DETECCIÓN DE MINADORES EN LOS CULTIVOS DE CRISANTEMOS POR MEDIO DE VISIÓN ARTIFICIAL

KEVIN ALEXANDER IBARRA OROZCO

ALEXIS GARCÍA RAMÍREZ

Trabajo de grado para optar al título de

INGENIERO DE SISTEMAS Y COMPUTACIÓN

Isis Bonet Cruz, PhD

UNIVERSIDAD EIA

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

ENVIGADO

2021

Dedicamos este trabajo de grado principalmente a nuestros familiares, ya que siempre nos brindaron su apoyo incondicional para cumplir nuestros objetivos y metas, pues son quienes nos impulsan a ser siempre mejores personas.

AGRADECIMIENTOS

Le agradecemos a nuestra directora de tesis Isis Bonet Cruz, quien nos guío durante todo el proceso, apoyándonos y brindándonos sus conocimientos. Siempre estuvo dispuesta a solucionar cualquier duda encontrada en el camino, demostrando su pasión por compartir el conocimiento y ayudarnos a mejorar. Nos sentimos muy orgullosos de ser sus estudiantes. Igualmente, agradecemos a Piedad Medina gerente del cultivo Inversiones Agrícolas las Acacias SAS. Gracias a ella logramos recolectar la información necesaria para nuestro modelo y tener un mejor acercamiento a como interactuaban los minadores con los cultivos.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ECUCACIONES

LISTA DE ANEXOS

RESUMEN

El aumento de los datos y su procesamiento ha potenciado el uso de la inteligencia artificial en el sector de la agricultura. El uso de estos avances tecnológicos, permiten mejorar la predicción y diagnóstico de plagas, lo que causa un incremento en la rentabilidad de los agricultores. Dentro de estas plagas se encuentra el minador, un insecto que deposita sus huevos en las hojas de las plantas perjudicando la masa foliar y facilitando el ingreso de infecciones en la misma, produciendo pérdidas en los cultivos principalmente para los agricultores de los crisantemos. Actualmente los cultivadores para combatir esta plaga realizan monitoreos manuales periódicos, en donde se estima la presencia de minador y se toman decisiones para su control. Sin embargo, estos monitoreos son costosos y lentos para las empresas por lo que su frecuencia no es la ideal.

Se creó un repositorio de imágenes, donde éstas debían de estar seleccionadas y clasificadas. Se realizó un aumento de los datos para ayudar a robustecer el modelo y evitar un poco de ruido por la disposición que pudieran tener las imágenes. Se procedió a extraer características de los datos para luego ser ingresados a los varios modelos de machine learning seleccionados: red neuronal, máquinas de vectores de soporte, KNN. También se probaron diferentes redes profundas como: ResNet50, MobileNet, MobileNet V2, Xception3 y red YOLO. Para esta clasificación, se dividieron los datos en dos conjuntos: entrenamiento y prueba, se ingresaron a los modelos, y por último se compararon los modelos entre sí por medio de la validación cruzada para obtener el mejor clasificador.

Los resultados obtenidos indicaron que el modelo más apropiado para la detección del minador en los crisantemos fue la red YOLO, con este modelo se lograron estadísticas muy buenas en contraste a las demás, debido, en parte, al aumento de datos autónomo y facilidad de implementación.

Palabras clave: Minador, YOLO, inteligencia artificial, red neuronal y validación cruzada.

ABSTRACT

The increase in data and its processing has boosted the use of artificial intelligence in the agricultural sector. The use of these technological advances allows improving the prediction and diagnosis of pests, which causes an increase in the profitability of farmers. Within these pests is the leafminer, an insect that lays its eggs on the leaves, damaging the leaf mass of the plant and facilitating the entry of infections, producing crop losses mainly for chrysanthemum growers. Currently, growers carry out periodic manual monitoring to control this pest, where the presence of the leaf miner is estimated, and decisions are made for its control. However, this monitoring is costly and time-consuming for the companies, so its frequency is not ideal.

The procedure started with the creation of an image repository where images had to be selected and classified to avoid noise in the model. We extracted the features from the data and their augmentation to then be entered into the selected models: Neural Network, Support Vector Machines, KNN, ResNet50, MobileNet, MobileNet V2, Xception3 and YOLO network. For this classification, the data were divided into training and testing, entered into the models, and finally the models were compared with each other by cross-validation to obtain the best classifier.

The results obtained indicated that the most appropriate model for the detection of the leaf miner in chrysanthemums was the YOLO network, with this model very good statistics were achieved in contrast to the others, due in part to its increased autonomous data and ease of implementation.

Key words: Leaf miner, YOLO, artificial intelligence, neural network and cross validation.

INTRODUCCIÓN

Los minadores de hojas son insectos cuyas larvas viven y se alimentan del tejido de las hojas, provocando un daño bastante visible. El cual crea zonas blanquecinas y caminos que no siguen un patrón alguno (Salvo & Valladares, 2007). A estos caminos se les llama galerías y son un problema para la planta ya que destruyen parte de la masa foliar, lo que disminuye la capacidad fotosintética en las hojas y permite el ingreso de fitopatógenos a las plantas (Yanes Figueroa & Téllez Navarro, 2004). Adicionalmente, en el caso de las flores este daño reduce el valor estético de la misma, lo que hace que no cumpla los estándares de calidad exigidos por los países a exportar (Gonzalez et al., 2018).

Los minadores son un desafío para los cultivadores, ya que estos insectos presentan resistencia a los insecticidas. El minador, al depositar sus larvas dentro de la hoja, se aprovecha de la defensa de la planta, y cuando está dentro de ella se hace casi imposible para el insecticida lograr eliminar la plaga sin afectar la flor (Salvo & Valladares, 2007). Al intentar eliminar la plaga con insecticidas lo que sucede es que se eliminan sus depredadores naturales lo que aumenta la población de minadores. El control integral para esta plaga es el uso de sus depredadores y el corte temprano de la hoja, todo esto se hace con el fin de evitar el uso masivo de insecticidas que contaminan las aguas y eliminan la biodiversidad local (Constantino et al., 2011).

Según Díaz Pongutá entre otros (2012) los plaguicidas terminan ingresando a nuestro organismo por medio de los alimentos que ingerimos, en su estudio demostró que estos químicos terminan en alimentos como la leche, ya que la vaca se alimenta de agua o pasto contaminado. Estos químicos en nuestro organismo se han relacionado con enfermedades como cáncer, hiper o hipotiroidismo y problemas reproductivos (Yucra et al., 2008). Por lo que el uso responsable de los pesticidas se convierte en un proceso vital para la seguridad alimentaria (Peres et al., 2020).

La reducción de pesticidas además de favorecer al ambiente también lo hace para las empresas, ya que esto reduce sus costos al utilizar menos pesticidas y cuidan la salud de sus empleados. Los empleados del sector agricultor están expuestos a estos químicos y a menor

exposición, menor es el daño debido a que se ha demostrado que las personas que han estado expuestas pesticidas por tiempos prolongados sufren de intoxicaciones por estos químicos, que pueden causar desde dermatitis hasta cáncer. (Peres et al., 2020).

1 PRELIMINARES

1.1 PLANTEAMIENTO DEL PROBLEMA

Los cultivos extranjeros se han ido automatizando en los últimos años, estos han podido sobresalir en sus producciones ya sea por su calidad o masa de producción. Por el contrario, el desarrollo colombiano ha presentado tazas de automatización muy bajas en contraste con sus principales competidores. Debido a esto, Colombia no puede compararse en términos de producción con dichos países y mucho menos en términos económicos. Puesto que, al producir bienes del sector agricultor en masa más costosos debido a su falta de automatización (Londoño, 2018), se le dificulta al agro colombiano posicionarse en el mercado. Lo anteriormente mencionado, resalta la idea de que la automatización es un factor clave para acelerar las tasas de crecimiento económico de un país. (Díaz, 2017).

Colombia es el segundo país exportador de flores a nivel mundial, con una participación del 16%, lo que representa aproximadamente 239.497 toneladas de flores (Minagricultura, 2019). Esto significa un gran ingreso para el país; para el año 2018 el sector floricultor en Colombia registró un ingreso de 1.400 millones de dólares aproximadamente, lo que aportó un 17% del impuesto de renta del agro colombiano. Sin embargo, pese a que el sector agropecuario representa una gran entrada para el país, esta producción no cuenta con la tecnificación necesaria para competir con los demás países. Por lo que el gobierno colombiano ha decidido promover incentivos para ser más competitivos en el sector internacional, un ejemplo de esto son los incentivos entregados por la presidencia en el año 2016 (Minesterio de agricultura y desarrollo rural, 2016).

Los crisantemos son el tercer tipo de flor más exportada dado que se encuentran entre las más apetecidas por los extranjeros (Salamanca et al., 2019) . Al ser una flor que se produce en masa, sus cultivadores deben de lidiar con plagas y enfermedades las cuales perjudican sus ingresos, entre ellas los minadores. Para controlar estas enfermedades y plagas las empresas productoras contratan personas, las cuales se encargan de hacer muestreos aleatorios y generar un reporte que posteriormente es analizado por la empresa para así tomar acciones preventivas, curativas o de erradicación. El monitoreo es un proceso vital para los

floricultores el cual se debe de realizar diario dado de que existen plagas o enfermedades que deben ser encontradas lo más rápido posible para no afectar las camadas de flores y se evite propagar la enfermedad, pero debido a los costos que estos monitoreos representan y el tiempo invertido, las empresas suelen hacerlos semanalmente lo que pone en riesgo la producción.

Por lo tanto, se requiere de la construcción de un modelo analítico que, basado en imágenes recolectadas por nosotros mismos, permita ayudar en la predicción temprana del minador en el proceso de muestreo, para facilitarle a los trabajadores su labor y aumentar su eficiencia.

1.2 OBJETIVOS DEL PROYECTO

1.2.1 Objetivo General

Desarrollar un modelo computacional para identificar la presencia de minadores en los crisantemos a través del análisis de imágenes RGB utilizando métodos de inteligencia artificial. (prototipo)

1.2.2 Objetivos Específicos

- Recolectar imágenes RGB de un cultivo de crisantemos, desde diferentes ángulos, donde se identifique la presencia de minadores.

- Crear la base de datos para el entrenamiento, etiquetando las imágenes de acuerdo con el nivel de presencia de minadores.

- Preprocesar las imágenes para reducir el efecto del ruido y disminuir el impacto negativo en la calidad de éstas.

- Implementar un modelo computacional, basado en inteligencia artificial, que permita identificar la presencia de minadores en los crisantemos por medio de imágenes RGB.

- Evaluar la viabilidad del modelo computacional mediante la valoración de predicción de minadores en plantas de crisantemos.

1.3 MARCO DE REFERENCIA

1.3.1 Marco teórico

Las investigaciones mencionadas en este trabajo siguen la metodología planteada por Roldán, Roshan y Sánchez (2019). La cual fue una metodología generalizada de diferentes artículos sobre la clasificación de enfermedades, en donde se definieron las siguientes fases:

- Adquisición de imágenes
- Preprocesamiento
- Segmentación
- Extracción de características
- Clasificación de la enfermedad.

Para la adquisición de imágenes se procede a realizar la captura por medio de cámaras o se obtienen por medio de algún repositorio. En estas imágenes se debe capturar las partes de la planta donde sea visible el daño causado por la enfermedad, así mismo, se recomienda capturar las imágenes a distancias iguales. Posteriormente, se realiza un preprocesamiento donde se aplican diferentes técnicas para lograr maximizar las características como la eliminación de ruido, la transformación del espacio de color, la ecualización de histograma, entre otras. De esta manera se puede mejorar la precisión en la etapa de segmentación, donde se utilizarán técnicas para separar los puntos de interés y así mejorar la precisión, obteniendo imágenes más descriptivas para el modelo. En otras palabras, la segmentación es el proceso de dividir las imágenes en grupos de pixeles, separando los objetos que se encuentran en ella, para así obtener imágenes más representativas para el modelo.

Antes de clasificar la enfermedad es necesario extraer las características de las distintas regiones de interés seleccionadas para determinar con una buena precisión el nivel de sospecha, este proceso puede ser complejo, por ello existen distintos algoritmos que nos facilitan esta tarea, estos algoritmos se clasifican en 5 tipos (Roldán et al., 2019).

- Características de texturas
- Características de forma
- Características en el color
- Histograma

- PCA

Una vez que se obtienen las características, se procede a clasificar las imágenes usando los distintos algoritmos que existen, entre ellos se encuentran lógica difusa, Bayes, KNN, redes neuronales convolucionales, entre otros. Según Roldán, Roshan y Sánchez (2019) los mejores resultados para la clasificación de enfermedades en las plantas se obtienen a partir del uso de las redes neuronales convolucionales puesto que arrojan resultados muy cercanos a los que un humano experto determinaría. No obstante, en este trabajo se hace una comparación de varios modelos para ver en este problema en específico cuál se ajusta mejor.

Roldán, Roshan y Sánchez resumieron las características de los principales modelos de machine learning:

Algoritmo	Propiedades
Lógica difusa	Basado en reglas heurísticas, utilizado para procesos no lineales.
SVM	Busca un hiperplano que funciona como separador. Es muy eficiente
Bayes	Es muy eficiente en donde se utiliza y no requiere de gran cantidad de datos.
KNN	Busca las observaciones más cercanas a las que está tratando de predecir y clasifica el punto de interés basado en los datos del entrenamiento
ANN	La clasificación es muy eficiente a costa de un entrenamiento costoso computacionalmente
CNN	Tienden a ser más precisas, son costosas computacionalmente y requieren un número considerable de imágenes para resultados fiables.

