EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE DURANGO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

INSTITUTO TECNOLÓGICO DEL VALLE DEL GUADIANA

“Identificación de mosquita blanca en plantas de pepino por medio de procesamiento de imágenes”

TESIS

Que como parte de los requisitos para obtener el grado de

Maestría en Ingeniería

Presenta:

Ing. Noel Aguirre Chávez

Director de tesis:

Dr. Rubén Guerrero Rivera

Co-Director:

Dr. Francisco Javier Godínez García

Durango, Dgo. México, abril 2024

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

IDENTIFICACIÓN DE MOSQUITA BLANCA EN PLANTAS DE PEPINO POR MEDIO DE PROCESAMIENTO DE IMÁGENES

Presenta:

Ing. Noel Aguirre Chávez

COMITÉ TUTORIAL

Dr. Rubén Guerrero Rivera Director	Firma
Dr. Francisco Javier Godínez García Codirector	Firma
M. C. Eduardo Gamero Inda Asesor	Firma
M. C. José Antonio Martínez Rivera Asesor	Firma

M.C. Norma Alicia García Vidaña

Coordinadora del programa de la Maestría en Ingeniería.

Dr. Francisco Javier Godínez García

Jefe de la División de Estudios de Posgrado e Investigación

Durango, Dgo., México, abril de 2024

Oficio de autorización de tema de tesis

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Durango
División de Estudios de Posgrado e Investigación

Victoria de Durango, Dgo., a 11 / Abril / 2024.

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
DEPI / C / 051 / 24.
ASUNTO: Autorización de Tema de Tesis de Maestría.

C. NOEL AGUIRRE CHÁVEZ
No. DE CONTROL G15041042
P R E S E N T E .

Con base en el Reglamento en vigor y teniendo en cuenta el dictamen emitido por el Jurado que le fue asignado, se le autoriza a desarrollar el tema de tesis para obtener el **Grado de Maestro en Ingeniería** cuyo título es:

"Identificación de Mosquita Blanca en Plantas de Pepino por medio de Procesamiento de Imágenes"

CONTENIDO:

A T E N T A M E N T E.

Excelencia en Educación Tecnológica®
"La Técnica al Servicio de la Patria"

C. FRANCISCO JAVIER GODÍNEZ GARCÍA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE DURANGO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

FJGG'amrmc

Av. Felipe Pescador #1830 Ote. Col. Nueva Vizcaya C.P. 34080 Durango, Durango.
Tel. (618) 8250500 e-mail: tecnm.mx | itdurango.edu.mx

2024 Felipe Carrillo PUERTO

Oficio de autorización de impresión de tesis

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Durango
División de Estudios de Posgrado e Investigación

Victoria de Durango, Dgo., a 11 / Abril / 2024.

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
DEPI / C / 052 / 24.
ASUNTO: Autorización de Impresión de Tesis de Maestría.

C. NOEL AGUIRRE CHÁVEZ
No. DE CONTROL G15041042
P R E S E N T E .

De acuerdo al reglamento en vigor y tomando en cuenta el dictamen emitido por el jurado que le fue asignado para la revisión de su trabajo de tesis para obtener el **Grado de Maestro en Ingeniería,** esta División de Estudios de Posgrado e Investigación le autoriza la impresión del mismo, cuyo título es:

"Identificación de Mosquita Blanca en Plantas de Pepino por medio de Procesamiento de Imágenes"

Sin otro particular de momento, quedo de Usted.

A T E N T A M E N T E.

Excelencia en Educación Tecnológica®
"La Técnica al Servicio de la Patria"

C. FRANCISCO JAVIER GODÍNEZ GARCÍA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE DURANGO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

FJGG'ammc.

Agradecimientos

Al Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCYT) por el apoyo económico para la realización de los estudios de posgrado, a mis padres, mi familia y a mis profesores y asesores, ya que, sin su apoyo, este proyecto no hubiera podido llegar a su culminación.

Contenido

Índice de Figuras

Índice de Tablas

Resumen

Este trabajo presenta un modelo de entrenamiento de red neuronal siguiendo la arquitectura U-net para la detección de lesiones causadas por la plaga de mosquita blanca en plantas de pepino. El trabajo comenzó con la captura de fotografías documentando el crecimiento de una planta de pepino en un ambiente similar al de un invernadero. Las fotografías fueron tomadas con un teléfono inteligente Samsung A52. Las semillas de pepino fueron plantadas en macetas y las fotografías con las que se documentó el crecimiento fueron utilizadas en un procesamiento de imágenes por medio de su etiquetado usando la técnica de segmentación semántica. Estas imágenes segmentadas fueron utilizadas en un proceso de entrenamiento de un modelo de red neuronal siguiendo la arquitectura U-net para usar dicho modelo en la inferencia en nuevos ejemplos. Este modelo además fue evaluado por medio de la métrica de Intersección sobre Unión, que es la métrica preferida para la evaluación de aplicaciones de detección de objetos gracias a la comparación entre las máscaras etiquetadas y las predicciones realizadas por el modelo entrenado.

Abstract

This work presents a training model of a neural network following the U-net architecture for detection of damage caused on the leaf surface by the whitefly disease. This work started with taking photographs of a cucumber plant in an environment resembling a greenhouse with a Samsung A52 smartphone. The cucumber seeds were planted in pots and the cucumber plants contracted whitefly disease. The disease was also documented through photographs, which were used for the training of a neural network. The photographs were labeled by using the semantic segmentation method and these masks were used to train a neural network using the U-net architecture proposed by Ronneberger. The model that was trained with the photographs of the leaves affected with whitefly disease was tested with the Intersection over Union metric, which is the one preferred in object detection applications by measuring the comparison between the labeled masks and the predictions generated by the trained model.

Introducción

El desarrollo de la agricultura permitió que la humanidad se estableciera en asentamientos permanentes hace miles de años y, desde ese entonces ha sido una de las actividades primordiales para la subsistencia de la sociedad humana. Durante todo este tiempo, los cultivos siempre han sido propensos a ser afectados por situaciones externas que merman el crecimiento y el desarrollo de las plantas, como pueden ser los desastres naturales o el ataque de otros organismos, como lo son otros animales u otros patógenos.

La agricultura ha continuado su desarrollo, siempre agregando técnicas para facilitar los pasos que forman parte del proceso. Hoy en día puede ser tan simple como un cultivo casero para uso personal, o tan grande como hectáreas de terreno que producen frutos con ayuda de tecnologías diversas, entre las que destacan máquinas, riego controlado, fertilizantes y todo tipo de pesticidas. Concretamente, el uso de pesticidas e insecticidas tienen aplicación en cultivos de todo tipo y de cualquier escala, ya que la posibilidad de perder el cultivo se encuentra siempre latente y esta es una técnica que funciona de manera eficaz.

Para llegar a la selección y el uso de algún tipo de pesticida o insecticida para un cultivo siempre ha sido necesario el diagnóstico de las enfermedades que pueden afectar a los cultivos. Este diagnóstico históricamente ha sido de manera visual, con ayuda de personas que tienen conocimientos de las lesiones que las diversas enfermedades dejan tanto en los frutos como en las hojas de las plantas. El mayor problema que este tipo de inspección presenta es que se lleva a cabo por una persona viendo cada hoja, buscando la posible presencia de síntomas de alguna enfermedad. Esto puede llevar a errores derivados de una apreciación equivocada, que puede deberse incluso a situaciones como el cansancio de la persona dedicada a la inspección. Además, esta inspección requiere de conocimientos específicos, lo cual limita la velocidad de la inspección, en una tarea que debe de ser rápida, ya que el avance de las diversas enfermedades puede tener un desarrollo rápido y devastador, que provoca pérdidas.

Una técnica que ha tenido una gran cantidad de aplicaciones recientemente es el uso de técnicas de aprendizaje de máquinas para tareas que usan visión por computadora para la detección de objetos. Hoy en día existen muchas arquitecturas de redes neuronales que están

diseñadas para predecir en imágenes para aplicaciones de visión artificial y estas técnicas han sido aplicadas en detección de enfermedades en plantas recientemente.

Por lo anterior, se presenta este trabajo cuya meta fue la creación de un modelo de predicción para la detección de mosquita blanca por medio de una red neuronal y procesamiento de imágenes. Este modelo usó imágenes tomadas en un cultivo casero de plantas de pepino las cuales fueron etiquetadas por medio de una técnica llamada segmentación semántica, y entrenadas por medio de la red neuronal de arquitectura U-net para obtener un modelo de entrenamiento que puede ser usado para la inferencia en nuevos ejemplos de lesiones causadas por la mosquita blanca en plantas de pepino.

Capítulo 1. Introducción

1.1 Antecedentes

El pepino es una hortaliza que tiene un ciclo vegetativo corto, adaptabilidad de siembra y que puede prosperar en climas invernales. La temperatura ideal para su producción se encuentra en el rango de 21 y 27°C para el suelo y de 21 a 29°C de temperatura ambiente. Se pueden tener entre 3 y 5 plantas por metro cuadrado (González Acevedo, 2021).

Además, el pepino es un cultivo que ha formado parte de la dieta humana por más de 5000 años, que probablemente se dio en India en un inicio y, con el tiempo, el uso de esta fruta se expandió hacia China y el sur de Europa, desde donde se cree que Colón lo trajo a América (Tatlioglu, 1993).

En México, en 2022 se produjeron 1,004,157.79 toneladas de pepino en una superficie de 18,180.40 hectáreas cosechadas. Los estados que aportaron más a estas cantidades de producción son Sinaloa, Sonora, Michoacán, Morelos, Guanajuato, San Luis Potosí y Jalisco (SIAP, 2022).

Por otro lado, la mosquita blanca puede crear pérdidas de arriba del 50% de la producción de la planta a la que afecta. Aunque las mosquitas blancas han sido descritas por biólogos desde el siglo XVIII, no se ha podido descubrir su origen. En su adultez pueden medir arriba de 2 mm de largo, pero sus alas pueden extenderse a 4 mm. Las mosquitas se guían por el color para elegir la planta que atacarán y el espectro de color amarillo o verde son los que más atraen a las mosquitas (Byrne & Bellows, s. f.).

Por otra parte, ya que las técnicas de aprendizaje de máquinas han tenido un auge importante en los últimos años, han sido usadas en una gran cantidad de aplicaciones diferentes; sin embargo, para este trabajo se hará un énfasis en describir las técnicas de aprendizaje de máquinas que se usaron en aplicaciones relevantes para la detección de enfermedades en plantas.

Comenzando con estas aplicaciones de detección, la primera de ellas presentó una aplicación de detección de mosquita blanca usando transformaciones del sistema de color RGB a otros como CMYK o CieLAB. Con estas transformaciones de color, es posible notar la presencia

de mosquita blanca en la hoja de manera más sencilla (Barbedo, 2014). Otra aplicación similar usó también la idea de los sistemas de color, transformaciones entre estos sistemas, agregando además una comparación con la imagen original para detectar la mosquita blanca en las trampas de color que son ampliamente usadas para frenar el avance de este patógeno (Cho et al., s. f.).

Dentro de las aplicaciones que hacen uso de una red neuronal para predicción e inferencia, se ha hecho uso del algoritmo YOLO para la detección de mosquita blanca y de la mosca de fruta. La inferencia de nuevos ejemplos se llevó a cabo en un sistema que usaba una tarjeta Raspberry Pi (Legaspi et al., 2021). Otra aplicación del algoritmo YOLO se tiene en una aplicación de detección de mosquita blanca en las trampas de colores usadas como método para evitar la infección de la mosquita blanca. Este trabajo además agrega un sistema de monitoreo que brinda información acerca de los lugares afectados por la plaga para que productores tengan información de las áreas afectadas en Georgia, Estados Unidos (Parab et al., 2022).

Existe otra aplicación que usa histogramas que grafican la presencia de ciertos colores en una imagen para detectar la presencia de mosquita en las trampas de colores ampliamente usadas para la detección. Además de lo anterior, se hizo uso de una red neuronal para la predicción de nuevos ejemplos, la cual tuvo resultados equiparables a los de otros trabajos (Espinoza et al., 2016).

Una arquitectura de red neuronal que ha tenido una gran cantidad de aplicaciones se tiene en la U-net que fue presentada por Ronneberger en 2015. Desde la publicación que la describió por primera vez, se han presentado una gran cantidad de cambios a esta arquitectura. La primera que se describirá presentó cambios en las capas que la conforman para que, en lugar de usar capas de convolución, se usan ahora U-nets más pequeñas, llamado U^2-net, el cual permite quitar el fondo de las imágenes para la detección de diversas enfermedades. Esta remoción del fondo es el primer paso para la detección, mientras que la red neuronal usada es la red EfficientNetV2-M, logrando una precisión que rondaba el 80 y 90% (C. K. et al., 2022).

Otra aplicación hizo uso de la U-net para el entrenamiento de un modelo de predicción de enfermedades diversas en hojas de café. Algunas de las enfermedades para las que se hizo un entrenamiento son la minadora de hojas, oxidación y enfermedad de la mancha café. Además, el sistema incluye una aplicación de teléfono celular que permite tomar fotografías y la inferencia de estos ejemplos por medio de un servidor en la nube. Esta inferencia arroja un estimado con las áreas de enfermedad y un porcentaje de avance (Esgario et al., 2022).

Como se puede notar, las aplicaciones mostradas en esta sección hacen uso de alguna técnica de segmentación con alguna arquitectura de red neuronal ya definida. El uso de las técnicas de segmentación o de procesamiento de imágenes en conjunto con una variedad de técnicas relacionadas a las máquinas de aprendizaje, como lo son las redes neuronales, permiten la detección de objetos como lo pueden ser las enfermedades presentes en plantas. Esta conjunción de técnicas generalmente lleva a precisiones altas y resultados de predicción competentes.

1.2 Planteamiento del Problema

En general, existen estos y muchos otros antecedentes al uso de sistemas de procesamiento de imágenes para la detección de plagas y enfermedades en plantas que son económicamente productivas. Sin importar el tipo de procesamiento, todos los sistemas adoptan algún tipo de máquina de aprendizaje con la finalidad de entrenar un modelo que prediga características buscadas por los equipos de trabajo.

Otro punto que destacar se encuentra en el hecho de que mucha de la producción de pepino en México es exportada a los Estados Unidos. Este es uno de los factores que permiten que el pepino sea un cultivo con un grado importante de rentabilidad económica. Sin embargo, esta rentabilidad puede perderse debido a una gran cantidad de factores, entre los que se encuentran los patógenos.

En general, el ataque de patógenos a las plantas puede generar pérdidas económicas a los productores, y uno de los patógenos que puede generar pérdidas importantes en las plantas de pepino es la mosquita blanca. Esta posibilidad de pérdidas en la producción debido a las

enfermedades de plantas ha derivado en una gran cantidad de técnicas para frenar el avance de estas enfermedades.

De todas las técnicas, el uso de máquinas de aprendizaje como lo pueden ser las diferentes arquitecturas de redes neuronales ha obtenido buenos resultados en la tarea de detección de enfermedades en plantas. El hecho de que este tipo de detección ha sido llevado a cabo en tantas ocasiones permite pensar que también puede replicarse en el caso de detección de mosquita blanca en plantas de pepino.

1.3 Objetivos

1.3.1 Objetivo General

Por todo lo anterior, el presente trabajo tiene como finalidad crear un modelo de entrenamiento basado en la arquitectura de red neuronal llamada U-net para la detección de mosquita blanca en plantas de pepino. Este modelo de entrenamiento puede usarse para aplicaciones de detección que permitirán tomar medidas

1.3.2 Objetivos Específicos

Los objetivos específicos de este proyecto serán los siguientes:

- Investigar las características comunes de la mosquita blanca en plantas de pepino
- Hacer un plantío casero de plantas de pepino y monitorear su crecimiento
- Infectar la planta con mosquita blanca y tomar fotografías de las lesiones
- Etiquetar las imágenes de acuerdo con el nivel de severidad de la lesión, con 3 niveles, así como 2 clases adicionales para la hoja y el fondo
- Usar la arquitectura de red neuronal U-net para el entrenamiento de un modelo de predicción
- Comprobar la eficiencia del modelo de entrenamiento por medio de la métrica de intersección sobre unión

1.4 Justificación

Cualquier enfermedad que se presente en cultivos que son usados para consumo humano tiene el potencial de representar grandes pérdidas para el productor, sin importar la escala de esta producción. La mosquita blanca es una plaga que se presenta en una gran cantidad de plantas y, por esta razón, existen técnicas para frenar o mermar su avance. Además, el uso de técnicas de visión artificial puede llevar a una detección más eficiente y temprana, tanto para productores de nivel industrial, como para productores caseros.

Como se comentó previamente, durante el año 2022, México produjo 1,004,157.79 toneladas de pepino en una superficie de 18,180.40 hectáreas (SIAP, 2022). Mucha de esta producción es exportada a Estados Unidos donde, las importaciones de vegetales solo han crecido desde la firma del Tratado de Libre Comercio de América del Norte (TLCAN). Otros factores que influyen en este aumento de importaciones en Estados Unidos son la tasa de cambio entre dólares y pesos, la diferencia en los costos de producción entre los dos países y los apoyos gubernamentales provistos por el gobierno mexicano para la producción y exportación de vegetales. Durante las recientes negociaciones para el Tratado México Estados Unidos Canadá (T-MEC), se esperaba una mayor transparencia para los apoyos que otorga México, pero esto igual ha desencadenado desacuerdos entre ambos países (Huang et al., 2022).

Además de lo anterior, para el caso de nuestro país, hay una cantidad de razones para preferir la agricultura protegida, como lo son para aumentar la producción de los cultivos y que estos cumplan con las normas necesarias para su exportación al mercado internacional. Esto va de la mano con la cantidad de apoyos que el gobierno mexicano ofrece para la producción; sin embargo, estos apoyos se otorgan a los productores que cumplen con criterios de viabilidad financiera, así como factores externos como el clima, el control de plagas y enfermedades, entre otros. En este trabajo, además se consideró que en una unidad de producción de agricultura protegida de 1000 m^2 obtuvo ingresos de $447,200 pesos, de los cuales las ganancias netas representaron un 19.5% de esta cantidad, descontando el costo de la tierra usada para este cultivo (Ramírez Abarca & Hernández Martínez, 2021).

