitarse por la pérdida de agua debido a que la transpiración es mayor que la cantidad promedio de agua absorbida por las raíces del suelo. Esto provoca que las células protectoras se hinchen menos, un mecanismo que disminuye el proceso de transpiración por el cierre de estoma. La falta de agua también estimula la síntesis y liberación de ácido abscísico en las hojas, una hormona que ayuda a mantener las estomas cerradas mediante membranas celulares protectoras. La falta del agua reduce la fotosíntesis, y ésta es una de las razones por las que la sequía reduce el rendimiento de la producción agrícola.

Estrés por exceso de agua. - El exceso de agua acabará con la planta mucho antes que la falta de ésta. En los suelos abnegados no hay suficiente oxígeno para las plantas, sin el oxígeno empieza la respiración anaeróbica en las raíces dando como resultado la producción de compuestos tóxicos en la planta. Algunos síntomas de exceso de agua pueden ser el marchitamiento y amarillamiento en las hojas, las raíces se pudren y tienen un crecimiento irregular.

Estrés por salinidad. - El exceso de cloruro de sodio y otras sales en un sustrato representan amenazas para las plantas por dos razones. Uno es que se reduce el potencial hídrico del sustrato haciendo que la cantidad que absorbe la planta sea menor. Otro motivo es que el sodio y otros iones resultan tóxicos ya que si están en gran cantidad dificultan

la permeabilidad selectiva en las membranas celulares de las raíces, así la planta no será capaz de absorber los nutrientes selectivamente.

2.3.2. Efectos

Varios de los efectos que trae consigo el estrés en las plantas son la reducción del crecimiento debido a que tiene menos cantidad de agua, por lo que la pared celular se vuelve inflexible e impide el transporte activo del agua. Cierre de estomas en las hojas para evitar la pérdida de agua por el exceso de calor. La fotosíntesis se disminuye debido al cierre de las estomas, debido a esto la planta deja de generar energía, se pausa la producción de azúcares y como consecuencia se detiene el metabolismo de la planta (Asemafor, 2021).

2.4. Agricultura Inteligente

La agricultura de precisión permite gestionar de manera eficiente la producción agrícola, facilitando el manejo de tierras y optimizando el uso de recursos, mediante la aplicación de un conjunto de tecnologías modernas como el uso de dispositivos electrónicos y sistemas de posicionamiento global para la recolección, interpretación y análisis de datos (InfoAgro, 2020).

Aplica el uso de herramientas como drones con sistemas GPS para la obtención de datos geográficos de extensas longitudes de terreno, cámaras de alta resolución para la captura de imágenes, procesamiento y control de calidad, sensores para el monitoreo de distintos agentes esenciales que intervienen en el ciclo de cultivo con el fin de determinar factores limitantes que afectan al rendimiento del cultivo (García & Flego, 2015).

Consta de 4 etapas principales, las cuales se muestran en la figura 2, donde el ciclo inicia con la etapa de recolección de datos, haciendo uso de sensores y muestreo, para obtener datos como propiedades de los suelos, estado de salud de cultivos, detección de

plagas y enfermedades; como acto seguido se realiza la extracción de datos, información que será útil para el agricultor (GRAP, 2020).

Una vez reunidos los datos necesarios se procede a realizar diferentes modelos que se usarán en la etapa de análisis e interpretación de la información, para determinar qué operaciones se deben ejecutar.

Finalmente, con la información reunida en las etapas nombradas anteriormente, se procede a realizar una serie de operaciones como: la aplanada y adecuación del suelo, la siembra del producto, el abonado adecuado para el producto, y la protección del sembrío con los tratamientos necesarios como la aplicación de nutrientes y plaguicidas (GRAP, 2020).

Figura 2

Esquema de Agricultura Inteligente

Adaptado de: (GRAP, 2020)

2.4.1. Impacto del uso de tecnología en la producción de lechuga.

La producción agrícola desde sus inicios ha sido actividad realizada por los pequeños productores, ya sea para el consumo interno del grupo familiar o para abastecer con la demanda de alimentos a pequeñas poblaciones; esta producción depende de condiciones climáticas o de la calidad del terreno, del uso de estiércol como abono, de los conocimientos básicos acerca del suelo, las plantas y el producto que el agricultor posee, además del uso de herramientas manuales y animales de carga para arar o preparar la tierra (Sánchez Galán, 2021).

Con el incremento de la población a lo largo de los años la demanda de productos alimenticios ha ido en aumento, lo cual mediante técnicas tradicionales no es posible debido a su baja producción, por lo que la agricultura tradicional se ha ido adaptando a la implementación de ciencia y tecnología como el uso de sensores para monitorear factores influyentes en la producción tales como la humedad, la temperatura, luminosidad, etc., control de riego, detección de enfermedades y plagas mediante aprendizaje automático para mejorar la calidad del producto e incrementar la cantidad para satisfacer con la demanda del mercado en un tiempo menor de cosecha (Peláez, 2017).

De esta manera, la agricultura moderna permite que los procesos antes realizados de manera empírica por el agricultor ahora tengan una base acorde a las necesidades del cultivo, mediante la recolección y análisis de datos de factores influyentes en la producción. Además, se reduce la dependencia de factores climáticos, se cuida el crecimiento y desarrollo de la planta con una nutrición más adecuada con el uso de menos fertilizantes y control de enfermedades y plagas (Miranda, 2017).

2.5. Edge Computing

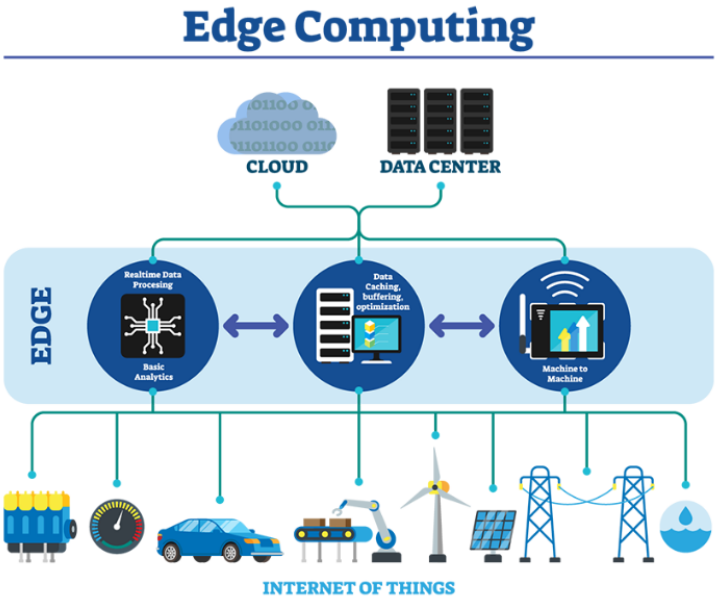
Actualmente la mayoría de los objetos están digitalizados, se crean dispositivos y se comparten datos en una gran escala, desde un celular hasta un juguete inteligente (Redacción España , 2020).

Muchas aplicaciones alojan sus datos obtenidos por dispositivos periféricos como sensores, actuadores o hasta teléfonos inteligentes en servidores en la nube, lo cual debido al incremento en la cantidad de datos requiere de una infraestructura cada vez más compleja, con mayor exigencia en cuanto a características como ancho de banda y latencia teniendo como resultado una comunicación lenta entre usuarios finales y la nube afectando de esta forma a la calidad del servicio y experiencia del usuario (Taleb, Di Francesco, & Premsankar, 2018).

El Edge Computing se encarga de procesar y almacenar información por medio de sensores o dispositivos loT de manera local sin comunicarse con ningún centro de datos externo como se muestra en la figura 3. Consiste en acercar la carga del procesamiento todo lo posible al lugar donde los datos son generados, de esta forma se reducen la latencia y los requerimientos en ancho de banda. Además, permite el análisis de mayor cantidad de datos facilitando la toma de decisiones en tiempo real; aporta un mayor grado de seguridad ya que al mantener los datos lo más cerca de su origen no expone los datos confidenciales (Nikolopoulos, Kilpatrick, Barbhuiya, Wang, & Varghese, 2016).

Figura 3

Infraestructura de una red Edge Computing



Fuente: (Camarillo, 2021)

2.5.1. Funciones

Según (Camarillo, 2021) Edge computing realiza las siguientes funciones:

Recopilación de datos. – Recolecta los datos lo más cercano de la fuente usando dispositivos conocidos como Edge gateways o routers inteligentes. Estos datos son agrupados, procesados y seleccionados previamente y son subidos a la nube únicamente en el caso de que no puedan ser analizados localmente.

Almacenamiento. – Se aplica cuando los requerimientos de ancho de banda son altos. Para grandes volúmenes de datos los Edge gateways se usan como servidores de réplica.

Monitoreo con Inteligencia Artificial. – en conjunto con algoritmos de aprendizaje automático se realiza un continuo monitoreo de dispositivos conectados en tiempo real.

Comunicación máquina a máquina. – este tipo de comunicación se usa para el intercambio automático de información entre dispositivos terminales facilitando así el control y supervisión de dispositivos e infraestructuras.

2.5.2. Edge Computing y Nvidia Jetson Nano

Con la aparición de la Inteligencia Artificial, muchos fabricantes han optado por usar sistemas y software especiales para tareas de aprendizaje profundo. En el proceso de Deep Learning son visibles dos fases: entrenamiento, etapa en la cual la red neuronal aprende características de las imágenes y ocasionalmente suele realizarse en centros de datos, y la etapa de inferencia en la cual se clasifica una imagen sin catalogar en una red ya entrenada. Ambas operaciones son procesos computacionalmente costosos y requieren de hardware y software especializado.

Nvidia y su plataforma informática paralela Compute Unified Device Architecture (CUDA) han introducido una placa de desarrollo diseñada para tareas de aprendizaje profundo, la Nvidia Jetson Nano integra GPU para realizar inferencias localmente en tiempo real. Además, el sistema ejecuta los principales marcos de procesamiento de aprendizaje profundo, como TensorFlow, PyTorch o Keras. También, es capaz de procesar información rápidamente sin comunicarse con servidores externos, permite desarrollar aplicaciones, pequeños, económicos y de bajo consumo, que con Raspberry Pi serían muy lentos de ejecutar debido a que no cuentan con hardware específico (Retana Ribeiro , 2020).

2.6. Visión Artificial

La visión artificial pertenece al campo de la inteligencia artificial, y abarca un conjunto de técnicas de procesamiento y obtención de particularidades de una imagen o video, son procesadas e interpretadas por computador o dispositivo electrónico, a través de la extracción de datos desde un dispositivo óptico como una cámara.

Este sistema se compone de dos elementos principales, el hardware encargado de captar la imagen y el software de análisis y procesamiento de datos. Actualmente, esta tecnología tiene diversas aplicaciones desde el control de calidad de productos mediante la detección de defectos en algún objeto, detección de intrusos o en combinación con la robótica colaborativa es posible seleccionar y ordenar piezas (Platero, 2009).

2.6.1. Aplicaciones de la Visión Artificial

Existe una variedad de aplicaciones asociadas a la Visión Artificial, las cuales se muestran en la figura 4. En la agricultura mediante la implementación de técnicas adecuadas es posible minimizar la aplicación de fertilizantes, detectar defectos del cultivo, presencia de plagas, de esta manera se logra adecuar las condiciones para obtener un producto sano (García & Caranqui, 2015).

En el campo de la biología con el fin de distinguir determinadas particularidades en vegetales o animales tomando en cuenta su tamaño, forma, color y textura, así como en la medicina es posible trabajar con imágenes de radiografías, mamografías, ecografías y resonancia magnética.

En el área de seguridad, permite detectar la presencia de algún intruso, acceder a instalaciones mediante reconocimiento ocular o a través de la huella digital. En el ámbito de la teledetección es posible detectar incendios, inundaciones, zonas deforestadas u obstáculos en carreteras mediante imágenes satelitales.

En el sector comercial se posible realizar un mayor control de calidad ya que al ser un proceso automatizado es importante verificar que los productos cumplan con requerimientos de calidad establecidos. al usar Visión Artificial es posible detectar defectos en la producción, y aplicando algún algoritmo, clasificar el producto por su apariencia (García & Caranqui, 2015).

Figura 4

Aplicaciones de la Visión Artificial

Adaptado de: (Platero, 2009)

2.6.2. Etapas

En la primera etapa se realiza la adquisición de imágenes. Esta debe ser lo más adecuada posible con el fin de que el proceso de reconocimiento sea exitoso, luego se pasará a la fase de procesamiento en donde se mejorará la calidad informativa de las imágenes adquiridas incluyendo técnicas para corregir la calidad de la imagen y realzar detalles (Platero, 2009).

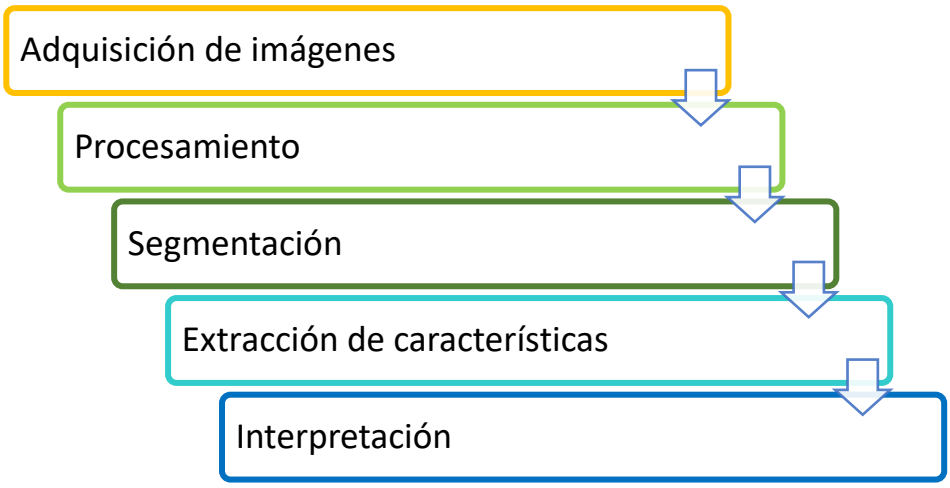
En la fase de segmentación se divide la imagen para poder aislar zonas que contengan un significado que pueda aportar al estudio; posteriormente, se procede a realizar la extracción de características, que pueden ser de tipo morfológico tales como área, perímetro, excentricidad, esqueletos, así como características basadas en textura y color (Platero, 2009).

Finalmente, se reúnen todas las imágenes que se han ido arreglando para que puedan ayudar en lo máximo posible para el estudio. Se utilizan las imágenes o filmación para poder construir un modelo tridimensional de la escena y entre más número de imágenes, se puede tener un mejor modelo de representación para el estudio (Platero, 2009).

En la figura 5 se presenta el proceso que se cumple en la visión artificial, indicando cana una de sus fases desde la adquisición de la imagen hasta la etapa de interpretación.

Figura 5

Etapas de Visión Artificial



Adaptado de: (Platero, 2009)

2.6.3. Herramientas para el Tratamiento de Datos

Existen varias herramientas para visión artificial y tratamiento de datos, a continuación, se detallan las siguientes:

Matblab. – Software matemático presentado por la empresa Mathworks, ofrece un entorno de desarrollo integrado (IDE) y lenguaje de programación M propio de la empresa disponible para sistemas Unix, Windows y Mac OS. Incluye herramientas graficas que en conjunto de algoritmos permiten la visualización, procesamiento, análisis

de imágenes; una de estas herramientas es Computer Vision System Toolbox la cual interpreta escenarios reales y mediante imágenes y videos detecta, clasifica y sigue objetos (García & Caranqui, 2015).

Modulo IMAQ Vision Algorithms. – Herramienta para la adquisición y procesamiento de imágenes aplicadas al sector industrial de la empresa National Instruments, ofrece algoritmos de reconocimiento de patrones, detección de bordes, reconocimiento de caracteres. etc.

Matrox Imaging Library. - La empresa Matrox Electronic Systems ofrece herramientas de software participativo y librerías de programación para desarrollar aplicaciones que permitan la captura, procesamiento, análisis de imágenes (García & Caranqui, 2015).

Librerías de Open Computer Vision. – Librerías de Visión Artificial desarrolladas en código C y C++ por la empresa Intel Corporation. Disponible para las plataformas de Linux, Windows y MAC OS, contiene funciones de captura de imágenes, calibración de cámaras, visión robótica, reconocimiento de objetos y procesamiento.

2.6.4. Manejo y Clasificación de imágenes por Computadora

Las cámaras digitales se rigen por modos de color, estándares y recomendaciones usadas para representar los colores. A continuación, se detallan los modelos aplicados en:

Imágenes RGB. – Usado para representar imágenes en monitores y cámaras. Se trata de imágenes a color conformadas por los colores primarios rojo, verde y azul; para representar un color RGB fija un valor de intensidad ara cada pixel para cada uno de los componentes. Este valor varía entre 0 y 255, siendo 0 la representación del color negro y 255 blanco (Ardila Urueña, Cortes, & Mendoza Vargas, 2011).

Espacio YIQ. – Toma en consideración la sensibilidad del ojo humano a los cambios de iluminación frente a la saturación. Proporciona información de vídeo importante. Los estándares YIQ representan la Y a la mayor cantidad de bits, y la I o Q para menos bits.

HSI. – Sus siglas corresponden a Tonalidad, Saturación e Intensidad. Es un modelo muy usado en aplicaciones de procesamiento de imágenes ya que se basa en las propiedades de percepción de color similar a la forma en la que lo haría un humano. Los componentes H y S están relacionados a la forma en la que los humanos distinguimos los colores.

2.6.6. Limitaciones

Según (Fernández García, 2016) la Visión Artificial posee limitaciones que se describen a continuación:

Variación de iluminación. – La imagen a procesar podría verse afectada debido a la presencia de sombras o reflejos, cambios en los tonos de luz y distintos niveles de iluminación durante el día.

Escala. – El modificar el tamaño de las imágenes dificulta la detección del objeto razón por la cual el tiempo de procesamiento aumenta y con ellos es necesario mayor uso de recursos computacionales.

Alteración de los objetos. – Una imagen desproporcionada impide la extracción de cualidades del objeto en cuestión.

Obstrucción. - Ocurre cuando un objeto se sobrepone a otro, dificultando la visibilidad, perdiendo datos sobre la imagen.

2.7. Aprendizaje de máquina

El aprendizaje automático o Machine Learning pertenece al área de la Inteligencia Artificial y genera algoritmos capaces de aprender sin necesidad de que el diseñador programe de manera específica las acciones a realizar; únicamente es necesario alimentar al algoritmo con una gran cantidad de datos para que éste aprenda y realice tareas específicas según se presente el caso (Sandoval, 2018).

2.7.1. Aprendizaje por refuerzo

El algoritmo aprende observando su entorno, en donde aplica un esquema de premio y castigo para tomar decisiones en base a la experimentación con los datos. Como se muestra en la figura 6, intervienen tres componentes: agente, ambiente y acción. El agente se encarga de la toma de decisiones y el entorno devuelve el nuevo estado y la recompensa conseguida, si esta es positiva se estará reforzando el comportamiento futuro, caso contrario existirá una penalización de forma que el agente actúe de forma distinta ante esta situación (Bagnato, 2020).

Figura 6

Esquema de Aprendizaje por Refuerzo

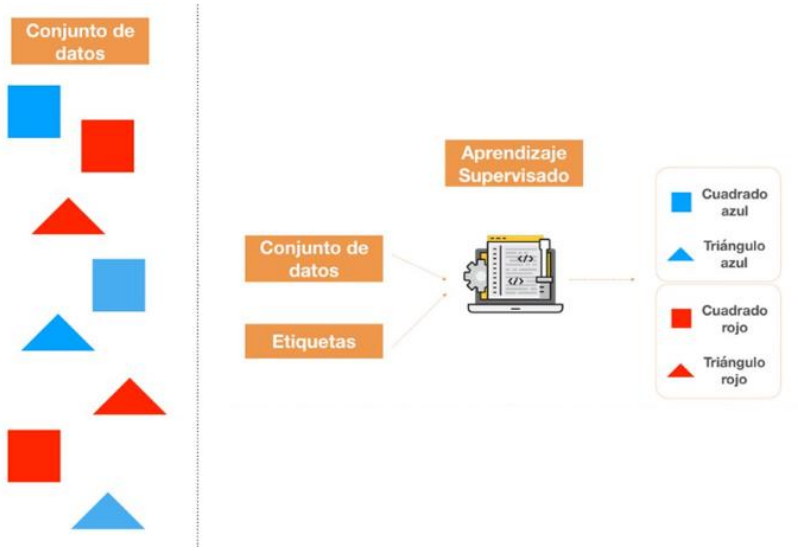
Fuente: (Bagnato, 2020)

2.7.2. Aprendizaje supervisado

Este tipo de aprendizaje se basa en la información que recibe sobre las características del objeto que quiere identificar, esto con el objetivo de clasificación o predicción. Para el caso de predicción se requiere un análisis de un conjunto de

características similares, mientras que, para la clasificación, como se muestra en la figura 7, se hace uso de patrones para situar al objeto dentro de un grupo en específico (Sandoval, 2018).

Figura 7
Esquema Aprendizaje Supervisado



Fuente: (Gonzalez, 2018)

2.7.3. Aprendizaje no supervisado

En el aprendizaje no supervisado de la figura 8 los algoritmos basan entrenamiento en datos sin una clasificación previa. Está orientado a realizar tareas de agrupamiento según sus características donde su objetivo es localizar patrones en los datos, de modo que, asume pertenecen a un mismo grupo (Sandoval, 2018).

Figura 8

Esquema Aprendizaje No Supervisado

Fuente: (Gonzalez, 2018)

2.7.4. Deep learning

El aprendizaje profundo es una de las aplicaciones de la Inteligencia Artificial, utilizado para solventar problemas de gran complejidad y que requiere un enorme conjunto de datos iniciales y amplia capacidad de procesamiento. Este proceso se realiza mediante redes neuronales organizadas por capas de menor a mayor complejidad para reconocer patrones. Entre las aplicaciones se encuentran el reconocimiento de voz, la visión artificial y la identificación de autos (Rouhiainen, 2018).

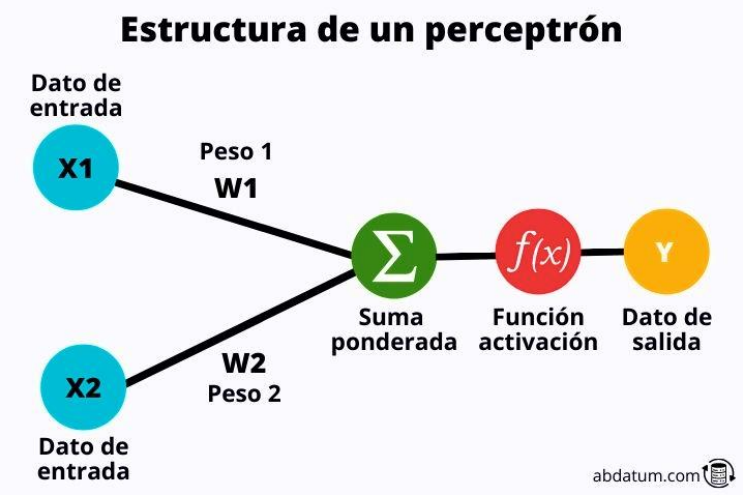
2.7.4.1. Red neuronal artificial (ANN)

Las redes neuronales artificiales se basan en el funcionamiento y comportamiento del cerebro humano, realiza el procesamiento mediante la aplicación de un algoritmo de aprendizaje que mediante una o varias funciones de entrada son capaces de encontrar y relacionar patrones basado en datos previos (Cañadas, 2021).

Las partes básicas consisten en las entradas encargadas de receptar información, pesos y el núcleo de la neurona, la función de activación la cual calcula el estado de actividad que hay en una neurona, y su valor de salida, como se muestra en la figura 9.

Figura 9

Estructura de una Neurona Artificial



Fuente: (Cañadas, 2021)

2.7.4.2. Red neuronal profunda (DNN)

Una red neuronal profunda es un tipo de red neuronal artificial, la cual cuenta con una entrada, varias capas ocultas y una salida. El objetivo de este tipo de red neuronal es recoger un conjunto de datos de entrada, realizar cálculos progresivos que permiten aprender particularidades del objeto tomando en cuenta los detalles presentes en la imagen y clasificar el objeto deseado (Id Digital School, 2021).

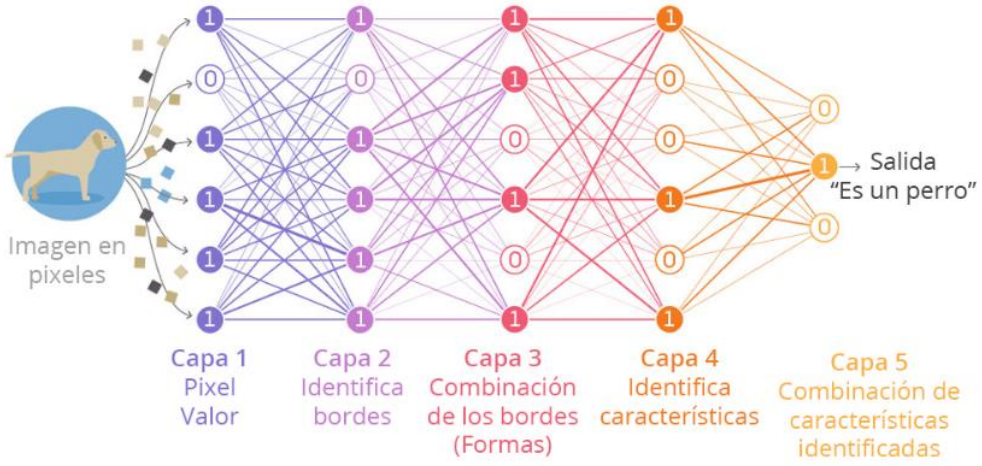
El entrenamiento de la red combina el aprendizaje supervisado y no supervisado. En los primeros niveles se procesan datos, de forma que se pueda identificar objetos simples; mientras que, en los niveles más profundos, se necesita mayor detalle para la identificación (Id Digital School, 2021).

Como se muestra en la figura 10, el proceso comienza con el primer nivel, en el cual se identifican los bordes de la imagen, zonas claras y oscuras; posteriormente en el segundo nivel, se construyen formas como rectas para en el tercer nivel obtener formas

más competas como círculos o rectángulos, en el nivel siguiente se combinan estas figuras hasta identificar características del objeto (Id Digital School, 2021).

Figura 10

Estructura de una Red Neuronal Profunda (DNN)



Fuente: (Id Digital School, 2021)

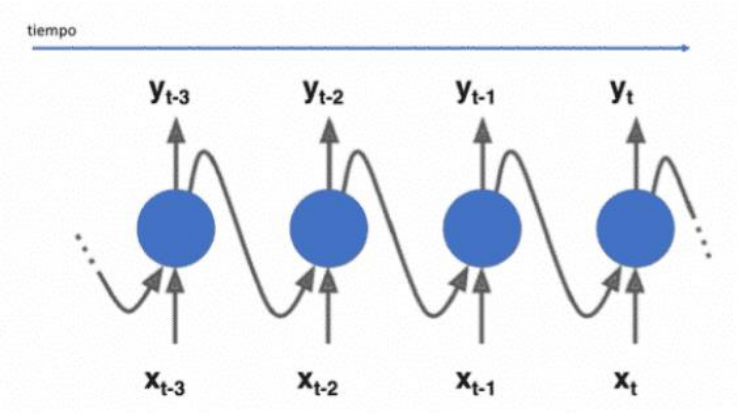
2.7.4.3. Red neuronal recurrente (RNN)

Este tipo de redes son utilizadas para estudiar datos temporales; a diferencia de una red neuronal simple, cuya función de activación actúa en una única dirección, desde la entrada hacia la salida, una red neuronal recurrente contiene vínculos que permiten una retroalimentación entre las neuronas, recordando valores previos (Torres, 2019).

Como se muestra en la figura 11, cada neurona absorbe dos entradas, la que pertenece a la capa preliminar y a su vez la salida del intervalo anterior de la misma capa.