Tabla 1. Características Algoritmos

1.3.1.1 Lógica difusa

La lógica difusa *o fuzzy logic*, consta de un enfoque de la informática basado en "grados de verdad" en lugar de la lógica booleana "verdadera o falsa". La lógica difusa es esencial para el desarrollo de capacidades similares a las humanas en la inteligencia artificial. La IA es la representación de habilidades cognitivas humanas generalizadas en software para que, frente a una tarea desconocida, el sistema pueda encontrar una solución. Añadimos datos y formamos una serie de verdades parciales que agregamos aún más en verdades superiores que, a su vez, cuando se superan algunos umbrales, causan ciertos resultados adicionales, como la reacción motora. Un tipo similar de proceso se utiliza en redes neuronales, sistemas expertos y otras aplicaciones de inteligencia artificial (Rouse, 2016).

Por último, otra implementación poco utilizada de la lógica difusa pero que genera buenos resultados, son los sistemas de control difusos, son una técnica usada en la inteligencia artificial para el procesamiento de imágenes, el cual cuenta con una suavidad en la clasificación de sistemas. Además, tiene un esquema de lenguaje y estandarización de los modelos difusos que lo convierten en un algoritmo sencillo de implementar (Lim et al., 2015).

1.3.1.2 Árboles de decisión

Los árboles de decisiones son una estructura de datos muy utilizada; esta comienza con un nodo principal llamado raíz, de este surgen todas las ramas y estas a su vez están compuestas por más nodos en donde las decisiones sobre los valores de atributo de condición se realizan en cada uno, lo que permite avanzar en el árbol. El avance en el árbol se detiene cuando llega a un nodo final llamado “hoja”, al recorrido de la raíz a la hoja se le denomina decisión. (Beynon, Peel, & Tang, 2004).

1.3.1.3 Árboles de decisión con lógica difusa

Los árboles de decisión con lógica difusa surgieron para combinar los árboles de decisión con el razonamiento ofrecido por la representación difusa, con la intención de utilizar las ventajas que ofrecen ambos. Los árboles de decisión facilitan la comprensión de los problemas y la lógica difusa tiene la capacidad de trabajar con información incierta (Janikow, 1996), por ello los árboles de decisión difusa ofrecen una alternativa más comprensible para el análisis de problemas que necesiten predicción y clasificación con un buen grado de exactitud (Sulla, Gómez, & Cossio, 2018).

1.3.1.4 Algoritmos de Inteligencia de enjambre

Los algoritmos de esta rama de la inteligencia artificial se basan en el comportamiento colectivo de sistemas autoorganizados. Inspirados comúnmente en los sistemas biológicos como en las abejas u hormigas, estos sistemas de inteligencia de enjambre se forman por elementos simples que siguen reglas sencillas e interactúan entre ellos hasta forma sistemas complejos. Estas técnicas utilizan pocos parámetros de configuración lo que le da una rápida convergencia para hallar la solución (Cui et al., 2017).

1.3.1.5 Máquinas de vectores de soporte

Otro modelo computacional muy utilizado en la clasificación de imágenes son las máquinas de vectores de soporte (SVM), son un conjunto de algoritmos que pertenecen al aprendizaje profundo en donde dado un conjunto de puntos que pertenecen a una categoría, se crea un modelo capaz de predecir a que categoría pertenece el nuevo punto. Operativamente SVM se basa en la idea de encontrar la mejor línea, plano o hiperplano que divida el conjunto de datos en sus respectivas categorías. El principal objetivo del hiperplano es separar de forma óptima a los puntos de una clase de las otras. La Separación óptima es la principal propiedad de los SVM, este algoritmo busca un hiperplano que contenga la mayor distancia entre los puntos de cada clase (Rüping, 2010).

1.3.1.6 Naïve Bayes

Este algoritmo hace parte de los clasificadores probabilísticos, los cuales se basan en el teorema de Bayes. Los parámetros del modelo son estimados por medio de las frecuencias del historial de datos del entrenamiento, a esto se le conoce como la estimación de máxima verosimilitud de las probabilidades.

En palabras simples, este clasificador asume la presencia o no de una característica como independiente de las demás, en otras palabras, dicha característica no está relacionada con las otras (Frank & Bouckaert, 2006).

$$P(C|F_1 \dots, F_n) = \frac{P(F_1,\dots,F_n|C)}{P(F_1,\dots,F_n)} \quad (1)$$

Ecuación 1. Teorema de Bayes

1.3.1.7 KNN (K-vecinos más cercanos)

Este modelo pertenece a la clasificación supervisada y no es paramétrico. Su principal característica es estimar la probabilidad de que un dato pertenezca a un grupo específico, buscando los valores más parecidos a los datos aprendidos en el entrenamiento. El Aprendizaje inicia con la suposición de datos independiente entre ellos, luego se introducen en vectores y se etiquetan según los vecinos más cercanos utilizando los atributos de cada elemento a predecir (Bijalwan et al., 2014).

1.3.1.8 Redes Neuronales

Las redes neuronales, son un modelo artificial que imita el funcionamiento del cerebro humano, el cual aprende de la experiencia. Igual que el cerebro, las unidades básicas de las redes neuronales son las neuronas, las cuales forman un conjunto de unidades conectadas entre sí que transmiten señales y que entran a la red, en donde se transforman para producir unos nuevos valores de salida (Rivas & Marzón, 2018).

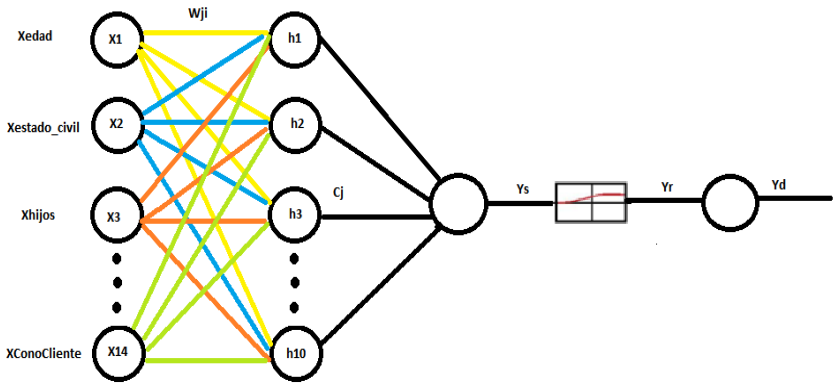

Figura 1. Ejemplo redes neuronales

Fuente: Elaboración propia

Como se ve en la figura 1 cada neurona se conecta con las demás formando enlaces. En ellos cada salida se multiplica por un peso. Aumentando o impidiendo el estado de activación de las neuronas vecinas. Además, cada neurona cuenta con una función de activación encargada de modificar el resultado. Los enlaces entre cada neurona forman la red y estos forman un sistema. Este está conformado principalmente por 3 tipos de capas: la primera capa está conformada por neuronas de entrada, la segunda es una capa que recibe información de la primera capa denominada capa oculta y finalmente la tercera es la capa de salida encargada de clasificar los datos. Para redes más complejas el número de capas puede variar en cada tipo (Rivas & Marzón, 2018).

El aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en respuesta a los datos suministrados por la capa de entrada. Estos cambios terminan cuando los valores permanecen estables:

$$\frac{dw_j}{dt} = 0$$

Ecuación 2. Estabilización de valores

Para determinar la forma en la que se van a modificar los pesos, se consideran dos tipos de reglas, aprendizaje supervisado y no supervisado. La diferencia entre ambas reglas es la existencia de un agente externo que controle el proceso. Como lo menciona Rivas & Marzón, 2018:

Aprendizaje supervisado: El aprendizaje es controlado por un agente externo el cual determina la respuesta que debería generar la red a partir de una entrada determinada, si la respuesta no coincide con la esperada se modifica los pesos de las conexiones. Posee tres categorías:

- **Aprendizaje por corrección de error:** Ajusta los pesos de la red en función de la diferencia entre los valores deseados y los obtenidos de las salidas. La fórmula para esto podría ser la siguiente:

$$e_k^2 = \frac{1}{2}(y_{dk} - y_{rk})$$

Ecuación 3. Error cuadrático medio

- **Aprendizaje por refuerzo:** Es menos veloz que el anterior y no indica una salida exacta que se desea que proporcione la red ante un dato determinado. Aquí, la función se encarga de indicar mediante una señal de refuerzo si la salida obtenida en la red se ajusta a la deseada y en función de ello se ajustan los pesos basándose en un mecanismo de probabilidad.
- **Aprendizaje estocástico:** Se hacen cambios aleatorios en los valores de los pesos y evalúan los resultados a partir del objetivo deseado y de distribuciones de probabilidad.

Aprendizaje no supervisado: No tienen un agente externo que ajuste los pesos de las conexiones. El sistema no recibe ninguna señal del entorno, el mismo sistema es capaz de ajustarse. Las redes establecen con los datos de entrada características, correlaciones, regularidades o categorías para reajustarse (Marín Diazaraque, 2007). Existen varias posibilidades para la salida:

- La salida representa el grado de familiaridad o similitud entre la información de entrada y los datos anteriores.
- Clusterización o establecimiento de categorías en donde la salida indica a qué categoría pertenece en donde la red crea las correlaciones oportunas.
- La salida es una versión codificada de los datos de entrada con menos bits.

- Se mapean las características obteniendo una disposición geométrica de los datos de entrada.

Los algoritmos de aprendizaje no supervisado suelen ser de dos tipos:

- **Aprendizaje hebbiano:** Extrae las características de los datos de entrada. Su aprendizaje consiste en el ajuste de los pesos con la correlación de los valores la capa de salida (Bishop, 2007).

$$Incr(w_{ij}) = y_i y_j$$

Ecuación 4. Correlación de los valores de salida

Si la salida es positiva se produce un refuerzo en los pesos, mientras que, si es negativa se modifican los pesos en función de los estados de las neuronas de salida.

- **Aprendizaje competitivo o cooperativo:** En este aprendizaje se busca que las neuronas compitan por su activación ya que se pretende que, al entrar ciertos datos a la red, solo una o un grupo de ellas se activen. El objetivo de este aprendizaje es categorizar los datos que se introduzcan a la red. De esta forma la información similar se clasificada en la misma categoría activando, por lo tanto, la misma neurona de salida (Marín Diazaraque, 2007).

1.3.1.9 Transferencia de aprendizaje

Unos de los mayores desafíos al trabajar con redes neuronales es el costo computacional ya que en algunos casos las redes pueden llegar a ser muy grandes al tener muchas conexiones y neuronas. Afortunadamente, existe una técnica que nos permite utilizar redes neuronales muy grandes, en las cuales se han invertido muchas horas de entrenamiento. Esta técnica se llama transferencia de aprendizaje y consta de seleccionar una red neuronal previamente entrenada en donde se reemplaza normalmente la última capa (aunque pueden ser más) por la nuestra. Logrando así entrenar nuestro propio clasificador con una arquitectura ya probada y en una fracción de su costo.

1.3.1.10 Redes neuronales convolucionales (CNN).

Se ha trabajado mucho en el desarrollo de algoritmos de detección de objetos utilizando una cámara estándar sin sensores adicionales. Los algoritmos de detección de objetos más avanzados utilizan redes neuronales profundas. Las Redes Neuronales Convolucionales son un subconjunto de algoritmos de aprendizaje profundo con impresionantes historias de éxito en el campo de la visión por computadora, como la extracción de características visuales y el reconocimiento de objetos (Shin et al., 2016). Su uso, en las mayorías de los casos, da muy buenos resultados al momento de realizar procesamiento de imágenes debido a su mayor poder de cómputo gracias al uso de las GPU; los datos de capacitación extensos para el modelado y la introducción de unidades lineales rectificadas; lo que resulta en una mayor velocidad de convergencia con alta calidad (Mansoorizadeh, Afrasiabi, & Khotanlou, 2019).

No vamos a profundizar en el funcionamiento de estas redes, pero es importante entender cómo funciona y su arquitectura para entender las redes YOLO. Como se mencionó anteriormente las redes neuronales son un paradigma de programación inspirados en la biología, que permiten a las computadoras aprender de los datos observados. Las redes neuronales tradicionales constan de una capa de entrada, capas ocultas y una capa de salida las cuales están conectadas entre sí. Tomando un vector como entrada, transformándose a medida que pasa entre las capas, hasta tomar un valor de salida el cual pertenece a las puntuaciones de las clases.

El principal problema de las redes tradicionales para la detección de objetos es que no se adaptan bien a las imágenes de gran tamaño. Ya que las capas totalmente conectas suponen un gran problema con la estabilidad de los parámetros y la mala generalización del modelo conocido como *overfiting* (Huang et al., 2019).

Para solucionar estas debilidades están las redes neuronales convolucionales dado que restringen su arquitectura para ello. Disponiendo las neuronas de las capas a tres dimensiones: anchura, altura y profundidad. Además, la imagen completa se reduce a un único vector de puntuaciones de clase. Cada capa de una CNN transforma el volumen anterior utilizando una función diferenciable (Yamashita et al., 2021).

Las redes neuronales convolucionales (CNN) son la principal arquitectura que se utiliza para la visión por ordenador. En lugar de tener capas totalmente conectadas, una CNN tiene una capa en la que un filtro se convoluciona con diferentes partes de la entrada para crear la salida. El uso de estas capas permite extraer patrones relacionales a partir de una entrada. Además, una capa de convolución tiende a tener menos pesos que necesitan ser aprendidos, que una capa totalmente conectada, ya que los filtros no necesitan un peso asignado de cada entrada a cada salida (Yamashita et al., 2021).

