CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco

Programa de

Sistemas Autónomos de Navegación Aérea y Submarina

¨Implementación de algoritmo de aprendizaje profundo CNN para la identificación de plantas invasivas utilizando conjuntos de datos pequeños¨

TESIS

Que presenta

OSCAR RAMÍREZ AYALA

Para obtener el grado de

MAESTRO EN CIENCIAS

En

Sistemas Autónomos de Navegación Aérea y Submarina.

Directores de la Tesis:

DR. ANTONIO OSORIO CORDERO
DRA. XIAOOU LI

México, Ciudad de México **Febrero 2019**

Dedicatoria

A mi familia, que siempre me han apoyado de todas las formas posibles cuando los he necesitado, a mis padres Salvador y Yolanda, a mis hermanos Jorge y Jessica y a mi novia Diana Ivonne.

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), que por medio de su programa de becas para posgrados nacionales me brindo el apoyo económico suficiente para concluir mis estudios de maestría. Al centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV-IPN), que mediante su programa de Sistemas Autónomos de Navegación Aérea y Submarina me dieron la oportunidad de desarrollarme profesionalmente en esta área de estudio. Agradezco a todos los integrantes del laboratorio UMI-LAFMIA, a los profesores que me brindaron las herramientas necesarias para trabajar en este campo de la ciencia, a los investigadores que me brindaron ayuda cuando me encontré con complicaciones, y especialmente a mis compañeros con los que he convivido durante estos años.

Especialmente a mis padres y hermanos que siempre han estado apoyándome en mis metas durante todo mi desarrollo académico.

Resumen

Las especies invasoras no nativas son un problema que genera daño ecológico, daños económicos anuales y algunos representan una amenaza para la salud humana. En la actualidad el monitoreo de plantas invasivas se realiza con diversos métodos como son expediciones y fotografía aérea con aeronaves tripuladas y no tripuladas (UAV). Esto convirtió el análisis de imágenes en un componente clave para la identificación oportuna de plantas invasivas. El continuo incremento de datos genera nuevos retos para darle sentido a la información almacenada. El campo del aprendizaje profundo, y en particular el tipo de red neuronal llamada red neuronal convolucional (CNN), es adecuada para tareas relacionadas con análisis de imágenes. Esta red está entrenada para buscar e identificar diferentes características de objetos, sin embargo pensar en Redes neuronales profundas implica tener previamente bases de datos de imágenes del objeto de interés y además se requiere equipo computacional moderno y costoso.

En esta tesis de maestría, se realizó una revisión del fundamento teórico actual sobre clasificación e identificación de plantas invasivas utilizando CNN. Se comprobó la implementabilidad de uno modelo CNN para la identificación de plantas invasivas utilizando recursos limitados, es decir, un equipo de cómputo convencional (PC) y un conjunto de datos pequeño. Esta propuesta surge considerando que cuando se inicia en el campo del aprendizaje profundo o perteneces a otra área de investigación, no se cuenta con demasiada experiencia y normalmente no cuentas con conjuntos de imágenes almacenados. Se realizó una implementación a un conjunto de datos de flores de hortensia disponible en la página de competencias *Kaggle*, con el que realizamos una propuesta de arquitectura CNN inicial utilizando *Jupyter Notebook*, *Python* y *Keras*. Se utilizaron funciones para incrementar el conjunto de datos pequeño y se realizaron diversos experimentos para optimizar el modelo inicial. Obtuvimos un modelo final con 8 capas de convolución y activación final *Softmax* que ofreció buenos resultados considerando las limitantes en datos y en equipo de cómputo. Finalmente utilizamos el modelo final en una implementación practica para la identificación de un arbusto de Tepozán, se utilizó un conjunto de imágenes acotado capturadas de forma directa con una cámara y utilizando un Dron (F450), realizamos acotaciones del modelo obteniendo resultados satisfactorios en identificación de arbustos de Tepozán, al mismo tiempo se implementó un algoritmo utilizando *Python*, *OpenCV* y *Keras* que nos permitió hacer predicciones de imágenes y archivos de video.

Abstract

Invasive non-native species are a problem that generates ecological damage, economic damage and some represent a threat to human health. At the present time the monitoring of invasive plants is done with different methods such as expeditions, aerial photography with manned and unmanned aircraft (UAV). This turned the analysis of images into a key component for the timely identification of invasive plants; the continuous increase of data generates new challenges to make sense of the stored information. The field of deep learning, and in particular the type of neural network called Convolutional Neuronal Network (CNN), is a network suitable for tasks related to images analysis. The network is trained to search and identify different characteristics of objects. However, thinking about deep neural networks implies having databases of images of the object of interest and also modern and expensive computational equipment is required.

In this master's thesis, a review of the current theoretical basis on classification and identification of invasive plants using CNN was made. We want to check the applicability of a CNN model for the identification of invasive plants using limited resources, that is, a conventional computer (PC) and a small data set. This proposal arises considering that when you start in the field of deep learning or belong to another area of research, you do not have much experience and usually do not have sets of images stored. To achieve this, we carry out two implementations of two sets of different images. The first implementation was made to a data set of hydrangea flowers available on the *Kaggle* competences page, with which we made an initial CNN architecture proposal using *Jupyter Notebook*, *Python* and *Keras*. Functions were used to increase a small data set and various experiments were performed to optimize the initial model. We obtained a final model with 8 layers of convolution and final activation *Softmax* that offered the best results considering the limitations in data and in computer equipment. Finally, we used the final model for a practical implementation for the identification of a Tepozán shrub, the process of capturing images was controlled, the set of images were captured directly with a camera and using a drone (F450), we performed annotations obtaining satisfactory results in identification of Tepozán shrubs, at the same time an algorithm was implemented using Python, *OpenCV* and *Keras* that allowed us to make predictions of images and video files.

Índice

DEDICATORIA ..... **III**

AGRADECIMIENTOS ..... **IV**

RESUMEN ..... **V**

ABSTRACT ..... **VI**

ÍNDICE ..... **VII**

ÍNDICE DE IMÁGENES ..... **X**

ÍNDICE DE TABLAS ..... **XII**

1 INTRODUCCIÓN **2**

1.1 PLANTEAMIENTO DEL PROBLEMA 3
1.2 OBJETIVO 4
1.2.1 Objetivos particulares. *4*
1.3 ESTADO DEL ARTE 5

2 MARCO TEÓRICO **12**

2.1 APRENDIZAJE AUTOMÁTICO 12
2.1.1 Tipos *13*
2.1.2 Características del aprendizaje automático *13*
2.1.3 Generalización: riesgos del sobreajuste *14*
2.2 REDES NEURONALES 14
2.2.1 Redes neuronales con capas múltiples *16*
2.2.2 Propagación hacia atrás *16*
2.2.3 Tipos de funciones de activación *17*
2.3 VISIÓN POR COMPUTADOR 18
2.3.1 Detección de objetos *19*
2.4 REDES NEURONALES CONVOLUCIONALES 19
2.4.1 Arquitectura de una CNN *20*
2.4.2 Imagen Digital (Input) *21*
2.4.3 Capa de Convolución *23*
2.4.4 Partes de una capa convolucional *25*

2.4.5 Capa de Pooling *27*
2.4.6 Activación *28*

3 CONFIGURACIÓN INICIAL 30

3.1 MODELO CNN INICIAL 31
3.1.1 Conjunto de dato *32*
3.1.2 Pre-procesamiento de datos *33*
3.1.3 Arquitectura de la CNN *35*
3.1.4 Entrenamiento *37*
3.1.5 Pruebas de datos nunca antes vistos por el modelo *40*
3.2 ENTRENAMIENTO CON AUMENTO DE DATOS 44
3.2.1 Entrenamiento del modelo con incremento de datos (Mid) *45*
3.2.2 Prueba de datos nunca antes vistos por el modelo con incremento de datos *46*
3.3 ANÁLISIS DE RESULTADOS DE LA IMPLEMENTACIÓN INICIAL Y CON EXTENSIÓN DE DATOS 49

4 IMPLEMENTACIÓN EXPERIMENTAL Y OPTIMIZACIÓN 51

4.1 OPTIMIZACIÓN CON 6 CAPAS DE CONVOLUCIÓN (6CC) 51
4.1.1 Entrenamiento modelo 6CC *52*
4.1.2 Prueba de datos nunca antes vistos por el modelo *53*
4.2 OPTIMIZACIÓN CON 8 CAPAS DE CONVOLUCION (8CC) 56
4.2.1 Entrenamiento modelo 8CC *57*
4.2.2 Prueba de datos nunca antes vistos por el modelo *57*
4.3 OPTIMIZACIÓN DEL MODELO MODIFICANDO LA PROPORCIÓN DE LOS SUB-CONJUNTOS DE IMÁGENES (80/20) 60
4.3.1 Entrenamiento modelo 80/20 ..... *61*
4.3.2 Prueba de datos nunca antes vistos por el modelo *62*
4.4 OPTIMIZACIÓN UTILIZANDO ACTIVACIÓN FINAL SOFTMAX 64
4.4.1 Entrenamiento modelo SoftMax *66*
4.4.2 Prueba de datos nunca antes vistos por el modelo Softmax *67*
4.5 ANÁLISIS DE RESULTADOS EXPERIMENTALES 70

5 IMPLEMENTACIÓN PRÁCTICA 72

5.1 CONJUNTO DE DATOS DEL MODELO PRACTICO 73
5.2 PRE-PROCESAMIENTO DE DATOS 75
5.3 ENTRENAMIENTO DEL MODELO TEPOZÁN 75
5.4 PRUEBA DE DATOS NUNCA ANTES VISTOS POR EL MODELO TEPOZÁN 77
5.4.1 Predicción en imagen nunca antes vista por el modelo utilizando Open CV *78*
5.4.2 Predicción para archivos de video *81*
5.5 ANÁLISIS DE RESULTADOS. 83

6 CONCLUSIONES Y TRABAJO FUTURO. 87

6.1 CONCLUSIONES 87
6.2 TRABAJO FUTURO 89

A CÓDIGOS.....90

A1. MODELO INICIAL CON EXTENSIÓN DE DATOS 90
A2. ARQUITECTURA DEL MODELO CON 8 CAPAS DE CONVOLUCION Y ACTIVACIÓN SIGMOID 92
A3. ARQUITECTURA MODELO SOFTMAX 93
A4. ARQUITECTURA MODELO CNN TEPOZÁN 94
A5. MODELO CNN TEPOZÁN PARA PREDICCIÓN DE IMAGEN 95
A6 MODELO CNN TEPOZÁN PARA PREDICCIÓN EN VIDEO 96

BIBLIOGRAFIA 98

Índice de Imágenes

IMAGEN 2.1: DIFERENCIA ENTRE RED NEURONAL CONVENCIONAL Y REDES NEURONALES PROFUNDAS....16
IMAGEN 2.2: ESTRUCTURA GENERAL DE UN MODELO CNN....21
IMAGEN 2.3: IMAGEN A ESCALA DE GRISES (1 CANAL [0-255])....22
IMAGEN 2.4: FIGURA A COLOR (3 CANALES [RGB])....22
IMAGEN 2.5: ARQUITECTURA DE UNA CAPA DE CONVOLUCIÓN EN UNA CNN....24
IMAGEN 2.6: REPRESENTACIÓN DE IMAGEN DE ENTRADA Y DE SALIDA EN FORMATO DE VOLUMEN....25
IMAGEN 2.7: REPRESENTACIÓN MATRICIAL DE UN KERNEL EN UNA CNN....26
IMAGEN 2.8: REPRESENTACIÓN MATRICIAL DE UNA OPERACIÓN POOLING EN UNA CNN....27
IMAGEN 2.9: PARTES PRINCIPALES DE LA ARQUITECTURA DE UNA CNN....28

IMAGEN 3.1: IMAGEN HORTENSIA (IZQUIERDA) Y BOSQUE O FOLLAJE (DERECHA)....33
IMAGEN 3.2: DIAGRAMA A BLOQUES DE LA ARQUITECTURA DEL MODELO CNN INICIAL....36
IMAGEN 3.3: EJEMPLO DEL PROCESO DE ENTRENAMIENTO DEL MODELO CNN INICIAL....38
IMAGEN 3.4: GRAFICAS DE PRECISIÓN Y PERDIDA DEL MODELO CNN INICIAL ENTRENADO DURANTE 50 ÉPOCAS....39
IMAGEN 3.5: MATRIZ DE CONFUSIÓN PARA 200 IMÁGENES NUNCA ANTES VISTAS POR EL MODELO INICIAL....41
IMAGEN 3.6: PREDICCIÓN DEL MODELO INICIAL EN IMÁGENES DONDE LA FLOR DE HORTENSIA SE VISUALIZA CLARAMENTE....41
IMAGEN 3.7: PREDICCIÓN DEL MODELO INICIAL EN IMÁGENES DONDE LA FLOR DE HORTENSIA NO SE APRECIA CLARAMENTE....42
IMAGEN 3.8: PREDICCIÓN DEL MODELO INICIAL EN IMÁGENES DE NATURALEZA....43
IMAGEN 3.9: PREDICCIÓN DEL MODELO INICIAL EN IMÁGENES DE NATURALEZA QUE CONTIENE OBJETO NO NATURALES....43
IMAGEN 3.10: EJEMPLOS DE UNA IMAGEN APLICANDO OPERACIONES DE TRANSFORMACIÓN....45
IMAGEN 3.11: GRAFICAS DE PRECISIÓN Y PÉRDIDA DE LOS DATOS DE ENTRENAMIENTO CON AUMENTO DE DATOS....46
IMAGEN 3.12: MATRIZ DE CONFUSIÓN PROBADAS A 200 IMÁGENES NUNCA ANTES VISTAS POR EL MODELO MID....47
IMAGEN 3.13: PREDICCIÓN DE IMÁGENES CON FLORES DE HORTENSIA CLARAS CON MODELO MID....47
IMAGEN 3.14: PREDICCIÓN EN IMÁGENES DONDE LA FLOR DE HORTENSIA ES DIFÍCIL DE APRECIAR....48
IMAGEN 3.15: PREDICCIÓN DE IMÁGENES CON MODELO MID EN IMÁGENES DE NATURALEZA....48
IMAGEN 3.16: PREDICCIÓN DE IMÁGENES CON MODELO MID EN IMÁGENES DE NATURALEZA COMPLEJAS....49

IMAGEN 4.1: ARQUITECTURA CNN CON 6 CAPAS DE CONVOLUCION....52
IMAGEN 4.2: GRAFICAS DE PRECISIÓN Y PÉRDIDA DE LOS DATOS DE ENTRENAMIENTO Y VALIDACIÓN DEL MODELO 6CC....52
IMAGEN 4.3: MATRIZ DE CONFUSIÓN PROBADA A 200 IMÁGENES NUNCA ANTES VISTAS POR EL MODELO 6CC.....53
IMAGEN 4.4: PREDICCIÓN DE IMÁGENES CON MODELO 6CC DE FLORES DE HORTENSIA CLARAS.....54
IMAGEN 4.5: PREDICCIÓN DE IMÁGENES CON MODELO 6CC DONDE LA FLOR DE HORTENSIA NO ES CLARA.....55
IMAGEN 4.6: PREDICCIÓN DE IMÁGENES CON MODELO 6CC EN IMÁGENES DE NATURALEZA.....55
IMAGEN 4.7: PREDICCIÓN DE IMÁGENES CON MODELO 6CC EN IMÁGENES DE NATURALEZA COMPLEJAS....55
IMAGEN 4.8: ARQUITECTURA CNN DEL MODELO CON 8CC.....56
IMAGEN 4.9: GRAFICAS DE PRECISIÓN Y PÉRDIDA DE LOS DATOS DE ENTRENAMIENTO Y VALIDACIÓN DEL MODELO 8CC....57
IMAGEN 4.10: MATRIZ DE CONFUSIÓN PROBADA CON 200 IMÁGENES NUNCA ANTES VISTAS POR EL MODELO 8CC.....58
IMAGEN 4.11: PREDICCIÓN DE IMÁGENES CON MODELO 8CC DE FLORES DE HORTENSIA CLARAS....59
IMAGEN 4.12: PREDICCIÓN DE IMÁGENES CON MODELO 8CC DONDE LA FLOR DE HORTENSIA NO ES CLARA.....59
IMAGEN 4.13: PREDICCIÓN DEL MODELO 8CC EN IMÁGENES DE NATURALEZA.....60
IMAGEN 4.14: PREDICCIÓN DEL MODELO 8CC EN IMÁGENES DE NATURALEZA QUE CONTIENE OBJETOS.....60
IMAGEN 4.15: GRAFICAS DE PRECISIÓN Y PÉRDIDA DEL PROCESO DE ENTRENAMIENTO DEL MODELO CON 80/20....61
IMAGEN 4.16: MATRIZ DE CONFUSIÓN PROBADA CON 200 IMÁGENES NUNCA ANTES VISTAS POR EL MODELO 80/20.....62
IMAGEN 4.17: PREDICCIÓN DE IMÁGENES CON MODELO 80/20 DE FLORES DE HORTENSIA CLARAS.....63
IMAGEN 4.18: PREDICCIÓN DE IMÁGENES CON MODELO 80/20 DONDE LA FLOR DE HORTENSIA NO CLARAS....63
IMAGEN 4.19: PREDICCIÓN DEL MODELO CON PROPORCIÓN 80/20 EN IMÁGENES DE NATURALEZA....64
IMAGEN 4.20: PREDICCIÓN DEL MODELO 80/20 EN IMÁGENES DE NATURALEZA QUE CONTIENE OBJETOS....64
IMAGEN 4.21: TIPO DE IMAGEN DE HORTENSIA RODEADA DE VEGETACIÓN QUE DIFICULTA LA PREDICCIÓN.....65
IMAGEN 4.22: ARQUITECTURA CNN DEL MODELO SOFTMAX.....66
IMAGEN 4.23: GRAFICAS DE PRECISIÓN Y PÉRDIDA DEL PROCESO DE ENTRENAMIENTO DEL MODELO SOFTMAX.....67
IMAGEN 4.24: MATRIZ DE CONFUSIÓN PROBADA CON 200 IMÁGENES NUNCA ANTES VISTAS POR EL MODELO SOFTMAX.....68
IMAGEN 4.25: PREDICCIÓN DE IMÁGENES CON MODELO SOFTMAX DE FLORES DE HORTENSIA CLARAS.....68
IMAGEN 4.26: PREDICCIÓN DE IMÁGENES CON MODELO SOFTMAX DONDE LA FLOR DE HORTENSIA NO ES CLARA.....69
IMAGEN 4.27: PREDICCIÓN DEL MODELO SOFTMAX EN IMÁGENES DE NATURALEZA.....69

IMAGEN 5.1: IMAGEN DE ARBUSTO TEPOZÁN....73
IMAGEN 5.2: TEPOZÁN CON PLAGA DE GUSANO MEDIDOR (IZQUIERDA), TEPOZÁN COMPITIENDO POR RECURSOS (DERECHA). 73
IMAGEN 5.3: AFECTACIÓN DE ARBUSTO TEPOZÁN EN CARRETERAS Y EN CONSTRUCCIONES. 74
IMAGEN 5.4: IMAGEN ARBUSTO TEPOZÁN (IZQUIERDA) Y ZONA BOSQUE (DERECHA). 74
IMAGEN 5.5: ARQUITECTURA CNN DEL MODELO TEPOZÁN CON 8 CAPAS CONVOLUCIONALES Y ACTIVACIÓN SOFTMAX. 76
IMAGEN 5.6: GRAFICAS DE PRECISIÓN Y PÉRDIDA DE LOS DATOS DE ENTRENAMIENTO Y VALIDACIÓN DEL MODELO TEPOZÁN.....77

IMAGEN 5.7: MATRIZ DE CONFUSIÓN PROBADA CON 200 IMÁGENES NUNCA ANTES VISTAS POR EL MODELO TEPOZÁN. 78
IMAGEN 5.8: PREDICCIÓN MODELO TEPOZÁN, IMAGEN NUNCA ANTES VISTA POR EL MODELO. 79
IMAGEN 5.9: PREDICCIÓN DEL MODELO TEPOZÁN EN CUATRO IMÁGENES QUE CONTIENEN ARBUSTO DE TEPOZÁN. 80
IMAGEN 5.10: PREDICCIÓN MODELO TEPOZÁN EN CUATRO IMÁGENES QUE NO CONTIENEN ARBUSTO DE TEPOZÁN. 81
IMAGEN 5.11: SECUENCIA DE IMÁGENES DE UN VIDEO CON PREDICCION DEL MODELO TEPOZAN EN UNA ZONA PLANA. 82
IMAGEN 5.12: PREDICCION DE UNA SECUENCIA DE IMÁGENES DE UN VIDEO DE COSTADOS DE CARRETERA. 83
IMAGEN 5.13: PREDICCIÓN MODELO TEPOZÁN DE IMAGEN CON FONDO DIFERENTE. 84
IMAGEN 5.14: PREDICCIÓN MODELO TEPOZÁN DE UNA ZONA DIFERENTE A LA ZONA DE ESTUDIO. 85

Índice de Tablas

Tabla 3.1 Parámetros obtenidas en el proceso de entrenamiento del modelo inicial……..39

Tabla 3.2 Parámetros de precisión y perdida en imágenes nunca antes vistas por el modelo inicial…….40

Tabla 3.3 Parámetros de precisión y perdida en imágenes nunca antes vistas por el modelo con incremento de datos…46

Tabla 4.1 Parámetros de precisión y perdida en imágenes nunca antes vistas por el modelo 6CC…….53

Tabla 4.2 Parámetros de precisión y perdida en imágenes nunca antes vistas por el modelo 8CC…….58

Tabla 4.3 Parámetros de precisión y perdida en imágenes nunca antes vistas por el modelo con proporción 80/20…….62

Tabla 4.4 Parámetros de precisión y perdida en imágenes nunca antes vistas por el modelo Softmax……..67

Tabla 5.1 Parámetros de precisión y perdida en imágenes nunca antes vistas por el modelo Tepozán…….....78

CAPÍTULO I

1 INTRODUCCIÓN

Las especies invasoras no nativas pueden ser definidas como aquellas plantas introducidas, de origen foráneo, que llegan a establecerse rápidamente en un nuevo ambiente, donde fácilmente proliferan y se propagan exitosamente, convirtiéndose en un problema debido a su elevada densidad. La propagación de las plantas invasoras resulta exitosa debido a que los diversos controles naturales que permiten mantenerlas bajo control en sus sitios nativos, tales como depredadores y agentes patológicos causantes de enfermedades, no existen en el nuevo hábitat [1].

Estas plantas tienden a crecer agresivamente mostrando gran competitividad por espacio y recursos disponibles, aun en aquellos que resultan difíciles a otras especies; aparecen en terrenos cultivados, jardines, orillas de caminos, sitios perturbados y diversos ecosistemas. Además del daño ecológico las especies invasoras causan enormes daños económicos anuales y algunos representan una amenaza para la salud humana [2].

El monitoreo de plantas invasivas implica el registro de información específica en forma constante a lo largo del tiempo, esto proporciona datos que conducen a un mejor entendimiento de las especies y contribuye a su manejo más efectivo. El tipo de dato comúnmente almacenado suele ser de tipo imagen, para su adquisición se aplican diversos métodos entre los que destacan expediciones, fotografía aérea con avionetas y helicópteros o fotografía con vehículos aéreos no tripulados (UAV). Por lo tanto el análisis de imágenes se ha convertido en un componente clave para la identificación oportuna de plantas Invasivas. El continuo incremento de datos genera nuevos retos para darle sentido a la información obtenida. Los datos generalmente no pueden ser explorados y visualizados eficientemente con métodos tradicionales, por lo que se requiere de técnicas de análisis y medición automatizadas que permitido un mayor rendimiento y sistemas de identificación más rápidos y precisos [3].

El aprendizaje profundo es un campo emergente que promete resultados incomparables en muchos problemas de análisis de datos, uno de los métodos más populares para la clasificación de plantas son las Redes Neuronales Convolucionales (CNN por siglas

en ingles). Una CNN convoluciona las características aprendidas con los datos de entrada y emplea capas convolucionales 2D, lo cual hace que esta arquitectura resulte adecuada para procesar datos en formato de imagen o video [4]. La implementación de un algoritmo de tipo CNN en conjunto con algoritmos de visión artificial para la identificación oportuna de plantas invasivas en imágenes, representaría una herramienta que puede reducir el tiempo de identificación, los costos de operación y los riegos para el personal de expedición.

1.1 Planteamiento Del Problema

Las especies invasoras representan un riesgo que genera efectos perjudiciales sobre el medio ambiente, la economía e incluso la salud humana. A pesar del impacto generalizado, los esfuerzos para rastrear la ubicación y propagación de las especies invasoras resultan lentos y costosos. Actualmente, el monitoreo de la distribución de los ecosistemas y plantas depende del conocimiento experto. Los científicos capacitados visitan las áreas designadas y toman información de las especies que las habitan. El uso de una fuerza laboral altamente calificada es costoso, ineficiente en el tiempo e insuficiente, ya que los humanos no pueden cubrir grandes áreas durante el muestreo [5]. Debido a la limitación de las técnicas convencionales de monitoreo e identificación de plantas invasivas se ha recurrido a la implementación de nuevas tecnologías como la fotografía digital, vehículos aéreos tripulados y vehículos aéreos no tripulados (UAV). La captura continúa de información en imagen utilizando *UAV*, expedición y fotografía con aeronave tripulada, genera un gran incremento de la cantidad de información de la región de estudio. El análisis de grandes cantidades de información en imagen obtenida en distintas misiones de monitoreo requiere la utilización de nuevas herramientas de software avanzadas, que puedan dar sentido a la información permitiendo la identificación oportuna de especies invasoras para evitar su propagación y reducir el daño ecológico [6].

Los enfoques de Aprendizaje Profundo como los de tipo CNN se especializan en procesar imágenes, este método ofrecen una alternativa de vanguardia de autoaprendizaje que permite la adaptación a diferentes especies de plantas invasivas. Una CNN se entrena pasando iterativamente imágenes de ejemplo que contienen la planta invasiva que se desea detectar con la red. Sin embargo surge otra problemática, y es que, cuando se habla de implementación de modelos de aprendizaje profundo CNN se asocia la necesidad de grandes cantidades de información para su realización y un investigador que

inicia en el área de aprendizaje no cuenta con la experiencia ni con bases de datos de información [7]. Existen alternativas para implementar una CNN con unos pocos cientos de imágenes, utilizando una biblioteca de Redes Neuronales de código abierto escrita en Python llamada *Keras*. La biblioteca *Keras* cuenta funciones y operadores que permites Entrenar un CNN desde cero en un conjunto de datos de imágenes muy pequeño. Lo que se busca en la presente investigación es realizar una estrategia para abordar el problema de detección de plantas invasivas utilizando un conjunto de datos pequeño, buscando producir resultados razonablemente precisos a pesar de la relativa falta de datos y la utilización de equipo de cómputo convencional. Considerando estas limitantes se busca integrar técnicas de visión por computadora que permitan utilizar el modelo para la identificación de plantas invasivas en archivos de imagen o video captadas de forma directa del área natural de estudio.