Figura 11

Red Neuronal Recurrente



Fuente: (Torres, 2019)

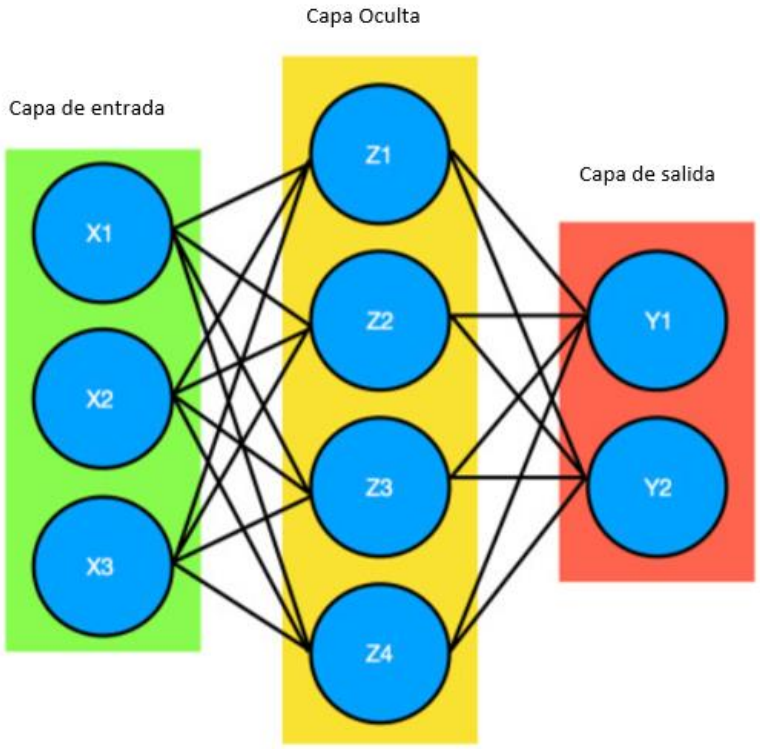
2.7.4.4. Red neuronal convolucional (CNN)

Actualmente, aplicaciones que emplean reconocimiento de objetos e imágenes se basan en el uso de Redes Neuronales Convolucionales. Esta arquitectura de red aprende de los datos, sin necesidad de extraer características manualmente, genera resultados muy precisos, y es posible re entrenar la red con nuevas operaciones de reconocimiento, aprovechando las redes preexistentes.

En la figura 12, se muestra la arquitectura de una red neuronal convolucional, es similar a otras redes neuronales, consta de capa de entrada, varias capas ocultas, y una capa de salida. La capa de entrada contiene neuronas que simbolizan los datos que la red neuronal va a entrenar, especifican el tamaño de los pixeles de una imagen y el número de canales, generalmente, suelen ser 3 canales representando así los valores RGB de cada pixel. Las capas ocultas permiten extraer características desde líneas, curvas, brillo, bordes, hasta características más complejas que definen el objeto de manera única. La capa de salida contiene neuronas que realizan el proceso de clasificación (Massiris, Delrieux, & Fernández , 2018).

Figura 12

Estructura de una red neuronal convolucional



Fuente: (Rodriguez, 2018)

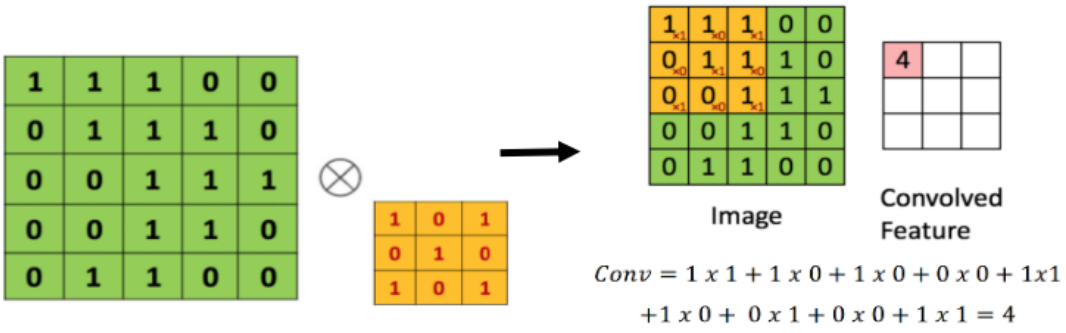
Las operaciones principales de una Red Neuronal Convolucional son: convolución, activación y pooling. Estas operaciones se repiten en varias capas, de esta forma cada capa aprende a identificar diferentes características.

- **Convolución**: a las imágenes de entrada se les aplican varios filtros convolucionales, cada uno de ellos activa ciertas características de las imágenes, para ellos se toma un grupo de pixeles cercanos de la imagen de entrada y se realiza un producto escalar por una matriz de 3x3 llamada kernel, esta matriz se desplaza a través de la imagen de entrada, de izquierda a derecha y de arriba hacia abajo, hasta que haya pasado por toda la imagen (Parada Torralba, 2022).

  El proceso de convolución se muestra en la figura 13.

Figura 13

Proceso de convolución



Fuente: (Martinez, 2018)

- **Función de activación:** La función de activación típicamente usada es la ReLU (Unidad de Rectificador Linear). Al finalizar la operación de convolución se asignan valores negativos y positivos, con la función de activación ReLu se remplazan los valores negativos de entrada por cero y solo las características activadas pasan a la siguiente capa, permitiendo un entrenamiento más rápido y eficaz (Sotaquirá, 2018)
- **Capa de agrupación o Pooling**: simplifica las dimensiones en la salida de la capa convolucional y reduce el número de parámetros que la red necesita aprender. Hay dos tipos de agrupación: Max Pooling, este método elige el pixel de mayor valor, descarta los demás valores y lo envía a la matriz de salida, y Average Pooling, se calcula el valor promedio de cada sub matriz (Pathak, 2022)

(Rodriguez, 2018) destaca que ara alcanzar un buen desempeño de la red entrenada es importante el uso de hiperparámetros, expuestos a continuación:

- **Epoch o ciclo:** se refiere al número de veces que el conjunto de datos debe pasar por el algoritmo de aprendizaje.
- **Batch size:** tamaño de lote que tiene cada iteración de un ciclo.

- **Iteración:** número de lotes necesarios para completar un ciclo.
- **Función de pérdida:** define que tan buena es la red neuronal.
- **Learning rate:** la tasa de aprendizaje es el porcentaje de cambio con el que se actualizan los pesos en cada iteración. Los pesos de entrada son actualizados cada que se realiza el proceso de iteración para obtener una mejor aproximación.
- **Optimizer:** mejora los valores de los parámetros para reducir el error de entrenamiento.

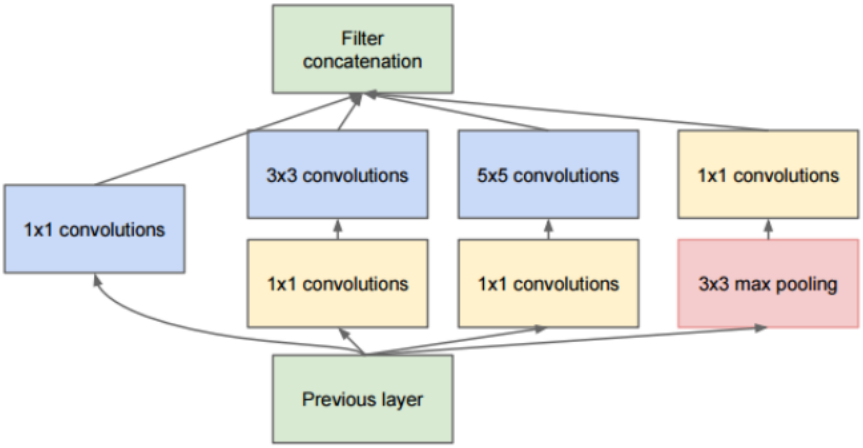
(Siyah, 2020) menciona que en el campo de las redes neuronales se han establecido varias arquitecturas que han sido puestas a prueba en el Concurso Anual de Visión por Computador sobre un conjunto de datos disponible en ImageNet (ILSVRC) realizado entre los años 2010-2017. Algunas de las arquitecturas convolucionales más destacadas son:

- **LeNet-5:** arquitectura antigua, propuesta para reconocer dígitos escritos a mano. Admite como entrada imágenes de tamaño 32 x 32 en escala de grises. Cuenta con 60000 parámetros.
- **AlexNet:** es similar a la LeNet-5 pero es una red más profunda, cuenta con 8 capas. Tiene un mayor número de filtros por capa y admite como entrada de tamaño 227 x 227 x 3(RGB). Tiene un índice de error del 15.3%.
- **GoogleNet:** fue el ganador del concurso en 2014, cuenta con 22 capas y 4 millones de parámetros. Se implementan módulos inception, con 6 capas convolucionales y una capa de pooling. Maneja un índice de error de 6.67%.
- **Inception:** es una versión mejorada de GoogleNet, la arquitectura de la figura 14 cuenta con 42 capas y existen menor número de parámetros en cada módulo inception, logrando reducir el costo computacional. En esta arquitectura se concatenan los resultados de aplicar diferentes filtros de diferentes tamaños

permitiendo así, la extracción de características tanto generales como locales. Admite como entrada de tamaño 300 x 300 x 3(RGB). Tiene un índice de error de 3.58% y un índice de precisión de 93.9%.

Figura 14

Arquitectura Inception con reducción de dimensiones



Fuente: (Stewart, 2019)

2.8. Trabajos relacionados

Como primera referencia se hace alusión al trabajo de titulación de grado del señor Lugo Noboa, con el título: “Diseño de un sistema de visión artificial mediante una plataforma usando un drone para identificar la plaga lancha en campos agrícolas de tomate riñón en Romerillo Bajo”. En el trabajo se realiza la adquisición de datos mediante técnicas de vuelo usando un drone; para la detección e identificación de enfermedades en las hojas se hace uso de redes neuronales convolucionales con el módulo DNN de OpenCV; posteriormente, se desarrolla una aplicación que proporciona alertas al agricultor (Lugo Noboa, 2021) .

Como segunda referencia se hace mención al trabajo de titulación realizado por el señor Acuña Chapa, con el título: “Diseño de un sistema de visión artificial para la clasificación de limón utilizando Raspberry Pi”. Este proyecto aplica técnicas de visión

artificial junto con procesamiento de imágenes, transformación del modelo de color RGB a HSV para medir ciertos parámetros como el tamaño, color y defecto del limón, con el fin de reducir costos en el proceso de clasificación (Acuña Chapa, 2020) .

El tercer trabajo relacionado se titula “Evaluación de la calidad de la lechuga utilizando Red neuronal artificial”. Esta investigación plantea un sistema de visión artificial para la estimación de la calidad del cultivo de lechuga mediante el procesamiento de la imagen receptando características y su posterior clasificación aplicando red neuronal artificial de retropropagación. Este proyecto se presenta como una solución a la clasificación manual del producto apto y no apto para el consumo humano (Valenzuela, y otros, 2017).

El cuarto trabajo relacionado “Sistema automático de reconocimiento de frutas basado en visión por computador”, presenta un sistema que permite identificar una fruta de entre un conjunto. La estructura es similar a una báscula con un sistema de luz en anillo para eliminar la presencia de sombras; además el documento compara métodos de clasificación como KNN y Bayesiano, junto con los modelos del color RGB y HSV (Montoya Holguín, Cortés Osorio, & Chaves Osorio, 2014).

Los trabajos mencionados anteriormente demuestran que los sistemas de visión artificial son de gran impacto en el campo de la agricultura inteligente. Aplicar este tipo de herramientas permiten tener un mayor control en el desarrollo de la producción, optimizando los procesos que van desde la post-cosecha hasta que el producto llega al consumidor final mediante la extracción de características que determinan la calidad del producto. Además, es posible reducir costos de producción y mitigar el margen de error en la clasificación del producto debido a factores humanos como cansancio físico o fatiga visual.

Capítulo III

DESARROLLO EXPERIMENTAL

En el tercer capítulo se describe la metodología aplicada al desarrollo del proyecto, el análisis de requerimientos para la selección tanto de Hardware como de Software, necesarios para el desarrollo del sistema; además, se presenta el diseño y diagramas del sistema.

3.1. Metodología

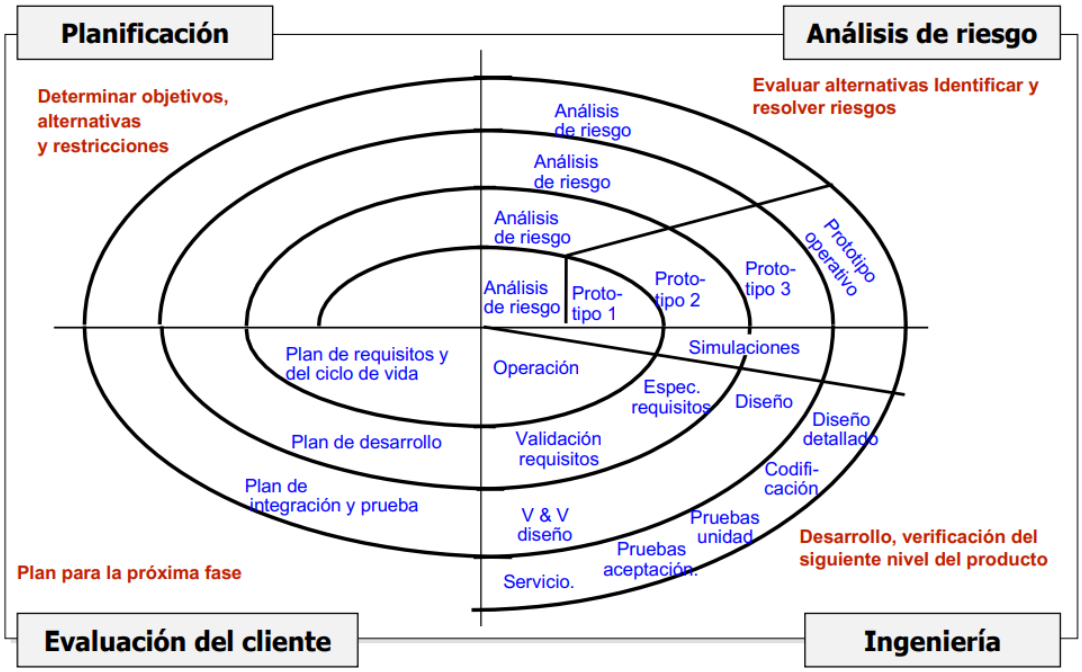
El presente proyecto se ejecuta en base al cumplimiento de los objetivos, para lo cual se han establecido las siguientes fases: preliminar, planificación, ejecución y resultados. En la fase preliminar se determina la metodología y algoritmos adecuados para la identificación y clasificación de objetos en tiempo real, así como la selección de tanto de Hardware como de Software necesarios para el desarrollo del sistema guiándose del estándar IEEE 29148. Para las fases de planificación y ejecución se aplica una metodología de desarrollo llamada Modelo en Espiral en sus cuatro etapas generales: planificación, análisis de riesgos, ingeniería y evaluación del cliente.

3.1.1. Modelo en espiral

El modelo en espiral es método de desarrollo de software evolutivo, el cual combina los modelos Iterativo y Cascada; su ciclo de vida se establece mediante una espiral con actividades estructurales llamadas regiones de tareas, las cuales se cumplen desde el centro de la espiral hacia el exterior, permitiendo realizar entregas progresivas al cliente de forma que el producto final haya ido evolucionando en cada ciclo repetitivo. A diferencia de otros modelos, este modelo considera los riesgos que pueden aparecer dentro del programa, estos son distribuidos en cada región para limitar el riesgo y aumentar la funcionalidad (Zumba Gamboa & León Arreaga , 2018). En la Figura 15 se muestra las fases que plantea el Modelo en Espiral.

Figura 15

Modelo en espiral



Fuente: (García Peñalvo, García Holgado, & Vázquez Ingelmo, 2020)

El modelo en espiral consta de cuatro fases:

- **Planificación:** en esta etapa se determinan los objetivos principales, se identifican posibles riesgos y se establecen alternativas a través de una comunicación entre desarrollador-cliente. Se realiza un plan administrativo en el cual se detalla un cronograma, recursos disponibles y costos.
- **Análisis:** se identifican los riesgos, se evalúan riesgos técnicos como componentes de hardware, además de información adicional referente al proyecto. Para reducir los riesgos se proponen alternativas y se realizan prototipos, simulaciones, diseños.

- **Ingeniería:** en esta etapa se añaden funcionalidades a los prototipos propuestos en la etapa anterior, posteriormente se realizan instalaciones, pruebas continuas para asegurar el correcto funcionamiento del sistema y se brinda soporte al usuario.
- **Evaluación:** Se cuestiona si los objetivos propuestos se han cumplido con éxito**.** Se establece si es pertinente finalizar con las fases del proyecto o repetir el ciclo para realizar mejoras del sistema.

3.2. Análisis

En esta sección se realiza un análisis de la situación actual junto con la información obtenida de trabajos relacionados, para conocer dimensiones, condiciones ambientales y restricciones que pueden presentarse en el desarrollo del presente proyecto, además se establecen los requerimientos iniciales del sistema, requisitos de hardware y software.

3.2.1. Situación Actual

El presente proyecto se desarrolla en un huerto de lechugas ubicado en el sector de Natabuela, el cual cuenta con dos escenarios que permitan probar el sistema de detección de estrés hídrico en un ambiente de producción. Ambos escenarios se encuentran al aire libre, cada hortaliza dentro de este lote será sembrada con una distancia adecuada y serán regadas con el método de riego por goteo. La diferencia entre los dos lotes, es que uno de los escenarios se encuentra en óptimas condiciones, preparado, abonado y listo para su producción, y será regado periódicamente; mientras que, el segundo escenario, tiene un regado esporádico, condiciones no aptas para el desarrollo de la hortaliza.

Actualmente el huerto no cuenta con ningún tipo de infraestructura especializada para monitoreo de siembras o de un sistema de evaluación de calidad del producto cosechado. El proceso de recolección se realiza de manera manual, lo cual puede traer como consecuencia un producto cosechado antes de tiempo o de mala calidad, debido a factores humanos como cansancio físico o fatiga visual. Finalmente, el usuario será capaz de tener un mayor control en el desarrollo de la producción, y efectuar recomendaciones y medidas correctivas del lote afectado para asegurar la obtención de un producto sano y de calidad. Para ello se realizará una interfaz amigable con el usuario que permita recibir alertas o información sobre la clasificación de cada lote del cultivo de lechuga.

3.2.2. Técnicas de Recolección de Información

Para la recopilación de los requisitos del sistema se hace uso de distintas técnicas de recopilación de información, entre ellas, una entrevista y la revisión de trabajos relacionados con el fin de conocer elementos de hardware aplicados en el diseño del sistema, además de herramientas de software y algoritmos aplicados en sistemas de visión artificial.

3.2.1.1. Entrevista

La entrevista realizada a la Sra. Rosa Gómez, encargada de los cultivos de lechuga en la parroquia de Natabuela, tiene el objetivo de recopilar información relevante sobre el ciclo de producción, tiempos de cosecha, cuidados, requerimientos de siembra de lechuga.

El cuestionario tiene un formato mixto con preguntas abiertas y cerradas. A continuación, en la tabla 2 se muestran las preguntas y en el ANEXO A se muestra el cuestionario en su respectivo formato.

Tabla 2

Preguntas de la entrevista

N°	Preguntas abiertas y cerradas
1	¿Cada qué tiempo se siembra la lechuga?
2	¿Cómo se prepara el suelo para el cultivo de lechuga?
3	¿Cuánto tiempo se tarda la lechuga para ser cosechada?
4	¿Cuál es la cantidad de agua que necesita el sembrío y el tiempo de riego?
5	¿Cuál es la cantidad de abono que se requiere durante el ciclo de producción de lechuga?
6	¿Cuáles son los cuidados que requiere el cultivo de lechuga?
7	¿Cuál es la hora recomendada para cosechar la lechuga?
8	De la siembra inicial, ¿cuál es el porcentaje esperado de producción?
9	¿Cuáles son las causas de la pérdida de la producción?
10	¿Qué medidas se utilizan para mitigar la pérdida?

3.2.1.2. Análisis de entrevista

Mediante la entrevista realizada a la Sra. Rosa Gómez, encargada de los cultivos de lechuga en la parroquia de Natabuela, respecto a las preguntas 1, 3 y 7, se pudo conocer el tiempo de siembra, el cual se realiza de manera periódica cada mes. El tiempo de cosecha depende del tipo de lechuga sembrada, se demora 2 meses en estar lista para el consumo, además, debe ser cosechada en la mañana para evitar daños en la hortaliza por el exceso de calor.

De las preguntas 2,4 y 5, se pudo observar que la producción de lechuga aún se realiza de manera tradicional, lo cual implica la preparación del terreno mediante el arado, el uso de abono animal y el riego de surcos. Estos sistemas tradicionales son poco eficientes ya que demandan mayor cantidad de agua y recursos.

De las preguntas restantes, se puede observar que el cultivo de lechuga no requiere mayores cuidados, ya que es una planta que se adapta a cualquier clima y terreno; sin embargo, existe un porcentaje de pérdida de producción en su mayoría debido a exceso

de sol o agua, razón por la cual una de las medidas para mitigar las pérdidas es mediante la fumigación cada 10 o 15 días. Si la planta está demasiado enferma significa pérdida, ya que es más difícil curarle.

Después de realizar esta entrevista se puede determinar que es factible implementar un Sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de Visión Artificial, de forma que el agricultor pueda mantener el control de las condiciones hídricas, evitar enfermedades en el cultivo y la pérdida de la producción, además, le brinda ventajas en cuanto a recursos económicos y al tiempo de recolección de la lechuga a través de la clasificación de producto bueno o malo.

3.2.1.3. Revisión de trabajos relacionados

Esta sección tiene el objetivo de recolectar información que permita establecer los requerimientos técnicos, funcionales para el diseño e implementación del sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de visión artificial. En el ANEXO B, se presenta una ficha bibliográfica de los proyectos mencionados y su respectivo análisis.

3.3. Requerimientos del Sistema

Para el desarrollo de esta sección se toma como base el estándar ISO/IEC/IEEE 29148, documento que brinda pautas y disposiciones relacionados a procesos de implementación de sistemas o productos de software, permitiendo reunir requerimientos que el proyecto requiere para su funcionamiento y aplicación. Los requerimientos son un elemento importante en este proyecto. En la tabla 3 se listan los diferentes actores involucrados en el desarrollo del presente proyecto de titulación.

Tabla 3

Actores Involucrados

3.3.1. Nomenclatura de Requerimientos

Actores Involucrados		
N°	**Actor**	**Función**
1	Carla Rivera	Desarrolladora
2	MSc. Fabián Cuzme	Director
3	MSc. Edgar Maya	Asesor

Cada uno de los repquerimientos se identifican mediante una abreviatura, para facilitar el manejo de información. En la Tabla 4 se muestran las abreviaturas utilizadas.

Tabla 4

Abreviatura de los requerimientos

Nomenclatura	Descripción
StSR	Requerimientos Stakeholders
SySR	Requerimientos del Sistema
SrSH	Requerimientos de Hardware y Software

Las tablas de requerimientos están conformadas por tres columnas, la primera columna posee un número identificativo del requerimiento, la segunda columna detalla el requerimiento y la columna final establece la prioridad o nivel de impacto que puede tener el implementar un requerimiento en el sistema, esta se encuentra dividida en tres niveles: Alta, Media y Baja.

3.3.3. Requerimientos de Stakeholders

Los requerimientos de stakeholders reúne requisitos y necesidades que los clientes, usuarios u otros interesados soliciten de su incorporación en el sistema. A continuación, en la tabla 5 se evalúan los requerimientos de usuario y operacionales.

Tabla 5

Requerimientos de stakeholders

Requerimiento de stakeholders (StSR)

#	Requerimiento	Prioridad		
		Alta	**Media**	**Baja**
	Requerimientos operacionales			
StSR1	Cámara de resolución mínimo de 5mpx.	X		
StSR2	Aseguramiento de parámetros ópticos mediante una estructura estable para la obtención de imágenes claras.	X		
StSR3	Periodo de adquisición de datos, las imágenes deben ser adquiridas en horas donde existan buenas condiciones de luz.		X	
StSR4	Acceso a un punto de energía eléctrica.	X		
StSR5	Almacenamiento de imágenes en base de datos o nube.		X	
	Requerimientos de usuarios			
StSR6	Interfaz gráfica amigable con el usuario.	X		
StSR7	Fácil acceso a la interfaz gráfica.	X		
StSR8	La información debe estar en un formato legible para el usuario.	X		

3.3.4. Requerimientos del Sistema

En este apartado se definen los requerimientos iniciales del sistema (SySR), los cuales establecen las funcionalidades, limitaciones y características de comportamiento que debe tener el sistema. Contiene los requisitos de uso, las interfaces, los modos, estados y físicos. La tabla 6 contiene los requerimientos del sistema.

Tabla 6

Requerimientos del sistema

Requerimiento del sistema (SySR)				
#	Requerimiento	Prioridad		
		Alta	Media	Baja
Requerimientos de interfaz				
SySR1	Uso de tecnología inalámbrica.	X		
Requerimientos de uso				
SySR2	Sistema fácil de manejar.	X		
SySR3	Acceso a resultados y parámetros de análisis.		X	
Requerimientos performance				
SySR4	El sistema no debe interferir con el crecimiento del cultivo.	X		
SySR5	Bajo consumo de energía.		X	
SySR6	Almacenamiento y procesamiento de datos en la nube.	X		
SySR7	Técnica de tratamiento de imágenes.	X		
SySR8	Visualización gráfica de resultados.		X	
Requerimientos de modo y estado				
SySR9	La interfaz gráfica permite visualización sobre la clasificación de cada lote del cultivo.	X		
Requerimientos físicos				
SySR10	Las cámaras deben ser ubicadas en zonas despejadas para evitar obstáculos.		X	
SySR11	Las cámaras deben tener protección debido a su ubicación en la intemperie.		X	
SySR12	El sistema debe tener una tarjeta de memoria con suficiente espacio para no saturarse.			X

3.3.5. Requerimientos de Arquitectura

Los requerimientos de arquitectura permiten determinar componentes tanto de hardware como de software, los cuales son esenciales para el funcionamiento e implementación del sistema. En la tabla 7 se establecen los requerimientos lógicos, diseño, hardware, software y eléctricos.

Tabla 7

Requerimientos de arquitectura

Requerimiento de Arquitectura (SrSH)				
#	**Requerimiento**	**Prioridad**		
		Alta	**Media**	**Baja**
	Requerimientos de diseño			
SrSH1	Hardware y software compatibles entre sí.	X		
SrSH2	Comunicación entre la plataforma de procesamiento de imágenes, base de datos e interfaz gráfica.	X		
SrSH3	Sistema accesible para los administradores.	X		
	Requerimientos lógicos			
SrSH4	Acceso a la base de datos.	X		
SrSH5	Compatibilidad con sistemas Linux.	X		
SrSH6	Compatibilidad con lenguaje de programación.	X		
	Requerimientos de hardware			
SrSH7	Placa embebida.	X		
SrSH8	Cámara para exteriores con resolución mínimo 5 Mpx.	X		
SrSH9	Conexión a internet.	X		
SrSH10	Memoria RAM entre 2 y 4 Gb.		X	
SrSH11	Ranura para tarjeta SD.		X	

	Requerimientos de software		
SrSH13	Compatibilidad entre librerías de visión artificial, tratamiento de imágenes y aprendizaje de máquina.	X	
SrSH14	Licencias Opensource.		X
SrSH15	Lenguaje de programación de alto nivel.	X	
SrSH16	Software para tratamiento de imagen.	X	
SrSH17	IDE para programación.		X
SrSH18	Compatibilidad con Python.	X	
SrSH19	Procesamiento gráfico de técnicas de visión artificial (GPU).		X
	Requerimientos eléctricos		
SrSH20	Tomacorriente para infraestructura.	X	

3.3.6. Selección de Hardware

En esta sección se elige el hardware específico mediante la ponderación de requerimientos de stakeholders, sistema y arquitectura; esta elección se realiza a través de la comparación de diferentes componentes, calificando entre 1 y 0 (1 si cumple, 0 no cumple), donde el elemento con mayor calificación será seleccionado para implementar en el proyecto.

- **Cámara**

Para el presente proyecto es necesario hacer uso de una cámara que tenga una gran apertura de enfoque, específicamente para exteriores y que posea una resolución recomendada para aplicaciones de visión artificial.