Los arquitectos de las redes son libres de diseñar la red de forma que deseen, pero normalmente un conjunto de operaciones/capas son comunes en todas las redes (Huang et al., 2019).

- Capa de entrada: contiene los valores brutos de la imagen, teniendo un volumen igual a las dimensiones de la imagen.
- Capa convolucional: Calcula la salida de las regiones conectadas localmente en la entrada de la capa aplicando el producto punto entre sus pesos y la región a las que están conectadas. Después de una capa convolucional, las dimensiones de altura y anchura suelen reducirse a favor de un crecimiento de la profundidad de los volúmenes de activación, que posteriormente representarán las puntuaciones de clasificación.
- Capa de unidad lineal rectificada (ReLu): Aplica una función de activación por elementos como max, en donde estas capas no alteran el tamaño del volumen como lo hacen las capas convolucionales.
- Capa de agrupación: En estas capas se realizan operaciones de muestreo descendente en dimensiones de anchura y altura alterando las activaciones del volumen.
- Capas totalmente conectadas: La capa de salida donde se calculan las puntuaciones de las clases que da lugar a un vector con una profundidad igual al número de clases. Todas las neuronas están conectadas a las neuronas de la capa anterior.

Todo lo explicado anteriormente corresponde a la tarea de clasificación, pero no de detección. Al realizar la tarea de clasificación, estamos limitados a una clase por imagen y no podemos obtener directamente la ubicación del objeto en la imagen. Para la detección de objetos

podemos aplicar varias técnicas, una de las más sencillas es aplicar una regresión, normalmente adjuntando otra capa totalmente conectada a la última capa convolucional para calcular el cuadro delimitador. Este método solo funciona con un objeto por imagen, dejándonos con lo que se conoce como un localizador de objetos. Para lograr la detección de múltiples objetos, podemos dividir la imagen de entrada en regiones, donde cada parte se trata individualmente. Sin embargo, estos detectores son lentos, ya que pueden requerir muchos pasos adicionales.

Las redes neuronales convolucionales basadas en regiones (R-CNN) consideran propuestas de regiones para la detección de objetos en imágenes. A partir de cada propuesta de región, se extrae un vector de características y se introduce en una red neuronal convolucional. Para cada clase, los vectores de características se evalúan con máquinas de vectores de apoyo (SVM). Aunque la R-CNN ofrece una gran precisión, el modelo no es capaz de alcanzar la velocidad en tiempo real ni siquiera con Fast R-CNN debido al costo en procesamiento y la ineficacia de la proposición de regiones (Huang et al., 2019).

1.3.1.11 Redes YOLO

Existe un sistema diseñado para la detección de objetos en tiempo real el cual utiliza redes neuronales convolucionales (CNN) para detectar objetos. Su nombre es YOLO (de las siglas en inglés de *You Only Look Once*, Tú sólo miras una vez) se diferencia de los demás por el hecho de que sólo necesita ver la imagen una vez sacrificando exactitud, pero ganando velocidad. Para llevar a cabo la detección primero divide la imagen en una cuadrícula de 13*13; cada división se denomina celda. En cada una de las celdas predice posibles bordes y le asigna una probabilidad. Después de obtener todas las predicciones, deja sólo los más **K-vecinos más cercanos** (Redmon et al., 2016)**.**

You Only Look Once (YOLO) fue desarrollada para crear un proceso de un solo paso que incluye la detección y clasificación. Las predicciones de la caja delimitadora y de la clase se hacen después de una evaluación de la imagen de entrada.

YOLO se diferencia de los algoritmos tradicionales en que las predicciones de la caja delimitadora y de clase ocurren simultáneamente. La primera imagen de entrada se divide en una retícula de S x S, se definen B cuadros delimitadores en cada celda con una puntuación de confianza (Huang et al., 2019). La confianza significa la probabilidad de que un objeto exista en cada cuadro delimitador y se define como:

$$C = Pr(Object) \cdot IOU$$

Ecuación 5. Confianza YOLO

Donde IOU representa la intersección como una fracción entre 0 y 1, en donde esta intersección es el área total entre el cuadro delimitador predicho y la verdad de este.

Simultáneamente, mientras se crean los cuadros delimitadores, cada celda también predice la probabilidad específica de clase para cada una y esta se calcula así:

$$Pr(Class\ i) \cdot IOU$$

Ecuación 6. Probabilidad para cada celda

YOLO usa la siguiente ecuación para calcular la pérdida:

$$\begin{aligned} Loss = \lambda_{coord} & \sum_{i=0}^{s^2} \sum_{j=0}^{A} 1_{ij}^{obj} \left[(b_{xi} - b_{\hat{x}i})^2 + \left(b_{yi} - b_{\hat{y}i}\right)^2 \right] \\ & + \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{A} 1_{ij}^{obj} \left[\left(\sqrt{b_{wi}} - \sqrt{b_{\hat{w}i}}\right)^2 + \left(\sqrt{b_{hi}} - \sqrt{b_{\hat{h}i}}\right)^2 \right] \\ & + \sum_{i=0}^{s^2} \sum_{j=0}^{A} 1_{ij}^{obj} (C_i - \hat{c}_i)^2 \\ & + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{A} 1_{ij}^{noobj} (C_i - \hat{c}_i)^2 + \sum_{i=0}^{s^2} 1_{ij}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \end{aligned}$$

Ecuación 7. Pérdida YOLO

Esta función se utiliza para corregir el centro y el borde de la caja delimitadora para cada predicción. Cada imagen es dividida en una celda S x S, con A cajas delimitadoras para cada celda. Donde b_x y b_y se refieren al centro de cada predicción, mientras que b_w y $b_h$se refieren

a las dimensiones de las cajas delimitadoras. Donde λ_{coord} y λ_{noobj} son variables usadas para incrementar el énfasis en las cajas con objetos y disminuir el énfasis en las cajas sin objetos. C se refiere a la confianza y p(c) se refiere a la clasificación de la predicción. El 1_{ij}^{obj} es el responsable de la predicción del objeto es 1 si está y 0 si no. (Huang et al., 2019).

Mientras que la pérdida es usada para calcular el rendimiento de un modelo, la exactitud de las predicciones hechas por el modelo en detección de objetos es calculada con la siguiente fórmula:

$$avgPrecision = \sum_{k=1}^{n} P(k)\Delta r(k)$$

Ecuación 8. Exactitud de las predicciones YOLO

P(k) se refiere a la precisión del umbral k, mientras que Δr(k) se refiere al cambio en la recuperación.

La arquitectura de la red YOLO se inspiró en el modelo GoogleNet para la clasificación de imágenes, contiene 24 capas convolucionales y 2 capas totalmente conectadas. En lugar de los módulos de inicio utilizados por GoogleNet, YOLO utiliza capaz de reducción 1 x 1 seguidos de capas convolucionales 3 x 3 Como se puede ver en la figura 2. Esta red ha sido mejorada con diferentes versiones como YOLOv2, YOLOv3 y YOLOv4 con la finalidad de minimizar errores de localización y mejorar las métricas (Serrano, 2017).

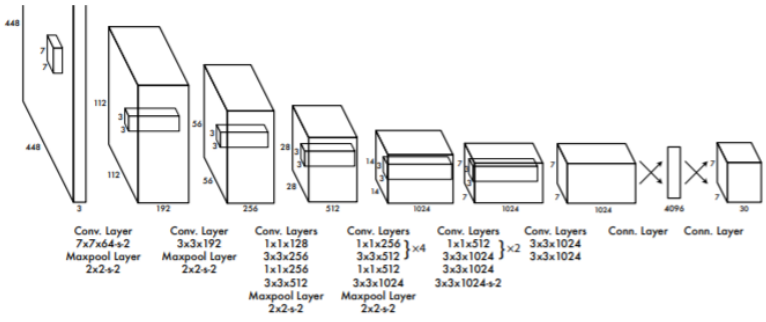

Figura 2. Arquitectura YOLO

Fuente (Serrano, 2017).

1.3.2 Antecedentes

Según García & Caranqui (2015) la visión artificial en los últimos años ha tenido un desarrollo muy rápido debido a las mejoras en capacidad de procesamiento y almacenamiento de los dispositivos electrónicos. Esto ha potenciado su aplicación en diferentes aéreas, como por ejemplo en la automatización de la agricultura donde los sistemas automatizados diseñados para el sector agrícola incrementarán la producción y brindarán informes de interés con los que serán mejores las decisiones tomadas (Bedolla, Salazar & Solano,2020). Además, la visión artificial no sólo es utilizada para mejorar la producción y ganancias de los agricultores, sino que también está siendo utilizada como tecnología para incrementar la seguridad alimentaria.

La seguridad alimentaria se entiende como la capacidad que tiene un gobierno de garantizar el acceso a alimentos nutritivos para toda la población. Es un tema en el cual países asiáticos y europeos están avanzando con ayuda de la visión artificial, mientras que en América Latina la aplicación de esta tecnología es poco frecuente, exceptuando a Brasil que está promoviendo centros de investigación para mejorar la calidad de exportaciones de su país. Debido a esto, se debe incentivar el desarrollo de esta tecnología para aumentar la seguridad alimentaria de los países latinoamericanos (Negrete,2018).

Es bien sabido que el diagnóstico temprano de cualquier enfermedad genera una ventaja estratégica sobre dicho padecimiento, en la agricultura sucede lo mismo. La detección temprana de la plaga o enfermedad incrementa las posibilidades de éxito en el tratamiento.Es por ello que este problema ha sido el centro de atención de muchos estudios, incluso existen múltiples empresas especializadas en ello. A continuación, se presentarán algunas de las investigaciones más significativas en este ámbito.

Suhartono et al (2013) fueron unos de los primeros en aplicar un sistema de lógica difusa y arboles de decisión para la detección de enfermedades del café. A pesar de que las cámaras más comunes no pasaban los 8 megapíxeles para el año 2013, lograron tener un accuracy del 85%. La investigación logró clasificar imágenes de enfermedades como hongos de

podredumbre blanca, nematodos, entre otros. Con ayuda del experto lograron sacar las reglas del árbol de decisión basándose en las características visibles de la planta.

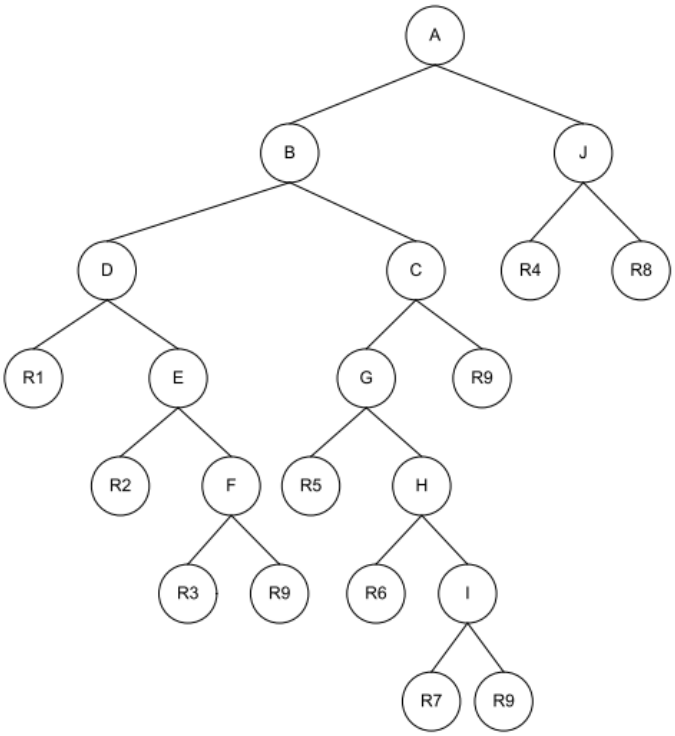

Figura 3. Árbol de decisión

Fuente: Tomado de (Suhartono et al., 2013)

Los nodos terminales identificados con las siglas R representan la clasificación de enfermedad, los demás representan las características determinadas por el experto para ayudar a clasificar la enfermedad. Por ejemplo, la A son las hojas que se tornan amarillas.

Otra investigación relacionada con el café es la de Mengistu et al (2016) de la cual decidieron evaluar tres modelos, los seleccionados fueron ANN, KNN y un híbrido de mapas auto organizables (SOM) con una función de base radial, todo con el fin de determinar el algoritmo que mejor clasificará las enfermedades de la roya y marchitamiento del café. La cual afectan a los cultivos de Etiopía y perjudica su economía. Los resultados fueron los siguientes: 58% para KNN, 80% ANN y para el híbrido 90% de *accuracy*. En donde se utilizaron 9100 imágenes y se separaron en un 70% para entrenamiento y 30% para test.

Johannes et al (2017) durante tres años utilizando siete dispositivos móviles lograron obtener alrededor de 3500 imágenes para implementar un modelo capaz de detectar las 3 principales plagas del trigo. A estas se les aplicaron técnicas para preprocesar y segmentar la imagen. Para su preprocesamiento se normalizó la constancia del color y se pasó de RGB a LAB, de esta manera, para la segmentación se aplicó ventaneo manual y SLIC. Gracias a ello el modelo logró obtener un *accuracy* del 80%.