1.2 Objetivo

Implementar un algoritmo de aprendizaje profundo CNN que permita identificar plantas invasivas en archivos imagen o videos captados directamente de la zona de estudio por las diferentes formas de monitoreo (expedición, UAV, Aeronaves tripuladas etc.), con la finalidad de que sirva como herramienta de apoyo de identificación oportuna de plantas invasivas, permitiendo reducir el tiempo de detección y los costos de las operaciones.

1.2.1 Objetivos particulares.

- Profundizar el fundamento teórico de las CNN y visión por computadora.
- Implementar una CNN inicial con un conjunto de datos pequeño existente.
- Optimizar el modelo CNN inicial.
- Implementación modelo CNN practico con un conjunto de datos acotado de una planta invasiva propuesta (Tepozán).
- Implementar programa de visión que permita detectar la planta invasiva propuesta en archivos imagen o de video utilizando el modelo CNN práctico.

1.3 Estado del Arte

Las técnicas de aprendizaje profundo en particular el método CNN se ha utilizado ampliamente en tareas de detección de características o de objetos dentro de una imagen, debido a los buenos resultados que una CNN ofrece se han desarrollado competencias con grandes conjuntos de datos de imágenes. El aprendizaje profundo es un campo emergente que promete resultados incomparables en muchos problemas de análisis de datos, por ejemplo en [8] se realizó una comparación entre dos arquitecturas Alexnet y VGGNet. Estos modelos son redes neuronales convolucionales pre-entrenadas que cuentan con filtros de clasificación definidos, ambas se aplicaron para clasificación de imágenes de ojos humanos. En la investigación se muestra el análisis general del funcionamiento y las características de las redes neuronales convolucionales.

Otro ejemplo de comparación de CNN pre-entrenadas se realizó en [9], este artículo muestra la comparación y evaluación de dos redes neuronales artificiales de tipo convolucional (ALEXNET y VGGNET), aplicadas a un sistema de reconocimiento automatizado de objetos. Para lograr esto, se programaron las dos topologías de redes neuronales convolucionales en MATLAB, las cuales fueron entrenadas con cuatro categorías de imágenes. Esta investigación determino qué las redes neuronales convolucionales presentan mayores beneficios en el proceso de clasificación. La arquitectura AlexNet realizo la identificación del objeto más rápido que el de VGG16, debido a la gran cantidad de capas convolucionales que contienen, afectando el tiempo para la obtención de una respuesta.

En el reconocimiento de especies animales también ha recurrido al uso de redes neuronales convolucionales por ejemplo en [10] se propusieron un nuevo algoritmo de reconocimiento de especies basado en una red neuronal convolucional, para la clasificación de animales salvajes en datos de imágenes. Los datos de las imágenes se capturaron con una cámara activada por movimiento y se segmentaron automáticamente utilizando el algoritmo de corte de gráficos. El primer plano en movimiento se selecciona como la región de interés y se alimenta al algoritmo de reconocimiento de especies propuesto. Está claro que el reconocimiento de especies basado en una red neuronal profunda convolucional propuesto logra un rendimiento superior. Según su propuesta, este es el primer intento de reconocimiento automático de especies basado en la visión por ordenador en las imágenes reales captadas por una cámara trampa. Otro caso de im-

plementación relacionada con animales se describe en [11], donde se utilizó un método de clasificación de tipo CNN para el estudio de abultamiento muscular en salmón. Este estudio se llevó a cabo para desarrollar un método de análisis automático de imágenes para la detección rápida, precisa y no invasiva de manchas enormes en las canales de salmón. Los filetes de salmón se clasifican como muestras sanas o defectuosas en función del número de regiones abiertas candidatas en el paso preliminar que aplica el umbral local adaptativo. Los resultados supervisados de la clasificación se compararon utilizando histogramas de gradientes orientados (HOG) y extractores de características de red neuronal convolucional (CNN). Se demostró que las características de la CNN superaron las características de HOG con tasas de clasificación correctas (CCR) de 0.927 y 0.916 para el conjunto de datos de prueba y validación cruzada, respectivamente.

Los avances científicos relacionados con clasificación de plantas se han incorporado métodos de aprendizaje profundo y visión por computadora para la automatización de la agricultura. Esto se debe a que, el conocimiento botánico de las plantas es esencial para mejorar el desarrollo agrícola. De forma directa la necesidad de desarrollar cultivos más resistentes genéticamente analizando los fenotipos de los cultivos ha impulsado el desarrollo de técnicas de detección con CNN, tal es el caso de [12], donde se demuestra la capacidad de un marco de aprendizaje automático para identificar y clasificar un conjunto diverso de tensiones foliares en la planta de soja con una precisión notable. También presenta un mecanismo de explicación que utiliza el mapeo de activación de clase ponderado por gradiente que aísla los síntomas visuales utilizados por el modelo para hacer predicciones. Esta identificación no supervisada de síntomas visuales únicos para cada estrés proporciona una medida cuantitativa de la gravedad del estrés, lo que permite la identificación, clasificación y cuantificación de la característica. El modelo hace buenas predicciones para otras especies (que no son de soja), lo que demuestra una capacidad de transferencia de aprendizaje.

En [13] se establece la importancia de medir muchas características de fenotipo de plantas en conjuntos grandes de imágenes, estas mediciones ayudan al descubrimiento genético. La motivación para encontrar un enfoque totalmente automático se limita por el tamaño de los conjuntos de datos, que a menudo se capturan de forma robótica, impiden la inspección manual. Se muestro el éxito que ofrece el aprendizaje profundo cuando se aplican al desafiante problema del fenotipado de plantas de trigo basado en imágenes y muestra resultados de vanguardia ($>$ 97% de precisión) para la identificación y localización de características de raíz y brote de trigo.

Debido a los resultados de precisión tan satisfactorios en la implementación de CNN en [14] propone la implementación de tecnología que facilite el proceso de detección de alto rendimiento de miles de líneas cultivadas en el campo para acelerar la mejora de los cultivos, buscando incrementar la tolerancia a las enfermedades. El desarrollo de métodos de fenotipado de alto rendimiento orientados a cultivos avanzo considerablemente en los últimos 10 años a través del avance tecnológico en el desarrollo de sensores y la computación de alto rendimiento, permitió utilizar técnicas de tipo máquinas de aprendizaje (ML), que requieren un gasto computacional considerable para obtener resultados precisos. Estas técnicas son diseñadas para analizar la genética de los rasgos cuantitativos, incluido el rendimiento de los cultivos y la tolerancia a las enfermedades.

En [15] Se Capacito a una gran red neuronal convolucional profunda para clasificar las 1.2 millones de imágenes de alta resolución en el concurso *ImageNet* LSVRC-2010 en 1000 clases diferentes. En los datos de las pruebas, logramos tasas de error top-1 y top 5 de 37.5% y 17.0%, lo cual es considerablemente mejor que el estado de la técnica anterior. La red neuronal, que tiene 60 millones de parámetros y 650,000 neuronas, consta de cinco capas convolucionales, algunas de las cuales son seguidas por capas de agrupación máxima, y tres capas totalmente conectadas con un *Softmax* final de 1000 vías. Para acelerar el entrenamiento, utilizaron neuronas no saturantes y una implementación de *GPU* muy eficiente de la operación de convolución. Para reducir el ajuste excesivo en las capas totalmente conectadas, emplearon un método de regularización recientemente desarrollado llamado abandono (*dropout*) que demostró ser muy efectivo.

La identificación del taxón es un paso importante en muchos estudios ecológicos de plantas. Su eficiencia y reproducibilidad podrían beneficiarse enormemente de la automatización parcial de esta tarea. En [16] desarrollaron un sistema de aprendizaje profundo para aprender características discriminativas de las imágenes de hojas de plantas junto con un clasificador para la identificación de especies de plantas. Al comparar nuestros resultados con sistemas personalizados como *LeafSnap*, demuestran que el aprendizaje de las funciones mediante una red neuronal convolucional (CNN) proporciona una mejor representación de las imágenes de hojas en comparación con las funciones creadas a mano.

Las investigaciones realizadas a la detección de enfermedades en plantas también ha ido adoptando los métodos de aprendizaje profundo de tipo CNN tal es el caso de [17] donde describen diferentes mecanismos tradicionales para reconocer y clasificar las en-

fermedades de la hoja del maíz mediante análisis químico u observación visual. Debido a los inconvenientes de los mecanismos tradicionales para reconocer las enfermedades de la hoja de maíz, como el costo, la inconsistencia, la propensión al error, a tomar más tiempo; requiere personal profesional, instrumentos especializados, ineficientes, etc. Por lo tanto, proponen el desarrollo del modelo de clasificación y reconocimiento de enfermedades de la hoja del maíz utilizando técnicas de formación de imágenes y aprendizaje automático para ayudar a los expertos. Desarrollaron técnicas de análisis de imágenes digitales basadas en características de textura, color y morfología para reconocer y clasificar las enfermedades de la hoja de maíz y la hoja sana. Utilizaron un conjunto de muestras de imágenes de la enfermedad de la hoja de maíz de la zona de cultivo de maíz de Etiopía. El algoritmo de los clasificadores de la Red Neural Artificial con características combinadas de precisión para cada cuatro clases (óxido común, tizón de la hoja, hoja sana y mancha de la hoja) son 92.5%, 100%, 90% y 95.0% respectivamente, y el rendimiento general fue de 94.4%. Concluyeron que, hubo un resultado satisfactorio para reconocer y clasificar las enfermedades de la hoja del maíz.

Otro ejemplo [18], describe desarrollaron modelos de redes neuronales convolucionales para realizar la detección y diagnóstico de enfermedades de las plantas utilizando imágenes de hojas simples de plantas sanas y enfermas, a través de metodologías de aprendizaje profundo. El entrenamiento de los modelos se realizó con el uso de una base de datos abierta de 87,848 imágenes, que contiene 25 plantas diferentes en un conjunto de 58 clases distintas de combinaciones de [plantas, enfermedades], incluidas las plantas sanas. Se capacitaron varias arquitecturas de modelos, y el mejor rendimiento alcanzó una tasa de éxito del 99.53% en la identificación de la combinación [planta, enfermedad] correspondiente (o planta sana). La tasa de éxito significativamente alta hace que el modelo sea una herramienta muy útil de asesoramiento o alerta temprana, y un enfoque que podría ampliarse aún más para respaldar un sistema integrado de identificación de enfermedades de plantas para operar en condiciones reales de cultivo.

Continuando con la identificación de hojas en [19] usaron un conjunto de datos públicos de 54,306 imágenes de hojas de plantas enfermas y sanas recolectadas en condiciones controladas, capacitamos a una red neuronal convolucional profunda para identificar 14 especies de cultivos y 26 enfermedades. El modelo entrenado logra una precisión del 99.35% en un conjunto de pruebas retenido, lo que demuestra la viabilidad de este enfoque. Al probar el modelo en un conjunto de imágenes recopiladas de fuentes en línea de confianza, es decir, tomadas en condiciones diferentes a las imágenes utilizadas para la capacitación, el modelo aún alcanza una precisión del 31,4%. Si bien esta preci-

sión es mucho mayor que la que se basa en la selección aleatoria (2.6%), se necesita un conjunto más diverso de datos de entrenamiento para mejorar la precisión general. En general, el enfoque de la capacitación de modelos de aprendizaje profundo en conjuntos de datos de imágenes cada vez más grandes y disponibles públicamente presenta un camino claro hacia el diagnóstico de enfermedades de cultivos asistidas por teléfono inteligente en una escala global masiva.

En [20], se implementan redes neuronales convolucionales (CNN) para aprender representaciones de características no supervisadas de 44 especies de plantas diferentes, recolectadas en el Real Jardín Botánico de Kew, Inglaterra. Para obtener la intuición de las características elegidas del modelo CNN en oposición a una solución de caja negra, se utiliza una técnica de visualización basada en las redes convolucionales. Los resultados experimentales que utilizan características CNN con diferentes clasificadores, muestran consistencia y superioridad en comparación con las soluciones de vanguardia que se basan en características creadas a mano.

En la identificación de plantas en imagen se ha convertido en un enfoque interdisciplinario tanto en taxonomía botánica como en visión computacional. En [21] se presenta un conjunto de datos de imágenes de plantas recogido por teléfono móvil en una escena natural, que contiene 10,000 imágenes de 100 especies de plantas ornamentales en el campus de la Universidad Forestal de Beijing. Implementaron un modelo de aprendizaje profundo de 26 capas que consta de 8 bloques de construcción residuales diseñado para la clasificación de plantas a gran escala en un entorno natural. El modelo propuesto alcanza una tasa de reconocimiento del 91,78% en el conjunto de datos BJFU100, lo que demuestra que el aprendizaje profundo es una tecnología prometedora para la silvicultura inteligente.

Las nuevas tecnologías acelerar la recopilación e integración de datos de observación botánica sin procesar es un paso crucial hacia un desarrollo sostenible de la agricultura y la conservación de la biodiversidad. En [22] se propone una herramienta de identificación basada en imágenes, disponible como aplicación web y móvil, se sincroniza con esos datos crecientes y permite a cualquier usuario consultar o enriquecer el sistema con nuevas observaciones. El modelo funciona con un rango de clasificación de hasta cinco órganos diferentes de manera contraria a los enfoques anteriores que se basan principalmente en la hoja. Esto permite consultar el sistema en cualquier período del año y con imágenes complementarias que componen una observación de la planta.

La problemática de plantas invasivas se aborda en [23], donde se investigó la combinación de procesamiento de imágenes y clasificación supervisada para identificar plantas invasoras de iris de bandera amarilla, en imágenes recolectadas por una cámara de luz visible no calibrada que lleva un vehículo aéreo no tripulado. Específicamente, los pasos de procesamiento de imágenes del umbral de color, el emparejamiento de plantillas o la eliminación de manchas antes de entrenar a un clasificador aleatorio de bosques supervisado se exploran en términos de sus beneficios para mejorar la clasificación resultante de plantas dentro de una imagen.

En [24], el desafío *PlantCLEF* 2016 se centró en la identificación de especies de árboles, hierbas y helechos basándose en diferentes tipos de imágenes. El objetivo de la tarea era clasificar las plantas en las imágenes por especies y dar un puntaje de confianza que describa la probabilidad de que una predicción sea cierta. Se elaboraron diferentes métodos de clasificación para este desafío. Se utilizó clasificador de tipo *SVM*, el algoritmo de clasificación de vectores con el núcleo función de base radial (RBF). Además, aplicaron un método de aprendizaje profundo para entrenar redes neuronales convolucionales (CNN) para el aprendizaje de características y capas totalmente conectadas con salida de *Softmax* para la clasificación.

CAPÍTULO II

2 MARCO TEÓRICO

En este capítulo, se proporciona los fundamentos teóricos necesarios para realizar la implementación de la red neuronal CNN para identificación de plantas invasivas. Se discutirán los detalles relevantes del aprendizaje automático, las redes neuronales de convolucion y la visión por computadora.

2.1 Aprendizaje Automático

Los algoritmos de aprendizaje automático son ampliamente utilizados en aplicaciones de visión artificial. Antes de considerar las tareas relacionadas con la imagen, vamos a echar un breve vistazo a los conceptos básicos del aprendizaje automático (*Machine Learning ML* por sus siglas en ingles). El aprendizaje automático se ha convertido en una herramienta útil para modelar problemas que de otra manera serían difíciles de formular con exactitud. Los programas de computadora clásicos están explícitamente programados a mano para realizar una tarea [25]. El aprendizaje automático se refiere a un grupo de enfoques de modelado computarizado que pueden aprender patrones a partir de los datos para tomar decisiones automáticas sin programar reglas explícitas. La idea principal de ML es utilizar de manera efectiva las experiencias o escenarios de ejemplo para descubrir estructuras subyacentes, similitudes o diferencias presentes en los datos, esto para explicar o clasificar adecuadamente una nueva experiencia o un escenario de ejemplo. Una habilidad clave de las herramientas de ML es su capacidad para generalizar tendencias y patrones a partir de los datos disponibles. [26] A medida que aumenta la disponibilidad de la capacidad y los datos computacionales, el aprendizaje automático se ha vuelto cada vez más práctico a lo largo de los años, hasta el punto de no se restringe a aplicaciones del área de ciencias de la computación sino que se emplea en otro campos de la investigación.

2.1.1 Tipos

El aprendizaje automático se puede dividir en 2 tipos fundamentales, el aprendizaje supervisado es un algoritmo de aprendizaje conformado por múltiples muestras o datos que han sido anotados o etiquetados por un desarrollador humano. Por ejemplo, en el problema de detección de objetos usamos imágenes de entrenamiento donde los humanos han marcado las ubicaciones y clases de objetos relevantes. Después de aprender de los ejemplos, el algoritmo es capaz de predecir las anotaciones o etiquetas de los datos nunca antes vistos. En la clasificación, el algoritmo intenta predecir la clase correcta de un dato nunca antes visto basándose en los datos utilizados en el entrenamiento [26]. En el aprendizaje no supervisado no se cuentan con etiquetas asociadas con los datos de entrada y, por lo tanto, no podemos corregir nuestro modelo si realiza una predicción incorrecta, el algoritmo intenta aprender propiedades útiles de los datos sin que un desarrollador humano indique cuál debe ser el resultado correcto.

2.1.2 Características del aprendizaje automático

De forma general cuando se desea realizar un desarrollo de *ML* se requiere realizar pre procesamientos de datos, previamente a iniciar con el entrenamiento del modelo. El pre procesamiento de los datos en un nuevo espacio variable más simple se denomina extracción de características [26]. A menudo, es poco práctico o imposible utilizar directamente los datos de entrenamiento de dimensión total. Más bien, los detectores están programados para extraer características interesantes de los datos, y estas características se utilizan como entrada para el algoritmo de aprendizaje automático. En el pasado, los detectores de características a menudo eran hechos a mano. El problema con este enfoque es que no siempre sabemos de antemano qué características son interesantes. La tendencia en el aprendizaje automático también se ha orientado hacia el aprendizaje de los detectores de características, lo que permite el uso de los datos completos [27].

2.1.3 Generalización: riesgos del sobreajuste

Debido a que los datos de entrenamiento no pueden incluir toda la característica posible de las entradas, el algoritmo de aprendizaje debe poder generalizar los datos evaluando que tipo de datos se considerarían redundantes. La estimación del modelo demasiado simple puede no capturar aspectos importantes del modelo real. Por otro lado, los métodos demasiado complejos pueden sobreponerse al modelar detalles y ruido poco importantes, lo que también conduce a una mala generalización. Normalmente, el ajuste excesivo (*overfitting*) ocurre cuando se utiliza un método complejo junto con muy pocos datos de entrenamiento. Un modelo con sobre ajuste aprende a modelar los ejemplos conocidos pero no entiende cual es la conexión entre ellos. El rendimiento del algoritmo puede evaluarse a partir de la calidad y cantidad de errores. Una función de pérdida, como el error cuadrático medio, se utiliza para asignar un porcentaje de error. El objetivo en la fase de entrenamiento es minimizar esta pérdida [26].

2.2 Redes Neuronales

Los primeros enfoques generales sobre las redes neuronales artificiales (NN, por sus siglas en inglés), fueron inspirados por las neuronas biológicas, que usan un conjunto de conexiones similares a neuronas simuladas. La investigación pionera incluye la unidad de lógica de umbral de Warren McCulloch y Walter Pitts en 1943 y la percepción de Frank Rosenblatt en 1957 [28].

El modelo biológico de una neurona consiste en recibir señales de entrada a través de sus dendritas las cuales producen señales que viajan a lo largo de una extensión en forma de tubo llamado *Axon*, el cual tiene la función de transmitir el potencial de acción a otra neurona a las dendritas de otra neurona por medio de la sinapsis [29]. En cambio una neurona artificial se puede entender como un conjunto de activaciones que se propagan a través de una estructura de red, desencadenadas por datos de entrada y que dan como resultado un patrón de activación de salida. Aunque la inspiración de la biología es aparente, sería engañoso poner demasiado énfasis en la conexión entre las neuronas artificiales y las neuronas biológicas.

El cerebro humano contiene aproximadamente 100 mil millones de neuronas que operan en paralelo [30]. Las neuronas artificiales son funciones matemáticas implementadas en más o menos computadoras en serie. La investigación en redes neuronales se guía principalmente por los desarrollos en ingeniería y matemáticas en lugar de la biología.

En el modelo computacional de una neurona, las señales que viajan a lo largo de los axones x_0 interactúan de manera multiplicativa w_0x_0 con las dendritas de la otra neurona basadas en la fuerza sináptica w_0 comúnmente llamada Pesos en el área de computación. La idea es que las fuerzas sinápticas o pesos w puedan aprenderse y en consecuencia controlan la fuerza de influencia de una neurona. En el modelo básico, las dendritas llevan la señal al cuerpo de la célula donde todas se suman. Si la suma final está por encima de un cierto umbral, la neurona puede enviar un potencial de acción o *Spike* a lo largo de su axón. En la Imagen 2.1 se muestra una neurona artificia.

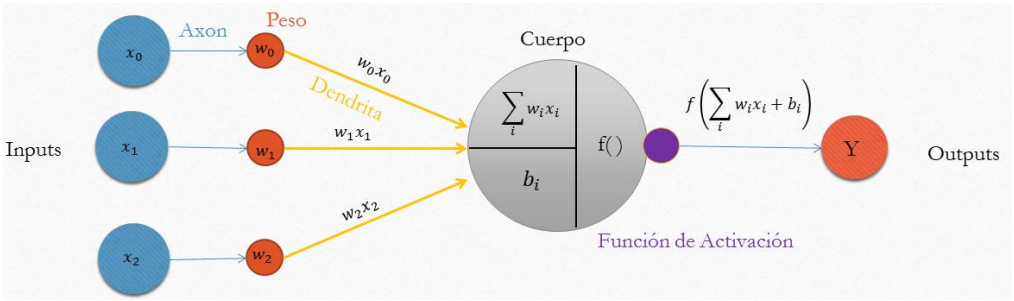

Ilustración 2.1:Modelo matemático neurona

La neurona recibe m parámetros de entrada x_i y m parámetros de peso w_i. Los parámetros de ponderación suelen incluir un término de polarización o "*bias*" que tiene una entrada ficticia con un valor comúnmente fijo. Las entradas y ponderaciones se combinan y suman de forma lineal. La suma se alimenta a una función de activación que produce la salida de la neurona, su ecuación se expresa de la siguiente forma.

$$y = f\left(\sum_i \; w_i x_i + b_i\right) \tag{1}$$

La neurona se entrena seleccionando cuidadosamente los pesos para producir una salida deseada para cada entrada.

2.2.1 Redes neuronales con capas múltiples

Las redes neuronales convencionales generalmente usan tres capas, una para la entrada, una capa interna y una capa de salida. Las redes neuronales profundas pueden contener muchas capas internas adicionales (de ahí el término "profundo") y con una complejidad incrementada aumentan significativamente el poder discriminativo [31].

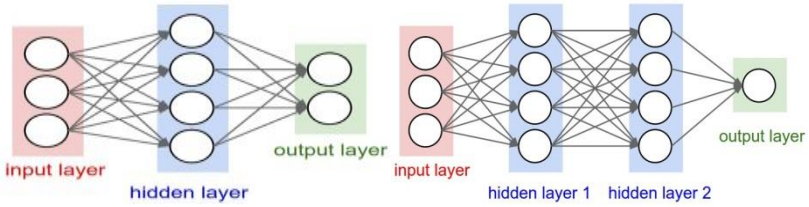

Imagen 2.1: Diferencia entre red neuronal convencional y redes neuronales profundas

Una red neuronal es una combinación de neuronas artificiales. Las neuronas se agrupan típicamente en capas. En una red multicapa de alimentación directa totalmente conectada, que se muestra en la Figura 2.2, cada salida de una capa de neuronas se alimenta como entrada a cada neurona de la siguiente capa. Así, algunas capas procesan los datos de entrada originales, mientras que otros procesan datos recibidos de otras neuronas. Cada neurona tiene un número de pesos igual al número de neuronas en la capa anterior [32].

En una red de múltiples capas de entrada generalmente solo pasa los datos sin modificarlos. La mayor parte del cálculo ocurre en las capas intermedias. La capa de salida convierte las activaciones de las capas ocultas en una salida. Una red de avance de múltiples capas con al menos una capa oculta puede funcionar como un aproximador universal, es decir, puede construirse para calcular casi cualquier función [33].

2.2.2 Propagación hacia atrás

Una red neuronal se entrena seleccionando los pesos de todas las neuronas para que la red aprenda a aproximarse a las salidas objetivo de las entradas conocidas. Resulta difícil resolver analíticamente los pesos neuronales de una red multicapa. El algoritmo de propagación hacia atrás proporciona una solución simple y efectiva para resolver los

pesos de manera iterativa. La versión clásica utiliza el descenso de degradado como método de optimización. El descenso de gradiente puede llevar bastante tiempo y no se garantiza que encuentre el mínimo global de error, pero con una configuración adecuada de hiperparámetros funciona lo suficientemente bien en la práctica [34].

En la primera fase del algoritmo, un vector de entrada se propaga hacia adelante a través de la red neuronal. Antes de esto, los pesos de las neuronas de la red se habían inicializado a algunos valores, por ejemplo, valores aleatorios pequeños. La salida recibida de la red se compara con la salida deseada, utilizando una función de pérdida. Luego se calcula el gradiente de la función de pérdida. Este gradiente también se llama el valor de error. Cuando se utiliza el error cuadrático medio como función de pérdida, el valor de error de la capa de salida es simplemente la diferencia entre la salida actual y la deseada [35].

Los valores de error se propagan luego a través de la red para calcular los valores de error de las neuronas de la capa oculta. Los gradientes de la función de pérdida neuronal oculta se pueden resolver utilizando la regla de la cadena de derivados. Finalmente, los pesos neuronales se actualizan calculando el gradiente de los pesos y restando una proporción del gradiente de los pesos. Esta relación se llama tasa de aprendizaje. La tasa de aprendizaje puede ser fija o dinámica. Una vez actualizados los pesos, el algoritmo continúa ejecutando las fases nuevamente con diferentes entradas hasta que los pesos converjan [36].

2.2.3 Tipos de funciones de activación

La función de activación φ determina la salida final de cada neurona. Es importante seleccionar la función correctamente para crear una red efectiva. Los primeros investigadores descubrieron que los perceptrones y otros sistemas lineales tenían cada vez más inconvenientes, ya que no podían resolver problemas que no fueran linealmente separables. A veces, los sistemas lineales pueden resolver este tipo de problemas utilizando detectores de características hechos a mano, pero este no es el uso más conveniente del aprendizaje automático. El simple hecho de agregar capas tampoco ayuda, porque una red compuesta de neuronas lineales sigue siendo lineal sin importar cuántas capas tenga [27].