Además, en años recientes, la empresa Gromich S. de P. R. de R. L. de C. V. ha trabajado el cultivo de pepino por medio de contratos con productores donde se comprometen a la compra de toda la producción para su posterior exportación a Estados Unidos o su venta para consumo dentro de México, siempre y cuando se encuentre dentro de los estándares convenidos. De acuerdo con este trabajo, lo anterior representa una ventaja para los productores, ya que así cuentan con un comprador al final de la cosecha con un precio asegurado, así como acceso a asesorías técnicas acerca del cultivo. Este trabajo también concluye que la producción de pepino tiene costos proporcionales al tamaño del invernadero usado. Además, estos costos varían entre diversas regiones de cultivo, ya que se encontraron pequeñas diferencias de costos entre un cultivo en el estado de Morelos y otro en el Estado de México (Rebollar-Rebollar et al., 2022).

Con base en la información anterior, se puede comentar que la producción de pepino en México en tiempos recientes ha tenido grandes ganancias debido a la posibilidad de exportar la producción principalmente a Estados Unidos y a los apoyos que el gobierno de México ofrece a productores. Esta exportación ha generado competencia con productores de Estados Unidos, ya que los costos de producción en México son menores.

Sin embargo, este alto nivel de rendimiento y de producción logrado por los productores mexicanos es posible gracias al uso de agricultura protegida en conjunto con el uso de fertilizantes y pesticidas. La cifra de ingresos propuesta por Abarca tiene incluida el uso de estos productos ya que son prácticamente necesarios para un cultivo exitoso. Sin embargo, en otras regiones como lo son Nicaragua, el cultivo de hortalizas como lo puede ser el pepino no tiene rendimientos tan altos y los frutos son de baja calidad, lo que disminuye su valor. Además, prácticas inadecuadas de parte de los productores han provocado un incremento en problemas derivados de la plaga, como lo puede ser la mosquita blanca (Elizabeth Castillo, 2015).

Con base en lo anterior, se puede concluir que el pepino es un cultivo de alta importancia gracias a las ganancias económicas que puede obtener. México cuenta con una cantidad de ventajas que permiten al pepino ser un cultivo de exportación, entre las que se pueden citar

la cercanía con los Estados Unidos y la gran diferencia en los costos de producción, debida en parte a los costos de la mano de obra.

Aun cuando la rentabilidad del pepino es una realidad, también se deben tener en cuenta las posibilidades de pérdidas, las cuales implican desastres naturales y el ataque de enfermedades y plagas, como lo puede ser la mosquita blanca. La mosquita blanca es una de las plagas que pueden generar grandes pérdidas, sin importar el tamaño del productor.

Por esta posibilidad de pérdidas, en conjunto con el auge de aplicaciones de visión artificial en años recientes, se ve como una opción viable el uso de inteligencia artificial para la detección de enfermedades y plagas como lo es la mosquita blanca en plantas de pepino. Estas aplicaciones de detección han sido probadas en una gran cantidad de plantas para detectar una cantidad de enfermedades obteniendo en la mayoría de los casos una detección competente, la cual puede ayudar tanto a productores a gran y menor escala, como los descritos en la sección, así como para personas con cultivos completamente caseros.

Para el caso de cultivos industrializados, la detección puede convertirse en una tarea más simple y menos tediosa para los jornaleros presentes en el cultivo; mientras que, para cultivos caseros, un modelo de detección puede ayudar a gente sin conocimientos de plagas y enfermedades a detectar la enfermedad. En ambos casos, sin importar si se tiene conocimiento de la plaga o no, la meta es una detección rápida y eficaz con la finalidad de reducir el tiempo de reacción y evitar el avance de la mosquita blanca en una planta de pepino.

Esta detección se hará por medio de un modelo de entrenamiento de red neuronal, programada en el lenguaje Python por su facilidad de uso. Este modelo de entrenamiento será la pieza clave en futuras aplicaciones que usen el modelo para detección en tiempo real y esta detección puede ser el factor que reduzca el tiempo de acción desde la detección de la presencia de la plaga hasta la toma de medidas preventivas, como lo es el uso de pesticidas u otras técnicas para frenar el avance de la mosquita blanca. Esta detección puede llevar a la reducción de pérdidas causadas por enfermedad, lo cual mejorará los rendimientos que tienen los cultivos de pepino, lo cual tiene ventajas desde el lado económico para productores de pequeña y gran escala.

1.5 Hipótesis

La hipótesis nula de este trabajo es que la técnica de procesamiento de imágenes conocida como segmentación semántica y la red neuronal U-net no generan predicciones certeras del avance de la plaga de mosquita blanca en plantas de pepino basado en las lesiones presentes en la superficie de la hoja.

Por otro lado, la hipótesis alternativa es que la técnica de procesamiento de imágenes conocida como segmentación semántica y la red neuronal U-net pueden trabajar en conjunto para obtener predicciones certeras del avance de la plaga de mosquita blanca en plantas de pepino basado en las lesiones presentes en la superficie de la hoja.

1.6 Alcances y Limitaciones

El objetivo final de este trabajo es la obtención de un modelo de predicción que pretende usar la red neuronal U-net e imágenes clasificadas por medio de la técnica de segmentación semántica para la detección del avance de la plaga de mosquita blanca basada en las lesiones que la plaga deja en la hoja.

Aunque el modelo de entrenamiento requiere de una aplicación formal para poder hacer detecciones el tiempo real, esta aplicación no se desarrollará ni forma parte del proyecto, por lo que el alcance de este trabajo es concretamente el entrenamiento del modelo y ciertos cálculos de mediciones para comprobar su funcionamiento correcto.

Aun cuando solo se llevará a cabo el entrenamiento de un modelo de red neuronal, se espera que este modelo pueda ser usado por otros proyectos, ya sea que lleven a cabo la aplicación para llevar a cabo la detección o hacer uso de más imágenes de entrenamiento para proveer robustez al modelo.

Se obtendrán las imágenes del cultivo de pepino de un plantío casero que fue afectado por la plaga de la mosquita blanca. En un inicio se esperaba que las imágenes fueran provistas por algún productor, pero esto no fue posible. Sin embargo, el modelo puede ser entrenado con

imágenes provenientes de un cultivo casero y este modelo puede usarse para llevar a cabo la inferencia en aplicaciones con más plantas, o en un ambiente de producción de pepino. Además, ya que la toma de fotografías de la planta de pepino es una tarea que forma parte de los objetivos de este trabajo, ahora se puede tener un control con la calidad de las imágenes, los ángulos en los que las imágenes son tomadas, y también se puede definir una cantidad de imágenes tomadas para el posterior etiquetado y entrenamiento del modelo.

En este trabajo, una limitación importante radica en el procesamiento necesario para el entrenamiento de la red neuronal. El entrenamiento de redes neuronales en aplicaciones que trabajan con visión artificial requiere de grandes cantidades de procesamiento. Por esta razón, si el entrenamiento se lleva a cabo en una computadora de gama media, sin una unidad de procesamiento gráfico dedicada, el entrenamiento tomará una cantidad de tiempo considerable. Como solución a esta limitación se tiene el uso de servicios como lo puede ser Google Colab, que es una plataforma que se usará para ejecutar el código en Python en un servidor, el cual puede hacer uso de una unidad de procesamiento gráfico, o GPU, por sus siglas en inglés. La limitación inherente que tiene Google Colab radica en que se tiene un tiempo limitado para el uso de la GPU que se ofrece sin costo adicional. Usar la GPU en la modalidad gratuita de Google Colab ayudará a reducir el tiempo que el entrenamiento de la red neuronal toma, reduciéndolo a un tiempo tolerable aun cuando no se esté pagando por una cantidad mayor de procesamiento.

Finalmente, otras limitaciones presentes en este trabajo se encuentran en la precisión del modelo, la cual dependerá de la cantidad de épocas, de imágenes usadas en el entrenamiento, del tamaño de lote de entrenamiento, y de las funciones de optimización y pérdida usadas para el entrenamiento. Concretamente, el tamaño de lote y la cantidad de épocas de entrenamiento deben tener números razonables ya que también se puede superar la cantidad de memoria RAM designada para el usuario de Google Colab, lo cual provoca un reinicio del entorno de programación. Por esta razón, tanto la cantidad de épocas como el tamaño de lote durante el entrenamiento deben tener números contenidos.

1.7 Estado del Arte

Diversos autores generalmente distinguen dos especies de mosquita blanca que generan afectaciones a una gran cantidad de cultivos. La primera de estas especies es conocida como Bemisia tabaci, mientras que la otra es Trialeurodes vaporariorum. Se ha comentado acerca de las diferencias morfológicas entre ambas especies de mosquita blanca, las cuales radican en el tipo de huevos que ponen, y mediciones de tamaños en cada una de las etapas de desarrollo físico. Por otro lado, ambas especies pasan por 6 etapas de crecimiento, que son el huevo, 4 estados intermedios de instar, y la etapa adulta (Carapia Ruiz & Castillo-Gutiérrez, 2013).

Además, otro trabajo recabó además los datos de las temperaturas y los días promedio que tarda cada fase de desarrollo de la mosquita blanca en tablas. En estas tablas, se tiene que las mosquitas blancas de la especie Trialeurodes vaporariorum tardan 23 días desde que se pone el huevo hasta la etapa adulta en plantas de pepino a 24°C, mientras que para Bemisia tabaci, el número reportado fue de 20.3 días de crecimiento, en este caso a una temperatura de 25.5°C. Este mismo trabajo compila ecuaciones de regresión para el cálculo de porcentaje acumulado de individuos en cada fase expresado en días, así como estadísticas de la eficacia de diversos tratamientos químicos contra esta plaga (Cabello García et al., 1996).

Las mosquitas blancas son insectos que consumen azúcar los cuales se originaron en el sur de Asia y han llegado a todas las regiones del mundo, siendo las regiones tropicales las que más sufren. La plaga de mosquita blanca provoca grandes pérdidas a la agricultura mundial (Abubakar et al., 2022). La mosquita blanca ataca a las plantas de dos maneras diferentes. La primera de ellas es por medio de alimentarse de la savia presente en la planta, lo cual afecta la vitalidad de la planta; mientras que la segunda es provocando una interferencia en la fotosíntesis debido al crecimiento de hongos en las áreas donde las mosquitas secretaron melazo. Además de esto, las mosquitas blancas son vectores de otros patógenos virales de plantas que provocan pérdidas en la producción (Ghongade & Sangha, 2021).

El ciclo de vida de la mosquita blanca comienza cuando los ejemplares hembra ponen huevos, los cuales eclosionan entre 5 y 9 días después de poner los huevos. El desarrollo físico de las mosquitas pasa por una cantidad de fases, hasta llegar a la fase donde se convierten en ninfas.

Las ninfas son estacionarias, y de la pupa de la ninfa nace la versión adulta de la mosquita blanca. En este punto de la vida de la mosquita, las hembras tienen un estómago grande y redondo, mientras que en los machos tiene un pico (Abubakar et al., 2022). Los ejemplares hembra de mosquita pueden vivir entre 10 y 24 días, en los cuales puede poner entre 66 y 300 huevos, dependiendo de la planta y la temperatura. Una hembra puede vivir hasta 60 días, pero los ejemplares macho viven entre 9 y 17 días (Ghosh, 2022).

Aunque el método de control de la plaga de mosquita blanca se basa en el uso de insecticidas, es difícil controlar la plaga a base de químicos ya que los ejemplares adultos y los que se encuentran en etapas de crecimiento generalmente se esconden en el reverso de las hojas de la planta, por lo que el tratamiento químico no está en contacto con ellas. Debido a esta situación, en conjunto con la resistencia de la mosquita blanca a ciertos insecticidas, se ha optado por encontrar enemigos naturales a la plaga, como lo son depredadores, parasitoides y patógenos fúngicos (Golshan et al., 2023).

La mosquita blanca es una plaga que ataca a una gran cantidad de cultivos y que puede provocar pérdidas importantes, ya que es difícil controlar la plaga usando solamente tratamientos químicos. Por esta razón, se ha optado en algunos casos por el uso de tratamientos biológicos, que se basan en poner a los depredadores naturales en el mismo ambiente donde se encuentra la mosquita, así como el uso de trampas de color, la cual es una técnica altamente eficaz. Esta técnica de las trampas de color está basada en la condición de que las mosquitas blancas son atraídas por plantas de color verde o amarillas, que constan el espectro de color que las atraen más, de acuerdo con (Byrne & Bellows, s. f.).

Una aplicación del uso de trampas de color se tiene en un trabajo que usó flores de crisantemo artificiales cuyos pétalos estaban hechos de las trampas de color amarillo con pegamento para atrapar mosquitas blancas de la especie Trialeurodes vaporariorum (Mainali & Lim, 2008). Este trabajo se llevó a cabo en plantas de tomate, las cuales son presa de la mosquita blanca comúnmente.

Se puede deducir de lo escrito en esta sección que la mosquita blanca es una enfermedad que ataca una cantidad importante de plantas y cultivos que son ampliamente consumidos en el

mundo y que, por ende, tienen un valor económico importante. Dentro de estos cultivos se encuentra el pepino, el cual es un cultivo propenso a la plaga de mosquita blanca y, aunque existen métodos diversos para mermar el avance de esta plaga, la tarea de detección de la plaga se convierte en una tarea de gran importancia.

Para satisfacer esta necesidad de detección de plagas o enfermedades en plagas, las aplicaciones de visión artificial han destacado recientemente como una solución que puede ser eficiente y asequible para productores de pequeña y gran escala. En general, estas aplicaciones de visión artificial aplican una técnica de segmentación o de procesamiento de imágenes y el uso de una arquitectura de red neuronal, para su posterior evaluación o su aplicación como parte de un sistema de detección más elaborado. A continuación, se presentan algunos de estos trabajos.

El primero de estos trabajos consistió en una aplicación de detección de diversas enfermedades en tomate, entre las que se encuentra la mosquita blanca, usando 4 algoritmos de red neuronal diversos, entre los que se encuentran YOLO v3 y una versión mejorada de este algoritmo. Esta versión logró una precisión de 92.39% y un tiempo de detección de 20.39 ms, los cuales son resultados competentes que permiten su uso en aplicaciones en tiempo real (Liu & Wang, 2020).

Otro trabajo usó un tipo de red neuronal conocida como redes adversarias generativas para aumentación de datos de un conjunto de datos consistente de imágenes de mosquita blanca en una serie de cultivos diferentes. Además de YOLO v3, y una versión más pequeña llamada YOLOv3-tiny, este trabajo usó una red basada en YOLO que inicialmente se había usado para la detección de esperma que para este trabajo llevó el nombre de PestNet. Esta red obtuvo resultados buenos de intersección sobre unión, así como un tamaño de modelo de 14 MB, lo cual es incluso mejor que el modelo obtenido por YOLOv3-tiny (Karam et al., 2022).

También se han comparado arquitecturas de red neuronal, entre las que está YOLOv4 para la detección de cochinillas, Coccidae y Diaspididae. Este trabajo también usó la técnica de aumentación de datos para incrementar la cantidad de imágenes de ejemplo de entrenamiento. Como resultado, se resalta el hecho de que, de las 3 arquitecturas probadas, YOLOv4 fue la

mejor de acuerdo con la medida F1, que es usada para comprobar el correcto funcionamiento de estos modelos. Además, este proyecto agregó una aplicación para teléfonos celulares con la finalidad de su uso en tiempo real. Los autores concluyeron que YOLOv4 es una arquitectura que puede implementarse en aplicaciones de detección de objetos en tiempo real (Chen et al., 2021).

El algoritmo YOLO ha sido una de las formas más importantes para llevar a cabo aplicaciones de detección de objetos por medio de visión artificial y este algoritmo sigue recibiendo mejoras con cada una de sus versiones subsecuentes, aun cuando, como lo muestran los trabajos descritos previamente, YOLO ha tenido un gran desempeño sin importar la versión que se use. Como lo muestra la literatura descrita acerca de YOLO, este algoritmo usa etiquetas en la forma de cajas rectangulares para la detección de objetos.

Así como el algoritmo YOLO hace uso de cajas rectangulares para el etiquetado, la técnica de segmentación semántica es la predilecta para su uso en las redes neuronales convolucionales y, sobre todo, las redes neuronales U-net. La segmentación semántica ha proliferado como una técnica de etiquetado para su uso como paso previo al entrenamiento de redes neuronales convolucionales para que predigan características de imágenes. La idea detrás de la creación de estas técnicas se basa en la creación de máscaras para aislar alguna característica de una imagen y repetir el proceso hasta aislar las partes que se quieren destacar de una imagen. Autores definen a la segmentación semántica como una tarea de alto nivel que provoca el entendimiento de una escena por completo. Este entendimiento es el que permite el uso de diversas arquitecturas de aprendizaje profundo, entre las que están las redes neuronales convolucionales. La segmentación semántica consiste en la creación de etiquetas para cada uno de los pixeles que conforman una imagen, de tal manera que se puede clasificar cada píxel que forma parte de una imagen, ya que cada pixel es etiquetado de acuerdo a la clase a la que pertenece (Garcia-Garcia et al., 2017). A continuación, se muestran algunos trabajos que han usado la segmentación semántica para la clasificación de datos.

En la primera de estas aplicaciones, se creó un programa de código abierto que puede ser consultado y ejecutado para la creación de redes neuronales convolucionales que usen el algoritmo de segmentación semántica para el procesado previo de las imágenes. Este

programa es conocido como Bonnet y se encuentra documentado en internet en el servicio GitHub, así como en otras páginas de internet, entre las que se muestra un video musical con la segmentación semántica de las personas a cada cuadro del video (Milioto & Stachniss, 2019).

Otro trabajo consistió en un proyecto que culminó en una red neuronal convolucional para la segmentación semántica. El proceso usado es similar al que se ha descrito previamente, pero este método ha logrado una precisión decente en un tiempo de procesamiento menor a un décimo de segundo. Este sistema puede variar la calidad de la predicción por medio de cambiar la cantidad de pixeles en las "stride nets" que utiliza, donde mientras este número sea mayor, la segmentación tendrá menor resolución (Shelhamer et al., 2016).

