



EDUCACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación
y Desarrollo Tecnológico

Tesis de Maestría

Sistema de visión artificial para la detección de
plantas enfermas mediante aprendizaje profundo

presentada por

ING. Andros Meraz Hernández

como requisito para la obtención del grado
de

Maestría en Ciencias de la Computación

Directora de tesis

Dra. Andrea Magadán Salazar

Codirector de tesis

Dr. Jorge Alberto Fuentes Pacheco

Cuernavaca, Morelos, **04/marzo/2022**

OFICIO No. DCC/017/2022
Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

DR. CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del C. ANDROS MERAZ HERNÁNDEZ, con número de control M20CE040, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado "SISTEMA DE VISIÓN ARTIFICIAL PARA LA DETECCIÓN DE PLANTAS ENFERMAS MEDIANTE APRENDIZAJE PROFUNDO", y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.



DRA. ANDREA MAGADÁN SALAZAR
Directora de tesis



DR. RAÚL PINTO ELÍAS
Revisor



DR. JORGE ALBERTO FUENTES PACHECO
Codirector de tesis



DR. NIMROD GONZÁLEZ FRANCO
Revisor

C.c.p. Depto. Servicios Escolares.
Expediente / Estudiante
JGGS/ibm

Cuernavaca, Mor., 10/marzo/2022
No. De Oficio: SAC/55/2022
Asunto: Autorización de impresión de tesis

ANDROS MERAZ HERNÁNDEZ
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS
DE LA COMPUTACIÓN
PRESENTE

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado "SISTEMA DE VISIÓN ARTIFICIAL PARA LA DETECCIÓN DE PLANTAS ENFERMAS MEDIANTE APRENDIZAJE PROFUNDO", ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

ATENTAMENTE
Excelencia en Educación Tecnológica®
"Educación Tecnológica al Servicio de México"



DR. CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

C. c. p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares



CMAZ/CHG

DEDICATORIAS

A mis padres Julio Meraz y Anita Hernández por estar siempre presentes en mí día a día, apoyándome incondicionalmente en todos los aspectos y por brindarme siempre lo mejor. Lamentablemente mi mamá fue una víctima de la pandemia y trascendió antes de que le entregara mi título de maestría...

¡Mamá! Donde quieras que te encuentres y papá, les dedico este trabajo fruto de todos sus sacrificios realizados, esperanzas añoradas y por todos los tiempos complicados que atravesamos en su momento y logramos salir adelante.

A mi abuelo Julio Meraz Lagunas por haberme deseado buenos deseos y una vida llena de éxitos antes de partir.

AGRADECIMIENTOS

A Dios por permitir que esté cumpliendo con un logro más en mi vida y por estar juntos en familia en estos tiempos difíciles de pandemia, gracias Dios!

Al Tecnológico Nacional de México, campus Cenidet por haberme brindado la oportunidad de continuar con mis estudios y superación académica.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por la oportunidad de formar parte del programa de becas para estudiar la Maestría en Ciencias de la Computación.

A cada uno de mis profesores del Cenidet por su tiempo, asesorías, apoyo y por compartir sus conocimientos.

A mi comité: el Dr. Raúl Pinto Elías y el Dr. Nimrod González Franco por permitir que este trabajo de tesis culmine y por todos sus comentarios, observaciones que me ayudarán a ir creciendo poco a poco y poder llegar a ser un profesional como ustedes.

Agradecer especialmente a la Dra. Andrea Magadán Salazar por haber depositado su confianza desde aquella entrevista que tuvimos, quien finalmente se convirtió en mi guía durante estos dos años. Agradezco mucho su atención, tiempo, dedicación y trabajo en equipo, gracias a usted estoy aquí cumpliendo un logro más.

Agradecer de igual manera al Dr. Jorge Alberto Fuentes Pacheco la invitación a sus clases, sus tiempos extras dedicados, sus aportaciones y comentarios apreciables para que esta tesis fuese posible, muchas gracias por ser mi otro guía durante estos dos años.

A mis profesores del Tecnológico Nacional de México, campus Zacatepec: Dra. Josefina Sámano Galindo, M.C. Mario Humberto Tiburcio Zúñiga y M.C. José Antonio Vázquez Santa Anna por el apoyo hasta el último momento.

Contenido

CAPÍTULO 1 INTRODUCCIÓN	1
1.1 Introducción.....	1
1.2 Análisis del problema.....	1
1.3 Complejidad del problema.....	2
1.4 Objetivos.....	2
1.5 Alcances.....	2
1.6 Limitaciones.....	2
1.7 Justificación y beneficios.....	3
1.8 Cultivos seleccionados.....	3
1.8.1 Patologías del jitomate.....	3
1.8.2 Patologías del maíz.....	6
1.9 Metodología de solución.....	7
1.10 Contenido.....	9
CAPÍTULO 2 ESTADO DEL ARTE	10
2.1 Antecedentes.....	10
2.2 Revisión de artículos.....	11
2.3 Análisis de artículos.....	54
CAPÍTULO 3 MARCO TEÓRICO	55
3.1 Patologías en las plantas.....	55
3.2 Desordenes fisiológicos en las plantas.....	56
3.3 Bancos de imágenes.....	58
3.4 Inteligencia artificial.....	60
3.5 Redes neuronales.....	60
3.6 Visión artificial.....	63
3.7 Métricas de evaluación.....	64
3.8 Lenguaje de programación, librerías y entornos de desarrollo.....	66
CAPÍTULO 4 ANÁLISIS Y DISEÑO DEL SISTEMA	69
4.1 Técnicas.....	69
4.1.1 Pre procesamiento.....	69
4.1.2 Aumento de datos.....	71
4.1.3 Hiperparámetros.....	71

4.1.4 Validación cruzada.....	71
4.1.5 Transferencia de aprendizaje.....	71
4.1.6 Ajuste fino	73
4.2 Diseño del sistema.....	74
4.2.1 Versión Móvil.....	74
4.2.2 Versión Web.....	75
4.2.3 Mapa de procesos de predicción.....	75
CAPÍTULO 5 EXPERIMENTACIÓN Y RESULTADOS.....	77
5.1 Experimentación.....	77
5.1.1 Aumento de datos.....	77
5.1.2 Hiperparámetros.....	77
5.1.3 Creación de particiones.....	78
5.1.4. Conjunto de imágenes para la prueba final	80
5.2 Configuración de los Experimentos.....	81
5.2.1 Transferencia de aprendizaje.....	81
5.2.2 Ajuste fino	82
5.2.3 Validación del caso A.....	84
5.2.4 Análisis de resultados caso A	85
5.2.5 Caso A: Prueba final.....	86
5.2.6. Validación del caso B.....	87
5.2.7 Análisis de resultados caso B	88
5.2.8. Caso B: Prueba final.....	89
5.3 Comparación de resultados.....	90
5.4 Aplicaciones	91
5.4.1 Versión móvil.....	91
5.4.2 Versión web	92
5.4.3 Recomendaciones de uso.....	93
CAPÍTULO 6 CONCLUSIONES.....	94
Referencias	100

ÍNDICE DE FIGURAS

Figura 1.1. Mancha bacteriana (MSU, s.f.)	4
Figura 1.2. Hoja con moho (MSU, s.f.).....	5
Figura 1.3. Enfermedad del virus de la hoja amarilla (MSU, s.f.)	6
Figura 1.4. Enfermedad del hongo común (CALs, s.f.)	6
Figura 1.5. Tizón del maíz (CALs, s.f.).....	7
Figura 1.6 Esquema de solución propuesta.....	8
Figura 2.1 Resultados de la detección de enfermedades y plagas que afectan a las plantas de tomate con Faster R-CNN y VGG-16, g) Bajas temperaturas, h) Exceso o deficiencia de nutrientes, i) Moho (Álvaro et al., 2017).	19
Figura 2.2. CNN AlexNet (H. Durmus et al., 2017).	19
Figura 2.3. Muestra del dataset PlantVillage (H. Durmus et al., 2017).....	20
Figura 2.4. Moho polvoriento (Priyanka G. Shinde et al., 2017).....	21
Figura 2.5. Resultados de la segmentación (Vijai Singh & A.K. Misra, 2017).....	22
Figura 2.6. CNN Squeezenet (A. Hidayatuloh et al., 2018).....	23
Figura 2.7. a) Saludable, b) Tizón tardío; c) Hoja con moho; d) Ataque de araña roja; e) Mancha blanca; f) Virus del mosaico del tomate; g) Virus de la hoja rizada amarilla (Aravind Krishnaswamy Rangarajan et al., 2018).	24
Figura 2.8. Ejemplos de enfermedades (Belal A. M. et al., 2018).	25
Figura 2.9. CNN lineal (M. Sardogan et al., 2018).	26
Figura 2.10. Segmentación de una hoja enferma de pepino hecha por SLIC (Shanwen Zhang et al., 2018).	28
Figura 2.11. Muestra aleatoria de las imágenes (Ocampo y Dadios, 2018).	28
Figura 2.12. Ejemplares de imágenes del dataset PlantVillage (P.K Kosamkar et al., 2018).	30
Figura 2.13. Ejemplos provenientes del dataset PlantVillage (Wang j. et al., 2018).....	31
Figura 2.14. Cultivos enfermos. (1) Mancha en hoja de pepino (2) Polvo de moho en hoja de pepino (3) Vellosoidad de moho en hoja de pepino (4) Tizón bacteriano de arroz (5) <i>Rice falsesmut</i> (6) <i>Rice blast</i> (7) Mancha de lino de arroz (8) Tizón de vaina de arroz (Wang j. et al., 2018).	31
Figura 2.15. Ejemplos de enfermedades de plantas del dataset: (A) enfermedad de costra, en la planta de la manzana, (B) podredumbre negra, enfermedad de la planta de uva (Andre da Silva Abade et al., 2019).	32
Figura 2.16. Arquitectura general de una M-CNN (Andre da Silva Abade et al., 2019).	33
Figura 2.17. CNN lineal (Alexandre Pereira Marcos et al., 2019).	33
Figura 2.18. Muestra de hojas provenientes del dataset (a) Hoja sana (B) Tizón de ampolla (c) <i>Leafhoppers</i> (d) <i>Looper caterpillars</i> , (Dikdik Krisnandi et al., 2019).....	36
Figura 2.19. Predicción del virus de Septoria de tomate con un 99.02% (Emebo Onyeka et al., 2019).	36
Figura 2.20. Predicción del virus del mosaico del tomate con un 99.01% (Emebo Onyeka et al., 2019).....	36
Figura 2.21. Red DCNN (Geetharamani G. et al., 2019).	38

Figura 2.22. Red <i>DenseNet</i> en (Laha Ale <i>et al.</i> , 2019).	40
Figura 2.23. Metodología de solución en (Lili Ayu W. <i>et al.</i> , 2019).	41
Figura 2.24. Red <i>CNN</i> basada en la arquitectura <i>U-Net</i> (Lin K <i>et al.</i> , 2019).	42
Figura 2.25. Hojas de maíz provenientes del <i>dataset PlantVillage</i> . (a) moho común, (b) hoja sana, (c) tizón tardío, (d) manchas en hojas (Prakruti Bhatt <i>et al.</i> , 2019).	43
Figura 2.26. Tres tipos de enfermedades (S.S, Kumar & B.K. Raghavendra, 2019)	45
Figura 2.27. Arquitectura de la red <i>CNN</i> (S.V. Militante <i>et al.</i> , 2019).	46
Figura 2.28. Diagrama de flujo de la detección de enfermedad (B. Santhikiran <i>et al.</i> , 2020).	48
Figura 2.29. Resultado final (B. Santhikiran <i>et al.</i> , 2020).	48
Figura 2.30. Red <i>CNN VGGNet</i> (Junde Chena <i>et al.</i> , 2020)	51
Figura 2.31. Síntomas de enfermedades y hoja sana. (a) Mancha bacteriana. (b) Tizón temprano. (c) Hoja sana. (d) Tizón tardío. (e) Leaf mould. (f) Mancha Septoria. (g) Ácaro de araña. (h) Target spot. (i) Virus mosaico. (j) Enrollamiento de la hoja (Parul Sharma <i>et al.</i> , 2020).	52
Figura 2.32. Metodología de solución propuesta (Sumita Mishraa <i>et al.</i> , 2020).	53
Figura 3.1. Enfermedad causada por hongo (CANNA, s.f.)	55
Figura 3.2. Estrés por temperaturas extremas (CANNA, s.f.)	57
Figura 3.3. Estrés por exceso de riego (CANNA, s.f.)	57
Figura 3.4. Muestras de hojas del maíz (Plant Village, s.f.)	58
Figura 3.5. Representación esquemática de una RNA (Nvidia Developer, s.f.)	61
Figura 3.6. Red <i>CNN</i> (H. Durmus <i>et al.</i> , 2017)	62
Figura 3.7. Bloques convolucionales de <i>MobileNetV2</i> (Mark Sandler <i>et al.</i> , 2019)	63
Figura 3.8. Matriz de confusión binaria (Sitiobigdata.com, s.f.)	65
Figura 3.9. Cálculo de la exactitud (Sitiobigdata.com, s.f.)	65
Figura 3.10. Cálculo de la sensibilidad (Sitiobigdata.com, s.f.)	66
Figura 3.11 Cálculo de la precisión (Sitiobigdata.com, s.f.)	66
Figura 3.12. Cálculo de la métrica <i>F1</i> (Sitiobigdata.com, s.f.)	66
Figura 4.1. Segmentación transparente	69
Figura 4.2. Segmentación con sombra	70
Figura 4.3. Borrado manual	70
Figura 4.4. Selección de color	70
Figura 4.5. Fondo en color negro	70
Figura 4.6. Hojas de maíz	71
Figura 4.7. Hojas de jitomate	71
Figura 4.8. Mapa de proceso de transferencia de aprendizaje	72
Figura 4.9. Mapa de proceso de afinación	73
Figura 4.10. Interfaz gráfica versión móvil	74
Figura 4.11. Interfaz gráfica versión web	75
Figura 4.12. Etapas del proceso de predicción.	76