Una forma ligera y efectiva de crear una red no lineal es utilizando unidades lineales rectificadas (*ReLu*) [27]. Una función lineal rectificada genera la salida utilizando una función de rampa como:

$$\varphi\ s\ = \max(0, s) \tag{2}$$

Este tipo de función es fácil de calcular y diferenciar para aplicar propagación hacia atrás. La función no es diferenciable en cero, pero esto no ha impedido su uso en la práctica. La función de activación *ReLu* se han vuelto muy populares últimamente, a menudo reemplazando las funciones de activación *sigmoide*, que tienen derivados suaves pero sufren problemas de saturación de gradiente y cálculos más lentos.

Para problemas de clasificación de clases múltiples, la función de activación *Softmax* [26], es la más utilizada, se usa normalmente en la capa de salida de la red:

$$\varphi\ s\ = \frac{e^{s_k}}{\sum_k^K e^{s_k}} \qquad para\ k = 1, \ldots, \ldots, K \tag{3}$$

La función *Softmax* toma un vector de k valores arbitrariamente grandes y genera un vector de valores K que oscilan entre 0.... 1 y suma a 1. Los valores generados por la unidad *Softmax* pueden utilizarse como probabilidades de clase.

2.3 Visión Por Computador

A continuación, se establecerá el fundamento teórico relacionado a la visión por computadora siendo esta la que se encarga de utilizar el modelo entrenado para realizar un proceso de detección de objetos, que para los objetivos del presente trabajo de investigación se enfoca en la identificación de plantas invasivas en imagen.

La visión artificial se ocupa de la extracción de información significativa de los contenidos de imágenes digitales o video. Esto es distinto del simple procesamiento de imágenes, que implica manipular información visual en el nivel de píxeles. Las aplicaciones de la visión por ordenador incluyen la clasificación de imágenes, la detección visual, la reconstrucción de escenas en 3D a partir de imágenes en 2D, la recuperación de imágenes, la realidad aumentada, la visión artificial y la automatización del tráfico [37]. Hoy en día, el aprendizaje automático es un componente necesario de muchos algorit-

mos de visión artificial [38]. Dichos algoritmos pueden describirse como una combinación de procesamiento de imágenes y aprendizaje automático. Las soluciones efectivas requieren algoritmos que puedan hacer frente a la gran cantidad de información contenida en imágenes visuales, esta característica sirve para muchas aplicaciones de identificación y clasificación en el mundo real [39].

2.3.1 Detección de objetos

La detección de objetos es uno de los problemas clásicos de la visión artificial y, a menudo, se describe como una tarea difícil. En muchos aspectos, es similar a otras tareas de visión artificial, ya que implica crear una solución invariable a la deformación y cambios en la iluminación y el punto de vista. Lo que hace que la detección de objetos sea un problema distinto es que implica la localización y clasificación de regiones de una imagen [40]. La parte de localización no es necesaria, por ejemplo, en la clasificación de la imagen completa. Para el desarrollo de nuestro algoritmo de identificación no se profundizará en localización dentro de la imagen, solo en identificar si la una planta invasiva se encuentra en una imagen completa.

Para detectar un objeto, necesitamos tener una idea de dónde podría estar el objeto y cómo se segmenta la imagen. Esto crea una problemática para reconocer la forma y la clase de un objeto, necesitamos conocer su ubicación, y para reconocer la ubicación de un objeto, necesitamos conocer su forma. Algunas características visualmente complejas, como es el caso de hojas de una planta o la cara de un ser humano, pueden ser partes del mismo objeto, lo anterior resalta la dificultad de identificación, si primero no se reconocer el objeto completo. Por otro lado, algunos objetos se destacan solo ligeramente del fondo, lo que requiere una separación antes del reconocimiento. [41].

2.4 Redes neuronales Convolucionales

El problema de resolver problemas de visión por computadora utilizando redes neuronales tradicionales es que incluso una imagen de tamaño modesto contiene una enorme cantidad de información. Una imagen monocroma 620x480 contiene 297 600 píxeles. Si cada intensidad de píxel de esta imagen se ingresa por separado a una red totalmente conectada, cada neurona requiere 297 600 pesos. Una imagen Full HD de 1920x1080 requeriría 2.073.600 pesos. Si las imágenes son policromadas, la cantidad de pesos se

multiplica por la cantidad de canales de color (generalmente tres RGB). Por lo tanto, podemos ver que el número total de parámetros libres en la red rápidamente se vuelve extremadamente grande a medida que aumenta el tamaño de la imagen. Los modelos demasiado grandes causan un ajuste excesivo y un rendimiento lento [26].

Además, muchas tareas de detección de patrones requieren que la solución sea invariable para la traducción. Es ineficiente entrenar a las neuronas para que reconozcan por separado el mismo patrón en la esquina superior izquierda y en la esquina inferior derecha de una imagen. Una red neuronal totalmente conectada no tiene en cuenta este tipo de estructura [42].

2.4.1 Arquitectura de una CNN

Las redes neuronales convolucionales (CNN) reemplazan las capas de neuronas con capas de convolución de detección de características (inspiradas biológicamente por la organización del campo visual) [43]. Una CNN está compuesta de capas, cada capa tiene una función simple: transforma un volumen 3D de entrada en un volumen 3D de salida con alguna función diferenciable que puede tener o no parámetros.

Una CNN simple es una secuencia de capas, y cada capa de una CNN transforma un volumen de activaciones en otra a través de una función diferenciable. Utilizamos cuatro tipos principales de capas para construir una arquitectura CNN:

- Capa convolucional (*Conv*).
- Capa agrupación (*Pool*)
- Capa de activación (*ReLU*)
- Capa totalmente conectada (*Dense*).

La arquitectura CNN que opera en una imagen realiza dos procesos principales, El primer proceso se realizan extracción de características utilizando operaciones alternas de convolución (*conv*) y agrupación (*pool*). Cada capa convolutiva extrae automáticamente características útiles, como bordes o esquinas, y genera una serie de mapas de características. Las operaciones de agrupamiento (*pool*) reducen el tamaño de los mapas de características para mejorar la eficiencia. El número de mapas de características se incrementa conforme se avanza en las capas de la red CNN para mejorar la precisión de la clasificación [44].

El segundo proceso utiliza capas de redes neuronales convencionales (*ANN*) comprenden las capas de clasificación totalmente conectadas (*dense*), que generan probabilidades para cada clase.

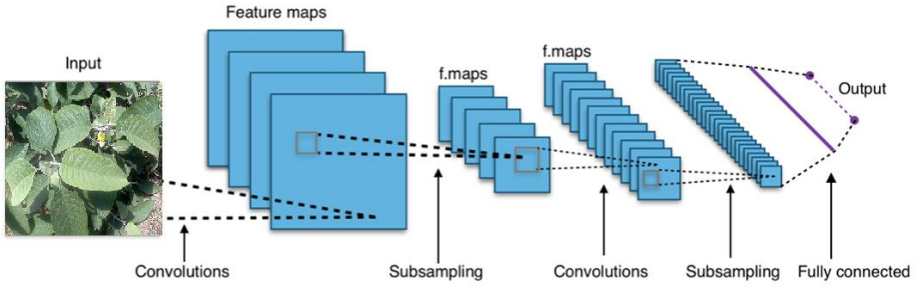

Imagen 2.2: Estructura general de un modelo CNN

En sus inicios el costo computacional impedía la implementación de redes neuronales de convolución, en la actualidad los algoritmos de vanguardia y el hardware computacional han reducido el tiempo de capacitación para dichas redes a niveles prácticos que se pueden desarrollar aplicaciones en la mayoría de los laboratorios de investigación. Debido a esto las redes CNN han sido rápidamente adoptadas por la comunidad de visión por computadora y diversas áreas de la ingeniería recientemente se han utilizado con éxito en las ciencias de la vida [45] y la medicina [46].

2.4.2 Imagen Digital (Input)

La característica principal de las redes neuronales convolucionales es que la entradas de la red son imágenes. En particular, a diferencia de una red neuronal normal, las capas de un CNN tienen neuronas conformadas por 3 dimensiones: ancho, alto y profundidad. (La dimensión profundidad se refiere los canales de color de la imagen puede ser un canal en escala de grises o tres canales para imágenes a color RGB).

Antes de adentrarnos al concepto de imagen de entrada de una red neuronal conveniente repasar algunas ideas clave relacionadas con la imagen digital.

Una imagen digital es un arreglo rectangular de pixeles donde el píxel es la unidad mínima de visualización de una imagen digital. Si aplicamos el zoom sobre una imagen observaremos que está formada por una parrilla de puntos o píxeles. Las cámaras digitales y los escáneres capturan las imágenes en forma de cuadrícula de píxeles.

La profundidad de color se refiere al número de bits necesarios para codificar y guardar la información de color de cada píxel en una imagen. Un bit es una posición de memoria que puede tener el valor 0 o 1. Cuanto mayor sea la profundidad de color en bits, la imagen dispondrá de una paleta de colores más amplia.

La mayoría de los píxeles están representados de dos maneras:

- Imagen a escala de grises (1 canal).
- Imagen a Color (RGB 3 canales).

En una imagen en escala de grises, cada píxel es un valor escalar entre 0 y 255, donde cero corresponde a negro y 255 representa el color blanco. Los valores entre 0 y 255 varían en los tonos de gris, donde los valores más cercanos a 0 son más oscuros y los valores más cercanos a 255 son más claros.

Imagen 2.3: Imagen a escala de grises (1 canal [0-255])

Los píxeles en el espacio de color RGB ya no son un valor escalar como en una imagen de escala de grises y de canal único, en cambio, los píxeles están representados por una lista de tres valores, el primero es un valor para el componente Rojo, uno para Verde y otro para Azul. Para definir un color en el modelo de color RGB, todo lo que tenemos que hacer es definir la cantidad de rojo, verde y azul que contiene un solo píxel.

Imagen 2.4: Figura a color (3 canales [RGB])

Cada canal Rojo, Verde y Azul puede tener valores definidos en el rango [0:255] para un total de 256 "sombras", donde 0 indica que no hay representación y 255 demuestra

representación completa. Dado que el valor de píxel solo necesita estar en el rango [0:255], normalmente utilizamos enteros de 8 bits sin signo para representar la intensidad. Por ejemplo, una imagen de entrada con dimensiones de 50x50x3 (ancho, alto, profundidad respectivamente) representa una imagen con ancho de 50 pixeles por una altura de 50 pixeles y profundidad de tres indica que es una imagen a color con canales RGB.

2.4.3 Capa de Convolución

La idea básica de la CNN se inspiró en un concepto en biología denominado campo receptivo [47]. Los campos receptivos son una característica de la corteza visual animal [48]. Actúan como detectores que son sensibles a ciertos tipos de estímulos, por ejemplo, los bordes. Se encuentran en todo el campo visual y se superponen entre sí.

Esta función biológica se puede aproximar en computadoras que utilizan la operación de convolución [49]. En el procesamiento de imágenes, las imágenes se pueden filtrar utilizando la convolución para producir diferentes efectos visibles como la detección de los bordes horizontales de una imagen, funcionando de manera similar a un campo receptivo. En su forma más general, la convolución es una operación en dos funciones de un argumento de valor real. La terminología de una red convolucional se representa por dos argumento principales, la entrada, y el segundo es el núcleo (Kernel). La salida se define como, mapa de características.

La operación de convolución discreta entre una imagen f y una matriz de filtro g se define como:

$$h[x,y] = f[x,y] * g[x,y] = \sum_{n} \sum_{m} f[n,m]\ g[x-n, y-m] \tag{4}$$

La convolución del filtro g y una sub-imagen de f con las mismas dimensiones que g) se centran en las coordenadas$[x,y]$, y produce el valor de píxel de h en las coordenadas $[x,y]$. El tamaño del campo receptivo se ajusta según el tamaño de la matriz del filtro. Alinear el filtro sucesivamente con cada sub-imagen $[x,y]$, produce la matriz de píxeles de salida $h[x,y]$. En el caso de las redes neuronales, la matriz de salida también se denomina mapa de características o mapa de activación después de calcular la función de

activación. Los bordes deben tratarse como un caso especial. Si la imagen $[x, y]$ no se rellena, el tamaño de salida disminuye ligeramente con cada convolución [27].

Haciendo la analogía del concepto matemático con relación al proceso de convolución de una CNN, se evalúa la cantidad de superposición de una imagen de entrada con un filtro definido. La capa de convolución es la componente básico de una CNN que realiza la parte más importante en el proceso de extracción de características y además es la que realiza el mayor gasto computacional [50]. Los parámetros de esta capa consisten en un conjunto de filtros que se pueden aprender. Cada filtro es pequeño espacialmente (a lo ancho y alto), pero se extiende a través de la profundidad total del volumen de entrada. Por ejemplo, un filtro típico en una primera capa de una CNN puede tener un tamaño de (30x30x3) (es decir, 30 píxeles de ancho y alto, y el numero 3 porque las imágenes tienen una profundidad 3 correspondiente a los canales de color RGB).Una forma de ver el proceso es que se desliza el *kernel* superponiendo cada filtro a través del ancho y alto del volumen de entrada y calculamos los productos de puntos entre las entradas del filtro y la entrada en cualquier posición [51].

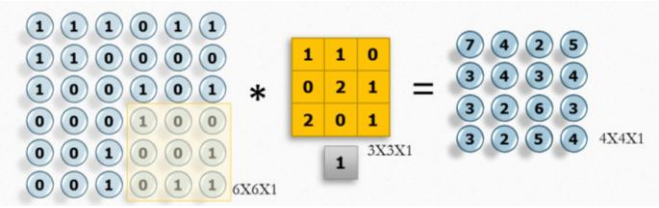

Imagen 2.5: Arquitectura de una capa de convolución en una CNN

A medida que deslizamos el filtro sobre el ancho y la altura del volumen de entrada se genera un mapa de activación bidimensional que proporciona las respuestas de ese filtro en cada posición espacial. Intuitivamente, la red aprenderá los filtros que se activan cuando ven algún tipo de característica visual, como un borde o una mancha de algún color en la primera capa. Eventualmente conforme el proceso avanza por las capas de convolución la red pode aprender filtros que se activan para reconocer objetos más complejos como puede ser la hoja de una planta, animales como gatos o patrones parecidos a ruedas de bicicletas entre otros. Al concluir el proceso, se tendrá un conjunto complejo de filtros en cada capa de convolucion, y cada uno de ellos producirá un mapa de activación bidimensional por separado. Finalmente se apilan estos mapas de activación a lo largo de la dimensión de profundidad y produciremos el volumen de salida [52].

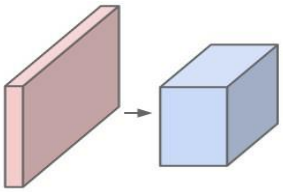

Imagen 2.6: Representación de imagen de entrada y de salida en formato de volumen

En base a lo anterior se definen tres características de una capa convolucional enlistadas a continuación [52]:

Escasa conectividad: Cuando tratamos con entradas como imágenes de grandes dimensiones, no es práctico conectar las neuronas a todas las neuronas en el volumen anterior debido a que involucra un gasto computacional grande y un incremento en el tiempo de procesamiento. En cambio, en una CNN se conecta cada neurona a solo una región local del volumen de entrada. La extensión espacial de esta conectividad es un híperparámetro llamado campo receptivo de la neurona. La extensión de la conectividad a lo largo del eje de profundidad siempre es igual a la profundidad del volumen de entrada.

Parámetros compartidos: Al obtener volumen de salida 3D resultante del proceso de convolución también puede interpretarse como una salida de una neurona que solo mira una pequeña región en la imagen de entrada por lo tanto comparte parámetros con todas las neuronas a la izquierda y derecho espacialmente y esto se debe a que todos estos números resultan de aplicar el mismo filtro en el proceso de convolución.

Representación equivalente: Hace referencia a la relación de las características de una imagen de entrada que pueden ser aprendidas por una CNN, si la imagen de entrada es modificado por consecuencia todos los mapas de características serán diferentes.

2.4.4 Partes de una capa convolucional

Para que se pueda realizar un proceso de convolución es importante describir las partes y parámetros involucrados en el proceso.

Kernel: Anteriormente se definió como el filtro que se convoluciona con la imagen de entrada, sin embargo formalmente cuando se estudia CNN se define como *Kernel*. El

Kernel es una imagen con dimensiones pequeñas (3x3, 5x5 etc.) comparadas a la de la imagen de entrada y representan las características que se desean resaltar en nuestra imagen de entrada en el proceso de clasificación. En términos de redes neuronales los *Kernels* representan el conjunto de Pesos y *Bias* de la red, pero es importante resaltar que a diferencia de las redes neuronales convencionales en una CNN estos son fijos ya que de forma visual un *Kernel* puede ser una porción de imagen que contenga simples bordes o manchas de color características de objetos e incluso objetos completos. El conjunto de todos los *Kernels* conforman las características que pueden clasificar algún objeto de interés y en este trabajo de investigación el objeto de interés son características de plantas [52].

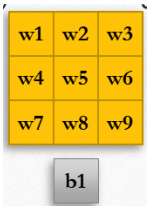

Imagen 2.7: Representación matricial de un kernel en una CNN

Para definir cuántas neuronas hay en el volumen de salida o cómo están organizadas existen tres hiperparámetros que controlan el tamaño del volumen de salida:

La profundidad del volumen de salida corresponde a la cantidad de filtros que se utilizan en el proceso de convolución, cada uno aprendiendo a buscar algo diferente en la imagen de entrada. Por ejemplo, si la primera capa convolucional toma como entrada la imagen en bruto, entonces diferentes neuronas a lo largo de la dimensión de profundidad pueden activarse en presencia de varios bordes, manchas de color orientadas en la imagen. En otras palabras nos referiremos a un conjunto de neuronas que miran la misma región de la entrada que una columna de profundidad.

Zancada: La zancada con la que deslizamos el filtro por la imagen cuando se convoluciona. Cuando la zancada es 1, movemos los filtros un pixel a la vez de forma horizontal hasta realizar un barrido completo de izquierda a derecha y de arriba hacia abajo por toda la imagen. [52]

Relleno cero: Es un hiperparámetro que permite controlar el tamaño espacial de los volúmenes de salida, es decir cuando se desliza el *kernel* por la imagen de entrada y las dimensiones no son compatibles en proporción de pixeles hay algunas operaciones de convolución donde partes del *kernel* no pueden multiplicarse con parámetros de la ima-

gen de entrada por lo que se hace un ajuste de la imagen de salida aplicando un relleno con números, en otras palabras se utiliza para preservar exactamente el tamaño espacial del volumen de entrada para que el ancho de entrada y salida y de igual forma la altura es la misma. [52]

2.4.5 Capa de Pooling

Se considera como una etapa de convolución la deferencia directa que tienen es que el *kernel* que se utiliza para un proceso de pool es un *kernel* vacío sin parámetros, Su función mejorar la eficiencia de la *CNN* reduciendo progresivamente el tamaño espacial del Mapa de Características para reducir la cantidad de parámetros del cálculo en la *CNN* y, por lo tanto, también controlar el sobreajuste. La capa *Pooling* opera independientemente en cada sector de profundidad de la entrada y lo cambia de tamaño espacialmente, usando la operación *Maxpool*. La forma más común es una capa de agrupamiento con filtros de tamaño 2x2 aplicados con una zancada de 2 muestras descendentes cada corte de profundidad en la entrada por 2 a lo largo de ancho y alto, descartando el 75% de las activaciones. Cada operación Maxpool en este caso tomaría un máximo de 4 números (pequeña región de 2x2 en algún segmento de profundidad). La dimensión de profundidad permanece sin cambios. También se puede definir a esta capa como capa de agrupación [52].

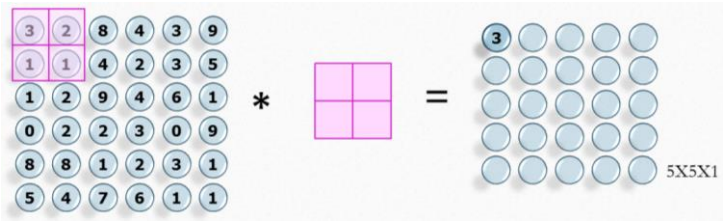

Imagen 2.8: Representación matricial de una operación Pooling en una CNN

La capa *Pooling* cuenta con dos parámetros principales:

1. Extensión espacial del kernel vacío.
2. Zancada

Finalmente la capa *pooling* ayuda a que la representación del modelo sea invariante a pequeños movimientos de la entrada.

2.4.6 Activación

Después de cada capa convolucional en una CNN, aplicamos una función de activación no lineal, como La Unidad Lineal Rectificada *ReLU*. Las capas de activación no son técnicamente "capas" debido a que no aprenden parámetros o pesos al aplicar la función de activación, algunas veces se omiten de los diagramas de arquitectura de red CNN ya que se supone que una activación sigue inmediatamente a una convolución.

Por lo general, denominamos capas de activación como *ReLU* en diagramas de red ya que las funsion de activacion (*ReLU*) se ha vuelto muy popular en los últimos años. Calcula la función $f\ x\ = \max(0,1)$[52].

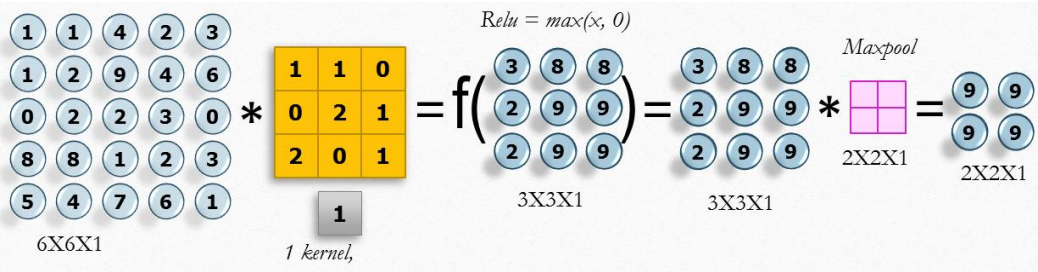

Imagen 2.9: Partes principales de la arquitectura de una CNN

Las capas ocultas finales de una CNN suelen ser capas totalmente conectadas. Una capa totalmente conectada puede capturar algunas relaciones interesantes que no pueden compartir las capas convolucionales que comparten parámetros. Sin embargo, una capa completamente conectada requiere un tamaño de volumen de datos suficientemente pequeño para que sea práctico. Las configuraciones de agrupación y zancada pueden usarse para reducir el tamaño del volumen de datos que llega a las capas totalmente conectadas. Una red convolucional que no incluye ninguna capa totalmente conectada, se llama una red completamente convolucional (FCN) [53].

CAPÍTULO III

3 CONFIGURACIÓN INICIAL

En este capítulo, comenzamos a discutir la parte experimental de la tesis. Primero, discutiremos los criterios de selección de métodos, luego describiremos los métodos seleccionados, sus parámetros y los conjuntos de datos seleccionados. Finalmente, discutiremos postproceso y evaluación de un modelo CNN inicial preliminar. La implementación de los métodos experimentales para optimizar el modelo inicial se discutirá en el siguiente capítulo. Sin embargo, algunos detalles de la implementación también se tratan en este capítulo, ya que influyen en la selección del método CNN aplicado a conjuntos de datos pequeños.

La idea que surge al querer implementar una Redes Neuronales Convolucionales para monitoreo y detección de plantas invasivas utilizando datos de tipo imagen, es la necesidad de grandes cantidades de imágenes que caractericen el objete que se desea clasificar, esto representa una complicación si no se cuenta con bases de datos que generalicen el objeto que se desea identificar, esto generando un problema común para investigadores de áreas diferentes a las ciencias de la computación. Pensar en desarrollar una clasificación utilizando CNN teniendo a disposición pocos datos condujo a los ingenieros en computación a desarrollar software y aplicaciones que faciliten la comprensión y aplicación de redes neuronales incluso para áreas de investigación que no cuentan con recursos computacionales e información necesaria.

Debido al aumento en popularidad y a la diversidad de aplicaciones de las redes neuronales, han surgido programas especializados para CNN que facilitan su comprensión y aplicación, incluso si se cuentas con poca experiencia en el tema. *Keras* es uno de los programas más populares para el estudio de Redes Neuronales y ofrece soluciones para el desarrollo de clasificadores de imagen con conjuntos de datos pequeños proporcionando paqueterías y funciones que adecuan el conjunto de datos para el desarrollo de modelos CNN precisos.

3.1 Modelo CNN inicial

La presente investigación busca implementar un algoritmo de clasificación para la identificación de plantas invasivas en imágenes capturadas por los distintos métodos de monitoreo mencionados en la introducción de esta documento. Para su desarrollo se requiere un conjunto de datos de imagen de dos tipos, el primero tipo debe contener la planta invasiva que se desea detectar en alguna parte de la imagen, y el segundo tipo contiene vegetación en general correspondiente al entorno donde comúnmente se encuentra la planta invasiva.

La paquetería *Keras* fue diseñado para permitir una rápida experimentación con redes neuronales profundas, se enfoca en ser fácil de usar, modular y extensible. De una forma particular y enfocada a la problemática de la presente investigación. *Keras* proporciona diversas funciones que con "pocas" muestras de un conjunto pequeño de imágenes se puede generar cientos o hasta algunas decenas de miles de imágenes permitiendo realizar aplicaciones CNN precisas.

La implementara del algoritmo de monitoreo e identificación de plantas invasivas en imagen se desarrollara realizando las siguientes tareas:

1. Descargara un conjunto de imágenes existente de flores invasivas.
2. Acondicionamiento de las imágenes del conjunto de datos previo al entrenamiento.
3. Proponer arquitectura del modelo (utilizando capas de *convolución*, *activación*, *pooling, dense y activación final*).
4. Entrenar el modelo CNN.

La implementación de esta CNN inicial, se modificara a lo largo de la presente investigación, modificando parámetros que permitan la optimización del modele, y que permitan obtener resultados satisfactorios en procesos de identificación de plantas invasivas para conjuntos de datos pequeños.

3.1.1 Conjunto de dato

El conjunto de datos se obtuvo de un desafío *Kaggle* para la detección de plantas invasivas llamado Invasive Species Monitoring. Kaggle es un sitio web que fomenta el desarrollo de aplicaciones de análisis de datos desarrollando competencias entre sus seguidores. En muchos de estos retos las CNN han demostrado ofrecer los mejores resultados, como es el clásico clasificador de Perro y Gato o el clasificador de plantas invasivas *Invasive Species Monitoring*, donde los desarrollador que utilizaron CNN han obtenido los resultados más sobresaliente. El conjunto de datos está integrado por imágenes de flores de hortensia e imágenes de naturaleza en general. El desafío consiste en el desarrollo de algoritmos que puedan identificar con mayor precisión si las imágenes de bosques o follaje contienen hortensias invasoras o no. El presente trabajo no se enfoca en competir en el desafío sino en desarrollar un modelo CNN desde cero que pueda utilizarse para detección de plantas invasivas utilizando *Jupyter Notebook*, *Python, Keras* y *OpenCV*. El conjunto de datos de imágenes es de libre descarga para cuestiones de investigación o para entrar en la competencia.