En el siguiente de estos trabajos consistió en el uso de la técnica de segmentación semántica para clasificar imágenes de hojas de tomate afectadas por la plaga de minadora de hojas. Esta clasificación dividió las imágenes, dándole clases al fondo, a la hoja y a las áreas de la hoja lesionadas. Con estas imágenes segmentadas, se usaron una cantidad de redes neuronales convolucionales, donde la U-net obtuvo el mejor desempeño para segmentar el fondo y la hoja. Como conclusión, este trabajo considera que el uso de estas combinaciones de técnicas puede ser implementado en campo, ya que tuvo un buen desempeño con fondos complejos e iluminación irregular (Martins Crispi et al., 2023).

Aunque la segmentación semántica ha sido usada en conjunto con diversas arquitecturas de red neuronal, una combinación entre la segmentación semántica y las diversas arquitecturas basadas en la red neuronal U-net, desarrollada por Ronneberger en 2015, ha sido usada en un sinfín de aplicaciones de visión artificial, obteniendo una detección eficiente.

Este trabajo de Ronneberger comenzó por mejorar una red completamente convolucional para que funcione con pocos ejemplos de entrenamiento y arroja segmentaciones más precisas. Las mejoras aplicadas llevaron a la creación de una red neuronal con una gran cantidad de canales de características, para propagar información de contexto a las capas con mayor resolución. La red entonces usa un camino de contracción de funciones, para luego

llegar a un camino de expansión. Estos dos caminos son lo que le da la forma de u a la arquitectura de la red que le otorga su nombre.

Además de definir la arquitectura U-net, Ronneberger la usó en imágenes de células que forman parte de un reto de seguimiento de células de parte de ISBI, logrando precisiones de 92% con un conjunto de imágenes y de 77% con otro. Esto además implicó un proceso de aumentación de datos para cada uno de los conjuntos de datos usados. Y, juzgando por el hecho de que la U-net es ahora una arquitectura ampliamente usada, la cual ya se encuentra lista para desplegarse en librerías de Python, se puede concluir que es una forma eficiente para aplicaciones de visión artificial (Ronneberger et al., 2015).

Como se comentó, se atribuye a Ronneberger la creación de la U-net y este trabajo ha generado nuevas aplicaciones y mejoras sobre este modelo original. Una de estas mejoras consistió en cambiar las capas de convolución dentro de la U-net por convoluciones separables. Chollet probó estas mejoras con la red Inception por medio de un conjunto de datos que pertenece a Google llamado JFT, que cuenta con 350 millones de imágenes y 17 mil etiquetas. El modelo Xception logró una precisión mejor que la de otros modelos con las que Chollet comparó su modelo, dentro de las que se encuentra la red VGG-16, que fue una de las primeras redes neuronales convolucionales que destacó por su eficiencia (Chollet, 2017).

Continuando con otras aplicaciones que han hecho uso de la U-net, se tiene una combinación entre una versión de la U-net y la ResNet para extracción de características para llevar a cabo experimentos de clasificación de árboles en un bosque de China basándose en su forma vista desde el aire. Los resultados de esta aplicación arrojaron una eficiencia de alrededor del 87% (Cao & Zhang, 2020). Además, esta aplicación también usó la segmentación semántica como paso previo al uso de la U-net modificada que se usó para el entrenamiento, lo cual es parte de la metodología que se usará para este trabajo.

En otra aplicación, se decidió usar la U-net con la finalidad de detectar la presencia y el daño causado por el COVID-19, con la finalidad de obtener diagnósticos rápidos y eficientes para evitar el uso de pruebas PCR. El modelo comparó la U-net usando diversas combinaciones

de funciones de activación, optimización y pérdida, para obtener una eficiencia cercana al 94%. En este caso, las diversas combinaciones y modificaciones realizadas a la U-net le permitieron al modelo ser una buena opción para detección de COVID-19, ya que solo necesita imágenes obtenidas por medio de un tomógrafo (Saeedizadeh et al., 2021). Esta es una aplicación creativa que probablemente debería extrapolarse a la detección de más enfermedades.

En otra aplicación de la red neuronal U-net, se usó para la detección de tumores cerebrales. Esta tarea de detección es complicada debido a que la delimitación entre el área sana del cerebro y el área donde se encuentra el tumor debe ser precisa. Las imágenes provienen de un conjunto de datos que fue usado para la detección de tumores y la precisión del modelo estuvo entre 0.75 y 0.92 (Lin et al., 2021).

En el siguiente trabajo que se desea destacar, se consiguió una precisión y una intersección sobre unión de cerca de 98% en una aplicación de detección de enfermedades presentes en imágenes de tomate obtenidas del conjunto de datos llamado Plant Village. Esta precisión se logró usando la red neuronal U-net, pero este trabajo usó otras arquitecturas, con las cuales se logró una precisión ligeramente superior (Chowdhury et al., 2021).

En una aplicación más, se entrenaron modelos de redes neuronales convolucionales para la detección de una cantidad de diversas enfermedades presentes en las plantas de pepino, como lo pueden ser el oídio y el mildiú. Además, probaron la detección en los casos donde se encuentran más de una enfermedad presente, lo cual de acuerdo con los autores es un aspecto innovador de este trabajo. En general, la precisión de las enfermedades, así como de diversas combinaciones de enfermedades dio como resultado una precisión de 95% (Tani et al., 2018).

Una simple búsqueda en línea acerca de las aplicaciones recientes que las redes convolucionales han tenido permitirá ver la gran cantidad de ambientes en los que la implementación de una red neuronal convolucional ha ayudado en tareas de visión artificial o de detección de objetos. Este uso para las redes neuronales ha estado en auge, comenzando con el algoritmo YOLO y, a continuación, con todas las otras arquitecturas que han sido creadas para estas tareas de detección.

Como se ha comentado en esta sección, la detección de enfermedades en plantas se ha hecho una tarea muy importante para poder cumplir con producciones para exportación y para consumo. Esta detección temprana da pie a tomar acciones de manera más eficiente para frenar el avance de las enfermedades y reducir la cantidad de pérdidas dadas al ataque de una enfermedad o de alguna plaga.

Capítulo 2. Marco Teórico

2.1 Pepino

El pepino proviene de la India y probablemente se originó en las faldas de los Himalayas. Aproximadamente hace 3,000 años se cultivó en la India y, se expandió rápidamente al oeste de Asia y al sur de Europa. Su expansión se dio al resto del mundo, al grado de que hoy en día, es cultivado en regiones templadas y tropicales en todo el mundo, y es el cuarto vegetal más consumido del mundo, solo detrás del tomate, repollo y la cebolla (Jia & Wang, 2021). La expansión del consumo de pepino al resto del mundo se dio gracias a los romanos, que llevaron la fruta a Grecia e Italia en el siglo II a.C. y llegó a Francia y Bretaña en los siglos IX y XIV, respectivamente. Los españoles lo llevaron a Haití en 1494 y a lo largo del siguiente siglo, llegó al resto de América (Staub et al., 2008).

El pepino es además una planta que tiene raíces poco profundas. Los tallos tienen forma similar a la de un rombo y, desde el tallo se generan ramas nuevas, cuya cantidad varía dependiendo de la variedad. Los frutos que da son cilíndricos y esféricos; cada fruta tiene entre 100 y 400 semillas (Jia & Wang, 2021).

El pepino tiene una gran cantidad de usos. Dentro de los usos cosméticos, la frescura del pepino permite que la piel esté suave y clara, así como para tratar arrugas. Esto es debido a que emana naturalmente ácido glicólico, láctico y salicílico, los cuales promueven la remoción de células muertas para mantener las capas superficiales de la piel saludables. Los usos culinarios que estos autores destacan su uso en una gran cantidad de platillos, entre los que están ensaladas, sopas y licuados. Sin embargo, la aplicación culinaria más importante es en pepinillos, que son pepinos fermentados. Los usos medicinales del pepino implican beneficios en la hidratación, manejo del peso corporal, salud de los huesos, manejo del nivel de azúcar en la sangre, el potencial para evitar el cáncer y beneficios cardiovasculares (Ugwu & Suru, 2021).

El pepino es usado en casi todo el mundo para su consumo en estado inmaduro, y en menor medida en la forma de conservas. En este caso, el término estado inmaduro se refiere a los

pepinos rebanados y usados para su consumo, mientras que el estado en conserva se refiere mayormente a los pepinillos (Barraza-Álvarez, 2015).

Finalmente, dentro de las propiedades del pepino, se puede comentar que contiene un 96.4% de agua, 2.8% de carbohidratos, 0.4% de proteína, 0.13% de grasa y 0.8% de fibra. Además, cuenta con las vitaminas del complejo B y la vitamina C, donde se tiene un rango de entre 7.5 y 18.1 mg/100g de fruta fresca (Zieliński et al., 2017).

2.1.1 Cultivo de pepino

El pepino es un cultivo resistente a las heladas, pero que crece en mejores condiciones con temperaturas arriba de los 20°C, así como con grandes cantidades de luz, de humedad en el ambiente y en el suelo. Además, el uso de fertilizantes propicia su crecimiento en invernaderos. El pepino puede ser plantado por medio de trasplante o llevando a cabo la plantación directamente. Los espacios entre las filas de plantas van de 120 a 150 centímetros y los espacios entre plantas están entre 30 y 45 centímetros (M. C. Singh et al., 2017). Es necesario un suelo bien drenado y con alta porosidad, la cual se logra por medio de uso de materia orgánica y compostas. Los suelos con un pH de entre 5.5 y 6.5 son aptos para la producción del pepino. Además, para el crecimiento correcto de la planta se requieren temperaturas de entre 13 a 25°C. La planta tiene una tolerancia de temperatura máxima de entre 38 y 40°C (Pal et al., 2020).

Además de los requerimientos para el crecimiento de pepino mostrados previamente, también se necesita un periodo de oscuridad diario para un crecimiento óptimo. En el caso del pepino, se ha comprobado que el uso de iluminación adicional para evitar un decremento en la temperatura ha tenido efectos adversos. El uso de este tipo de iluminación por una semana provocó un rendimiento menor de la planta al momento de dar fruto (M. C. Singh et al., 2017).

El rendimiento del cultivo de pepino ha sido un tema importante por más de 50 años. Estadísticas revelan que entre 1949 y 1979 la producción de pepino pasó de 4,685 kg/ha a 11,455 kg/ha en Estados Unidos y mucho de este incremento se debe a mejoras en las

prácticas culturales y la creación de cultivos resistentes. Este crecimiento en la producción provocó investigaciones, que implican mejoras en la cruza y optimizar el rendimiento, con miras a elegir los cultivos que den el mayor rendimiento.

El tamaño de los frutos también es importante. En Estados Unidos, el pepino es clasificado de acuerdo con su tamaño, donde los frutos más pequeños implican un precio mayor. La relación entre largo y diámetro también es importante y, para el mercado estadounidense esta relación se encuentra entre 2.9 a 3.3. O sea, el largo debe superar por alrededor de 3 veces el diámetro de cada pepino. (Staub et al., 2008).

2.2 Mosquita Blanca

Como se comentó en secciones previas de este trabajo, existen dos especies predominantes de mosquita blanca. Estas especies son Bemisia tabaci y Trialeurodes vaporariorum. B. tabaci fue descrita por Gennadius en 1889, como una plaga de tabaco en Grecia (Sani et al., 2020), mientras que Trialeurodes vaporariorum fue descrita por Westwood en 1856 al encontrarla en invernaderos en el Reino Unido (Van Lenteren et al., 1996).

Existen comparaciones entre una mosquita blanca de la familia Bemisia argentifolii y Trialeurodes vaporariorum. Una de las conclusiones que se obtuvieron fue que B. argentifolii tenía una mejor adaptación a las altas temperaturas que T. vaporariorum. Esto se comprobó en las tasas de supervivencia de ambas especies a 30°C. Además, B. argentifolii usa hojas de la planta que están más altas que las que usa T. vaporariorum (Tsueda & Tsuchida, 1998).

La mosquita blanca es un insecto que tiene un ciclo de vida que consta de 6 etapas, las cuales son el huevo, 4 estados de inmadurez y la etapa adulta. Luego de que los huevos son depositados en el envés de la hoja, ocurre un periodo de incubación de entre 5 y 9 días. Luego de la eclosión del huevo, el primer instar de la mosquita se arrastra hasta llegar al punto de la hoja en el cual se alojará para alimentarse de la savia presente en la hoja. La mosquita no se mueve de este punto hasta que llega a la etapa adulta (Sani et al., 2020).

Además, durante los 4 estados de desarrollo que se encuentran entre el huevo y la edad adulta de las mosquitas blancas, el color de las mosquitas siempre es un blanco amarillento. La mayor diferencia se presenta en el cuarto estado, conocido como la pupa porque tiene ojos de color rojizo. Esta característica es tan conocida que se le conoce como "ninfa de ojos rojos" a esta fase (Abubakar et al., 2022).

Otro dato importante para destacar es que las mosquitas blancas eligen sus plantas huésped de manera mayormente aleatoria, ya que solo se atraen a los colores verdes y amarillos. Luego de que la mosquita aterriza en una planta, prueba el tejido de la hoja y es en este punto donde las mosquitas muestran una preferencia por unas plantas sobre otras (Van Lenteren et al., 1996).

Los daños que B. tabaci puede causar en las plantas se debe a dos causas. La primera de estas es daño provocado por la alimentación de la mosquita blanca, ya que, como se ha comentado previamente, la melaza que excretan las mosquitas da pie para el crecimiento de hollín en la hoja y los frutos, lo cual reduce la capacidad de la planta para llevar a cabo la fotosíntesis y esto a su vez merma el crecimiento de la planta y produce deformidades en los frutos.

El otro tipo de daño que se puede dar en las plantas afectadas viene debido a que la mosquita blanca puede transmitir más de 200 virus que afectan a las plantas. Dentro de las frutas más afectadas por esta situación se tiene el tabaco, la papa, la soya, el camote, las cucurbitáceas, lechuga, entre otros más. Es necesario recordar que las cucurbitáceas son el grupo al cual pertenece la planta de pepino. Estas pérdidas en la producción causadas por virus pueden ir desde el 20 hasta el 100% de la producción (Sani et al., 2020).

La mosquita blanca puede también provocar enfermedades con Begomovirus, los cuales provocan amarillamiento de las hojas y pérdidas de rendimiento. Este trabajo implicó recolección de ambas especies de mosquita en diversos campos en estados de México y, para el caso del cultivo de pepino, este estudio solo recolectó B. tabaci. Sin embargo, en el caso de la recolección de mosquitas blancas presentes en otros cultivos, la proporción era mucho más equilibrada, por lo que se concluyó que en la mayoría de las parcelas ambas especies coexisten (Velásquez - Valle, 2020).

Como se ha comentado previamente, las pérdidas que la mosquita blanca puede ocasionar pueden ser catastróficas y, ya que atacan a una cantidad importante de cultivos que son consumidos por la sociedad, se tienen una gran cantidad de teorías y técnicas propuestas para frenar el avance de la mosquita.

La atracción de las mosquitas blancas al color amarillo fue reportada por primera vez por Lloyd en 1922 cuando vio que había una mayor proporción de mosquita blanca en trampas amarillas con pegamento. Por su parte, se ha descubierto que las mosquitas blancas se acercan a luces con longitud de onda de 550 nm más que a luces con longitud de onda de 400nm. Las mosquitas blancas aterrizan en plantas verdes, que reflejan justamente ondas de luz de 550 nm (Coombe, 1982).

Confirmando la idea de Combe de los espectros de luz a los que se atraen las mosquitas blancas, se ha propuesto una evolución a las trampas de color de papel con adhesivo usando diodos emisores de luz para atrapar a las mosquitas blancas (Stukenberg & Poehling, 2019).

Otra técnica de prevención contra la plaga de mosquita blanca consiste en el uso del parasitoide Encarsia Formosa como un depredador natural contra la mosquita blanca, el cual puede matar a las mosquitas antes de que puedan hacer daños a la planta huésped (Van Lenteren et al., 1996). Otra técnica para combatir a las mosquitas blancas consiste en el uso de aceites esenciales de tomillo y hierbabuena, de los cuales el aceite de tomillo tuvo una gran eficacia (Aroiee et al., 2005).

Como se comentó en esta sección, dos de las especies de mosquita blanca que hoy en día continúan atacando a una gran cantidad de plantas y cultivos, fueron descritas originalmente en el siglo XIX. Desde ese entonces, se han agregado una gran cantidad de conocimientos que nos permiten entender la mosquita blanca, desde su esperanza de vida, las etapas de su vida, su reproducción, y finalmente, la forma en la que puede ser combatida. Este interés en esta plaga se debe en gran parte al potencial de pérdidas que puede provocar.

2.3 Python

El lenguaje de programación Python nació de otro lenguaje llamado ABC como una alternativa a BASIC para su uso en microcomputadoras, sin dejar la facilidad para programar de lado. Python siempre ha sido un lenguaje de código abierto que permite que los desarrolladores puedan hacer contribuciones que enriquecen tanto el lenguaje como la comunidad de programadores. Acerca de estos usuarios, en años recientes la proporción entre científicos de datos y desarrolladores web se ha nivelado, llegando a tener cantidades similares de usuarios que usan este lenguaje para tareas de ciencia de datos y desarrollo web (Unpingco, 2021).

Python fue creado por Guido van Rossum en 1990, y continúa teniendo un rol en el desarrollo subsecuente que el lenguaje ha tenido y tiene poder de decisión en las disputas dentro de la comunidad. Python ha tenido 3 versiones desde su lanzamiento en 1991, con el lanzamiento de Python ocurriendo en el año 2000 y el de Python 3, en 2008. El cambio entre las versiones 2 y 3 de este lenguaje fue radical, al grado que algunos códigos escritos en las versiones previas no funcionan correctamente.

Python es uno de los lenguajes de programación que más crecimiento tiene, IEEE Spectrum lo puso como el mejor lenguaje de programación en 2017, por delante de lenguajes como C, Java, C++, C#, JavaScript, entre otros. Dentro de los frentes donde Python ha tenido crecimiento se tiene el lado académico, como una alternativa a Matlab y para programar microcomputadoras como la Raspberry Pi, así como también en herramientas científicas gracias a las librerías que tiene Python, máquinas de aprendizaje que usan las herramientas científicas como base, el análisis de datos en conjunción con programas establecidos como Microsoft Excel, estadística, entre otros más (S & A, 2019).