Tabla 8

Selección de cámaras

	REQUISITOS							
	StSR1	**StSR5**	**SySR11**	**SySR12**	**SrSH8**	**SrSH9**	**SrSH10**	**Val.**
Ezviz C8C Lite	0	1	1	1	1	1	1	6
Cámara Inteligente Ptz Tuya Smart C18b	1	1	1	1	1	1	1	7
Cámara Ptz Wifi, 1080p exterior (robótica), ip66	0	1	1	1	1	1	1	6
Cámara Cw8 con Wifi 2K Pan & Tilt	1	1	1	1	1	1	1	7

Cumple “1” No cumple “0”

Elección: La Cámara seleccionada es Cw8 con Wifi 2K Pan & Tilt ya que es apta para exteriores, posee una alta claridad de video, campo de visión 360 lo cual permite cubrir amplias áreas y desplazarse tanto horizontal como verticalmente, además permite almacenar los videos de forma local o en la nube.

- **Placa embebida**

El uso de una placa embebida permite realizar varias operaciones de visión artificial. Este dispositivo debe cumplir con los requerimientos establecidos, por lo cual se realiza una comparación entre diferentes opciones, como se muestra a continuación e la tabla 9.

Tabla 9

Selección de placa embebida

	REQUISITOS							
	SrSH 2	**SrSH 3**	**SrSH 8**	**SrSH 10**	**SrSH 11**	**SHRS 12**	**SrSH 19**	**Val.**
Nvidia Jetson Nano	1	1	1	1	1	1	1	7
Raspberry PI 4	1	1	1	1	1	1	0	6
Arduino Mega 2560 R3	0	1	1	1	0	0	0	3
NanoPC-T4	1	1	1	1	1	1	1	7
ASUS Tinker Board S	1	1	1	1	1	0	1	6
Cumple “1” No cumple “0”								

Elección: la placa embebida que cumple con los requerimientos es: Nvidia Jetson Nano de 4GB de RAM, posee compatibilidad con sistema Linux; su cantidad de memoria RAM, un consumo de energía bajo (5W) y el rendimiento de su CPU cuenta con 472 GFLOP.

3.3.7. Selección de Software

La selección del software se realiza en base a los requerimientos de sistema y arquitectura, se analiza el cumplimiento de los requerimientos mediante una tabla comparativa para elegir el lenguaje de programación de alto nivel, una red de entrenamiento y la tecnología de transmisión adecuados.

- **Lenguaje de programación de alto nivel**

Para la elección del mejor lenguaje de programación se consideran requerimientos que aseguren la compatibilidad entre librerías de visión artificial, tratamiento de imagen, aprendizaje de máquina, si tiene licencia de código abierto, que sea

considerado como un lenguaje de programación de alto nivel. A continuación, en la tabla 10 se muestra las opciones de lenguaje de programación.

Tabla 10

Selección de lenguaje de programación

Lenguaje de programación	**REQUISITOS**				
	SrSH 11	**SrSH 12**	**SrSH 13**	**SrSH 15**	**VALOR**
C++	1/2	1	0	1	2.5
Python	1	1	1	1	4
Java Script	1	1	1	1	4
Java	1	1	0	1	3
Cumple “1” No cumple “0”					
Elección. Se elige el lenguaje de programación Python debido a que es muy utilizado en aplicaciones de visión artificial, además permite corregir errores usando diferentes IDEs y bibliotecas como Numpy, OpenCV, Scioy y Pybrain					

- **Tecnología de transmisión inalámbrica**

Para la elección de la tecnología de transmisión inalámbrica se considera el requerimiento SySR1 por la conectividad a la tecnología de comunicación inalámbrica, también, se examina el requerimiento SrSH2, el cual permite establecer la comunicación con la plataforma de procesamiento de imágenes y la interfaz gráfica, también se integra el requerimiento SrSH5, establece la conexión entre la base de datos y la interfaz gráfica para visualización de resultados. A continuación, en la tabla 11, se muestra una comparación entre algunas tecnologías inalámbricas.

Tabla 11

Selección de tecnología inalámbrica

Plataforma IoT	**REQUISITOS**			
	SySR 1	**SrSH 2**	**SrSH 5**	**VALOR**
Wi-Fi (802.11 n)	1	1	1	3
WiMAX (802.16 e)	1	0	1	2
GSM	1	0	1	2
Cumple “1” No cumple “0”				

Elección. La tecnología de transmisión inalámbrica Wi-fi es la más apta, permite establecer un nodo de comunicación como punto de acceso, la conectividad es estable para la transmisión de datos; además, tiene altas prestaciones con demandas de tráfico, tienen una cobertura entre 30 a 300.

3.4. Diseño del sistema

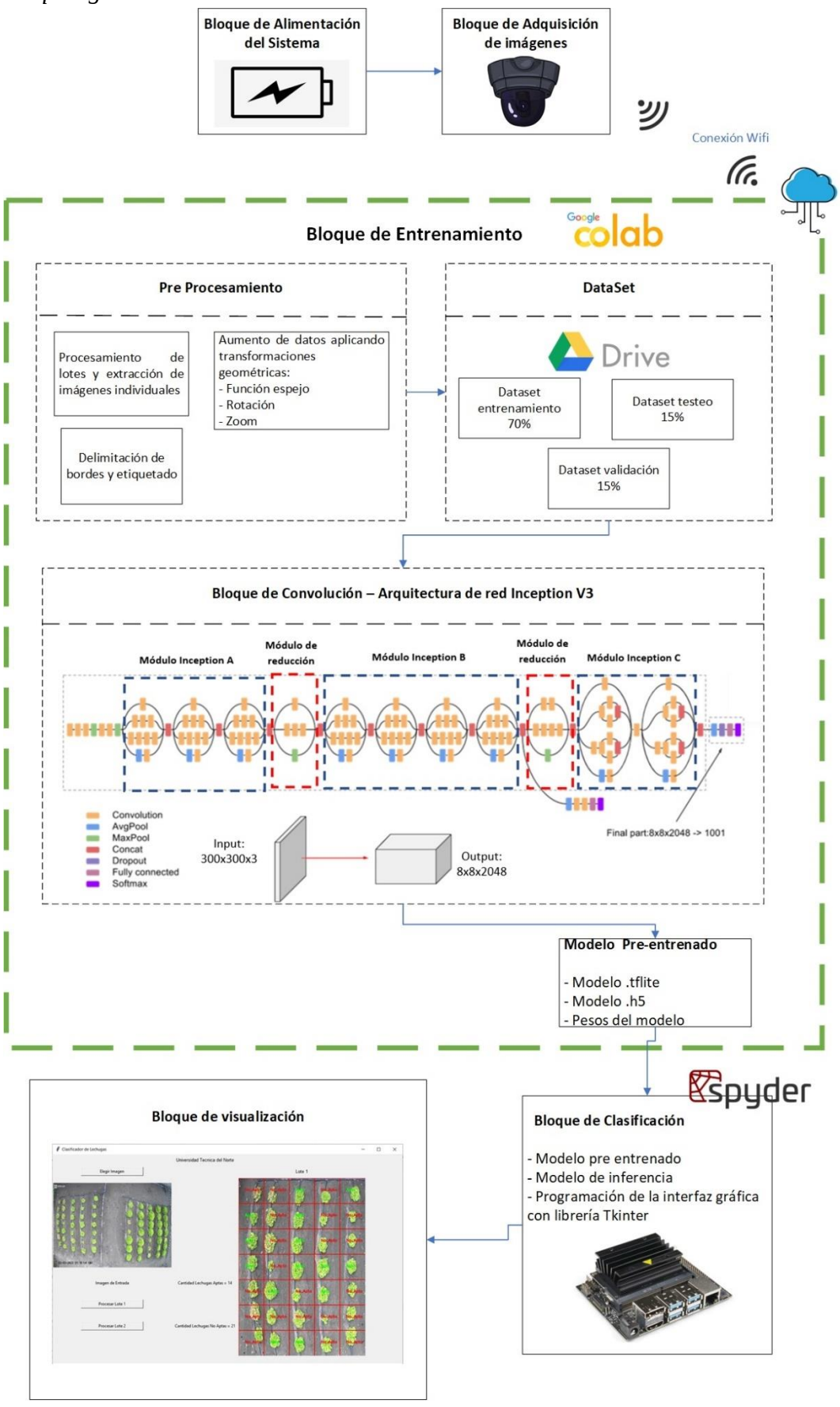
Una vez finalizado el análisis de requerimientos y la selección de Hardware y Software, se procede a realizar el diseño del sistema. Para entender de mejor manera la arquitectura y el funcionamiento del sistema se presentan diagramas de bloque y flujo.

3.4.1. Diagrama general de bloques del sistema

En la figura 16 se muestra el diagrama general el cual detalla los procesos principales del sistema, permitiendo comprender el funcionamiento integral del sistema. Posteriormente se profundizará a detalle cada uno de estos bloques.

Figura 16

Diagrama de bloques general



El diagrama se encuentra dividido en bloques como: bloque de alimentación, bloque adquisición de imagen, bloque de preprocesamiento, bloque de entrenamiento, bloque de clasificación y bloque de visualización. A continuación, se define el comportamiento de cada uno.

- **Bloque de alimentación:** establece el modo de alimentación del sistema electrónico específicamente de la cámara.
- **Bloque de adquisición:** sección donde se adquiere una imagen del cultivo de lechuga, mediante el uso de dos cámaras ubicadas en una estructura fija a una distancia aproximada de 2-3m de altura. Una vez adquiridas las imágenes; éstas son cargadas a la nube de Ezviz mediante conexión Wi-Fi.
- **Bloque de entrenamiento:** Para el entrenamiento de la red se utilizan los servidores "Colab" de Google por su gran capacidad de procesamiento, además permite optimizar recursos. Esta acción se realiza una sola vez, de manera que sus resultados serán utilizados posteriormente en el proceso de clasificación de cultivo apto y no apto en la placa embebida.

  En la etapa de entrenamiento se realizan varios procesos explicados a continuación:

  - **Pre procesamiento:** en esta etapa se realiza la extracción de imágenes individuales de lechuga, seguido se realizan transformaciones geométricas a las imágenes como función espejo, rotación y zoom. También se realiza la detección de bordes y etiquetado, y se obtiene el Dataset para el entrenamiento, testeo y validación de la red neuronal.
  - **Selección de datos:** al Dataset obtenido en la sección de preprocesamiento almacenada en Google Drive se divide

obteniendo así un porcentaje asignado para el entrenamiento, testeo y validación de la red neuronal siguiendo la regla de 70%, 15%, 15% respectivamente.

- **Bloque de Convolución:** se realizan los procesos necesarios de una red neuronal convolucional que permitan extraer los atributos y características de la imagen. Se hace uso de la arquitectura de red neuronal Inception V3 mediante transferencia de aprendizaje y se valida el modelo pre entrenado evaluando métricas de pérdida y exactitud.

- **Bloque de clasificación:** una vez obtenido el modelo pre entrenado, éste se carga en la placa embebida para realizar la clasificación de cultivo apto y no.
- **Bloque de visualización:** se trata de una interfaz gráfica para el usuario, la cual permite mostrar y almacenar resultados de la clasificación de cultivo apto y no apto para la cosecha.

3.4.2. Diagrama de flujo

En esta sección se describen los diagramas de flujo del proceso de preprocesamiento de la imagen, entrenamiento de la red y clasificación respectivamente.

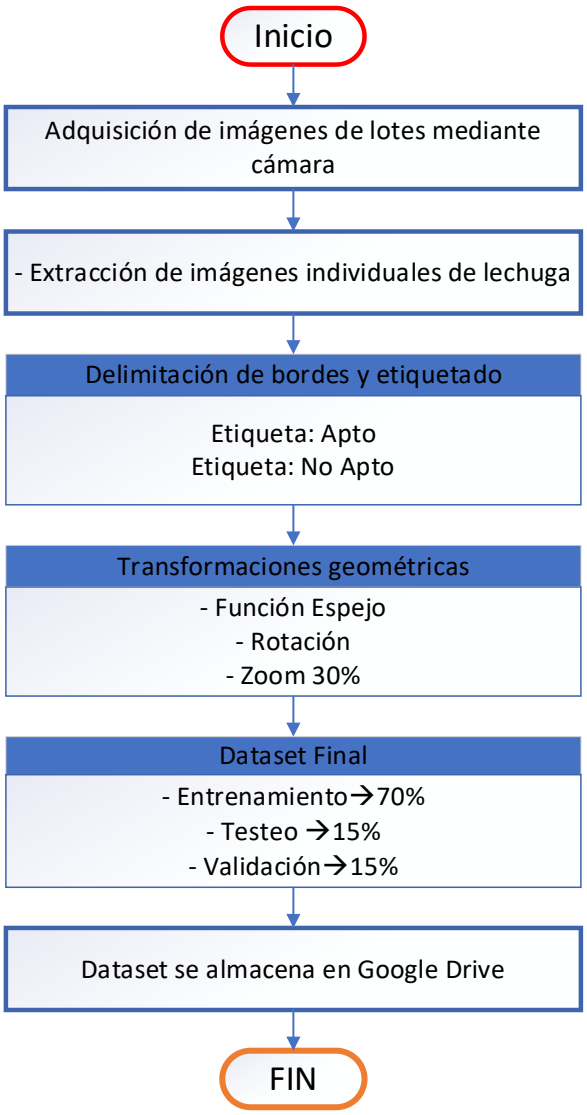
3.4.2.1. Flujograma Preprocesamiento de imágenes

El diagrama de flujo de la figura 17, se inicializa con la adquisición de imágenes de los lotes propuestos mediante una cámara, éstas son almacenadas en la nube de Ezviz mediante conexión Wi-Fi, se realiza la extracción de las imágenes individuales de lechuga, luego se realizan transformaciones geométricas a las imágenes, aplicando cambios de posición, rotación, zoom. Posteriormente se realiza la delimitación de bordes y etiquetado haciendo uso del software labelImg y dos clases: apta y no apta, obteniendo

un dataset más amplio para el entrenamiento de la red y validación del sistema, finalmente, este dataset es almacenado en Google Drive.

Figura 17

Flujograma de preprocesamiento de imágenes

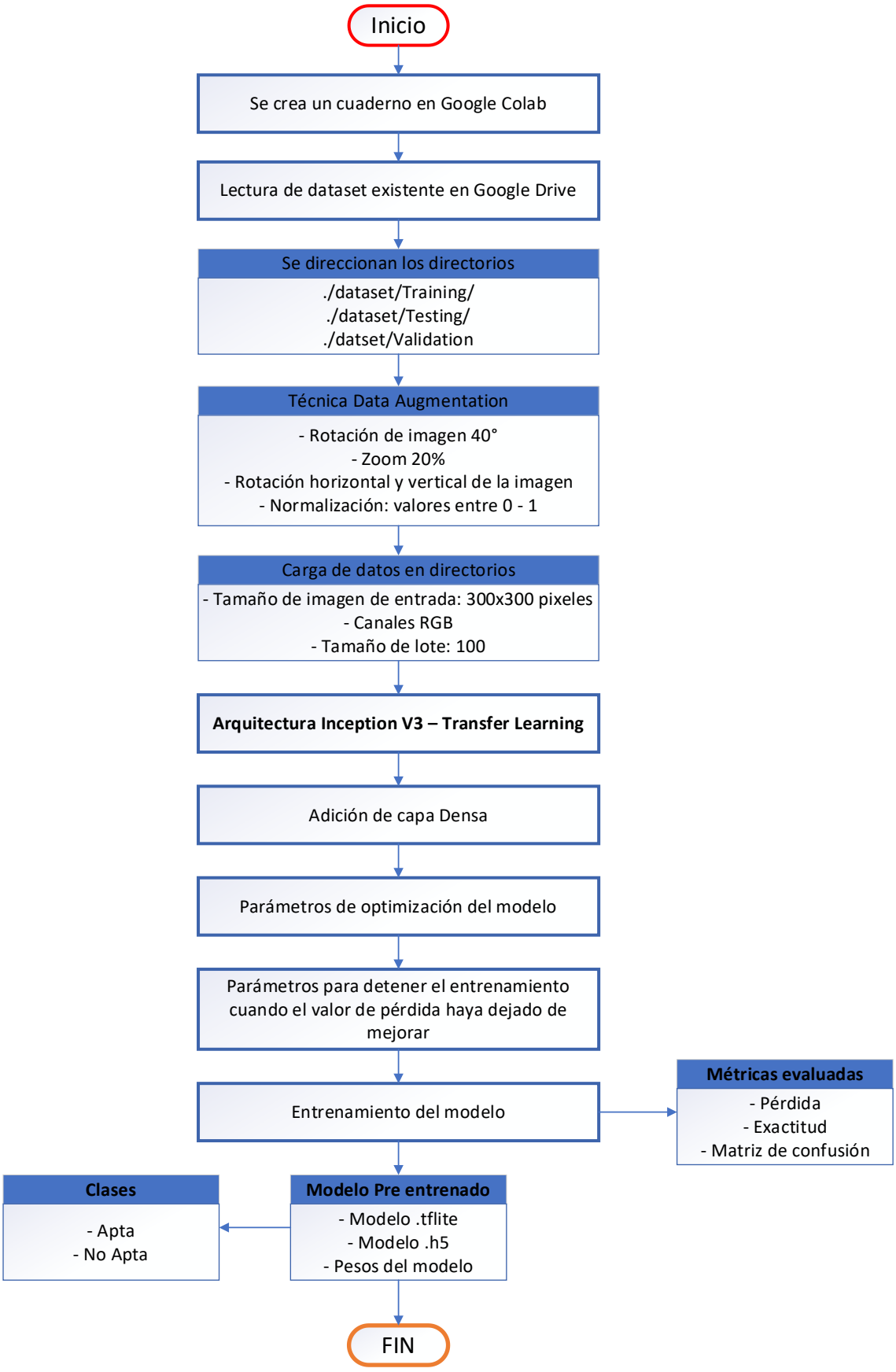


3.4.2.2. Flujograma del entrenamiento de la red

Para el entrenamiento de la red se utilizan los servidores “Colab” de Google, en donde se realizan los procesos de convolución para obtener el mapa de detección de caracteres haciendo uso de una arquitectura de red neuronal Inception V3. En este proceso se extraen dos clases, “lechuga apta” y “lechuga no apta”, consiguiendo así el modelo pre entrenado.

Figura 18

Diagrama de flujo del entrenamiento de la red neuronal



3.4.2.2. Flujograma de integración del sistema

Una vez obtenido el modelo pre entrenado, se lo integra en el sistema embebido NVIDIA Jetson Nano, en el cual se importa el archivo del modelo pre- entrenado que contiene el mapa de caracteres con etiquetas y la configuración de las capas de entrenamiento.

A continuación, se importan las librerías necesarias para trabajar con aplicaciones de visión artificial. Estas librerías y sus funcionalidades se encuentran especificados en la tabla 12.

Posteriormente se realiza el tratamiento de la imagen a reconocer, se establecen las dimensiones de 300 x 300 pixels, se convierte la imagen en un arreglo y se redimensiona la escala de la imagen en 1/255, normalizando la imagen de modo que los valores de los pixeles que oscilan entre 0 y 256 se conviertan en valores entre 0 y 1, lo que hace el cálculo más fácil y rápido. A continuación, se extraen las operaciones del entrenamiento y se clasifica a la lechuga en apta y no apta, para luego almacenar esta imagen en un directorio.

En la figura 19 se muestra el diagrama de flujo del proceso antes mencionado para obtener una mejor comprensión.

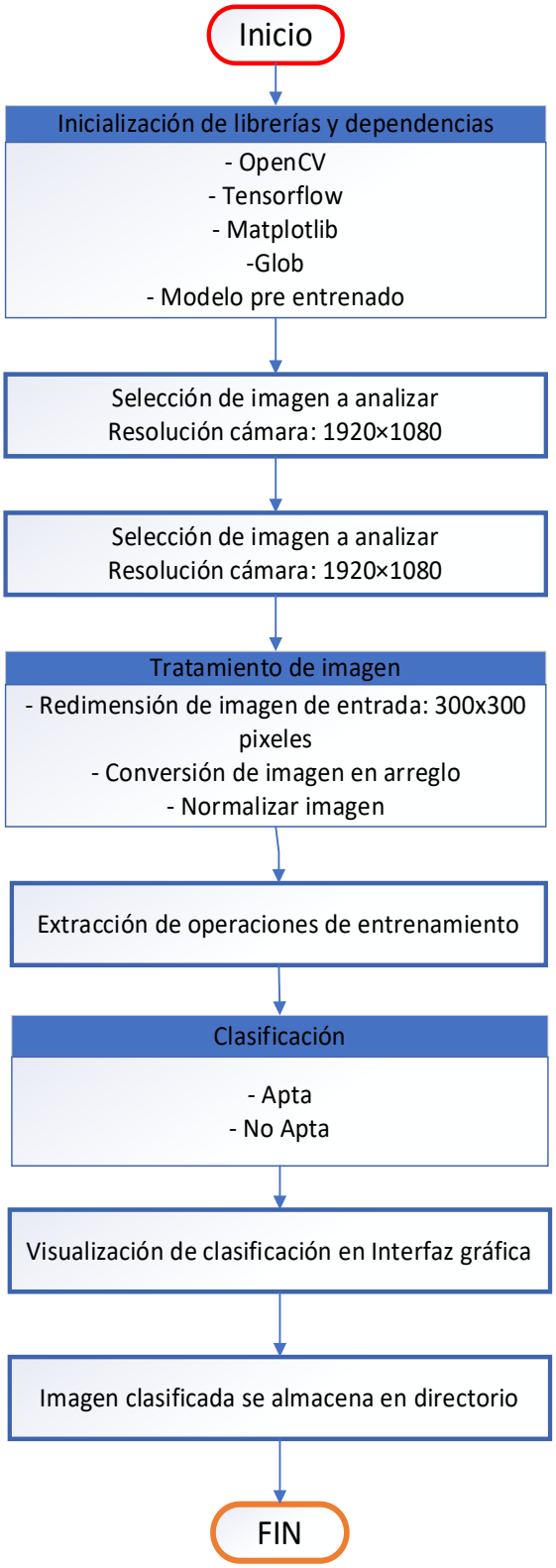
Tabla 12

Librerías usadas

Librerías usadas	
Nombre	**Funcionalidad**
CV2	Librería de visión artificial, permite desarrollar aplicaciones de procesamiento y análisis de imágenes.
Tensorflow	Permite crear y entrenar redes neuronales buscando patrones**.**
Numpy	Librería de código numérico, cuenta con herramientas de cálculos matemáticos e incluye funciones como transformada de Fourier, álgebra lineal, además permite implementar arreglos y matrices.
Matplotlib	Permite crear gráficos personalizados en dos dimensiones como diagramas de barras, tomando como entrada listas o arreglos.
Keras	Biblioteca de aprendizaje automático se utiliza para desarrollar redes neuronales profundas, ya que incluye capas que permiten entrenar y desentrenar modelos de redes neuronales usando código no muy extenso.
Os	Módulo que permite acceder a funcionalidades del Sistema Operativo, provee información sobre el entorno y facilita el manejo de directorios, permitiendo leer y escribir archivos.
Glob	Permite encontrar rutas que coincidan con un patrón especificado.

Figura 19

Flujograma clasificación de lote afectado



3.4.4. Descripción de escenarios

Previo a la implementación del "Sistema Inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de Visión Artificial" se han establecido escenarios que serán objeto de prueba con el fin de verificar la funcionalidad de dicho sistema en un ambiente de producción controlado para identificar posibles lotes afectados como se muestra en la figura 20 y 21.

Figura 20

Escenarios de prueba

Para obtener una óptima producción de lechuga es necesario que el terreno cumpla con ciertos requerimientos edafoclimáticos que aportan al crecimiento y desarrollo de la planta, además de mantener el control de las condiciones hídricas; los escenarios planteados se diferencian en el cumplimiento de dichos requerimientos, de forma que se obtienen escenarios en condiciones aptas y no aptas para el cultivo de lechuga crespa.

El cultivo de lechuga no requiere rigurosos cuidados, la plantación se adapta muy bien a cualquier tipo de terreno y clima; tiene un mejor desarrollo en temperaturas que oscilan entre los 15 y 18°, requiere suelos ricos en nutrientes, ligeros y arenosos, libre de malezas y rocas.

En cuanto a la humedad del cultivo, no soporta largos periodos de sequía y es sensible a la falta de humedad. Para las que las raíces no se estresen es necesario mantener un nivel moderado de humedad.

El adecuado manejo del riego del cultivo determina la calidad de la producción, (Antúnez B., Felmer E., & Vidal S. , 2019) realizaron una evaluación de distintos métodos de riego, entre ellos riego por aspersión, riego por goteo y surcos, siendo el riego por goteo el de mayor eficiencia; mediante este método el agua puede ser distribuida en cantidades de $6mm^3$ diarios en meses frescos y $10mm^3$ en meses cálidos, los intervalos de aplicación son de dos a tres días debido a que son volúmenes de agua muy pequeños.

En este sentido, para llevar a cabo las pruebas se establecen dos escenarios de dimensiones de 2m x 2m, con una cantidad de 35 lechugas cada lote, estas hortalizas se encuentran sembradas con una separación de 30 cm y ubicadas hileras de 7 filas por 5 columnas. El escenario 1, mostrado en la figura 21 a) cuenta con un terreno preparado, abonado y con un regadío periódico. El escenario 2, de la figura 21 b) tiene un terreno más seco, sin abono, y con un regadío esporádico.

Figura 21

Escenarios de prueba

a) Escenario Apto

b) Escenario No Apto

3.5. Desarrollo del software

En esta sección se desarrolla las etapas que conforman el software del sistema de visión artificial. El desarrollo inicia con el procesamiento de las imágenes obtenidas en las cámaras, de modo que se obtenga imágenes individuales de lechugas, posteriormente se realiza un aumento de datos aplicando transformaciones geométricas como rotación, función espejo y zoom. Luego a estas imágenes se delimitan los bordes y etiqueta para obtener la base de datos para el entrenamiento de la red, seguido se realiza el entrenamiento de la red neuronal y la obtención del modelo pre entrenado, posteriormente se realiza la integración del modelo pre entrenado en la placa embebida Nvidia Jetson Nano, mediante un script de modo que al ejecutarlo se obtiene una clasificación de lechuga apta y no apta.

3.5.1 Preprocesamiento de imágenes

En este apartado se abordan los procesos ejecutados para la obtención de dataset para alimentar y entrenar a la red neuronal. Dentro de estos procesos se destacan: el

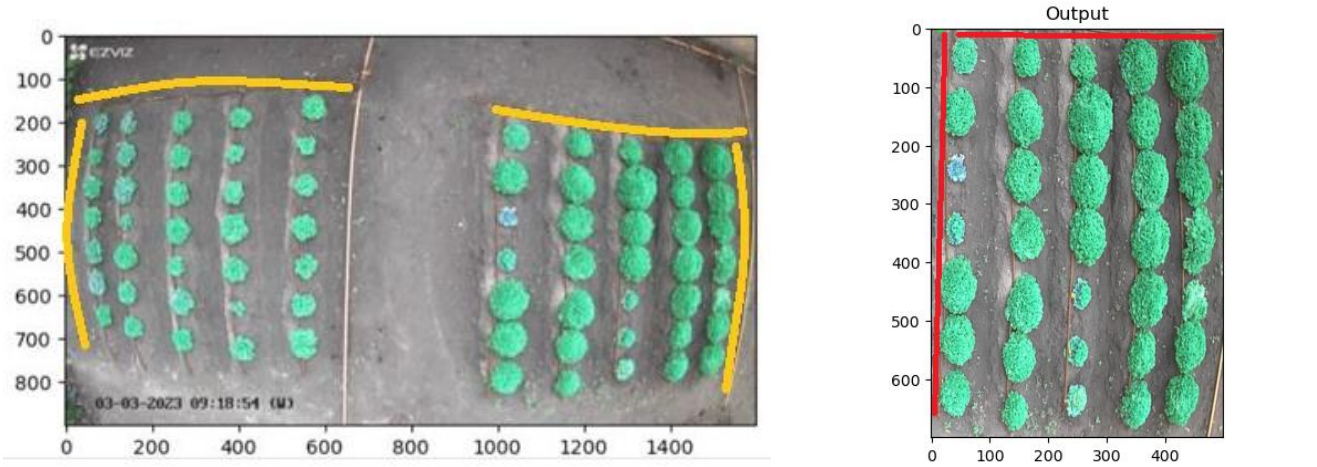
procesamiento de los lotes y extracción de imágenes individuales, el aumento de datos aplicando transformaciones geométricas y la delimitación los bordes y etiquetado.