Islam et al (2017) lograron aplicar un modelo SVM para detectar tizón en los cultivos de papa, utilizando apenas 300 fotos lograron obtener estadísticas muy buenas, como un *accuracy* del 93% y una precisión del 95%. Ellos atribuyen sus resultados a su segmentación en la cual aplicaron el método del valor del umbral. Sin embargo, otros estudios como el de Chouhan et al (2018) al usar SVM para la misma enfermedad obtuvieron mejores resultados con KM-Clustering. Por otra parte, Saady et al (2016) aplicaron dos SVM en serie para la misma enfermedad del tizón y el moho, usando KM-Clustering para la segmentación y un filtro de media para el preprocesamiento lograron obtener una exactitud del 88%. Por lo que hablar de un modelo específico para ciertas categorías de enfermedades resulta incorrecto: el mismo problema puede tener diferentes aproximaciones y resultados.

Singh & Misra (2017), buscaron el modelo correcto para clasificar las quemaduras presentadas en las hojas del frijol, plátano, rosas y limón. Propusieron segmentar las hojas con algoritmos genéticos y realizar la extracción de características por el método de concurrencia de color. Logrando así un 96% de exactitud utilizando el modelo SVM como promedio en todos los cultivos de estudio.

Li et al (2020) en su investigación probaron distintas redes neuronales muy famosas en la comunidad con el fin de clasificar plagas en distintos cultivos. Utilizaron validación cruzada para seleccionar la mejor red neuronal, las redes seleccionadas fueron VGG-16, VGG-19, ResNet50, ResNet152 y GoogLeNet. A la técnica de utilizar estas redes para otros problemas de las cuales fueron diseñadas se les conoce como Transferencia de aprendizaje.

Li et al (2020) lograron juntar 5629 imágenes de 10 clases de plagas de internet, estas imágenes fueron preprocesadas con varios métodos como: conversión a escala de grises, valor del umbral y *watersheld* para así, obtener imágenes casi listas para el entrenamiento.

Antes de entrar al modelo, dichas imágenes fueron aumentadas gracias a las técnicas de rotación, reflejado, adición de ruido y aumento para lograr que los modelos aprendieran mejor. Finalmente, el mejor desempeño lo obtuvo la red GoogLeNet con un *accuracy* del 96%.

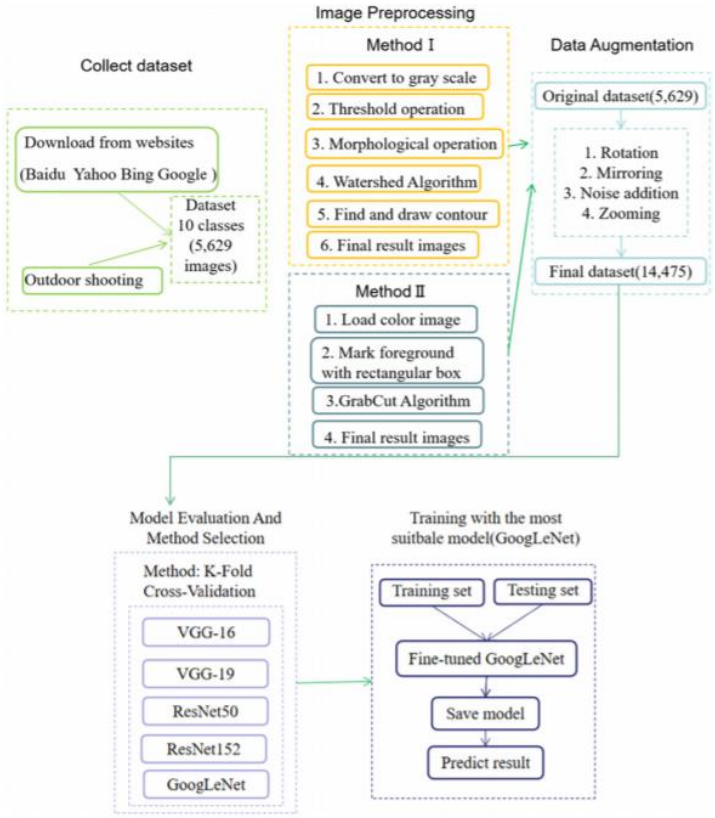

Figura 4. Proceso transferencia de aprendizaje utilizado en el artículo.

Fuente: (Li et al. ,2020)

Por último tenemos el estudio de Morbekar et al (2020) en donde se usó la red YOLO para detectar enfermedades de los principales 26 cultivos ubicados en La India, en donde lograron clasificar 14 de las enfermedades más importantes, y obtener del entrenamiento un *accuracy* del 99%. Sin embargo, con imágenes provenientes de internet su modelo sólo logró clasificar con un *accuracy* del 50%. Se atribuyen los malos resultados a la variedad en los tipos de imágenes y las diferentes distancias.

2 METODOLOGÍA

Para el desarrollo de esta investigación se tomó como referente la metodología expuesta por Roldán et al (2019) mencionada anteriormente, la cual cuenta con 5 pasos.

- **Adquisición de imágenes:** Es el paso más importante de la investigación y el de más cuidado, ya que los datos ingresados al modelo deben de evitar generar ruido dentro del él. En esta etapa se generan las fotos o se obtienen de algún repositorio, lo ideal es que sea en ambientes controlados.
- **Preprocesamiento:** Es el conjunto de técnicas que se aplican a las imágenes digitales con el fin de evitar ruido dentro del modelo o resaltar características, estás técnicas facilitan la búsqueda de información para el modelo. En otras palabras, a partir de una imagen se genera otra que contenga ciertas características más significativas para el modelo. Los principales objetivos que se busca con esta fase son: Suavizar la imagen, eliminar ruido, realzar bordes y detectar bordes.
- **Segmentación:** Es el proceso de dividir una imagen en varias partes, a cada píxel se le asigna una categoría, estas varían según la segmentación elegida (binaria, asistida, semántica, instanciada etc.). El objetivo de esta etapa es ubicar regiones que tengan significado para el modelo
- **Extracción de características:** Se trata de una fase en donde un conjunto de datos sin procesar se reduce a características más manejables para el ser humano o significativas para el modelo. Lo cual facilita el aprendizaje y generalización de los modelos, ya que reduce los recursos utilizados para procesar la información.
- **Clasificación:** Es el paso en donde se predice la etiqueta de las imágenes con ayuda del modelo elegido, este modelo será el que obtenga los mejores resultados en la validación cruzada.

Las etapas mencionadas anteriormente, se tomarán como guía para la elaboración de este trabajo.

2.1 Recolección de Imágenes

En este punto inicial, se tomaron varias imágenes en el cultivo de Inversiones Agrícolas las Acacias S.A.S, el proceso se llevó a cabo en dos visitas, en las cuáles se recorrió todo el cultivo en busca de plantas con daños en su follaje. En la primera salida se dificultó encontrar imágenes puesto que la plaga estaba controlada, debido a esto se decidió tomar varias imágenes de una misma planta en diferentes ángulos y distancias, pero en la segunda la situación fue distinta. En esta salida hubo un brote por lo que solo se realizó una foto por planta, estás fueron tomadas a 15 cm de la planta en donde se ignoró la cantidad de luz puesto que el modelo fue pensado para ser utilizado en esas circunstancias.

De la salida 1 se obtuvieron 176 imágenes de plantas con presencia de huevos y 291 con daño hecho por las larvas del minador para un total de 467 imágenes, estas fueron separadas en 2 subconjuntos. Para el entrenamiento se utilizaron 420 imágenes, 158 imágenes con huevos y 262 con daño de larva, la prueba utilizó 18 con huevo y 29 para daño de larva.

De la salida 2 se tomaron 402 fotos, 110 de huevos y 263 con daño de larva. Igual que en la salida número uno las imágenes se dividieron en subconjuntos. Para el entrenamiento se separaron 99 imágenes de huevos y 263 de daño de larva, para la prueba se usaron 11 imágenes de huevos y 29 de daño de larva en cada categoría.

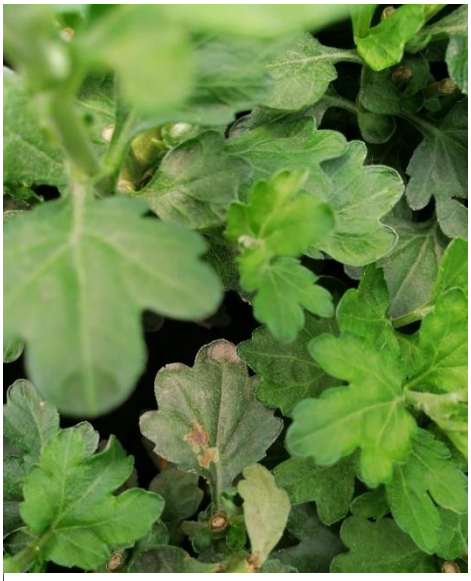

Figura 5. Daño larva

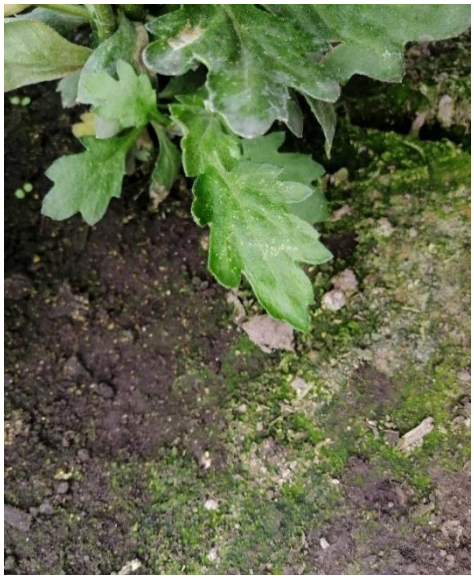

Figura 6. Presencia de Huevos

Para un total de 869 imágenes, 33% de imágenes con huevo y 67% con presencia de daño larva.

2.2 Selección del área afectada

La red YOLO necesita un archivo personalizado que le indique el cuadro que limita al objeto para ir validando su entrenamiento y un archivo que le indique las clases a predecir. El contenido del archivo de las clases para nuestro modelo contiene los nombres de las clases ordenadas alfabéticamente, en nuestro caso en la línea uno huevo y en la línea dos larvas. El contenido del archivo con las ubicaciones sigue la siguiente estructura:

1 0.528863 0.592014 0.059462 0.076389

El primer número indica a que clase pertenece, en nuestro caso pertenece a la clase larva. Los demás datos están normalizados y estos representan los puntos del rectángulo que encierra el daño.

Realizar este proceso manualmente por cada archivo resulta casi imposible, por lo que se usó una herramienta llamada *labelImg* que nos facilita el proceso. Esta se encarga de generar todos los archivos necesarios para el entrenamiento fácilmente.

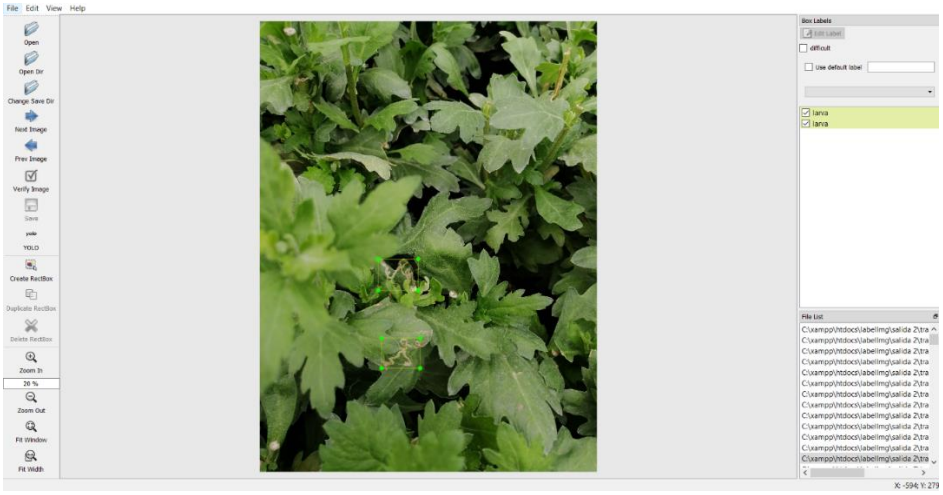

Figura 7. Programa LabelImg

2.3 Aumento de datos

Luego de separar las imágenes en sus respectivos grupos, se evidenció la poca cantidad de imágenes que disponíamos en ese momento, por lo que se procedió a aumentar el número de datos. Se uso la técnica de ventaneo en donde logramos obtener 20931 imágenes para el entrenamiento y 3226 para la prueba. No se aumentó el grupo de validación ya que se usará con el fin de validar el funcionamiento del modelo.

2.4 Extracción de características

Para la extracción de características se seleccionaron los modelos más usados en la comunidad: VGG-16, VGG-19 y Inception V3. La extracción sólo fue utilizada para la red neuronal y el KNN.

2.5 Implementación del modelo.

Para la implementación del modelo se seleccionaron los siguientes: Red neuronal, Máquina de vectores de soporte, KNN, ResNet50, MobileNet, MobileNet V2, Xception y red YOLO. Para KNN y la red neuronal se usó la librería SKlearn, para las redes profundas se usó keras y para la red YOLO se utilizó su entorno optimizado llamado Darknet. Sus configuraciones fueron las siguientes:

- **Red neuronal:** Una red neuronal con 100 neuronas, función de activación lineal rectificada (relu), optimizador *lbfgs*, con una tasa de aprendizaje 0.001 y los demás valores por defecto.
- **KNN:** El número de vecinos a revisar va desde 3 hasta el 10.
- **Redes profundas:** Se les quitó la última y se agregaron dos capas más, la antepenúltima capa se le colocaron 128 neuronas con una función de activación lineal rectificada con una función de activación exponencial normalizada (softmax)
  Solo las últimas dos capas se configuraron para ser entrenadas. Las demás no debían de hacerlo.