Python ha proliferado tanto entre los usuarios programadores gracias a que es gratis para usar y desarrollar en él, a la calidad del software, la productividad para los desarrolladores, la portabilidad de los programas, la cantidad de librerías de soporte que existen, la integración de componentes que le permite el uso de códigos en C, C++ y otros lenguajes, y porque, al final de todo, es una experiencia disfrutable. Si se quisiera agregar una desventaja de Python, ésta se encuentra en el hecho de que puede tardar un poco más en procesar los comandos al

ser un lenguaje de programación cuando se compara con C, por ejemplo. Este retardo viene de que Python convierte el programa escrito en un lenguaje intermedio llamado “byte code” antes de que la computadora lleve a cabo las instrucciones que se le piden.

Python es usado por compañías tales como Google, YouTube, NASA, Intel, JPMorgan, Pixar y muchas otras. Esto es gracias a la gran cantidad de cosas que pueden hacerse con este lenguaje de programación. Python permite programar sistemas, desarrollar interfaces gráficas, páginas de internet, programación de bases de datos, desarrollo de funciones matemáticas y científicas, desarrollo de videojuegos, procesamiento de imágenes, entre otras (Lutz, 2009).

2.4 Aprendizaje de Máquinas

Como se comentó en la sección previa, una de las aplicaciones de Python más importantes se tiene en su uso en aplicaciones de aprendizaje de máquinas. Uno de esos primeros avances en el campo de aprendizaje de máquinas consistió en tareas como la predicción de recuperación de pacientes con neumonía, detección del uso fraudulento de tarjetas de crédito, jugar juegos de mesa al nivel de campeones mundiales, así como los primeros ejemplos de manejo autónomo vinieron de esta época de investigación (Mitchell, 1997).

En su definición más simple, el aprendizaje de máquinas es justamente enseñarle a una computadora a llevar a cabo una tarea por medio de un código. Un objetivo del aprendizaje de máquinas es construir sistemas computacionales que mejoren a través de las experiencias. Este objetivo se puede lograr con una combinación de otros campos de estudio, como lo son la ciencia computacional, la estadística, así como otras disciplinas interesadas en la mejora automática y en la inferencia y decisión en situaciones inciertas. Otras disciplinas son el estudio del aprendizaje humano, el estudio de patrones de educación y neurociencia (Jordan & Mitchell, 2015).

Una manera de definir al aprendizaje de máquinas es como el campo de estudio que le brinda a las computadoras la habilidad de aprender sin ser explícitamente programadas para ello. El aprendizaje puede darse por varios caminos, como lo es el aprendizaje supervisado, que

requiere asistencia externa como lo es etiquetar los ejemplos para que el sistema aprenda basado en la información que obtuvo y las etiquetas que se le dio. El aprendizaje no supervisado deja al algoritmo buscar por sí solo las coincidencias presentes en los datos que se le presentan, y cuando tiene datos nuevos, usa lo que aprendió en fases previas para clasificar los datos nuevos.

Otra técnica es el aprendizaje semi supervisado en el cual se puede entrenar un algoritmo con datos etiquetados que incluyen la información de salida. Otro grupo es el algoritmo multitareas, donde el algoritmo busca coincidencias entre dos tareas distintas. Dentro de este grupo se encuentran las redes neuronales, las cuales buscan relaciones entre datos en un proceso similar a la forma en la que el cerebro humano opera (Mahesh, 2018).

Como conclusión se puede comentar que el aprendizaje de máquinas es una técnica que tiene una gran cantidad de aplicaciones, dentro de las cuales se pueden resaltar las redes neuronales. El objetivo de las redes neuronales es resolver problemas con algoritmos que imiten la forma en la que el cerebro humano funciona, pero también es importante que el algoritmo aprenda de los datos que fueron parte del aprendizaje. Las redes neuronales se describirán con más detalle en las siguientes secciones.

2.5 TensorFlow y Keras

Como se comentó previamente, Python ofrece una gran cantidad de librerías que permiten la creación de una gran cantidad de aplicaciones y, por ello, es una de las herramientas predilectas de los científicos de datos para aplicaciones de aprendizaje de máquinas. Dentro de las técnicas de aprendizaje de máquinas son de gran importancia las redes neuronales y, Python ofrece una serie de librerías que ayudan en el proceso que va desde el procesamiento de datos, pasando por la creación de la red neuronal y el modelo de entrenamiento, hasta llegar a las métricas para comprobar la eficiencia de dicho modelo.

Esta sección hablará de dos librerías presentes en Python, las cuales son TensorFlow y Keras. Ambas son importantes para el desarrollo de redes neuronales, por lo cual se explicarán a continuación. TensorFlow originalmente fue creado por un grupo de desarrolladores en

Google, donde su librería para el aprendizaje profundo se convirtió en una herramienta que el público en general podía usar. Y, aunque otras librerías existen que apoyan en la creación de redes neuronales, TensorFlow ofrece una ventaja importante, ya que permite que el programador se concentre en las definiciones simbólicas de lo que debe computarse, dejando de lado la forma en la que esto ocurre. Otra función que TensorFlow ofrece se encuentra en TensorBoard, que es una herramienta que permite visualizar parámetros de rendimiento de la red que se esté entrenando. Estos autores agregan finalmente que TensorFlow permite paralelización en procesadores múltiples, sean CPU o GPU, y esto agrega poder computacional (Rampasek & Goldenberg, 2016).

El funcionamiento de TensorFlow se basa en la operación de tensores. Un tensor es una entidad matemática con la que se representan propiedades diversas; en resumen, son arreglos multidimensionales que cuentan con propiedades dinámicas. Estos tensores son operados en gráficas, que muestran el flujo que los tensores toman. En estos flujos hay vértices que pueden tener operaciones por los que pasan los tensores y cambian su valor (P. Singh & Manure, 2020).

Teniendo en cuenta lo anterior, TensorFlow es una herramienta poderosa para llevar a cabo cálculos matemáticos complejos. Esto hace que TensorFlow sea una herramienta ideal para su uso en los cálculos complejos que son necesarios para aplicaciones en las que se hace uso de las redes neuronales.

Sin embargo, se sabe que las redes neuronales tienen una cantidad de capas que hacen operaciones diferentes dependiendo de la función de activación que usan. Estas capas y estas funciones de activación son definidas gracias a la otra librería importante para la programación de redes neuronales, la cual es Keras.

Keras es una librería que se encuentra dentro de TensorFlow, la cual puede definirse como una librería que le permite a los programadores enfocarse en la arquitectura de la red neuronal, mientras que Keras se ocupa de los detalles relativos a los tensores, como lo son su forma y detalles matemáticos (Manaswi, 2018). TensorFlow es entonces la parte trasera, que se encarga del aspecto matemático de la red neuronal, mientras que Keras es la parte frontal,

amigable para el usuario, que solo necesita ir agregando las capas sin pensar demasiado en las formas de los tensores.

Esta combinación entre Keras y TensorFlow es lo que permite a los usuarios la creación de redes neuronales y modelos de entrenamiento que pueden ser usados de manera sencilla. Esto permite que, aunque no se tengan conocimientos avanzados en estadística y matemáticas, se puedan programar las redes neuronales fácilmente. Sin embargo, aunque esto facilita la tarea en gran medida, aun es necesario que el desarrollador cuente con habilidades en el campo de la programación.

2.6 Redes Neuronales

Como se comentó, el aprendizaje de máquinas es un campo científico que agrupa una cantidad de conocimientos que agrupan la estadística y las matemáticas con otras ciencias, buscando que una computadora aprenda algún conocimiento. Dentro del aprendizaje de máquinas, se resaltan las redes neuronales, las cuales logran el objetivo de hacer que una computadora aprenda por medio de imitar el cerebro humano. A continuación, se describen este tipo de aprendizaje de máquinas.

Una red neuronal artificial se define como un modelo de procesamiento de datos basado en la forma en la que los sistemas nerviosos biológicos, como el cerebro, procesan datos. El cerebro humano tiene a las neuronas interconectadas en forma de red y estas son a su vez los nodos de la red. Las redes neuronales artificiales se inspiran en la forma en la que el sistema nervioso biológico opera para procesar datos e información con la finalidad de crear conocimientos, donde se tiene un sistema con elementos de procesamiento llamados neuronas que colaboran para resolver un problema y transmitir información por medio de sinapsis. Las neuronas están organizadas en capas, donde la primera capa es la capa de entrada que recibe los datos, la última capa es la capa de salida que arroja el resultado, mientras que entre estas dos hay una cantidad de capas ocultas (Dastres & Soori, 2021).

Cada neurona tiene un peso asignado a ella. Este peso es multiplicado por el dato que se venga de la entrada, o de la capa anterior y son computados por medio de una función

matemática que determina si la neurona se activa o no. Otra función además calcula la salida de la neurona que puede depender de algún umbral (Gupta, 2013).

El primer modelo de neurona artificial fue propuesto por McCulloch y Pitts en los 1940s. Este modelo lleva a cabo una transformación lineal a través de una sumatoria ponderada por los pesos. Las operaciones que hace una neurona biológica son imitadas por medio de una transformación lineal que calcula los pesos seguida de una función de umbral que puede ser no lineal llamada función de activación (Kiranyaz et al., 2021).

De lo anterior se puede comentar que la idea de crear algoritmos que imiten el comportamiento del cerebro humano y que puedan aprender de los datos que ven son los dos puntos que dieron pie a la investigación acerca de redes neuronales, las cuales solo pudieron recrearse en computadoras décadas después de la primera investigación que propuso las neuronas artificiales.

Aun cuando su desarrollo tomó un largo tiempo, hoy en día las redes neuronales son una técnica que tiene gran importancia dentro de las máquinas de aprendizaje. Las redes neuronales han tenido un gran auge recientemente en una gran cantidad de aplicaciones y una de estas aplicaciones se tiene en entrenar la red con datos que son los pixeles de una imagen. Para este trabajo con imágenes, es común el uso de las redes neuronales convolucionales, las cuales se describen a continuación.

Desde la definición matemática de neurona que McCulloch y Pitts presentaron, comenzaron las mejoras a esta idea. Dentro de estas, es necesario destacar el trabajo de Rumelhart et al., quien agregó la idea de una red con propagación en reversa con la finalidad de usar el error para el entrenamiento de la red neuronal. El otro trabajo de importancia es el de LeCun et al., quien hizo uso por primera vez de la palabra “convolución”, en un trabajo de una red neuronal que reconocía el código postal escrito a mano, lo cual tomó el nombre de LeNet (Kiranyaz et al., 2021).

Una red neuronal convolucional se define como una red neuronal con propagación hacia adelante que puede extraer características de datos con estructuras de convolución. Estas redes no necesitan extraer características de forma manual y su arquitectura se inspira en la

percepción visual, donde los filtros de la red neuronal convolucional representan receptores que responden a diversas características. Algunas ventajas importantes contra las redes neuronales son una cantidad de conexiones más pequeña, los pesos son compartidos entre varias neuronas, y las funciones de reducción que permiten reducir la información mientras que se retiene la información relevante (Li et al., 2022).

Además, las redes neuronales convolucionales están diseñadas de tal manera que, no solo imitan la forma en la que el cerebro de los animales procesa información, sino que también procesan imágenes intentando imitar la forma en la que el cerebro animal lo hace. En este caso, las capas de convolución presentes en las redes convolucionales tienen como objetivo la detección de características, como bordes, colores, textura y orientación. Estas capas están hechas de filtros que pueden aprender que llevan por nombre "filtros convolucionales". Estos filtros son usados para llevar a cabo la operación de convolución entre el largo y ancho de la imagen y esta operación de convolución consiste en llevar a cabo el producto punto entre el filtro y la imagen y computar el resultado. Al final de las capas de convolución, la salida es llevada a una capa de función de activación (Sarvamangala & Kulkarni, 2022).

Las redes neuronales convolucionales también han experimentado un gran auge, debido a su facilidad para el trabajo con aplicaciones donde se hace uso de imágenes. Muchos de los trabajos que implican visión artificial hacen uso de redes neuronales convolucionales, ya sea con arquitecturas o con procesamientos previos al entrenamiento diferentes, pero el resultado final generalmente es un modelo de entrenamiento que se usará en aplicaciones de detección de objetos por medio de visión por computadora.

2.7 Funciones de Activación, Optimización y Pérdida

Para llevar a cabo el entrenamiento de una red neuronal, existen una cantidad de parámetros que pueden ser modificados para tener un modelo de entrenamiento que pueda ser usado en una aplicación de inferencia. Para modificar el desempeño en el entrenamiento de un modelo, se tienen las funciones de activación, optimización y pérdida, las cuales pueden generar un modelo de entrenamiento con mayor precisión cuando se eligen correctamente. Sin embargo,

para elegirlas correctamente, es necesario entenderlas y por ello, se describirán a continuación.

Las funciones de activación son necesarias en las redes neuronales porque, sin ellas, la red neuronal se convierte en una aplicación de regresión lineal, lo cual quita muchas de las ventajas que las redes neuronales tienen. Es necesario agregar una función de activación para hacer una red neuronal dinámica que pueda extraer información complicada de los datos y poder representar mapas funcionales no lineales entre la entrada y la salida de la red, la cual se puede agregar por medio de funciones de activación no lineales.

Algunas de las funciones de activación que se pueden listar son la función lineal, sigmoide, tangente hiperbólica, ReLU, y Softmax. De las anteriores, la función sigmoide está dada de tal manera que los datos de entrada negativos dan una respuesta que se aproxima a -1, y los positivos, arrojan una respuesta de 1. Si la función ReLU recibe un valor negativo, la función arrojará un 0 de resultado, mientras que, si el dato de entrada es positivo, la salida será igual al dato de entrada. Finalmente, la función Softmax es una combinación de funciones sigmoidales las cuales permiten la predicción de clasificación con clases múltiples. En este caso, el resultado es la probabilidad de que el resultado pertenezca a esa clase (Sharma et al., 2020).

Las siguientes funciones que se describirán son las funciones de optimización. La primera de estas funciones agrupa a todas las versiones del gradiente descendiente. El gradiente descendiente es la forma más común para optimizar redes neuronales, que está presente en algunas de las librerías más usadas para el desarrollo de redes neuronales, como lo es Keras. La versión más básica de gradiente descendiente calcula el valor de la función de costo para todos los ejemplos de entrenamiento.

La otra variante que se presenta es el gradiente descendiente estocástico, que lleva a cabo la actualización de parámetros para cada ejemplo y etiqueta de entrenamiento. Al trabajar de un ejemplo por uno, es más rápida que el gradiente descendiente básico (Ruder, 2017).

Otras técnicas que se resaltan a continuación son RMSprop y Adam. Tanto RMSprop como Adam son funciones de optimización que cuentan con gradientes adaptativos. RMSprop

soluciona el problema que la función AdaGrad tiene. La solución consiste en usar pesos que decaen exponencialmente para gradientes pasados. Esta función de optimización es el paso previo para tener la función Adam, que Sun describe como RMSprop con el método del momento (Sun, 2020).

Por su parte, la función de optimización Adam computa las tasas de aprendizaje adaptativas para cada parámetro. Además, Adam también guarda un promedio que decae de los gradientes cuadrados previos, así como de otros gradientes previos que decaen de forma exponencial. Adam es una función de optimización que trabaja de buena forma cuando se compara con otros métodos de aprendizaje adaptativo (Ruder, 2017).

Las funciones de pérdida son definidas como la medición que le permite a una red neuronal saber si está aprendiendo en la dirección correcta. Una forma de ver a las funciones de pérdida es verlas como una medición entre el resultado obtenido por el entrenamiento y el resultado real, es ver qué tan cerca estuvo el modelo de la realidad. Para esto, se tienen varias funciones como el error medio cuadrado, el error medio absoluto, las cuales comparan y suman las diferencias entre el valor real y el predicho, donde el error medio cuadrado se eleva esta diferencia al cuadrado, mientras que el error absoluto toma el valor absoluto de esta diferencia, como lo dice su nombre.

Sin embargo, las dos funciones de pérdida más importantes en el campo de las redes neuronales son entropía cruzada y entropía cruzada categórica. La diferencia entre estas dos es su aplicación, mientras la que la entropía cruzada binaria es usada en aplicaciones donde el resultado es una variable binaria, la entropía cruzada categórica es usada cuando el resultado es una variable no binaria, o sea, que puede tener más de dos resultados (Moolayil, 2019).

Estos son algunos de los tantos parámetros que pueden modificarse para generar mejoras en el modelo de entrenamiento, es necesario tener al menos conocimientos básicos de estas funciones para elegir las funciones correctas de acuerdo con las diversas aplicaciones que se pueden llevar a cabo.

Teniendo un modelo de predicción entrenado, se debe comprobar que el modelo tenga un desempeño correcto, lo cual se logra por medio de comprobar su precisión, por ejemplo. Sin embargo, existen aplicaciones en las cuales la precisión no es la estadística correcta, y se tienen otras, las cuales se describirán a continuación.

2.7.1 Precisión y otras métricas de evaluación

La evaluación cuantitativa de los modelos de segmentación puede hacerse comparando las etiquetas de predicción con las que se entrenó el modelo con las predicciones que hizo el modelo. Esta comparación puede ser por medio de comparación de píxel por píxel o comparando las áreas. La comparación píxel por píxel necesita de matrices de confusión y el cálculo para obtener el positivo verdadero, negativo verdadero, falso positivo y falso negativo para conformar la matriz de confusión, así como para obtener parámetros como la precisión y la sensibilidad.

Por el lado de las mediciones basadas en el traslape de áreas, se tiene el coeficiente de Dice y el índice de Jaccard. El coeficiente de Dice se define como la intersección entre dos conjuntos A y B dados sobre la suma de los valores absolutos de A y B, todo esto multiplicado por dos. Por otro lado, el índice de Jaccard también es conocido como intersección sobre unión, y como su nombre lo indica, es la intersección de los conjuntos A y B dividida por la unión de ellos (Asgari Taghanaki et al., 2021).