Figura 5.1. Curvas de exactitud y pérdida obtenidas en la transferencia de aprendizaje caso A.	82
Figura 5.2. Curvas de exactitud y pérdida obtenidas en el ajuste fino caso A.	83
Figura 5.3. Matriz de confusión caso A.....	84
Figura 5.4. Métricas de evaluación I caso A.	85
Figura 5.5. Matriz de confusión caso A prueba final.....	86
Figura 5.6. Métricas de evaluación II caso A.	87
Figura 5.7. Matriz de confusión caso B.....	87
Figura 5.8. Métricas de evaluación I caso B.	88
Figura 5.9. Matriz de confusión prueba final.....	89
Figura 5.10. Métricas de evaluación II caso B.....	90
Figura 5.11. Predicción en la aplicación móvil.....	92
Figura 5.12. Predicción en la versión WEB.....	92

ÍNDICE DE TABLAS

Tabla 2.1. Paradigmas de redes neuronales de aprendizaje profundo en el reconocimiento de patologías en plantas.....	11
Tabla 2.2. Paradigmas de redes neuronales de aprendizaje profundo en el reconocimiento de patologías en plantas continuación.....	12
Tabla 2.3. Paradigmas de redes neuronales de aprendizaje profundo en el reconocimiento de patologías en plantas <i>continuación</i>	13
Tabla 2.4. Paradigmas de redes neuronales de aprendizaje profundo en el reconocimiento de patologías en plantas <i>continuación</i>	14
Tabla 2.5. Síntesis de cultivos, patologías y bancos de imágenes.....	14
Tabla 2.6. Síntesis de cultivos, patologías y bancos de imágenes continuación.....	15
Tabla 2.7. Síntesis de cultivos, patologías y bancos de imágenes <i>continuación</i>	16
Tabla 2.8. Síntesis de cultivos, patologías y bancos de imágenes <i>continuación</i>	17
Tabla 2.9. Lista de las clases de enfermedades y plagas de tomate usadas en (Álvaro et al, 2017).....	18
Tabla 2.10. Resultados obtenidos del experimento (Vijai Singh & A.K. Misra, 2017).....	22
Tabla 2.11. Clases (A. Hidayatuloh <i>et al.</i> , 2018).....	23
Tabla 2.12. Matriz de confusión resultante en (M. Sardogan <i>et al.</i> , 2018).	26
Tabla 2.13. Arquitectura de LeNet (Serawork Walleign <i>et al.</i> , 2018).....	27
Tabla 2.14. Número de ejemplares (Serawork Walleign <i>et al.</i> , 2018).	27
Tabla 2.15. Resultados de precisiones (P.K Kosamkar <i>et al.</i> , 2018).....	30
Tabla 2.16. Porcentaje de precisión de los experimentos (Wang j. <i>et al.</i> , 2018).....	32
Tabla 2.17. Clasificadores y precisiones (Balzhoyt Roladàn Ortega <i>et al.</i> , 2019).....	35
Tabla 2.18. Configuración de parámetros (I.Z. Mukti & D. Biswas, 2019).	38
Tabla 2.19. Resultados primera etapa (I.Z. Mukti & D. Biswas, 2019).	39
Tabla 2.20. Resultados segunda etapa (I.Z. Mukti & D. Biswas, 2019).	39
Tabla 2.21. Resultados (Laha Ale <i>et al.</i> , 2019)	40
Tabla 2.22. Hiperparámetros utilizados para el entrenamiento (S. Arya & R. Singh, 2019)	44
Tabla 2.23. Resultados del experimento (Solemane Coulibaly <i>et al.</i> , 2019)	44
Tabla 2.24. Resultados (Abhinav Sagar <i>et al.</i> , 2020).....	47
Tabla 2.25. Clases (F.J.P Montalbo & A.A. Hernández, 2020).....	49
Tabla 2.26. Hiperparámetros (F.J.P Montalbo & A.A. Hernández, 2020).	49
Tabla 2.27. Arquitecturas empleadas (José G. M. <i>et al.</i> , 2020).....	50
Tabla 2.28. División del conjunto de datos (Sumita Mishraa <i>et al.</i> , 2020).	53
Tabla 2.29. Precisión de clasificación (Sumita Mishraa <i>et al.</i> , 2020).....	54
Tabla 3.1. Selección de cultivos.....	59
Tabla 3.2. Bancos de imágenes de prueba	60
Tabla 3.3. Características generales de la red <i>MobileNetV2</i> (Mark Sandler <i>et al.</i> , 2019)..	63

Tabla 5.1. Configuraciones de aumento de datos.	77
Tabla 5.2. Configuraciones de hiperparámetros.	78
Tabla 5.3. Bancos de imágenes.....	78
Tabla 5.4. Referencias caso A.	79
Tabla 5.5. Tamaño de cada lote de imágenes del caso A.	79
Tabla 5.6. Referencias caso B.	79
Tabla 5.7. División de la partición de entrenamiento y validación, caso B.	80
Tabla 5.8. Banco de imágenes prueba final.....	80
Tabla 5.9. Banco de imágenes prueba final continuación.	81
Tabla 5.10. Clases consideradas en el modelo.....	84
Tabla 5.11. Resultados de la validación cruzada caso A.....	85
Tabla 5.12. Resultados de la validación cruzada caso B.....	88
Tabla 5.13. Comparación de exactitud con diferentes trabajos del estado del arte.	90
Tabla 6.1. Objetivos específicos	95
Tabla 6.2. Alcances.....	96

ACRONIMOS

Abreviatura	Significado
AI o IA	Artificial Intelligence o Inteligencia Artificial
AP	Average Precision
ANN o RNA	Artificial Neuronal Network o Red Neuronal Artificial
AV o VA	Artificial Vision o Visión Artificial
CC	Ciencias Computacionales
CUDA	Compute Unified Device Architecture
CNN	Convolutional Neuronal Network
CMYK	Cian Magenta Yellow Key
DL	Deep Learning
F-CNN	Fully Convolutional Neuronal Network
GPU	Graphics Processing Unit
HSI	Hue, Saturation, Intensity
IDE	Integrated Development Environment
JWS	Java Web Service
LVQ	Learning Vector Quantization
M-CNN	Multi-Chanel Convolutional Neuronal Network
ML	Machine Learning
MLP	Multi Layer Perceptron
NDK	Native Developer Kit
PCA	Principal Component Analysis
PHOG	Pyramid of Histograms of Orientation Gradients
PPI	Pixels Per Inch
RBFK	Radial Basis Function Kernel
RGB	Red Green Blue
RPN	Region Proposal Network
SGD	Stochastic Gradient Descent
SGDM	Stochastic Gradient Descent with Momentum
SIFT	Scale Invariant Feature Transform
SLIC	Simple Linear Iterative Clustering
SSD	Single Shot Multibox Detector
SVM	Support Vector Machine

RESUMEN

En este proyecto de tesis se llevó a cabo la implementación de un modelo predictivo capaz de clasificar anomalías, causadas por los agentes patógenos: bacterias, hongos y virus, a partir de fotografías de hojas de las plantas de maíz y jitomate, mediante el enfoque de aprendizaje profundo.

Se analizaron varias redes neuronales convolucionales y se seleccionó e implementó la red *MobileNetV2* puesto que, esta red genera un modelo ligero pero, que ha demostrado tener un buen desempeño para clasificar imágenes capaz de ser ejecutado en la mayoría de los dispositivos, especialmente en aquellos que no poseen un *hardware* de alto rendimiento. Con la gran ventaja de que, en una aplicación móvil, no se requiere establecer una conexión vía internet para llevar a cabo un diagnóstico, lo cual lo hace especial para su uso en cualquier lugar.

Los resultados que se obtuvieron en este trabajo fueron: la creación de un modelo predictivo capaz de inferir enfermedades de los cultivos antes mencionados bajo dos implementaciones: aplicación web y móvil (*Android*), con un desempeño del 99%, valor ligeramente mayor a lo reportado en el estado del arte.

CAPÍTULO 1

INTRODUCCIÓN

En este capítulo se exponen: el análisis del problema, los objetivos (tanto general y como específicos), así como los alcances, las limitaciones, la selección de cultivos, la metodología de solución propuesta y una descripción breve del resto de capítulos que integran el presente trabajo de tesis.

1.1 Introducción

La detección de enfermedades en los cultivos, en una etapa temprana, es de suma importancia para evitar la pérdida total de las cosechas y afectar negativamente la economía de los agricultores. Sin embargo, esto no siempre es posible ya que, en algunos casos, los campesinos carecen de conocimientos y habilidades para realizar un diagnóstico a tiempo y preciso, por lo tanto, se ven obligados a contratar de los servicios de un experto en el área para llevar a cabo dicha tarea en el menor tiempo posible.

Gracias a los recientes avances que ha tenido la Inteligencia Artificial en los últimos años, es posible desarrollar redes neuronales artificiales capaces de proporcionar diagnósticos con una alta precisión de certeza en segundos. Por lo tanto, es posible desarrollar aplicaciones (herramientas de apoyo) donde los usuarios finales (agricultores) puedan monitorear sus cultivos y actuar, de manera oportuna, ante cambios causados por agentes patógenos en las plantas.

1.2 Análisis del problema

La detección de enfermedades en los cultivos es un problema que los agricultores han enfrentado desde tiempos remotos; no importa si son causadas por hongos, virus o bacterias, todas impactan de manera negativa, principalmente, en la producción de alimentos y en la inversión económica que realizan los campesinos.

Un aspecto que limita la detección de las enfermedades es la enorme variedad de características que tienen, por lo cual, no es sencillo su detección temprana. Además, muchas veces los agricultores no poseen los conocimientos de un experto para identificar las patologías presentes en los cultivos y sus diferencias para aplicar el pesticida que se requiere para preservar la salud de los productos. La confusión es mayor debido a que, algunas enfermedades presentan síntomas que se pueden confundir con deficiencias en la nutrición de las plantas, plagas o estrés hídrico.

El problema de investigación fue proponer e implementar un sistema de visión artificial, que mediante aprendizaje profundo, identifique si una planta está enferma, independientemente

si se deba a virus, hongo o bacteria, mediante el análisis de imágenes con hojas de plantas reales.

1.3 Complejidad del problema

El desarrollo del sistema que se propone es complejo, porque se debe considerar dar solución a los siguientes aspectos:

- Analizar imágenes con diferente resolución, contraste, iluminación, escala y perspectiva.
- Analizar un ser vivo, implica que las imágenes pueden contener a las plantas u hojas en diversas etapas de crecimiento; y esto implica tener cambios en tamaño, formas de las hojas, colores e incluso en la textura.
- Conocer y describir de manera clara los síntomas que provocan las enfermedades en las plantas.

1.4 Objetivos

a) Objetivo general

Desarrollar e implementar un sistema de visión artificial, utilizando técnicas de aprendizaje profundo, que identifique si una planta presenta síntomas de enfermedad.

b) Objetivos específicos

- Estudiar las características visuales de las patologías presentes en las hojas de las plantas.
- Estudiar las técnicas de aprendizaje profundo e implementar la mejor para el problema propuesto con la finalidad de poder reconocer si la planta presenta una enfermedad.
- Evaluar el rendimiento del sistema desarrollado de manera cuantitativa a través de las métricas tradicionales en el reconocimiento de patrones.

1.5 Alcances

- El sistema de visión artificial debe ser robusto a cambios en la escala, perspectiva y diferente resolución presentes en las imágenes.
- El sistema debe ser capaz de identificar síntomas de enfermedad en al menos cuatro diferentes tipos de patologías.
- El sistema debe ser capaz de reconocer si una planta presenta síntomas de enfermedad debido a virus, bacterias u hongos.
- Se requiere *hardware* de alto poder para llevar a cabo la etapa de entrenamiento del modelo, puesto que los algoritmos de aprendizaje profundo demandan demasiado costo computacional.

1.6 Limitaciones

- La imagen a procesar contiene hojas de un solo tipo de planta a analizar.

- El sistema trabaja con bancos de imágenes públicos, que contengan hojas de plantas en un ambiente controlado y no controlado.
- No se realiza el reconocimiento de la patología específica; es decir, no se diagnostica el nombre, especie o familia del agente causante de la enfermedad en la planta.
- El sistema no realiza el reconocimiento de si la planta presenta algún problema de plaga causada por insectos, estrés hídrico o falta de nutrimentos.

1.7 Justificación y beneficios

Generalmente, el agricultor cuida el aspecto y salud de su producto, ya que de su calidad depende el precio de venta y su consumo, y una de las razones que pueden provocar la pérdida de su producción es debido a la presencia de enfermedades.

Con el diseño de este sistema de visión artificial se busca apoyar a los agricultores de la zona, en uno de los problemas que enfrentan el cual es la detección de alguna patología en los cultivos.

1.8 Cultivos seleccionados

Se llevó a cabo un análisis con base en: los a) cultivos preponderantes en la región de Morelos y, b) la disponibilidad de los conjuntos de imágenes. Se encontró que en el Estado de Morelos se cosecharon 2 mil 365 hectáreas de jitomate en 2018, con un rendimiento promedio de 47.63 toneladas por hectárea, de acuerdo al Servicio de Información Agroalimentaria y Pesquera (SIAP) (El Sol de Cuernavaca, 2019). Los productores destacaron la importancia de contar con apoyo de técnicos especialistas que permitan facilitar el proceso de certificación. Señalaron que el manejo orgánico de los productos bajo invernadero como es el caso del jitomate ha mostrado un crecimiento exponencial en los últimos años (Portal Morelos, 2019).