La configuración de la red YOLO es un proceso más largo, por lo que se procederá a explicar en los siguientes incisos.

2.5.1 Configuración entorno DarkNet

Luego de generar los archivos en formato YOLO por imagen, se procede a configurar el entorno DarkNet, este es un framework de código abierto que permite utilizar la GPU, el cual esta optimizado para las redes YOLO. La configuración de la cual se guio esta investigación es la presentada por la documentación oficial expuesta en la plataforma GitHub del usuario AlexeyAB (2016).

Su configuración comenzó descargando los pesos pre-entrenados de la red para así aprovechar las horas de entrenamiento invertidas. Sin embargo, se obtuvieron mejores resultados sin estos pesos, por lo que más tarde se eliminaron. Posteriormente se modificó el archivo de configuración de la red llamado yolov4.cfg ubicado en la carpeta de configuración. En este archivo se especifican varios parámetros en donde el valor en paréntesis es el asignado a nuestro trabajo:

Parámetros de red:

- **Batch:** es la cantidad de imágenes en las que se procesa un lote, sirve para crear un gradiente y actualizar los pesos en hacia atrás (64).
- **Subdivisions:** Número de divisiones en un lote (24).
- **Width y Height:** Tamaño de la red, por lo que cada imagen se redimensionará al tamaño de la red durante el entrenamiento y la detección (512).
- **Channels:** Tamaño de la red en canales, en donde cada imagen se convertirá a este número de canales en el entrenamiento y la detección (3).

Parámetros de optimizador:

- **Momentum:** Acumulación del movimiento, cuanto afecta el historial al cambio posterior (0.949).

- **Decay:** Una actualización más débil de los pesos de las características típicas, lo que elimina el desbalance en los datos (0.0005).
- **Learning_Rate:** tasa de aprendizaje inicial en el entrenamiento (0.001).
- **Burn_in:** Grabación inicial (1000).
- **Max_batches:** El entrenamiento será procesado con este número de iteracciones (500200).
- **Policy:** Política para cambiar la tasa de aprendizaje (steps).
- **Steps:** si la política fue configurada con steps, son los números de iteraciones que el entrenamiento usará para multiplicar por el factor de escala (1280,3600).
- **Scales:** Si la política fue configurada como steps, es el número por el cual se multiplique después de alcanzar el step (0.1, 0.1).

Parámetros para aumentar los datos

- **Angle:** rotación al azar de imágenes durante el entrenamiento (0).
- **Saturation:** cambios en la saturación al azar durante el entrenamiento (1.5).
- **Exposure:** cambios en la exposición al azar durante el entrenamiento (1.5).
- **Hue:** cambios en el matiz del color al azar durante el entrenamiento (0.1).

Parámetros de cada capa interna:

Estos parámetros varían dependiendo de cada capa.

- **Batch_normalize:** Normalización de los lotes.
- **Activation:** función de activación.
- **Filters:** Cuantos núcleos convolucionales hay en una capa.
- **Size:** Tamaño de la capa.
- **Stride:** Paso de compensación.

Parámetros para la última capa:

- **Classes:** Número de clases a predecir (2).
- **Num:** Número de anclas (9).

- **Jitter:** cambios en el tamaño de cada imagen al azar (0.3).
- **Random:** cambia aleatoriamente el tamaño de la red después de cada 10 lotes manteniendo la relación de aspecto inicial (1).

Luego de la configuración se hizo un *script* capaz de calificar el grupo de imágenes. Ya que *DarkNet* nativamente no provee está función.

2.7. Evaluación o prueba del modelo.

Para la validación de los modelos se utilizó la validación cruzada, la cual es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son totalmente independientes las pruebas del entrenamiento. Posteriormente se calcularon los parámetros de *accuracy, recall, precisión, F1 y AUC* mediante sus respectivas ecuaciones:

$$precision = \frac{Verdaderos\ positivos}{Verdaderos\ positivos + falsos\ positivos}$$

Ecuación 9. Métrica precisión

$$Recall = \frac{Verdaderos\ positivos}{Verdaderos\ positivos + falsos\ negativos}$$

Ecuación 10. Métrica recall

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Ecuación 11. Métrica F1

$$accuracy = \frac{Verdaderos\ p + verdaderos\ n}{Verdaderos\ p + veradaderos\ n + falsos\ p + falsos\ n}$$

Ecuación 12. Métrica Accuracy

En nuestro caso la validación cruzada se dividió en dos grupos, entrenamiento y prueba. Siguiendo el método 90/10, 90% para entrenamiento y 10% para la prueba, se utilizó esto ya que hay muy pocos datos por lo que se priorizó su entrenamiento.

3 PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS

3.1 Análisis de resultados Red Neuronal, KNN y SVM.

A continuación, se presentan los resultados de los modelos implementados. Como se mencionó en la metodología, los modelos seleccionados fueron: Red neuronal, Máquina de vectores de soporte, KNN, ResNet50, MobileNet, MobileNet V2, Xception3 y red YOLO.

En principio se utilizó Orange Data Mining (Janez et al., 2013) para observar cuáles modelos serían los más viables a implementar, se observó que la red neuronal, SVM y KNN fueron los mejores candidatos ya que presentaron métricas más altas que los demás. Por lo que se procedió a utilizar la red VGG-16 como extractor de características, ya que en otras investigaciones esta dio resultados positivos.

Luego de extraer las características, se procedió a aumentar los datos con la técnica de ventaneo y se implementaron en la plataforma Google Colab, ya que esta nos provee de recursos potentes para entrenar los modelos. Los resultados obtenidos fueron los siguientes:

Modelo	**Accuracy**	**Recall**	**Precision**	**F1**	**AUC**
Red neuronal	0.601	0.601	0.681	0.603	0.812
SVM RBF	0.524	0.524	0.65	0.554	0.806
SVM lineal	0.472	0.472	0.774	0.5	0.841
KNN	0.59	0.512	0.709	0.608	0.821

Tabla 2. Resultados primeros modelos

Los resultados obtenidos no fueron los esperados, las métricas mostraban que estos modelos no cumplieron las expectativas y no lograban predecir las imágenes correctamente. Sus predicciones indicaban alta dispersión de los datos además la métrica *recall* en cada modelo fue la más baja de todas, lo que indica que los clasificadores no lograban discriminar entre casos positivos y negativos correctamente. Esto se puede observar en la figura 8 en donde podemos observar que estos modelos eran débiles contra el ruido, por lo que constantemente se equivocaban con las predicciones.

Para entender mejor la figura 8, cada imagen fue dividida en cuadros de 512*512 megapíxeles y cada sub-imagen fue ingresada a los modelos para validar sus resultados. Si estas tenían una probabilidad mayor al 50% de ser clasificadas como huevo se pintaba el cuadro verde o como larva se pintaba el cuadro rojo.

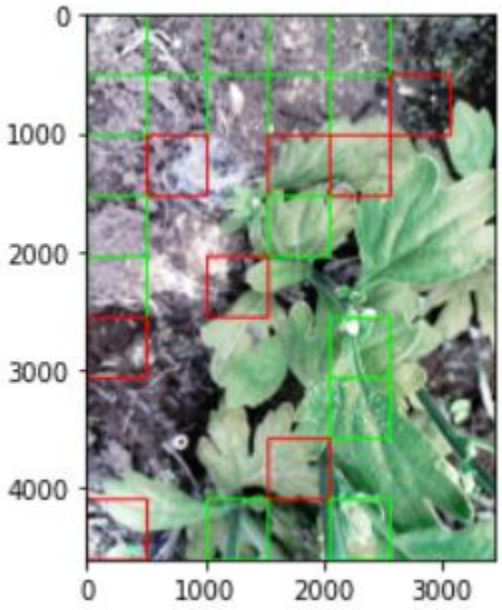

Figura 8. Predicción Red Neuronal

Como se puede ver en los anexos los resultados en ningún modelo fueron aceptables, los resultados de las pruebas fueron menores al 50%, por lo que se procedió a implementar las redes profundas, las cuales en otras investigaciones dieron resultados positivos.

3.2 Análisis de resultados CNNs

Para estos modelos se decidió implementar una red por clasificador, en otras palabras, un modelo para huevo y otro para la larva. Así se tendría más control sobre cada modelo y ver sus posibles falencias. Como se mencionó anteriormente los modelos seleccionados fueron ResNet50, MobileNet, MobileNet V2 y Xception3. Estos modelos necesitaban las imágenes de entrada redimensionadas, por lo que para ResNet50, MobileNet y MobileNet V2 las imágenes fueron redimensionadas a 224*224 y para Xception3 se redimensionaron a 299*299.

Como se mencionó en la metodología a estas redes se les retiró la última capa y se le agregaron dos capas más. Una con 128 neuronas y función de activación RELU, la otra capa de salida con dos neuronas y función de activación *softmax*.

Después de sus entrenamientos los resultados no fueron los adecuados, a pesar de que presentaban métricas excelentes la validación no fue la esperada. Las predicciones con imágenes eran muy similares a los anteriores modelos. Estas se veían afectadas por el ruido y no hubo mejora alguna en contraste a los anteriores modelos.

Por ejemplo, como se puede ver en la figura 9 y 10, los modelos que implementaban ResNet presentaban un accuracy de entrenamiento muy bueno y creciente, pero sus accuracy de prueba por épocas era muy cambiante lo que indica que sus resultados no serían fiables y no estaban aprendiendo correctamente. En cada época variaban, aunque sus resultados intentaban estabilizarse, al final no lo lograron.

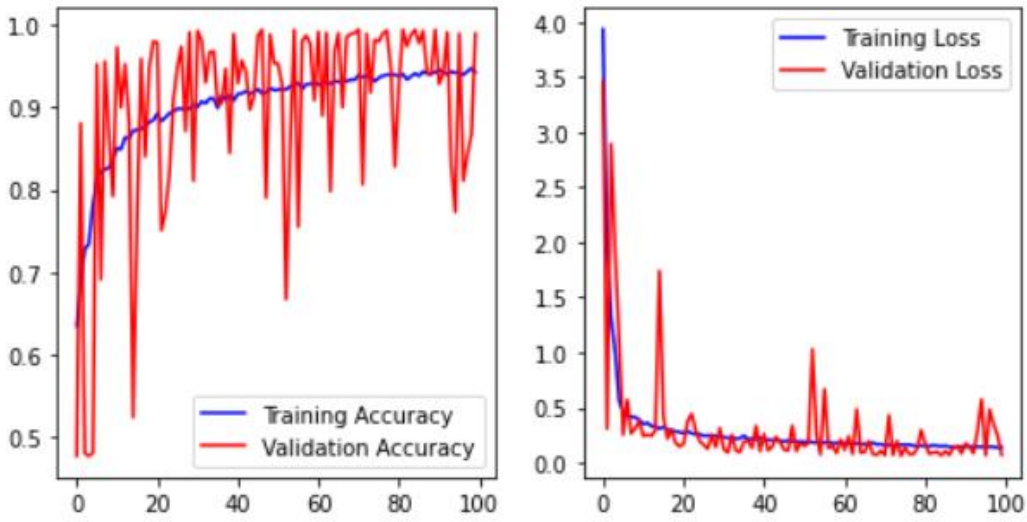

Figura 9. Resultados modelo huevo ResNet

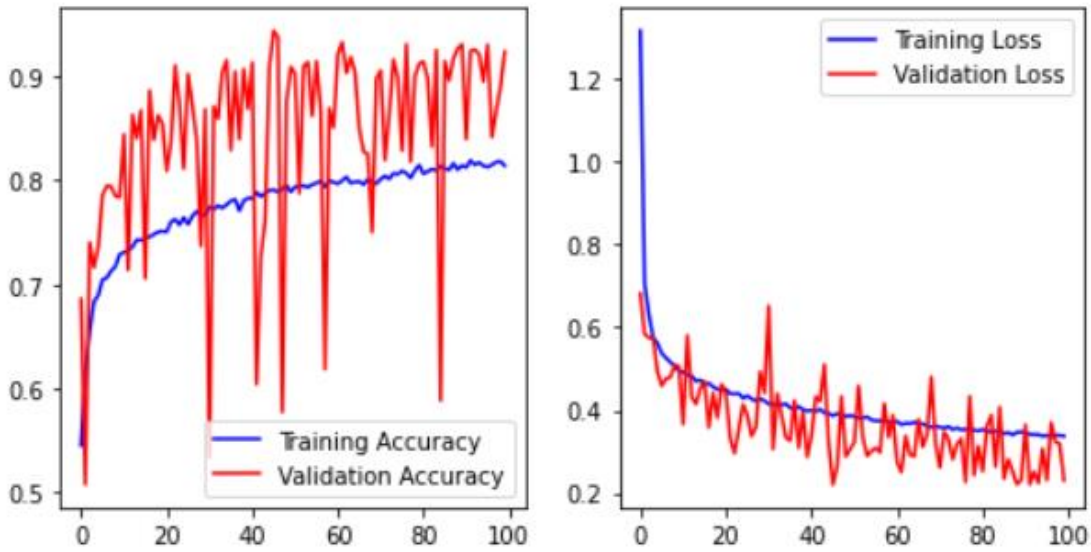

Figura 10. Resultados modelo larva ResNet

Los modelos que utilizaron ResNet al final terminaron con estas métricas:

- Modelo huevo última época:
  - Training Accuracy: 0.9425
  - Training Loss: 0.1411
  - Validation Accuracy: 0.9894
  - Validation Loss: 0.0699
- Modelo larva última época:
  - Training Accuracy: 0.8138
  - Training Loss: 0.3375
  - Validation Accuracy: 0.9232
  - Validation Loss: 0.2301

Podemos ver en la figura 11 ejemplos de lo anterior mencionado, sus predicciones no fueron aceptables y se veían afectadas por el ruido.