3.5.1.1. Procesamiento de lotes y extracción de imágenes individuales

Previo a entrenar la red neuronal, se realiza el preprocesamiento de las imágenes obtenidas de las cámaras, para lo cual se aplica una transformación de perspectiva, que permite que la imagen distorsionada o poco visible como muestra la figura 22 a) mejore y se alinee eliminando curvas como se muestra en la figura 22 b) para obtener una mejor comprensión de la información de los lotes 1 y 2.

Figura 22

Captura de imagen lotes 1 y 2



b) Imagen de lote 1 y 2, con curvas

a) Imagen con transformación de perspectiva y eliminación de curvas

Para aplicar esta transformación, se establecen las coordenadas de la imagen que se quiere delimitar tomando en cuenta que la información a recopilar debe estar dentro de estos límites para los lotes 1 y 2, como se muestra en las figuras 23 y 24.

Figura 23

Delimitación de coordenadas lote 1

```
# arriba izquierda verde
lote = cv.imread("lote1.jpg")
print(lote.shape)
lote[60:70,30:40]=[0,255,0]
# abajo izquierda azul
lote[600:610,30:40]=[0,0,255]
# arriba derecha rojo
lote[0:10,630:640]=[255,0,0]
# abajo derecha blanco
lote[690:700,630:640]=[255,255,255]
plt.imshow(lote), plt.title("Input")
```

```
(700, 640, 3)
(<matplotlib.image.AxesImage at 0x19cf2af4220>, Text(0.
```

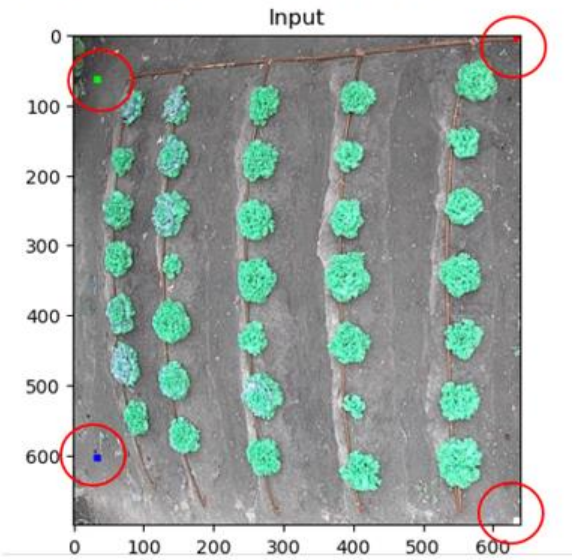
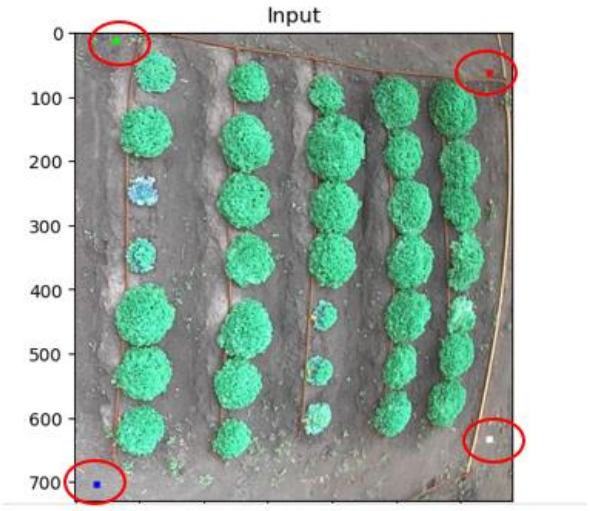


Figura 24

Delimitación de coordenadas lote 2

```
# arriba izquierda verde
lote = cv.imread("lote2.jpg")
print(lote.shape)
lote[10:20,60:70]=[0,255,0]
# abajo izquierda azul
lote[700:710,30:40]=[0,0,255]
# arriba derecha rojo
lote[60:70,640:650]=[255,0,0]
# abajo derecha blanco
lote[630:640,640:650]=[255,255,255]
plt.imshow(lote), plt.title("Input")
```

```
(730, 680, 3)
(<matplotlib.image.AxesImage at 0x2726995d040>, Text(0.5,
```



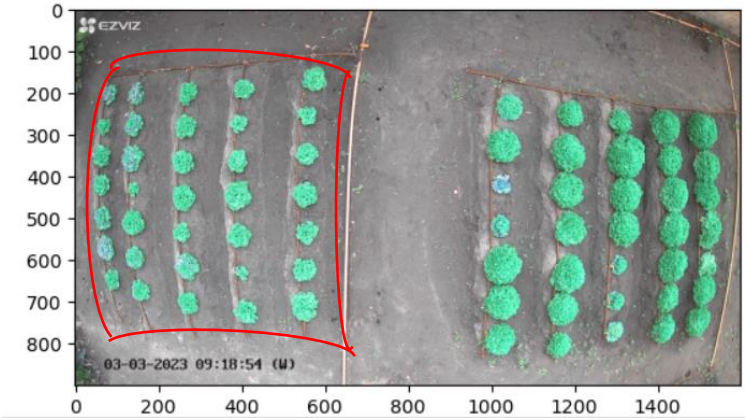
A continuación, para aplicar la transformación de perspectiva se establecen los puntos de la imagen de origen mostrada en la figura 25 a) de la cual se van a recopilar los datos y los puntos que enmarcan la imagen de salida mostrada en la figura 25 b).

Luego mediante la función *"PerspectiveTransform()"* se obtiene una matriz de perspectiva con los puntos establecidos anteriormente y la función *WarpPerspective()"*

se obtiene la imagen alineada y sin curvas, como se muestra en las figuras 25 y 26 para los lotes 1 y 2 respectivamente.

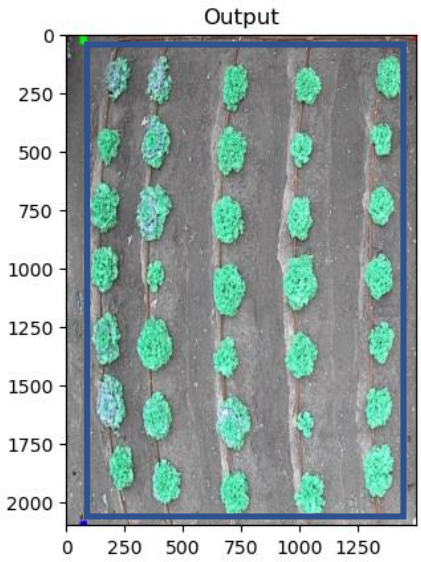
Figura 25

Matriz alineada lote 1



b) Puntos de imagen de entrada (pst1)

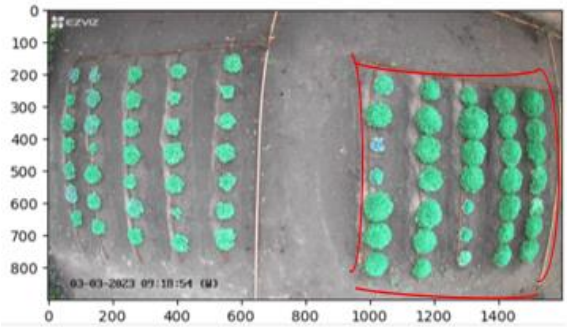
```
pts1 = np.float32([[10,60],[10,600],[640-1,0],[640-1,700-1]])
pts2 = np.float32([[0,0],[0,2100-1],[1500-1,0],[1500-1,2100-1]])
matriz = cv.getPerspectiveTransform(pts1,pts2)
dst = cv.warpPerspective(lote,matriz,(1500,2100))
plt.imshow(dst), plt.title("Output")
```



a) Puntos de imagen de salida (pst2)

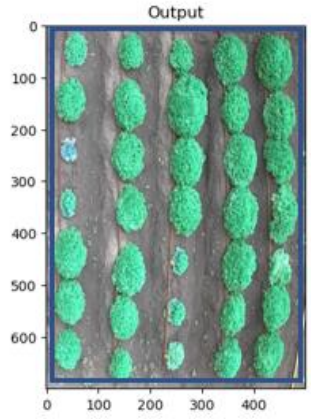
Figura 26

Matriz alineada lote 2



a) Puntos de imagen de entrada (pst1)

```
pts1 = np.float32([[40,10],[40,700],[650,70],[650,620]])
pts2 = np.float32([[0,0],[0,700-1],[500-1,0],[500-1,700-1]])
matriz = cv.getPerspectiveTransform(pts1,pts2)
dst = cv.warpPerspective(lote,matriz,(500,700))
plt.imshow(dst), plt.title("Output")
```



b) Puntos de imagen de salida (pst2)

Finalmente, se obtiene la posición de cada objeto que forma la matriz de 7x5, y se extrae las imágenes que corresponde a cada punto de coordenadas x, y, como se muestra en la figura 27 a) y 27 b) respectivamente.

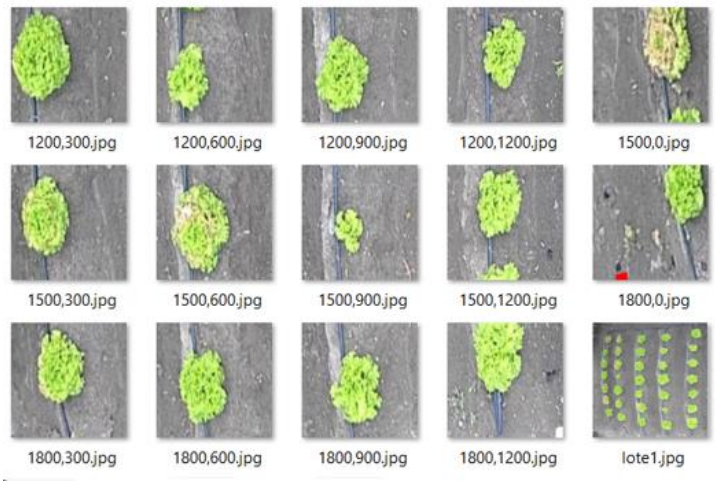
Figura 27

Extracción de coordenadas de cada objeto de matriz 7x5

```
for pos_y in range(0,2100,300):
    for pos_x in range(0,1500,300):
        print(str(pos_y)+","+str(pos_x)+"  ", end="")
    print("\n")
```

```
0,0  0,300  0,600  0,900  0,1200

300,0  300,300  300,600  300,900  300,1200

600,0  600,300  600,600  600,900  600,1200

900,0  900,300  900,600  900,900  900,1200

1200,0  1200,300  1200,600  1200,900  1200,1200

1500,0  1500,300  1500,600  1500,900  1500,1200

1800,0  1800,300  1800,600  1800,900  1800,1200
```

a) Coordenadas x,y de cada objeto de la matriz de 7x5



b) Imágenes individuales extraídas

3.5.1.2. Aumento de datos aplicando transformaciones geométricas

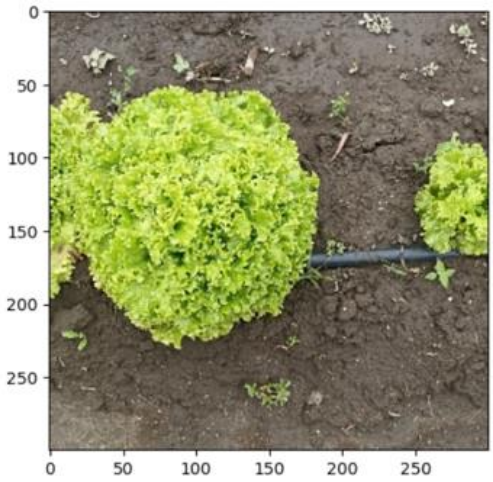
Se procede a realizar el aumento de datos aplicando transformaciones geométricas a las imágenes, manejando cambios de posición, rotación, zoom.

La primera transformación aplicada es la función espejo, esta permite girar una imagen horizontal o verticalmente, para lo cual primero se importan las librerías a utilizar, seguido se hace una búsqueda de las imágenes en el directorio *"path_work"* y se carga la imagen para ser leída y redimensionada a un tamaño de 300x300 pixeles. Se definen los tres puntos en la imagen de entrada y se asignan los tres puntos correspondientes en la imagen de salida para encontrar la matriz de transformación de 2x3, esta matriz se calcula usando la función *"getAffineTransform( )"* y mediante la función *"warpAffine( )*" se obtiene la imagen transformada, tal y como se muestra en la figura 28.

Figura 28

Transformación geométrica de la imagen – Función espejo

```
import glob
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

imagenes =  glob.glob(path_work)
path=""
for path in imagenes:
    img = cv.imread(path)
    img = cv.resize(img, (300,300))
    #img = cv.cvtColor(img, cv.COLOR_BGR2RGB)

    filas,cols, _ = img.shape
    # punto de entrada
    pt1 = np.float32([[0,0],[cols-1,0],[0,filas-1]])
    # puntos de salida
    pt2 = np.float32([[cols-1,0],[0,0],[cols-1,filas-1]])
    # creando matri de transformacion
    m = cv.getAffineTransform(pt1,pt2)
    # transformando la imagen
    dst = cv.warpAffine(img,m,(cols,filas))

    path = path.replace(".jpg","_espejo.jpg")
    path = path.replace("\\","/")
    cv.imwrite(path,dst)
print(path)
img = cv.imread(path)
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)
```



La segunda transformación aplicada es la función rotación. Se aplican tres rotaciones, de 90°, 180° y de 270° como se muestran en las figuras 29, 30 y 31, respectivamente.

Para esta transformación, primero se importan las librerías a utilizar, seguido se hace una búsqueda de las imágenes en el directorio *"path_work"* y se carga la imagen para ser leída y redimensionada a un tamaño de 300x300 pixeles. Luego, para obtener la matriz de rotación se aplica la función "*getRotationMatrix2D( )"* con los parámetros del centro de rotación de la imagen, el ángulo de rotación en grados, y el factor de escala. Mediante la función *"warpAffine( )*" se obtiene la imagen transformada.

Figura 29

Transformación geométrica de la imagen – Rotación 90°

```
import glob
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

imagenes =  glob.glob(path_work)
path=""

for path in imagenes:
    img = cv.imread(path)
    img = cv.resize(img, (300,300))
    # girar 90 grados
    rows, cols, _ = img.shape
    #cols -1 y rows -1 son los limites de las coordenadas para calcular
    m = cv.getRotationMatrix2D(((cols-1)/2.0,(rows-1)/2.0), 90, 1)
    # transformacion de la matriz, giro de 90 grados
    dst = cv.warpAffine(img, m, (cols,rows))


    path = path.replace(".jpg","_rot90.jpg")
    path = path.replace("\\","/")
    cv.imwrite(path,dst)

print(path)
img = cv.imread(path)
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)
```

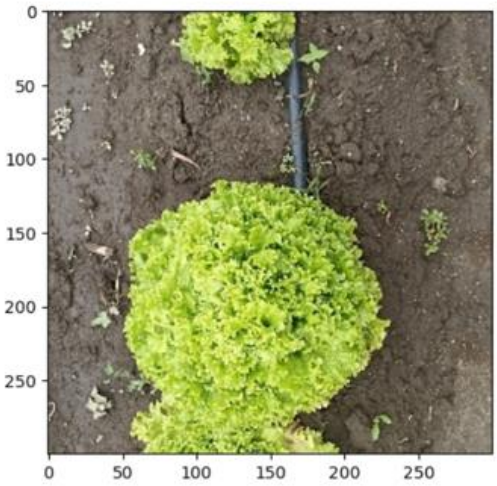


Figura 30

Transformación geométrica de la imagen – Rotación 180°

```
import glob
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

imagenes =  glob.glob(path_work)
path=""

for path in imagenes:
    img = cv.imread(path)
    img = cv.resize(img, (300,300))
    # girar 180 grados
    rows, cols, _ = img.shape
    #cols -1 y rows -1 son los limites de las coordenadas para calcular
    m = cv.getRotationMatrix2D(((cols-1)/2.0,(rows-1)/2.0), 180, 1)
    # transformacion de la matriz, giro de 180 grados
    dst = cv.warpAffine(img, m, (cols,rows))


    path = path.replace(".jpg","_rot180.jpg")
    path = path.replace("\\","/")
    cv.imwrite(path,dst)

print(path)
img = cv.imread(path)
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)
```

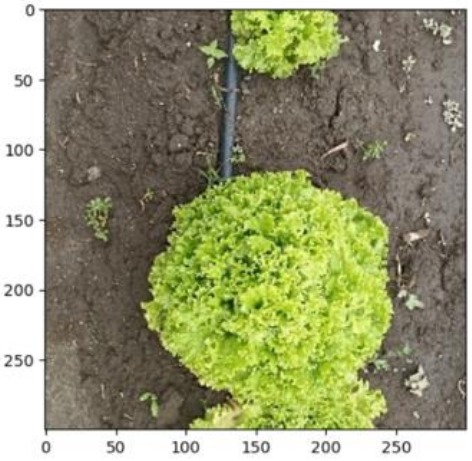
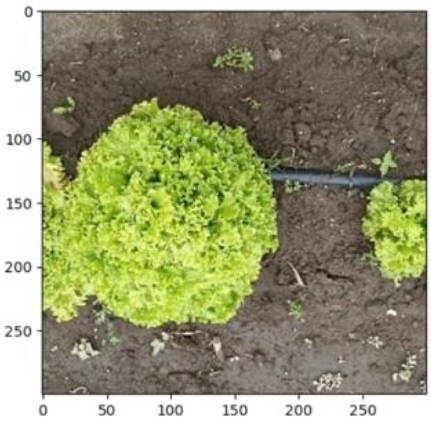


Figura 31

Transformación geométrica de la imagen – Rotación 270°

```
import glob
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

imagenes =  glob.glob(path_work)
path=""

for path in imagenes:
    img = cv.imread(path)
    img = cv.resize(img, (300,300))
    # girar 270 grados
    rows, cols, _ = img.shape
    #cols -1 y rows -1 son los limites de las coordenadas para calcular
    m = cv.getRotationMatrix2D(((cols-1)/2.0,(rows-1)/2.0), 270, 1)
    # transformacion de la matriz, giro de 270 grados
    dst = cv.warpAffine(img, m, (cols,rows))


    path = path.replace(".jpg","_rot270.jpg")
    path = path.replace("\\","/")
    cv.imwrite(path,dst)

print(path)
img = cv.imread(path)
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)
```

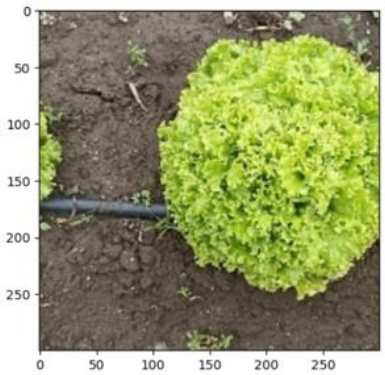


La última transformación aplicada es la función de zoom 30% mostrada en la figura 32. Primero se importan las librerías a utilizar, seguido se hace una búsqueda de las imágenes en el directorio *"path_work"* y se carga la imagen para ser leída y redimensionada a un tamaño de 300x300 pixeles. Luego, se aplica la función "*getRotationMatrix2D( )"* manteniendo los parámetros del centro de rotación de la imagen; el ángulo se establece en 0 y el factor de escala en 1.3 y se obtiene la imagen transformada mediante la función *"warpAffine( )*".

Figura 32

Transformación geométrica de la imagen – Zoom 30%

```
import glob
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

imagenes =  glob.glob(path_work)
path=""

for path in imagenes:
    img = cv.imread(path)
    img = cv.resize(img, (300,300))
    # girar 90 grados
    rows, cols, _ = img.shape
    #cols -1 y rows -1 son los limites de las coordenadas para calcular
    m = cv.getRotationMatrix2D(((cols-1)/2.0,(rows-1)/2.0), 0, 1.3)
    # transformacion de la matriz, giro de 90 grados
    dst = cv.warpAffine(img, m, (cols,rows))


    path = path.replace(".jpg","_zoom30.jpg")
    path = path.replace("\\","/")
    cv.imwrite(path,dst)

print(path)
img = cv.imread(path)
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img)
```



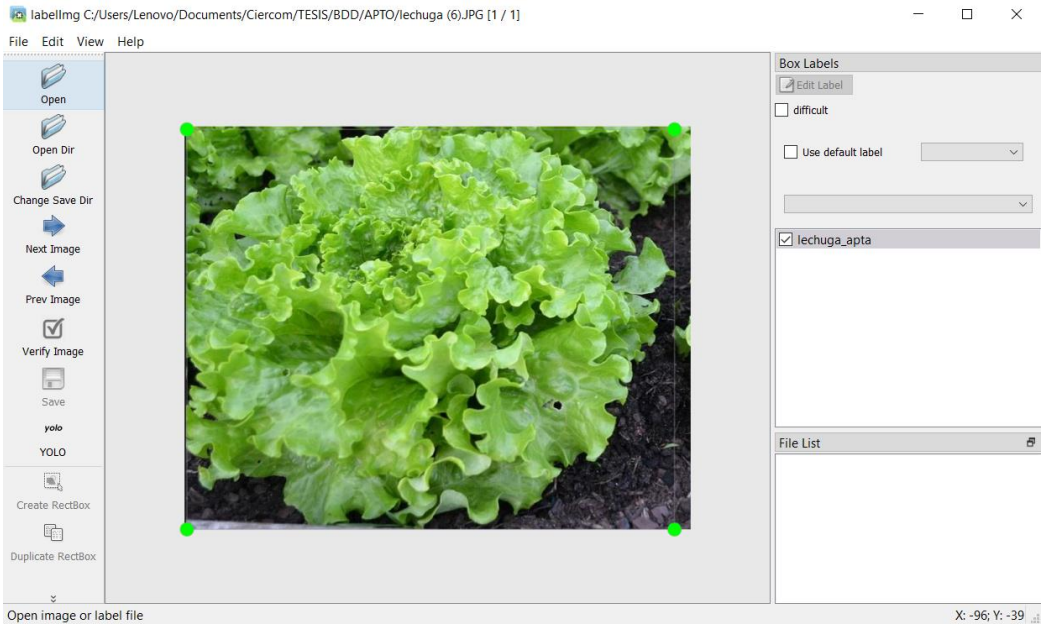
3.5.1.3. Delimitación de bordes y etiquetado

Dentro de la etapa de Pre procesamiento se realizan dos actividades importantes, que son la detección de bordes y el etiquetado de imágenes. La primera acción permite delimitar el objeto en un recuadro y la segunda acción establece una o varias etiquetas de la clase que se desea identificar. Ambos procedimientos se realizan mediante la aplicación LabelImg, obteniendo archivos en formatos como VOC, XML o Yolo.

En la figura 33 se observan una imagen de lechuga, delimitada mediante un recuadro. En el panel de la derecha *"Box Labels"* se encuentran las dos etiquetas creadas en este caso "*lechuga apta*" y "*lechuga no apta"*.

Figura 33

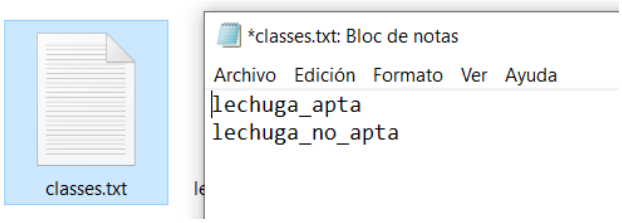
Detección de bordes y etiquetado



Terminado el proceso de etiquetar las imágenes, el software ofrece dos archivos de texto, uno nombrado *"classes"* en el cual se encuentran las etiquetas creadas como se muestra en la figura 34.

Figura 34

Archivo de texto de etiquetas creadas



El segundo archivo de texto que se genera conserva el mismo nombre que la imagen etiquetada y contiene las etiquetas elaboradas en la imagen.

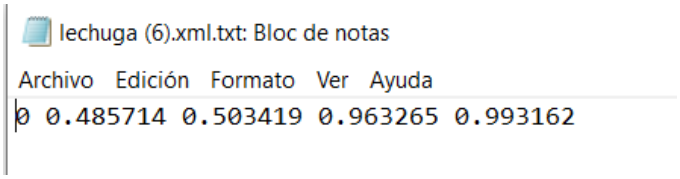
Figura 35

Archivo de etiqueta por imagen

En la figura 36 se muestra el archivo de etiqueta de la imagen “6”, cuyo nombre es lechuga(6).txt, en el archivo de texto se observan cinco columnas, la primera “0” pertenece a la etiqueta en este caso “lechuga_apta” y las columnas restantes son las coordenadas de los cuatro puntos del recuadro que encierran al objeto.

Figura 36

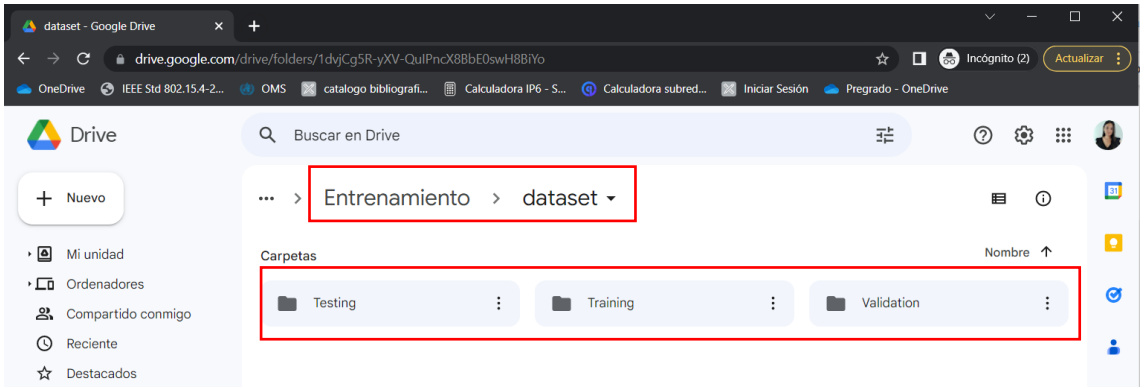
Contenido de la etiqueta



Todas las imágenes y etiquetas son subidas a Google Drive para su uso posterior en el proceso de entrenamiento de la red. El dataset se divide en 70% para entrenamiento, 15% para testeo y 15% para validación como se muestra en a figura 37.

Figura 37

Dataset para entrenamiento de la red neuronal en Google Drive



3.5.2 Entrenamiento de la red neuronal

El entrenamiento se lo realiza en el software Google Colab, el cual permite ejecutar código Python en el navegador, y es adecuado para tareas de aprendizaje automático y análisis de datos sin coste adicional.

Como primer paso es necesario cargar el dataset existente en Google Drive, como se muestra en la figura 38.

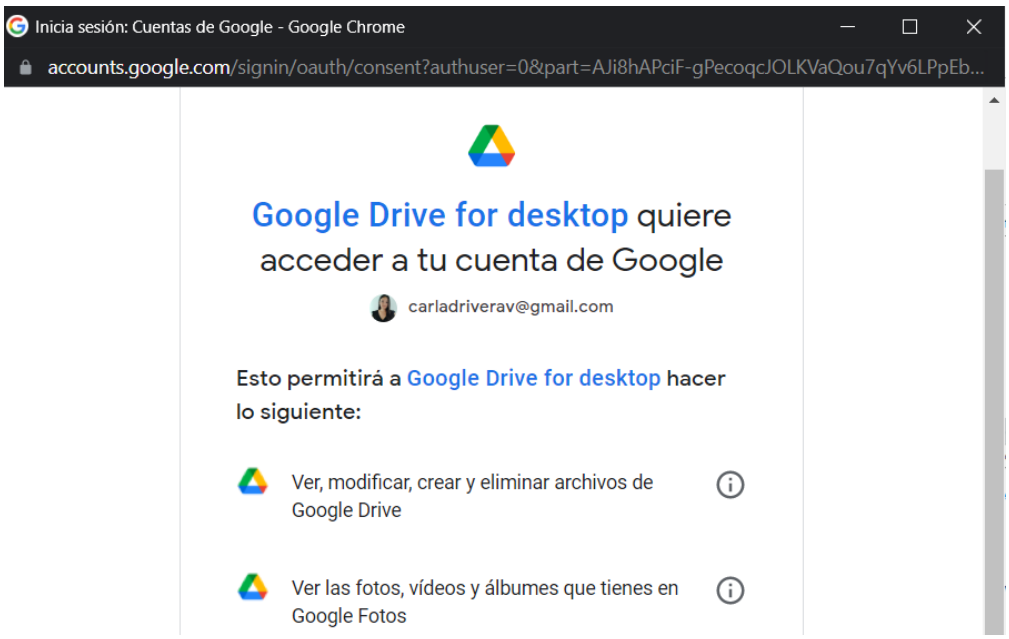
Figura 38

Sincronización de Google Drive con Google Colab

En la figura 39 se muestran los permisos que son necesarios para el uso de Google Drive. Con esto será posible crear y modificar archivos desde Google Drive.