Con respecto al cultivo de maíz, los principales productores del grano son: Chiapas, Jalisco, Puebla, Oaxaca y Guerrero. En el estado de Morelos, se cultivaron 38,124.00 hectáreas con un volumen de producción de 149,243 toneladas, dedicándose a este cultivo 15, 239 pequeños productores (Delegación SADER Morelos, 2020).

Los sembradíos que se optaron para llevar a cabo este trabajo de tesis son: jitomate y maíz; las enfermedades a estudiar fueron: dos patologías que se presentan en el maíz llamadas *common_rust* y *northern blight* y tres anomalías que se manifiestan en plantíos de jitomate denominadas: *bacterial spot*, *leaf mold* y *yellow leaf curl virus*. En la siguiente sección se detalla la información respecto a cada una de las enfermedades consideradas.

1.8.1 Patologías del jitomate

a) *Bacterial spot*

Bacterial spot (mancha bacteriana): Es una patología causada por la bacteria *Xanthomonas spp.* En el cultivo de tomate, esta enfermedad puede ser difícil de controlar cuando las condiciones climáticas apoyan a su multiplicación. Las precipitaciones abundantes y las temperaturas cálidas favorecen el desarrollo de la enfermedad. El agente

patógeno puede proliferarse en semillas y por una mala manipulación cuando el cultivo ya presenta esta enfermedad (MSU, s.f.).

Los síntomas que identifican a este padecimiento en el cultivo son:

- Las hojas con muchas lesiones pueden desarrollar una apariencia amarillenta.
- A medida que las lesiones crecen pueden causar daño o una destrucción rápida y extensa.
- El follaje muerto puede permanecer en la planta.

En la Figura 1.1 se coloca una fotografía de la enfermedad mancha bacteriana.



Figura 1.1. Mancha bacteriana (MSU, s.f.)

b) Leaf mold

Leaf mold (hoja con moho): Es un padecimiento común en los cultivos de tomates causado por los hongos *Fulvia fulva*, *formerly Cladosporium fulvum* y *Passalora fulva*. Los niveles altos de humedad y temperaturas templadas contribuyen al desarrollo de la enfermedad. El agente patógeno puede sobrevivir en los restos de cultivos por lo que, las esporas pueden ser esparcidas por la lluvia, el viento, las herramientas de trabajo, ropa e insectos (MSU, s.f.).

Los síntomas que distinguen el estado enfermo del tomate son:

- Se desarrollan lesiones de color verde pálido o amarillo con márgenes irregulares en la superficie superior de los folíolos.
- Crecimientos de hongos de colores verde oliva, similares al terciopelo, que se desarrollan en la parte inferior de los folíolos directamente debajo de las lesiones amarillas.
- Las hojas infectadas normalmente tienden a colgarse y se marchitan.

En la Figura 1.2 se observa la manifestación de la anomalía hoja con moho.



Figura 1.2. Hoja con moho (MSU, s.f.)

c) *Yellow leaf curl virus*

Yellow leaf curl virus (Virus de rizado amarillo): Es un mal que se manifiesta en los tomates causado por virus. Es una de las enfermedades virales más devastadoras de los tomates y se ha reportado de pérdidas totales de cultivos cuando la población del agente es grande. El virus es transmitido por moscas blancas adultas. Las moscas blancas pueden adquirir el virus al alimentarse de una planta infectada en aproximadamente 15 minutos y pueden transmitir el virus después de aproximadamente 6 horas. La transmisión del virus a plantas no infectadas puede ocurrir en aproximadamente 15 minutos de alimentación y las moscas blancas pueden retener el virus durante varias semanas (MSU, s.f.).

Los síntomas más evidentes de esta enfermedad son:

- Provoca un severo retraso en el crecimiento, abscisión de las flores y reducciones significativas en la cosecha.
- Las hojas son de tamaño reducido y pueden exhibir rizado hacia arriba (ahuecamiento).

La Figura 1.3 ilustra una planta de jitomate enferma del virus de la hoja amarilla.



Figura 1.3. Enfermedad del virus de la hoja amarilla (MSU, s.f.)

1.8.2 Patologías del maíz

d) *Common_rust*

Common rust (hongo común): Es una enfermedad del maíz causado por el hongo *Puccinia sorghi*. El hongo pasa el invierno en las plantas y las esporas son transportadas por el viento a los estados del norte durante la temporada de crecimiento. El desarrollo de la enfermedad se ve favorecido por el clima fresco y húmedo (60 - 70 ° F) (CALs, s.f.).

Los síntomas de la enfermedad suelen aparecer después de que el cultivo alcanza la madurez. Algunos de ellos son los siguientes:

- Se forman pequeñas pústulas redondas o alargadas de color marrón en ambas superficies de las hojas y otras partes de la planta por encima del suelo.
- A medida que las pústulas maduran, se vuelven de color marrón a negro.
- Si la enfermedad es grave, las hojas pueden amarillear y morir temprano.

La Figura 1.4 muestra una hoja de maíz enferma del hongo común.



Figura 1.4. Enfermedad del hongo común (CALs, s.f.)

e) *Northern leaf blight*

Northern leaf blight (tizón de la hoja): Es una anomalía del maíz causado por el agente patógeno *Exerohilum turcicum* (hongo). Las esporas del hongo que causa esta enfermedad pueden ser transportadas por el viento a grandes distancias de los campos infectados. La propagación localmente dentro y entre campos también depende de las esporas

transportadas por el viento. El tizón de la hoja del maíz se ve favorecido por el clima húmedo frío. Se manifiesta en la temporada de crecimiento (CALs, s.f.).

Algunos de los síntomas que caracterizan a esta enfermedad son:

- Las lesiones amarillentas del tizón de la hoja del maíz son delgadas y oblongas y se estrechan en los extremos y su tamaño varía entre 1 y 6 pulgadas.
- Las lesiones son paralelas a los márgenes de las hojas comenzando en las hojas inferiores y subiendo por la planta. Pueden fusionarse y cubrir la hoja entera.
- Las esporas se producen en el envés de la hoja debajo de las lesiones dando la apariencia de una pelusa verde polvorosa.

En la Figura 1.5 se ilustra un ejemplo de este padecimiento en la hoja del maíz.



Figura 1.5. Tizón del maíz (CALs, s.f.)

1.9 Metodología de solución

En la Figura 1.6 se presenta la metodología propuesta para dar solución al problema de tesis, contempla cinco etapas: 1) creación del conjunto de imágenes, 2) preprocesamiento de las imágenes, 3) diseño e implementación del sistema, 4) experimentación e 5) implementación móvil.

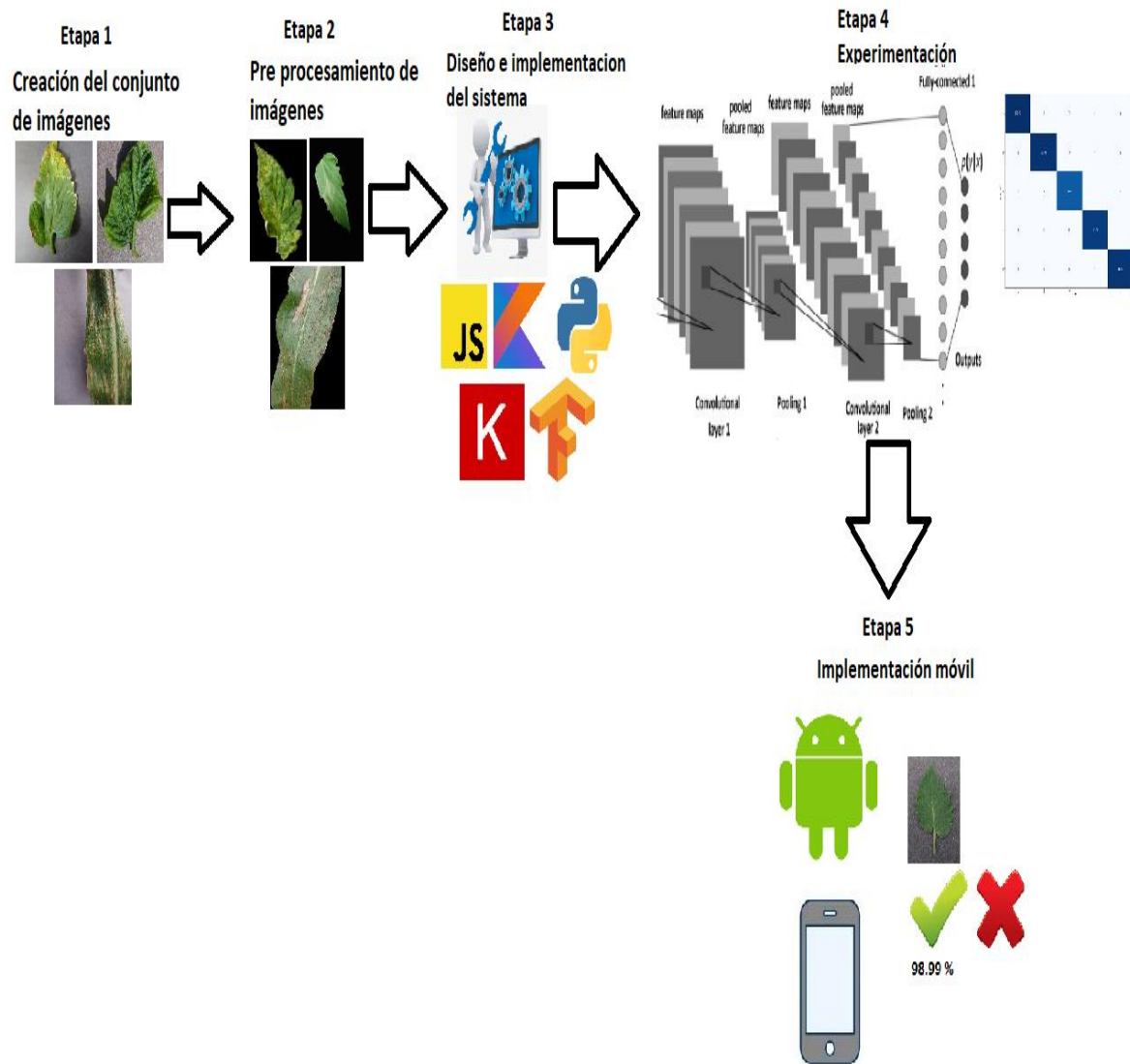


Figura 1.6 Esquema de solución propuesta

En la primera etapa se realizó la búsqueda y selección de imágenes, en este caso se decidió trabajar con los cultivos de jitomate y maíz.

En la segunda etapa se aplicó un preprocesamiento a las imágenes que consiste en localizar la zona correspondiente al fondo de la imagen y asignarle el color negro resaltando las hojas en cada muestra.

En la tercera etapa, se elaboró el diseño del sistema y de las interfaces gráficas de los sistemas web y móvil. Además, se hace la mención de las tecnologías, técnicas empleadas y anotaciones correspondientes.

Como cuarta etapa se realizó el proceso de entrenamiento y ajuste fino (*fine tuning*) de la red neuronal *MobileNetV2* en donde, se realizaron los experimentos necesarios hasta obtener un modelo predictivo con una tasa de exactitud aceptable.

En la última etapa, se adaptó el modelo predictivo creado previamente para poder funcionar tanto en una versión móvil (Android) y versión web.

1.10 Contenido

Este trabajo de tesis está conformado por seis capítulos:

- En este capítulo 1 se dio a conocer los objetivos generales y específicos, el análisis del problema, la selección de cultivos y la metodología de solución propuesta.
- El capítulo 2 presenta el análisis de artículos y trabajos antecedentes relacionados con el reconocimiento de enfermedades en plantas, causados por los agentes patógenos: bacterias, hongos y virus.
- El capítulo 3 trata la información correspondiente a las áreas involucradas en este trabajo de tesis que son: la agricultura y las ciencias computacionales.
- El capítulo 4 muestra el diseño de las interfaces gráficas de las aplicaciones web y móvil. Además, detalla las técnicas y herramientas empleadas en el proceso de desarrollo.
- El capítulo 5 presenta los casos de experimentación efectuados, sus configuraciones, desarrollos y análisis de resultados respectivos, utilizando las métricas tradicionales de aprendizaje supervisado con la finalidad de, seleccionar el mejor modelo y poder implementarlo en las aplicaciones: móvil y web.
- Finalmente, en el capítulo 6 se exponen las conclusiones generales del proyecto de tesis, la justificación de los objetivos y alcances logrados. También se anexan las aportaciones y actividades académicas adicionales hechas durante el programa de posgrado.

CAPÍTULO 2

ESTADO DEL ARTE

El capítulo aborda la lectura de los antecedentes y artículos relacionados con el tema de identificación de patologías causadas por: bacterias, hongos y virus en distintas plantas recurriendo como apoyo al área de las ciencias computacionales, específicamente a las ramas de: inteligencia artificial, visión artificial y aprendizaje profundo, para dar solución al problema planteado respectivamente.

2.1 Antecedentes

Se revisaron varios trabajos de tesis relacionados a la inspección de plantas; sin embargo, no se encontró alguno referente a la detección de enfermedades en ellas. El único trabajo relacionado, por tratarse de frutas, fue el que se presenta a continuación. También se presenta el reporte de una propuesta de tesis de un trabajo relacionado.