Este conjunto de datos está conformado por 3,826 imágenes de hortensia, bosque y follaje (1,913 de cada clase), con un total de 2.8 GB. Se realizó una selección y agrupación de las imágenes creando un nuevo conjunto de datos que contiene tres subconjuntos: un conjunto de entrenamiento con 1,000 muestras de cada clase, un conjunto de validación con 500 muestras de cada clase y un conjunto de pruebas con 250 muestras de cada clase. Este es un problema equilibrado de clasificación binaria, donde solo se tienen dos posibles resultados detección de planta invasiva o no.

La mayoría de las imágenes positivas del conjunto de datos tienen arbustos de hortensias de fotograma completo. También hay una cantidad importante de imágenes que tienen solo pequeñas porciones de la imagen ocupadas por hortensias. Las imágenes negativas son principalmente escenas de bosques sin hortensias, se puede ver un ejemplo de cada clase en la Imagen 3.1.

Imagen 3.1: Imagen hortensia (izquierda) y bosque o follaje (derecha)

Algunas de las imágenes del conjunto de datos contienen otras características como casas, caballos, cobertizos o personas sin embargo se tomaran como parte del entorno enfocando la atención en la detección de la flor de hortensia o ausencia de flor.

3.1.2 Pre-procesamiento de datos

Las imágenes del conjunto de datos tienen un tamaño de 866x1154 píxeles, sin embargo para el proceso de entrenamiento, utilizar imágenes de estas dimensiones re-presenta un gran gasto computacional por lo que es recomendable aplicar funciones que acondicionen las imágenes previas al proceso de entrenamiento utilizando la función *ImageDataGenerator* de *Keras*.

Las imágenes se encuentran con formato JPEG que se codifica en una cuadricula de pixeles con diferentes valores de intensidad. Sin embargo estas deben ser pre-procesadas adecuadamente en tensores de punto flotante antes de entrenar la CNN. La siguiente lista muestra los pasos para el acondicionar del conjunto de imágenes:

1. Lee los archivos de imagen JPEG.
2. Decodifique el contenido JPEG en cuadrículas RGB de píxeles.
3. Convierta estos en tensores de punto flotante.
4. Normalizar los valores de píxel (entre 0 y 255) al intervalo [0, 1], esto se debe a que las redes neuronales prefieren tratar con pequeños valores de entrada.

El software *Keras* cuenta con un módulo con herramientas de ayuda para el procesamiento de imágenes, ubicado en *keras.preprocessing.image*. Este módulo cuenta con

utilidades que se encargan automáticamente de realizar los pasos enlistados anteriormente. En particular, contiene la clase *ImageDataGenerator*, que le permite configurar utilizando operadores en lenguaje *Python* que convierten automáticamente los archivos de imagen en el disco en lotes de tensores pre-procesados.

Código 3.1 Uso de ImageDataGenerator para leer imágenes de directorios.

```
# Setting Image and Data Generators
train_idg = ImageDataGenerator(rescale=1. / 255,
                                        zoom_range=[1.0, 1.25],
                                        width_shift_range=0.1,
                                        height_shift_range=0.1,
                                        fill_mode='reflect')

train_g = train_idg.flow_from_directory(directory=r'data2\train',
                                                target_size=(120, 120),
                                                class_mode='binary',
                                                batch_size=100,shuffle=True)
```

En el Código 3.1 muestran dos bloques con la misma estructura, el ´primero acondiciona el subconjunto de imágenes de entrenamiento y el segundo acondiciona el subconjunto de imágenes de validación. La primera línea de código de ambos bloques se le asigna a una variable la función *ImageDataGenerator* con el parámetro *rescale* que normaliza los valores de intensidad de una imagen de [0-255] a valores entre [0-1]. La segunda línea de código aplica los parámetros de la función *ImageDataGene-rator* previamente definidos a un directorio especifico que contiene el subconjunto de imágenes que se desean acondicionar, el parámetro *target_size* redimensiona las imágenes de [866x1154] a [120x120], *clasmode* indica el número de clases que se utilizaran, en la presente implementación serán dos [hortensia o naturaleza] y finalmente *batch_size* indica que se aplicaran estos parámetros a las imágenes en lotes de 20 imágenes.

Este generador acondiciona lotes de imágenes RGB de 120 × 120x 3 en el formato siguiente (forma (20, 150, 150, 3)) donde el primer valor representa el tamaño del lote de 20 muestras por lote, el segundo y tercer valor representa las dimensiones de la ima-

gen y el ultimo valor representa los canales de la imagen donde el número tres representa que es una imagen a color.

3.1.3 Arquitectura de la CNN

La arquitectura de una CNN está conformada por un conjunto de capas alternadas de tipo Conv2D, MaxPooling2D y activación *ReLU* para la extracción de características, para la clasificación utilizan capas de tipo *Flatten*, Dense y activación *Sigmoid*. Estas capas contienen parámetros que indican la forma en que las imágenes pasaran atreves de la red durante los procesos de extracción de características y el proceso de clasificación. En las capas convolucionales se establecen *kernels* con dimensión 3x3 y en cada capa convolucional se obtienen mapas de características resultantes, 32 filtros en la primer capa convolucional, 64 en la segunda y tercer capa y en la última capa un total de 128 filtros. Las capas de activación utilizada en la extracción de características es la función *ReLU* [f(x)=max(0,x)], y finalmente las capas de redimensionamiento *pooling* se utilizan *kernels* vacíos de 3x3 que tienen la finalidad de reducir los mapas de características resultantes.

En la parte de clasificación se utilizaron tres clases de capas. La capa *flatten* se utiliza para convertir el arreglo matricial resultante en un arreglo vectorial unidimensional que pueda ser procesada por una red neuronal convencional, seguido de dos capas Dense totalmente conectadas de 100 nodos y 1 nodo respectivamente. Final-mente, como se está tratando un problema de clasificación binaria, la red CNN ter-minará con una sola unidad (una capa Densa de tamaño 1) y una activación *Sigmoid*. Esta unidad codificará la probabilidad de que la red esté mirando una clase u otra. La arquitectura antes descrita se puede apreciar en la Imagen 3.2 como un diagrama a bloques que representan las capas incluidas en este modelo CNN inicial.

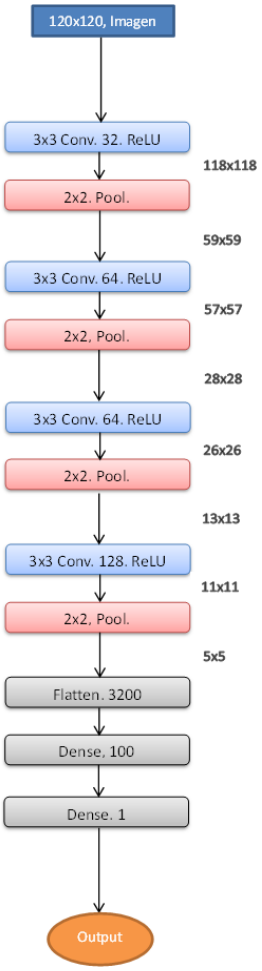

Imagen 3.2: Diagrama a bloques de la arquitectura del modelo CNN inicial

La arquitectura inicial de nuestra CNN muestra una secuencia de capas consecutivas con las siguientes características a resaltar. Se redimensionan las imágenes de entrada [866x1154] a [120x120]. Otro aspecto importante de la secuencia de capas es la de aumentar la capacidad de la red así como reducir el tamaño de los mapas de características, de modo que no sean demasiado grandes cuando llegue a la capa *Flatten* que lo convertirá en un vector unidimensional de valores.

Se inicia con imágenes de entrada con un tamaño de 120 × 120, terminas con mapas de características de tamaño 5 × 5 seguido de una conversión del arreglo 2D de [5x5] en un vector 1D utilizando la capa *Flatten*. La profundidad de los mapas de características aumenta progresivamente en la red (de 32 a 128). Este es un patrón que se puede observar en todas las CNN, se inicia con una imagen de dimensión establecida y se reduce progresivamente con la finalidad de resaltar las características más representativas del conjunto de datos. Esta arquitectura representa el primer acerca-miento a la implementación de la CNN para la detección de plantas invasivas,

3.1.4 Entrenamiento

El entrenamiento es un proceso donde el modelo aprende características de las imágenes de entrenamiento a través de las capas de convolución de la red neuronal. Un conjunto de mapas de características puede resaltar desde bordes simples de las hojas de la flor de hortensia, hojas completas, conjuntos de hojas y finalmente considerar la planta por completo.

Antes de entrenar nuestro modelo inicial, se debe configurar el proceso de aprendizaje, utilizando el método de compilación (compile). Este proceso de aprendizaje recibe tres argumentos principales:

1. Un optimizador. Este podría ser el identificador de cadena de un optimizador existente.
2. Una función de pérdida. El modelo intentará minimizar este parámetro ya que indica la desviación del modelo. Puede ser el identificador de cadena de una función de pérdida existente (como *categori_crossentropy*), o puede ser una función objetivo.
3. Una lista de métricas. Para cualquier problema de clasificación en una CNN, se evalúan dos métricas principales las cuales son precisión (*accuracy*) y perdida (*loss*).

Código 3.2 Método de optimización y función de perdida.

```
# Model Loss function and Optimizer method
compile=my_model.compile(optimizer=optimizers.sgd(lr=0.15),
                                    loss='binary_crossentropy',metrics=['accuracy'])
```

Con los parámetros de aprendizaje establecidos pasamos al proceso de entrenamiento utilizando el método *fit_generator*, este método utiliza al generador utilizado en el acondicionamiento de las imágenes (*ImageDataGenerator*). Utilizando el generador de imágenes acondicionadas se producen lotes de entradas, debido a que los datos se están generando infinitamente, el argumento *steps_per_epoch* indica cuántas mues-tras ex-

traer del generador antes de declarar una época. Terminando el total de imágenes del lote, el proceso pasa a una nueva época. En este caso, se definieron lotes de 20 muestras, por lo que la función *fit_generator* tomará 100 lotes de 20 imágenes hasta que alcance su objetivo de 2000 muestras.

Código 3.3 Opciones de entrenamiento.

```
#Training Options
history =my_model.fit_generator(generator=train_g,steps_per_epoch=20,
                                 epochs=100, validation_data=valid_g,
                                 validation_steps=50)
```

En el proceso de acondicionamiento también se define un generador para las imagines de validación, para definir se usa el argumento *validation_data* asignado a un generador de datos de validación previamente definido. Es importante tener en cuenta que se permite que este argumento sea un generador de datos, pero también podría ser una *tupla* de matrices *Numpy*. Si pasa un generador como *validation_data*, se espera que este generador genere lotes de datos de validación sin fin; por lo tanto, también debe especificar el argumento *validation_steps*, que le indica al proceso cuántos lotes extraer del generador de validación para cada época de entrenamiento. Los lotes de imágenes se definieron de 20 imágenes y el número de lotes por época es de 50.

Con los parámetros de los generadores de entrenamiento y validación establecidos se puede proceder al proceso de entrenamiento indicando el número de veces que deseas que el modelo visualice los datos de entrenamiento y validación. Este parámetro es conocido como épocas (*epochs*).

```
Epoch 1/50
100/100 [==============================] - 156s 2s/step - loss: 0.6737 - acc: 0.5880 - val_loss: 0.6146 - val_acc: 0.6580
Epoch 2/50
100/100 [==============================] - 148s 1s/step - loss: 0.6190 - acc: 0.6735 - val_loss: 0.5831 - val_acc: 0.6900
Epoch 3/50
100/100 [==============================] - 147s 1s/step - loss: 0.5243 - acc: 0.7370 - val_loss: 0.5764 - val_acc: 0.7180
Epoch 4/50
100/100 [==============================] - 147s 1s/step - loss: 0.4600 - acc: 0.7860 - val_loss: 0.3840 - val_acc: 0.8530
Epoch 5/50
100/100 [==============================] - 146s 1s/step - loss: 0.3933 - acc: 0.8255 - val_loss: 0.3596 - val_acc: 0.8390
```

Imagen 3.3: Ejemplo del proceso de entrenamiento del modelo CNN inicial

La imagen anterior muestra 5 épocas del proceso de entrenamiento de 50 en total épocas, cada época se puede visualizar cómo evoluciona el proceso de entrenamiento para cada lote de imágenes del generador, al término de una época, se obtienen los

valores de precisión y pérdida correspondientes, que representan la precisión y la pérdida del modelo al pasar una época, para los datos de entrenamiento como para los datos de validación.

Al término del proceso de entrenamiento se guarda el modelo con formato (.h5), este archivo contiene los pesos en cada etapa de convolución y clasificación aprendidos durante el entrenamiento.

Los valores máximos de precisión y de perdida obtenidos en el proceso de entrenamiento se observan en la Tabla 3.1.

Tabla 3.1 Parámetros obtenidas en el proceso de entrenamiento del modelo inicial.

Métricas	**Train_acc**	**Train_loss**	**Val_acc**	**Val_loss**
	0.977 (97.7%)	0.021	0,918 (91.8%)	0.3535

La evolución de los parámetros de entrenamiento y validación durante 50 épocas se aprecian con mayor claridad en las gráficas de la Imagen 3.4. La gráfica de la izquierda muestra los parámetros de precisión (*acc*), para los datos de entrenamiento y para los datos de validación. La gráfica de la derecha muestra los parámetros de perdida (*loss*), para los datos de entrenamiento y para los datos de validación.

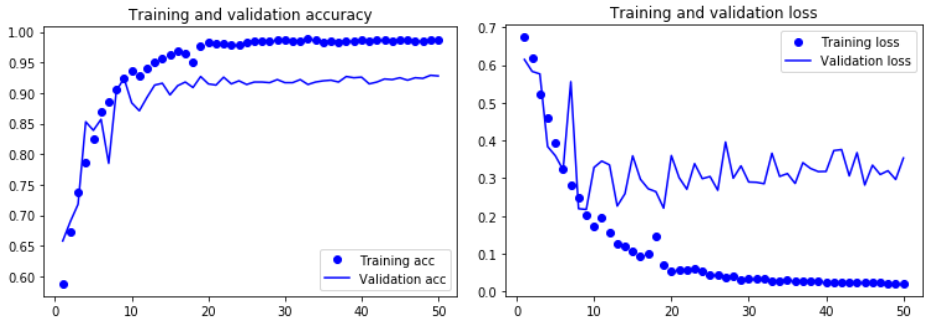

Imagen 3.4: Graficas de precisión y perdida del modelo CNN inicial entrenado durante 50 épocas

En las gráficas de la Imagen 3.4 se puede observar una separación (offset) de las curvas de precisión (*acc*) y en la curvas de perdida (*loss*), esta separación se define como sobre-entrenamiento (*Overfiting*). Este sobre-entrenamiento indica que el modelo tiene mejores condiciones para clasificar imágenes similares a los datos de entrenamiento y poca precisión para clasificar imágenes nunca antes vistos por el modelo. En las

gráficas se puede observar que la precisión de entrenamiento aumenta linealmente con el tiempo, hasta alcanzar casi el 100%, mientras que la precisión de validación se detiene en 85-90%. La pérdida de validación alcanza su mínimo después de solo diez épocas, mientras que la pérdida delos datos de entrenamiento sigue disminuyendo linealmente hasta alcanzar casi 0. El *Overffiting* se presenta de forma constante en modelos que tienen relativamente pocas muestras de entrenamiento en nuestro caso (2,000). Existen diversas técnicas para la reducir el sobre-entrenamiento, el primordial es incrementar los datos de entrenamiento y validación en conjunto te métodos como selección aleatoria de nodos (*Dropout*) y el decaimiento de pesos (*weight decay*). Para mitigar el *Overfitting* el presente trabajo se enfoca más en la utilización de la selección aleatoria de nodos y en el incremento de datos utilizando operaciones de transformación de imágenes existentes.

3.1.5 Pruebas de datos nunca antes vistos por el modelo

Evaluando 200 imágenes da datos nunca antes vistos por el modelo previamente entrenado se obtuvieron los siguientes resultados.

Tabla 3.2 Parámetros de precisión y perdida en imágenes nunca antes vistas por el modelo inicial.

# imágenes de Prueba	Precisión	Perdida
200	0.84	0.2437

De las 200 imágenes nunca antes vistas por el modelo, 100 son imágenes que contienen flores de hortensia en alguna región de la imagen y las otras 100 imágenes son de bosque en general. Al realizar la predicción del conjunto de imágenes de prueba se obtuvo la siguiente matriz de confusión que muestra los errores de clasificación (falsos positivos).

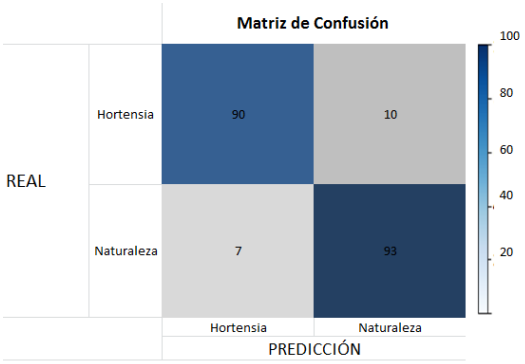

Imagen 3.5: Matriz de confusión para 200 imágenes nunca antes vistas por el modelo inicial

Analizando las predicciones del conjunto de datos nunca antes vistos por el modelo se puede generalizar cuatro tipos de imágenes. Imágenes que contienen flor de hortensia clasificadas correctamente, este tipo de imágenes tienen la característica general de visualizar las flores de hortensia de forma clara en alguna porción de la imagen.

La Imagen 3.6 muestra cuatro muestras del sub conjunto de imágenes típicas de hortensia donde la flor se aprecia claramente en la imagen.

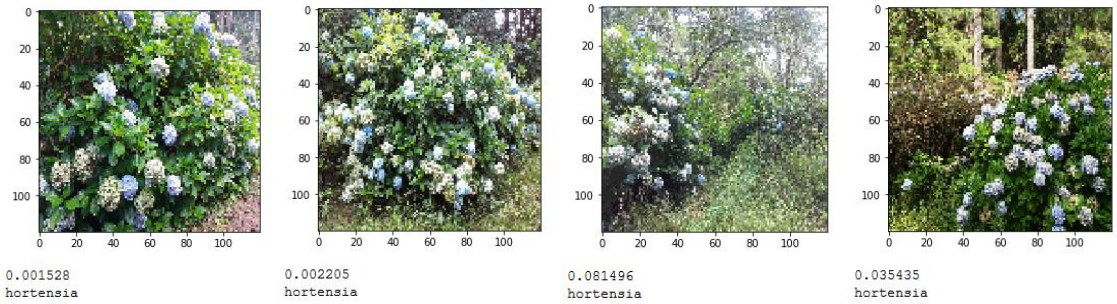

Imagen 3.6: Predicción del modelo inicial en imágenes donde la flor de hortensia se visualiza claramente

De forma general cuando las imágenes contienen flores de hortensia de forma clara en la mayor parte de la imagen e incluso cuando la flor solo se encuentra en algún sector de la imagen se genera una predicción correcta. En la parte inferior se aprecia el valor de predicción asignado a la imagen por el modelo, para las cuatro muestras son valores cercanos a cero indicando que el modelo clasifica la imagen como una flor de hortensia, lo anterior se define debido a la función de activación final (*sigmoid*), los valor menores a 0.5 representan la categoría hortensia y mientras más cercano a valor 0 indica una mayor precisión y en caso contrario par avalores mayores a 0.5 se trata de una imagen de naturaleza general. Para nuestras cuatro muestras de estudio se aprecia

valores menores a 0.1 indicando que el modelo ofrece buenas predicciones considerando que se trata de un modelo inicial.

En el caso de imágenes de hortensia donde la flor de hortensia se encontraba rodeada por vegetación, o se encontraban presentes en la imagen pero de forma no tan clara, en este caso se clasificaron incorrectamente. Las predicciones se observan en la Imagen 3.7, corresponden a valores superiores a 0.5.

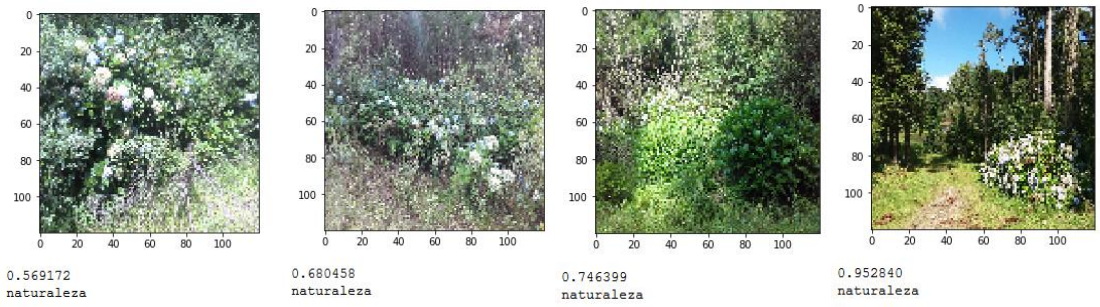

Imagen 3.7: Predicción del modelo inicial en imágenes donde la flor de hortensia no se aprecia claramente

Analizando la imagen anterior, en las dos figuras de la izquierda la flor se encuentra rodeada por follaje, el valor de predicción es ligeramente mayor a 0.5 indicando que el modelo puede identificar características de hortensia en las imágenes pero de forma global predomina las características de naturaleza. Ahora visualizando las dos figuras de la derecha, las flores de hortensia se encuentran con una mayor profundidad al resto de la vegetación y en otros casos se encuentra rodeada de árboles. Los valores de predicción son más cercanos al valor 1, que representa la clase naturaleza e indican que el modelo se le complica identificar flores de hortensia en imágenes con estas características.

Para la clase naturaleza al ser más general se seleccionó 4 muestras clasificadas correctamente mostradas en la Imagen 3.8. De forma general toda imagen constituida en su totalidad por vegetación, será clasificada como imagen de naturaleza. Los va-lores obtenidos por el modelo se colocan debajo de cada imagen, para las cuatro muestras se predijeron valores mayores a 0.8 indicando una buena clasificación de la clase naturaleza.

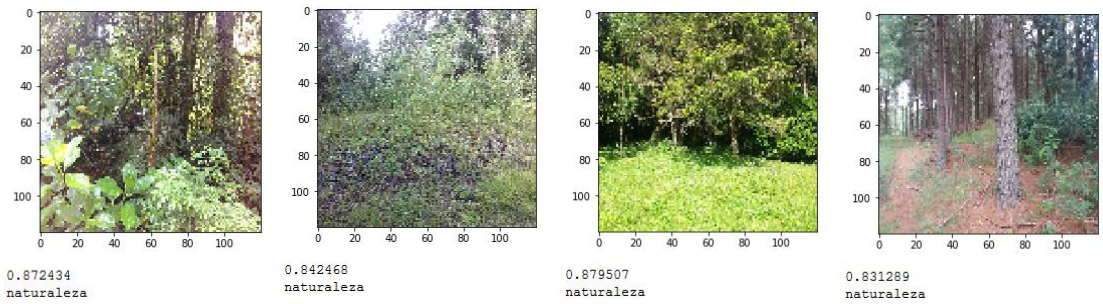

Imagen 3.8: Predicción del modelo inicial en imágenes de naturaleza

El modelo base clasifica correctamente imágenes que contengan solo naturaleza en general sin embargo cuando encontraba objetos como son el caso de autos, casas o letreros, el modelo tiende a clasificar incorrectamente. La imagen siguiente muestra cuatro muestras clasificadas incorrectamente por el modelo.

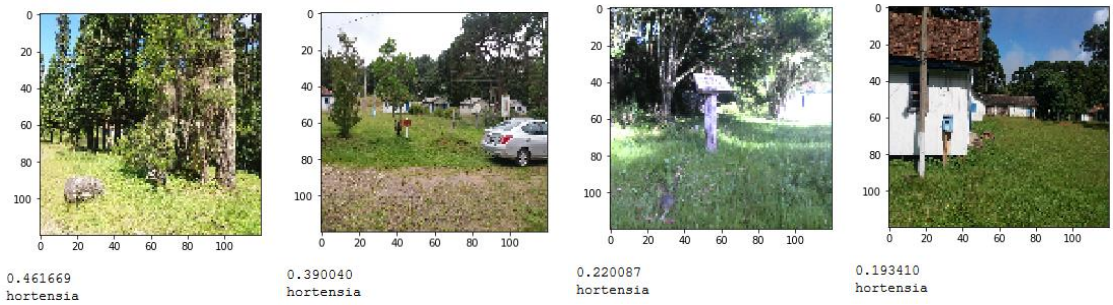

Imagen 3.9: Predicción del modelo inicial en imágenes de naturaleza que contiene objeto no naturales

En este conjunto de muestras se percibe una irregularidad ya que el modelo predice flor de hortensia con valores menores a 0.5, sin embargo los objetos como la casa, el auto, el letrero y la roca no tienen características similares a una flor de hortensia. Lo anterior indica que el modelo ofrece una predicción equivocada al no poder darle sentido a las porciones de imagen correspondiente a objetos no naturales. Esta limitación del modelo se debe a la utilización de un conjunto de imágenes de entrena-miento pequeño, para que el modelo pueda clasificar imágenes de este tipo requiere más información.

Para todos casos una posible solución es incrementar el número de imágenes que permitan al modelo contar con un conjunto de filtros más completo que pueda percibir de una manera más óptima las características de la flor de hortensia o de vegetación en general. Como se ha definido en los objetivos se busca implementar un modelo CNN que utilice conjuntos de datos pequeños y que no representa un gasto computacional considerable por lo que se buscaron alternativas para mejorar el modelo CNN inicio

solo utilizando los datos definidos. La posible solución fue utilizar un generador de datos de la paquetería *Keras* la cual por medio de operaciones de trasformación genera imágenes nuevas de las ya existentes.

3.2 Entrenamiento con aumento de datos

El principal motivo de que nuestro modelo presente sobre-entrenamiento (*Overfiting*), se debe a que tiene muy pocas imágenes de donde aprender características, lo que le impide capacitar a un modelo que puede generalizarse a datos nunca antes vistos. Dada una cantidad infinita de datos, su modelo estaría expuesto a todos los aspectos posibles de la distribución de datos: nunca estaría sobredimensionado. El aumento de datos (*data_argumentation*) adopta el enfoque de generar más datos de capacitación a partir de muestras de capacitación existentes, mediante el aumento de las muestras a través de una serie de transformaciones aleatorias que producen imágenes de aspecto creíble. El objetivo es que durante el entrenamiento, el modelo nunca vea exactamente la misma imagen dos veces. Esto ayuda a exponer el modelo a más características de los datos, generalizando de mejor forma el proceso de clasificación.

Keras proporciona herramientas que realizan varias transformaciones aleatorias en las imágenes leídas por el generador *ImageDataGenerator*, al aplicar una transformación de acercamiento (zoom), una imagen puede representar diversas imágenes con diferente perspectiva.

Algunas de las opciones disponibles para generar transformaciones aleatorias a los datos existentes se describen a continuación.