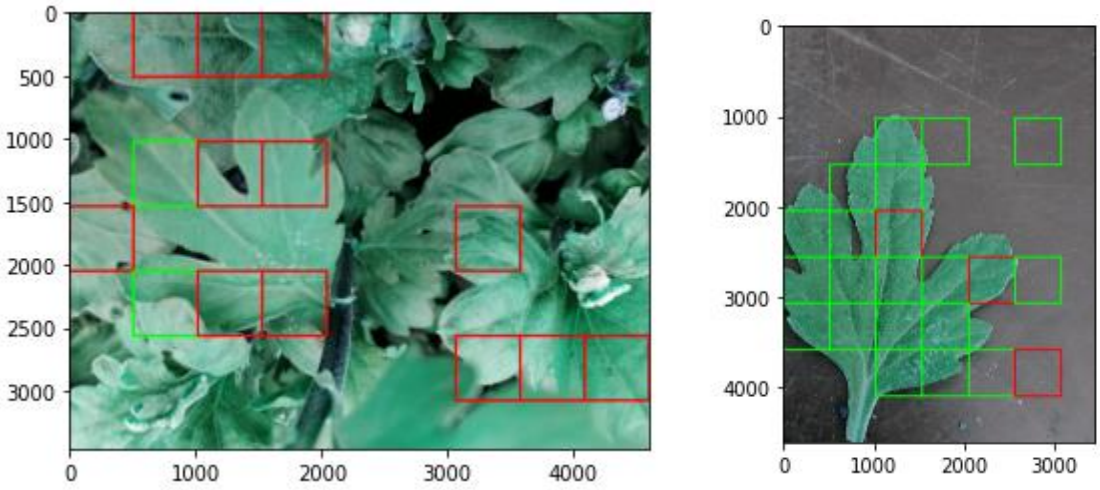

Figura 11. Clasificación ResNet

ResNet presentaba métricas muy cambiantes en sus pruebas, pero a pesar de que las demás redes profundas no presentaban este cambio en sus pruebas tampoco dieron resultados positivos. Miremos MobileNet para validar esto, como se observa en las figuras 12 y 13, estos modelos que implementaban MobileNet no presentaban variaciones en sus métricas al contrario eran métricas que presentaban crecimiento y eran muy estables.

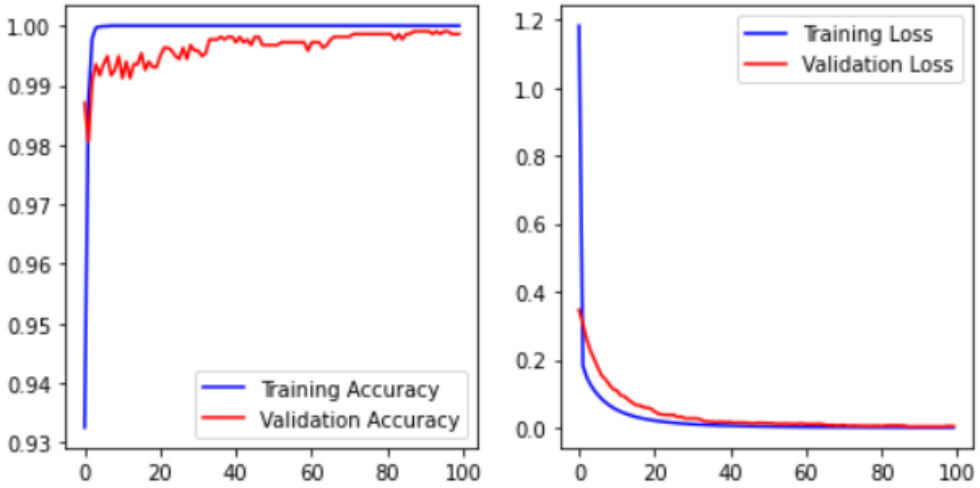

Figura 12. Resultados modelo huevo MobileNet

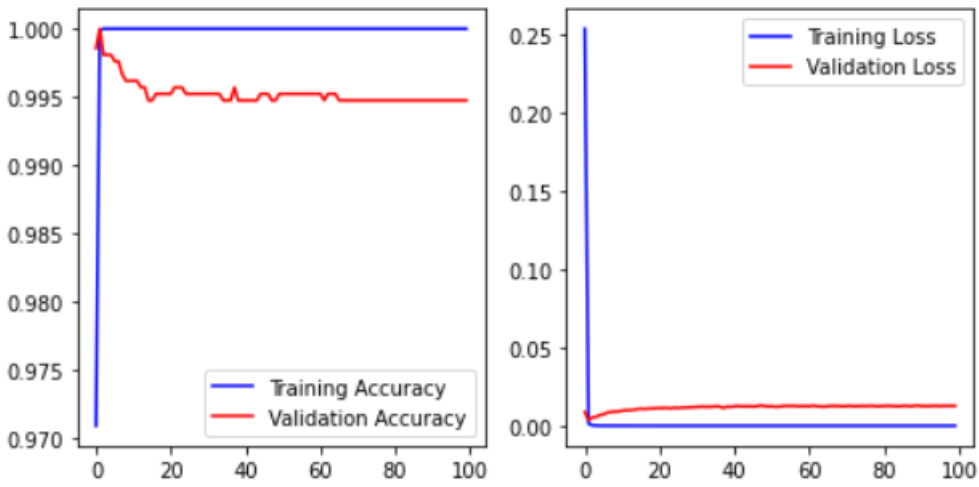

Figura 13. Resultados modelo larva MobileNet

Métricas MobileNet:

- Modelo huevo última época:
  - Training Accuracy: 1
  - Training Loss: 2.42e-4
  - Validation Accuracy: 0.9986
  - Validation Loss: 0.0041
- Modelo larva última época:
  - Training Accuracy: 1
  - Training Loss: 9.25e-9
  - Validation Accuracy: 0.9948
  - Validation Loss: 0.0127

Sin embargo, como se puede observar en la figura 14 a la hora de hacer sus predicciones con imágenes nuevas, los modelos se comportaban igual que los anteriores.

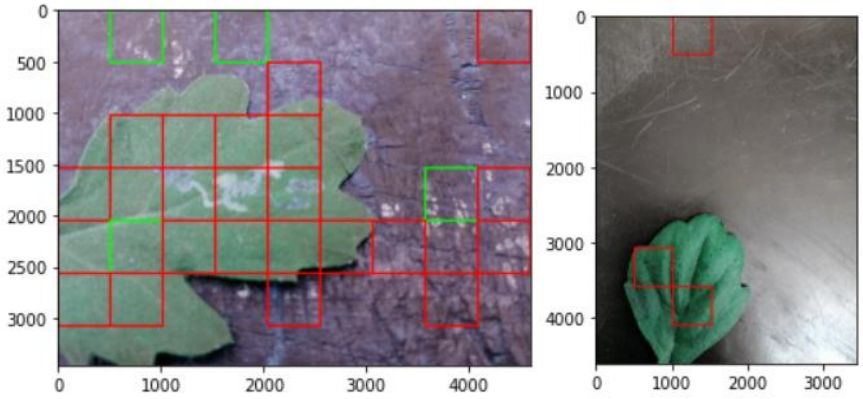

Figura 14. Predicciones MobileNet

En resumen, las métricas lanzadas por las redes profundas fueron muy buenas. Sin embargo, los resultados no fueron los esperados. Los modelos no eran capaces de clasificar las imágenes correctamente, confundiéndose bastante con el ruido de estas. Nuestro análisis de lo ocurrido es que los datos ingresados no eran los mejores para los modelos, posiblemente el ventaneo manual hecho por nosotros no fue el correcto, ya que eran más de 30.000 imágenes lo que dificultaba su revisión. Además las imágenes tenían pequeños cambios entre ellas, lo que posiblemente limitaba el aprendizaje de ellos.

Por lo que se decidió implementar la red YOLO, ya que esta se encarga de manejar el aumento de datos autónomamente y no dependía de nosotros para ello. Además, utiliza una gran cantidad de métodos para el aumento de datos.

3.3 Análisis de resultados red YOLO

La red YOLO obtuvo métricas muy positivas, como se puede ver en las tablas 3, 4 y 5. El modelo presento métricas por encima del 80%, tenía poca dispersión con los datos y muy buen recall. Por lo que posiblemente el ruido no le afectará tanto como a los modelos anteriores.

YOLO	Larva	Huevo
Precisión	0.96202532	0.825
Recall	0.97058824	0.87356322
F1	0.91566265	0.89189189
Accuracy	0.84946237	0.80487805

Tabla 3. Métricas YOLO

Matriz de confusión larva

MC Larva	Positive	Negative
True	76	3
False	3	11

Tabla 4. Matriz de confusión larva

Matriz de confusión huevo

MC huevo	Positive	Negative
True	33	0
false	7	1

Tabla 5. Matriz de confusión huevo

Como se puede ver en las figuras 15 y 16, las predicciones fueron más que aceptables. La red YOLO no era afectada por el ruido, era muy precisa y delimitaba muy bien la enfermedad. Las probabilidades asignadas a las imágenes de pruebas eran muy altas en el caso de la larva, para los huevos no se comparaban con los resultados de la larva, pero seguían siendo muy altos. Los huevos presentan más dificultad debido a su tamaño y a su complejidad. Sin embargo, los resultados son muy buenos. Por lo que la red YOLO puede ser útil para determinar la presencia de minador en las plantas, comportándose muy bien en imágenes con ruido.

Figura 15. Predicción larva

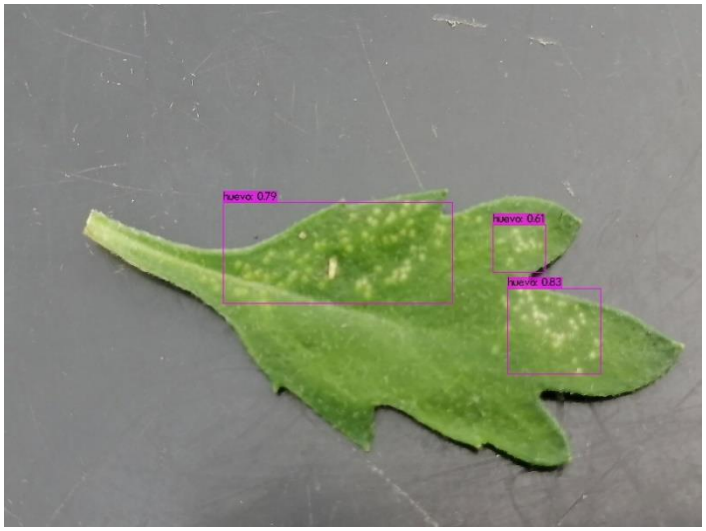

Figura 16. Predicción huevo

4 CONCLUSIONES Y CONSIDERACIONES FINALES

La investigación arrojó resultados importantes para determinar la presencia de la plaga minador en las flores crisantemos. En donde, se pudo identificar y delimitar las zonas de la plaga, permitiendo visualizar mejor las ubicaciones de las zonas infectadas para los agricultores.

Para predecir el minador se eligió la red YOLO por sus buenas métricas y predicciones acertadas, además usando el programa *labelimg se* facilita el uso de esta red para problemas de visión artificial en la agricultura ya que este facilita al ser humano la ubicación de la plaga.

Aunque la red YOLO dio buenos resultados con pocas imágenes, se recomienda en investigaciones futuras aumentar su número para obtener resultados más fuertes con cualquier método de inteligencia artificial ya que este fue nuestro principal problema. Se usó el método de ventaneo manua para aumentar los datos, pero no dio buenos resultados debido principalmente al factor humano.

Con estos resultados se abre la posibilidad a la investigación de otras plagas utilizando métodos de inteligencia artificial en especial usando la red YOLO debido a su buen manejo autónomo del aumento de datos, su facilidad de implementación y su visualización de las zonas infectadas.

REFERENCIAS

Alexey Bochkovskiy. (2016). *Documentación oficial Darknet*. https://github.com/AlexeyAB/darknet

Bedolla, J., Salazar, E., & Solano, S. (2020). *Reconocimiento automático de patrones y características de las imágenes de los cultivos como alternativa para el desarrollo agrícola*. *19*(33), 90–105.

Bijalwan, V., Kumar, V., Kumari, P., & Pascual, J. (2014). KNN based machine learning approach for text and document mining. *International Journal of Database Theory and Application*, *7*(1), 61–70. https://doi.org/10.14257/ijdta.2014.7.1.06

Bishop, C. (2007). Neural Networks for Pattern Recognition. In *Oxford*. Oxford University Press.

Chouhan, S. S., Kaul, A., Singh, U. P., & Jain, S. (2018). Bacterial foraging optimization based radial basis function neural network (BRBFNN) for identification and classification of plant leaf diseases: An automatic approach towards plant pathology. *IEEE Access*, *6*, 8852–8863. https://doi.org/10.1109/ACCESS.2018.2800685

Constantino, L., Flórez, J., Benavides, P., & Bacca, T. (2011). Minador de las hojas del cafeto : Una plaga potencial por efectos del cambio climático. *Cenicafé*, *409*, 1–12. https://www.cenicafe.org/es/index.php/nuestras_publicaciones/avances_tecnicos/avance_tecnico_0409

Cui, L., Li, G., Zhu, Z., Lin, Q., Wen, Z., Lu, N., Wong, K. C., & Chen, J. (2017). A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization. *Information Sciences*, *414*, 53–67. https://doi.org/10.1016/j.ins.2017.05.044

Díaz, H. (2017). Tecnologías de la información y comunicación y crecimiento económico. *Economía Informa*, *405*, 30–45. https://doi.org/10.1016/j.ecin.2017.07.002

Díaz Pongutá, B., Lans Ceballos, E., & Barrera Violeth, J. L. (2012). Residuos de insecticidas organoclorados presentes en leche cruda comercialziada en el

departamento de Córdoba, Colombia. *Acta Agronómica*, *61*(1), 10–15.