En la tesis “Algoritmos para la Detección y Cuantificación de Defectos en Manzanas por Inspección Visual” (E. López, 2008) se presentó un sistema de inspección visual aplicado a la detección de defectos (podredumbre, raspones y magulladuras) compuesto por cuatro módulos que localizan una manzana, detectan las áreas de tallo, cáliz, defectos, así como asignarla a una clase (alta, regular y baja) a manzanas “Golden Delicious” y “Red Delicious” de origen mexicano.

El método propuesto para detectar áreas de tallo y cáliz en manzanas funciona de manera rápida y correcta sin importar su perspectiva, siempre y cuando se aprecie su área cóncava.

La detección de defectos se realizó mediante una función de color difusa (para cada variedad de manzanas) la cual se encarga de clasificar los píxeles pertenecientes al área de la manzana en sanos y defectuosos. Esta función de color difusa se sintoniza automáticamente mediante el algoritmo propuesto por Hiroyoshi Nomura aplicado a muestras de imágenes de piel sana y defectuosa.

La propuesta de “Sistema de visión artificial para determinar enfermedades provocadas por bacterias en plantas de jitomates” (A.A. Pedroza, 2017) hace mención de un sistema de visión artificial que realice la inspección visual automatizada de hojas de plantas específicamente de jitomates *saladet* para estudiar la factibilidad de detectar enfermedades producidas por bacterias, en una etapa temprana. Dicho trabajo aún se encuentra en desarrollo.

2.2 Revisión de artículos

La Tabla 2.1, resume brevemente los diferentes tipos de redes neuronales de aprendizaje profundo que emplean los trabajos estudiados para elaborar el estado del arte; la información se estructura de la siguiente manera:

- **Problema:** Objetivo principal a resolver.
- **Imagen:** Condiciones de captura de las muestras.
- **Agente patógeno:** Nombre de los organismos causantes de las anomalías en los sembradíos.
- **Input:** Las características de la imagen de entrada al modelo.
- **Clasificador:** Nombre de la red.
- **Resultados:** Mejores resultados obtenidos en el trabajo.
- **N/D:** No definido.

Tabla 2.1. Paradigmas de redes neuronales de aprendizaje profundo en el reconocimiento de patologías en plantas.

Referencia	Problema	Imagen	Agente patógeno(s)	Input	Clasificador(es)	Resultados
Álvaro <i>et al.</i> , 2017	Detección de enfermedades y reconocimiento de plaga	<i>Diferentes tipos de ambientes</i>	Bacteria Hongo Plaga Estrés Virus	RGB	CNN Faster Region-based Convolutional NetNetwork (Faster R-CNN), Region-based Fully Convolutional Network (R-FCN)	Promedio de exactitud: 96.46%
H. Durmus <i>et al.</i> , 2017	Detección	N/D	Bacteria Hongo Virus	RGB	CNN AlexNet SqueezeNet	Precisiones: AlexNet: 0.9565 SqueezeNet: 0.943 exactitud
Vijai Singh & A.K. Misra, 2017	Detección	N/D	Bacteria Hongo Exposición solar	Segmentación	Algoritmo <i>k means</i> + SVM	Promedio de exactitud: 95,71 %
A. Hidayatuloh <i>et al.</i> , 2018	Detección y clasificación	Intemperie	Hongo Plaga	RGB	CNN Squeezenet	Exactitud: 86.92%
Anton, 2018	Detección y reconocimiento	N/D	Bacteria Hongo	RGB	CNN Mobilenet	Exactitud: 89 %

Tabla 2.2. Paradigmas de redes neuronales de aprendizaje profundo en el reconocimiento de patologías en plantas *continuación.*

Referencia	Problema	Imagen	Agente patógeno(s)	Input	Clasificador(es)	Resultados
Belal A. M. <i>et al.</i> , 2018	Detección	N/D	Bacteria Hongo Virus	RGB y escala de grises	CNN	Exactitud: A color : 99,84% y Grises: 95,54%.
M. Sardogan <i>et al.</i> , 2018	Detección y clasificación	N/D	Bacteria Hongo Virus	RGB	CNN + LVQ (<i>Learning Vector Quantization</i>)	Promedio de la exactitud: 86%
P.K Kosamkar <i>et al.</i> , 2018	Detección y recomendación de pesticida	N/D	Acaro, Bacteria Hongo Virus	RGB	CNN Tres, cuatro y cinco capas.	Exactitud: 95.05 %
Serawork <i>et al.</i> , 2018	Detección y clasificación	Condiciones no controladas	Hongo	Escala de grises	CNN LeNet	Exactitud: 99.32%
Shanwen <i>et al.</i> , 2018	Reconocimiento de enfermedad	Intemperie	Bacteria Hongo Virus	Imagen L*a*b*	Simple linear iterative clustering (SLIC) + Algoritmo K-means	Exactitud: 90.43 % y 92.15%
T. Kodoma & Y. Hata, 2018	Clasificación	Intemperie	Hongo	RGB	SVM (<i>Support Vector Machine</i>)	Exactitud: 95%
Wang j. <i>et al.</i> , 2018	Clasificación de enfermedades	Condiciones controladas	Bacteria Hongo	RGB	CNN	Exactitud: 90.84%
Alexandre <i>et al.</i> , 2019	Detección	N/D	Hongo	RGB Fondo blanco	CNN	Exactitud 95%
Andre <i>et al.</i> , 2019	Reconocimiento de enfermedad	Condiciones controladas	Hongo Virus	RGB, Grises y segmentada	M-CNN (<i>Multicanal-CNN</i>) (AlexNet y GoogleNet)	Exactitudes: AlexNet: 99.59% GloogleNet: 99.55%
Dikdik <i>et al.</i> , 2019	Clasificación de enfermedades	Condiciones controladas	Hongo Plaga	RGB	CNN GoogleNet, Xception y Inception-ResNet-v2	Mejor exactitud: 89.64 %

Tabla 2.3. Paradigmas de redes neuronales de aprendizaje profundo en el reconocimiento de patologías en plantas *continuación.*

Referencia	Problema	Imagen	Agente patógeno(s)	Input	Clasificador(es)	Resultados
I.Z. Mukti & D. Biswas, 2019	Detección y clasificación	N/D	N/D	RGB	CNN AlexNet VGG16 VGG19 ResNet50	Exactitudes: AlexNet: 93.51 % VGG16: 97.77% VGG19: 95.95% ResNet:98.42%
José G.M. Esgario, 2019	Clasificación	Parcialmente controlado	Hongo Plaga	RGB	CNN AlexNet GoogleNet VGG16 ResNet50 MobileNetV2	Exactitud: ResNet50: 95.24 %
Laha Ale <i>et al.</i> , 2019	Detección	N/D	N/D	RGB	CNN DenseNet	Exactitud: 89.7%
Lin K <i>et al.</i> , 2019	Reconocimiento	Condiciones controladas	Hongo	RGB	CNN U-Net	Exactitud: 96.08 %
Prakruti <i>et al.</i> , 2019	Identificación	Intemperie	Hongo	Segmentada	CNN VGG-16, MobileNet-v1 ResNet-50 y Inception-v2	Mejor exactitud: 98 % Inception-v2
S.V. Militante <i>et al.</i> , 2019	Detección y clasificación	N/D	Bacteria Hongo	RGB	CNN	Exactitud: Superior al 90%
Solemane <i>et al.</i> , 2019	Identificación	N/D	Hongo	RGB	CNN VGG16	Exactitud: 95%
S. Arya & R. Singh, 2019	Detección y clasificación	N/D	N/D	RGB	CNN AlexNet	Exactitudes: CNN: 90.85% AlexNet: 98.33%
Abhinav <i>et al.</i> , 2020	Detección	Condiciones controladas	Ácaro Bacteria Hongo Virus	RGB	CNN Inception v3, InceptionResNet v2, ResNet50, MobileNet, DenseNet169	Exactitud ResNet50: 98.2 %
B. Santhikiran <i>et al.</i> , 2020	Detección	N/D	Bacteria Hongo Virus	RGB	CNN	Exactitud: 90%

Tabla 2.4. Paradigmas de redes neuronales de aprendizaje profundo en el reconocimiento de patologías en plantas *continuación.*

Referencia	Problema	Imagen	Agente patógeno(s)	Input	Clasificador(es)	Resultados
F.J:P Montalbo & A.A. Hernández, 2020	Detección y clasificación	N/D	Plaga Hongo	RGB	CNN VGG16	Exactitud: 100%
Junde <i>et al.</i> , 2020	Detección y clasificación	Condiciones No controladas	Bacteria Hongo Nematodos	RGB	CNN VGGNet	Exactitud: Arroz: 92% Maíz: 80.38%
Parul <i>et al.</i> , 2020	Detección	Intemperie	Bacteria Hongo Virus	Original Segmentada	CNN F-CNN (fully- imagen) S-CNN (segmented image)	Mejor resultado: 98.6% exactitud S-CNN
Sumita <i>et al.</i> , 2020	Reconocimiento	Diferentes tipos de ambiente	Hongo	RGB	D-CNN	Exactitud: 88.46%

La Tabla 2.2, muestra de manera compacta el uso de los *datasets*, tipos de cultivos, patologías a tratar el número de y distribución de las imágenes que fueron utilizados en los artículos citados donde, la información se encuentra estructurada de la siguiente manera:

- **Dataset:** Nombre del conjunto de datos utilizado o si este fue creado desde cero junto con los dispositivos digitales (cámaras).
- **Cultivo(s):** Enlista los tipos de hortalizas con las que se trabaja.
- **Núm. Enfermedades(s):** Muestra el número de patologías a tratar.
- **Agentes patógenos:** Nombre de los organismos causantes de las anomalías en los sembradíos.
- **Núm. de muestras:** La cantidad de imágenes utilizadas en el experimento.
- **Distribución:** Describe la división de imágenes dentro del conjunto (*dataset*).
- **N/D:** No definido.

Tabla 2.5. Síntesis de cultivos, patologías y bancos de imágenes.

Referencia	Dataset	Cultivo(s)	Núm. Enfermedades(s)	Agentes patógenos	Núm. de muestras	Distribución
Álvaro <i>et al.</i> , 2017	Propio Cámara digital	Tomates	10	Bacteria Hongo Plaga Estrés Virus	5,000 RGB	80% entrenamiento 10 % validación 10% pruebas

Tabla 2.6. Síntesis de cultivos, patologías y bancos de imágenes *continuación*.

Referencia	Dataset	Cultivo(s)	Núm. Enfermedades(s)	Agentes patógenos	Núm. de muestras	Distribución
H. Durmus <i>et al.</i> , 2017	<i>PlantVillage</i>	Tomates	9	Bacteria Hongo Virus	54,309 <i>RGB</i>	80% entrenamiento. 20% validación
Vijai Singh & A.K. Misra, 2017	Propio Cámara digital	Plátano Frijol Limón	5	Bacteria Hongo Exposición solar	106 <i>RGB</i>	56.60% entrenamiento 43.39% pruebas
A. Hidayatuloh <i>et al.</i> , 2018	Propio	Tomate	6	Hongo Plaga	1,400 224x224 224x224 Escala de grises	N/D
Aravind <i>et al.</i> , 2018	<i>PlantVillage</i>	Tomates	6	Hongo Plaga Virus	13,262 227x227x3 224x224x3 Segmentadas	N/D
Belal A. M. <i>et al.</i> , 2018	<i>PlantVillage</i>	Tomates	5	Bacteria Hongo Virus	9,000 150x150x3 y escala de grises	N/D
M. Sardogan <i>et al.</i> , 2018	<i>PlantVillage</i>	Tomates	4	Bacteria Hongo Virus	500 <i>RGB</i>	80% entrenamiento 20% validación
Ocampo & Dadios, 2018	Propio/ <i>ImageNet</i>	Varios	4	Hongo Plaga	6,970 224x224x3 60,000 224x224x3	63% entrenamiento, 16% validación 21 % pruebas
P.K Kosamkar <i>et al.</i> , 2018	<i>PlantVillage</i>	Varios	38 y 16	Bacteria Hongo Plaga Virus	21,917 150x150x3	80% entrenamiento 20% validación
Serawork <i>et al.</i> , 2018	<i>PlantVillage</i>	Soya	3	Hongo	12,673 128x128 escala de grises	70% entrenamiento, 10% validación 20% pruebas.

Tabla 2.7. Síntesis de cultivos, patologías y bancos de imágenes *continuación*.

Referencia	Dataset	Cultivo(s)	Núm. Enfermedades(s)	Agentes patógenos	Núm. de muestras	Distribución
T. Kodoma & Y. Hata, 2018	Propio Cámara digital	Arroz	1	Hongo	418 <i>RGB</i>	N/D
Wang j. <i>et al.</i> , 2018	<i>PlantVillage</i> / propuesto	Pepino Arroz	8	Bacteria Hongo	10,052 <i>RGB</i> 256x256	80% entrenamiento 20% validación
Alexandre <i>et al.</i> , 2019	Propio Sony Cyber-shot DSC-W210 de 12.1 Megapixel	Café	1	Hongo	51,462	N/D
Andre <i>et al.</i> , 2019	<i>PlantVillage</i>	Fresa Manzana Uva	38	Hongo Virus	54,306 <i>RGB</i> , Grises, segmentación 256x256	80% entrenamiento 20% validación
Dikdik <i>et al.</i> , 2019	Propio Dos cámaras digitales y cinco cámaras de <i>smartphone</i>	Planta del Té	3	Hongo Plaga	4,727 <i>RGB</i> 64x64	80% entrenamiento, 10% validación 10% pruebas.
Emebo <i>et al.</i> , 2019	<i>Plant Village</i>	Tomates	3	Virus	643 96x96x3	80% entrenamiento 20% validación
I.Z. Mukti & D. Biswas, 2019	<i>GitHub</i>	Varios	38	N/D	87,867 224x224x3	80% entrenamiento 20% validación
José G.M. Esgario, 2019	Propio Cámaras de <i>Smartphone</i>	Café especie arábica	4	Bacteria Hongo Virus	1,747 224x224x3	70% entrenamiento 15 % validación 15 % pruebas.
Lin K <i>et al.</i> , 2019	Propio	Pepino	1	Hongo	50 512x512x3 No se reporta el tamaño final del <i>dataset</i> pero, utilizaron el <i>data</i> <i>augmentation</i>	80% entrenamiento 20% validación

Tabla 2.8. Síntesis de cultivos, patologías y bancos de imágenes *continuación*.