1. rotation_range: es un valor en grados (0-180), un rango dentro del cual aleatoriamente rotar imágenes.
2. width_shift y height_shift: son rangos (como una fracción del ancho total o la altura) dentro de los cuales se pueden traducir imágenes de forma aleatoria u horizontal.
3. shear_range: realiza una corte en la imagen
4. zoom_range: Realiza un zoom al azar dentro de las imágenes.
5. horizontal_flip: Voltear aleatoriamente la mitad de las imágenes de forma horizontal-relevante cuando no hay suposiciones de asimetría horizontal (por ejemplo, imágenes del mundo real).

6. fill_mode: Se utilizada para rellenar píxeles recién creados, lo que puede aparecer después de una rotación o un cambio de ancho / altura.

Para ilustrar una de las operaciones de transformación en una imagen, se muestra en la siguiente imagen una secuencia de operaciones creando tres nuevas imágenes obtenidas de una original.

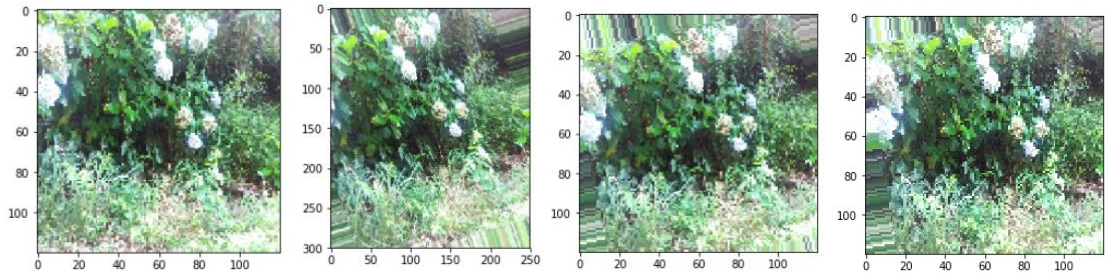

Imagen 3.10: Ejemplos de una imagen aplicando operaciones de transformación

Si entrena una nueva red usando esta configuración de aumento de datos, la red nunca verá la misma entrada dos veces. Pero las entradas que ve todavía están intercorrelacionadas, porque provienen de un pequeño número de imágenes originales: no puede producir nueva información, solo puede re-mezclar la información existente. Esto puede no ser suficiente para deshacerse completamente del *Overfitting*. Para combatir aún más el sobreajuste, también se agregará una capa de *Dropout* a nuestro modelo, justo antes del clasificador densamente conectado.

3.2.1 Entrenamiento del modelo con incremento de datos (Mid)

Entrenar a la red usando el aumento y el abandono de datos pretende reducir el sobre entrenamiento del modelo inicial. Trazando los resultados nuevamente se puede observar la evolución de la precisión y la perdida durante 100 épocas de entrena-miento. Gracias al incremento de imágenes de los conjuntos de datos y al abandono de datos, el sobre-ajuste disminuye considerablemente como se observa en la Imagen 3.11.

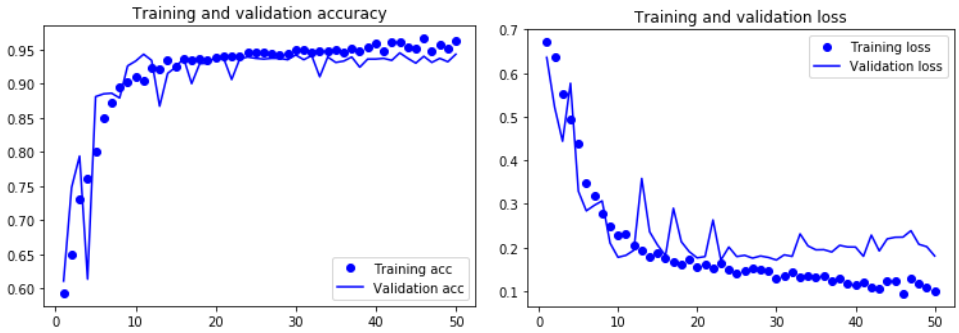

Imagen 3.11: Graficas de precisión y pérdida de los datos de entrenamiento con aumento de datos

Las curvas de entrenamiento siguen de cerca las curvas de validación. Ahora alcanza una precisión superior al 95%, una mejora relativa para datos nunca antes vistos por el modelo que en el modelo inicial ronda el 89%-90% de precisión.

3.2.2 Prueba de datos nunca antes vistos por el modelo con incremento de datos

Evaluando nuevamente las 200 imágenes de prueba nunca antes vistos por el modelo, utilizadas en el modelo base se obtuvieron los siguientes resultados.

Tabla 3.3 Parámetros de precisión y perdida en imágenes nunca antes vistas por el modelo con incremento de datos.

# imágenes de Prueba	Precisión	Perdida
200	0.915	0.1341

De las 200 imágenes evaluadas se obtuvo la siguiente matriz de confusión.

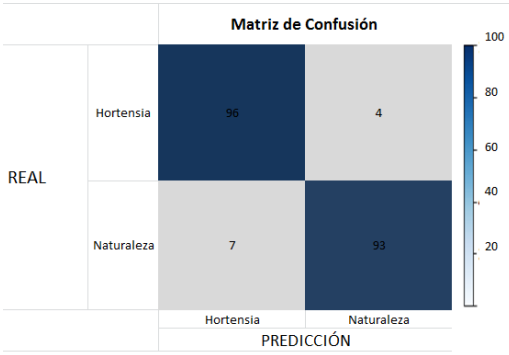

Imagen 3.12: Matriz de confusión probadas a 200 imágenes nunca antes vistas por el modelo Mid.

Utilizando el incremento de datos de entrenamiento y validación mejoro la predicción para flores de hortensia, en el modelo inicial se predijeron 10 falsos positivos de 100 imágenes de prueba. El modelo con incremento de datos (Mid) predijo 4 falsos positivos para las mismas 100 imágenes de prueba, en cambio para la clase naturaleza se mantuvo un número de 7 falsos positivos de 100 imágenes de prueba.

Analizando las 16 muestras del conjunto de datos de prueba empleadas en el modelo inicial sin extensión de datos se realizara una comparación de resultados para cuantificar la mejoría del modelo extendido.

Las 4 imágenes donde la flor de hortensia se aprecia claramente presentaron una mejoría en precisión, obtuvieron predicciones menores a 0.001, siendo valores más cercanos al valor cero por lo tanto correspondiente a la clase hortensia. La Imagen 3.13 muestra las imágenes con sus nuevas predicciones.

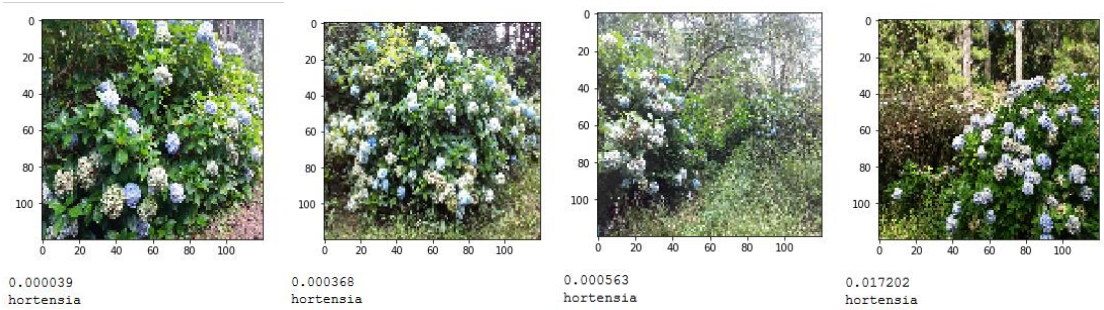

Imagen 3.13: Predicción de imágenes con flores de hortensia claras con modelo Mid.

Para el bloque de imágenes donde no se aprecia de forma clara la flor de hortensia, el nuevo modelo extendido mejoro la predicción para imágenes donde flores de hortensia se encuentra rodeadas por vegetación, clasificándolas como hortensia. Estas predicciones se aprecian en la figura siguiente.

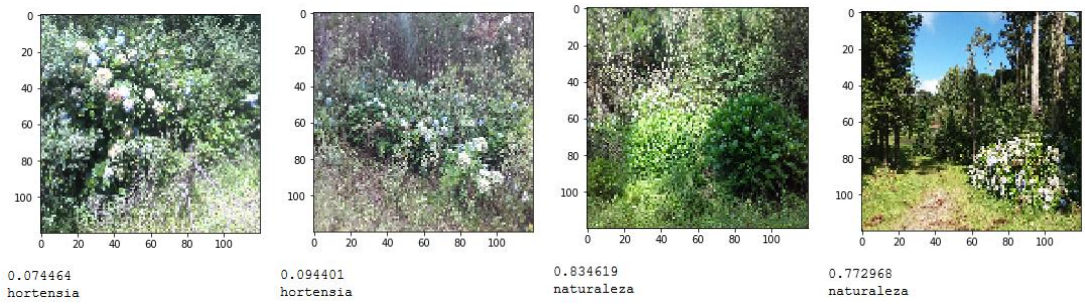

Imagen 3.14: Predicción en imágenes donde la flor de hortensia es difícil de apreciar.

En los casos donde la flor de hortensia se encuentra con una mayor profundidad al resto de la vegetación el modelo extendido sigue clasificándolas como naturaleza, como se muestra en las 2 figuras de la izquierda en la Imagen 3.14.

Analizando las cuatro muestras de naturaleza general, se aprecia una mejoría notable en los valores de predicción. Los valores de predicción para las cuatro muestras fueron superiores a 0.95, indicando que el modelo extendido tiene mejores cualidades en precisión para la predicción de ambas clases, comparado con el modelo inicial.

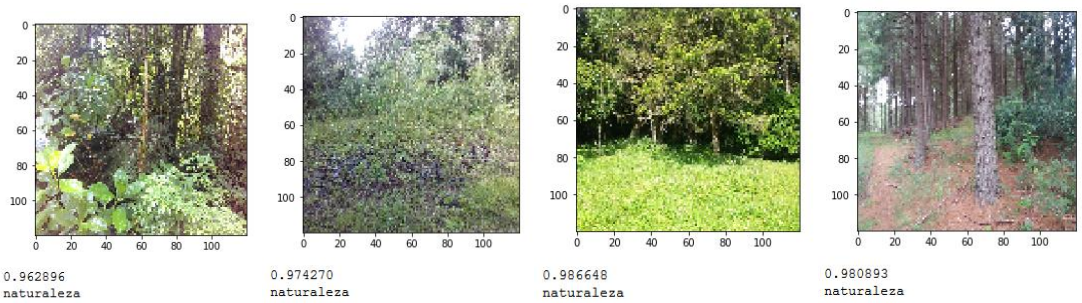

Imagen 3.15: Predicción de imágenes con modelo Mid en imágenes de naturaleza.

Finalmente para las cuatro muestras de naturaleza con objetos extraño no se observó mejoría en su predicción, el modelo extendido las sigue clasificándolas de forma incorrecta como flores de hortensia, tal y como se muestra en la Imagen 3.16. Comparando los valores de la predicción con los obtenidos en el modelo inicial se observa que los valores de predicción se reducen acercándose más a cero por lo que no existe ninguna mejoría para estas muestras.

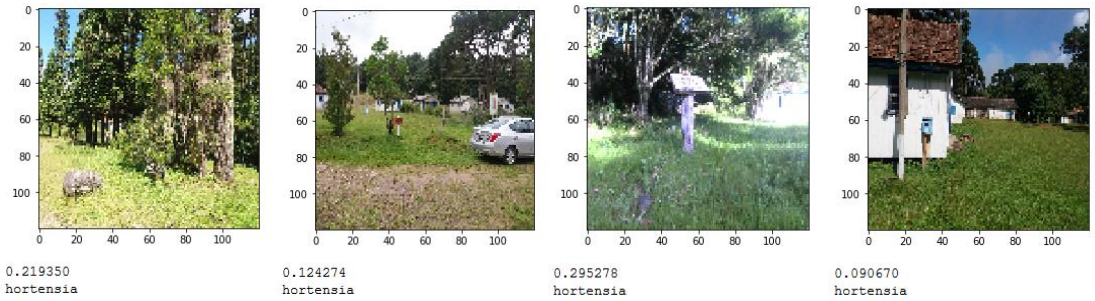

Imagen 3.16: Predicción de imágenes con modelo Mid en imágenes de naturaleza complejas.

3.3 Análisis de resultados de la implementación inicial y con extensión de datos

De forma general el modelo con incremento de datos proporciono mejores la predicción para imágenes de hortensia con diferentes características sin embargo cuando las flores no son claras es difícil para el modelo predecir la presencia de flores hortensia. Para la clase naturaleza mejoro la predicción sin embargo el modelo extendido no puede predecir correctamente imágenes con objetos desconocidos, arrojar valores de predicción erróneos.

En este punto el incremento de imagen por generador de *Keras* tiene una limitante y es la relación que tienen las imágenes generadas. Debido a esta limitante se modificaran diversos parámetros de nuestro modelo con incremento de datos en busca de obtener mejores resultados.

CAPÍTULO IV

4 IMPLEMENTACIÓN EXPERIMENTAL Y OPTIMIZACIÓN

Para mejorar los resultados previamente obtenidos en el capítulo anterior se recurrirá a la modificación de diversos parámetros en la estructura de la red los cuales permitan mejorar la precisión de clasificación. Los parámetros considerados fueron:

1. Incrementar el número de capas convolucionales.
2. Modificar la proporción de los conjuntos de datos
3. Modificar la función de activación final.

Incrementar las capas de convolucion nos garantiza contar con un mayor número de filtros que pueden identificar de mejor forma las características de una flor de hortensia en una imagen. Aumentar los datos de entrenamiento reduciendo datos en el subconjunto de validación es otra forma de garantizar un mayor número de imágenes que pueden apartar características que generalicen de mejor forma una planta invasiva. Finalmente se implementara un nueva función de activación final (*Sofmax*) que en ligar de normaliza los valores de predicción de forma categórica.

4.1 Optimización con 6 capas de convolución (6CC)

Buscando mejorar la precisión de predicción de flores de hortensia en imagen se incrementa las capas convolucionales de la arquitectura inicial de 4 capas a 6 capas. Este incremento de capas de convolución proporciona un incremento de filtros o *Kernels* que pueden resaltar un mayor número de características de una imagen, permitiendo clasificar de una forma más óptima las clases de nuestro modelo. La Imagen 4.1 muestra la nueva arquitectura implementada en nuestro modelo extendido.

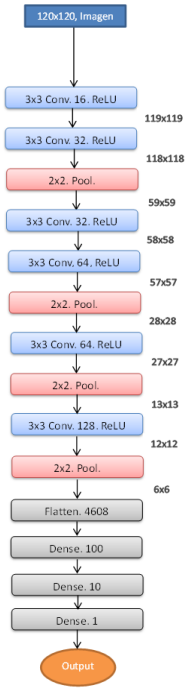

Imagen 4.1: Arquitectura CNN con 6 capas de convolucion.

Como se observa en el diagrama de bloques las dos nuevas etapas de convolucion implementadas en nuestro modelo extendido incrementan de 288 filtros a 336 filtros convolucionales. Este incremento busca mejorar la predicción de las clases.

4.1.1 Entrenamiento modelo 6CC

Trazando los resultados nuevamente se puede observar la evolución de la precisión y la perdida durante 100 épocas de entrenamiento. Gracias al incremento de imágenes de los conjuntos de datos y al abandono de datos, el sobre-entrenamiento disminuye considerablemente como se observa en la Imagen 4.2.

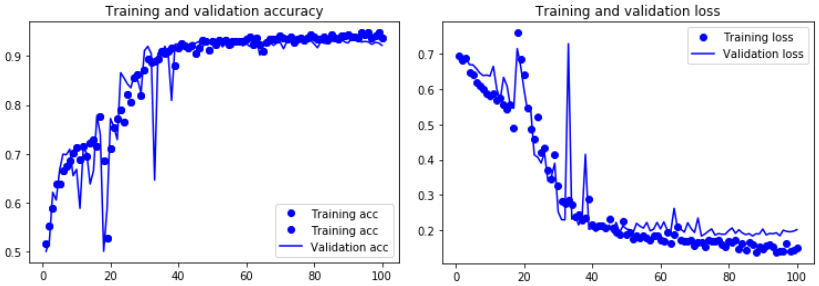

Imagen 4.2: Graficas de precisión y pérdida de los datos de entrenamiento y validación del modelo 6CC.

Las curvas de entrenamiento siguen de cerca las curvas de validación. Ahora alcanza una precisión superior al 95%, una mejora relativa para datos nunca antes vistos por el modelo que en el modelo inicial ronda el 89%-90% de precisión.

Al utilizar aún más las técnicas de regularización y al ajustar los parámetros de la red (como el número de filtros por capa de convolución o el número de capas en la red), es posible que obtenga una precisión aún mayor, probablemente hasta un 96% de precisión.

4.1.2 Prueba de datos nunca antes vistos por el modelo

Evaluando 200 imágenes da datos nunca antes vistos por el modelo, evaluando las muestras con el modelo previamente entrenado se obtuvieron los siguientes resultados.

Tabla 4.1 Parámetros de precisión y perdida del modelo inicial en imágenes nunca antes vistas por el modelo 6CC

# imágenes de Prueba	Precisión	Perdida
200	0.95	0.1272

En cuanto a precisión global se presentó un incremento de 94%-95% y una reducción del parámetro de pérdida del modelo, este pequeño incremento se ve reflejado en la matriz de confusión de la Imagen 4.3.

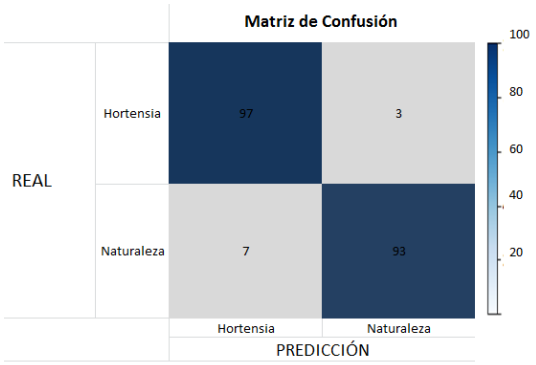

Imagen 4.3: Matriz de confusión probada a 200 imágenes nunca antes vistas por el modelo 6CC.

El número de falsos positivos para la clasificación se redujo de 4 en el modelo con incremento de datos a 3 falsos positivos en el modelo con 6 capas convolucionales.

De las 16 de imágenes de prueba analizadas en modelos anteriores se obtuvieron mejorías en los parámetros de predicción. Las cuatro imágenes típicas de hortensia con predicción correcta obtuvieron valores más cercanos a cero comparando con los modelos anteriores.

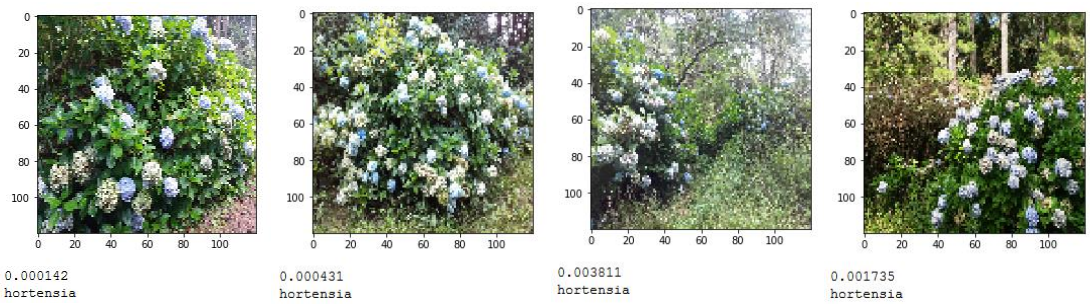

Imagen 4.4: Predicción de imágenes con modelo 6CC de flores de hortensia claras.

Los valores de predicción para las dos imágenes de la izquierda donde la flor de hortensia se encuentra en la mayor parte de la imagen tienen valores menores a 0.0005, siendo un valor muy cercano a cero. Los valores para las dos imágenes de la derecha en la Imagen 4.4, son valores menores a 0.004, en ambos casos son predicciones muy precisas.

Analizando las cuatro imágenes de hortensia con predicción incorrecta por el modelo inicial se obtiene un resultado similar al segundo modelo con incremento de imágenes, donde las dos imágenes dela izquierda de la imagen 4.5 son clasificadas correctamente y con mayor precisión al modelo con incremento de imagen. Las dos imágenes de la derecha continúan siendo clasificadas incorrectamente, sin embargo una de estas empieza a presentar más características de flores de hortensia obteniendo un valor cercano a (0.5). Lo anterior indica que el incremento de capas convolucionales es una buena alternativa para el mejorar el modelo.

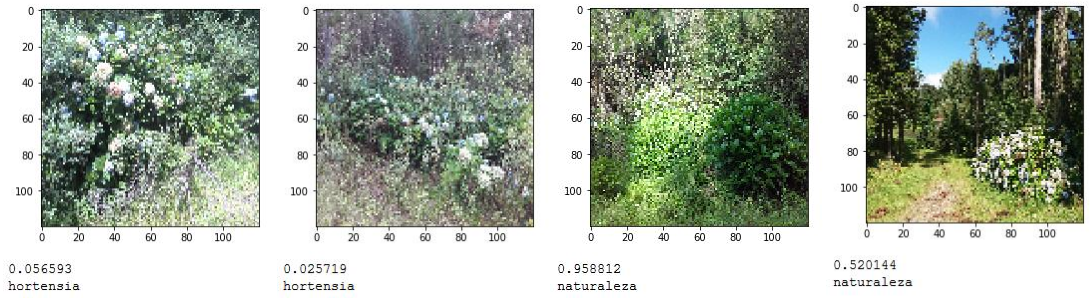

Imagen 4.5: Predicción de imágenes con modelo 6CC donde la flor de hortensia no es clara.

Las cuatro imágenes típicas de naturaleza que fueron clasificadas correctamente con el modelo con incremento de datos, no se reflejó mejoras en las predicciones del modelo de 6 capas de convolucion, continúan clasificadas correctamente con la clase naturaleza.

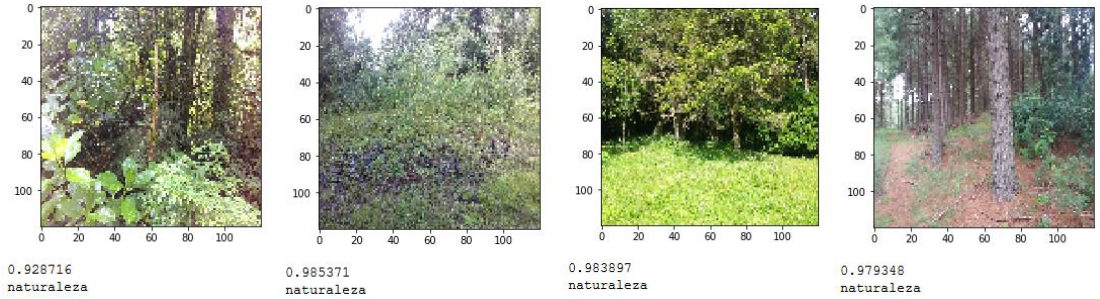

Imagen 4.6: Predicción de imágenes con modelo 6CC en imágenes de naturaleza.

Para las imágenes del conjunto de naturaleza clasificadas incorrectamente por los dos previos modelos, no se presentaron mejoras para este nuevo modelo con 6 capas de consolación.

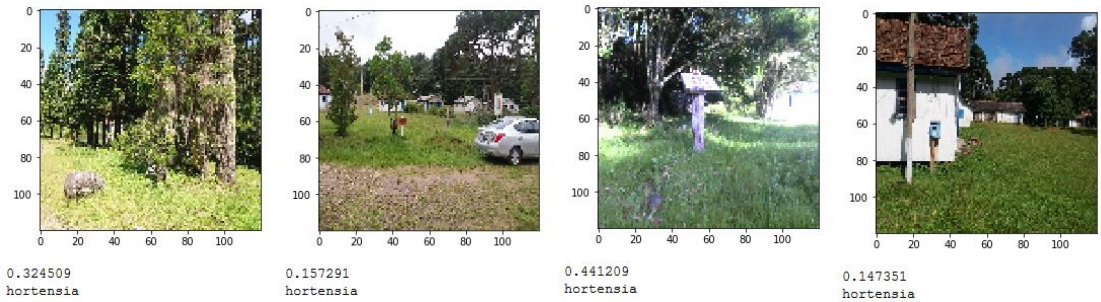

Imagen 4.7: Predicción de imágenes con modelo 6CC en imágenes de naturaleza complejas.

En general el modelo 6CC demostró brindar mayor precisión para la detección de flores de hortensia, los valores de predicción se han acercado más a el valor 0 correspondiente a la clase hortensia, por lo que el incremento de filtros de clasificación de modelo permite mejor la clasificación. Sin embargo el modelo no mostro mejoría para la clasifi-

cación de la clase naturaleza por lo que se propondrá un nuevo incremento de capas convolucionales buscando mejorar un poco más el modelo.

4.2 Optimización con 8 capas de convolucion (8CC)

El incremento de las capas convolucionales genero aumentos en la precisión de predicción de flores de hortensia por lo que se realizara un nuevo incremento en la arquitectura de 6 a 8 capas convolucionales. Este incremento de capas de convolución incrementa el número de filtros que permite considerar características de las clases que no se consideraron en modelos anteriores, resaltando un mayor número de características de una imagen. La imagen 4.8 muestra la nueva arquitectura implementada con 8 capas convolucionales.

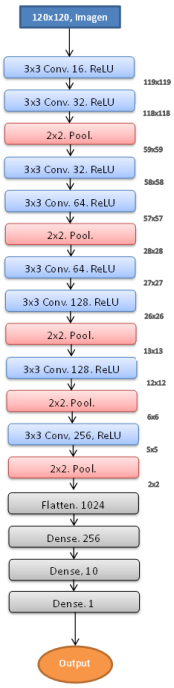

Imagen 4.8: Arquitectura CNN del modelo con 8CC.

Como se observa en el diagrama de bloques las dos nuevas etapas de convolucion implementadas en nuestro modelo extendido, incrementan de 336 filtros a 720 filtros convolucionales. Este incremento busca mejorar la predicción de las clases hortensia/naturaleza.

4.2.1 Entrenamiento modelo 8CC

Trazando los resultados nuevamente se puede observar la evolución de la precisión y la perdida durante 100 épocas de entrenamiento. Gracias al incremento de imágenes de los conjuntos de datos y al abandono de datos, el sobre-entrenamiento disminuye considerablemente como se observa en la Imagen 4.9.