Figura 39

Permisos de uso Google Drive



A continuación, se direccionan los directorios, de forma que exista un directorio para pruebas, test y validación, como se muestra en la figura 40.

Figura 40

Creación de directorios

```
import os
train_dir = "./Dataset/Training/"
test_dir = "./Dataset/Testing/"
val_dir = "./Dataset/Validation/"

train_aptas = os.path.join(train_dir,"Aptas/")
train_no_aptas = os.path.join(train_dir,"No_Aptas/")
```

Posteriormente, se importa el módulo ImageDataGenerator para realizar aumento de datos, es posible ajustar los datos de entrenamiento, test y validación mediante un ajuste de dimensiones de imágenes, rotaciones, cambios de escala, como se muestra en la figura 41.

Figura 41

Módulo ImageDataGenerator

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(#rotation_range=40, # rota las imagenes 40 grados
                                   zoom_range=0.2,  # zoom del 20€
                                   horizontal_flip=True, ## voltear la imagen de forma horizontal
                                   vertical_flip=True,
                                   rescale=1./255) # rescala la imagen de 0-255 a 0-1

test_datagen = ImageDataGenerator(#rotation_range=40, # rota las imagenes 40 grados
                                   zoom_range=0.1,  # zoom del 20€
                                   horizontal_flip=True, ## voltear la imagen de forma horizontal
                                   vertical_flip=True,
                                   rescale=1./255) # rescala la imagen de 0-255 a 0-1

val_datagen = ImageDataGenerator(#rotation_range=40, # rota las imagenes 40 grados
                                   zoom_range=0.2,  # zoom del 20€
                                   horizontal_flip=True, ## voltear la imagen de forma horizontal
                                   vertical_flip=True,
                                   rescale=1./255) # rescala la imagen de 0-255 a 0-1
```

Luego, los datos generados en la figura 42, son cargados a las variables *"train_generator", "test_generator"* y *"val_generator"* y se especifican las dimensiones que tomarán las imágenes encontradas en este caso de 300x300 pixeles, en el modo de

clase se especifica el uso de “categorical”, de esta forma los datos se codifican en números antes de que puedan ser usados para ajustes y evaluación del modelo y se especifica el tamaño de los lotes de datos en este caso 100, como se muestra en la figura 43.

Figura 42

Carga de datos en variables

```
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size=(300,300),
                                                    class_mode='categorical',
                                                    batch_size=100)

test_generator = test_datagen.flow_from_directory(test_dir,
                                                  target_size=(300,300),
                                                  class_mode='categorical',
                                                  batch_size=100)

val_generator = val_datagen.flow_from_directory(val_dir,
                                                target_size=(300,300),
                                                class_mode='categorical',
                                                batch_size=100)
```

Para la construcción del modelo de entrenamiento se ha escogido la arquitectura de red neuronal convolucional Inception V3, haciendo uso de transferencia de aprendizaje con pesos preentrenados en 0 y agregando nuevas capas densas es posible llevar a cabo la tarea de clasificación en dos grupos.

Para hacer uso de esta arquitectura primero se importa el modelo Inception V3 y las capas, luego se descargan los pesos pre entrenados del modelo y se los guarda en la variable “*local_weighs_file*”. Posteriormente se configuran las dimensiones de los datos de entrada a un tamaño de 300 x300 pixeles en 3 canales RGB y se eliminan las capas densas como se muestra en la figura 43. Por último, se congelan las capas previamente entrenadas, de este modo se usará únicamente para la extracción de características y se evita destruir la información en futuros entrenamientos.

Figura 43

Importación de la arquitectura Inception V3

```
#https://storage.googleapis.com/mledu-datasets/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5

from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras import layers

local_weights_file = './inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

#configurando la forma de entrada y removiendo las capas densas.
pre_trained_model = InceptionV3(include_top = False, # Incluye o elimina las capas densas
                                input_shape = (300,300, 3), # forma de la entrada
                                weights = None)

## Cargar los pesos preentrenados del modelo descargado
pre_trained_model.load_weights(local_weights_file)

for layer in pre_trained_model.layers:
    layer.trainable = False
```

La capa que será usada para la extracción de características se llama Mixed7. Se usa para mantener un mapa de características suficientemente grande en este caso 17x17x768 como se muestra en la figura 44.

Figura 44

Forma de salida de la última capa

```
last_layer = pre_trained_model.get_layer('mixed7')
print('forma de salida de la ultima capa: ', last_layer.output_shape)
last_output = last_layer.output
```

```
forma de salida de la ultima capa:  (None, 17, 17, 768)
```

Se realiza el aumento de capas densas, para lo cual se importa el tipo de optimizador a usar para reducir pérdidas y aumentar el porcentaje de éxito durante el entrenamiento. Además, se importan las librerías Model y layers, esto permite agrupar capas en un objeto con características de entrenamiento.

Se añade la capa Flatten, la cual permite aplanar una matriz que proviene de capas convolucionales y hacerla unidimensional, es decir convierte una imagen de dos o tres dimensiones en una sola. Luego se añade una capa Densa de 1024 neuronas y funciones

de activación “Relu” para eliminar valores negativos. Después se tiene una capa de Dropout o capa de regularización, se usa para reducir el sobreajuste o sobre entrenamiento. Para ello se apaga una parte aleatoria de la red neuronal, en este caso se encuentra establecido en 0.4. Por último, se tiene una capa de salida con dos neuronas y funcion de activación “Softmax” para calcular probabilidades de que una imagen pertenece a una clase determinada en este caso existen dos clases, aptas y no aptas, como se muestra en la figura 45.

Figura 45

Aumento de capa densa

```
from tensorflow.keras.optimizers import RMSprop
from tensorflow. keras import Model
from tensorflow.keras import layers

# Capa de aplanado de una dimension
x = layers.Flatten()(last_output)
# Capa densa(full conectada con 1024 neuronas y activacion relu)
x = layers.Dense(units=1024, activation='relu')(x)
# Capa de dropout con un ratio de 0.4
x = layers.Dropout(rate=0.4)(x)
# Capa de salida con una neurona y activacion sigmoide para modelos de clasificacion
x = layers.Dense(units=2, activation='softmax')(x)

# anadimos la red densa al modelo
cnn_model = Model(pre_trained_model.input, x)
```

Una vez que la red neuronal ha sido definida, se crea un modelo de compilación agregando un optimizador el cual determine cuál es la mejor combinación de variables para obtener un mejor resultado, además de métricas de rendimiento.

El modelo de optimización utilizado es Adam, busca resolver problemas en cuatro pasos, primero entiende el problema, establece un plan de solución, lo ejecuta y analiza resultados. Este método permite ejecutar búsquedas en menor tiempo y cantidad de pasos. Para saber qué tan buena es la red se establece la función de pérdida “categórical_crossentropy”. Y las métricas de rendimiento utilizadas son la precisión y la precisión categórica como se presenta en la figura 46.

Figura 46

Parámetros de optimización del modelo

```
from tensorflow.keras.optimizers import Adam
cnn_model.compile(optimizer=Adam(learning_rate=0.0001),
                  loss='categorical_crossentropy',
                  metrics=["accuracy","categorical_accuracy"])
```

A continuación, en la figura 47, se establece una función que permita detener el entrenamiento cuando alguna métrica deje de mejorar, en este caso como el objetivo es minimizar la pérdida, se establecen los siguientes argumentos:

- Función *"EarlyStopping":* parada anticipada
- Métrica evaluada: *"val_loss"* o valor de pérdida.
- Modo*"min_delta",* el entrenamiento se detendrá cuando el valor de pérdida deje de reducir.
- *"patience":* número de épocas sin mejora, al establecer en "10", significa que después de diez épocas sin mejora, el entrenamiento se detendrá.
- *"verbose":* se establece en "1" para que muestre un mensaje.
- *"mode"*: Al establecerse en *min*, el entrenamiento se detendrá cuando la cantidad monitoreada haya dejado de disminuir.
- *"baseline"*: El entrenamiento se detiene si el modelo no muestra una mejora con respecto a los parámetros establecidos.
- *"restore_best_weights"*: establece si es necesario restaurar los pesos del modelo. Al establecerse en Falso, se manejan los pesos alcanzados en el último ciclo del entrenamiento.

Figura 47

Parámetros para detener el entrenamiento

```
import tensorflow as tf
callback=tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0,
    patience=10,
    verbose=1,
    mode="min",
    baseline=None,
    restore_best_weights=True,
)
```

Se procede a realizar el entrenamiento de la red de la figura 48. Para ello se establece el atributo de historial, el cual registra valores de exactitud y pérdida obtenidos a lo largo del entrenamiento de la red neuronal. Los argumentos empleados son:

- "*train_generator*": especifica los datos de entrada.
- *"epochs":* el número de iteraciones de entrenamiento en 500.
- *"steps_per_epoch":* se establece el tamaño de lote en 10
- *"validation_data"***:** datos sobre los que se evalúa las métricas de pérdida y exactitud.
- *"verbose":* se establece en "1" para que muestre un mensaje.
- *"validation_steps"***:** Número total de pasos a dibujar antes de detenerse al realizar la validación.

Figura 48

Compilación de entrenamiento

```
history = cnn_model.fit(train_generator,
                        epochs=500,
                        steps_per_epoch=10,
                        validation_data=test_generator,
                        verbose=1,
                        callbacks=[callback],
                        validation_steps=5,
                        shuffle=True)
```

En la figura 49, se observa que una vez entrenado el modelo la primera iteración se obtuvo algo de suerte y se acertó la mitad de las salidas; en el segundo ciclo el porcentaje de aciertos aumenta en un 30%, a partir del tercer ciclo se obtiene un porcentaje de exactitud sobre el 90% y va en aumento debido a que los pesos han sido ajustados correctamente.

Figura 49

Entrenamiento primeros 25 ciclos

En la figura 50, se observa que a partir del ciclo 85 se obtiene un porcentaje de aciertos del 100% y un valor de pérdida mínimo dentro de un rango de 0.11% y 0.31%. Finalmente, en el ciclo 98 se mantiene el porcentaje de aciertos en 100% y un valor de

pérdida de 0.06%. Además, se ejecuta la función de parada anticipada, esta función detendrá el entrenamiento después de diez épocas sin mejora, de esta forma se evita el sobreajuste del entrenamiento de la red neuronal.

Figura 50

Entrenamiento últimos 14 ciclos

```
Epoch 85/500
10/10 [==============================] - 75s 8s/step - loss: 0.0011 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 5.1335e-04
Epoch 86/500
10/10 [==============================] - 74s 7s/step - loss: 0.0016 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 3.6385e-04
Epoch 87/500
10/10 [==============================] - 75s 8s/step - loss: 0.0015 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 3.2128e-04
Epoch 88/500
10/10 [==============================] - 76s 8s/step - loss: 0.0017 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 2.3220e-04
Epoch 89/500
10/10 [==============================] - 75s 8s/step - loss: 0.0019 - accuracy: 0.9990 - categorical_accuracy: 0.9990 - val_loss: 8.3387e-04
Epoch 90/500
10/10 [==============================] - 77s 8s/step - loss: 0.0031 - accuracy: 0.9990 - categorical_accuracy: 0.9990 - val_loss: 4.4290e-04
Epoch 91/500
10/10 [==============================] - 75s 8s/step - loss: 0.0012 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 4.6500e-04
Epoch 92/500
10/10 [==============================] - 74s 7s/step - loss: 0.0017 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 3.9203e-04
Epoch 93/500
10/10 [==============================] - 74s 8s/step - loss: 9.6233e-04 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 3.4787e
Epoch 94/500
10/10 [==============================] - 75s 8s/step - loss: 0.0016 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 2.3753e-04
Epoch 95/500
10/10 [==============================] - 75s 8s/step - loss: 0.0010 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 2.4624e-04
Epoch 96/500
10/10 [==============================] - 75s 8s/step - loss: 7.5867e-04 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 2.8211e
Epoch 97/500
10/10 [==============================] - 76s 8s/step - loss: 8.5604e-04 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 3.1347e
Epoch 98/500
10/10 [==============================] - ETA: 0s - loss: 6.8595e-04 - accuracy: 1.0000 - categorical_accuracy: 1.0000Restoring model weights
10/10 [==============================] - 76s 8s/step - loss: 6.8595e-04 - accuracy: 1.0000 - categorical_accuracy: 1.0000 - val_loss: 2.6133e
Epoch 98: early stopping
```

Por último, se realiza una validación del modelo, tomando en cuenta las métricas de valor de pérdida, valor de exactitud y exactitud categórica. Obteniendo así, un valor de pérdida de 0.06%, exactitud de 100%, y una exactitud categórica de 100%, valores mostrados en la figura 51.

Figura 51

Validación del modelo

```
cnn_model.metrics_names
```

```
['loss', 'accuracy', 'categorical_accuracy']
```

```
cnn_model.evaluate(val_generator)
```

```
5/5 [==============================] - 22s 4s/step - loss: 6.0628e-04 - accuracy: 1.0000
[0.0006062801112420857, 1.0, 1.0]
```

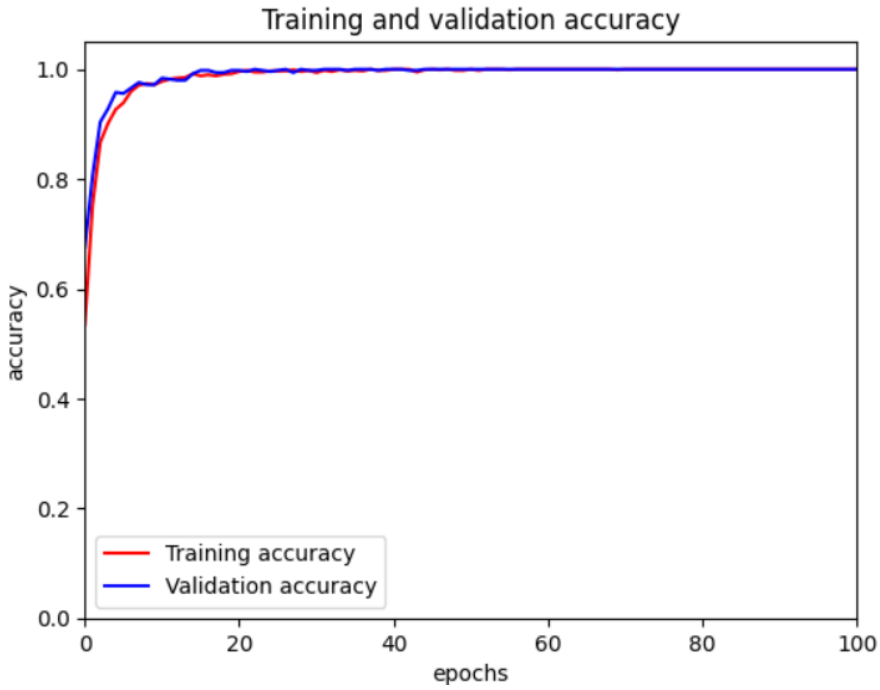
3.5.2.1. Validación del modelo: Gráficas de exactitud y pérdida del modelo entrenado

Posteriormente, se realizan las gráficas de pérdida y exactitud con el fin de analizar la evolución del entrenamiento de la red, a su vez se realiza una matriz de confusión para conocer el porcentaje de precisión del modelo entrenado.

En la figura 52 se muestra la gráfica que representa la exactitud del modelo en función de los ciclos realizados para el conjunto de datos de entrenamiento y validación, se observa que al iniciar con las iteraciones se tiene una exactitud media, pero a medida que avanzan las iteraciones la curva de exactitud crece y se mantiene constante aumentando el porcentaje de precisión sobre el 95% a partir del décimo ciclo.

Figura 52

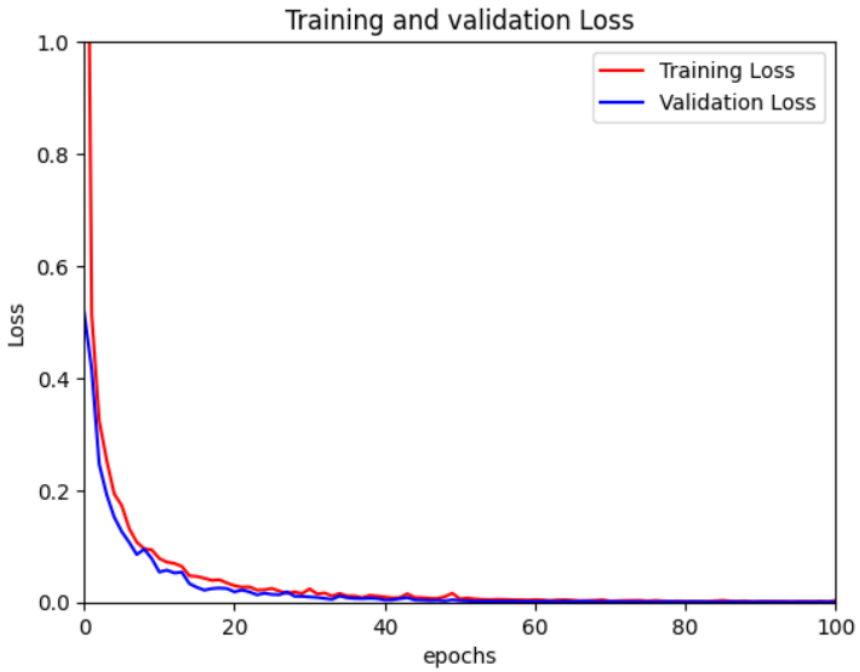
Gráfica de exactitud



En la figura 53 se muestra la gráfica que representa la pérdida en función de las épocas o ciclos para el conjunto de datos de entrenamiento y validación y se observa que al iniciar el modelo tiene un valor medio de pérdidas, y la curva va decreciendo hasta llegar a la época cincuenta obteniendo un valor de pérdida mínimo.

Figura 53

Gráfica de pérdida



De las dos gráficas anteriores se observa que la precisión aumenta y la pérdida disminuye en función de los ciclos realizados, tanto para el conjunto de entrenamiento como para el de validación, lo que significa que el modelo cometió pocos errores en algunos datos. Además, al utilizar la arquitectura de red neuronal Inception V3 se logra reducir Overfitting o sobreajuste del modelo, ya que en la gráfica se observa que la curva es similar para los datos de entrenamiento y validación. En caso de que la pérdida disminuyera en los datos de entrenamiento y no en los de validación el modelo estaría aprendiendo demasiado de los datos de entrenamiento.

3.5.2.2. Matriz de confusión

En inteligencia artificial y aprendizaje automático, se propone el uso de matrices de confusión como herramienta de visualización del rendimiento del algoritmo utilizado, de forma que se puedan conocer la cantidad de éxitos y fracasos del modelo planteado a

medida que avanza en el proceso de aprendizaje. Esta matriz consta de filas que representan los valores reales y las columnas representan los valores de predicciones.

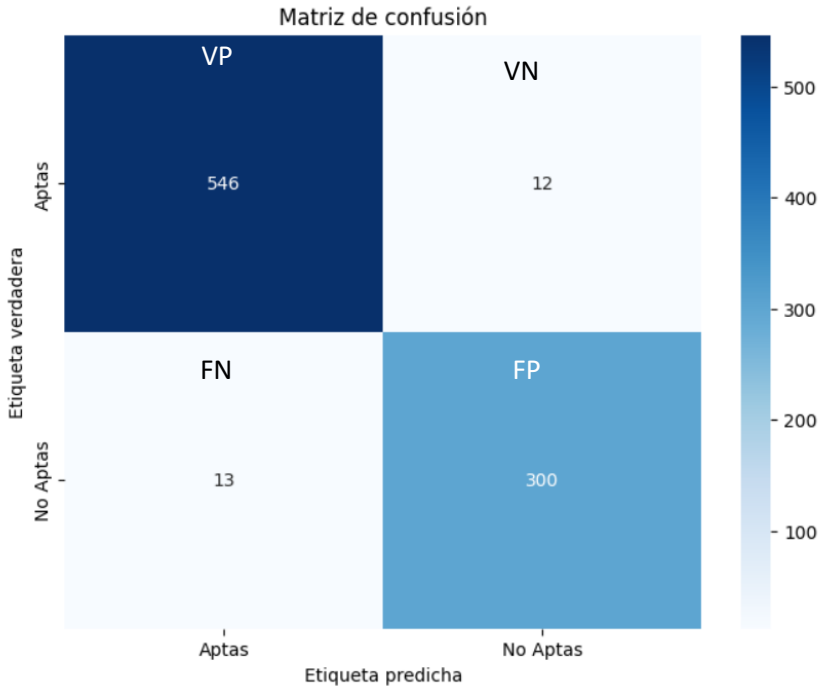
En este caso, al tratar con dos etiquetas “aptas” y “no aptas”, se trata de una matriz binaria y surgen cuatro casos:

- Verdadero positivo (VP): número de casos en donde la lechuga pertenece a la clase apta y el modelo lo clasificó como apta.
- Verdadero negativo (VN): número de casos en donde la lechuga pertenece a la clase no apta y el modelo lo clasificó como no apta.
- Falso negativo (FN): número de casos en donde la lechuga pertenece a la clase apta y el modelo lo clasificó como no apta.
- Falso positivo (FP): número de casos en donde la lechuga pertenece a la clase no apta y el modelo lo clasificó como apta.

En la figura 54 se establece la matriz de confusión, la cual consta de etiquetas reales y etiquetas predichas. El conjunto de datos puestos a prueba es de 871 muestras de las cuales 558 pertenecen a la etiqueta apta, mientras que 313 pertenecen a la etiqueta no aptas. En la matriz de confusión se observa que para la etiqueta aptas existen 546 muestras acertadas, mientras que para la etiqueta no apta existen 300 muestras acertadas.

Figura 54

Matriz de confusión del modelo entrenado



En la figura 55 se muestra un reporte de clasificación como métrica para evaluar el rendimiento del modelo entrenado. Los parámetros evaluados son:

- **Precisión:** métrica para conocer el porcentaje de casos positivos detectados. Para ello se realiza el cálculo de verdaderos positivos dividido entre los resultados positivos tomando en cuenta verdaderos positivos y falsos positivos. El valor de precisión del modelo entrenado es de 97%.

$$P = \frac{VP}{VP + FP} x100 \qquad (1)$$

- **Recuperación(recall):** es la relación entre los verdaderos positivos y la suma de verdaderos positivos y falsos negativos.

$$R = \frac{VP}{VP + FN} x100 \qquad (2)$$

- **Puntuación F1 (f1-score):** es la media de precisión y recuperación. El rendimiento será mayor cuanto más cerca esté el valor de 1.0

$$F1\ score = 2 * \frac{Precisión * Recuperación}{Precisión + Recuperación} \quad (3)$$

- **Soporte del modelo (support):** Es el número de datos existentes para cada clase.

Figura 55

Reporte de clasificación

	precision	recall	f1-score	support
0	0.977	0.978	0.978	558
1	0.962	0.958	0.960	313
accuracy			0.971	871
macro avg	0.969	0.968	0.969	871
weighted avg	0.971	0.971	0.971	871

Una vez comprobada la precisión del modelo entrenado se guarda el modelo en formato .tflite y .h5 para ser usado posteriormente en el proceso de inferencia realizado en la placa embebida Nvidia Jetson Nano como se muestra en la figura 56.

Figura 56

Modelo almacenado en formato .tflite y .h5

3.5.3 Integración del sistema

Una vez obtenido el modelo pre entrenado, se procede a realizar la integración junto con el sistema embebido Nvidia Jetson Nano. Para ello es necesario realizar un proceso de inferencia, el cual consiste en probar el modelo de Tensorflow en un dispositivo para realizar predicciones tomando en cuenta datos de entrada.

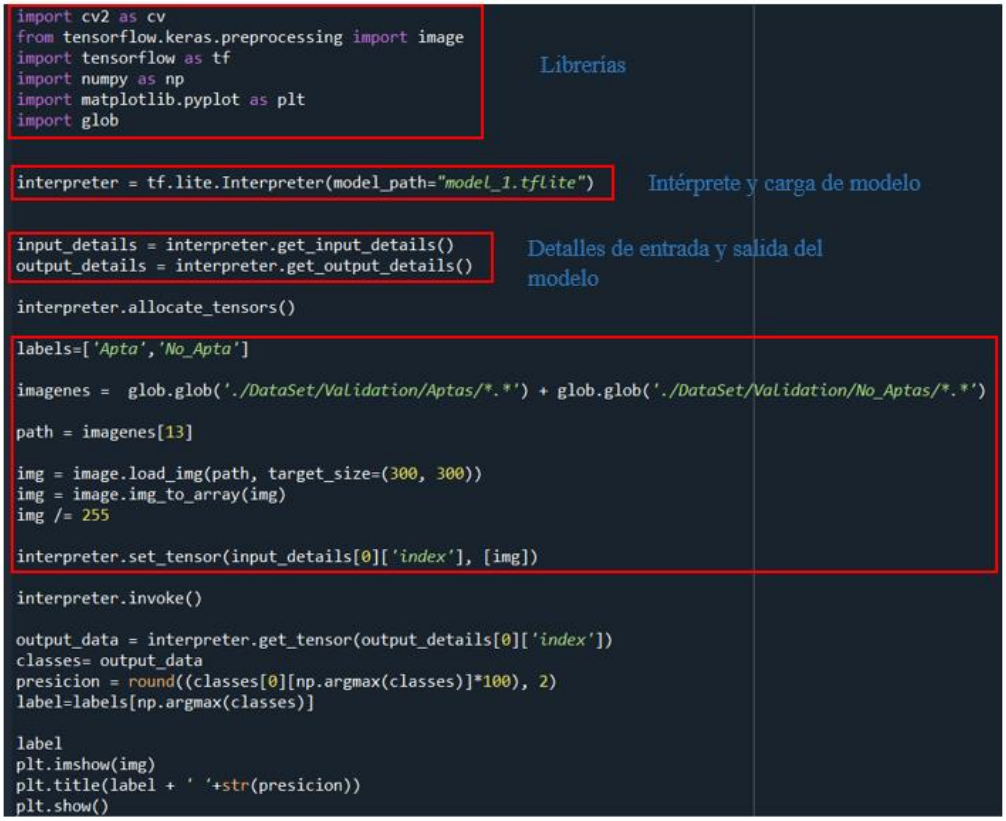
A continuación, en la figura 57, se muestra el código generado para realizar las pruebas de funcionamiento del sistema con imágenes individuales. Primero se importan las librerías listadas en la tabla 12, entre ellas CV2 librería de visión artificial, Tensorflow permite encontrar patrones, Numpy proporciona funciones matemáticas y arreglos, Matlplotlib crea gráficos y la librería Glob permite encontrar rutas que coincidan con un patrón especificado.

Posteriormente, se crea una variable intérprete y se carga el modelo. Esta variable permitirá probar el modelo y hacer predicciones en base a datos de entrada. Luego se obtienen los detalles tanto de entrada como de salida del modelo. A continuación, se llama a la función *"allocate_tensor",* esta función optimiza el proceso de inferencia de forma que se planifica previamente la asignación de tensores o matrices que almacenan los valores numéricos de los datos.

Después, en la variable *"labels"* se almacenan las etiquetas, en este caso "Apto" y "No Apto"; en la variable *"imagenes"* se almacenan las imágenes encontradas luego de realizar la búsqueda de rutas que coincidan con el patrón establecido. Se escoge la imagen a clasificar y se realiza un tratamiento a la imagen, en el cual se especifican las dimensiones, en este caso de 300 x 300 pixels, convierte la imagen en un arreglo y normaliza la imagen de modo que los valores de los pixeles que oscilan entre 0 y 256 se conviertan en valores entre 0 y 1, lo que hace el cálculo más fácil y rápido.

Figura 57

Código de modelo de inferencia