Referencia	Dataset	Cultivo(s)	Núm. Enfermedades(s)	Agentes patógenos	Núm. de muestras	Distribución
Prakruti <i>et al.</i> , 2019	<i>PlantVillage / ImageNet</i>	Maíz	3	Hongo	2,000 RGB 224x224 299x299	80% entrenamiento 20% validación
S.V. Militante <i>et al.</i> , 2019	Propio Cámara digital	Caña	6	Bacteria Hongo Virus	13,842 96x96x3 Formatos jpg y png	N/D
Solemane <i>et al.</i> , 2019	Propio Cámara digital	Mijo	1	Hongo	711 150x150x3	80% entrenamiento 20 % validación.
S. Arya & R. Singh, 2019	Propio/ <i>PlantVillage</i>	Papa y mango	2	Hongo	4,000 150x150x3	80% entrenamiento 20% validación
Abhinav <i>et al.</i> , 2020	<i>PlantVillage</i>	Varios	26	Bacteria Hongo Virus Plaga	54,306 256x256x3	N/D
B. Santhikiran <i>et al.</i> , 2020	N/D	Tomates	5	Bacteria Hongo Virus	No especifica resolución y numero de imágenes	N/D
F.:J:P Montalbo & A.A. Hernández, 2020	Propio	Café	3	Hongo Plaga	3,958 224x224x3	N/D
Junde <i>et al.</i> , 2020	Propio	Arroz y maíz	8	Bacteria Hongo	966 224x224x3 formato jpg	70% entrenamiento 30% validación
Parul <i>et al.</i> , 2020	Propio / <i>Plan Village</i> Cámaras	Tomates	10	Bacteria Hongo Virus	17,929 256x256x3	N/D
Sumita <i>et al.</i> , 2020	Propio / <i>PlantVillage</i>	Maíz	2	Hongo	4,832 150x150x3	70% entrenamiento, 10% pruebas 20% validación

A continuación se expone el análisis de los artículos que dan sustento al desarrollo de esta tesis. En cada trabajo se resalta: el objetivo, cultivos, agentes patógenos, tipo de red neuronal, *hardware*, *software* y resultados obtenidos.

A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition (Alvaro *et al.*, 2017)

Este artículo utiliza una red de aprendizaje profundo para detectar enfermedades y plagas presentes en la planta del tomate, utilizando imágenes de varias resoluciones capturadas por una cámara digital. La meta fue encontrar la arquitectura de la red adecuada, por lo que se consideraron las siguientes: *Faster Region-based Convolutional Net Network (Faster R-CNN)*, *Region-based Fully Convolutional Network (R-FCN)* y *Single Shot Multibox Detector (SSD)*. Tanto *R-CNN* como *R-FCN* utilizan el mismo método *Region Proposal Network (RPN)* para extraer características de la última capa de la *CNN*.

El *dataset* se construyó con aproximadamente 5,000 imágenes que fueron recolectadas de diferentes plantíos. Cada imagen fue recolectada bajo diferentes condiciones (enfermedades en cada estación del año, temperatura y humedad). El número de imágenes corresponden a cada una de las diferentes clases tal y como se muestra en la Tabla 2.9.

Para efectos del experimento, el *dataset* fue dividido en dos conjuntos, el 80% para el entrenamiento, 10% para la validación y 10% para las pruebas. Durante la etapa de entrenamiento se utilizó un procesador *Core i7 3.5 GHz* y dos *GPUs NVidia GeForce Titan X*.

Tabla 2.9. Lista de las clases de enfermedades y plagas de tomate usadas en (Álvaro *et al.*, 2017).

Class	Number of Images in the Dataset ¹	Number of Annotated Samples (Bounding Boxes) ²	Percentage of Bounding Box Samples (%)
Leaf mold	1350	11,922	27.47
Gray mold	335	2768	6.37
Canker	309	2648	6.10
Plague	296	2570	5.92
Miner	339	2946	6.78
Low temperature	55	477	1.09
Powdery mildew	40	338	0.77
Whitefly	49	404	0.93
Nutritional excess	50	426	0.98
Background ³	2177	18,899	43.54
Total	5000	43,398	100

¹ Number of images in the dataset; ² Number of annotated samples after data augmentation; ³ Transversal category included in every image.

Aunque el *AP (Average Precision)* para todo el sistema propuesto es mayor al 80%, en algunos casos, algunas enfermedades, como el moho de las hojas, el moho gris, el chancro y la peste, muestran un rendimiento variable. Los autores concluyen que el sistema es capaz de detectar eficientemente las clases y la localización de las enfermedades y plagas. En la Figura 2.1 se muestran ejemplos de los resultados obtenidos.

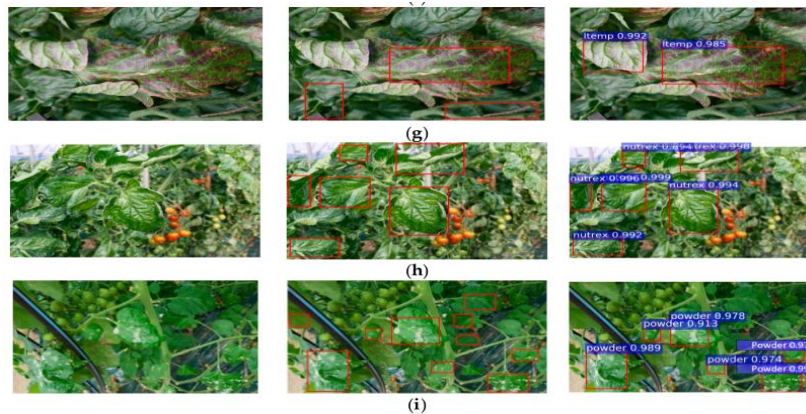


Figura 2.1 Resultados de la detección de enfermedades y plagas que afectan a las plantas de tomate con Faster R-CNN y VGG-16, g) Bajas temperaturas, h) Exceso o deficiencia de nutrientes, i) Moho (Álvaro *et al.*, 2017).

Disease Detection on the Leaves of the Tomato Plants by Using Deep Learning (H. Durmus *et al.*, 2017)

En este artículo se utiliza el enfoque de aprendizaje profundo para la detección de enfermedades a partir del análisis de las hojas de la planta de tomate. Dos tipos de redes neuronales se evalúan: la *AlexNet* y *SqueezeNet*; las cuales fueron entrenadas con imágenes provenientes del *dataset Plantvillage*. En la Figura 2.2 se muestra el modelo de la *CNN* conocida como *AlexNet*.

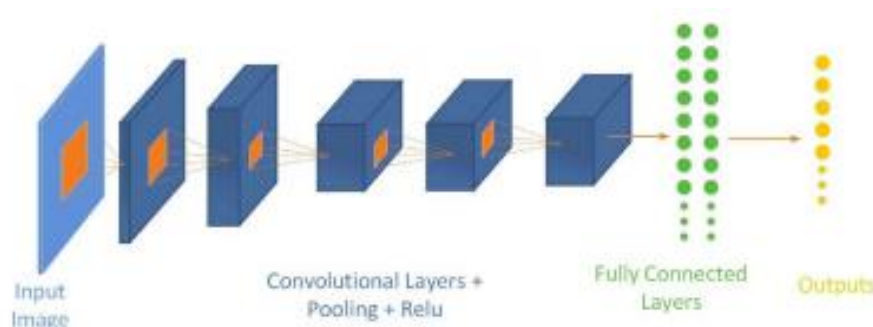


Figura 2.2. *CNN AlexNet* (H. Durmus *et al.*, 2017).

Utilizaron el *dataset Plant Village* que contiene 54,309 imágenes clasificadas de 14 cultivos diferentes. Hay diez diferentes clases de imágenes de tomates, incluyendo la categoría de imágenes sanas. Un ejemplo de las imágenes del *dataset* se muestra en la Figura 2.3.



Figura 2.3. Muestra del dataset PlantVillage (H. Durmus *et al.*, 2017).

Plant Leaf Disease Detection using Deep Learning and Convolutional Neural Network (Hanson A.J *et al.*, 2017)

El objetivo principal del estudio es explorar el enfoque del método de aprendizaje profundo para clasificar y detectar automáticamente las enfermedades de las plantas de la hoja.

Primero, ellos crearon su *dataset* (conjunto de datos) descargando las imágenes de internet, según su enfermedad y nombre de planta en varias fuentes. Ellos consideran imágenes con una resolución mayor a 500 píxeles, de esa manera se aseguró que las imágenes contenían toda la información necesaria para el aprendizaje. Como procesamiento recortaron las imágenes manualmente, alrededor de la hoja, con la finalidad de resaltar regiones interesantes (hoja).

Las técnicas de aprendizaje profundo han demostrado mejores resultados en el reconocimiento de patrones, en la segmentación y detección de objetos. Después de ajustar los parámetros de la red, lograron buenos resultados, incluso después de la iteración 30 de entrenamiento. Los resultados experimentales de precisión lograda fueron entre 91% y 98% para las pruebas, la precisión global final del modelo entrenado fue del 95%.

Plant Disease Detection Using Raspberry PI By K-means Clustering Algorithm (Priyanka G. Shinde *et al.*, 2017)

En este artículo se hace la mención de un sistema que alerta a los agricultores por medio de un *SMS (Short Message Service)* e *email* acerca de la enfermedad que el sistema haya detectado, a su vez el sistema posee un monitor donde muestra el nombre de la patología detectada.

Fue indispensable utilizar técnicas del procesamiento para el análisis de las imágenes; además, se utilizó el algoritmo de agrupamiento *k-means*.

La metodología seguida en el artículo es la siguiente:

1. Capturar la imagen en formato *RGB*.
2. Generar la estructura de transformación de color.
3. Se convierten los valores de color de *RGB* al espacio especificado en la estructura del paso 2.
4. Aplicar el algoritmo de agrupación *K-means* a la imagen
5. Enmascarar los píxeles con color verde.

6. Eliminar los píxeles enmascarados presentes dentro de los bordes del agrupamiento infectado.
7. Convertir el agrupamiento infectado de *RGB* a *HIS*.
8. Generar la matriz *SGDM* para los canales H y S.
9. Llamar la función *GLCM* para calcular las características.
10. Calcular las estadísticas de textura.
11. Configurar el clasificador (*K-means*) para el reconocimiento.

Las enfermedades siguientes fueron seleccionadas para la detección: moho polvoriento (ver Figura 2.4), moho suave, pudrición negra. Las tecnologías utilizadas fueron: *OpenCV*, *Python*, *Tomcat server*, *Raspberry PI*, cámara y fuente de poder.



Figura 2.4. Moho polvoriento (Priyanka G. Shinde et al., 2017)

Detection of plant leaf diseases using image segmentation and soft computing techniques (Vijai Singh & A.K. Misra, 2017)

Este artículo presenta un sistema de segmentación y detección de enfermedades en hojas. Algunos ejemplos que se consideraron son: la hoja del frijol con una enfermedad causada por bacteria y por hongo, hoja de limón con la enfermedad *Sun burn* y la hoja de plátano con la enfermedad temprana de quemadura.

Procesaron las imágenes para mejorar contraste. Definieron un umbral para el tono verde y removieron las zonas de no interés. Para obtener las áreas potencialmente infectadas las segmentaron con algoritmos genéticos. Posteriormente, extrajeron características de color y textura calculando primero la matriz de coocurrencia y después obteniendo homogeneidad local, contraste, energía área con mayor área. Para la clasificación utilizaron, primero el criterio de la mínima distancia con el algoritmo *kmeans* y, posteriormente, lo combinaron con una *SVM (Support Vector Machine)*. Alcanzaron una exactitud promedio del 95.71%. Los experimentos fueron realizados en *MATLAB*.

La Figura 2.5 muestra las imágenes originales las cuales fueron sometidas al proceso de segmentación y la Tabla 2.10 muestra los resultados obtenidos para cada categoría analizada.

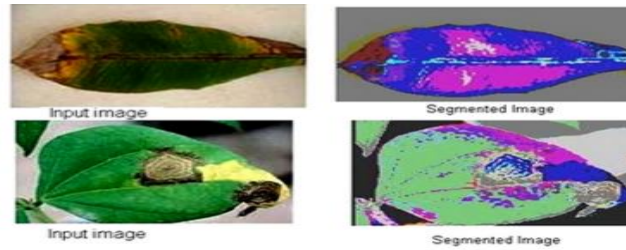


Figura 2.5. Resultados de la segmentación (Vijai Singh & A.K. Misra, 2017).

Tabla 2.10. Resultados obtenidos del experimento (Vijai Singh & A.K. Misra, 2017).