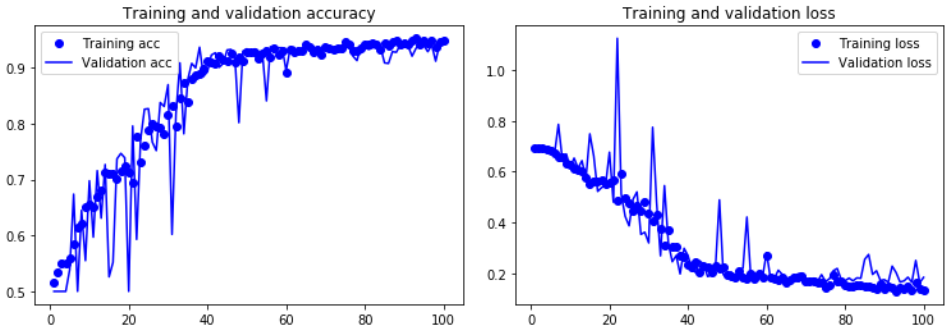

Imagen 4.9: Graficas de precisión y pérdida de los datos de entrenamiento y validación del modelo 8CC.

Las curvas de entrenamiento y de pérdida del modelo de 8 capas de convolución tienen una convergencia más notable, en este modelo el sobre-entrenamiento prácticamente desaparece. Analizando la curva de precisión durante 100 épocas, el modelo alcanza una precisión superior al 97%, una mejora considerable comparándola con el modelo con 6 capas de convolución. En el caso de las curvas de pérdida, se observa que decrecen para datos de validación como para datos de entrenamiento de forma continua y uniforme llegando a valores menores de 0.1 indicando que el modelo tiene condiciones de predicción superiores a los modelos anteriores.

4.2.2 Prueba de datos nunca antes vistos por el modelo

Evaluando las imágenes del conjunto de evaluación conformada por 200 imágenes da datos nunca antes vistos por el modelo, se obtuvieron los siguientes resultados.

Tabla 4.2 Parámetros de precisión y perdida del modelo inicial en imágenes nunca antes vistas por el modelo 8CC.

# imágenes de Prueba	Precisión	Perdida
200	0.97	0.1289

La precisión obtenida representa una precisión superior al modelo con 6 capas de convolucion y una pérdida o desviación relativamente pequeña. Evaluando las 200 imágenes obtenemos la siguiente matriz de confusión que muestra los errores de clasificación (falsos positivos).

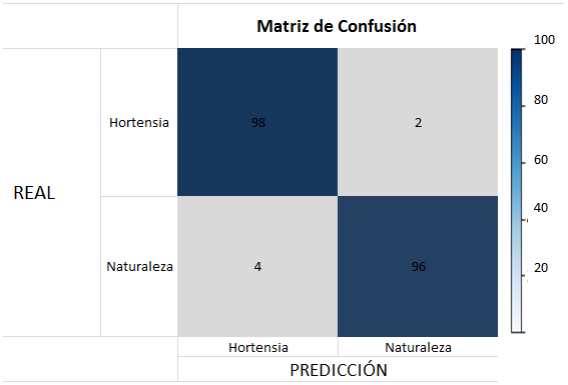

Imagen 4.10: Matriz de confusión probada con 200 imágenes nunca antes vistas por el modelo 8CC.

Las predicciones para datos nunca antes vistos por el modelo mejoraron para ambas clases. La clase hortensia se predijo correctamente 98 de 100 imágenes y la clase naturaleza se predijo correctamente 96 de 100 imágenes.

Evaluando los valores de predicción para las 16 imágenes de muestra del conjunto de prueba se obtuvieron los siguientes resultados. El primero conjunto de cuatro imágenes de hortensia fueron clasificadas correctamente por el modelo 8CC y con una mejora considerable en la precisión para ambas clases.

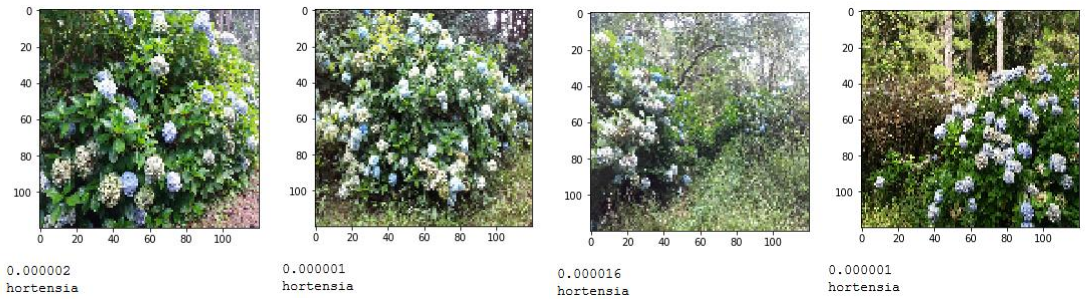

Imagen 4.11: Predicción de imágenes con modelo 8CC de flores de hortensia claras.

El modelo de ocho capas de convolucion generó un incremento notable en la presión con valores menores a 0.000001 para imágenes donde las flores de hortensia se encuentran presentes de forma clara, estos valores representan valores cercanos al cero indicando que en la imagen se encuentran flores de hortensia, Imagen 4.11.

Para el segundo conjunto de cuatro imágenes obtuvieron buenos resultados para imágenes donde la flor se encuentra rodeada por vegetación.

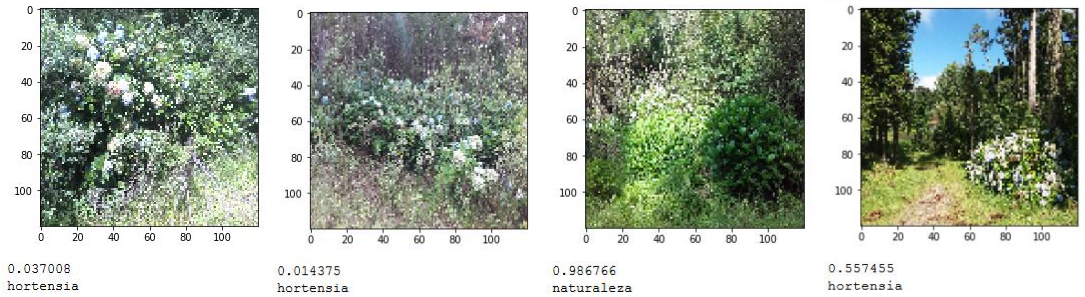

Imagen 4.12: Predicción de imágenes con modelo 8CC donde la flor de hortensia no es clara.

Las dos figuras del lado derecho de la Imagen 4.12 representan las dos muestras clasificadas incorrectamente por el modelo, en ambas imágenes las flores de hortensia se encuentran con una profundidad en la imagen, y se resaltan características de vegetación, arboles, cielo, etc.

El primer grupo imágenes de naturaleza evaluadas con el modelo 8CC incremento la precisión con valores más cercanos la clase naturaleza con un promedio de precisión superior a 97%.

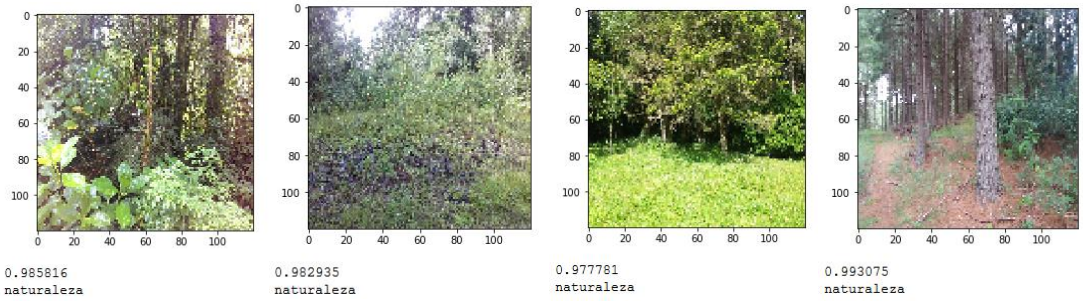

Imagen 4.13: Predicción del modelo 8CC en imágenes de naturaleza.

Para el grupo de imágenes de naturaleza objetos extraños se redujo la precisión obtenida por el modelo de 6 capas de consolación.

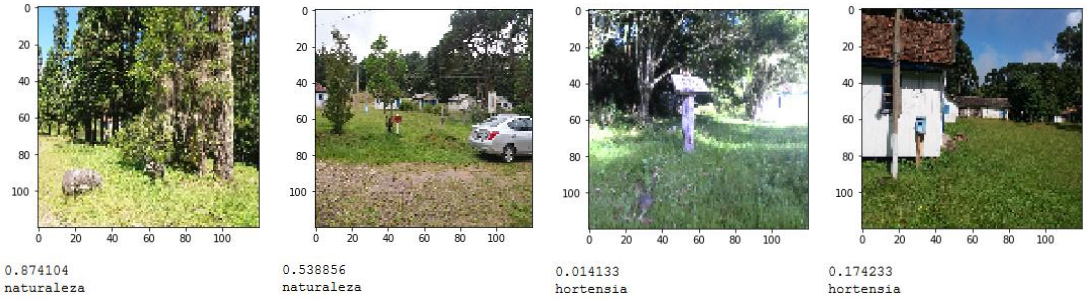

Imagen 4.14: Predicción del modelo 8CC en imágenes de naturaleza que contiene objetos.

Se clasifico correctamente las dos figuras de la izquierda de la Imagen 4.14, en estas imágenes se encuentran objetos como la parte trasera de un automóvil y una roca, sin embargo la mayor parte de la imagen está constituida por vegetación. Las dos figuras de la derecha de la Imagen 4.14 se clasificaron incorrectamente debido a falta de información de objetos como la casa o letreros, estos objetos representan una complicación debido a que en el conjunto de entrenamiento no cuenta con suficientes muestras que caractericen este tipo de objetos.

4.3 Optimización del modelo modificando la proporción de los sub-conjuntos de imágenes (80/20)

Aunque el modelo de ocho capas de convolucion representa una implementación de alta precisión para la clasificación de flores de hortensia/naturaleza, se intentara mejorar estos resultados modificando la proporción del conjunto de entrenamiento y validación aplicado al modelo previo 8CC. La proporción utilizada en todos los modelos an-

teriores fue de 2000 imágenes de entrenamiento 1000 imágenes de validación equivalente a 2/3 de imágenes de entrenamiento por 1/3 de imágenes de validación.
Lo que se busca es modificar esta proporción aumentando el número de imágenes del conjunto de entrenamiento buscando abarcar un mayor número de características de imágenes distintas.

La propuesta es dividir el conjunto total de imágenes (3000 imágenes) en una proporción 80% imágenes de entrenamiento por 20% de imágenes de validación.

4.3.1 Entrenamiento modelo 80/20

Para el proceso de entrenamiento se utilizó la estructura de 8 capas de convolicion y se modificaron los subconjuntos de imágenes, aumentando los datos de entrenamiento en 2400 imágenes y reduciendo los datos de validación en 600 imágenes. La evolución del modelo durante 100 épocas se muestra en la Imagen 4.15.

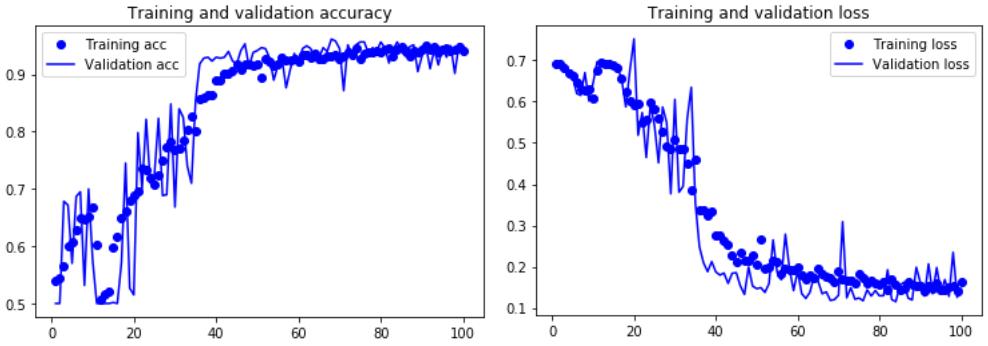

Imagen 4.15: Graficas de precisión y pérdida del proceso de entrenamiento del modelo con 80/20.

La evolución de la precisión del modelo se torna ruidosa para los datos de validación durante 40 épocas, seguido de convergencia para las dos métricas. En el caso de la pérdida del modelo se visualiza ruido por parte de los datos de validación y se atenúa conforme se acerca a la época 100.

Este tipo de ruido en los datos de validación se origina por la reducción de su conjunto. Para evaluar el modelo con respecto de los entrenados anteriormente es necesario evaluar su comportamiento para datos nunca antes vistos.

4.3.2 Prueba de datos nunca antes vistos por el modelo

Evaluando 200 imágenes da datos nunca antes vistos por el modelo, se obtiene una precisión menor al modelo 8CC.

Tabla 4.3 Parámetros de precisión y perdida del modelo inicial en imágenes nunca antes vistas por el modelo con proporción 80/20.

# imágenes de Prueba	Precisión	Perdida
200	0.955	0.1397

Analizando las predicciones obtenidas del conjunto de prueba se obtuvo la siguiente matriz de confusión.

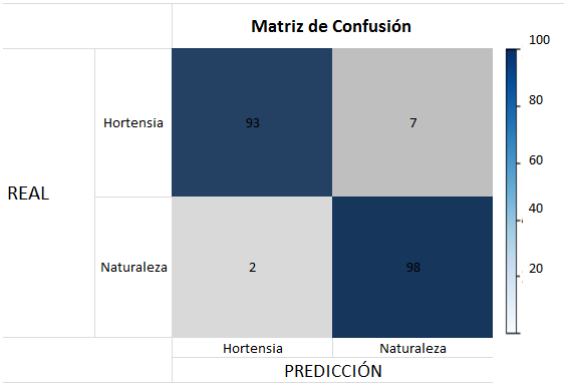

Imagen 4.16: Matriz de confusión probada con 200 imágenes nunca antes vistas por el modelo 80/20.

Como se aprecia en la imagen la reducción de la precisión general de modelo se ve reflejada en la clasificación de flores de hortensia incrementándose de dos falsos positivos a 7 falsos positivos. En cambio para la clase naturaleza se reduce el número de falsos positivos de siete a dos. Lo anterior indica que modificando la proporción imagen aumentando el número de imágenes del conjunto de entrenamiento se refleja en una mejor clasificación de imágenes de naturaleza. Lo anterior ocurre ya debido a que aumentando las imágenes se pueden encontrar más características de clasificación de naturaleza al no tener una restricción única de una tipo de planta en particular.

Evaluando las imágenes de muestra propuestas se obtuvieron los siguientes resultados.

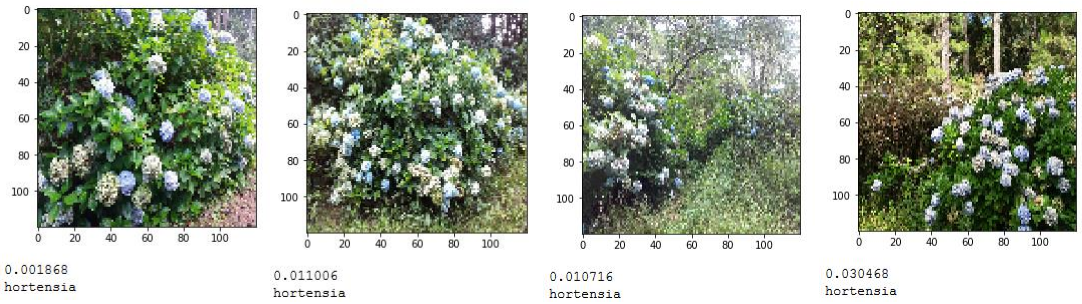

Imagen 4.17: Predicción de imágenes con modelo 80/20 de flores de hortensia claras.

Para las imágenes donde la flor de hortensia se encuentra de forma clara se clasificaron correctamente pero con una reducción de predicción mucho menor.

Para el caso de imágenes de hortensia difíciles de detectar se mantuvo la clasificación, las dos figuras de la izquierda se clasificaron correctamente y las dos figuras de la derecha se clasificaron incorrectamente con valores de precisión menores a los obtenidos con el modelo de ocho capas de convolucion.

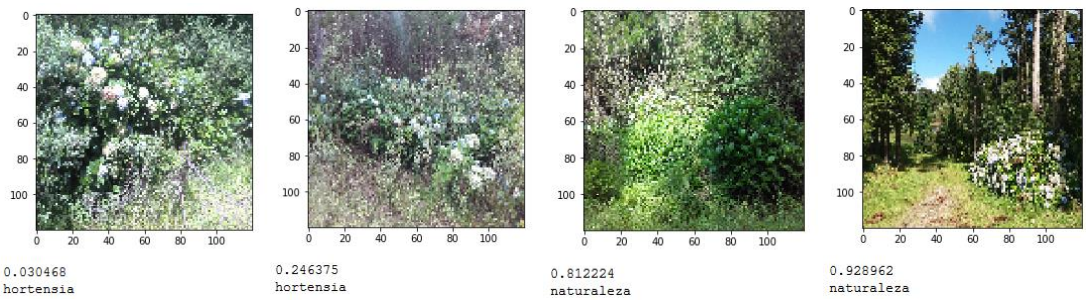

Imagen 4.18: Predicción de imágenes con modelo 80/20 donde la flor de hortensia no claras.

Las primeras cuatro muestra de naturaleza se clasificaron correctamente como se muestra en la Imagen 4.19, obteniendo valores de clasificación más precisos comparado con los modelos anteriores.

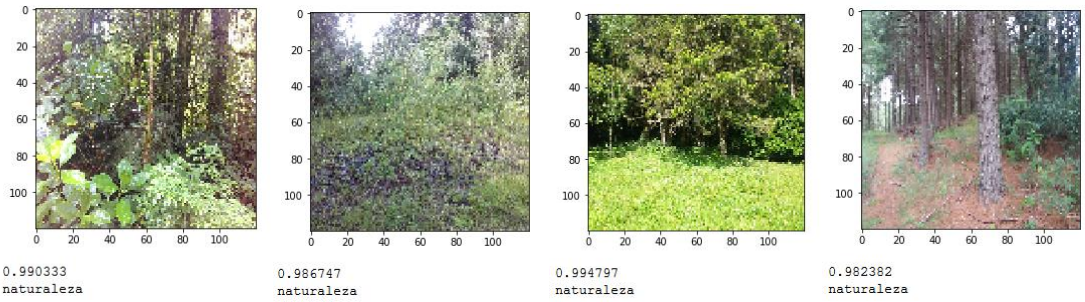

Imagen 4.19: Predicción del modelo con proporción 80/20 en imágenes de naturaleza.

Para las cuatro muestras de naturaleza con objetos no naturales se clasificaron correctamente mejorando las capacidades del modelo 8CC.

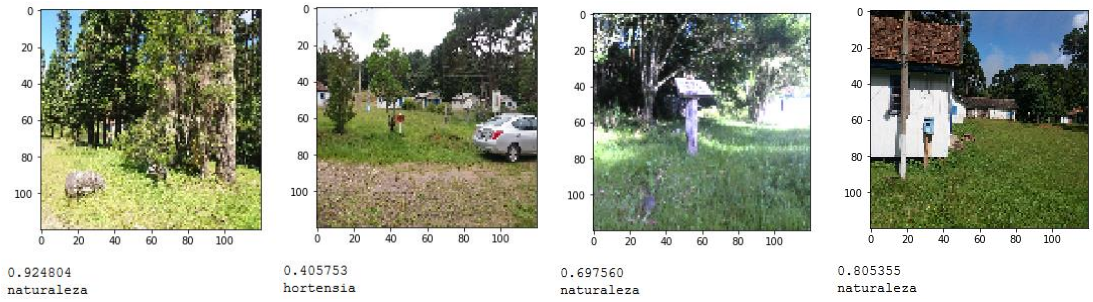

Imagen 4.20: Predicción del modelo 80/20 en imágenes de naturaleza que contiene objetos.

Lo resultados anteriores muestran que el modelo de proporción 80/20 dio mejores resultados para clasificar la clase naturaleza, sin embargo el objetivo principal del presente trabajo se enfoca en detección de plantas invasivas, para este conjunto de imágenes se busca identificar cuando una flor de hortensia se encuentra en una imagen, por lo que modificar la proporción de los conjuntos de imagen no representa una alternativa para mejorar la identificación de plantas invasivas.

4.4 Optimización utilizando activación final Softmax

Los resultados obtenidos hasta este momento son satisfactorios específicamente los obtenidos por el modelo 8CC donde se obtiene una precisión de clasificación del 97%, considerando que se trata de un conjunto de datos de entrenamiento pequeño. La predicción para datos nunca antes vistos por el modelo solo tiene problemas para detectar flores de hortensia cuando se encuentran muy profundas en la imagen, analizando una de las dos muestras del conjunto de prueba de flores de hortensia, se clasificada como

naturaleza. Si visualizamos la imagen de la figura 4.21 se aprecia que las características típicas de una hortensia no son del todo claras y el modelo clasifica la imagen como naturaleza. Talvez para este tipo de casos es complicado decir que el modelo clasifico incorrectamente por la pequeña porción de la imagen ocupada por flores de hortensia.

Imagen 4.21: Tipo de imagen de hortensia rodeada de vegetación que dificulta la predicción.

La imagen anterior muestra las flores alejadas del resto de la vegetación, incluso de forma visual es complicado visualizar si se trata de una hortensia. Por lo tanto se puede concluir que el modelo de ocho capas de convolucion es una alternativa aceptable para la clasificación de plantas invasivas.

Hasta este momento los resultados podrían ser exportados para implementar una aplicación de identificación de plantas invasivas pero se intentara obtener mejores predicciones utilizando una función de activación final diferente. Una de las complicaciones surgidas en la implementación de los modelos CNN fue la propiedad muy indeseable de la neurona *Sigmoid* la cual es que cuando la activación de la neurona se satura en los límites de 0 o 1, el gradiente en estas región es casi cero, durante la propagación hacia atrás, este gradiente (local) se multiplicará al gradiente de la salida. Por lo tanto, si el gradiente local es muy pequeño, "matará" el gradiente y casi ninguna señal fluirá a través de la neurona a sus pesos y recursivamente a sus datos.

El clasificador tipo *Softmax* ofrece un resultado un poco más intuitivo proporcionando las probabilidades de clase normalizadas, es decir el clasificador *Softmax* proporciona "probabilidades" para cada clase. A diferencia de la activación *Sigmoid* que calcula puntuaciones no calibradas y no fáciles de interpretar para todas las clases, el clasificador *Softmax* nos permite calcular las "probabilidades" para todas las etiquetas.

En su lugar, el clasificador de *Softmax* interpreta las puntuaciones como probabilidades de registro (no normalizadas) para cada clase y luego alienta a que la probabilidad

de registro (normalizada) de la clase correcta sea alta (de manera equivalente, el negativo de ella sea bajo).

4.4.1 Entrenamiento modelo SoftMax

Para el proceso de entrenamiento se utilizó la arquitectura del modelo 8CC que hasta el momento ha brindado los mejores resultados. Se realizaron algunos cambie en relación al número de filtros por capas, definiéndolos en 21 filtros. La Imagen 4.22 muestra la arquitectura modificada.

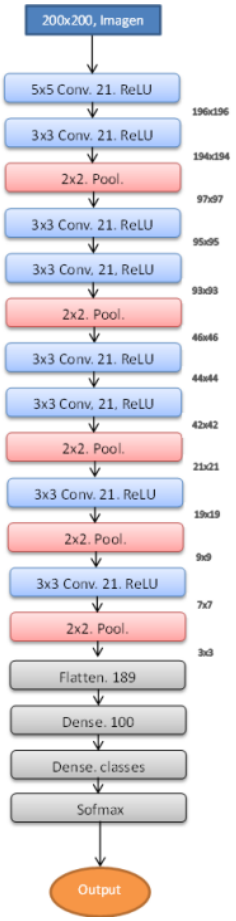

Imagen 4.22: Arquitectura CNN del modelo Softmax.

Las gráficas que evalúan las métricas de precisión y perdida para los datos de validación como para los datos de entrenamiento se visualizaron uniformes sin mucho ruido y convergiendo en las ultimas cuarenta épocas del proceso de entrenamiento.

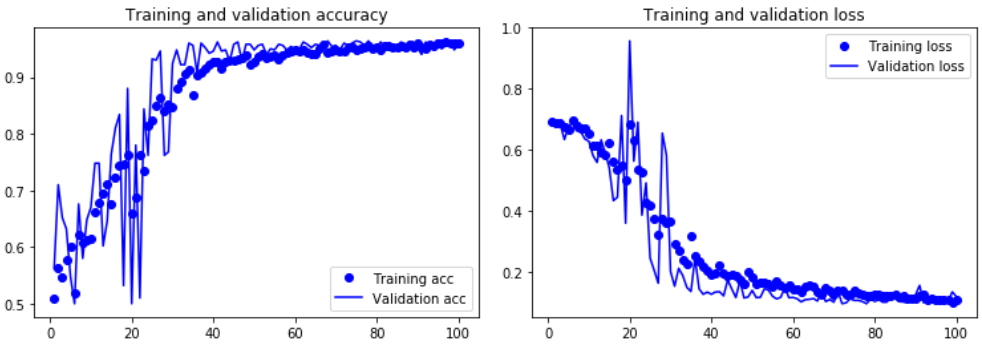

Imagen 4.23: Graficas de precisión y pérdida del proceso de entrenamiento del modelo Softmax.

Como se puede observar en la Imagen 4.23, no se presenta sobre-entrenamiento de los datos de entrenamiento con los datos de validación para ambas gráficas, esto indica que el entrenamiento tiene buenas características para clasificar datos nunca antes vistos por el modelo.

4.4.2 Prueba de datos nunca antes vistos por el modelo *Softmax*

Evaluando las 200 imágenes de prueba nunca antes vistas por el modelo, los resultados de clasificación del modelo *Softmax* fueron más precisos que los del modelo 8CC con activación *Sigmoid*.

Tabla 4.4 Parámetros de precisión y perdida del modelo inicial en imágenes nunca antes vistas por el modelo Softmax.

# imágenes de Prueba	Precisión	Perdida
200	0.98	0.1289

La matriz de confusión de la Imagen 4.24 muestra que las 200 imágenes fueron clasificadas correctamente mejorando las predicciones con relación a los modelos anteriores.

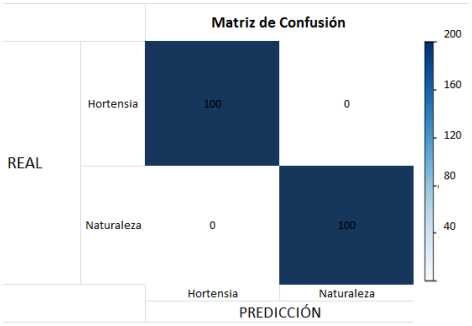

Imagen 4.24: Matriz de confusión probada con 200 imágenes nunca antes vistas por el modelo Softmax.

A diferencia de modelo con activación *Sigmoid*, donde las predicciones dependía de un valor entre 0-1 y para valores menores a 0.5 correspondía a la clase hortensia y valores mayores a 0.5 a la clase naturaleza. Para la activación final de tipo *Sigmoid* al ser un clasificador por clases, las predicciones se realizan de forma directa es decir la predicción de cada imagen es un porcentaje de precisión de que sea alguna de las dos clases. Para las muestras de imagen de hortensia donde la flor está presente de forma clara se obtuvieron las siguientes predicciones.