Aunque generalmente se puede comprobar el correcto funcionamiento de un modelo de entrenamiento solo usando métricas como la precisión y la pérdida, existen casos donde la precisión puede ser una medición engañosa, ya que, aunque el modelo está prediciendo y acertando, es posible que se deba a que una clase cuenta con mucha más representación que las otras en los datos de entrenamiento. Es aquí donde entran las mediciones como el coeficiente de Dice y el índice de Jaccard, las cuales evalúan el modelo desde el punto de vista del traslape de las áreas que ocupan las etiquetas. Es necesario agregar que estas métricas son idóneas para comprobar el funcionamiento de modelos donde se habla de

detección de objetos y que son ampliamente usadas, por lo que, también han sido usadas en este trabajo.

Las redes neuronales han sido un campo de estudio que ha dado una gran cantidad de aplicaciones, entre las que están las aplicaciones de detección de objetos en imágenes. Una de estas arquitecturas ampliamente usadas para la detección de objetos se tiene en la U-net, la cual se describe a continuación.

2.8 U-net

Ya que las redes neuronales convolucionales han tenido una gran cantidad de aplicaciones y mejoras, se ha llegado a una red neuronal convolucional que creó Ronneberger en 2015, llamada U-net. Para Ronneberger, las redes neuronales convolucionales son ampliamente usadas en tareas de clasificación donde el resultado para una imagen con una sola etiqueta. Sin embargo, existen aplicaciones donde el resultado del entrenamiento es asignar una etiqueta a cada píxel de la imagen, entre las que se encuentran aplicaciones en el campo de la biomedicina (Ronneberger et al., 2015). La necesidad de contar con redes neuronales convolucionales que puedan clasificar los pixeles dentro de la imagen, a diferencia de redes que solo clasifican imágenes de acuerdo con una etiqueta dada, es la base de la creación de la U-net.

En este trabajo se tomó como base la red neuronal convolucional para poder usarla con pocas imágenes de entrenamiento obteniendo mejores segmentaciones. Las redes neuronales convolucionales trabajan con una contracción de los datos conforme se avanza en las capas de la red neuronal. La red neuronal que se propuso agrega un camino de expansión luego de la contracción presente en las redes neuronales convolucionales. En este camino de expansión se tienen canales de características, que le permiten a la red propagar información de contexto a las capas de mayor resolución (Ronneberger et al., 2015).

Ya que la mejora principal provista por la U-net radica en la predicción en imágenes al nivel de píxel por píxel, la principal aplicación de esta arquitectura se da en proyectos en los cuales se busca detectar objetos dentro de la imagen basado en el área que ocupan dentro de la

imagen. Ronneberger usó la U-net para detección en imágenes biomédicas y se probó su eficiencia, a tal grado que las aplicaciones haciendo uso de la U-net han ido aumentando desde ese momento, y hoy en día la U-net y sus variantes son una de las arquitecturas preferidas para el entrenamiento de modelos de inferencia en aplicaciones de detección de objetos.

Al igual que con muchas de las tecnologías y conceptos que se han descrito en esta sección, la U-net también ha tenido mejoras desde el trabajo presentado por Ronneberger et al., ya que la aplicación más común para la U-net es la detección de objetos y las áreas que éstos ocupan en la imagen, ha sido usada también en aplicaciones de segmentación de imágenes de satélite (McGlinchy et al., 2019).

Aunque se podría seguir hablando de aplicaciones de la U-net y de avances en esta arquitectura de red neuronal, a continuación, se describirá uno de los avances más importantes que está siendo usado como parte de este trabajo. Este trabajo es el de Chollet, quien comienza por explicar el módulo Inception de Google, que se basa en extractores de características convolucionales. Este módulo Inception factoriza el proceso en operaciones que buscan por su cuenta correlaciones espaciales y entre canales.

La tarea que hace el módulo Inception puede hacerse usando una convolución de 1x1 para mapear las correlaciones entre canales y luego mapear las correlaciones espaciales de cada canal de salida. Esta forma es muy similar a una convolución separable de acuerdo con la profundidad, y este tipo de convoluciones ha estado presente desde 2014, y fue añadido a TensorFlow en 2016.

El uso de convoluciones separables como las que están presentes en la librería de TensorFlow dio pie al modelo Xception creado por Chollet y que fue comparado con el modelo Inception, de Google (Chollet, 2017). Este antecedente es importante para este trabajo, ya que la arquitectura de red neuronal usada para este trabajo hace uso de la U-net de Ronneberger et al., así como de las convoluciones separables que fueron probadas a fondo en el trabajo de Chollet para redes neuronales convolucionales.

2.9 Segmentación Semántica

En la sección anterior se comentó que la U-net tiene el objetivo de predecir la clase a la que pertenecen los píxeles presentes en una imagen, lo cual le da una ventaja importante sobre otras redes neuronales convolucionales que solo predicen en base a la etiqueta otorgada a la imagen. Ahora, ya que es necesario proveer a la U-net de datos de entrada, los cuales deben estar etiquetados a un nivel de píxel por píxel para que cada uno tenga la clase a la que pertenece, se hace necesario el uso de técnicas diversas para llevar a cabo este tipo de segmentación. La técnica de segmentación semántica cumple con la necesidad de llevar a cabo una clasificación a nivel de píxel por píxel y se describirá a continuación.

La segmentación semántica es uno de los problemas más grandes a los que se enfrenta el campo de la visión por computadora. La segmentación semántica es una tarea de alto nivel que lleva al entendimiento de una escena por completo, y este entendimiento es necesario ya que las aplicaciones que usan, entre las que se destacan el manejo autónomo, interacciones entre humanos y máquinas, realidad aumentada, entre otros. Esta tarea de segmentación semántica es resuelta por medio del uso de arquitecturas de aprendizaje profundo, como lo son las redes neuronales convolucionales (Garcia-Garcia et al., 2017).

La meta de la segmentación semántica es asignar a cada píxel de una imagen una etiqueta de clasificación. Esto es una tarea de aprendizaje supervisado que necesita de datos clasificados al nivel de cada píxel. Las aplicaciones que destaca para la segmentación semántica incluyen comprensión de escena, remover objetos no deseados de fotografías, copiar objetos y pegarlos en otra imagen, entre otras (Csurka et al., 2013).

La técnica de segmentación semántica es un paso más en el camino a una inferencia precisa. Este camino comienza con la clasificación básica, para predecir cuál es el objeto de una imagen, y continua en la localización y detección para obtener tanto las clases como información adicional relacionada con la localización espacial de esas clases. Esto nos lleva a decir que la segmentación semántica es el paso natural para una inferencia precisa, cuyo objetivo es inferir etiquetas para cada pixel que compone la imagen, para que cada uno de los pixeles tenga una etiqueta que corresponde a la clase a la que pertenecen (Garcia-Garcia et al., 2017).

Se debe de recordar que la U-net fue diseñada para llevar a cabo predicciones que consisten en clasificar los pixeles dentro de la imagen de acuerdo con las etiquetas creadas por medio de la segmentación semántica. Esta segmentación semántica permite etiquetar cada píxel dentro de la imagen como parte de un grupo y, esta característica es la que hace de la segmentación semántica la técnica de preprocesamiento predilecta para la implementación de aplicaciones que usen la U-net como arquitectura de red neuronal.

La segmentación semántica es más rápida y eficiente que el proceso de ir moviendo un filtro por una imagen para obtener las características que la componen. Este proceso de detección de objetos por medio de ventanas también tiene que lidiar con más ruido de fondo y si un algoritmo de segmentación de imágenes funciona bien, el ruido se removerá en automático y llevará a una mayor precisión. La segmentación de imágenes también nos da una idea de cómo los ojos humanos llevan a cabo la misma tarea. Esto implica que también se verá la forma en la que estos sistemas fallarán para entender conceptos visuales u objetos en el mundo.

Además, este trabajo cita algunas desventajas de la segmentación de imágenes son que los métodos están diseñados para aplicaciones muy específicas, por lo que usarlo en aplicaciones generales puede ser complicado; en otros casos, también es un problema la cantidad de datos necesarios, así como la capacidad computacional para su procesamiento. Tampoco se sabe qué tan específicos tienen que ser los datos presentes en el entrenamiento para que el sistema pueda inferir qué hacer en situaciones que no estuvieron presentes en el entrenamiento (Guo et al., 2018).

En resumen, los conocimientos descritos como parte de este marco teórico son algunos de los más relevantes para la elaboración de este modelo de entrenamiento. Técnicas como lo son la segmentación semántica y la red neuronal U-net permiten el entrenamiento de modelos que llevan a cabo la tarea de detección de objetos. En general, tanto las técnicas de procesamiento de imágenes y las máquinas de aprendizaje han tenido una explosión en la cantidad de aplicaciones en las cuales se ha hecho uso de ellas. Todos estos conocimientos permiten el desarrollo de nuevas aplicaciones, como la presentada en este trabajo.

Capítulo 3. Metodología

Este trabajo tuvo una serie de pasos y procesos que se llevaron a cabo para lograr los objetivos descritos previamente. El primero de estos pasos fue investigar características tanto del pepino como de la mosquita blanca y lo que se investigó se encuentra descrito en este trabajo. Los siguientes pasos se llevaron a cabo para obtener el modelo de entrenamiento, donde se llevó a cabo una plantación de pepinos, los cuales fueron infectados por mosquita blanca de manera involuntaria y posteriormente de forma voluntaria. A continuación, se hizo la toma de fotografías de la planta de pepino con las lesiones de mosquita blanca, se clasificaron las imágenes usando la técnica de segmentación semántica y se usaron las imágenes para el entrenamiento de una red neuronal usando como base la arquitectura U-net. Finalmente, se evaluó el modelo de entrenamiento por medio de su precisión en validación y del índice de Jaccard entre las imágenes etiquetadas y las inferencias que el modelo hizo. A continuación, se describen a detalle estas etapas.

3.1 Plantación de Pepinos

Para lograr el objetivo principal de este proyecto, que consiste en un modelo de entrenamiento de red neuronal usando la arquitectura U-net para la detección de mosquita blanca, se comenzó por llevar a cabo un plantío de pepino en un ambiente casero. Este plantío comenzó como solución a la limitación consistente en la falta de imágenes pertenecientes a un plantío de pepino en un invernadero.

Por ello, se compraron semillas de pepino de la marca Kristen Seed basada en Guadalajara, Jalisco en una tienda dedicada a la venta de insecticidas y otros productos agropecuarios de la ciudad de Durango, México. El plantío se realizó en tres macetas, dos de las cuales eran macetas convencionales mientras que la tercera fue hecha a base de una caja de madera, cubierta de plástico adhesivo por fuera de la caja y de hule por dentro. Esta caja se llenó con tierra para macetas, la cual también fue usada para llenar las macetas convencionales. A continuación, se muestran fotografías de las semillas, así como una de las macetas usadas en esta primera etapa.

Figura 1.

Semillas usadas y Maceta

Ya que este proceso de plantar pepinos se dio en la última semana de enero de 2023, y las temperaturas se acercan a 0°C, se optó por usar más hule para cubrir la caja de cartón. Es necesario comentar que este plantío se llevó a cabo en el área del jardín de una casa ubicada en la ciudad de Durango, la cual cuenta con un cancel de vidrio como división entre la cochera y el jardín. Este cancel de vidrio permite que la temperatura dentro de la cochera nunca baje al nivel de las bajas temperaturas presentes en la ciudad de Durango en el invierno y es en esta área de la casa en la cual una maceta fue colocada.

La plantación ocurrió el día 26 de enero de 2023. En el momento de la plantación, se plantaron una cantidad de semillas en 3 puntos de la maceta grande que se encontraba en el jardín y en dos grupos en la maceta que se encontraba en la cochera. El crecimiento de las plantas fue entonces documentado por medio de fotografías y, para el día 3 de febrero de 2023, se notó la germinación de la mayoría de las semillas. A continuación, las imágenes de la primera fila muestran la germinación de ambas macetas mientras que las imágenes de la segunda fila muestran un punto más adelantado del crecimiento de la planta, concretamente del 16 de febrero, donde se nota que, a pesar de haber sido plantadas el mismo día, hay una diferencia en el crecimiento.

Figura 2.

Germinación de las Plantas (arriba) y Diferencias en el Crecimiento (abajo)

Como se comentó previamente, el día 16 de febrero de 2023 fue cuando se notó la diferencia en el crecimiento de las plantas de ambas macetas. En este punto, la diferencia en el crecimiento de ambos grupos de plantas se hizo finalmente notoria y las plantas que se encontraban protegidas por el vidrio de la cochera tenían un crecimiento más rápido que las plantas pertenecientes a la maceta en el jardín. Se debe destacar que la maceta en el jardín tenía una capa de hule por encima justo con la finalidad de darle cierto nivel de protección contra el frío. Se puede considerar también que esta protección no fue suficiente para permitir que ambos grupos de plantas crecieran al mismo ritmo, y que las bajas temperaturas sí mermaron el crecimiento de la planta en gran medida.

3.2 Infección con Mosquita Blanca y Toma de Fotografías

Hacia finales de febrero de 2023, se notaron en las plantas de la maceta que se encontraba en el jardín y se notó la presencia de mosquita blanca en este plantío. La mosquita logró marchitar la mayoría de las plantas presentes en la maceta del jardín y para el inicio del mes de marzo de 2023, se optó por usar solamente las plantas que estaban en la cochera, las cuales tenían un crecimiento mayor. Ambos grupos de plantas tenían riego en promedio cada dos días, así que se puede atribuir la diferencia en el crecimiento a dos factores principales: el primero es que la maceta de la cochera estaba en un ambiente más templado gracias al cancel, lo cual evitaba que la temperatura tuviera los cambios drásticos propios del invierno.

Para llevar a cabo el riego de la planta de la cochera, la maceta era movida hacia el jardín para que mucha del agua que cae por la parte inferior de la maceta cayera al pasto del jardín. Se cree que estos momentos de riego y un pedazo del jardín afectado por mosquita blanca fueron las causas de que las plantas en la maceta que estaba dentro de la cochera también comenzaran a mostrar pequeñas manchas blancas en la hoja, así como la presencia de mosquitas blancas en las hojas. Por lo anterior, las imágenes que se encuentran a continuación muestran algunas de las primeras lesiones en el tejido de la hoja causadas por la mosquita blanca, las cuales fueron tomadas el día 13 de marzo de 2023.

Figura 3.

Lesiones en la planta de pepino causadas por la mosquita blanca

En esta fase de contagio involuntario fue cuando se decidió aumentar la cantidad de fotografías tomadas de las plantas de pepino, donde se tomaban fotografías en los días que se llevaba a cabo el riego de las plantas. Las fotografías fueron tomadas por medio de un teléfono Samsung A52. Se decidió que las fotografías tendrían una relación de aspecto de 1:1, con una resolución de 3472x3472 píxeles. Se eligió esta relación de aspecto cuadrada ya que muchas arquitecturas de red neuronal suelen cambiar la relación de aspecto de las fotografías para tener esta relación de aspecto.

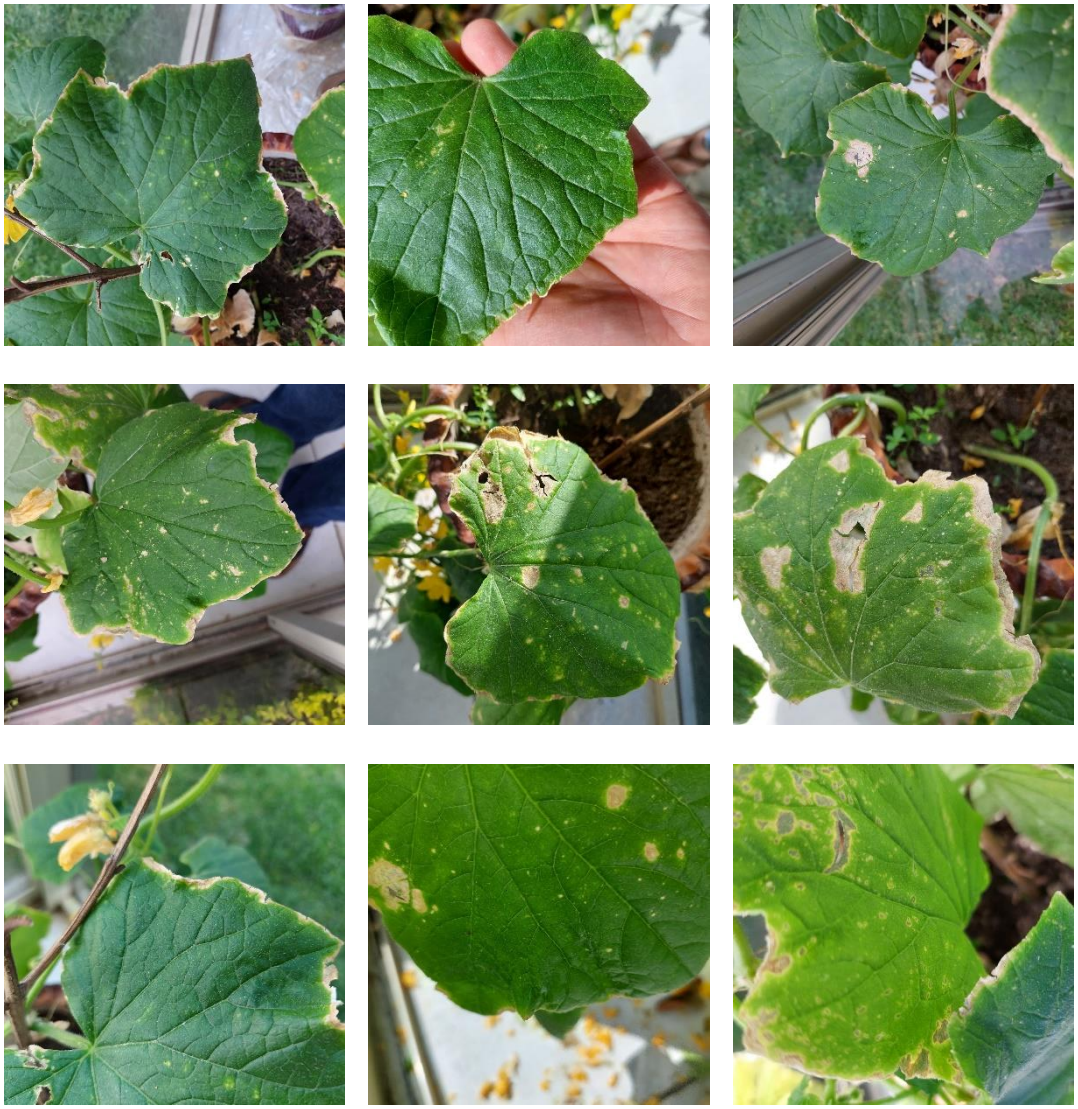
Luego de haber logrado un contagio involuntario de la planta de pepino con mosquita blanca, se optó llevar a cabo un contagio voluntario en la planta el día 29 de marzo de 2023. El contagio consistió en acercar a la planta al área del jardín donde se sabía que se encontraba la mosquita blanca, dejando la planta en esta área en contacto con las mosquitas por 4 horas en la tarde. Se optó por hacer este nuevo contagio con miras a tener una mayor cantidad de imágenes, así como para contar con una secuencia en el crecimiento de las lesiones en las hojas conforme el tiempo avanzó. Esto permitió obtener una mayor cantidad de imágenes para usar en el entrenamiento del modelo de red neuronal.