Disease samples	No. of images used for training	No. of images used for testing	Detection accuracy/%		
			MDC with K mean	MDC with proposed algorithm	SVM with proposed algorithm
Banana	15	10	80.00	90.00	90.00
Beans	15	14	92.85	92.85	92.85
Lemon	15	10	90.00	100.00	100
Rose	15	12	83.33	91.66	100
Overall accuracy			86.54	93.63	95.71

Identification of Tomato Plant Diseases by Leaf Image Using Squeezenet Model (A. Hidayatuloh *et al.*, 2018)

El trabajo tiene como objetivo realizar la detección y clasificación de enfermedades presentes en los cultivos de tomate, utilizando una red *CNN* basada en el modelo *Squeezenet*.

Con la arquitectura *Squeezenet* (ver Figura 2.6) es posible crear un modelo de tamaño relativamente pequeño el cual, es capaz de poder ser implementado en equipos con pocos recursos (poder computacional, memoria, etc.), como los teléfonos inteligentes, micro controladores y computadoras.

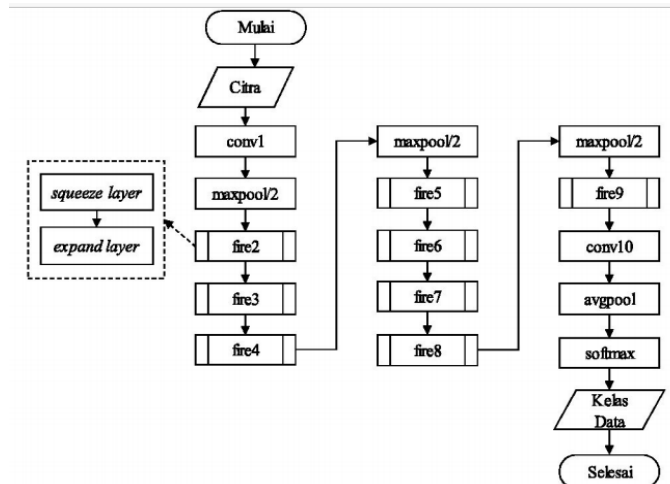


Figura 2.6. *CNN Squeezenet* (A. Hidayatuloh *et al.*, 2018).

Para la construcción del *dataset* se adquirieron 1,400 fotografías, las cuales fueron tomadas bajo condiciones de exposición de luz solar, provenientes del instituto de investigación *The Vegetable Crops Research Institute (Balitsa) Lembang, Jawa Bara*. Fue necesario aplicar a todas las capturas un redimensionamiento de 224×224 píxeles y transformarlas a niveles de gris. A partir de las muestras se definieron siete clases (seis clases referentes a patologías y una clase saludable). En la Tabla 2.11 se enlistan dichas clases.

Con la finalidad de evaluar el desempeño de la red *CNN* se utilizó validación cruzada con un valor de $K=5$. Esta técnica es utilizada para garantizar que los datos, de entrenamiento y prueba son independientes de la partición. El tamaño del modelo creado es relativamente pequeño (de 8.64 *Mbytes*) a comparación de otras redes *CNN*, obteniendo una precisión del 86.92%.

Tabla 2.11. Clases (A. Hidayatuloh *et al.*, 2018)

Variable	Class Name
C0	Early Blight
C1	Late Blight
C2	Healthy
C3	Phosphorus Deficiency
C4	Calcium Deficiency
C5	Magnesium Deficiency
C6	Tomato Leaf Miner

Tomato crop disease classification using pre-trained deep learning algorithm (Aravind Krishnaswamy Rangarajan *et al.*, 2018)

El estudio propuesto en este artículo tiene el objetivo de realizar la detección y clasificación de enfermedades en la planta del jitomate, haciendo el uso de dos modelos de aprendizaje profundo: *AlexNet* y *VGG16* pre-entrenados en *ImageNet*.

Estos modelos pre-entrenados fueron modificaciones en su capa *output* respectivamente, donde se tuvo que igualar al número de clases: seis clases de enfermedades y una clase sana. Adicionalmente las capas *softmax* y *fully connected* fueron añadidas a las arquitecturas.

La segmentación de imágenes obtenidas del *dataset PlantVillage*, consistió en poner el fondo de la imagen en color negro dejando intacta las hojas, tal y como se muestra en la Figura 2.7. Fue necesario aplicar un redimensionamiento en proporción a las características de la arquitectura, para *AlexNet* fue de 227x227 píxeles y para *VGG16* de 224x224 píxeles. El número de muestras disponibles en el banco de imágenes en total es de 13,262. En la figura 2.7 se observan ejemplares de los seis tipos de enfermedades y una muestra sana.



Figura 2.7. a) Saludable, b) Tizón tardío; c) Hoja con moho; d) Ataque de araña roja; e) Mancha blanca; f) Virus del mosaico del tomate; g) Virus de la hoja rizada amarilla (Aravind Krishnaswamy Rangarajan *et al.*, 2018).

El experimento de ambos modelos se llevó a cabo en una computadora con 4GB de *GPU* 1050 con *CUDA* disponible (640 núcleos) y 8 GB de memoria *RAM*. Los resultados finales fueron los siguientes: 97.29% para *VGG16* y 97.49% para *AlexNet*. Este último modelo demostró una buena precisión y un tiempo de ejecución menor en comparación con *VGG 16*.

Image-Based Tomato Leaves Diseases Detection Using Deep Learning (Belal A. M. *et al.*, 2018)

El objetivo principal del artículo es demostrar la flexibilidad del uso de las redes *CNN* para la clasificación de enfermedades en la planta del tomate, por medio de imágenes que contengan alguna patología que la planta presente.

Las imágenes para construir el *dataset* se obtuvieron del *dataset Plant Village*, el cual contiene aproximadamente 50,000 figuras de las cuales, sólo se tomaron 9,000 ejemplares de hojas de tomates. Estas fotografías fueron redimensionadas al tamaño de 150x150 píxeles y divididas en correspondencia a las siguientes clases: *class (0): Bacterial Spot; class (1): Early Blight; class (2): Healthy. class (3): Septorial Leaf Spot. class (4): Leaf mold. class (5): Yellow Leaf Curl Virus*, tal y como se ilustra en la Figura 2.8



Figura 2.8. Ejemplos de enfermedades (Belal A. M. *et al.*, 2018).

Los modelos propuestos consisten de dos partes: en la primera parte se encuentran cuatro capas de convolución y cuatro capas *Max Pooling*, cada capa de convolución se encuentra seguida de la capa *Max Pooling*, esto ocurre para ambas redes. En la segunda parte, después de la *flatten layer*, se encuentran dos capas densas, para ambos casos. En la arquitectura para las imágenes a colores, la primera tiene 256 unidades ocultas, lo que hace que el total número de parámetros entrenables de la red sea de 3,601,478, por el otro lado, para el caso de imágenes en niveles de gris la primera capa densa tiene 128 unidades ocultas, dando un total de 1,994,374 parámetros de entrenamiento. Se tuvo que reducir el tamaño de la red para el caso de la escala de grises para evitar problema de sobreajustes (*overfitting*) ya que la última capa para ambos tiene *Softmax* como activación y seis salidas que representan las seis clases.

El mejor resultado se obtuvo con la arquitectura para las imágenes a color el cual logró una precisión del 99.84%. El segundo modelo (escala de grises) logró una precisión de 95.54%.

Detection of unhealthy plant leaves using image processing and genetic algorithm with Arduino (M.S. Arya & D. Unni, 2018)

El enfoque principal del artículo es reconocer las enfermedades donde la velocidad y exactitud juegan un papel importante. Primero realizan la adquisición de imágenes en *RGB* y posteriormente son transformadas al espacio de color *HSI (Hue, Saturation, Intensity)*, ya que es un modelo de color basado en la percepción humana. Después identifican los píxeles de color verde, porque su mayoría representan el área sana de las hojas y no agregan ningún valor de identificación a la enfermedad. Se utilizan las redes neuronales, el clasificador *Bayes*, lógica difusa y algoritmos híbridos.

La porción infectada de la hoja se extrae y es segmentada en un número de parches de igual tamaño. El tamaño de los parches es tomado de tal manera que la información relevante no se pierda. Enseguida se extraen los segmentos que tienen más del 50% de información para el próximo paso.

Se realizó todo el experimento en *MATLAB*. Para la entrada datos, se tomó una muestra de hojas de plantas de pimienta, papa, tomate con las enfermedades del tizón tardío y mancha foliar. El método les permite identificar las enfermedades de manera temprana.

Plant Leaf Disease Detection and Classification Based on CNN with LVQ Algorithm (M. Sardogan *et al.*, 2018)

El artículo utiliza una red *CNN lineal* y el algoritmo *Learning Vector Quantization (LVQ)* para la detección y clasificación de hojas enfermas del tomate. Se modeló una *CNN* para clasificación y extracción automática de características. Se aplicaron tres filtros basados en

RGB. El *LVQ* fue alimentado por la salida que la red *CNN* obtuvo en el entrenamiento. La combinación de la red con el algoritmo *LVQ* produce un poderoso algoritmo de heurística cuya finalidad es resolver problemas de clasificación. La arquitectura del método propuesto se muestra en la Figura 2.9.

Para verificar el rendimiento del método propuesto lo evaluaron con un conjunto de experimentos. Uno de los retos principales en la detección de enfermedades y clasificación para su estudio es que las hojas con diferentes enfermedades son muy similares a otras. Por lo tanto, esa similitud puede causar que algunas hojas se coloquen en una clase equivocada.

El *dataset* contiene 500 imágenes de hojas de la planta del tomate, con cuatro síntomas de enfermedad, tal y como se muestra en la Tabla 2.12.

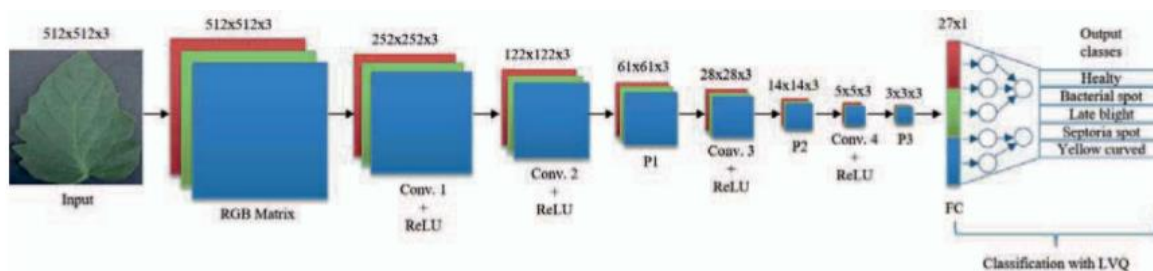


Figura 2.9. *CNN* lineal (M. Sardogan *et al.*, 2018).

Tabla 2.12. Matriz de confusión resultante en (M. Sardogan *et al.*, 2018).

Leaf Disease	Healthy	Bacterial spot	Late blight	Septoria spot	Yellow curved	Accuracy
Healthy	18	0	0	0	2	90%
Bacterial spot	0	18	0	0	2	90%
Late blight	0	0	17	0	3	85%
Septoria spot	1	0	0	16	3	80%
Yellow curved	0	0	0	3	17	85%
Average						86%

Soybean Plant Disease Identification Using Convolutional Neural Network (Serawork Wallelign *et al.*, 2018)

En este artículo se desarrolla una red *CNN* que tiene como objetivo realizar la detección y clasificación de enfermedades en los cultivos de soja.

El modelo *LeNet* fue introducido por primera vez por LeCun *et al.*, 1998. Para reconocer números digitales escritos a mano. Esta red está formada por dos capas de convolución y dos capas de submuestreo seguida por la capa *fully connected MLP*. En la Tabla 2.13 se muestra la estructura del modelo que se propuso utilizar tomando *LeNet* como referencia.

Tabla 2.13. Arquitectura de LeNet (Serawork Walleign *et al.*, 2018).

Layer	Type	Filter Size	Stride	Output size
L1	Conv	3x3	1	128x128x32
	Pool	2x2	2	64x64x32
L2	Conv	4x4	1	61x61x64
	Pool	2x2	2	64x64x32
L3	Conv	1x1	1	30x30x128
	Pool	2x2	2	15x15x128

Para la construcción del *dataset* se obtuvieron las imágenes provenientes del *dataset PlantVillage*. En la Tabla 2.14 se enlista cuatro tipos de enfermedades y el número de muestras obtenidas respectivamente, dando un total de 12,673 ejemplares.

Tabla 2.14. Número de ejemplares (Serawork Walleign *et al.*, 2018).

No.	Type of Disease	Number
1	Healthy Leaf	6234
2	Septorial leaf blight	3565
3	Frogeye leaf spot	2023
4	Downy Mildew	851
Total		12673

Ya que las fotografías adquiridas fueron en ambientes no controlados con poca luz, cada ejemplar tuvo que ser transformado a una imagen en blanco y negro y ser redimensionadas al tamaño de 128x128 píxeles con la finalidad de evitar sesgos al momento de llevar a cabo el entrenamiento. El *dataset* fue dividido en tres partes: el 70% para el entrenamiento, 10% para la validación y el 20% para las pruebas.

El mejor resultado que la lectura reporta es del 99.32%, demostrando que las técnicas de preprocesamiento de imágenes ayudaron a la red *CNN* a realizar exitosamente la extracción de características importantes de las imágenes.

Plant diseased leaf segmentation and recognition by fusion of superpixel, K-means and PHOG (Shanwen Zhang *et al.*, 2018)

El agrupamiento de píxeles es adecuado para la representación de manchas en una hoja enferma considerando color y textura. Trabajan con el algoritmo *Simple linear iterative clustering (SLIC)* para extraer súper-píxeles debido a su simplicidad y eficacia. En cada súper-píxel, el centroide puede ser fácilmente estimado lo cual puede guiar la segmentación de imagen de hojas enfermas. La principal contribución es que proponen realizar la segmentación de hojas enfermas agrupando regiones en súper píxeles mediante el algoritmo *K-means* y la descripción *mediante pyramid of histograms of orientation gradients (PHOG)*, ver la Figura 2.10.