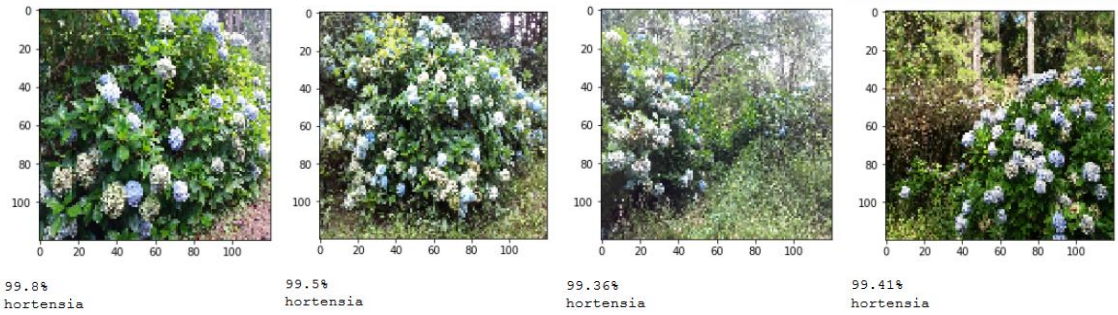

Imagen 4.25: Predicción de imágenes con modelo Softmax de flores de hortensia claras.

Los valores de precisión superan el 99% mejorando los resultados obtenido por el modelo de ocho capas de convolucion con activación *Sigmoid*, Imagen 4.25. En el caso de imágenes de hortensia difíciles de detectar se obtuvieron mejoras clasificando correctamente, Imagen 4.26.

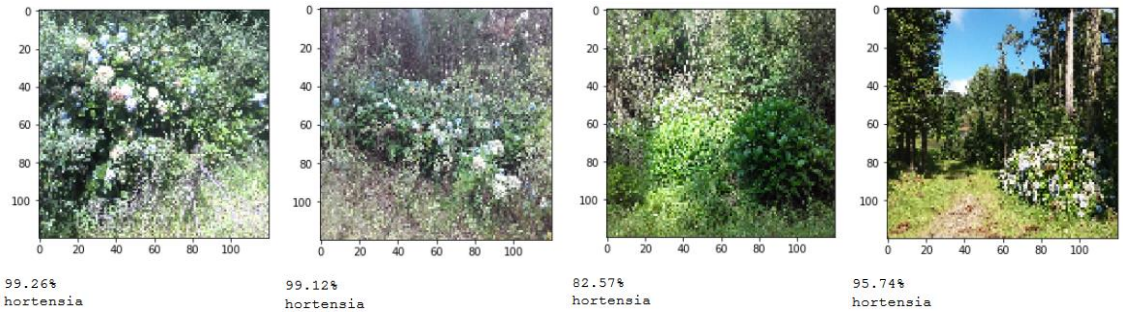

Imagen 4.26: Predicción de imágenes con modelo Softmax donde la flor de hortensia no es clara.

Como se observa las dos imágenes de la derecha fueron clasificadas correctamente con predicciones 82.57% y 95.74% respectivamente pero que demuestra que el modelo *Softmax* da solución al problema de clasificación de flores difusas en la vegetación.

Finalmente para las ocho muestras de imágenes de naturaleza de igual forma se obtuvieron clasificaciones correctas, Imagen 4.27.

Imagen 4.27: Predicción del modelo Softmax en imágenes de naturaleza.

El modelo con capa de activación final categórica clasifica imágenes de naturaleza con objetos extraños de forma correcta clasificándolas como imágenes de naturaleza.

4.5 Análisis de resultados experimentales

Analizando de forma general los tópicos desarrollados a lo largo del presente capitulo se observó que incrementar el número de capas convolucionales representaba una alternativa confiable para incrementar la precisión, sin embargo el incremento de capas convolucionales de igual forma representaba un gasto computacional cada vez mayor que no podíamos continuar debido a las limitaciones de nuestro equipo de cómputo. Por lo que se generaron nuevas propuestas como la modificación de proporción en los conjuntos de entrenamiento y validación, esta modificación nos ofrecían mejorías en la clasificación de la clase Naturaleza pero para la clase Hortensia no fue productiva, disminuyendo la precisión y aumentando el número de falsos positivos para datos nunca antes vistos por el modelo, por lo que se buscaron otras alternativas.

Se propuso un nuevo función de activación final llamada activación *Softmax* realiza la clasificación de forma categórica. El modelo con ocho capas de convolucion y activación final *Softmax* mejoro los resultados obtenidos con una activación binaria de tipo *Sigmoid*. La precisión del modelo *Sofmax* rebasa las expectativas que se tenían al inicio del proyecto que se estimaba obtener por lo menos una precisión cercana al 98%, la cual se consideraba suficiente para proceder con la implementación práctica del modelo CNN con conjunto de datos pequeños. Ahora desde este punto se buscara combinar el modelo optimizado para monitoreo de plantas invasivas junto a *OpenCV* buscando implementar una herramienta que permita identifique alguna tipo de planta invasiva propuesta capturada en imagen o video.

CAPÍTULO V

5 IMPLEMENTACIÓN PRÁCTICA

En este punto del proyecto ya contamos con el modelo optimizado con 8 Capas Convolucionales y activación final *Softmax*. Este modelo nos ofreció los mejores resultados para la identificación de plantas invasivas en el conjunto de datos hortensia/naturaleza. Ahora se busca utilizar el modelo de una forma práctica combinando las características de clasificación del modelo 8CC con activación *Softmax* con herramientas de la paquetería OpenCV, que permita identificar una planta invasiva propuesta e imprimir la detección y precisión en la imagen de estudio.

Para la implementación practica propondremos una nueva especia de planta invasiva que permita evaluar la capacidad de transferencia de aprendizaje del modelo, debido a que el modelo desarrollado hasta el momento, busca ser utilizado no solo para un tipo de especia sino que pueda adaptarse a diferentes especies con solo cambiar el conjunto de imágenes de entrenamiento.

La implementación de un nuevo conjunto de datos permitirá capturar las imágenes de una forma más acotada enfocándonos a la detección de plantas invasivas en una zona de estudio determinada, El acotamiento de zona busca mejorar los resultados de nuestro modelo previamente optimizado modificando hiperparametros. Las imágenes serán capturadas considerando dos formas utilizadas en los procesos de monitoreo:

- De forma directa por una persona.
- Utilizando un UAV (F450).

Considerar imágenes de diferentes perspectivas con diferentes métodos de captura permite generar un conjunto de datos más completo y aplicable a diferentes circunstancias de monitoreo que dependerá de la zona de estudio, tal es el caso de zonas de difícil acceso para el personal, donde es posible capturar imagen o video con un UAV que posteriormente pueda ser evaluado por nuestro modelo en tierra.

5.1 Conjunto de datos del modelo practico

La especia de planta seleccionada para este desarrollo práctico fue el arbusto Tepozán. Este arbusto es muy abundante en México y aunque no es toxico si es un arbusto resistente a diferentes climas, propagándose exitosamente y convirtiéndose en un problema debido a su elevada densidad, Imagen 5.1.

Imagen 5.1: Imagen de arbusto Tepozán.

Estas plantas tienden a crecer agresivamente mostrando gran competitividad por espacio, recursos en zonas de cultivo y albergando plagas que también pueden afectar otras especies, Imagen 5.2.

Imagen 5.2: Tepozán con plaga de gusano medidor (izquierda), Tepozán compitiendo por recursos (derecha).

Aparecen en terrenos cultivados, jardines, orillas de caminos, sitios perturbados y diversos ecosistemas, como se muestra en la Imagen 5.3.

Imagen 5.3: Afectación de arbusto Tepozán en carreteras y en construcciones.

Para la conformación del conjunto de datos se capturaron 3000 imágenes, divididas en 1500 imágenes que contienen arbustos de Tepozán en alguna zona de la imagen, y 1500 imágenes que no contiene arbustos de Tepozán, en su mayoría contienen árboles, follaje, bardas, construcciones arquitectónica y estructuras de protección de derrumbes en carreteras. Se realizó una selección y agrupación de las imágenes creando subconjuntos de datos. Las 1500 imágenes con arbustos de Tepozán de dividieron en 1000 imágenes para entrenamiento y 500 para validación, el conjunto de imágenes que no contienen Tepozán se agruparon de la misma forma. Finalmente se captaron 200 imágenes nunca antes vistas por el modelo que se utilizaran para la evaluar la precisión del modelo Tepozán. La Imagen 5.4 muestra una imagen de cada clase que se utilizara en el proceso de entrenamiento.

Imagen 5.4: Imagen arbusto Tepozán (izquierda) y zona bosque (derecha).

El conjunto de imágenes que contiene arbustos de Tepozán, en su mayoría fueron tomadas desde una perspectiva superior y una pequeña porción de forma frontal, se procuró que el arbusto se encuentra de forma clara en alguna región de la imagen. En el caso de imágenes sin arbusto de Tepozán se trató de considerar las zonas más comunes donde se propagan.

5.2 Pre-procesamiento de datos

Las imágenes del conjunto de datos tienen un tamaño de 1088x612 píxeles, como se realizó en el modelo de Hortensia, utilizar imágenes de estas dimensiones representa un gasto computacional elevado por lo que se aplicaron funciones que acondicionen las imágenes para el proceso de entrenamiento.

Las imágenes se encuentran con formato JPEG que se codifica en una cuadricula de pixeles con diferentes valores de intensidad. Sin embargo estas deben ser pre-procesadas adecuadamente en tensores de punto flotante previo a realizar el proceso de entrenamiento, siguiendo la siguiente secuencia de procedimientos:

1. Lee los archivos de imagen JPEG.
2. Decodifique el contenido JPEG en cuadrículas RGB de píxeles.
3. Convierta estos en tensores de punto flotante.
4. Normalizar los valores de píxel (entre 0 y 255) al intervalo [0, 1], esto se debe a que las redes neuronales prefieren tratar con pequeños valores de entrada.

5.3 Entrenamiento del modelo Tepozán

Para el proceso de entrenamiento utilizaremos la arquitectura con 8 Capas Convolucionales y activación final *Softmax*. Se realizaron cambios en la imagen de entrada, antes de entrar a la red se necesita realizar una redimensionamiento de [1088x612] pixeles a [224x224], esto con el objetivo de reducir gasto computacional. De igual forma el número de filtros para cada capa convolucional de la arquitectura se definió en 21 filtros.

Como el proceso de captura de datos se realizó de forma más controlada se pretende ingresar imágenes de una dimensión más grande comparada a los modelos anteriores (hortensia). Lo que se busca es que los filtros puedan abarcar una mayor área de la imagen durante el barrido de convolucion. Esta nueva implementación genera un mayor gasto computacional por tal motivo se redujo el número de filtros en cada capa convolucional. La Imagen 5.5 muestra la arquitectura del modelo Tepozán.

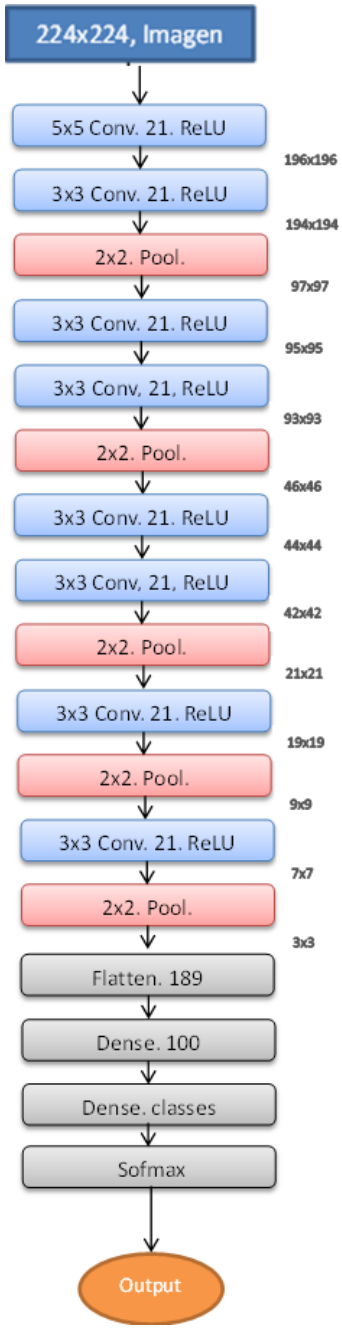

Imagen 5.5: Arquitectura CNN del modelo Tepozán con 8 capas convolucionales y activación Softmax.

Como se desea combinar el modelo CNN con funciones de visión en *Open CV*, es necesario utilizaran funciones más estandarizadas que puedan utilizarse en ambas paqueterías, para registrar la evolución de nuestro proceso de entrenamiento se utilizó una librería de graficación llamada *Matplotlib*. La librería *Matplotlib* es una biblioteca de trazado 2D en *Python* que produce cifras de calidad de publicación en una variedad de formatos y entornos interactivos de diversas plataformas.

El proceso de entrenamiento para el modelo Tepozán consistió en 100 épocas. Los resultados del proceso de entrenamiento se pueden observar en la Imagen 5.6.

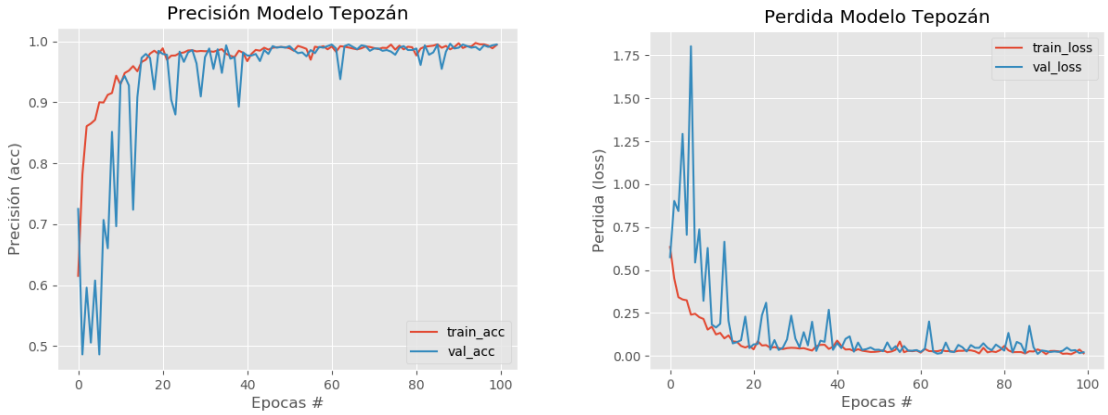

Imagen 5.6: Graficas de precisión y pérdida de los datos de entrenamiento y validación del modelo Tepozán.

En la gráfica de precisión se observa que los valores de entrenamiento con respecto a los datos de validación se regulan a las 20 épocas de entrenamiento, continua convergiendo los paramentos hasta alcanzar precisión superiores a 0.99 equivalente al 99 % de precisión, estos valores rebasa las expectativas con relacionadas a el puntaje de precisión obtenido con los modelos de Hortensia. Lo anterior indica un acierto en incrementar la dimensión en las imágenes de entrada y en acotar el tipo y calidad de nuestro conjunto de datos. En el caso de la pérdida del modelo se aprecia que se reduce su valor cercano a cero, indicando de forma global que nuestro modelo practico para detección de planta invasivas cuenta con una buena estructura que permite la transferencia de aprendizaje a diferentes especies de plantas invasivas.

5.4 Prueba de datos nunca antes vistos por el modelo Tepozán

Se tomaron 200 imágenes de datos que no se utilizaron en el proceso de entrenamiento, estas imágenes en forma general se tomaran en zonas cercanas donde fueron tomadas los datos de entrenamiento, y corresponden a diferentes arbustos de Tepozán con un entorno similar. Evaluando las 200 imágenes de evaluación, 100 imágenes que contienen arbustos de Tepozán y 100 de naturaleza en general, se obtuvieron los siguientes valores de precisión y pérdida para datos nunca antes vistos por el modelo Tepozán, Tabla 5.1.

Tabla 5.1 Parámetros de precisión y perdida del modelo inicial en imágenes nunca antes vistas por el modelo Tepozán.

# imágenes de Prueba	Precisión	Perdida
200	0.997	0.001

Evaluando las 200 imágenes obtenemos la siguiente matriz de confusión que muestra los errores de clasificación (falsos positivos), se obtuvieron cero falsos positivos del conjunto de 200 imágenes de evaluación. La matriz de confusión se muestra en la Imagen 5.7.

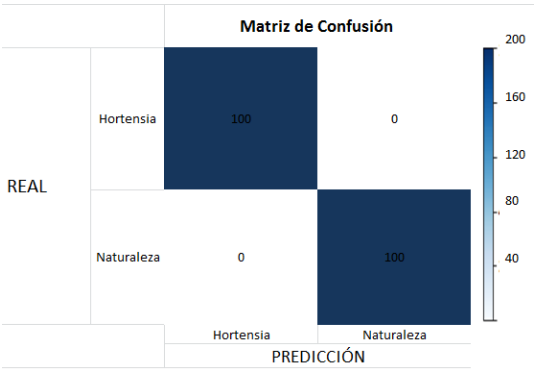

Imagen 5.7: Matriz de confusión probada con 200 imágenes nunca antes vistas por el modelo Tepozán.

5.4.1 Predicción en imagen nunca antes vista por el modelo utilizando Open CV

En esta sección mostraremos algunas predicciones de nuestro modelo Tepozán para imágenes nunca antes vistas por el modelo pertenecientes al conjunto dc datos de evaluación. Como se desea realizar una aplicación práctica en conjunto con Open CV, se realiza un acondicionamiento de la imagen para que pueda ser manipulada con el software *Python* y *Open CV*, la predicción será impresa directamente en la parte superior izquierda de imagen de estudio.

Para realizar la predicción se requiere dos argumentos:

- Modelo (Modelo Tepozán previamente entrenado).
- Entrada (Imagen que vamos a clasificar).

A partir de este punto, cargaremos la imagen y la procesaremos, se realiza una copia de ella, esto nos permite recuperar la imagen original más adelante e imprimir la predicción en ella. Con la imagen cargada es necesario normalizarla en un rango [0, 1], convirtiéndola en una matriz y agregando una dimensión adicional. Añadiendo una dimensión extra a la matriz a través de la paquetería *Numpy* y la función (*np . expand_dims*), permite que nuestra imagen tenga la forma [1 , ancho , alto, 3), correspondientes al formato utilizado en el paquete *Keras*. Esta dimensión agregada es importante porque es la forma en que *Keras* generaliza las imágenes para realizar el proceso de entrenamiento, por lo tanto si se desea realizar alguna predicción con el modelo Tepozán entrenado en *Keras* se requiere que tengan el mismo formato.

Lo siguiente paso fue cargar el modelo *Tepoza.h5* el cual contiene todos los pesos y filtros obtenidos en el proceso de entrenamiento del modelo. Finalmente se realiza la predicción de la imagen procesada y se imprimirá en la imagen original guardada.

Se definieron las etiquetas ("Tepozán" o "No Tepozán"), utilizando una función condicional y la probabilidad obtenida en la predicción de la imagen de evaluación, se creó la etiqueta que se mostrara en la esquina superior izquierda de las imágenes de salida.

La Imagen 5.8 muestra la imagen de salida del algoritmo de predicción utilizando Python, *Keras* y Open CV.

Imagen 5.8: Predicción modelo Tepozán, imagen nunca antes vista por el modelo.

Tomando cuatro muestras del conjunto de evaluación que contienen arbusto tepozán y que no se utilizaron en el proceso de entrenamiento, se obtuvieron resultados destacados. En las cuatro muestras se obtuvo predicción del 100 %. Las predicciones de estas imágenes de prueba se muestran en la Imagen 5.9.

Imagen 5.9: Predicción del modelo Tepozán en cuatro imágenes que contienen arbusto de Tepozán.

Tomando cuatro muestras de imágenes que no contienen arbustos de tepozán del conjunto de datos de evaluación no utilizadas en el proceso de entrenamiento se obtuvieron de igual forma resultados con precisión del 100%, estas predicciones se muestran en la Imagen 5.10.

Imagen 5.10: Predicción modelo Tepozán en cuatro imágenes que no contienen arbusto de Tepozán.

Para ambas clases [Tepozán/No Tepozán] la predicción obtenidas han demostrado ser buenos para la predicción de plantas invasivas en una zona acotada, resaltando la capacidad del aprendizaje profundo en especial la implementación de CNN para tareas de identificación.

5.4.2 Predicción para archivos de video

Para realizar predicciones a un archivo de video en lugar de una entrada fija correspondiente a una imagen se utilizara una entrada continua de imagen, es decir un archivo de video o un canal de video. Para realizarlo se utilizaron librerías creadas en *OpenCV* y en código *Python* para acceder a un canal de video *VideoStream* o un archivo de video gravado previamente.

El programa funciona realizando algo similar a lo realizado en la predicción de imágenes fijas, solo que para evaluar videos es necesario seccionarlo en *frames* utilizando la función (*vs.read*), que nos permite realizar el proceso de acondicionamiento del *frame* de video para que tenga un formato compatible al modelo Tepozán entrenado previamente. Después del pre-procesamiento de la imagen se pasa a través del modelo Tepo-

zán generando una predicción que se mostrara en la parte superior izquierda del *frame* en la pantalla de salida.

Para evaluar el programa se capturaron archivos de video de dos zonas donde se encontraban arbustos de tepozán, La primera corresponde a un campo plano donde abunda pasto y se aprecia un poco de escombro perteneciente a una construcción antigua. Se presentan cuatro imágenes generales de la secuencia de video captada obteniendo los resultados mostrados en la Imagen 5.11. Las dos figuras superiores se identificaron como imágenes que contiene arbustos de tepozán, en estas figuras se aprecia de forma clara las ramas y hojas del arbusto de tepozán, en cambio en las figuras inferiores el arbusto se encuentra en menor proporción y la predicción es correcta, no se encuentran suficiente características que relacionen con un arbusto de tepozán.

Imagen 5.11: Secuencia de imágenes de un video con prediccion del modelo Tepozan en una zona plana.

Se grabo otra zona diferente donde encontre arbustos de tepozan nunca antes vistos por el modelo, la zona correspone a muros de concreto que previenen derrumbes en carreteras. La Imagen 5.12 muestra una secuencia general de lo que el video capto, en las figuras superiores el arbusto se aprecia de forma clare generando una prediccion correcta con valores de presicion altos (100% y 98.54%), para las figuras inferiores se realiza un prediccion de arbusto de tepozan considerando que el arbusto se encuentra en una pequeña porcion de la imagen, esto se puede dever al fondo relativamente

neutro en color del muro de concreto. La segunda figura inferior muestra solo caracteristicas del muro de concreto y parte de la carretera, esta prediccion se realiza de forma correcta al no mostrar ningun arbusto de tepozan.

Imagen 5.12: Prediccion de una secuencia de imágenes de un video de costados de carretera.

Analizando los resultados obtenidos el modelo Tepozán ofrece buenas predicciones para diferentes zonas de muestreo de videos.

5.5 Análisis de Resultados.

Es importante resaltar que las 200 imágenes del conjunto de datos de evaluación, no se utilizaron en el proceso de entrenamiento, fueron tomadas de arbustos de Tepozán diferentes a los que se utilizaron en el proceso de entrenamiento, pero estos arbutos se encontraban en zonas cercanas.

Las imágenes de evaluación fueron capturadas de arbustos diferentes en las mismas regiones donde se capturaron las imágenes de los conjuntos de entrenamiento y validación. Aunque se trataban de arbustos diferentes, el entorno o fondo de las imágenes resulta ser similar. Esta similitud es la causa de obtener resultados de precisión tan altos.

El acotar una zona de estudio garantiza una mejora en los resultados que permite que aplicaciones CNN con conjunto de datos pequeños pueda obtener resultados similares a redes neuronales con grandes cantidades de datos, con la limitante de zona previamente establecida.

Analizando imágenes con arbustos de Tepozán con un fondo diferente como puede ser el cielo, el modelo genero resultados con una precisión menor. El modelo fue entrenado con imágenes tomadas desde una perspectiva superior y en su mayoría el fondo de la imagen está conformado por suelo con vegetación, muros de piedra, muros de concreto para evitar derrumbes y otros tipos de suelo. La reducción en la precisión se debe a que el modelo no cuenta con filtros que caractericen una zona de cielo en una imagen de prueba, sin embargo, si la imagen tiene zona que contiene arbustos de tepozán, el modelo genera predicciones correctas. La descripción anterior se muestra en la Imagen 5.13.

Imagen 5.13: Predicción modelo Tepozán de imagen con fondo diferente.

De igual forma analizando imágenes que no tiene nada que ver con la zona de estudio, es decir una imagen aleatoria de una región de suelo diferente puede ser difícil de clasificar ya que nuestro modelo no cuenta con la información suficiente para poder clasificarlo con alta precisión. Tal es el caso de la predicción de la imagen 5.14.

Imagen 5.14: Predicción modelo Tepozán de una zona diferente a la zona de estudio.

La predicción de la imagen anterior muestra una clasificación correcta de la clase [No Tepozan], pero con un puntaje de predicción mucho menor comparado con las predicciones de imágenes de los datos de evaluación. Esto muestra la limitante principal en la utilización de Redes Neuronales Convolucionales con conjuntos de datos pequeños, la cual es que no se puede considerar todos los tipos de suelos posibles requeriría una cantidad de información muy grande, y un gasto computacional considerable, la necesidad de equipo computacional de última generación aumentando el costo y tiempo de análisis. Por lo tanto si se desea realizar una aplicación de identificación de algún planta y se conoce cuál es el entorno donde se propaga, realizar una acotación de zona de estudio permite implementar modelos CNN con conjuntos de imágenes pequeños resulta una alternativa fiable para la identificación oportuna de planta invasivas a bajo costo.

CAPÍTULO VI

6 CONCLUSIONES Y TRABAJO FUTURO.

6.1 Conclusiones

En este capítulo, revisaremos nuestros resultados y proporcionaremos algunas observaciones finales. Nos propusimos implementar una Red Neuronal Convolucional desde cero, enfocada a la identificación de plantas invasivas utilizando conjuntos de datos pequeños. Implementar un modelo inicial, optimizarlo modificando diversos parámetros y explorar posibles mejoras.

Nuestra primer implementación fue un modelo CNN con cuatro capas de consolación y se utilizó un conjunto de datos de 3000 imágenes, como se esperaba los resultados de predicción no fueron muy alentadores pero durante su desarrollo pude entender más a fondo lo que representa una Red Neuronal de Convoluicon, Para esta tesis, no tuvimos acceso a servidores o una GPU de gama alta utilizada para fines de investigación, por lo que utilizar pocos datos e implementar modelo en una computadora portátil de consumo común eran los principales limitaciones pero a la ves representaban los objetivos planteados en este trabajo de investigación. Con una perspectiva más amplia del funcionamiento de una CNN y una modelo inicial buscamos alternativas que permitieran obtener mejores resultados, para solventar el número de datos se utilizaron funciones de la paquetería *Keras* que generaban imágenes alternas aplicando operaciones de rotación, barrido, traslación, etc. sobre el mismo conjunto de datos, esto permitió incrementar el número de datos, aunque estos fueran generados del mismo conjunto base representaron una alternativa. Se desarrolló un nuevo modelo CNN con extensión de datos que ofreció mejores resultados, en cuestión de clasificación de flores de hortensia mejoro la predicción para datos nunca antes vistos por el modelo, sin embargo continuaba teniendo problemas para clasificar imagen donde la flor se encontraba rodeada por vegetación o alejada en la escena.