```
import cv2 as cv
from tensorflow.keras.preprocessing import image
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import glob

interpreter = tf.lite.Interpreter(model_path="model_1.tflite")

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

interpreter.allocate_tensors()

labels=['Apta','No_Apta']

imagenes =  glob.glob('./DataSet/Validation/Aptas/*.*') + glob.glob('./DataSet/Validation/No_Aptas/*.*')

path = imagenes[13]

img = image.load_img(path, target_size=(300, 300))
img = image.img_to_array(img)
img /= 255

interpreter.set_tensor(input_details[0]['index'], [img])

interpreter.invoke()

output_data = interpreter.get_tensor(output_details[0]['index'])
classes= output_data
presicion = round((classes[0][np.argmax(classes)]*100), 2)
label=labels[np.argmax(classes)]

label
plt.imshow(img)
plt.title(label + ' '+str(presicion))
plt.show()
```

Para realizar las pruebas de funcionamiento del sistema con imágenes de los lotes completos se hace uso de tres Scrips. El primero mostrado en la figura 57 permite realizar el proceso de inferencia, el segundo realiza el procesamiento de las imágenes de entrada de cada lote y el tercero crea la interfaz gráfica.

En la figura 58 se muestra el procesamiento de las imágenes de entrada de cada lote, para lo cual primero se importan las librerías de CV2, Numpy, Matlplotlib, a su vez se importa el modelo de inferencia de la figura 57.

Luego, en el mismo Script, se establecen seis funciones:

- La primera función "*write_pred()"*agrega texto en la imagen, usando la fuente *"Hershey Simplex"* y como parámetros se especifican la posición, el color y el tamaño de la fuente.

- La segunda función, "*division_lotes()"* permite que de la imagen original se obtenga la imagen de los lotes por separado.
- La tercera función "*cuadros_img()"* dibuja la cuadrícula en las posiciones en las que se ubican las imágenes de los lotes divididos. Los argumentos especificados son: la imagen donde se quiere dibujar la cuadrícula, la posición, el color y el tipo de línea.
- La cuarta y quinta función "*proceso_lote1()"* y "*proceso_lote2()"* realizan el procesamiento de la imagen de entrada, se aplica una transformación de perspectiva para mejorar la visibilidad de la imagen y eliminar curvas. Para ello se establecen las coordenadas de la imagen que se quiere delimitar y mediante la función *"PerspectiveTransform()"* se obtiene una matriz de perspectiva con los puntos establecidos y la función*"WarpPerspective()"* obtiene la imagen alineada y sin curvas.
- La sexta función, es la principal y proporciona un orden jerárquico de ejecución de las funciones explicadas anteriormente.

Figura 58

Script de procesamiento de la imagen

```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import model_inf

def write_pred(labels,img_div):
    predicciones_values = list(labels.values())
    cont=0
    for pos_y in range(150,2100,300):
        for pos_x in range(75,1500,300):
            predic = predicciones_values[cont]
            if predic == "Apta":
                img_div = cv.putText(img_div, predic, (pos_x, pos_y), cv.FONT_HERSHEY_SIMPLEX, 1.5,( 0,255,0),6 )
            else:
                img_div = cv.putText(img_div, predic, (pos_x, pos_y), cv.FONT_HERSHEY_SIMPLEX, 1.5,( 255,0,0) ,6)
            cont +=1
    return img_div

def division_lotes(img_ori):

    lote_1=img_ori[100:800,0:640]
    img_l2 = img_ori.copy()
    lote_2=img_l2[170:900,920:]
    return lote_1, lote_2

def cuadros_img(img_dib):
    for pos in range(0,2100,300):
        cv.line(img_dib,(0,pos),(1500,pos),(255,0,0),8)

    for pos in range(0,1500,300):
        cv.line(img_dib,(pos,0),(pos,2100),(255,0,0),8)
    #plt.imshow(img_dib)

    return img_dib

def proceso_lote1(img_lote):
    conten={}
    pts1 = np.float32([[10,60],[10,600],[640-1,0],[640-1,700-1]])
    pts2 = np.float32([[0,0],[0,2100-1],[1500-1,0],[1500-1,2100-1]])
    matriz = cv.getPerspectiveTransform(pts1,pts2)
    dst = cv.warpPerspective(img_lote,matriz,(1500,2100))
    cont=1
    for pos_y in range(0,2100,300):
        for pos_x in range(0,1500,300):
            #name = "lote1_"+str(pos_y)+","+str(pos_x)+".jpg"
            img_temp = dst[pos_y:pos_y+300, pos_x:pos_x+300]
            conten.setdefault(cont,img_temp)
            #cv.imwrite(name,img_temp)
            cont+=1
    img_cuadros=cuadros_img(dst.copy())
    return conten, img_cuadros


def proceso_lote2(img_lote):
    conten={}
    pts1 = np.float32([[40,10],[40,700],[650,70],[650,620]])
    pts2 = np.float32([[0,0],[0,2100-1],[1500-1,0],[1500-1,2100-1]])
    matriz = cv.getPerspectiveTransform(pts1,pts2)
    dst = cv.warpPerspective(img_lote,matriz,(1500,2100))
    cont=1
    for pos_y in range(0,2100,300):
        for pos_x in range(0,1500,300):
            #name = "lote2_"+str(pos_y)+","+str(pos_x)+".jpg"
            img_temp = dst[pos_y:pos_y+300, pos_x:pos_x+300]
            conten.setdefault(cont,img_temp)
            #cv.imwrite(name,img_temp)
            cont+=1
    img_cuadros=cuadros_img(dst.copy())
    return conten, img_cuadros

def main():

    path = "./camara1/img1.jpg"
    img = cv.imread(path)
    img_temp = img.copy()

    img_lote1,img_lote2 = division_lotes(img_temp)


    img_ind_lote1, img_div_cuadros_1=proceso_lote1(img_lote1)
    img_ind_lote2, img_div_cuadros_2=proceso_lote2(img_lote2)

    print(img_ind_lote2)
    plt.imshow(img_ind_lote2[1])

    print(model_inf.inferencia_modelo(img_ind_lote1[20]))

if __name__=="__main__":
    main()
```

Librerías y modelo de inferencia

Para la creación de la interfaz gráfica, se importan las siguientes librerías:

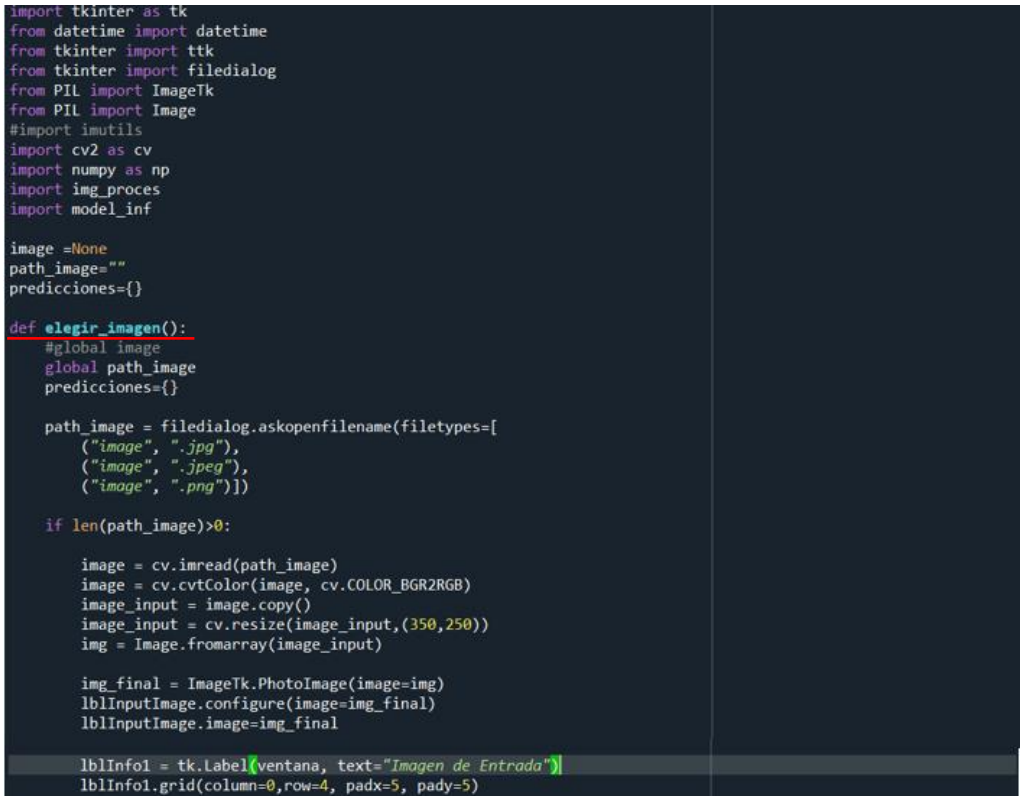
- Tkinter: este paquete proporciona herramientas para administrar ventanas gráficas.
- Datetime: módulo que permite manipular fechas y horas.
- PIL: librería que facilita el manejo de formatos de archivos de imágenes.
- CV2: librería de visión artificial.
- Numpy: módulo para manejo de arreglos y funciones numéricas.

Además, se importa el modelo de inferencia de la figura 57, el script del procesamiento de la imagen de la figura 58, y se establecen tres funciones para cargar la imagen a analizar, realizar predicciones del lote 1 y del lote 2.

En la figura 59 se muestra la función "*elegir_imagen()". E*sta función permite cargar la imagen a analizar, para lo cual se establece que el formato de la imagen de la entrada que se admite, éstos son jpg, jpeg y png. Además, a la imagen de entrada se le aplica una función para convertir la imagen en espacios de color de BGR a RGB y se redimensiona a un tamaño de 350x250 pixels para la visualización en la interfaz, finalmente a la imagen se la guarda en un arreglo.

Figura 59

Creación de la interfaz gráfica –Carga de imagen de entrada



```
import tkinter as tk
from datetime import datetime
from tkinter import ttk
from tkinter import filedialog
from PIL import ImageTk
from PIL import Image
#import imutils
import cv2 as cv
import numpy as np
import img_proces
import model_inf

image =None
path_image=""
predicciones={}

def elegir_imagen():
    #global image
    global path_image
    predicciones={}

    path_image = filedialog.askopenfilename(filetypes=[
        ("image", ".jpg"),
        ("image", ".jpeg"),
        ("image", ".png")])

    if len(path_image)>0:

        image = cv.imread(path_image)
        image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
        image_input = image.copy()
        image_input = cv.resize(image_input,(350,250))
        img = Image.fromarray(image_input)

        img_final = ImageTk.PhotoImage(image=img)
        lblInputImage.configure(image=img_final)
        lblInputImage.image=img_final

        lblInfo1 = tk.Label(ventana, text="Imagen de Entrada")
        lblInfo1.grid(column=0,row=4, padx=5, pady=5)
```

En las figuras 60 y 61, se muestra la función "*Predecir_lote1()"* y "*Predecir_lote2()",* respectivamente. Con esta función, a la imagen de cada lote obtenida de la división de la imagen original, se le aplica el modelo de inferencia y se realiza la clasificación de lechugas aptas y no aptas, además de un contador para ambas clases.

Figura 60

Creación de la interfaz gráfica –Predicción lote 1

```
def predecir_lote1():
    predicciones={}
    global path_image
    #print(path_image)
    if len(path_image)>0:
        image = cv.imread(path_image)
        image = cv.cvtColor(image, cv.COLOR_BGR2RGB)

        img_lote1,img_lote2 = img_proces.division_lotes(image.copy())
        img_ind_lote1, img_div_cuadros_1 = img_proces.proceso_lote1(img_lote1)

        for key in img_ind_lote1.keys():
            predicciones.setdefault(key,model_inf.inferencia_modelo(img_ind_lote1[key]))

        predicciones_values = list(predicciones.values())
        Aptas = predicciones_values.count("Apta")
        print(Aptas)
        No_Aptas = predicciones_values.count("No_Apta")
        print(No_Aptas)

        lblInfo_Aptas = tk.Label(ventana, text="Cantidad Lechugas Aptas = "+str(Aptas))
        lblInfo_Aptas.grid(column=1,row=4, padx=5, pady=5)

        lblInfo_NoAptas = tk.Label(ventana, text="Cantidad Lechugas No Aptas = "+str(No_Aptas))
        lblInfo_NoAptas.grid(column=1,row=6, padx=5, pady=5)

        lblInfo_lote = tk.Label(ventana, text="Lote 1", font=("bold",10))
        lblInfo_lote.grid(column=4,row=2, padx=5, pady=5)


        img_resize = img_div_cuadros_1.copy()

        img_resize = img_proces.write_pred(predicciones, img_resize)

        row,cols,chan = img_resize.shape
        img_resize = cv.resize(img_resize,(cols//4, row//4))

        img = Image.fromarray(img_resize)
        img_final = ImageTk.PhotoImage(image=img)
        lblOutputImage.configure(image=img_final)
        name =str(datetime.now())
        name=name.replace("-","_")
        name=name.replace(" ","__")
        name=name.replace(":","_")
        name=name[:20]
        img_resize = cv.cvtColor(img_resize, cv.COLOR_RGB2BGR)
        save = cv.imwrite("./Predicciones/lote1_"+name+".jpg",img_resize)
```

Figura 61

Creación de la interfaz gráfica –Predicción lote 2

```
def predecir_lote2():
    global path_image
    #print(path_image)
    if len(path_image)>0:
        image = cv.imread(path_image)
        image = cv.cvtColor(image, cv.COLOR_BGR2RGB)

        img_lote1,img_lote2 = img_proces.division_lotes(image.copy())
        img_ind_lote2, img_div_cuadros_2 = img_proces.proceso_lote2(img_lote2)

        for key in img_ind_lote2.keys():
            predicciones.setdefault(key,model_inf.inferencia_modelo(img_ind_lote2[key]))

        predicciones_values = list(predicciones.values())
        Aptas = predicciones_values.count("Apta")
        print(Aptas)
        No_Aptas = predicciones_values.count("No_Apta")
        print(No_Aptas)

        lblInfo_Aptas = tk.Label(ventana, text="Cantidad Lechugas Aptas = "+str(Aptas))
        lblInfo_Aptas.grid(column=1,row=4, padx=5, pady=5)

        lblInfo_NoAptas = tk.Label(ventana, text="Cantidad Lechugas No Aptas = "+str(No_Aptas))
        lblInfo_NoAptas.grid(column=1,row=6, padx=5, pady=5)

        lblInfo_lote = tk.Label(ventana, text="Lote 2", font=("bold",10))
        lblInfo_lote.grid(column=4,row=2, padx=5, pady=5)

        img_resize = img_div_cuadros_2.copy()

        img_resize = img_proces.write_pred(predicciones, img_resize)

        row,cols,chan = img_resize.shape
        img_resize = cv.resize(img_resize,(cols//4, row//4))

        img = Image.fromarray(img_resize)
        img_final = ImageTk.PhotoImage(image=img)
        lblOutputImage.configure(image=img_final)
        lblOutputImage.image=img_final

        name =str(datetime.now())
        name=name.replace("-","_")
        name=name.replace(" ","__")
        name=name.replace(":","_")
        name=name[:20]
        img_resize = cv.cvtColor(img_resize, cv.COLOR_RGB2BGR)
        save = cv.imwrite("./Predicciones/lote2_"+name+".jpg",img_resize)
```

En la figura 62, se muestra la creación de la interfaz gráfica, títulos, botones para seleccionar la imagen a analizar y procesar lotes 1 y 2, además se especifica la posición de las imágenes de entrada y de salida.

Figura 62

Interfaz gráfica – Ventana de visualización

```
ventana = tk.Tk()
ventana.title("Clasificador de Lechugas")

lbltitle = tk.Label(ventana, text="Universidad Tecnica del Norte", font=("bold",10))
lbltitle.grid(column=1,row=1, padx=5, pady=5)

btn = tk.Button(ventana,text="Elegir Imagen", width=25, command=elegir_imagen)
btn.grid(column=0, row=2, padx=5, pady=5)



btn_proces_l1 = tk.Button(ventana,text="Procesar Lote 1", width=25, command=predecir_lote1)
btn_proces_l1.grid(column=0, row=5, padx=5, pady=5)

btn_proces_l2 = tk.Button(ventana,text="Procesar Lote 2", width=25, command=predecir_lote2)
btn_proces_l2.grid(column=0, row=6, padx=5, pady=5)