El descriptor de imágenes *PHOG* es un método de extracción de características efectivo para describir color y textura, consiste de un histograma de orientación de gradientes sobre cada subregión de la imagen en cada resolución de nivel.

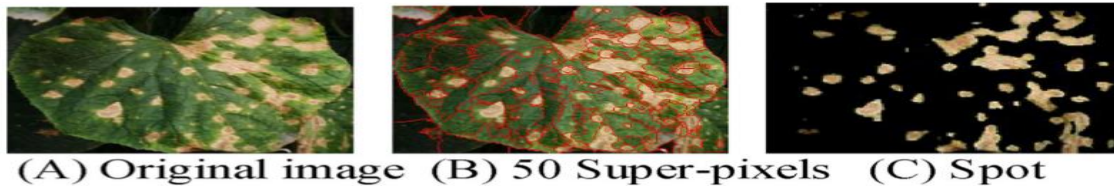


Figura 2.10. Segmentación de una hoja enferma de pepino hecha por SLIC (Shanwen Zhang et al., 2018).

El costo computacional que reportan es reducido. Validan el reconocimiento de enfermedades de plantas en dos bancos de datos de hojas enfermas, específicamente en manzana y pepino; y comparan el rendimiento de su propuesta con las técnicas *KSNNC*, *SIFT* y *IRT*. Finalmente, concluyen que con su método logra el 90.43% de exactitud con las manzanas y el 92.15% para el pepino.

Mobile Platform Implementation of Lightweight Neural Network Model for Plant Disease Detection and Recognition (Ocampo y Dadios, 2018)

En este artículo se presenta un tipo de red neuronal computacionalmente ligera que realiza la detección y reconocimiento de enfermedades en plantas y su implementación en una plataforma móvil, que contempla dos etapas de entrenamiento: el preentrenamiento del *dataset* de *ImageNet* con una gran variedad de objetos y el reentrenado con un *dataset* con enfermedades de plantas específicas.

El banco de imágenes contiene 6,970 muestras de enfermedades de plantas, estas imágenes fueron manualmente etiquetadas en seis diferentes clases: cinco para las principales patologías de las plantas y una clase categorizada como “saludable”. Las imágenes fueron aleatoriamente sometidas a rotaciones y acercamientos (*Zoom in*) para crear diferentes vistas de los mismos ejemplares. Cada imagen contiene las partes defectuosas que se redimensionan a 224x224 píxeles. El *dataset* fue dividido en tres partes de las cuales el 63% fue destinado al entrenamiento, 16% para la validación y el 21 % para el *testing*. La Figura 2.11 muestra ejemplos de las imágenes.



Figura 2.11. Muestra aleatoria de las imágenes (Ocampo y Dadios, 2018).

El modelo que se utilizó en este trabajo está preentrenado con 60,000 imágenes que incluye cosas, personas, plantas y animales. Con la finalidad de tener un control en el mecanismo inferencia del modelo, la última capa fue reentrenada con el *dataset* que se preparó en esta investigación.

Para procesar las imágenes de entrada de 224x224, el número total de parámetros del proceso suman 1.3 millones, lo cual es relativamente más pequeño en comparación con otros modelos de *CNN*.

La última capa del clasificador se entrenó 4,000 veces y recibe el ajuste de los pesos mientras que las otras capas se entrenan solo una vez. Este enfoque en el reciclaje lo hizo más rápido y factible para las máquinas menos potentes (dispositivos móviles). Después del reciclaje, se lleva a cabo la optimización para eliminar las operaciones de entrenamiento que son redundantes o tienen muy poco efecto sobre los pesos del clasificador. Los pesos del modelo preentrenado están calculados para reducir espacio de memoria en teléfono móvil.

El artículo reporta como mejor resultado un *test* con una exactitud de 89.0%. El modelo logró una buena precisión, incluso cuando se implementa en una plataforma móvil. El clasificador basado en la arquitectura *Mobilenet* reconoce eficazmente las enfermedades de las plantas. Todos los procesos inferenciales pueden hacerse de forma automática en dispositivos móviles que abren más oportunidades para otras aplicaciones.

Leaf Disease Detection and Recommendation of Pesticides using Convolution Neural Network (P.K Kosamkar *et al.*, 2018)

En este artículo se propone el uso de un sistema, el cual realiza el preprocesamiento, la extracción de características, la clasificación de la enfermedad usando una *CNN* y la recomendación del pesticida utilizando *TensorFlow* y las imágenes de hojas provenientes del *dataset PlantVillage*. Los dos principales procesos que se utilizan en el sistema es una aplicación *android* con servicios *web* de *java* (*JWS*) y aprendizaje profundo. Se hace el uso de *CNN* con cinco diferentes capas, cuatro y tres para el entrenamiento del modelo y la *APP android* como una interfaz de usuario con *JWS* para la interacción entre los sistemas.

El análisis del rendimiento de la *CNN* para la clasificación y predicción del pesticida para las hojas enfermas se llevó a cabo con el *dataset PlantVillage* (ver Figura 2.12). Se efectuaron dos experimentos en donde se utilizaron 38 clases y 16 clases. Se dividió el conjunto de datos, tomando 18,917 imágenes para entrenamiento y 3,000 imágenes para las pruebas.



Figura 2.12. Ejemplares de imágenes del dataset PlantVillage (P.K Kosamkar *et al.*, 2018).

Los resultados obtenidos (Tabla 2.15) muestran que en el entrenamiento la precisión más alta que se alcanzó se obtuvo con el modelo de cinco capas, clasificando 16 clases con un 95.57% de exactitud, entrenado con 15 épocas.

Tabla 2.15. Resultados de precisiones (P.K Kosamkar *et al.*, 2018)

Epoch	38-Classes		16-Classes	
	Accuracy	Validation Accuracy	Accuracy	Validation Accuracy
10	92.19	87.17	94.78	91.43
15	92.35	86.77	95.57	91.73
20	93.72	86.43	95.32	90.33

Development of Classification System of Rice Disease Using Artificial Intelligence (T. Kodoma & Y. Hata, 2018)

El artículo se enfoca en la creación de un sistema para clasificar plantas de arroz en enfermas o sanas. Las imágenes empleadas fueron redimensionadas al tamaño de 200x200 píxeles, con la finalidad de unificar el vector dimensional de todas las imágenes. Las características utilizadas se basan en 256 niveles de colores para cada canal (RGB).

Sin embargo, como se requiere mucho tiempo para el aprendizaje, se redujo las dimensiones del vector de características aplicando el algoritmo de *Principal Component Analysis (PCA)*. El clasificador utilizado fue un *SVM* no lineal usando *RBFK* (Radial Basis Function Kernel) y validación cruzada.

Los resultados obtenidos muestran una efectividad del 90%, la sensibilidad de detección de enfermedades un 95.6% y la especificidad un 71.4%. Para mejorar el rendimiento del sistema, incrementaron el número de imágenes de entrenamiento, esto llevó a tener una mejora del 97% en la exactitud.

CNN Transfer Learning for Automatic Image-Based Classification of Crop Disease (Wang j. *et al.*, 2018)

En este trabajo se presenta un sistema de detección de enfermedades en plantas de pepino y arroz mediante aprendizaje profundo. El clasificador propuesto se compone de cuatro partes: capas convolucionales, capas *relu*, capas *pooling* y capas *fully-connected*. La capa convolucional es el núcleo de la *CNN* la cual extrae las características abstractas a partir de las imágenes de hojas enfermas. La capa *relu* se encarga de no hacer un mapeo tipo lineal a la salida de datos. La capa *pooling* reduce la dimensión de los mapas de características. La capa *fully-connected* es responsable de realizar la inferencia lógica (clasificar las imágenes de la enfermedad en clases predefinidas).

Con la finalidad de evitar problemas de sobreajuste causado por el tamaño del *dataset*, se realiza la comparación de dos maneras. Primero, el *dataset* ingresa a la red con diferentes pesos y se evalúa el aprendizaje. En el segundo caso, se aplica la transferencia del aprendizaje; es decir, se utiliza el modelo entrenado en los campos relacionados y se hicieron algunos ajustes para reutilizar el modelo.

El *dataset* propuesto es semejante al *dataset PlantVillage*, por lo tanto, se aprovecha la similitud entre ambos conjuntos de datos de enfermedades para utilizar el aprendizaje de transferencia. Se utilizaron ocho tipos de enfermedades provenientes del *dataset PlantVillage* (ver Figura 2.13) para obtener los modelos pre-entrenados. Este conjunto se integra de 10,052 ejemplos de plantas recolectadas bajo condiciones controladas.



Figura 2.13. Ejemplos provenientes del *dataset PlantVillage* (Wang j. et al., 2018).

El *dataset* propuesto contiene 2,430 imágenes de enfermedades del pepino y arroz (ver Figura 2.14), las cuales fueron asignadas a ocho clases. Todas las imágenes fueron capturadas en un ambiente natural, con diferentes tamaños, ángulos de captura, poses, fondos e iluminación. Se tuvo que redimensionar la resolución de las imágenes a 256x256 píxeles antes de ingresar a la red. El 80% de las imágenes se utilizó para el entrenamiento y el 20% fue utilizado para la validación. La tarea fue clasificar las ocho enfermedades de las plantas de pepino y arroz.

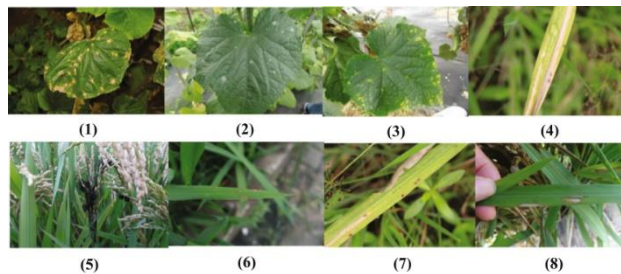


Figura 2.14. Cultivos enfermos. (1) Mancha en hoja de pepino (2) Polvo de moho en hoja de pepino (3) Velloso de moho en hoja de pepino (4) Tizón bacteriano de arroz (5) *Rice falsesmut* (6) *Rice blast* (7) Mancha de lino de arroz (8) Tizón de vaina de arroz (Wang j. et al., 2018).

En el caso de transferencia de aprendizaje, primero se entrenó con las imágenes de *PlantVillage* para obtener un modelo pre-entrenado, y luego reentrenar el *dataset* propuesto basado en este modelo para ajustar los parámetros. La *CNN* con 5 capas logra una precisión del 90.84% mediante el uso de transferencia de aprendizaje. Los resultados experimentales (ver Tabla 2.16) demostraron que la combinación de *CNN* y transferencia de aprendizaje es efectiva para clasificación de imágenes de enfermedades de cultivos con un conjunto de datos a pequeña escala.

Tabla 2.16. Porcentaje de precisión de los experimentos (Wang j. *et al.*, 2018).

	3conv	4conv	5conv	6conv	7conv	8conv
Pre-trained model (%)	0.9490	95.76	98.53	97.93	95.66	94.95
Training from scratch (%)	0.8125	86.83	87.94	87.27	81.91	77.45
Transfer learning (%)	0.7455	84.59	90.84	89.95	86.83	85.04

Plant Diseases Recognition from Digital Images using Multichannel Convolutional Neural Networks (Andre da Silva Abade *et al.*, 2019)

En este artículo se explora el potencial que tienen las redes *CNN* enfocadas a la identificación de enfermedades. Se evaluaron los modelos clásicos *AlexNet* y *GoogleNet*, entrenados desde cero, así como una red tipo *Multichannel CNN (M-CNN)*, todos ellos enfocados para entrenar y evaluar el *dataset PlantVillage* el cual contiene una gran cantidad de enfermedades de plantas.

El método propuesto utiliza el *dataset PlantVillage* el cual contiene 54,306 imágenes en condiciones controladas de hojas de plantas de 38 diferentes clases; con imágenes en color original, en escala de grises y la imagen segmentada, tal y como se muestra en la Figura 2.15. Todas las imágenes tienen la misma resolución de 256x265 píxeles.

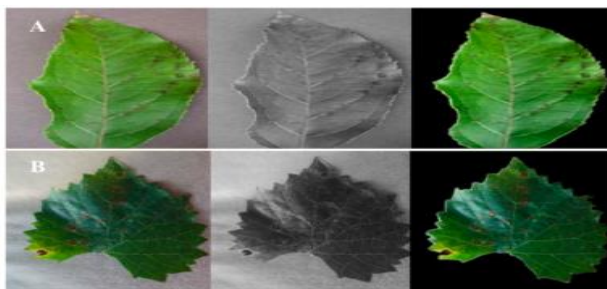


Figura 2.15. Ejemplos de enfermedades de plantas del dataset: (A) enfermedad de costra, en la planta de la manzana, (B) podredumbre negra, enfermedad de la planta de uva (Andre da Silva Abade *et al.*, 2019).

En el modelo propuesto existen dos canales individuales separados, las redes con parámetros compartidos se fusionan en la primera capa totalmente conectada, calculando las características globales mediante la comparación de ambas salidas. En la Figura 2.16 se presenta la arquitectura genérica de la red, con dos canales de entrada. Cada canal recibe un diferente tipo de *dataset*, generando tres tipos de versiones: A) Color + escala de grises, B) color + segmentación, C) escala de grises + segmentación.