Luego de esta exposición a la mosquita blanca, entre los días 30 de marzo y 4 de abril de 2023 se llevó a cabo un proceso de tomar 100 fotografías, 3 veces al día en la resolución mencionada previamente, así como con el teléfono que se mencionó previamente. De esta segunda etapa de captura de fotografías, se debe comentar que el único día donde no se cumplió con la meta de tomar 100 fotografías 4 veces al día fue el 30 de marzo de 2023. En este día se llevó a cabo la toma de fotografías 2 veces.

En total, contando las fotografías tomadas desde mediados de febrero hasta el día 4 de abril de 2023, se tienen 1924 imágenes, que están tomadas en una relación de aspecto cuadrada, que muestran hojas con lesiones causadas por la mosquita blanca en varios niveles de avance. Se llegó a la cantidad de 1924 imágenes luego de descartar otra cantidad de imágenes que no contaban con una iluminación o enfoque competentes. A continuación, se muestran algunas de estas imágenes.

Figura 4.

Ejemplos de imágenes usadas para el entrenamiento



Como se puede notar en las imágenes mostradas en esta sección, se aprecian diversas afectaciones generadas por la mosquita blanca. Estas lesiones van desde pequeñas manchas blancas presentes en la hoja, hasta lesiones que han perforado el tejido de la hoja. Estas fueron algunas de las imágenes que se usaron en el proceso de etiquetado de imágenes, el cual se describe a continuación.

3.3 Etiquetado de Imágenes

El siguiente paso que se llevó a cabo fue la clasificación y el etiquetado de las imágenes. Como se ha comentado previamente, la técnica elegida para la clasificación es la segmentación semántica y el servicio elegido para este proceso es llamado Roboflow. La razón principal para elegir este servicio se tiene en la facilidad para llevar a cabo el etiquetado, así como que el producto final del etiquetado consiste en una máscara única que contiene todas las clases usadas para esta clasificación. Esta característica no se encuentra en otros servicios de etiquetado de imágenes, los cuales arrojan máscaras para cada una de las clases presentes en una imagen dada.

Acerca de la clasificación que forma parte del etiquetado, se decidió usar 5 clases dentro de las imágenes. La primera es el fondo, la segunda es la hoja, y de ahí se destacan 3 niveles de lesiones en la hoja causadas por la mosquita blanca, los cuales van de menor nivel a mayor.

En este caso, se definió el fondo como el área de la imagen que no forma parte de la hoja; esta área tiene un valor de 0. La hoja tiene un valor de 1, y se decidió solamente clasificar una hoja por imagen. De ahí, se tienen 3 niveles de avance de lesiones de mosquita blanca, donde el primer nivel son los puntos amarillentos que denotan la primera afectación causada por la mosquita blanca en el tejido de la hoja, el segundo nivel implica un cambio de color en la hoja teniendo una mancha blanca de mayor tamaño, mientras que el tercer nivel de avance se tiene cuando las lesiones han crecido tanto que se combinaron con otras lesiones, así como que la lesión en la hoja ha llegado a dañar el tejido creando lesiones que atraviesan la hoja. Cada nivel de avance de las lesiones de la mosquita blanca tiene un valor de píxel diferente, teniendo 2 para el primer nivel de avance de lesión, 3 para el segundo nivel, y 4 para el tercer nivel de avance. Es necesario comentar que, dentro de la clasificación de imágenes, se clasificó un área como la hoja de la planta, pero la cantidad de lesiones de cada nivel va variando de imagen a imagen. Para estos niveles, se usaron nombres de variables para su etiquetado en el servicio de Roboflow, los cuales se muestran en la tabla a continuación.

Tabla 1.

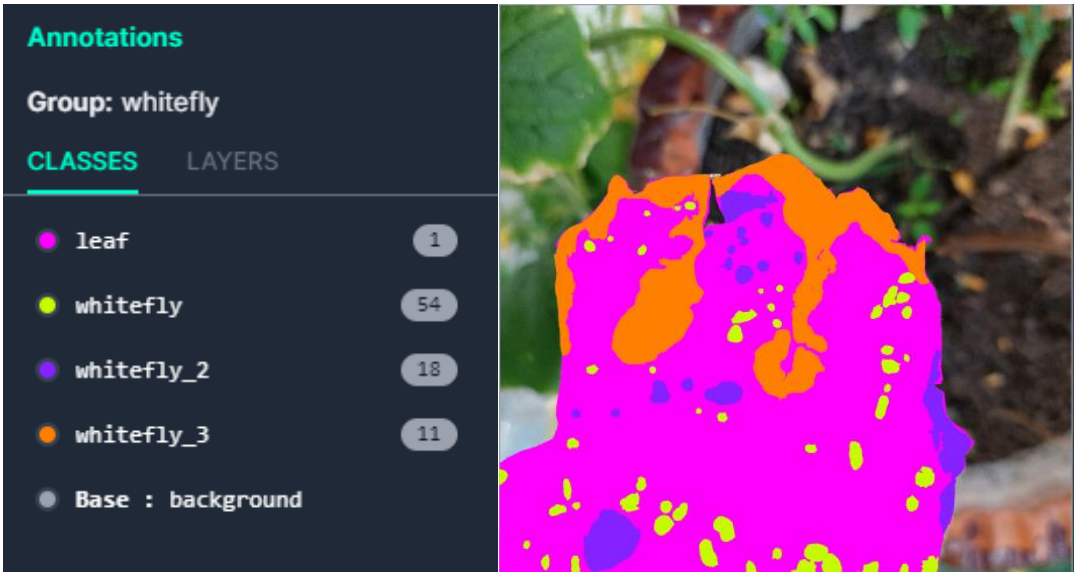
Niveles de Píxel y Clase Correspondiente en el Etiquetado

Valor de Píxel	Clase
0	background
1	leaf
2	whitefly
3	whitefly_2
4	whitefly_3

Las clases mostradas en la tabla anterior se muestran como parte del segmentado de una imagen dentro de la plataforma Roboflow a continuación. En la imagen, el fondo no cuenta con color ya que para Roboflow, el fondo es el área que no pertenece a ninguna otra clase. La hoja está resaltada con el color fucsia, el primer nivel de daño está de color amarillo, el segundo está resaltado con un color morado, y el tercer nivel, de color naranja.

Figura 5.

Clases y Segmentación en Roboflow



Acerca de las imágenes, de las 1924 imágenes que se tomaron para este proceso, se dividieron en 4 grupos tomando como base las fechas en las que estas fotografías se tomaron. De los cuatro grupos en los que dividieron las imágenes, se tienen dos grupos de alrededor 350 imágenes cada uno. De estos grupos, el primero abarca del 22 de febrero al 28 de marzo de 2023 y muestra las primeras lesiones causadas por la mosquita blanca. Como se comentó, la infección voluntaria de la planta con mosquita blanca se dio el 29 de marzo de 2023 y la tarde del día 30 se comenzó con el proceso de fotografías, tomando 100 fotografías 3 veces al día. Así, el segundo grupo abarca las imágenes tomadas los días 30 y 31 de marzo de 2023, que consiste también de 350 imágenes.

Los otros dos grupos constan de aproximadamente 600 imágenes y el primero consta de imágenes tomadas los días 1 y 2 de abril de 2023, mientras que el segundo consta de imágenes de los días 3 y 4 de abril 2023. Estos dos grupos también forman parte de las sesiones de 100 fotografías tomadas, 3 veces al día y por ello constan de alrededor de 600 imágenes cada uno.

Con esta división en grupos de imágenes, se comenzó con el etiquetado de imágenes en 4 etapas distintas, en las cuales la cantidad de ejemplos de entrenamiento fue aumentando. En una primera etapa se etiquetaron 101 imágenes, con etapas subsecuentes usando 202, 302 y 402 imágenes sucesivamente.

El servicio de Roboflow, que fue usado en el etiquetado de imágenes, cuenta con funciones que permiten exportar las imágenes segmentadas en conjunto con las imágenes originales usando una resolución que puede ser modificada dentro de este servicio. La resolución elegida para el entrenamiento tanto de las imágenes originales como de las máscaras fue de 1024x1024 píxeles. Además de esto, el servicio Roboflow permite la creación de ejemplos de entrenamiento que hacen uso de la técnica de aumentación de datos. Esta técnica hace cambios aleatorios a la imagen, como lo son aumentos, transformaciones de colores, cambios en el color, así como acercamientos; entre otras cosas.

Para este trabajo, además de los sets de entrenamiento que se describieron anteriormente, se crearon más sets que hacen uso de la técnica de aumentación de datos, en los cuales se logró triplicar la cantidad de ejemplos de entrenamiento. Sin embargo, debido a problemas con las

imágenes obtenidas por el servicio, se decidió no hacer uso de los sets de entrenamiento creados con aumentación de datos.

Teniendo los sets de entrenamiento, el siguiente paso que se llevó a cabo consistió en el entrenamiento del modelo de inferencia, lo cual se describe a continuación.

3.4 Entrenamiento del Modelo y Resultados de Entrenamiento

El entrenamiento del modelo de red neuronal U-net se llevó a cabo por medio del servicio Google Colab. Google Colab provee los medios necesarios para ejecutar códigos escritos en Python sin tener que instalar una interfaz gráfica para programar o el intérprete necesario para este lenguaje de programación. Sin embargo, la característica más importante de Colab es que permite el uso de una unidad de procesamiento gráfico para ejecutar códigos, lo cual es de alta importancia en aplicaciones que requieren gran cantidad de entrenamiento, como lo es el entrenamiento de un modelo de red neuronal que usa imágenes para entrenar. Por esta razón, se eligió Google Colab para el entrenamiento.

Sin embargo, existen límites de uso de Google Colab, así como la GPU que se ofrece de forma gratuita y, aunque existen planes de pago para obtener una mayor cantidad de procesamiento, para este trabajo no se hizo uso de algún plan de pago. Aunque esto representa limitaciones en el número de épocas de entrenamiento, así como en el tamaño de lote que puede usarse para el entrenamiento, estas limitaciones no fueron impedimento para llegar a un modelo de entrenamiento con buena precisión.

Acerca de la etapa de entrenamiento, se debe comentar que se hicieron varios modelos de entrenamiento, los cuales variaron en el número de imágenes utilizadas, el número de épocas de entrenamiento, el número de imágenes usadas para cada lote de entrenamiento, así como la función de optimización usada en el proceso de entrenamiento.

Dentro del entrenamiento se destacan dos etapas diferentes. La primera fue una etapa de pruebas en las cuales se hizo uso de 101 imágenes y una cantidad de épocas que van de 100 a 500 épocas de entrenamiento. Con 101 imágenes de entrenamiento, la pérdida disminuyó,

mientras que la precisión aumentó para los ejemplos de entrenamiento, lo cual es lo que se espera. Sin embargo, para los datos de validación, la pérdida fue aumentando y la precisión de validación estuvo alrededor de 0.75, lo cual no es ideal. Estos resultados se muestran en la tabla 1 que se presenta a continuación.

Tabla 2.

Resultados de la primera etapa de entrenamiento usando 101 imágenes

Épocas	Pérdida	Precisión	Pérdida de Validación	Precisión de Validación
100	0.0850	0.9733	1.3152	0.7520
200	0.0828	0.9720	1.1540	0.7576
300	0.0181	0.9926	1.9040	0.7567
400	0.0131	0.9945	2.3604	0.7433
500	0.0111	0.9954	2.4750	0.7479

Estos primeros resultados que forman parte de una primera etapa de entrenamiento muestran una red neuronal que pudo tener buen desempeño en los ejemplos de entrenamiento, pero que falló en los ejemplos de validación ya que la pérdida en la validación solo fue incrementándose, mientras que la precisión se estancó cerca del 75%. Para esta primera etapa de modelos entrenados, no se llevó a cabo la medición de intersección sobre unión, la cual es una métrica importante para la evaluación de modelos de entrenamiento de detección de objetos.

Ya que la precisión en los datos de validación obtenida en la primera etapa de entrenamiento no es competente, se optó por volver a la fase de entrenamiento. En esta nueva etapa de entrenamiento se aumentó el tamaño de imágenes etiquetadas, llegando a una cantidad de 402 en total, todas con resolución de 1024x1024 píxeles.

Para el entrenamiento, en el código de la red neuronal se eligió un tamaño de lote de 4 imágenes, se decidió dedicar el 75% de las imágenes para el entrenamiento y el 25% para la validación. Sin embargo, el punto más importante a destacar de esta segunda etapa de

entrenamiento se tiene el uso de “callbacks” en el entrenamiento de la red. Estos “callbacks” consisten en guardar los pesos que son el resultado de un entrenamiento del modelo para después comenzar otra etapa de entrenamiento con los pesos previamente entrenados como punto de partida.

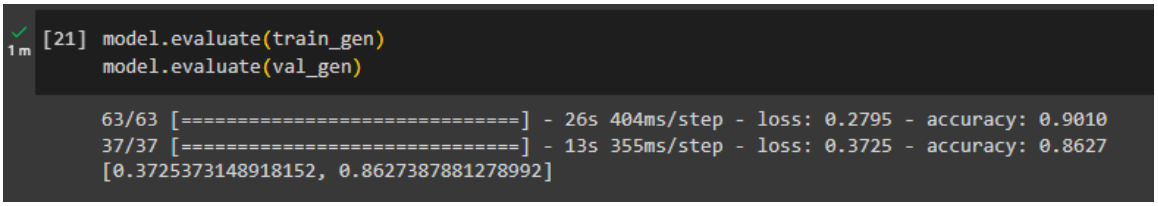
Haciendo uso de “callbacks”, se obtuvieron modelos de entrenamiento luego de una cantidad diversa de épocas de entrenamiento. Se comenzó con 100 épocas de entrenamiento y se fue aumentando la cantidad hasta tener un modelo que tuvo un total de 175 épocas de entrenamiento. Este modelo usó la función de optimización Adam en cada una de las etapas de entrenamiento hasta llegar a las 175 épocas mencionadas previamente. Este modelo obtuvo una precisión cercana al 85%, lo cual es un incremento considerable.

Sin embargo, también se decidió llevar a cabo el entrenamiento hasta llegar a una cantidad de 180 épocas, usando el optimizador de gradiente descendiente estocástico en lugar de Adam. Este cambio permitió aumentar ligeramente la precisión de validación, llegando finalmente a 86%.

Esta precisión se obtuvo en el código por medio del comando “evaluate”, presente en las librerías para la creación de modelos de entrenamiento, y se especifica sobre cuál grupo de datos se desea calcular la precisión. En la primera línea de código se obtiene la precisión en el grupo de imágenes de entrenamiento, mientras que, en la segunda, se hace lo propio con el grupo de imágenes de validación. Esta ejecución de código se muestra a continuación en la figura siguiente donde se puede notar que la precisión en el grupo de imágenes de entrenamiento rondó el 90%, mientras que, para el grupo de imágenes de validación, se logró un 86% de precisión, los cuales muestran un modelo de entrenamiento competente.

Figura 6.

Evaluación del Modelo de Entrenamiento usando el Gradiente Descendiente Estocástico como Optimizador



Por la precisión obtenida en el modelo que tuvo 180 épocas de entrenamiento, usando el gradiente descendiente estocástico, se decidió usar este modelo de entrenamiento para la inferencia en ejemplos. Sin embargo, la precisión de un modelo de red neuronal que tiene una aplicación de detección de objetos puede ser una métrica engañosa. Esto es debido a que en tareas de detección como lo es este trabajo, puede ocurrir la situación en la cual el modelo está prediciendo y acertando con las clases que tienen mucha más representación que las otras.

Es por esta razón por la cual se usan mediciones de evaluación de un modelo de entrenamiento de red neuronal como el índice de Jaccard para medir la coincidencia entre el área que predijo el modelo de entrenamiento y el área que se había etiquetado en el proceso de segmentación. En este caso, el índice de Jaccard se puede calcular comparando la imagen que forma la inferencia que el modelo predijo contra la imagen que fue segmentada como parte del procesamiento de imágenes previo al entrenamiento.

Se llevó a cabo este proceso de cálculo por medio de un código en Python, haciendo uso de 195 imágenes predichas por el modelo y 195 máscaras correspondientes a cada una de las predicciones del modelo, con la finalidad de compararlas una por una. En estas comparaciones, los resultados se aproximaron a un rango entre 0.40 y 0.60 en el índice de Jaccard. Considerando que un índice de Jaccard de 0.50 es considerado aceptable, se puede decir que el modelo también es aceptable cuando al ser evaluado desde este punto de vista. Estos resultados se describirán en mayor detalle en la siguiente sección.

Capítulo 4. Resultados

El resultado principal del presente trabajo consiste en la creación de un modelo de entrenamiento para la detección de lesiones de mosquita blanca en el tejido de hojas de pepino. El modelo hizo uso de una versión de la arquitectura U-net, está entrenado con imágenes de una planta de pepino que formaba parte de un cultivo casero y fue entrenado en el servicio de Google Colab, que es ampliamente usado para el entrenamiento de redes neuronales gracias a la posibilidad de usar una GPU para el entrenamiento.