# Etiqueta donde se muestra la Imagen de entrada
lblInputImage=tk.Label(ventana)
lblInputImage.grid(column=0,row=3)

# Etiqueta donde se muestra la imagen de salida
lblOutputImage = tk.Label(ventana)
lblOutputImage.grid(column=4, row=3,rowspan=6)
```

Capítulo IV

EJECUCIÓN Y PRUEBAS DE FUNCIONAMIENTO

En el cuarto capítulo se realiza la implementación del sistema propuesto tomando en cuenta los dos escenarios planteados, además se realizan las pruebas de funcionamiento del sistema de detección de estrés hídrico para verificar el cumplimiento de los requerimientos establecidos en el proyecto. De igual manera, se establecen conclusiones de las resultados obtenidos y recomendaciones que pueden ser tomados en cuenta para trabajos futuros.

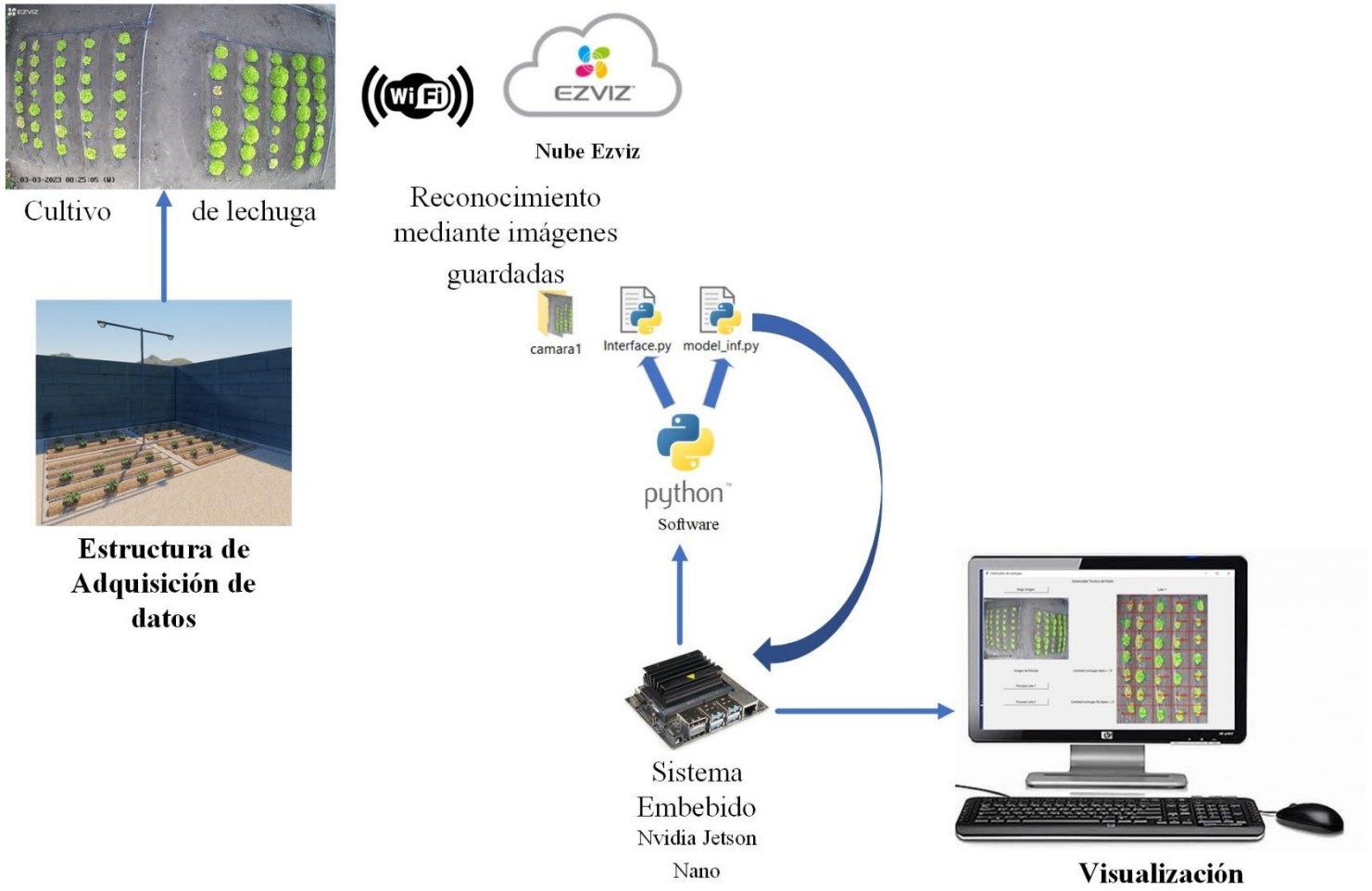
4.1. Diagrama de funcionamiento

Para comprender de mejor manera el funcionamiento del sistema se plantea la topología de la figura 63. El sistema inicia con la adquisición de imágenes del cultivo de lechuga, la imagen se almacena en la nube de Ezviz. Posteriormente se ejecuta el modelo pre entrenado en la placa embebida Nvidia Jetson Nano. El modelo pre entrenado se obtiene mediante el software Google Colab, debido a que, a pesar de que la placa embebida Nvidia Jetson Nano ejecuta códigos de visión artificial, tiene una limitante, no es apta para el entrenamiento de la red. En este bloque se hace uso de mapas de características, operaciones del entrenamiento existentes del modelo pre entrenado y las clases establecidas.

Para finalizar, una vez procesada la información, el usuario puede acceder a la información de la clasificación de la lechuga mediante una interfaz amigable.

Figura 63

Diagrama de funcionamiento del sistema



4.2. Pruebas funcionales

En esta sección se muestran los resultados obtenidos de la implementación del sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de Visión Artificial. A continuación, se listan las pruebas realizadas en este bloque:

- Prueba de clasificación individual, mediante la ejecución del código del modelo de inferencia de la figura 57 y el uso de imágenes individuales de lechugas se cataloga como apta o no apta y se obtiene el porcentaje de acierto.
- Prueba de clasificación de lotes, usando imágenes de escenarios completos se procesa la imagen de lote 1 o lote 2 y se obtiene la cantidad de lechugas etiquetadas como aptas y no aptas.

4.2.1. Prueba de clasificación individual

Para esta prueba se optó por usar imágenes individuales de lechugas, de modo que se pueda probar la funcionalidad del modelo previamente entrenado en Google Colab, haciendo uso de la placa embebida Nvidia Jetson Nano. Una vez ejecutado el modelo de inferencia de la figura 57, se obtuvo como resultado las figuras 64 y 65, mostradas a continuación. La primera imagen es catalogada con la etiqueta No Apta y tiene un porcentaje de acierto de 99.85%, mientras que la segunda imagen es catalogada con la etiqueta Apta y tiene un porcentaje de acierto de 100%

Figura 64

Prueba de clasificación individual – Lechuga no apta

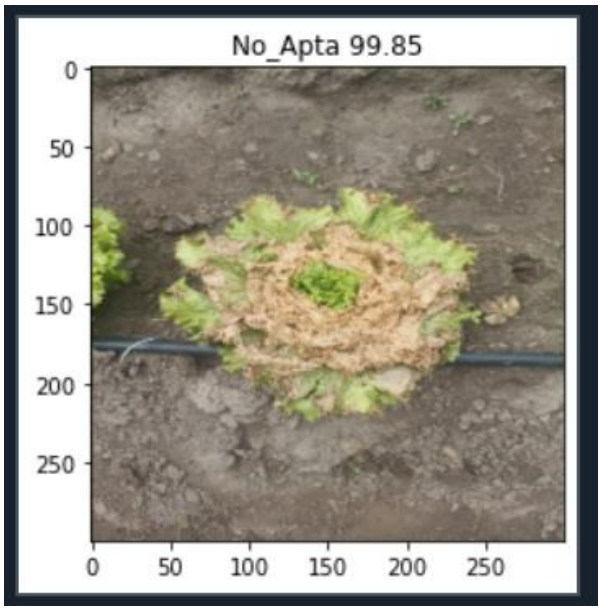
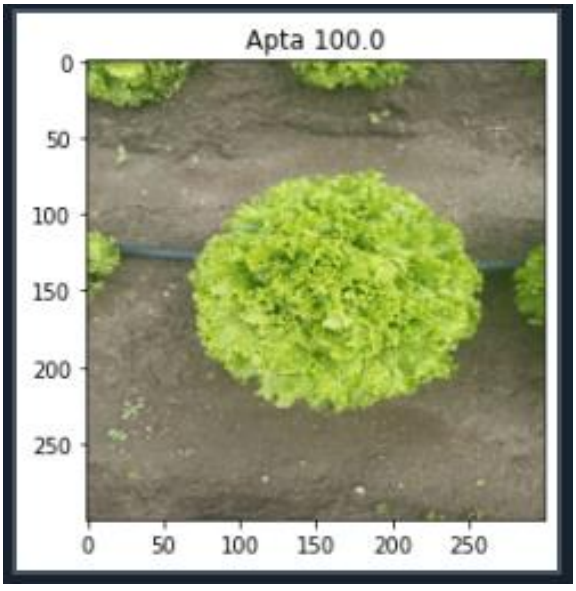


Figura 65

Prueba de clasificación individual – Lechuga apta

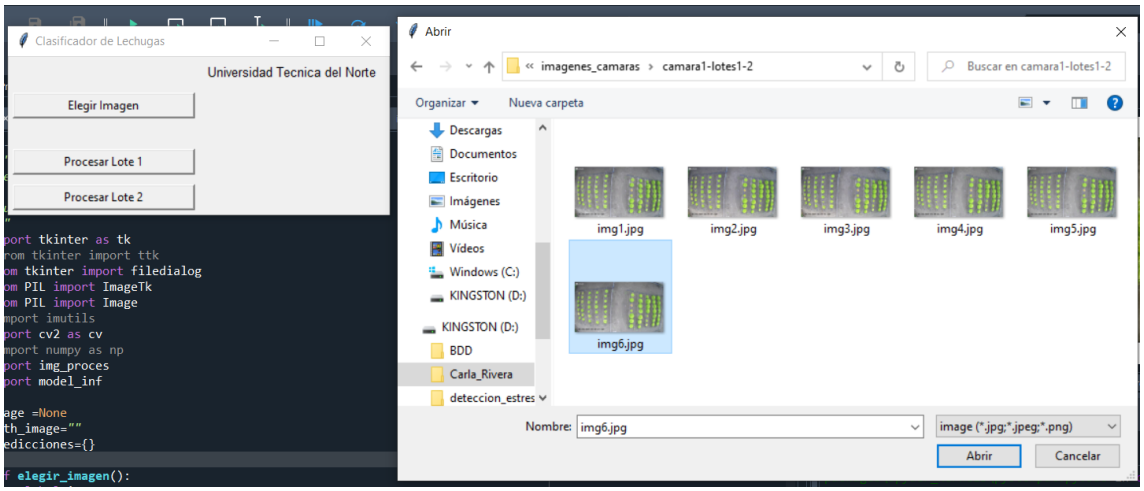


4.2.2. Prueba de clasificación de lotes

Para visualizar los resultados de la prueba se requiere de una pantalla. Se realiza mediante imágenes, para lo cual se hace uso de imágenes guardadas de escenarios completos, se escoge la imagen de la cámara 1 el cual contiene a ambos lotes, como se muestra en la figura 66.

Figura 66

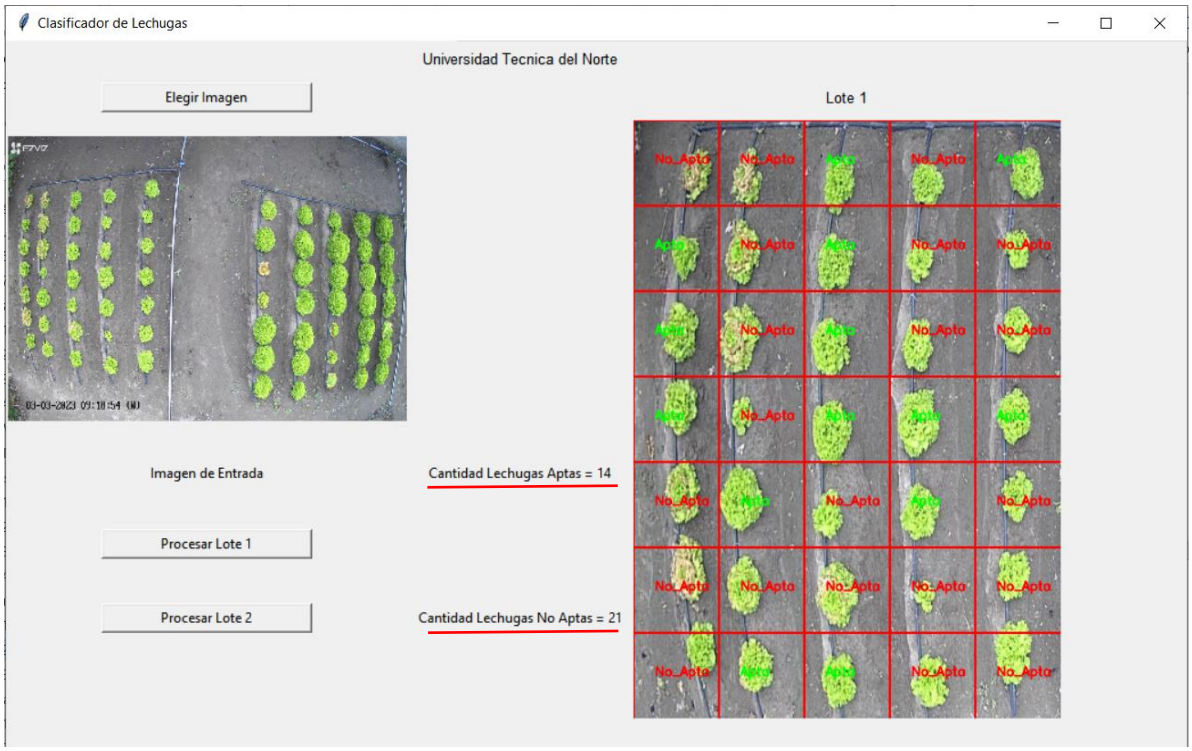
Selección de imagen a analizar



Luego al seleccionar el botón "*Procesar Lote 1*" se obtiene la imagen procesada y dividida del lote 1, a su vez se muestra la cantidad de imágenes clasificadas como aptas y no aptas, obteniendo así el valor de 14 lechugas aptas y 21 no aptas, como se muestra en la figura 67.

Figura 67

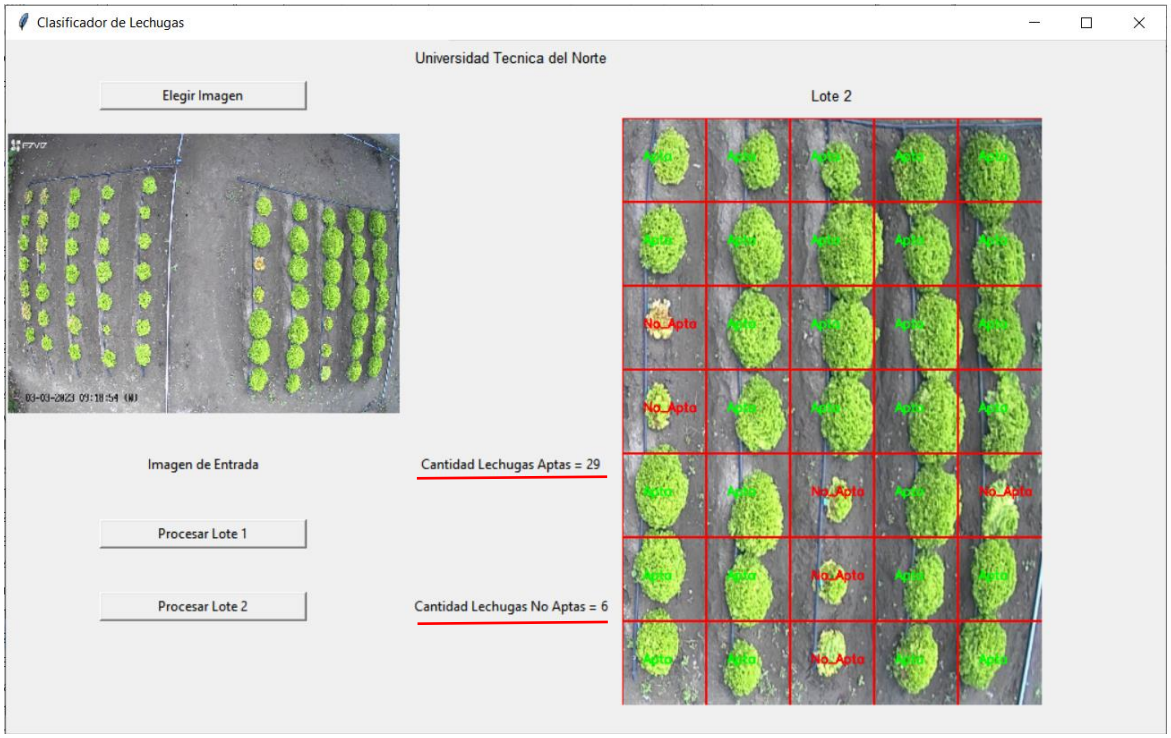
Clasificación de lote 1



Y al seleccionar el botón "*Procesar Lote 2*" se obtiene la imagen procesada y dividida del lote 2, junto con la cantidad de imágenes clasificadas como aptas y no aptas, obteniendo así el valor de 29 lechugas aptas y 6 no aptas, como se muestra en la figura 68.

Figura 68

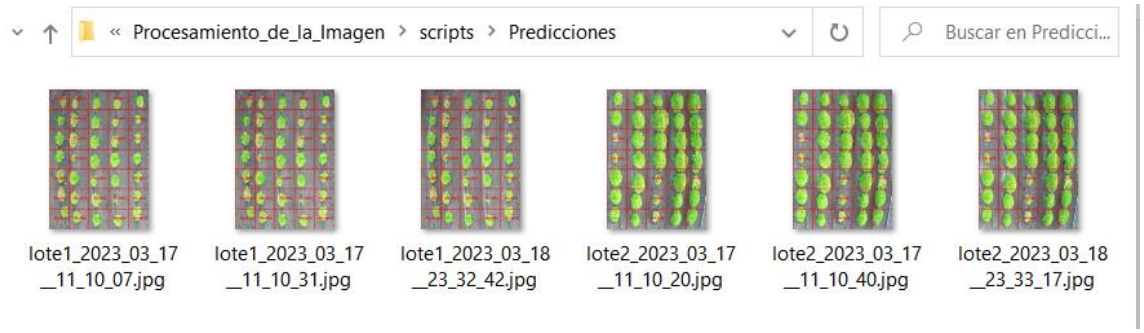
Clasificación Lote 2



Por último, las imágenes procesadas de cada lote se almacenan en el directorio: *"C:\Users\Lenovo\Desktop\Carla_Rivera\Procesamiento_de_la_Imagen\scripts\Predic ciones"*, como se muestra en la figura 69.

Figura 69

Almacenamiento de imágenes procesadas



4.3. Comparación entre cultivos

Se realiza un análisis de los dos escenarios en los cuales se ha probado el sistema de detección de estrés hídrico en un ambiente de producción. El primer escenario cuenta con un terreno preparado, abonado y con un regadío esporádico, mientras que el escenario 2 tiene un terreno más seco y con un regadío esporádico. En la tabla 13 se muestra la comparación realizada entre los dos escenarios.

Tabla 13

Comparación de escenarios

Comparación entre escenarios		
	Escenario 1 (apto)	**Escenario 2 (no apto)**
Planta	- Coloración verde oscuro - Presenta abundantes hojas - Planta erguida - Señales de salud óptima - Altura: 9cm 	- Coloración verde claro, amarillo - Existe menor cantidad de hojas - Planta flácida, caída - Señales de debilidad - Altura 5cm 
Periodo de crecimiento	8 semanas	12 semanas
Cualidades de las hojas	- Hojas gruesas - Coloración verde oscuro - Bordes Sanos 	- Hojas flácidas - Coloración verde claro-amarillo - Bordes secos 

De la tabla 14 se observa el producto obtenido de ambos escenarios, del escenario que cumple con condiciones aptas se obtuvo lechugas de hojas abundantes, coloración verde oscuro, con bordes sanos y una altura de 9cm en un periodo de 8 semanas, mientras que, para el escenario de condiciones no aptas, se obtuvo lechugas de coloración verde claro y amarillo, con bordes secos y una altura de 5cm en un periodo de 12 semanas.

4.4. Análisis Costo/Beneficio

En esta sección se describe el costo de los componentes que integran el sistema, se consideran costos de hardware, software, infraestructura e ingeniería.

4.4.1. Costos de hardware

En la tabla 14 se muestran todos los componentes de hardware necesarios para el desarrollo del proyecto, además se señala la cantidad y costos de cada elemento.

Tabla 14

Costos de hardware

Cantidad	Elemento	Costo Unitario	Costo Total
1	Placa embebida Jetson Nano	$ 180	$ 180
2	Cámara Ezviz C3W Pro	$ 95.47	$ 190.95
	TOTAL		$ 370.95

4.4.2. Costos de software

En la tabla 15 se muestran los costos de software, se optó por hacer uso de software sin pago de licencia, a excepción del almacenamiento de las cámaras. Éste se lo realiza en la nube de Ezviz y ofrece un plan familiar de almacenamiento de 7 días de hasta cuatro cámaras.

Tabla 15

Costos de Software

Elemento	Costo Unitario	Costo Total
Google Drive	0	0
Google Colab	0	0
Editor Spyder	0	0
Python	0	0
Nube Ezviz	$ 7.5	$ 15
TOTAL		$ 15

4.4.3. Costos de infraestructura

En la tabla 16 se detalla el costo de los elementos asociados a la instalación de infraestructura metálica y cámaras, además de las respectivas protecciones.

Tabla 16

Costos de infraestructura

Cantidad	Elemento	Costo Unitario	Costo Total
1	Estructura metálica	$ 60	$ 60
60m	Cable de red	$ 0.41	$ 24.6
6	Conector RJ45	$ 0.25	$ 1.5
1	Splitter HDMI 5 puertos	$ 15	$ 15
2	Caja de protección 15x15	$ 2	$ 4
1	Caja de protección 22x17	$ 10	$ 10
35m	Cable de luz	$ 0.76	$ 26.60
1	Material extra	$ 25	$ 25
	TOTAL		$ 166.7

4.4.4. Costos de ingeniería

Para el costo del trabajo de ingeniería se han considerado actividades como: diseño de hardware, programación de código para los procesos de entrenamiento y clasificación, implementación del sistema y costos adicionales de estudio de campo y documentación. Los costos asociados al trabajo de ingeniería mostrados en la tabla 17 son asumidos por la autora del presente proyecto.

Tabla 17

Costos de ingeniería

Actividades	Costo Unitario	Costo Total
Estudio de campo y documentación	$ 20	$ 20
Diseño de hardware, programación de códigos de entrenamiento y clasificación	$ 200	$ 200
Implementación de infraestructura	$100	$ 100
TOTAL		$ 320

4.4.5. Costo general del sistema

En la tabla 18 se muestra una recopilación de los costos asociados al desarrollo del presente proyecto. El costo total del sistema tuvo un valor de $877.65, cabe mencionar que el valor del trabajo de ingeniería es de $ 320 y se han considerado las horas dedicadas a actividades de investigación, diseño, construcción del proyecto y programación de códigos. Este valor es asumido por la autora por lo cual el valor neto del sistema es de $ 537.65.

Es oportuno enfatizar que para futuras implementaciones específicamente en terrenos medianos y extensos de mayor producción, se reducirían los costos de hardware e infraestructura considerablemente, ya que las cámaras, el armazón metálico y el

cableado tanto eléctrico como de red pueden ser reemplazados por el uso de drones con cámara incluida.

Tabla 18

Costo general del sistema

Descripción	Costo Unitario
Costos de hardware	$ 370.95
Costos de software	$ 0
Costos de Infraestructura	$ 166.7
Costos de Ingeniería	$ 320
TOTAL	$ 857.65

4.4.6. Beneficios

A continuación, se detallan los beneficios contemplados de la implementación del sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de Visión Artificial:

- Junto con la aplicación de otro tipo de tecnologías modernas, como la recopilación, interpretación y análisis de datos mediante dispositivos electrónicos, GPS y métodos de visión artificial y lenguajes de aprendizaje automático, es posible gestionar eficazmente la producción agrícola, facilitar la gestión de la tierra y optimizar el uso de recursos.
- Permite a los agricultores evaluar la calidad del cultivo de lechuga durante el mantenimiento y la cosecha, facilitando la detección temprana del estrés hídrico en los cultivos, de forma que se puedan aplicar medidas correctivas para recuperar la planta afectada y evitar pérdidas en la producción.

- Asegura una producción completa y productos con suficiente valor nutricional que aporten a una mejor calidad de vida; además, no solo reduce los costos de producción, sino que también reduce la cantidad de máquinas y personal.
- Durante el proceso de recolección es posible y mitigar el margen de error en la clasificación del producto debido a factores humanos como cansancio físico o fatiga visual.

CONCLUSIONES

Mediante la fundamentación teórica se pudo recopilar información acerca del estrés hídrico, tipos y factores que afectan al crecimiento y desarrollo de la planta, al igual que las características que permiten reconocer visualmente la presencia del estrés hídrico en el cultivo, como el color, tamaño y defectos, mismos que han servido como base para el entrenamiento de la red neuronal.

De igual forma, a través del marco teórico se obtuvo conocimiento sobre los distintos tipos de redes neuronales, siendo utilizada en este caso la arquitectura red neuronal convolucional Inception V3 debido a que permite extraer características profundas del objeto; presenta un índice de error de 3.58% y una precisión de 93.9% además, se obtuvo información acerca de métodos de procesamiento de imagen para la extracción de características de muestras del objeto en cuestión.

Se plateó un esquema de adquisición de datos, mediante el uso de una estructura metálica fija, ubicada a una altura que permita obtener una visión amplia del terreno y para la ejecución del sistema se utilizó la placa Nvidia Jetson Nano, sistema embebido escogido gracias a sus características de alto procesamiento y aptitud para aplicaciones de Inteligencia Artificial.

Previo a la etapa de entrenamiento se realizó la etapa de pre procesamiento para lo cual se realizó la extracción de imágenes individuales de lechuga, se aplicó la técnica de aumento de datos mediante transformaciones geométricas, por último, se realizó la detección de bordes y el etiquetado de imágenes, obteniendo un dataset de un total de 6000 imágenes, de las cuales 3500 imágenes pertenecen a la etiqueta aptas y 2500 imágenes no aptas.

El aplicar la arquitectura de red neuronal Inception V3 con transferencia de aprendizaje permitió reducir Overfitting o sobreajuste del modelo. Esto se observa al validar el modelo mediante las gráficas de pérdida y exactitud, en las cuales se muestra que la precisión aumenta y la pérdida disminuye en función de los ciclos realizados, tanto para el conjunto de entrenamiento como para el de validación, lo que significa que el modelo cometió pocos errores en algunos datos.

Debido a la limitación que tiene la placa Nvidia Jetson Nano, se optó por realizar el sistema en dos fases, la fase de entrenamiento se realizó dentro del software de Google Colab debido a que permite ejecutar código Python en el navegador, y es adecuado para tareas de aprendizaje automático y análisis de datos sin coste adicional; mientras que para la fase de validación del modelo se lo realizó dentro de la placa Nvidia Jetson Nano.

El establecer varios escenarios en un ambiente controlado, siendo el escenario 1 un espacio que se encuentra en óptimas condiciones, preparado, abonado y con riego esporádico, y el escenario 2 un terreno más seco, sin abono y con regadío esporádico, permitió diferenciar las características que presenta el cultivo al cumplir con los requerimientos antes mencionados, obteniendo así, del escenario 1 lechugas con coloración verde oscuro, abundantes hojas gruesas, con bordes sanos y una altura de 9cm, mientras que, del escenario 2, se obtuvo lechugas de coloración verde claro-amarillo, con hojas flácidas, bordes secos y tamaño de 5 cm.

La ejecución de pruebas permitió evaluar el modelo de entrenamiento realizado en Google Colab, obteniendo como resultado una correcta detección de dos clases lechugas aptas y no aptas y un porcentaje de precisión de 97%.

Del análisis costo-beneficio se concluye que al implementar el “Sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de Visión Artificial” se obtienen grandes beneficios en cultivos de gran magnitud

donde se requiere mayor control de la producción de forma que facilita evaluar la calidad del cultivo de lechuga durante el mantenimiento y la cosecha, permitiendo una detección temprana del estrés hídrico en el cultivo y su recuperación, asegurando así una producción completa y de mejor calidad.

RECOMENDACIONES

Es recomendable que al realizar la investigación teórica se indague y analice acerca de las redes neuronales, tipos y arquitectura, con el fin de escoger una red neuronal que no consuma demasiados recursos computacionales de forma que no afecte el rendimiento del sistema.

Es importante que la adquisición de datos se lo realice en una hora del día en la que no exista demasiada iluminación, ya que esto afecta a la calidad de la imagen y disminuye la capacidad de detectar las características que posee una planta con estrés hídrico.

Es de gran importancia que una vez etiquetadas las imágenes para el proceso de entrenamiento se realice la división del conjunto de muestras en conjunto de entrenamiento, conjunto de test y validación siguiendo la regla de 70-15-15 respectivamente. Esto con el fin de evitar un sobre entrenamiento y disminución de precisión del modelo.

Es aconsejable que previo a la implementación del modelo pre entrenado a la placa embebida Nvidia Jetson Nano, se realice una gráfica de pérdida y exactitud para evaluar la evolución del modelo durante el proceso de entrenamiento, junto con pruebas de predicción utilizando imágenes del conjunto de test.

Es recomendable que las pruebas se realicen tomando en cuenta que mientras más sometido esté el cultivo a condiciones de estrés hídrico, mayor será la visibilidad del deterioro de la planta, lo que permite comprobar el nivel de precisión del sistema.

Para futuras implementaciones en terrenos extensos de mayor producción, se recomienda reemplazar las cámaras, el armazón metálico y el cableado tanto eléctrico como de red por el uso de drones, ya que esto permitiría reducir los costos de hardware e infraestructura considerablemente.

Bibliografía

Axpuac Yoc, B. R. (22 de agosto de 2020). "Agrotecnología: Una opción para la agricultura guatemalteca". *Escuela de Ingeniería en Ciencias y Sistemas - Ecys*. Obtenido de https://revistaecys.github.io/17Edicion/articulo07.html

Nikolopoulos, D., Kilpatrick, P., Barbhuiya, S., Wang, N., & Varghese, B. (20 de noviembre de 2016). "Desafíos y oportunidades en Edge Computing". *IEEE International Conference on Smart Cloud (SmartCloud)*. Obtenido de https://ieeexplore.ieee.org/abstract/document/7796149

Sánchez Galán, J. (10 de febrero de 2021). *Economipedia*. Obtenido de https://economipedia.com/definiciones/agricultura-tradicional.html#:~:text=La%20agricultura%20tradicional%20es%20el,el%20cuidado%20de%20la%20naturaleza.

Acuña Chapa, M. Á. (2020). *"Diseño de un sistema de visión artificial para la clasificación de limón utilizando Raspberry Pi".* Tesis Pregrado, Universidad Nacional de Piura, Facultad de Ciencias Escuela Profesional de]Ingeniería Electrónica y Telecomunicaciones, Piura, Perú.

Antúnez B., A., Felmer E., S., & Vidal S. , M. (2019). "Aspectos técnicos de cultivo, riego y nutrición en lechuga, tomate y melón para la zona central de Chile ". *Boletín INIA - Instituto de Investigaciones Agropecuarias*(no. 406), 33-47. Obtenido de https://biblioteca.inia.cl/bitstream/handle/20.500.14001/6808/NR41705.pdf?sequence=10&isAllowed=y

Ardila Urueña, W., Cortes, J. A., & Mendoza Vargas, J. A. (abril de 2011). "TÉCNICAS ALTERNATIVAS PARA LA CONVERSIÓN DE IMÁGENES A COLOR A ESCALA DE GRISES EN EL TRATAMIENTO DIGITAL DE IMÁGENES". *Scientia et Technica Año XVII*(no. 47). Obtenido de https://www.researchgate.net/publication/277198540_Tecnicas_alternativas_para_la_conversion_de_imagenes_a_color_a_escala_de_grises_en_el_tratamiento_digital_de_imagenes

Bagnato, J. I. (24 de diciembre de 2020). *Aprende Machine Learning*. Obtenido de https://www.aprendemachinelearning.com/aprendizaje-por-refuerzo/

Cabrera Díaz, J. (2021). *"Evaluación de cuatro cultivares de lechuga en parámetros agronomicos similares en la Granja Santa Inés"*. Tesis Pregrado, Universidad Técnica de Machala, Facultad de Ciencias Agropecuarias, Machala. Obtenido de http://repositorio.utmachala.edu.ec/bitstream/48000/16544/1/TTUACA-2021-IA-DE00010.pdf

Calvo, D. (20 de julio de 2017). *"Red Neuronal Convolucional CNN "*. Obtenido de https://www.diegocalvo.es/red-neuronal-convolucional/

Camarillo, A. (13 de mayo de 2021). *"¿Qué es el Edge Computing?"*. Obtenido de https://blog.330ohms.com/2021/05/13/que-es-el-edge-computing/

Cañadas, R. (4 de noviembre de 2021). *Ab Datum*. Obtenido de https://abdatum.com/machine-learning/redes-neuronales-artificiales

Delgado Olivera, L. d., & Díaz Alonso , L. M. (enero-marzo de 2021). "Modelos de Desarrollo de Software". *Revista Cubana de Ciencias Informáticas , Vol 15*. Obtenido de http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S2227-18992021000100037

Editorial de Botanica-online. (12 de 02 de 2021). *Botanica-online*. Obtenido de Botanica-online: https://www.botanical-online.com/alimentos/lechuga-valor-nutricional

El Comercio. (28 de mayo de 2011). *El Comercio*. Obtenido de https://www.elcomercio.com/actualidad/negocios/sierra-hay-seis-tipos-de.html

Escobar Figueroa, D., & Roa Guerrero, E. (ene-jun de 2016). "Sistema de visión artificial para la identificación del estado de madurez de frutas (granadilla). *Redes de Ingeniería, vol.7*(no.1). Obtenido de https://geox.udistrital.edu.co/index.php/REDES/article/view/10056/11679

FAO. (2021). *Organización de las Naciones Unidas para la Alimentación y la Agricultura*. Obtenido de http://www.fao.org/ecuador/fao-en-ecuador/ecuador-en-una-mirada/es/

FAOSTAT. (2022). *Organización de las NAciones Unidas para la Alimentación y la Agricultura*. Obtenido de fao.org/faostat/es/#data/QV/visualize

Fariño R, G. (2011). *"Modelo Espiral de un proyecto de desarrollo de software"*. Universidad Estatal de Milagro (UNEMI), Milagro, Ecuador. Obtenido de https://www.ojovisual.net/galofarino/modeloespiral.pdf

Fernández García, N. L. (2016). *"Visión Artificial Avanzada"*. Universidad de Córdova, Departamento de Informática y Análisis Numérico, Argentina. Obtenido de https://docplayer.es/9893156-Tema-1-introduccion-a-la-vision-artificial.html

García , E., & Flego, F. (2015). "Agricultura de precision". *Ciencia y Tecnología, vol. 8*. Obtenido de https://www.palermo.edu/ingenieria/downloads/pdfwebc&T8/8CyT12.pdf

García , J. A., & Jiménez, A. E. (11 de 06 de 2019). *Metroflor-agro*. Obtenido de Metroflor-agro: https://www.metroflorcolombia.com/el-estres-en-las-plantas/

García Peñalvo, F. J., García Holgado, A., & Vázquez Ingelmo, A. (2020). *"Ingenieria de Software I- Proceso"*. Universidad de Salamanca, Departamento de Informática y Automática. Obtenido de https://repositorio.grial.eu/bitstream/grial/1953/1/3.%20Proceso-2020.pdf

García, I., & Caranqui, V. (2015). *"LA VISIÓN ARTIFICIAL Y LOS CAMPOS DE APLICACIÓN"*. Universidad Politécnica Estatal del Carchi, Facultad de Industrias Agropecuarias y Ciencias Ambientales.

Gonzalez, L. (15 de junio de 2018). *Aprende IA*. Obtenido de https://aprendeia.com/diferencia-entre-aprendizaje-supervisado-y-no-supervisado/

GRAP. (07 de abril de 2020). *Grupo de Investigación en Agrótica y Agricultura de Presición*. Obtenido de http://www.grap.udl.cat/es/presentacion/ap.html

Grupo Inesta. (27 de junio de 2018). *Grupo Inesta*. Obtenido de https://www.grupoinesta.com/que-es-el-estres-hidrico/

Hinojosa Pinto , S. (2019). *"Diseño de una arquitectura IoT para el control de sistemas hidropónicos "*. Tesis pregrado, Universidad Politécnica de Valencia, Valencia, Esopaña.

Id Digital School. (26 de noviembre de 2021). *Id Digital School*. Obtenido de https://iddigitalschool.com/bootcamps/que-es-el-deep-learning-y-sus-aplicaciones-en-la-actualidad/

INEC. (2018). *Instituto Nacional de Estadística y Censos*. Obtenido de https://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_agropecuarias/espac/espac-2018/Manual%20del%20encuestador%20y%20supervisor%20ESPAC%202018.pdf

InfoAgro. (25 de marzo de 2020). *InfoAgro*. Obtenido de https://mexico.infoagro.com/agrotecnologia-para-un-campo-mas-productivo/

InfoAgro. (25 de marzo de 2020). *InfoAgro*. Obtenido de https://mexico.infoagro.com/agrotecnologia-para-un-campo-mas-productivo/

La Hora. (08 de febrero de 2021). "La Agricultura en Ecuador se desarrolla entre la informalidad y la subsistencia". *La Hora*. Obtenido de https://lahora.com.ec/noticia/1102340131/la-agricultura-en-ecuador-se-desarrolla-entre-la-informalidad-y-la-subsistencia

Lugo Noboa, D. J. (2021). *"Diseño de un sistema de visión artificial mediante una plataforma usando un drone para identificar la plaga lancha en campos agrícolas de tomate riñón en romerillo bajo.",.* Tesis Pregado, Universidad Técnica del Norte, Facultad de Ingeniería en Ciencias Aplicadas, Ibarra.

Massiris, M., Delrieux, C., & Fernández , A. (5-7 de septiembre de 2018). "Deteccion de equipos de proteccion personal mediante red convolucional Yolo". *Jornadas de Automática*(no. 39). Obtenido de https://ruc.udc.es/dspace/bitstream/handle/2183/24891/2018_Massiris_Manlio_Detecci%c3%b3n-equipos-protecci%c3%b3n-personal-red-neuronal-convolucional-YOLO.pdf?sequence=3&isAllowed=y

Ministerio de Agricultura y Ganadería. (24 de abril de 2020). *Ministerio de Agricultura y Ganadería*. Obtenido de https://www.agricultura.gob.ec/en-imbabura-mag-impulsa-produccion-agricola-en-huertos/

Ministerio de Agricultura y Ganadería, MAG. (09 de septiembre de 2019). *Ministerio de Agricultura y Ganadería*. Obtenido de

https://www.agricultura.gob.ec/agricultura-la-base-de-la-economia-y-la-alimentacion/

Miranda, J. (23 de mayo de 2017). *Terras de Miranda*. Obtenido de https://terrasdemiranda.es/blog/agricultura-tradicional-versus-agricultura-moderna/

Moncayo Suárez, K. F. (2022). *"Visión por computador para reconocimiento de malezas en cultivo de tomate riñon de invernadero, mediante redes neuronales"*. Tesis pregrado, Universidad Técnica del Norte, Ibarra.

Montoya Holguín, C., Cortés Osorio, J. A., & Chaves Osorio, J. A. (octubre de 2014). "Sistema automático de reconocimiento de frutas basado en visión por computador". *Ingeniare. Revista chilena de ingeniería*, *vol.22*(no.4). Obtenido de https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-33052014000400006

Nelson, J. (16 de marzo de 2020). Obtenido de https://blog.roboflow.com/labelimg/

Peláez, B. (08 de diciembre de 2017). *SofOS*. Obtenido de http://www.sofoscorp.com/impacto-tecnologia-aplicada-agricultura/

Platero, C. (2009). *"Apuntes de Visión Artificial"*. Universidad Politécnica de Madrid, Departamento de Electrónica, Automática e Informática Industrial. , Madrid, España.

Redacción España . (01 de 09 de 2020). *Tech4Business*. Obtenido de Tech4Business: https://agenciab12.com/noticia/que-es-edge-computing-beneficios

Rouhiainen, L. P. (2018). *"Inteligencia Artificial: 101 cosas que debes saber hoy sobre nuestro futuro"*. España: Alienta. Obtenido de https://static0planetadelibroscom.cdnstatics.com/libros_contenido_extra/40/39308_Inteligencia_artificial.pdf

Saavedra Del R., G. (2017). *"Manual de producción de lechuga"*. Instituto de Investigaciones Agropecuarias, Santiago, Chile. Obtenido de https://www.inia.cl/wp-content/uploads/ManualesdeProduccion/09%20Manual%20Lechuga.pdf

Sandoval, L. J. (enero - diciembre de 2018). "Algoritmos de aprendizaje automático para análisis y predicción de datos". (ITCA, Ed.) *Revista Tecnologíca ITCA-FEPADE*(num 11). Obtenido de http://redicces.org.sv/jspui/bitstream/10972/3626/1/Art6_RT2018.pdf

Senplades. (2017). *"Plan Nacional de Desarrollo 2017-2021. Toda una Vida".* Secretaría Nacional de Planificación y Desarrollo, Quito, Ecuador. Obtenido de https://www.gobiernoelectronico.gob.ec/wp-content/uploads/downloads/2017/09/Plan-Nacional-para-el-Buen-Vivir-2017-2021.pdf

Taleb, T., Di Francesco, M., & Premsankar, G. (02 de abril de 2018). "Edge Computing para Internet de las cosas: un caso de estudio". *IEEE Internet of Things Journal , vol. 5*(no. 2). Obtenido de https://ieeexplore.ieee.org/abstract/document/8289317

Torres, J. (22 de septiembre de 2019). *Torres AI*. Obtenido de https://torres.ai/redes-neuronales-recurrentes/

Valdivia Arias, C. J. (2016). *"Diseño de un Sistema de Visión Artificial para la Clasificacion de Chirimoyas".* Tesis pregrado, Universidad Católica del Perú, Perú. Obtenido de http://repositorio.concytec.gob.pe/bitstream/20.500.12390/235/3/2016_Valdivia_Dise%c3%b1o-sistema-vision.pdf

Valenzuela, I. C., V. Puno, J. C., Bandala, A. A., Baldovino, R. G., de Luna, R. G., De Ocampo, A. L., . . . Dadios, E. P. (2017). "Evaluación de la calidad de la lechuga utilizando Red neuronal artificial". *2017IEEE 9th International Conference on Humanoide*.

Valladares, F., Vilagrosa, A., Peñuelas, J., Ogaya, R., Camarero, J. J., Corcuera, L., . . . Gil-Pelegrín, E. (2004). *"Estrés hídrico: ecofisiología y escalas de la sequía ".* Ministerio de Medio Ambiente, Madrid. Obtenido de https://d1wqtxts1xzle7.cloudfront.net/40129502/Estrs_hdrico_ecofisiologa_y_escalas_de_l20151118-30238-1gyaowb-with-cover-page-v2.pdf?Expires=1628387091&Signature=OyQoPRVs-AENMJ8ijGNKLG24I2c63-

iYxEOtQnBpANIWBtgKqg3KWr6fvhzZilE24uQdNWAvSgjVx3uCjmrAmJ4O2qvfw

Viteri Vera, M. d., & Tapia Toral , M. C. (2018). "Economía ecuatoriana: de la producción agrícola al servicio". *Espacios, vol.39*(n.32). Obtenido de https://www.revistaespacios.com/a18v39n32/a18v39n32p30.pdf

Zumba Gamboa , J. P., & León Arreaga , C. A. (2018). "Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software". (U. I. Ecuador, Ed.) *INNOVA Research Journal, Vol 3*(No 10). Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=6777227

ANEXOS

ANEXO A: Formato de entrevista

Entrevista

Tema: Sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de Visión Artificial

Objetivo: La entrevista tiene el objetivo de recopilar información relevante sobre el ciclo de producción, tiempos de cosecha, cuidados, requerimientos de siembra de lechuga. El cuestionario consta de preguntas abiertas y cerradas y su participación en esta investigación es completamente voluntaria. Las respuestas obtenidas serán de uso exclusivo para el presente proyecto de titulación y en los repositorios de la Universidad Técnicas del Norte.

Fecha: 05/01/2023

Nombre de entrevistado: Rosa Gómez

Gracias por su tiempo.

1. **¿Cada que tiempo se siembra la lechuga?**

2. **¿Cómo se prepara el suelo para el cultivo de lechuga?**

3. **¿Cuánto tiempo se tarda la lechuga para ser cosechada?**

    a) 40 – 60 días

    b) 60 – 80 días

    c) 90 – 130 días

    d) Otros_____

4. **¿Cuál es la cantidad de agua que necesita el sembrío y el tiempo de riego?**

5. **¿Cuál es la cantidad de abono que se requiere durante el ciclo de producción de lechuga?**

6. **¿Cuáles son los cuidados que requiere el cultivo de lechuga?**

7. **¿Cuál es la hora recomendada para cosechar la lechuga?**

    a) Mañana

    b) Tarde

    c) Noche

8. ¿De la siembra inicial, cual es el porcentaje esperado de producción?

9. ¿Cuáles son las causas de la pérdida de la producción?

10. ¿Qué medidas se utilizan para mitigar la pérdida?

Revisado por:	**Realizado por:**
Director Msc. Fabián Geovanny Cuzme Rodríguez	Estudiante Carla Dayanara Rivera Vaca

CUESTIONARIO RESPONDIDO

Tema: Sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de Visión Artificial

Objetivo: La entrevista tiene el objetivo de recopilar información relevante sobre el ciclo de producción, tiempos de cosecha, cuidados, requerimientos de siembra de lechuga. El cuestionario consta de preguntas abiertas y cerradas y su participación en esta investigación es completamente voluntaria. Las respuestas obtenidas serán de uso exclusivo para el presente proyecto de titulación y en los repositorios de la Universidad Técnicas del Norte.

Fecha: 05/01/2023

Nombre de entrevistado: Rosa Gómez

Gracias por su tiempo.

1. **¿Cada que tiempo se siembra la lechuga?**

La lechuga se siembra mensualmente, con un tiempo de descanso del terreno de 3 semanas a un mes

2. **¿Cómo se prepara el suelo para el cultivo de lechuga?**

Previo a iniciar con la siembra es necesario tractorar o arar el terreno y se utiliza abono animal

3. **¿Cuánto tiempo se tarda la lechuga para ser cosechada?**

    a) 40 – 60 días

    b) 60 – 80 días

c) 90 – 130 días

d) Otros: La lechuga se demora en estar lista para su cosecha aproximadamente 2 meses dependiendo el tipo

4. ¿Cuál es la cantidad de agua que necesita el sembrío y el tiempo de riego?

En cultivos tradicionales se utiliza agua de sequia

En riego por goteo el agua puede ser distribuida en 6mm diarios en meses frescos y 10mm en meses cálidos en intervalos de dos a tres días

5. ¿Cuál es la cantidad de abono que se requiere durante el ciclo de producción de lechuga?

De 3 a 5 Kg por m^2 anualmente

6. ¿Cuáles son los cuidados que requiere el cultivo de lechuga?

El cultivo de lechuga no requiere mayores cuidados, se controla la cantidad de agua, el uso de abono adecuado y fumigación al cultivo

7. ¿Cuál es la hora recomendada para cosechar la lechuga?

a) Mañana

b) Tarde

c) Noche

8. ¿De la siembra inicial, cual es el porcentaje esperado de producción?

Existe una pérdida de aproximadamente el 15%

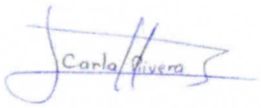
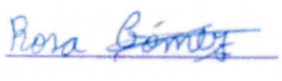
9. ¿Cuáles son las causas de la pérdida de la producción?

La producción se pierde por presencia de babosa Por exceso de sol o lluvia

10. ¿Qué medidas se utilizan para mitigar la pérdida?

Curar a la planta fumigando cada 10 o 15 días Si la planta está demasiado enferma significa pérdida, ya que es más difícil curarle

Firma Autorización

 Estudiante Carla Dayanara Rivera Vaca	 Encuestado Rosa Gómez

ANEXO B: Revisión de trabajos relacionados

Proyecto de titulación: Sistema inteligente de detección de estrés hídrico para determinar la calidad de cultivo de lechuga a través de Visión Artificial

Objetivo de análisis: Extraer información relevante de cada uno de los trabajos relacionados que permita recopilar requisitos aplicados en los diseños e implementaciones de sistemas de visión artificial.

Fecha de realización: 21 de noviembre del 2022

Trabajos relacionados:

1. "Diseño de un sistema de visión artificial mediante una plataforma usando un drone para identificar la plaga lancha en campos agrícolas de tomate riñón en romerillo bajo.". Lugo Noboa David. 2021
2. "Diseño de un sistema de visión artificial para la clasificación de limón utilizando Raspberry Pi". Acuña Chapa. 2020
3. "Evaluación de la calidad de la lechuga utilizando Red neuronal artificial". 2013
4. " Sistema automático de reconocimiento de frutas basado en visión por computador". Montoya Holguín, Cortés Osorio, & Chaves Osorio. 2014
5. Sistema de visión artificial para la identificación del estado de madurez de frutas (granadilla). Escobar D. & Roa E. 2016
6. Visión por computador para reconocimiento de malezas en cultivos de tomate riñón de invernadero, mediante redes neuronales. Moncayo Karla. 2022
7. Diseño de un Sistema de Visión Artificial para la Clasificación de Chirimoyas basado en medidas. Valdivia César. 2016
8. Implementación de un Sistema de Visión Artificial para la clasificación de naranja producida en el departamento del Quindío. Torres O, Navarro. A. 2018

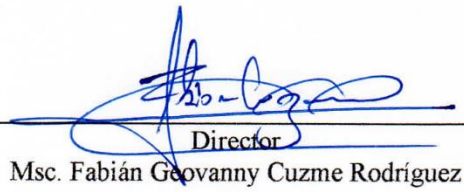
9. Sistema de clasificación por visión artificial de mangos tipo Tommy. Romero. A; Marín. A; Jiménez. J. 2015

10. Sistema de visión artificial para apoyar en la identificación de plagas y enfermedades del cultivo de sandía

Requerimiento de Stakeholders			
Nomenclatura	**Requerimiento**	**Descripción**	**Prioridad**
StSR1	Cámara de resolución mínimo de 5mpx	Para la selección de la cámara se utiliza el método FOV, estima los valores de distancia entre pixeles, pixeles activados, longitud focal, intentando acercarse lo más posible al área de interés deseado. La cámara debe tener características de alta velocidad mayor a 90fps, contar con interfaz para una rápida comunicación y transmisión de datos con el Computador. (Valdivia Arias, 2016)	**Alta**
StSR2	Aseguramiento de parámetros ópticos mediante una estructura estable para la obtención de imágenes claras	El sistema de iluminación permite independizar el proceso de captura de imágenes de día y noche, por eso es necesarios aislar la iluminación del ambiente, a una fuente de iluminación artificial. (Acuña Chapa, 2020). Las imágenes deben ser obtenidas en un periodo de tiempo donde exista una buena iluminación.	**Alta**
StSR3	Periodo de adquisición de datos, las imágenes deben ser adquiridas en horas donde existan buenas condiciones de luz	Es necesario considerar las condiciones edafoclimáticas por lo cual la hora idónea para volar es a las 6 de la mañana, obteniendo como resultado imágenes con mayores prestaciones de enfoque. (Lugo Noboa, 2021)	**Media**
StSR4	Almacenamiento de imágenes en base de datos o nube	El uso de la plataforma que integra los servicios de Amazon EC2 para ejecutar el modelo de entrenamiento y la base de datos Amazon RDS para el almacenamiento en la nube, además la transferencia de datos se realiza mediante un hotspot, optimiza el uso de recursos. (Lugo Noboa, 2021)	**Media**
StSR5	Acceso a la página web con autenticación	El usuario debe observar la información de una manera organizada y clara en gráficos para obtener los resultados, que cumpla con requerimientos de seguridad además de alertar al usuario sobre posibles plagas para tomar medidas de prevención. (Lugo Noboa, 2021)	**Alta**

Requerimiento del Sistema			
Nomenclatura	**Requerimiento**	**Descripción**	**Prioridad**
SySR1	Uso de tecnología inalámbrica	La tecnología de transmisión inalámbrica debe ofrecer una conectividad estable para la transmisión de datos, asegurar que los datos lleguen sin existir pérdidas de paquetes, tener una cobertura amplia entre 30 a 300 metros y aportar un nivel de seguridad. (Lugo Noboa, 2021)	**Alta**
SySR3	Bajo consumo de energía		**Media**
SySR4	Técnica de tratamiento de imágenes para cambio de escala de color	Las técnicas de procesamiento de imágenes son una solución que aportan velocidad y precisión para disminuir errores, además con el preprocesamiento de imágenes y técnicas de binarización de imágenes, morfología matemática hasta transformación del modelo de color RGB a HSV es posible disminuir variaciones entre pixeles. (Escobar Figueroa & Roa Guerrero, 2016)	**Alta**
SySR5	Las cámaras deben ser ubicadas en zonas despejadas para evitar obstáculos.	La adquisición de datos se realiza mediante una plataforma física similar a una báscula, con un sistema de iluminación en anillo, de forma que se anule la sombra en la imagen, además se hace uso de una cámara web de Creative Labs con una resolución de 640x480 pixeles y almacenamiento en el modelo de color RGB. (Montoya Holguín, Cortés Osorio, & Chaves Osorio, 2014)	**Media**
SySR7	El sistema debe tener una tarjeta de memoria con suficiente espacio para no saturarse.		**Alta**

Requerimiento de Arquitectura			
Nomenclatura	**Requerimiento**	**Descripción**	**Prioridad**
SrSH2	Placa embebida	Requiere de una placa embebida de versión simplificada, compatible con Linux y que permita ejecutar los algoritmos a una mayor velocidad. (Acuña Chapa, 2020)	**Alta**
SrSH6	Licencias Opensource	Es necesario escoger una librería de código abierto que esté orientada a aplicaciones de procesamiento de imágenes y que sea compatible para su interacción con diferentes sistemas operativos. (Escobar Figueroa & Roa Guerrero, 2016)	**Media**
SrSH7	Lenguaje de programación de alto nivel	Debe cumplir con características como: su soporte en tiempo real, Software libre, librerías extensas y potentes y de libre acceso. (Moncayo Suárez, 2022)	**Alta**
SrSH8	Software para tratamiento de imagen	Se aplica la técnica de segmentación del modelo RGB y HSV, además del procesamiento morfológico para eliminar imperfecciones. Las variables tomadas en cuenta son color, forma. Tamaño, área, perímetro, eje mayor, eje menor y valor medio de los canales RGB y HSV. (Montoya Holguín, Cortés Osorio, & Chaves Osorio, 2014)	**Alta**
SrSH10	Compatibilidad con Python	Uso de lenguaje de programación Python versión 3.5, junto con librerías de OpenCV. (Acuña Chapa, 2020)	**Alta**

Revisado por:	**Realizado por:**
	
Director Msc. Fabián Geovanny Cuzme Rodríguez	Estudiante Carla Dayanara Rivera Vaca

ANEXO C: Herramienta LabelImg

LABELIMG

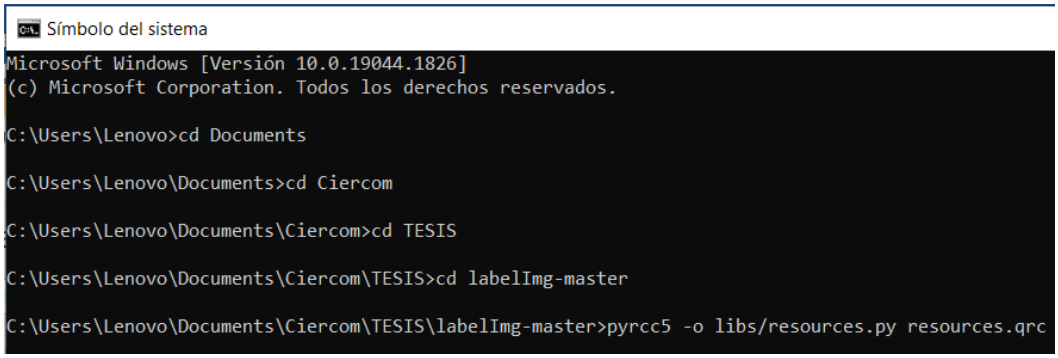
Es una herramienta gratuita de código abierto que permite etiquetar imágenes para el entrenamiento profundo de la red. Está escrito en Python y usa QT para su interfaz gráfica. (Nelson, 2020)

La información sobre el software, así como el link de descarga se lo puede obtener accediendo al repositorio de Github: https://github.com/heartexlabs/labelImg.

LabelImg establece tres requerimientos, la instalación de **Python, PyQt5** y **lxml.** A continuación, se proporcionan los links de descarga:

- **Python**: https://www.python.org/downloads/windows/
- **PyQt5:** https://www.riverbankcomputing.com/software/pyqt/download
- **Lxml:** https://lxml.de/installation.html

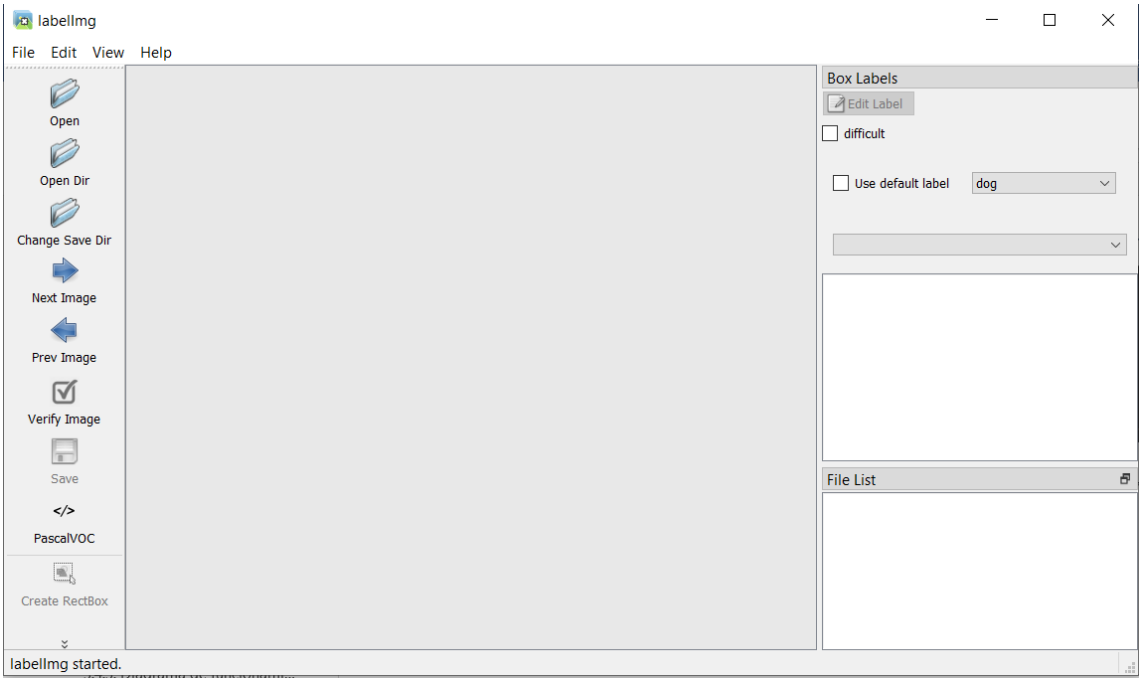
Una vez instalados los tres requerimientos, se accede al Símbolo del Sistema (CMD), y se dirige al directorio de la herramienta LabelImg y se ejecuta el comando "*pyrcc5 -o libs/resources.py resources.qrc",* que permite que un módulo Python se pueda importar a una aplicación PyQt5.



Para ejecutar la herramienta en Windows, se ejecuta el comando "*python labelImg.py"*