Para solucionar esta problemática se desarrollaron diversos experimentos modificando parámetros del modelo, se incrementaron el número de capas convolucionales los modelos con mejores resultados fueron con 6 capas y con 8 capas obteniendo valores de precisión superiores al 90%. El incremento en la precisión se debió al incremento de filtros que describían de mejor manera nuestras clases. También modificamos la propor-

ción de los conjuntos de entrenamiento pero solo mejoro la clasificación de clase Naturaleza por lo que no se continuó modificando los subconjuntos. Finalmente para el conjunto Hortensia se utilizó el modelo con mejores resultados hasta el momento correspondiente a el modelo con 8 capas convolucionales, no se continuo incrementando el número de capas considerando la restricción de equipo de cómputo por lo que se cambió la función de activación final a activación *Softmax* que genera clasificaciones de forma categórica. Este último modelo del conjunto Hortensia ofreció los mejores resultados en precisión, pero particularmente clasifico correctamente todas las muestras del conjunto de evaluación,

Se encontraron restricciones del modelo hortensia, como son datos repetidos, imágenes con objetos no naturales que contaban con pocas muestras y por lo tanto eran las imágenes que obtenía menor porcentaje de precisión. Para la implementación practica se buscó probar la trasferencia de aprendizaje de nuestro modelo optimizado, ya que esta implementación busca utilizarse para identificación de diferentes especies de plantas invasivas por lo que se propuso una nueva planta invasiva. La planta seleccionada fue el Tepozán de la cual se realizó un estudio de las zonas comunes donde suelen aparecer, en base a este estudio se acoto y se definió la forma de capturar las imágenes procurando no utilizar imágenes lo más diversas posible. Entrenando el modelo obtuvimos resultados superiores a lo esperado ($> 99\%$), Para datos nunca antes vistos por el modelo no se obtuvieron falsos positivos, se realizaron pruebas a archivos imagen y archivos de video con arbustos de tepozán y otros objetos obteniendo buenos resultados siempre y cuando nos restringiéramos al tipo de zona acotada, en nuestro casi zonas boscosas, bardas, casas y estructuras para prevenir derrumbes en carretera.

La principal limitante se aprecia al probar imágenes que se salen de la zona acotada en el caso de una imagen con un fondo diferente como es el cielo, si la imagen tiene arbustos de tepozán de forma clara el modelo tiene las condiciones de clasificarlo correctamente con una precisión menor como se observó en el capítulo anterior.

Finalmente se concluyó que la implementación de modelo CNN para la identificación de plantas invasivas con restricciones de datos y equipo de cómputo, es posible realizarla de forma funcional, practica, con un menor costo y accesible para desarrolladores que no cuentan con la experiencia necesaria en el campo de las ciencias de la computación.

6.2 Trabajo futuro

Como trabajo futuro se centraría en la implementación de los algoritmos desarrollados en la presente investigación en un UAV, de tal forma que se pudiese proponer un dron autónomo para el monitoreo con la alternativa de utilizar transferencia de video por internet permitiendo a los monetarista analizar diferentes zonas desde sus laboratorios.

A CÓDIGOS

En esta sección anotaremos todos los códigos utilizados en este trabajo de investigación

A1. Modelo inicial con extensión de datos

```
# Se importan las librerias nesesarias para el desarrollo del modelo.
from os import environ, chdir
from keras.preprocessing.image import ImageDataGenerator#Setting Image and Data Generators
from keras import models, layers, optimizers, callbacks
environ['TF_CPP_MIN_LOG_LEVEL'] = '3'chdir(r'C:\Users\Ozcar\Desktop\CNN\hydrangea')

# configuracion del generador de imagen Image and Data Generators para datos de entrenamiento
train_idg = ImageDataGenerator(rescale=1. / 255,
                                    zoom_range=[1.0, 1.25],
                                    width_shift_range=0.1,
                                    height_shift_range=0.1,
                                    fill_mode='reflect')

train_g = train_idg.flow_from_directory(directory=r'data2\train',
                                            target_size=(120, 120),
                                            class_mode='binary',
                                            batch_size=20)

# Configuración del generador de imagen  para datos de validación
valid_idg = ImageDataGenerator(rescale=1. / 255)

valid_g = valid_idg.flow_from_directory(directory=r'data2\valid',
                                            target_size=(120, 120),
                                           class_mode='binary',
                                           batch_size=20)
# Architecture CNN
my_model = models.Sequential()
    #cargar imagen de entrada y primer capa convolucional
my_model.add(layers.Conv2D(filters=32, kernel_size=(3, 3), strides=(1, 1),
                                  input_shape=(120, 120, 3)))
my_model.add(layers.Activation('relu'))
my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    #segunda  capa
my_model.add(layers.Conv2D(filters=64, kernel_size=(3, 3), strides=(1, 1)))
my_model.add(layers.Activation('relu'))
my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    #Tercera capa
my_model.add(layers.Conv2D(filters=64, kernel_size=(3, 3), strides=(1, 1)))
my_model.add(layers.Activation('relu'))
```

```
40 my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
41     #cuarta capa
42 my_model.add(layers.Conv2D(filters=128, kernel_size=(3, 3), strides=(1, 1)))
43 my_model.add(layers.Activation('relu'))
44 my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
45     #FCL
46 my_model.add(layers.Flatten())
47
48 my_model.add(layers.Dense(units=100))
49 my_model.add(layers.Activation('relu'))
50
51 my_model.add(layers.Dense(units=1))
52 my_model.add(layers.Activation('sigmoid'))
53
54 # Funciones de perdida y optimizadores.
55 compile = my_model.compile(optimizer=optimizers.sgd(lr=0.1),loss= 'binary_crossentropy',
56 metrics=['accuracy'])
57
58 # Configuración Callbacks
59 check_p = callbacks.ModelCheckpoint(filepath='hydrangea_cnn_{val_acc:.2f}.model',
60                                     monitor='val_acc', verbose=1,
61                                     save_best_only=True, save_weights_only=False)
62 reduce_lr = callbacks.ReduceLROnPlateau(monitor='val_acc', factor=0.9, patience=3,
63                                         verbose=1, cooldown=2)
64
65 #ahora lo que necesito es enlistar (small_list) indicándole a Keras los callback que se utilizaron para
66 eso se salvaron en las variables llamadas (check_p, reduce_lr ).
67
68 callb_l = [check_p, reduce_lr]
69
70 #Opciones de entrenamiento
71 history =my_model.fit_generator(generator=train_g,steps_per_epoch=100,epochs=50,
72 valida     tion_data=valid_g,validation_steps=50)
73
74 # Listar e imprimir los parámetros de entrenamiento.
75 print(history.history.keys())
76
77 import matplotlib.pyplot as plt
78
79 acc = history.history['acc']
80 val_acc = history.history['val_acc']
81 loss = history.history['loss']
82 val_loss = history.history['val_loss']
83 epochs = range(1, len(acc) + 1)
84
85 plt.plot(epochs, acc, 'bo', label='Training acc')# list all data in history
86 print(history.history.keys())
87 plt.plot(epochs, val_acc, 'b', label='Validation acc')
88 plt.title('Training and validation accuracy')
89 plt.legend()
90 plt.figure()
```

```
91
92  plt.plot(epochs, loss, 'bo', label='Training loss')
93  plt.plot(epochs, val_loss, 'b', label='Validation loss')
94  plt.title('Training and validation loss')
95  plt.legend()
96  plt.show()
```

A2. Arquitectura del modelo con 8 Capas de convolucion y activación Sigmoid

```
1   # Arquitectura modelo 8CC y activación Sigmoid
2   my_model = models.Sequential()
3   my_model.add(layers.Conv2D(filters=16, kernel_size=(2, 2), strides=(1, 1),
4                                       input_shape=(120, 120, 3)))
5   my_model.add(layers.Activation('relu'))
6   my_model.add(layers.Conv2D(filters=32, kernel_size=(2, 2), strides=(1, 1)))
7   my_model.add(layers.Activation('relu'))
8   my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
9   my_model.add(layers.Conv2D(filters=32, kernel_size=(2, 2), strides=(1, 1)))
10  my_model.add(layers.Activation('relu'))
11  my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
12
13  my_model.add(layers.Dropout(rate=0.5))
14
15  my_model.add(layers.Conv2D(filters=64, kernel_size=(2, 2), strides=(1, 1)))
16  my_model.add(layers.Activation('relu'))
17  my_model.add(layers.Conv2D(filters=64, kernel_size=(2, 2), strides=(1, 1)))
18  my_model.add(layers.Activation('relu'))
19  my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
20
21  my_model.add(layers.Conv2D(filters=128, kernel_size=(2, 2), strides=(1, 1)))
22  my_model.add(layers.Activation('relu'))
23  my_model.add(layers.Conv2D(filters=128, kernel_size=(2, 2), strides=(1, 1)))
24  my_model.add(layers.Activation('relu'))
25  my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
26
27  #my_model.add(layers.Dropout(rate=0.5))
28  my_model.add(layers.Conv2D(filters=256, kernel_size=(2, 2), strides=(1, 1)))
29  my_model.add(layers.Activation('relu'))
30  my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
31
32  # Red conpletamente conectada.
33  my_model.add(layers.Flatten())
34  my_model.add(layers.Dense(units=100))
35  my_model.add(layers.Activation('relu'))
36  my_model.add(layers.Dense(units=10))
37  my_model.add(layers.Activation('relu'))
38  my_model.add(layers.Dense(units=1))
39  my_model.add(layers.Activation('sigmoid'))
```

A3. Arquitectura Modelo Softmax

```
1  # CNN Architecture
2  my_model = models.Sequential()
3  my_model.add(layers.Conv2D(filters=16, kernel_size=(2, 2), strides=(1, 1),
4                             input_shape=(120, 120, 3)))
5  my_model.add(layers.Activation('relu'))
6
7  my_model.add(layers.Conv2D(filters=32, kernel_size=(2, 2), strides=(1, 1)))
8  my_model.add(layers.Activation('relu'))
9  my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
10 my_model.add(layers.Conv2D(filters=32, kernel_size=(2, 2), strides=(1, 1)))
11 my_model.add(layers.Activation('relu'))
12 my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
13
14 my_model.add(layers.Dropout(rate=0.3))
15
16 my_model.add(layers.Conv2D(filters=64, kernel_size=(2, 2), strides=(1, 1)))
17 my_model.add(layers.Activation('relu'))
18 my_model.add(layers.Conv2D(filters=64, kernel_size=(2, 2), strides=(1, 1)))
19 my_model.add(layers.Activation('relu'))
20 my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
21
22 my_model.add(layers.Dropout(rate=0.3))
23
24 my_model.add(layers.Conv2D(filters=128, kernel_size=(2, 2), strides=(1, 1)))
25 my_model.add(layers.Activation('relu'))
26 my_model.add(layers.Conv2D(filters=128, kernel_size=(2, 2), strides=(1, 1)))
27 my_model.add(layers.Activation('relu'))
28 my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
29
30 my_model.add(layers.Dropout(rate=0.3))
31
32 my_model.add(layers.Conv2D(filters=256, kernel_size=(2, 2), strides=(1, 1)))
33 my_model.add(layers.Activation('relu'))
34 my_model.add(layers.MaxPooling2D(pool_size=(2, 2)))
35
36 # Red conpletamente conectada.
37 my_model.add(layers.Flatten())
38
39 my_model.add(layers.Dense(units=100))
40 my_model.add(layers.Activation('relu'))
41 my_model.add(layers.Dense(units=2))
42 my_model.add(layers.Activation('softmax'))
```

A4. Arquitectura modelo CNN Tepozán

```
# Inicializacion del modelo
        model = Sequential()
        inputShape = (height, width, depth)

# Arquitectura modelo CNN Tepozan
        model.add(Conv2D(filters=21, kernel_size=(2, 2), strides=(1, 1),input_shape=inputShape))
        model.add(Activation("relu"))

        #Segunda capa
model.add(Conv2D(filters=21, kernel_size=(2, 2), strides=(1, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
        #Tercera capa
model.add(Conv2D(filters=21, kernel_size=(2, 2), strides=(1, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

        #Cuarta capa
 model.add(Dropout(rate=0.2))
model.add(Conv2D(filters=21, kernel_size=(2, 2), strides=(1, 1)))
model.add(Activation('relu'))
       #Quinta capa
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(filters=21, kernel_size=(2, 2), strides=(1, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

        #Sexta capa
model.add(Dropout(rate=0.2))
model.add(Conv2D(filters=21, kernel_size=(2, 2), strides=(1, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
        #Septima capa
model.add(Conv2D(filters=84, kernel_size=(2, 2), strides=(1, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
        #Octava capa
model.add(Conv2D(filters=21, kernel_size=(2, 2), strides=(1, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Capa completamente conectada
        model.add(Flatten())
        model.add(Dense(500))
        model.add(Activation("relu"))


```

```
48  # Clasificador Softmax
49          model.add(Dense(classes))
50          model.add(Activation("softmax"))
```

A5. Modelo CNN Tepozán para predicción de imagen

```
1   # Importar paquetes necesarios
2   from keras.preprocessing.image import img_to_array
3   from keras.models import load_model
4   import numpy as np
5   import argparse
6   import imutils
7   import cv2
8
9   # Cargar la imagen de entrada
10  image = cv2.imread("Cam 3d.jpg")
11  orig = image.copy()
12
13  # Pre-procesamiento de la imágenes
14  image = cv2.resize(image, (224, 224))
15  image = image.astype("float") / 255.0
16  image = img_to_array(image)
17  image = np.expand_dims(image, axis=0)
18
19  # Cargar el modelo CNN Tepozan
20  print("[INFO] loading network...")
21  model = load_model("Tepozan.h5")
22
23  # classify the input image
24  (nature, hortensia) = model.predict(image)[0]
25
26  # Crear la etiqueta
27  label = "Tepozan" if hortensia > nature else "No Tepozan"
28  proba = hortensia if hortensia > nature else nature
29  label = "{}: {:.2f}%".format(label, proba * 100)
30
31  # Colocar la etiqueta de predicción en la imagen
32  output = imutils.resize(orig, width=800)
33  cv2.putText(output, label, (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,
34                                      0.7, (0, 0, 255), 2)
35  # Mostrar la imagen de salida
36  cv2.imshow("Output", output)
37  cv2.waitKey(0)
38
39  # Cerrar las ventanas abiertas
40  cv2.destroyAllWindows()
```

A6 Modelo CNN Tepozán para predicción en video

```
1  # paqueteria requerida
2  from keras.preprocessing.image import img_to_array
3  from imutils.video import FPS
4  from keras.models import load_model
5  import numpy as np
6  import argparse
7  import imutils
8  import cv2
9  #cargar la dirección del modelo CNN Tepozan.
10 model_path = "Tepozan.h5"
11 model = load_model(model_path )
12
13 # inicializar el número de frames que contienen arbustos de tepozán y implemento de un gatillo de
14 alarma
15 TOTAL_CONSEC = 0
16 TOTAL_THRESH = 1
17
18 # Inicializacion de alarma de Tepozan
19 TEPOZAN = False
20
21 #Abrir el punter para video stream and start the FPS timer
22 stream = cv2.VideoCapture('ex8.mp4')
23
24 # Crear un loop sobre los fremes del archive de video
25 while True:
26
27         (grabbed, frame) = stream.read()
28         if not grabbed:
29                 break
30         frame = imutils.resize(frame, width=680)
31         frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
32
33
34 # Preparación de imagen (frame) para clasificación con modelo CNN Tepozán
35         image = cv2.resize(frame, (224, 224))
36         image = image.astype("float") / 255.0
37         image = img_to_array(image)
38         image = np.expand_dims(image, axis=0)
39
40         (notTepozan, Tepozan) = model.predict(image)[0]
41         label = "Not Tepozan"
42         proba = notTepozan
43
44         if Tepozan > notTepozan:
45                 label = "Tepozan"
46                 proba = Tepozan
47
48                 TOTAL_CONSEC += 1
```

```
                if not TEPOZAN and TOTAL_CONSEC >= TOTAL_THRESH:
                        # indicate that santa has been found
                        TEPOZAN = True
        else:
                TOTAL_CONSEC = 0
                TEPOZAN = False

        label = "{}: {:.2f}%".format(label, proba * 100)
        frame = cv2.putText(frame, label, (10, 25),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
        # show the output frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        if key == ord("q"):
                break

# Finalizar la ejecuciónon
print("[INFO] cleaning up...")
stream.release()
cv2.destroyAllWindows()
```

BIBLIOGRAFIA

[1] Isabel López Zamora, El monitoreo de las plantas invasoras, Revista de divulgación científica y Tecnológica de la universidad de Veracruz, Volumen XXI, Numero 1, Enero-Abril 2018. https://www.uv.mx/cienciahombre/revistae/vol21num1/articulos/monitoreo/index.html.

[2]Aguirre Muñoz, A., R. Mendoza Alfaro. Especies exóticas invasoras: impactos sobre las poblaciones de flora y fauna, los procesos ecológicos y la economía, en Capital natural de México, vol. II: Estado de conservación y tendencias de cambio. Conabio, 2009, México.(pages 277-318).

[3] Walter A, Liebisch F, Hund A. Plant phenotyping: from bean weighing to image analysis. Plant Methods 2015;11(1):1–11.

[4] Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T, eds. Computer Vision – ECCV 2014. Cham, Switzerland: Springer International Publishing, 2014:818–33.

[5] Kaggle Competition, Invasive Species Monitoring Identify images of invasive hydrangea, 2017. https://www.kaggle.com/c/invasive-species-monitoring.

[6] Salvador Mandujano, Alberto Rísquez Valdepeña, INECOL, El Instituto de Ecologia, Drones: observación de fauna y de hábitats desde el aire, 2017. https://www.inecol.mx/inecol/index.php/es/ct-menu-item-25/ct-menu-item-27/17-ciencia-hoy/563-drones-observacion-de-fauna-y-de-habitats-desde-el-aire.

[7] Salvador Mandujano, Margarita Mulero-Pázmány, UNA NUEVA TECNOLOGÍA PARA EL ESTUDIO Y MONITOREO DE FAUNA Y HÁBITATS, Red Biología y Conservación de Vertebrados, Instituto de Ecología A.C, November 2017.

[8] J. A. Luna-González. S. Paris, M.Nakano-Miyatake, D. Robles-Camarillo, Comparación de Arquitecturas de Redes Neuronales Convolucionales para la Clasificación de Imágenes de Ojos: Simposio Iberoamericano Multidisiplinario de Ciencias e Ingenieria, 2016.

[9] Olivetto Rendón Alexis Josué, Villanueva Tavira Jonathan, Valdez Martínez Jorge Salvador, Magadán Salazar Andrea, Aprendizaje Profundo para la Identificación de Objetos en Robótica Móvil, Artículo seleccionado del4° Congreso Internacional de Sistemas Embebidos (ICES18), Enero 2018, Vol. 7, No. 1, páginas 1 – 9.

[10] Guobin Chen, Tony X. Han, Zhihai He. DEEP CONVOLUTIONAL NEURAL NETWORK BASED SPECIES RECOGNITION FORWILD ANIMALMONITORING, Univeristy of MissouriElectitral and Comptuer Engineering Department Columbia, MO 65203, USA.

[11] Jun-Li XuDa-Wen SunEmail author. Computer Vision Detection of Salmon Muscle Gaping Using Convolutional Neural Network Feature , Food Analytical Methods, January 2018, Volume 11, Issue 1, pp 34–47.
https://link.springer.com/article/10.1007/s12161-017-0957-4.

[12] Sambuddha Ghosal, David Blystone, Asheesh K. Singh. Interpretable Deep Learning applied to Plant Stress Phenotyping, Iowa State University, Mechanical Engineering and Agronomy Department, arXiv:1710.08619v3 [stat.ML] 28 Oct 2017.

[13] Michael P. Pound1, Jonathan A. Atkinson2, Deep machine learning provides state-of-the-art performance in image-based plant phenotyping, GigaScience, 6, 2017, 1–10.

[14] Nadia Shakoor, Scott Lee and Todd C Mockler, High throughput phenotyping to accelerate crop breeding and monitoring of diseases in the field, Current Opinion in Plant Biology 2017, 38:184–192.

[15] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 Pages 1097-1105.

[16] PierreBarréa, Ben C.Stöverb, Kai F.Müllerb, Volker Steinhagea, LeafNet: A computer vision system for automatic plant species identification, Ecological Informatics Volume 40, July 2017, Pages 50-56.

[17] Enquhone Alehegn, Maize Leaf Diseases Recognition and Classification Based on Imaging and Machine Learning Techniques, International Journal of Innovative Research in Computer and Communication Engineering (A High Impact Factor, Monthly, Peer Reviewed Journal) Vol. 5, Issue 12, December 2017.

[18] Konstantinos P. Ferentinos. Deep learning models for plant disease detection and diagnosis; Computers and Electronics in Agriculture Volume 145, February 2018, Pages 311-318.

[19] Sharada Prasanna Mohanty, David Hughes, and Marcel Salathé. Using Deep Learning for Image-Based Plant Disease Detection; Digital Epidemiology Lab, EPFL, Switzerland; School of Life Sciences, EPFL, Switzerland; Department of Entomology, College of Agricultural Sciences, Penn State University, USA; April 15, 2016.

[20] Sue Han Lee, Chee Seng Chan, Paul Wilkin, Paolo Remagnino.
DEEP-PLANT: PLANT IDENTIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS, Kingston University, United Kingdom.

[21] Yu Sun, Yuan Liu, Guan Wang, and Haiyan Zhang. Deep Learning for Plant Identification in Natural Environment; Computational Intelligence and Neuroscience Volume 2017, Article ID 7361042, 6 pages. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5458433/

[22] Alexis Jolya Hervé Goëaua Pierre Bonnetb Vera Bakića Julien Barbec Souheil SelmiaItheri Yahiaouid Jennifer Carrée Elise Mouyssete Jean-François Molinof Nozha Boujemaag Daniel Barthélémyh. Interactive plant identification based on social image data; Ecological Informatics Volume 23, September 2014, Pages 22-34.

[23] Jackson Baron ORCID Icon, D.J Hill ORCID Icon & H Elmiligi.Combining image processing and machine learning to identify invasive plants in high-resolution images; International Journal of Remote Sensing Volume 39, 2018 - Issue 15-16: Unmanned Aerial Systems (UAS) for Environmental.

[24] Bálint Pál Tóth, Márton Osváth, Dávid Papp, Gábor Szűcs. Deep Learning and SVM Classification for Plant Recognition in Content-Based Large Scale Image Retrieval; Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics Magyar Tudósok krt. 2., H-1117, Budapest, Hungary.

[25] Shai Shalev-Shwartz, Shai Ben-David. Understanding MachineLearning: from theory to Algorithms. Cambridge University Press, USA, 2014, (p 19-21).

[26] Bishop, C.M, Pattern Recognition and Machine Learning (Informa-tion Science and Statistics). Springer-Verlag New York, Inc., Secaucus,NJ, USA, 2006.(p 2,3,4,9,41,115,203).

[27] Goodfellow, I., Bengio, Y., and Courville, A. Deep Learning. MIT Press, 2016. (pages 2-5, 171-177) http://www.deeplearningbook.org.

[28] Rojas R. Neural Networks - A Systematic Introduction). Springer -Verlag, Berlin, New-York, 1996.

[29] Huang, K-Y. (2007) Application of artificial neural network for detecting Phalaenopsis seedling diseases using color and texture features. Electron. Agric. 57, p 3–11.

[30] Long L. N., and Gupta, A. Scalable massively parallel artificial neural networks. Journal of Aerospace Computing, Information, and Communication, 2008, p 3-15.

[31] Lecun Y, Bottou L, Bengio Y et al. Gradient-based learning applied to document recognition. Proc IEEE 1998.

[32]LISA Lab,Deep learning Tutorial: Release 0.1.University of Montreal, September 2015, p 35.

[33] Hornik K. Approximation capabilities of multilayer feedforward networks. Neural networks, 1991, p 251-257.

[34] Michael Nielsen. Chapter 2: How the backpropagation algorithm works. http://neuralnetworksanddeeplearning. com/chap2.html. 2017 (p 137, 141).

[35] Andrew Ng. Machine Learning.https://www.coursera.org/learn/machine-learning.(p 88, 132, 137, 141).

[36] Andrej Karpathy. CS231n: Convolutional Neural Networks for Visual Recognition. http://cs231n.stanford.edu/. 2016 (p 57, 84, 94, 106, 137, 142).

[37] Szeliski R. Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

[38] Sebe N. Machine learning in computer vision, vol. 29. Springer Science & Business Media, 2005.

[39] Huang, T. Computer vision: Evolution and promise. CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH-REPORTS-CERN (1996), p 21-26.

[40] Girshick R. Fast R-CNN. IEEE International Conference on Computer Vision. 2015, p 1440-1448.

[41] Uijlings J. R, Van de Sande, K. E, Gevers T, and Smeulders, A. W. M. Selective search for object recognition. International Journal of Computer Vision, 2013, p 154-171.

[42] LeCun Y, Boser B, Denker J. S, Henderson D, Howard R. E, Hubbard W, and Jackel L. D. Backpropagation applied to handwritten zip code recognition. Neural computation,1989, p 541-551.

[43] Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In: Fleet D, Pajdla T, Schiele B, TuytelaarsT, eds. Computer Vision – ECCV 2014. Cham, Switzerland:Springer International Publishing, 2014.

[44] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, eds. Advances in Neural Information Processing Systems 25. New York, USA: Curran Associates, Inc., 2012.

[45] Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning-based sequence model. Nat Methods 2015;12(10):931–4.

[46] Esteva A, Kuprel B, Novoa RA et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature 2017; 542(7639): p115–8.

[47] Fukushima, K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. Neural networks, 1988, p 119-130.

[48] Hubel, D. H., and Wiesel, T. N. Receptive fields and functional architecture of monkey striate cortex. The Journal of Physiology, 1968, p 215-243.

[49] Marr, D., and Hildreth, E. Theory of edge detection. Proceedings of the Royal Society of London B: Biological Sciences 207, 1167 (1980), p 187-217.

[50] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks

[51] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, p 253–256. IEEE, 2010.

[52] «CS231n Convolutional Neural Networks for Visual Recognition». [En línea]. Disponible en: http://cs231n.github.io/convolutional-networks/.

[53] Ren S., He K., Girshick R., and Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, 2015, p. 91-99.

[54] Manning P. Co, Deep Learning with Python, Francois Chollet, 2018: Python, 2018, (p 117-138)