Este modelo tiene una precisión de validación que supera el 85%. Y aunque esta métrica es muy buena, el resultado más importante que se desea mencionar en este caso es el índice de Jaccard que este modelo obtuvo. Ya que una forma de medir el índice de Jaccard se tiene por medio de comparar una imagen que predijo el modelo con la misma imagen que fue etiquetada en pasos previos de este trabajo, se obtuvieron 195 resultados de estas imágenes. Estas imágenes fueron elegidas de forma aleatoria por medio del programa usado para el entrenamiento de la red neuronal, el cual permite separar las imágenes entre un grupo de imágenes para el entrenamiento y otro para la validación. El índice de Jaccard arroja como resultado es una medición que va de 0 a 1 que representa las áreas que coinciden entre la máscara segmentada y la que el modelo predijo. De estas 195 imágenes usadas en esta etapa, el valor mínimo fue de 0.3985, mientras que el valor máximo es de 0.6116. Además, el promedio estuvo dentro de 0.5657 para 200 imágenes elegidas de forma aleatoria. Finalmente, la desviación estándar fue de 0.0479.

Además, se hizo este proceso de índice de Jaccard para las imágenes obtenidas con el modelo que usó 175 épocas de entrenamiento usando el optimizador Adam para el entrenamiento, esta vez usando la cantidad de 195 imágenes para la comparación necesaria para llevar a cabo el índice de Jaccard. En este caso, el valor mínimo de índice de Jaccard obtenido fue de 0.3896, el valor máximo fue de 0.6060, mientras que se tuvo un promedio de 0.5610 y una desviación estándar de 0.0403. Las medidas de tendencia central y de dispersión que se describieron previamente se encuentran en la tabla a continuación y fueron calculadas por medio del software Microsoft Excel.

Tabla 3.

Medidas Estadísticas de los Índices de Jaccard Obtenidos

Modelo de Entrenamiento Utilizado	180 épocas, optimizador SGD	175 épocas, optimizador Adam
Mínimo	0.3985	0.3896
Máximo	0.6116	0.6060
Promedio	0.5657	0.5610
Desviación Estándar	0.0479	0.0403

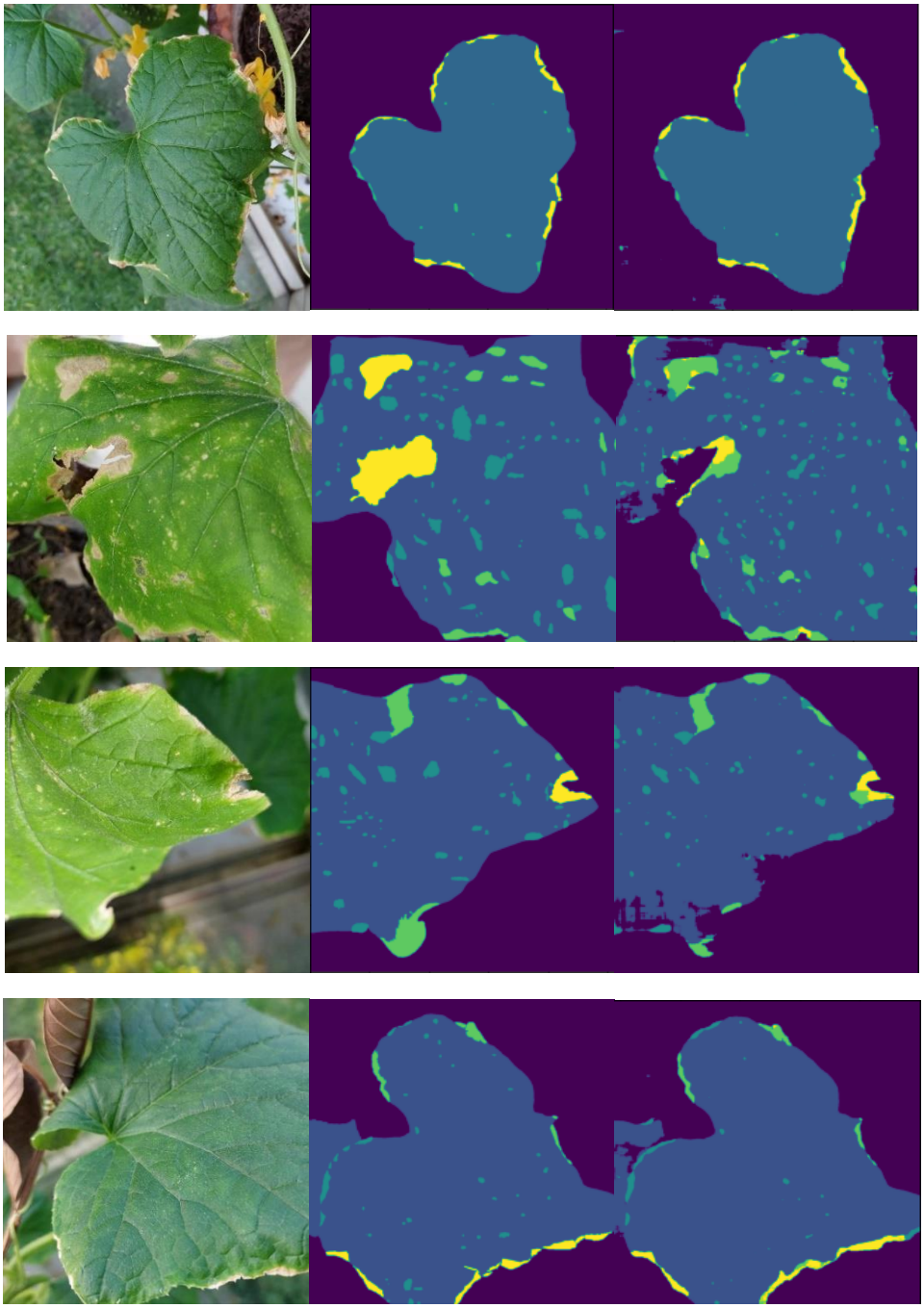
Aunque las mediciones como la precisión de validación y el índice de Jaccard permiten entender el desempeño de un modelo desde el punto de vista numérico, es necesario entender su desempeño desde un punto de vista visual. Por esto, se presenta el siguiente grupo de imágenes donde se compara la imagen original, con la máscara segmentada y finalmente la inferencia que el modelo llevó a cabo, mostradas en este orden de izquierda a derecha. Todas estas predicciones fueron hechas haciendo uso el modelo de entrenamiento que usó el gradiente descendiente estocástico como optimizador.

En este caso, es notorio que el modelo puede imitar el contorno de la hoja junto de manera correcta, así como que también logra reproducir las lesiones en la hoja causadas por la mosquita de manera competente. Sin embargo, las lesiones de mosquita blanca en algunos casos no fueron clasificadas de forma completamente correcta, lo cual puede ser atribuido a la diferencia entre la cantidad de lesiones de cada uno de los niveles de avance en las imágenes de entrenamiento, en las cuales el primer nivel de avance se presenta en una proporción mucho más alta que los dos niveles sucesivos.

Dentro de las imágenes mostradas a continuación, el color morado es el área denotada como el fondo de la imagen, un tono azul representa la hoja y los tonos verdes y amarillos representan las lesiones de la mosquita blanca, siendo amarillo el nivel más alto presente en la imagen.

Figura 7.

Imágenes usadas en el entrenamiento (izquierda), Máscaras (centro) y Predicciones del Modelo (derecha).



Como se puede notar en las imágenes anteriores, las predicciones creadas por el proceso de inferencia usando el modelo de red neuronal que se entrenó como parte de la metodología de este trabajo se acercaron bastante a las máscaras que fueron etiquetadas a mano como parte del preprocesamiento de imágenes previo al entrenamiento de la red. Como se comentó previamente, el promedio de todos los índices de Jaccard de las imágenes comparadas estuvo en 0.56 aproximadamente; esta cifra es mayor a 0.50, lo cual se considera como un índice de Jaccard competente o bueno. Además, la precisión del modelo se aproximó a 86%. Todas estas estadísticas del desempeño del modelo permiten mostrar que el modelo funciona bien y es competente. El índice de Jaccard es la medición más fiel al desempeño del modelo, ya que permite comprobar que el modelo se encuentra prediciendo las áreas que pertenecen a alguna de las clases de manera correcta. Este tipo de desempeño es esencial en aplicaciones de detección de objetos, en las cuales es necesario que el modelo haga la predicción de las clases correctas en la localización correcta de la imagen. Es fácil que los modelos logren una precisión alta, sin que esto implique que el modelo está haciendo la detección de las áreas pertenecientes a las clases de manera correcta.

Estos resultados, que abarcan el modelo de entrenamiento y las formas para comprobar su desempeño, que se describieron previamente, permiten concluir que el modelo logra un desempeño aceptable con base en la precisión en los datos de validación y en el índice de Jaccard. Sin embargo, el modelo aún tiene una gran cantidad de oportunidades de mejora, las cuales se describen a continuación.

Una de las formas más sencillas para lograr este objetivo es agregar una mayor cantidad de imágenes segmentadas y entrenar el modelo con ellas. Ya sea usando las imágenes que fueron tomadas como parte de la metodología de este trabajo, o llegando a acuerdos con productores para obtener imágenes de plantaciones a mayor escala, una mayor cantidad de imágenes puede llevar a un modelo más robusto. Agregar imágenes que provengan del cultivo casero que se hizo para este trabajo en conjunto con imágenes de campo debe ser la manera por la cual se obtendrá un modelo de entrenamiento más robusto que pueda identificar las lesiones aun cuando las imágenes presenten un nivel de variabilidad importante. Se debe agregar que este trabajo originalmente esperaba contar con imágenes provistas por productores; sin

embargo, esto no se concretó, por lo cual se optó por el cultivo casero y la toma de imágenes propias. Esta es una de las limitaciones presentes en el trabajo referentes a las imágenes.

La otra limitación referente a las imágenes se encuentra en el proceso de segmentación. Aun cuando herramientas como Roboflow permiten un proceso de segmentado sencillo, este proceso puede llegar a ser tardado. Para este trabajo se decidió llegar a la segmentación de 400 imágenes ya que se consideró que estas serían cantidad suficiente para un modelo de inferencia competente, lo cual se logró. Sin embargo, y como se comentó en el párrafo anterior, una mayor cantidad de imágenes segmentadas permite una mejor precisión y un índice de Jaccard más alto.

Otra limitación que debe destacarse se encuentra en la cantidad de procesamiento computacional necesaria para el entrenamiento de la red. Una computadora personal de gama media tardaría demasiado tiempo. Es por esta razón que se optó por el uso de la plataforma de Google Colab para la ejecución del entrenamiento de la red, la cual permite un entrenamiento más rápido gracias a la posibilidad de ejecutar códigos en un servidor que cuenta con una GPU. Sin embargo, la versión gratuita de Google Colab también cuenta con límites en la capacidad de entrenamiento, por lo cual no se optó por entrenamientos con una mayor cantidad de épocas y lo cual llevó al uso de "callbacks" para el entrenamiento. En este caso en concreto, debido a los límites de memoria de Google Colab, el tamaño de lote nunca pudo ser mayor de 8 para el entrenamiento del modelo, y fue necesario llevar a cabo el proceso de entrenamiento en etapas que constaban de alrededor de 100 épocas, hasta obtener un modelo con una precisión de validación que se acercara al 85% como mínimo. Esta limitación no presentó una gran cantidad de repercusiones en el desarrollo del proyecto, considerando que el uso de "callbacks" fue uno de los factores que permitió elevar la precisión del modelo de alrededor del 75% a cerca del 85%.

Otra limitación presente en los resultados se tiene debido a que el producto de este trabajo consta del modelo de entrenamiento. Aunque se hicieron aplicaciones simples para la predicción con la finalidad de evaluar el desempeño del modelo, no pueden usarse para inferencia de nuevos ejemplos de parte de los usuarios. Como se describió en este trabajo, algunas aplicaciones como la de Esgario et al., ya lograron tanto el entrenamiento de una red

neuronal para detección de enfermedades como la implementación del modelo obtenido para la inferencia en nuevos ejemplos (Esgario et al., 2022). Esta limitación es la más importante, ya que el desarrollo de una aplicación amigable para el usuario permite la implementación de este modelo en el mundo real y, como se comentó, ya existen aplicaciones que lograron la implementación en la literatura.

Sin embargo, aunque este trabajo tuvo que lidiar con ciertas limitaciones importantes, se logró el objetivo principal de este trabajo, el cual era la creación del modelo. Aun cuando la implementación del modelo en aplicaciones que permitan su uso en la inferencia es primordial, el entrenamiento del modelo, así como la generación y segmentado de las imágenes son pasos previos y altamente importantes en el proceso para llevar este tipo de aplicaciones al mundo real. Aun con las limitaciones descritas previamente, se obtuvieron resultados competentes, los cuales también se han descrito con anterioridad y permiten llegar a la conclusión de que las aplicaciones de visión artificial pueden llegar a ser primordiales en una gran cantidad de campos. Con la gran cantidad de investigaciones referentes a las arquitecturas de red neuronal usadas, a los procesos de segmentado, y a aplicaciones que ya son parte de la vida cotidiana de muchos campos de estudio, es seguro decir que la visión artificial está aquí para quedarse y seguirá siendo parte importante de una gran cantidad de procesos.

Capítulo 5. Conclusiones y Recomendaciones

En la introducción de este trabajo, se designó que la hipótesis alternativa sería que la combinación entre la segmentación semántica y la red neuronal U-net pueden detectar lesiones causadas por la plaga de mosquita blanca en plantas de pepino. El presente trabajo tuvo como meta probar que se podía lograr este modelo usando la red neuronal U-net y la segmentación semántica, basado en la gran cantidad de aplicaciones diversas que se han presentado que hacen uso de esta combinación de técnicas.

Este trabajo logró lo propuesto en la hipótesis alternativa, obteniendo un modelo de entrenamiento que puede inferir en lesiones en las hojas causadas por la mosquita blanca. Este objetivo fue cumplido, en conjunto con la evaluación del modelo en base al índice de Jaccard y a su precisión de validación. El resultado final de este trabajo es el modelo de entrenamiento, así como las imágenes tomadas para el entrenamiento, las cuales generaron un set de datos que consta de 1924 imágenes, en diferentes condiciones de iluminación y ángulos diferentes.

Este trabajo es resultado de un proceso de investigación, el cual sufrió una cantidad de limitaciones a lo largo de los pasos de este proyecto. Dentro de estas limitaciones, se tiene en primer lugar la falta de imágenes provistas por un productor de pepino. Esta limitación es importante debido a que se esperaba contar con imágenes directamente tomadas por un productor para tener ejemplos del lugar donde se da la producción de hortalizas. En este caso, de haber obtenido las imágenes por medio del productor, se tendrían ejemplos más diversos que podrían otorgarle al modelo otro nivel de robustez.

Otra limitación presente en el trabajo se tiene en el modelo de entrenamiento, el cual, aunque funciona correctamente, pudo ser entrenado tanto con imágenes provistas por productores como por una mayor cantidad de épocas y un mayor tamaño de lote. Aunque esto no representa una gran limitación en el desarrollo de este proyecto, una mayor cantidad de imágenes hubiera hecho el modelo más robusto y un entrenamiento con diferentes parámetros podría haber llevado a la obtención de un modelo con un índice de Jaccard y una precisión más altos. Como se comentó previamente, no se pudo hacer uso de una mayor cantidad de épocas de entrenamiento o de tamaño de lote, debido a que un tamaño de lote mayor de 4

sobrepasaba la memoria que Google Colab provee. Por otro lado, aunque una cantidad mayor de épocas de entrenamiento puede mejorar la precisión, se hace necesario el uso de "callbacks" para el entrenamiento de la red.

Por lo anterior, algunas de las recomendaciones que se hacen para trabajos subsecuentes en este campo son las siguientes. En primer lugar, se tiene la recomendación de etiquetar más imágenes para usarlas en otro entrenamiento del modelo, en busca de una mejor precisión. Además, en este punto de agregar más imágenes como ejemplos de entrenamientos, se sugiere agregar imágenes que provengan de invernaderos reales.

La siguiente sugerencia es referente al entrenamiento de la red. Como se ha comentado, el entrenamiento estuvo limitado a las capacidades que Google Colab ofrece de manera gratuita; por lo que, pagar por una mayor cantidad de procesamiento, o usar una computadora con una unidad de procesamiento gráfico permitirá cambiar el tamaño del lote, la resolución de las imágenes y la cantidad de épocas de entrenamiento, con la finalidad de aumentar la precisión y el índice de Jaccard.

Sin embargo, la sugerencia principal que se desea resaltar de este trabajo es el desarrollo de una aplicación que haga uso ya sea el modelo de entrenamiento desarrollado en este trabajo o de una versión mejorada de este modelo para obtener una detección casi instantánea. Concretamente, se puede desarrollar una aplicación de teléfono celular que use la cámara para la captura de imágenes y se lleve a cabo la inferencia en esta imagen con el modelo de entrenamiento. Esta inferencia podría darse en el mismo teléfono celular o por medio de enviar a la imagen a un servidor en la nube para que la inferencia se lleve a cabo en este punto. En resumen, esta es la recomendación que hará de este trabajo una aplicación que puede ser usada en el mundo real, ya sea por productores de cualquier escala, así como por gente con huertos caseros.

Finalmente, la última recomendación que se plantea es explorar el desempeño de una aplicación de detección por medio de algún otro algoritmo u otra arquitectura de red neuronal como lo puede ser el algoritmo YOLO. Este entrenamiento implica un etiquetado nuevo de las imágenes, pero fuera del proceso de etiquetado, tanto el entrenamiento por medio de

YOLO como la inferencia usando el modelo de entrenamiento son un poco más sencillos. El usar el algoritmo YOLO en lugar de la U-net que usó este trabajo puede llevar a un modelo de inferencia más rápido, en conjunto con una mayor precisión, y también podría facilitar la implementación del modelo en una aplicación de tiempo real.