```
C:\Users\Lenovo\Documents\Ciercom\TESIS\labelImg-master>python labelImg.py
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen DPI.
   QT_SCALE_FACTOR to set the application global scale factor.
```

Y automáticamente se accede a la interfaz gráfica de LabelImg.



ANEXO D: Datasheet Nvidia Jetson Nano

NVIDIA.

DATA SHEET

NVIDIA Jetson Nano System-on-Module

Maxwell GPU + ARM Cortex-A57 + 4GB LPDDR4 + 16GB eMMC

Maxwell GPU ◦

128-core GPU | End-to-end lossless compression | Tile Caching | OpenGL® 4.6 | OpenGL ES 3.2 | Vulkan™ 1.1 | CUDA® | OpenGL ES Shader Performance (up to): 512 GFLOPS (FP16)
Maximum Operating Frequency: 921MHz

CPU

ARM® Cortex® -A57 MPCore (Quad-Core) Processor with NEON Technology | L1 Cache: 48KB L1 instruction cache (I-cache) per core; 32KB L1 data cache (D-cache) per core | L2 Unified Cache: 2MB | Maximum Operating Frequency: 1.43GHz

Audio

Industry standard High Definition Audio (HDA) controller provides a multichannel audio path to the HDMI interface.

Memory

Dual Channel | System MMU | Memory Type: 4ch x 16-bit LPDDR4 | Maximum Memory Bus Frequency: 1600MHz | Peak Bandwidth: 25.6 GB/s | Memory Capacity: 4GB

Storage

eMMC 5.1 Flash Storage | Bus Width: 8-bit | Maximum Bus Frequency: 200MHz (HS400) | Storage Capacity: 16GB

Boot Sources

eMMC and USB (recovery mode)

Networking

10/100/1000 BASE-T Ethernet | Media Access Controller (MAC)

Imaging

Dedicated RAW to YUV processing engines process up to 1400Mpix/s (up to 24MP sensor) | MIPI CSI 2.0 up to 1.5Gbps (per lane) | Support for x4 and x2 configurations (up to four active streams).

Operating Requirements

Temperature Range (T_j): -25 – 97C* | Module Power: 5 – 10W | Power Input: 5.0V

Display Controller

Two independent display controllers support DSI, HDMI, DP, eDP:

- MIPI-DSI (1.5Gbps/lane): Single x2 lane | Maximum Resolution: 1920x960 at 60Hz (up to 24bpp)
- HDMI 2.0a/b (up to 6Gbps) | DP 1.2a (HBR2 5.4 Gbps) | eDP 1.4 (HBR2 5.4Gbps) | Maximum Resolution (DP/eDP/HDMI): 3840 x 2160 at 60Hz (up to 24bpp)

Clocks

System clock: 38.4MHz | Sleep clock: 32.768kHz | Dynamic clock scaling and clock source selection

Multi-Stream HD Video and JPEG

Video Decode

- H.265 (Main, Main 10): 2160p 60fps | 1080p 240fps
- H.264 (BP/MP/HP/Stereo SEI half-res): 2160p 60fps | 1080p 240fps
- H.264 (MVC Stereo per view): 2160p 30fps | 1080p 120fps
- VP9 (Profile 0, 8-bit): 2160p 60fps | 1080p 240fps
- VP8: 2160p 60fps | 1080p 240fps
- VC-1 (Simple, Main, Advanced): 1080p 120fps | 1080i 240fps
- MPEG-2 (Main): 2160p 60fps | 1080p 240fps | 1080i 240fps

Video Encode

- H.265:2160p 30fps | 1080p 120fps
- H.264 (BP/MP/HP): 2160p 30fps | 1080p 120fps
- H.264 (MVC Stereo per view): 1440p 30fps | 1080p 60fps
- VP8: 2160p 30fps | 1080p 120fps

JPEG (Decode and Encode): 600 MP/s

Peripheral Interfaces

xHCI host controller with integrated PHY: 1 x USB 3.0, 3 x USB 2.0 | USB 3.0 device controller with integrated PHY | EHCI controller with embedded hub for USB 2.0 | 4-lane PCIe: one x1/2/4 controller | single SD/MMC controller (supporting SDIO 4.0, SD HOST 4.0) | 3 x UART | 2 x SPI | 4 x I2C | 2 x I2S: support I2S, RJM, LJM, PCM, TDM (multi-slot mode) | GPIOs

Mechanical

Module Size: 69.6 mm x 45 mm | PCB: 8L HDI | Connector: 260 pin SO-DIMM

Note: Refer to the software release feature list for current software support; all features may not be available for a particular OS.

◦ Product is based on a published Khronos Specification and is expected to pass the Khronos Conformance Process. Current conformance status can be found at www.khronos.org/conformance.

* See the *Jetson Nano Thermal Design Guide* for details. Listed temperature range is based on module T_j characterization.

ANEXO E: Datasheet Cámara Ezviz C8W

C8W

Detalle Especificaciones Soporte

C8W

Cámara con Wi-Fi 2K+ Pan & Tilt

Especificaciones

General

Condiciones de funcionamiento	-30 °C a 60 °C (-22 °F a 140 °F), 95% de humedad o menos (sin condensación)
Grado IP	Diseño impermeable
Alimentación	CC 12V / 1A
Consumo de energía	Máx. 6W
Dimensiones	158 x 157 x 149 mm (6.22 x 6.18 x 5.87 pulgadas)

Almacenamiento

Almacenamiento local	Ranura para tarjeta MicroSD (256 GB máx.)
Almacenamiento en la nube	Almacenamiento en la nube EZVIZ

Certificaciones

Certificaciones	UL / FCC / CE / WEEE / REACH / RoHS / UKCA

Cámara

Sensor de imágenes	CMOS de barrido progresivo 1/2.7"
Velocidad del obturador	Obturador autoadaptativo
Lentes	4mm a F1.6, ángulo de visión: 87° (Horizontal), 105° (Diagonal) 6mm a F1.6, ángulo de visión: 55° (Horizontal), 66° (Diagonal)
Ángulo de desplazamiento horizontal y vertical	Desplazamiento horizontal (pan): 352°, Desplazamiento vertical (tilt): 95°
Iluminación mínima	0.5 Lux con (F1.6, AGC ON), 0 Lux con IR
Montura del lente	M12
Día y noche	Filtro con corte IR, con activación automática
Reducción digital de ruido (DNR)	DNR 3D
Rango amplio dinámico (WDR)	WDR digital
Compensación de contraluz (BLC)	Disponible
Visión nocturna blanco y negro	30 m / 98 ft.

Audio y Video

Resolución máx.	2560 × 1440
Cuadros por segundo (fps)	30 fps máx. autoadaptativos durante la transmisión por red
Compresión de video	H.265 / H.264
Tipo de H.265	Perfil Main
Tasa de bits para video	Quad HD; Full HD; Hi-Def; estándar. Tasa de bits adaptativa.
Tasa de bits para audio	Autoadaptativa
Tasa de bits máx.	2Mbps

Red

Estándar para Wi-Fi	IEEE802.11b, 802.11g, 802.11n
Rango de frecuencias	2.4 GHz ~ 2.4835 GHz
Ancho de banda de canal	Soporta 20MHz
Seguridad	64 / 128-bit WEP, WPA / WPA2, WPA-PSK / WPA2-PSK
Velocidad de transmisión	11b: 11 Mbps, 11g: 54 Mbps, 11n: 72 Mbps
Emparejamiento de Wi-Fi	Emparejamiento por AP
Protocolo	Protocolo patentado por EZVIZ, basado en la nube
Protocolo de interfaz	Protocolo patentado por EZVIZ, basado en la nube
Cableado de red	RJ45 × 1 (puerto Ethernet con transmisión adaptativa 10 M / 100 M)