Frank, E., & Bouckaert, R. (2006). Naive bayes for text classification with unbalanced classes. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/11871637_49

García, I., & Caranqui, V. (2015). La visión artificial y los campos de aplicación. *Tierra Infinita*, *1*, 94–103. https://doi.org/2631-2921

Gonzalez, M., Baglio, C., Pivano, M. V., Pisi, G., & D'Agostino, L. (2018). Plagas en cultivos de flores y ornamentales de Mendoza. In *INTA Ediciones* (Vol. 5, Issue 1). https://ejournal.poltektegal.ac.id/index.php/siklus/article/view/298%0Ahttp://repositorio.unan.edu.ni/2986/1/5624.pdf%0Ahttp://dx.doi.org/10.1016/j.jana.2015.10.005%0Ahttp://www.biomedcentral.com/1471-2458/12/58%0Ahttp://ovidsp.ovid.com/ovidweb.cgi?T=JS&P

Huang, R., Pedoeem, J., & Chen, C. (2019). YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers. *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, 2503–2510. https://doi.org/10.1109/BigData.2018.8621865

Islam, M., Dinh, A., Wahid, K., & Bhowmik, P. (2017). Detection of potato diseases using image segmentation and multiclass support vector machine. *Canadian Conference on Electrical and Computer Engineering*, 8–11. https://doi.org/10.1109/CCECE.2017.7946594

Janez, D., Tomaz, C., Ales, E., Crt, G., Tomaz, H., Mitar, M., Martin, M., Matija, P., Marko, T., Anze, S., Miha, S., Lan, U., Lan, Z., Jure, Z., Marinka, Z., & Blaz, Z. (2013). Orange: Data Mining Toolbox in Python. *Journal of Machine Learning Research*, 2349–2353. https://jmlr.org/papers/volume14/demsar13a/demsar13a.pdf

Johannes, A., Picon, A., Alvarez, A., Echazarra, J., Rodriguez, S., Navajas, A. D., & Ortiz, A. (2017). Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case. *Computers and Electronics in Agriculture*, *138*, 200–209. https://doi.org/10.1016/j.compag.2017.04.013

Li, Y., Wang, H., Dang, L. M., Sadeghi-Niaraki, A., & Moon, H. (2020). Crop pest recognition in natural scenes using convolutional neural networks. *Computers and Electronics in Agriculture*, *169*(January), 105174. https://doi.org/10.1016/j.compag.2019.105174

Lim, C. H., Vats, E., & Chan, C. S. (2015). Fuzzy human motion analysis: A review. *Pattern Recognition*, *48*(5), 1773–1796. https://doi.org/10.1016/j.patcog.2014.11.016

Londoño, D. (2018). *Impacto de la automatización en la empleabilidad en paises en vía de desarrollo*. https://repository.unimilitar.edu.co/bitstream/handle/10654/32102/LONDOÑO QUINTANA DANIEL FERNANDO2018.PDF.pdf?sequence=1&isAllowed=y

Marín Diazaraque, J. M. (2007). Introducción a las redes neuronales aplicadas. *Manual Data Mining*, 1–31. halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf

Mengistu, A. D., Alemayehu, D. M., & Mengistu, S. G. (2016). Ethiopian Coffee Plant Diseases Recognition Based on Imaging and Machine Learning Techniques. *International Journal of Database Theory and Application*, *9*(4), 79–88. https://doi.org/10.14257/ijdta.2016.9.4.07

Minagricultura. (2019). *Cadena de Flores*. https://sioc.minagricultura.gov.co/Flores/Documentos/2019-06-30 Cifras Sectoriales.pdf

Minesterio de agricultura y desarrollo rural. (2016). *Incentivos e inversión en la ciencia y la tecnología para el sector agropecuario*. https://www.agronet.gov.co/Noticias/Paginas/Incentivos-e-inversión-en-la-ciencia-y-la-tecnología-para-el-sector-agropecuario---.aspx

Morbekar, A., Parihar, A., & Jadhav, R. (2020). Crop disease detection using YOLO. *2020 International Conference for Emerging Technology*, 1–5. https://doi.org/10.1109/INCET49848.2020.9153986

Peres, F., Costa, J., Meneses, K., Lerner, R., & Claudio, L. (2020). El uso de pesticidads en la agricultura la salud del trabajador rural en Brasil. *Revista Enfermeria La Vanguardia*, *6*(2), 40–47. https://doi.org/10.35563/revan.v6i2.210

Redmon, J., Divvala, S., & Girshik, R. (2016). You Only Look Once: Unified, Real-Time Object Detection. *University of Washington*.

Rivas, W., & Marzón, B. (2018). Redes neuronales artificiales aplicadas al reconocimiento de patrones. In *Redes neuronales artificiales aplicadas al reconocimiento de patrones* (Issue June, pp. 11–35). http://repositorio.utmachala.edu.ec/handle/48000/12499

Roldán, B., Roshan, R., & Sánchez, E. (2019). *Detección de enfermedades en el sector agrícola utilizando Inteligencia Artificial*. *148*(7), 419–427.

Rouse, M. (2016). *What Is Fuzzy Logic?* SearchEnterpriseAI. https://searchenterpriseai.techtarget.com/definition/fuzzy-logic

Rüping, S. (2010). SVM classifier estimation from group probabilities. *LWA 2010 - Lernen, Wissen Und Adaptivitat - Learning, Knowledge, and Adaptivity, Workshop Proceedings*, *April*, 129–135.

Saady, Y., Massi, I., Yassa, M., Mammass, D., & Benazoun, A. (2016). Automatic recognition of plant leaves diseases based on serial combination of two SVM classifiers. *Proceedings of 2016 International Conference on Electrical and Information Technologies, ICEIT 2016*, 561–566. https://doi.org/10.1109/EITech.2016.7519661

Salamanca, J., Yohan, M., & Vásquez, S. (2019). *Efecto del TLC con la Unión Europea en la exportación de las flores colombianas y los mercados que se pueden potencializar*. *2*, 21.

Salvo, A., & Valladares, G. R. (2007). Parasitoides de minadores de hojas y manejo de plagas. *Ciencia e Investigación Agraria*, *34*(3), 167–185. https://doi.org/10.4067/s0718-16202007000300001

Serrano, A. S. i. (2017). YOLO Object Detector for Onboard Driving. *Universitat Autonoma De Barcelona*, *1*(Cvc).

Singh, V., & Misra, A. K. (2017). Detection of plant leaf diseases using image segmentation and soft computing techniques. *Information Processing in Agriculture*, *4*(1), 41–49. https://doi.org/10.1016/j.inpa.2016.10.005

Suhartono, D., Aditya, W., Lestari, M., & Yasin, M. (2013). Expert System in Detecting

Coffee Plant Diseases. *International Journal of Electrical Energy*, *April*, 156–162. https://doi.org/10.12720/ijoee.1.3.156-162

Yamashita, R., Nishio, M., Do, R., & Togashi, K. (2021). Convolutional Neural Networks: An Overview and Its Applications in Pattern Recognition. *Smart Innovation, Systems and Technologies*, *195*, 21–30. https://doi.org/10.1007/978-981-15-7078-0_3

Yanes Figueroa, M., & Téllez Navarro, M. (2004). Estudio del parasitismo natural del minador de hojas, Liriomyza spp. en cultivo de judía bajo invernadero plástico en la provincia de Almería. *Boletín de Sanidad Vegetal. Plagas*, *30*(3), 563–572.

Yucra, S., Gasco, M., Rubio, J., & Gonzales, G. F. (2008). Revisión Exposición Ocupacional a Plomo Y Pesticidas Órganofosforados : Efecto Sobre La Salud Reproductiva Masculina Occupational Exposure To Lead and Organophosphorus Pesticides : Effect on Male Reproductive Health. *Rev. Peru Med Exp Salud Pública*, *25*(4), 394–402.

5 ANEXOS

Anexo 1. Ejemplos Predicciones Red Neuronal, SVM y KNN

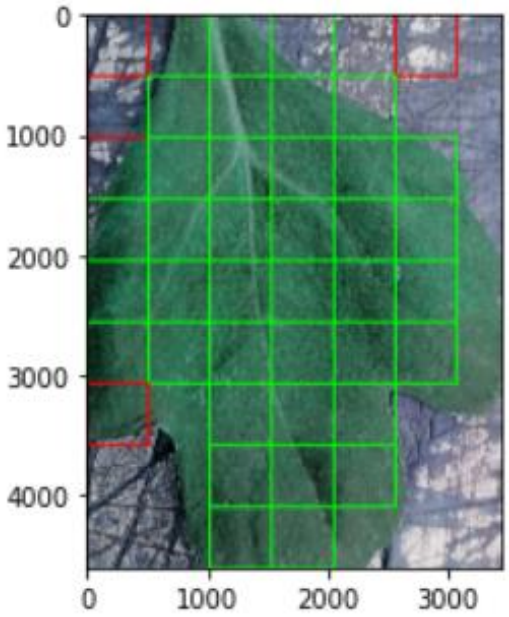

Figura 17. Predicción red neuronal

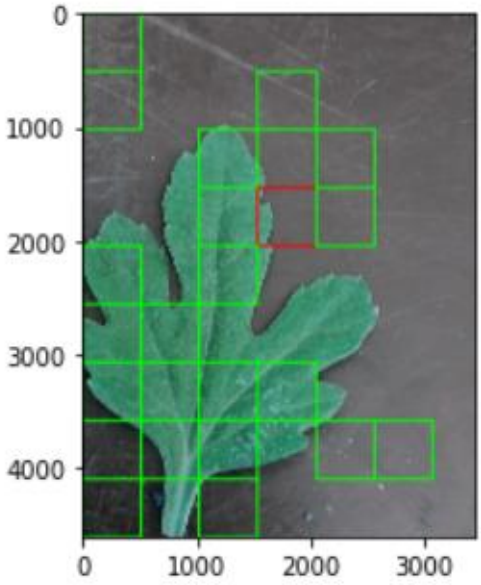

Figura 18. Predicción SVM lineal

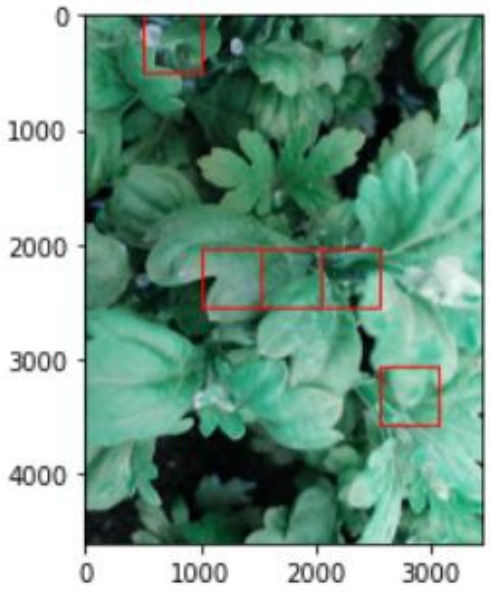

Figura 19. Predicción SVM RGF

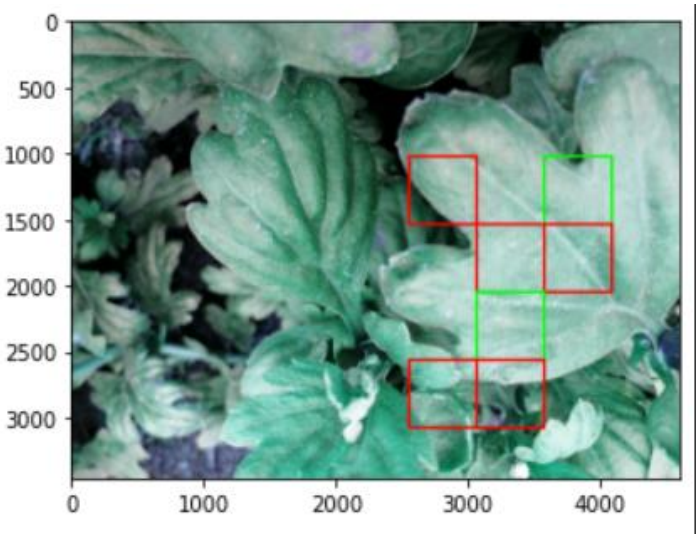

Figura 20. Predicción KNN

Anexo 2. Resultados Redes profundas

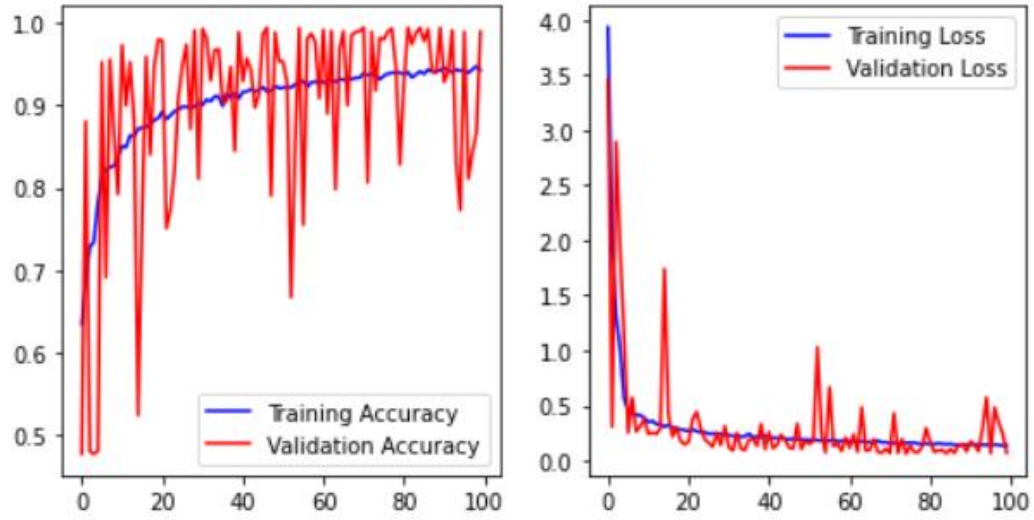

Figura 21. Resultados modelo huevo ResNet

Modelo huevo última época:

- Training Accuracy: 0.9425
- Training Loss: 0.1411
- Validation Accuracy: 0.9894
- Validation Loss: 0.0699

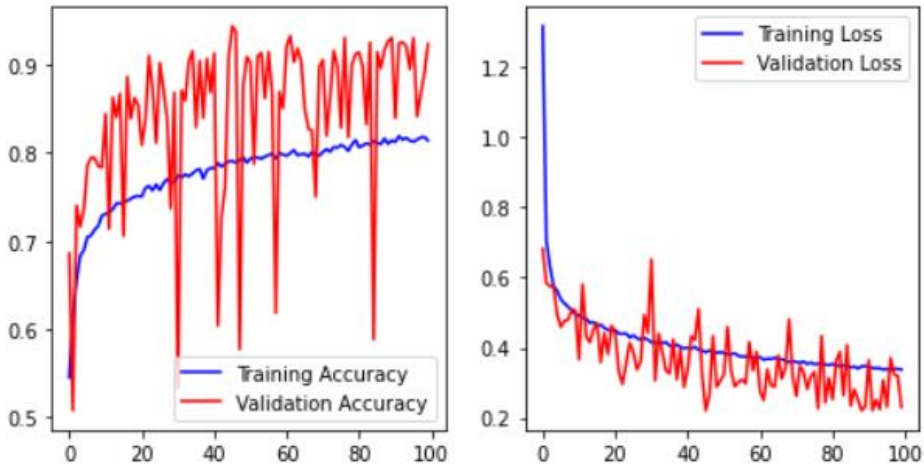

Figura 22. Resultados modelo larva ResNet

Modelo larva última época:

- Training Accuracy: 0.8138
- Training Loss: 0.3375
- Validation Accuracy: 0.9232
- Validation Loss: 0.2301

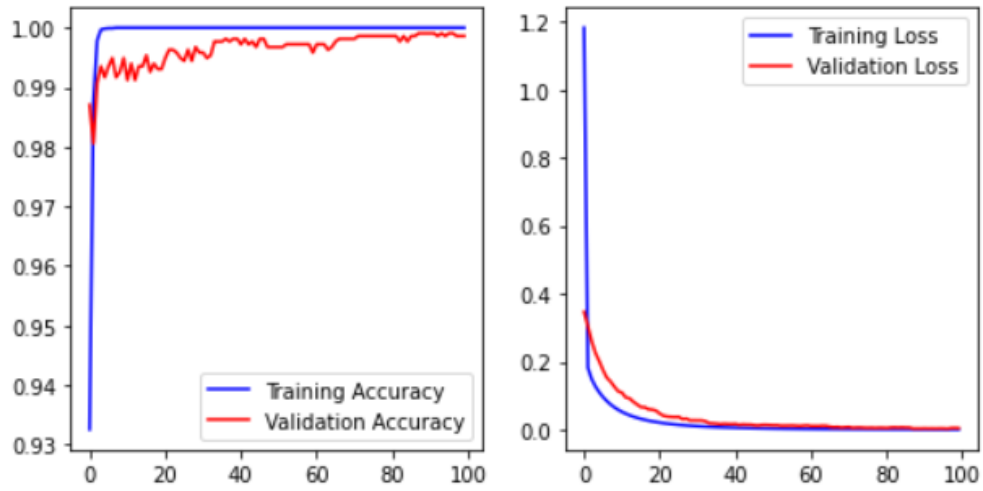

Figura 23. Resultados modelo huevo MobileNet

Modelo huevo última época:

- Training Accuracy: 1
- Training Loss: 2.42e-4
- Validation Accuracy: 0.9986
- Validation Loss: 0.0041

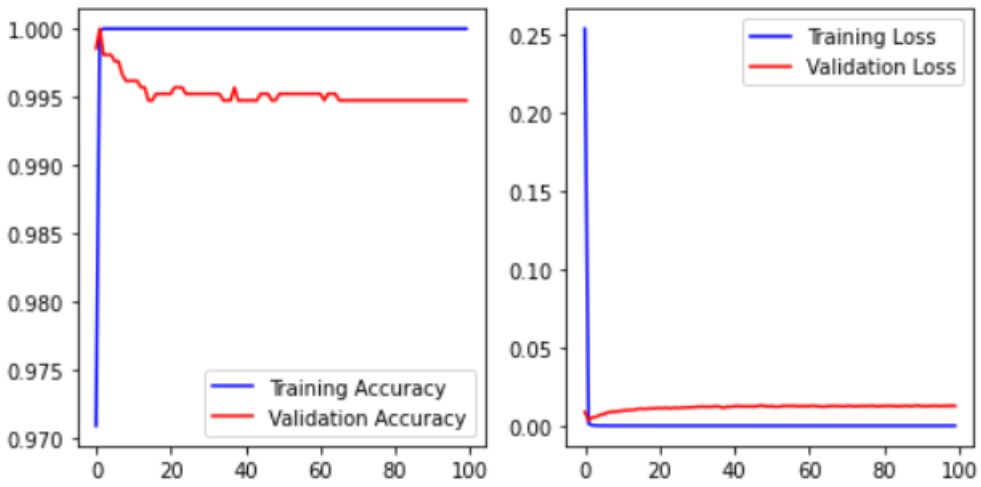

Figura 24. Resultados modelo larva MobileNet

Modelo larva última época:

- Training Accuracy: 1
- Training Loss: 9.25e-9
- Validation Accuracy: 0.9948
- Validation Loss: 0.0127

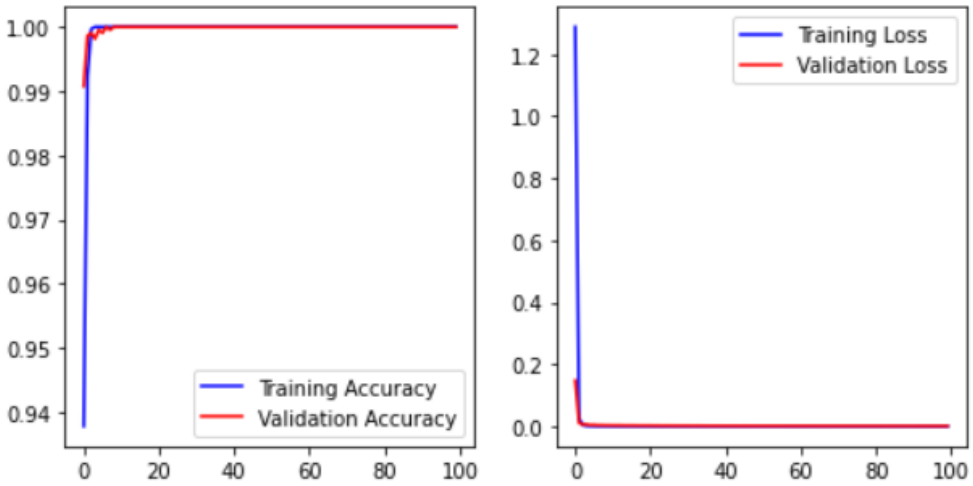

Figura 25. Resultados modelo huevo MobileNet V2

Modelo huevo última época:

- Training Accuracy: 1
- Training Loss: 9.91e-08

- Validation Accuracy: 1
- Validation Loss: 4.68e-04

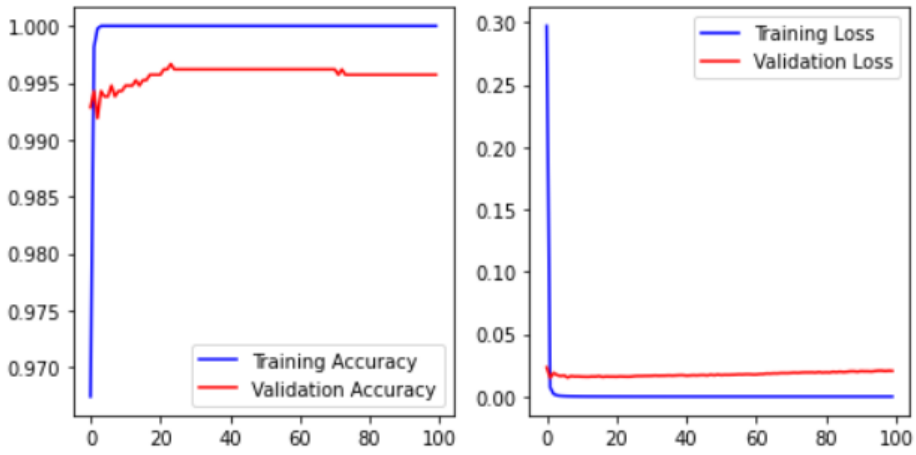

Figura 26. Resultados modelo larva MobileNet V2

Modelo larva última época:

- Training Accuracy: 1
- Training Loss: 9.51e-8
- Validation Accuracy:0.9957
- Validation Loss: 0.0207

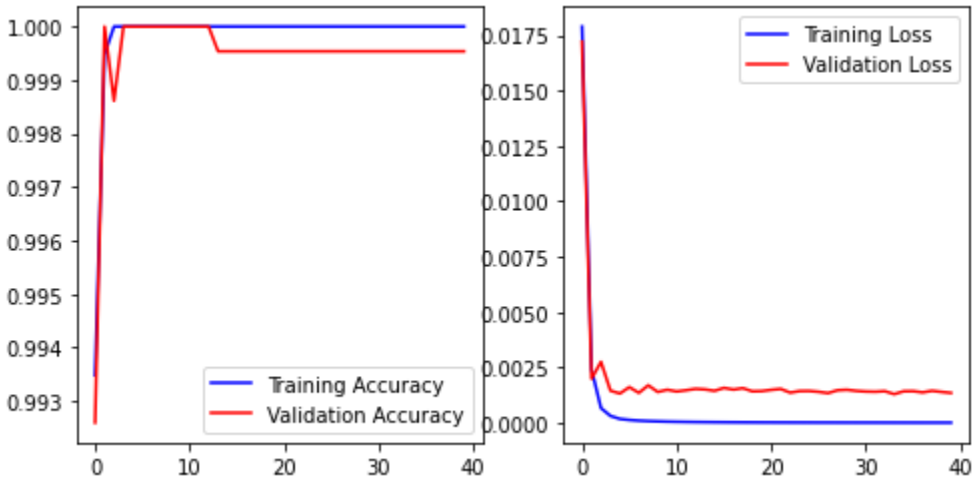

Figura 27. Resultados modelo huevo Xception

Modelo huevo última época:

- Training Accuracy: 1
- Training Loss: 2.69e-06
- Validation Accuracy: 0.9995
- Validation Loss: 0.0013

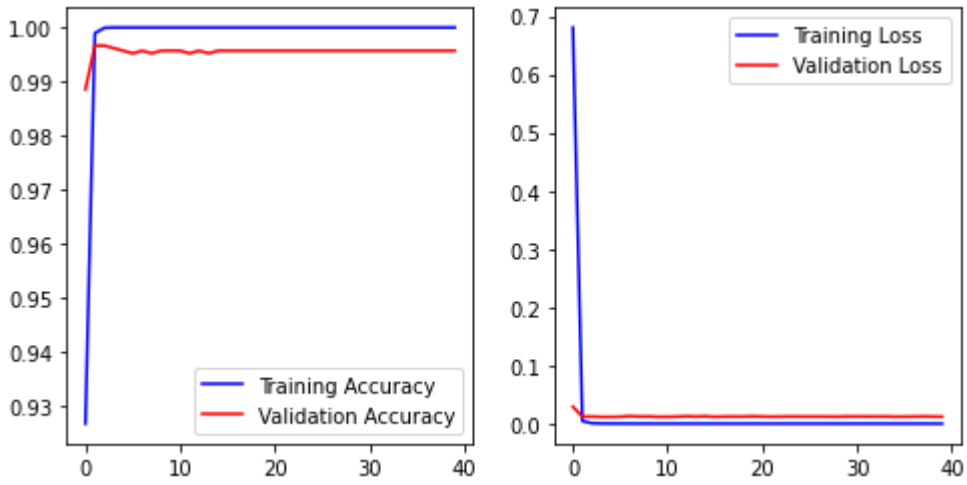

Figura 28. Resultados modelo larva Xception

Modelo larva última época:

- Training Accuracy: 1
- Training Loss: 1.57e-06
- Validation Accuracy:0.9957
- Validation Loss: 0.012