En general, este trabajo logró presentar un modelo de entrenamiento que puede inferir la presencia y el nivel de las lesiones causadas por la mosquita blanca en plantas de pepino. Este modelo de entrenamiento puede ser usado en aplicaciones diversas de detección, las cuales podrían incluso incluir la detección de otras enfermedades y otras plantas. Lo que se desea destacar de lo anterior es que las mejoras pueden ser variadas y que justamente se espera que este trabajo obtenga una continuación, ya sea que se usen las imágenes, el modelo de entrenamiento o la investigación como base de trabajos nuevos. Al igual que los trabajos previos que fueron referenciados en este trabajo, se espera que este trabajo sea justamente el referente para trabajos futuros.

Finalmente, las condiciones actuales del planeta obligan a la humanidad a buscar el mejor aprovechamiento de los recursos naturales de cualquier manera que sea posible. La solución de problemas por medio de máquinas de aprendizaje y técnicas de inteligencia artificial permite ver problemas desde otro ángulo y proveer soluciones distintas. Aquí, se hace uso de estas técnicas en busca de una predicción competente que pretende reducir las pérdidas causadas por la mosquita blanca. Esta predicción oportuna que ofrece el modelo permite una reducción en las pérdidas, da pie a un uso más eficiente de los recursos naturales, el cual es bienvenido.

Referencias

Abubakar, M., Koul, B., Chandrashekar, K., Raut, A., & Yadav, D. (2022). Whitefly (Bemisia tabaci) Management (WFM) Strategies for Sustainable Agriculture: A Review. *Agriculture*, *12*(9), 1317. https://doi.org/10.3390/agriculture12091317

Aroiee, H., Mosapoor, S., & Karimzadeh, H. (2005). Control of greenhouse whitefly (Trialeurodes vaporariorum) by thyme and peppermint. *KMITL Science Journal*, *5*(2), 511-514.

Asgari Taghanaki, S., Abhishek, K., Cohen, J. P., Cohen-Adad, J., & Hamarneh, G. (2021). Deep semantic segmentation of natural and medical images: A review. *Artificial Intelligence Review*, *54*(1), 137-178. https://doi.org/10.1007/s10462-020-09854-1

Barbedo, J. G. A. (2014). Using digital image processing for counting whiteflies on soybean leaves. *Journal of Asia-Pacific Entomology*, *17*(4), 685-694. https://doi.org/10.1016/j.aspen.2014.06.014

Barraza-Álvarez, F. V. (2015). Calidad morfológica y fisiológica de pepinos cultivados en diferentes concentraciones nutrimentales. *Revista Colombiana de Ciencias Hortícolas*, *9*(1), 60. https://doi.org/10.17584/rcch.2015v9i1.3746

Byrne, D. N., & Bellows, T. S. (s. f.). *Whitefly Biology*.

C. K., S., C. D., J., & Patil, N. (2022). Cardamom Plant Disease Detection Approach Using EfficientNetV2. *IEEE Access*, *10*, 789-804. https://doi.org/10.1109/ACCESS.2021.3138920

Cabello García, T., Carricondo Martínez, I., Justicia del Río, L., & Belda Suárez, J. E. (1996). *Biología y control de las especies de mosca blanca Trialeurodes vaporariorum (Gen.) y Bemisia tabici (West.) (Hom.; Aleyroridae) en cultivos horticolas en invernaderos*. Junta de Andalucía. Consejería de Agricultura y Pesca.

Cao, K., & Zhang, X. (2020). An Improved Res-UNet Model for Tree Species Classification Using Airborne High-Resolution Images. *Remote Sensing*, *12*(7), 1128. https://doi.org/10.3390/rs12071128

Carapia Ruiz, V. E., & Castillo-Gutiérrez, A. (2013). Estudio comparativo sobre la morfología de Trialeurodes vaporariorum (Westwood) y Bemisia tabaci (Gennadius) (Hemiptera: Aleyrodidae). *ACTA ZOOLÓGICA MEXICANA (N.S.)*, *29*(1), 178-193. https://doi.org/10.21829/azm.2013.291394

Chen, J.-W., Lin, W.-J., Cheng, H.-J., Hung, C.-L., Lin, C.-Y., & Chen, S.-P. (2021). A Smartphone-Based Application for Scale Pest Detection Using Multiple-Object Detection Methods. *Electronics*, *10*(4), 372. https://doi.org/10.3390/electronics10040372

Cho, J., Choi, J., Qiao, M., Ji, C., Kim, H., Uhm, K., & Chon, T. (s. f.). Automatic identification of whiteflies, aphids and thrips in greenhouse based on image analysis. *International Journal of Mathematics and Computers in Simulation*.

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1800-1807. https://doi.org/10.1109/CVPR.2017.195

Chowdhury, M. E. H., Rahman, T., Khandakar, A., Ayari, M. A., Khan, A. U., Khan, M. S., Al-Emadi, N., Reaz, M. B. I., Islam, M. T., & Ali, S. H. M. (2021). Automatic and Reliable Leaf Disease Detection Using Deep Learning Techniques. *AgriEngineering*, *3*(2), 294-312. https://doi.org/10.3390/agriengineering3020020

Coombe, P. E. (1982). Visual behaviour of the greenhouse whitefly, Trialeurodes vaporariorum. *Physiological Entomology*, *7*(3), 243-251. https://doi.org/10.1111/j.1365-3032.1982.tb00297.x

Csurka, G., Larlus, D., & Perronnin, F. (2013). What is a good evaluation measure for semantic segmentation? *Procedings of the British Machine Vision Conference 2013*, 32.1-32.11. https://doi.org/10.5244/C.27.32

Dastres, R., & Soori, M. (2021). *Artificial Neural Network Systems*.

Elizabeth Castillo, E. M. (2015). *Costo de producción del pepino (Cucumis Sativus L.), bajo condiciones protegidas en macro túnel en la Universidad Nacional Agraria, Enero-Abril 2014*. Universidad Nacional Agraria.

Esgario, J. G. M., De Castro, P. B. C., Tassis, L. M., & Krohling, R. A. (2022). An app to assist farmers in the identification of diseases and pests of coffee leaves using deep

learning. *Information Processing in Agriculture*, *9*(1), 38-47. https://doi.org/10.1016/j.inpa.2021.01.004

Espinoza, K., Valera, D. L., Torres, J. A., López, A., & Molina-Aiz, F. D. (2016). Combination of image processing and artificial neural networks as a novel approach for the identification of Bemisia tabaci and Frankliniella occidentalis on sticky traps in greenhouse agriculture. *Computers and Electronics in Agriculture*, *127*, 495-505. https://doi.org/10.1016/j.compag.2016.07.008

Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., & Garcia-Rodriguez, J. (2017). *A Review on Deep Learning Techniques Applied to Semantic Segmentation* (arXiv:1704.06857). arXiv. http://arxiv.org/abs/1704.06857

Ghongade, D. S., & Sangha, K. S. (2021). Efficacy of biopesticides against the whitefly, Bemisia tabaci (Gennadius) (Hemiptera: Aleyrodidae), on parthenocarpic cucumber grown under protected environment in India. *Egyptian Journal of Biological Pest Control*, *31*(1), 19. https://doi.org/10.1186/s41938-021-00365-x

Ghosh, S. K. (2022). *CROPS AND USE OF SAFE PESTICIDES FOR BIODIVERSITY CONSERVATION*.

Golshan, R., Shishehbor, P., & Esfandiari, M. (2023). *Biological characteristics, functional and numerical responses of the predatory mite Amblyseius swirskii (Acari: Phytoseiidae) feeding on cotton whitefly, Bemisia tabaci (Hemiptera: Aleyrodidae)*.

González Acevedo, J. C. (2021). *Diseño de una escala diagramática para evaluar la severidad del mildiú (Pseudoperonospora cubensis) en pepino en Morelos, México* [Tesis]. Facultad de Ciencias Agropecuarias.

Guo, Y., Liu, Y., Georgiou, T., & Lew, M. S. (2018). A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval*, *7*(2), 87-93. https://doi.org/10.1007/s13735-017-0141-z

Gupta, N. (2013). *Artificial Neural Network*.

Huang, K.-M., Guan, Z., & Hammami, A. (2022). The U.S. Fresh Fruit and Vegetable Industry: An Overview of Production and Trade. *Agriculture*, *12*(10), 1719. https://doi.org/10.3390/agriculture12101719

Jia, H., & Wang, H. (2021). Introductory Chapter: Studies on Cucumber. En H. Wang (Ed.), *Cucumber Economic Values and Its Cultivation and Breeding*. IntechOpen. https://doi.org/10.5772/intechopen.97360

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255-260. https://doi.org/10.1126/science.aaa8415

Karam, C., Awad, M., Abou Jawdah, Y., Ezzeddine, N., & Fardoun, A. (2022). GAN-based semi-automated augmentation online tool for agricultural pest detection: A case study on whiteflies. *Frontiers in Plant Science*, *13*, 813050. https://doi.org/10.3389/fpls.2022.813050

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, *151*, 107398. https://doi.org/10.1016/j.ymssp.2020.107398

Legaspi, K. R. B., Sison, N. W. S., & Villaverde, J. F. (2021). Detection and Classification of Whiteflies and Fruit Flies Using YOLO. *2021 13th International Conference on Computer and Automation Engineering (ICCAE)*, 1-4. https://doi.org/10.1109/ICCAE51876.2021.9426129

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(12), 6999-7019. https://doi.org/10.1109/TNNLS.2021.3084827

Lin, M., Momin, S., Lei, Y., Wang, H., Curran, W. J., Liu, T., & Yang, X. (2021). Fully automated segmentation of brain tumor from multiparametric MRI using 3D context deep supervised U-Net. *Medical Physics*, *48*(8), 4365-4374. https://doi.org/10.1002/mp.15032

Liu, J., & Wang, X. (2020). Tomato Diseases and Pests Detection Based on Improved Yolo V3 Convolutional Neural Network. *Frontiers in Plant Science*, *11*, 898. https://doi.org/10.3389/fpls.2020.00898

Lutz, M. (2009). *Learning Python* (4th ed). O'Reilly.

Mahesh, B. (2018). *Machine Learning Algorithms—A Review*. *9*(1).

Mainali, B. P., & Lim, U. T. (2008). Use of flower model trap to reduce the infestation of greenhouse whitefly on tomato. *Journal of Asia-Pacific Entomology*, *11*(2), 65-68. https://doi.org/10.1016/j.aspen.2008.04.005

Manaswi, N. K. (2018). Understanding and Working with Keras. En N. K. Manaswi, *Deep Learning with Applications Using Python* (pp. 31-43). Apress. https://doi.org/10.1007/978-1-4842-3516-4_2

Martins Crispi, G., Valente, D. S. M., Queiroz, D. M. D., Momin, A., Fernandes-Filho, E. I., & Picanço, M. C. (2023). Using Deep Neural Networks to Evaluate Leafminer Fly Attacks on Tomato Plants. *AgriEngineering*, *5*(1), 273-286. https://doi.org/10.3390/agriengineering5010018

McGlinchy, J., Johnson, B., Muller, B., Joseph, M., & Diaz, J. (2019). Application of UNet Fully Convolutional Neural Network to Impervious Surface Segmentation in Urban Environment from High Resolution Satellite Imagery. *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 3915-3918. https://doi.org/10.1109/IGARSS.2019.8900453

Milioto, A., & Stachniss, C. (2019). Bonnet: An Open-Source Training and Deployment Framework for Semantic Segmentation in Robotics using CNNs. *2019 International Conference on Robotics and Automation (ICRA)*, 7094-7100. https://doi.org/10.1109/ICRA.2019.8793510

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Moolayil, J. (2019). Keras in Action. En J. Moolayil, *Learn Keras for Deep Neural Networks* (pp. 17-52). Apress. https://doi.org/10.1007/978-1-4842-4240-7_2

Pal, A., Adhikary, R., Shankar, T., Sahu, A. K., & Maitra, S. (2020). Cultivation of Cucumber in Greenhouse. En New Delhi Publishers & S. Maitra, *Protected Cultivation and Smart Agriculture* (1.ª ed.). New Delhi Publishers. https://doi.org/10.30954/NDP-PCSA.2020.14

Parab, C. U., Mwitta, C., Hayes, M., Schmidt, J. M., Riley, D., Fue, K., Bhandarkar, S., & Rains, G. C. (2022). Comparison of Single-Shot and Two-Shot Deep Neural Network Models for Whitefly Detection in IoT Web Application. *AgriEngineering*, *4*(2), 507-522. https://doi.org/10.3390/agriengineering4020034

Ramírez Abarca, O., & Hernández Martínez, J. (2021). *ANÁLISIS ECONÓMICO DEL PEPINO PERSA EN CONDICIONES DE INVERNADERO EN GUERRERO Y ESTADO DE MÉXICO, 2020*. *48*.

Rampasek, L., & Goldenberg, A. (2016). TensorFlow: Biology's Gateway to Deep Learning? *Cell Systems*, *2*(1), 12-14. https://doi.org/10.1016/j.cels.2016.01.009

Rebollar-Rebollar, S., Ramírez-Abarca, O., & Hernández-Martínez, J. (2022). Competitividad y valor agregado de pepino Persa (Cucumis sativus L.) en agricultura por contrato: Estudio de caso. *REVISTA TERRA LATINOAMERICANA*, *40*. https://doi.org/10.28940/terra.v40i0.952

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation* (arXiv:1505.04597). arXiv. http://arxiv.org/abs/1505.04597

Ruder, S. (2017). *An overview of gradient descent optimization algorithms* (arXiv:1609.04747). arXiv. http://arxiv.org/abs/1609.04747

S, G., & A, V. (2019). *Introduction to Python programming*. CRC Press, Taylor & Francis Group.

Saeedizadeh, N., Minaee, S., Kafieh, R., Yazdani, S., & Sonka, M. (2021). COVID TV-Unet: Segmenting COVID-19 chest CT images using connectivity imposed Unet. *Computer Methods and Programs in Biomedicine Update*, *1*, 100007. https://doi.org/10.1016/j.cmpbup.2021.100007

Sani, I., Ismail, S. I., Abdullah, S., Jalinas, J., Jamian, S., & Saad, N. (2020). A Review of the Biology and Control of Whitefly, Bemisia tabaci (Hemiptera: Aleyrodidae), with Special Reference to Biological Control Using Entomopathogenic Fungi. *Insects*, *11*(9), 619. https://doi.org/10.3390/insects11090619

Sarvamangala, D. R., & Kulkarni, R. V. (2022). Convolutional neural networks in medical image understanding: A survey. *Evolutionary Intelligence*, *15*(1), 1-22. https://doi.org/10.1007/s12065-020-00540-3

Sharma, S., Sharma, S., & Athaiya, A. (2020). ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology*, *04*(12), 310-316. https://doi.org/10.33564/IJEAST.2020.v04i12.054

Shelhamer, E., Long, J., & Darrell, T. (2016). *Fully Convolutional Networks for Semantic Segmentation*. 12.

SIAP. (2022). *Avance de Siembras y Cosechas* [dataset]. https://bit.ly/3V82sFZ

Singh, M. C., Singh, J. P., Pandey, S. K., Mahay, D., & Shrivastva, V. (2017). Factors Affecting the Performance of Greenhouse Cucumber Cultivation-A Review. *International Journal of Current Microbiology and Applied Sciences*, *6*(10), 2304-2323. https://doi.org/10.20546/ijcmas.2017.610.273

Singh, P., & Manure, A. (2020). *Learn TensorFlow 2.0: Implement Machine Learning and Deep Learning Models with Python*. Apress. https://doi.org/10.1007/978-1-4842-5558-2

Staub, J. E., Robbins, M. D., & Wehner, T. C. (2008). Cucumber. *Vegetables I*, 42.

Stukenberg, N., & Poehling, H. (2019). Blue–green opponency and trichromatic vision in the greenhouse whitefly (*Trialeurodes vaporariorum*) explored using light emitting diodes. *Annals of Applied Biology*, *175*(2), 146-163. https://doi.org/10.1111/aab.12524

Sun, R.-Y. (2020). Optimization for Deep Learning: An Overview. *Journal of the Operations Research Society of China*, *8*(2), 249-294. https://doi.org/10.1007/s40305-020-00309-6

Tani, H., Kotani, R., Kagiwada, S., Uga, H., & Iyatomi, H. (2018). Diagnosis of Multiple Cucumber Infections with Convolutional Neural Networks. *2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 1-4. https://doi.org/10.1109/AIPR.2018.8707385

Tatlioglu, T. (1993). Cucumber. En *Genetic Improvement of Vegetable Crops* (pp. 197-234). Elsevier. https://doi.org/10.1016/B978-0-08-040826-2.50017-5

Tsueda, H., & Tsuchida, K. (1998). Differences in spatial distribution and life history of two sympatric whiteflies, the greenhouse whitefly (Trialeurodes vaporariorum Westwood) and the silverleaf whitefly (Bemisia argentifolii Bellows & Perring), under greenhouse and laboratory conditions. *Appl. Entomol. Zool.*, *33*, 379-383.

Ugwu, C., & Suru, S. (2021). Cosmetic, Culinary and Therapeutic Uses of Cucumber (*Cucumis sativus* L.). En H. Wang (Ed.), *Cucumber Economic Values and Its Cultivation and Breeding*. IntechOpen. https://doi.org/10.5772/intechopen.96051

Unpingco, J. (2021). *Python Programming for Data Analysis*. Springer International Publishing. https://doi.org/10.1007/978-3-030-68952-0

Van Lenteren, J. C., Van Roermund, H. J. W., & Sütterlin, S. (1996). Biological Control of Greenhouse Whitefly (Trialeurodes vaporariorum) with the ParasitoidEncarsia formosa:How Does It Work? *Biological Control*, *6*(1), 1-10. https://doi.org/10.1006/bcon.1996.0001

Velásquez - Valle, R. (2020). Presencia de Bemisia tabaci Gennadius y Trialeurodes vaporariorum Westwood en el norte-centro de México. *Revista Mexicana de Ciencias Agrícolas*, *11*(1), 213-219. https://doi.org/10.29312/remexca.v11i1.1521

Zieliński, H., Surma, M., & Zielińska, D. (2017). The Naturally Fermented Sour Pickled Cucumbers. En *Fermented Foods in Health and Disease Prevention* (pp. 503-516). Elsevier. https://doi.org/10.1016/B978-0-12-802309-9.00021-2