Para la evaluación de los resultados obtenidos se utilizó *means F1* puntaje y precisión general. Para una mejor visualización *AlexNet* y *GoogleNet* ambas con una arquitectura multicanal se nombraron como *M-AlexNet* y *M-GoogleNet*. Durante el entrenamiento se utilizó una GPU NVIDIA GeForce GTX Titan Xp, todos los modelos fueron desarrollados usando el API *TensorFlow* version 1.6 y *Keras* version 2.2.1.

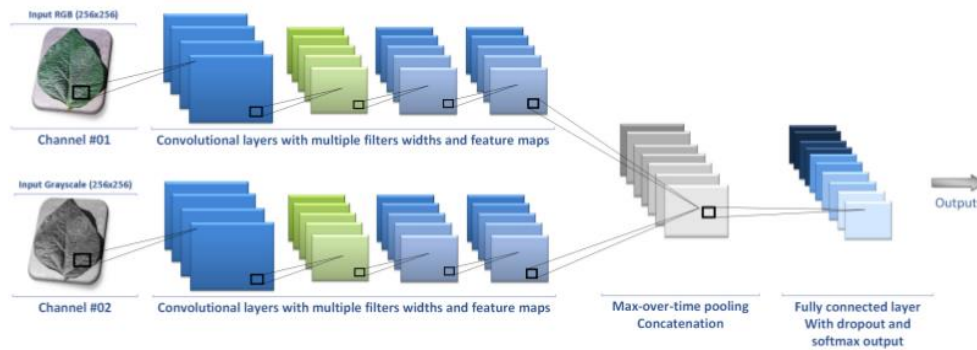


Figura 2.16. Arquitectura general de una *M-CNN* (Andre da Silva Abade *et al.*, 2019).

La arquitectura multicanal demuestra que las tres versiones del conjunto de datos (a color, escala de grises y segmentadas) pueden contribuir a mejorar la precisión, agregando información relevante a la red neuronal artificial propuesta.

Coffee Leaf Rust Detection Using Convolutional Neural Network (Alexandre Pereira Marcos *et al.*, 2019)

Este artículo tiene como objetivo crear y entrenar una red *CNN* con la finalidad de, detectar hojas enfermas presentes en los cultivos del café causados por un hongo.

La arquitectura (Figura 2.17) propuesta de la red *CNN* para este experimento está formada por dos capas de convolución, seguidas por un el filtro *non-linearity ReLU* y una capa *max-pooling*. Para complementar el rendimiento de la red, se observó que la estructura de *AlexNet* aplica una normalización de datos después de la capa *max-pooling*, por lo que se decidió aplicar esta normalización a cada salida de las capas *max-pooling*.

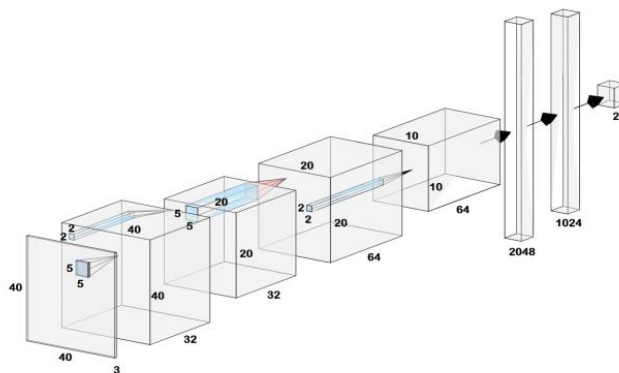


Figura 2.17. *CNN* lineal (Alexandre Pereira Marcos *et al.*, 2019).

Para la construcción del *dataset*, se obtuvieron 159 imágenes de hojas del café sobrepuestas en un fondo blanco, las cuales fueron capturadas a través de una cámara digital *Sony Cyber-shot DSC-W210* de 12.1 Megapixel. La resolución de cada ejemplar es de 2,340x4,160 píxeles. Se requirió el apoyo de un experto para realizar manualmente el etiquetado de la patología correspondiente según las características de las hojas en cada muestra.

Las técnicas de *data augmentation* que se emplearon para incrementar el tamaño del *dataset* fueron: rotaciones, escalados, recortes, redimensionamientos de 37x37 píxeles y 40x40 píxeles y ajustes en el brillo en cada una de las fotografías originales. El tamaño final de *dataset* fue de 51,462 imágenes.

Para llevar a cabo las etapas de entrenamiento y validación, se tomaron 4,086 imágenes de 40x40 píxeles las cuales fueron agrupadas en dos clases: la clase enferma si el ejemplar presenta al menos el 50% de la superficie con alguna anomalía, de lo contrario es clasificado en la clase normal. De estos mismos ejemplares se tomaron 2,859 para el entrenamiento y 1,227 para la validación.

El experimento se llevó a cabo en una PC con *Intel(R) Core(TM) i5-4440 CPU @ 3.10GHz, 8GB RAM, 64-bit Operating System and NVIDIA GeForce GTX 1050 Ti with Tensorflow 1.10.0*. El mejor resultado de precisión que se reporta fue del 95%.

Detección de enfermedades en el sector agrícola utilizando Inteligencia Artificial (Balzhoyt Roladàn Ortega *et al.*, 2019)

El artículo trata sobre la detección de enfermedades en los cultivos agrícolas, utilizando técnicas de inteligencia artificial. El sistema se integra de las etapas de adquisición de los datos, preprocesamiento, extracción de características y reconocimiento.

Una vez adquiridas las imágenes, pasan al preprocesamiento donde les tratan para el escalado, eliminación de ruido, transformación del espacio de color, ecualización de histograma y todo lo que se pueda hacer para maximizar las características.

Extraídas las características, la siguiente fase es la clasificación para la cual el artículo evalúa varios algoritmos en diferentes cultivos como se puede ver en la Tabla 2.17.

Tabla 2.17. Clasificadores y precisiones (Balzhoyt Roladàn Ortega *et al.*, 2019).

Cultivo	Clasificador	Precisión en los resultados
Café	Fuzzy logic	85.00%
Granada	SVM	82.00%
Café	KNN	58.16%
	ANN	79.04%
	N. BAYES	53.47%
	RBF y SOM	90.07%
Plátano, Frijol, Limón, Rosa	MDC + K-Means	86.54%
	MDC + propuesta	93.63%
	SVM + propuesta	95.71%
Frijol		50.00%
Mandioca		46.00%
Agrios		56.00%
Árbol de coco	Propuesta Clasificación por pares	71.00%
Café		53.00%
Maíz		40.00%
Algodón		76.00%
Uva		58.00%

Cultivo	Clasificador	Precisión en los resultados
Maracuyá		56.00%
Soja		58.00%
Caña de Azúcar		59.00%
Trigo		70.00%
Alfalfa	SVM	94.74%
Trigo, girasol, uva, maíz, pepino, algodón, col, tomate	ANN	87.48%
	SVM	92.17%
Fresa	Fuzzy logic	97.00%
Varios	CNN	99.35%
Tomate	CNN	99.84%

Evidentemente se concluye que los mejores resultados se obtienen con redes neuronales convolucionales, pues se obtienen diagnósticos más cercanos a lo que un experto humano determinaría. El único problema es que el entrenamiento es costoso computacionalmente y se requiere una gran cantidad de datos para hacerlo. Si no se cuenta con suficientes datos, la recomendación sería utilizar árboles de decisión con lógica difusa, ya que es el algoritmo con mejores resultados después de las *CNN*.

Diseases Classification for Tea Plant Using Concatenated Convolution Neural Network (Dikdik Krisnandi *et al.*, 2019)

En este artículo se propone el uso de redes *CNN* para la detección de enfermedades en plantas de té; específicamente las redes empleadas son: *GoogleNet*, *Xception* y *Inception-ResNet-v2*.

Para la construcción del *dataset* se capturaron 4,727 imágenes de hojas de té (Figura 2.18) las cuales fueron recolectadas de los campos en el *Research Institute for Tea and Cinchona, Gambug, West Java, Indonesia*. Se emplearon dos tipos de cámaras digitales y cinco cámaras de teléfonos inteligentes. Todas las imágenes se tomaron en el interior y bajo condiciones controladas. La distancia de la cámara a las hojas no fue determinada.

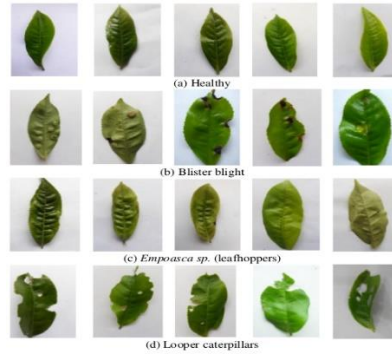


Figura 2.18. Muestra de hojas provenientes del *dataset* (a) Hoja sana (B) Tizón de ampolla (c) *Leafhoppers* (d) *Looper caterpillars*, (Dikdik Krisnandi *et al.*, 2019)

El *dataset* contiene 3,479 imágenes de plantas enfermas y 1,248 de sanas. Las imágenes recolectadas fueron redimensionadas a una resolución de 64x64 píxeles y se extrajeron los valores RGB, obteniendo las características de cada imagen. El *dataset* se dividió en tres subconjuntos: entrenamiento, validación y pruebas. Se seleccionó 80% de los datos para el entrenamiento, 10% para la validación y 10% para las pruebas.

Los resultados obtenidos en el experimento muestran que el rendimiento de *Xception* y *Inception-ResNet-v2* es mejor que *GoogLeNet*. Sin embargo, el rendimiento puede ser mejorado al agregar más datos de entrenamiento. Se indica que el desempeño tiende a disminuir cuando se usan más etiquetas de clase. El mejor resultado al usar dos clases de enfermedades logran el 98.09% de precisión. Al utilizar tres clases el porcentaje de exactitud es 90.05 %.

Development of Tomato Septoria Leaf Spot and Tomato Mosaic Diseases Detection Device Using Raspberry Pi and Deep Convolutional Neural Networks (Emebo Onyeka *et al.*, 2019)

El artículo se enfoca en el uso de *DCNN* para detectar enfermedades en la planta del tomate utilizando las librerías de ML: *keras* y *tensor flow*; sistema desarrollado en un dispositivo *Raspberry pi*.

El *dataset* que se utilizó en este proyecto fue *PlantVillage* con 643 imágenes, el 80% se destinó para el proceso de entrenamiento y el 20% para el proceso de validación. Para el proceso de entrenamiento las imágenes fueron sometidas a rotaciones, redimensionamiento, *zooming*, *flipping*; aumentando el contenido del *dataset* y evitando que la red sea capaz de ajustarse a diferentes imágenes a pesar del tamaño u orientación. En la etapa de entrenamiento, el modelo obtuvo una exactitud en promedio del 99.02% y en el proceso de validación alcanzó un 99.01% de precisión. En las Figuras 2.19 y 2.20 se ilustran los resultados de dos muestras que presentan virus.



Figura 2.19. Predicción del virus de Septoria de tomate con un 99.02% (Emebo Onyeka *et al.*, 2019).



Figura 2.20. Predicción del virus del mosaico del tomate con un 99.01% (Emebo Onyeka *et al.*, 2019)

Deep learning application for plant diseases detection (F. Jakjoud et al., 2019)

En este artículo se expone el uso de la red *DCNN* para el reconocimiento de hojas enfermas y sanas. Con la finalidad de mejorar el rendimiento de la *DCNN*, fue necesario utilizar un gran *dataset*. Se recolectaron imágenes de hojas de las plantas de diferentes fuentes como *free dataset* e imágenes capturadas con cámaras de teléfonos inteligentes. Las imágenes fueron preprocesadas mediante traslaciones, rotaciones, *zooming* en el objeto de interés y agregaron ruido aleatoriamente en cada imagen, creando un *dataset* de 13,692 imágenes. El 80% se destinó para el entrenamiento y el 20% para las pruebas. Fue necesario redimensionar las imágenes a un tamaño de 100x100 píxeles con la finalidad de reducir el tiempo de entrenamiento de la red.

Durante el experimento se desarrollaron cinco modelos basados en *SGD (Stochastic Gradient Descent)*, *RMSprop (Root Mean Square propagation)*, *Adadelta (una extensión del algoritmo AdaGrad)* y *AdaGrad* con diferentes configuraciones. Los modelos fueron entrenados con el *dataset* propuesto. El *hardware* empleado para el entrenamiento y pruebas fue una laptop con un procesador *Intel i7-7700 CPU (2.80 GHz)*, *RAM, 12GB*, *GPU GeForce GTX 1050 Ti x1*, *Windows*.

El modelo *SGD* demostró ser el más rápido, robusto, y con un valor de precisión del 90% y un tiempo de respuesta de 0.228 segundos, entre los otros modelos. También se trabajó con una placa *Raspberry pi 3 model B* con las siguientes características de hardware: *Core processor (1.2Ghz)* y *1 Gb RAM*.

Identification of plant leaf diseases using a nine-layer deep convolutional neural network (Geetharamani G. et al., 2019)

Este artículo presenta un modelo de detección de enfermedades en las hojas de las plantas basado en el uso de la red neuronal convolucional profunda (*DCNN*). La *DCNN* es un tipo de red neuronal de aprendizaje profundo complejo que agrega más representaciones jerárquicas en el modelo. La Figura 2.21 muestra la arquitectura empleada en este artículo. La *DCNN* tiene amplias aplicaciones en: clasificación de imágenes, detección de objetos, reconocimiento de voz, y el procesamiento de lenguaje natural. La *DCNN* requiere de un gran volumen de entrenamiento de datos para lograr mejores resultados.

Usaron el *dataset plantvillage (Geetharamani G. et al., 2019)* el cual contiene 54,305 imágenes de 13 diferentes tipos de hojas que fueron utilizados en la etapa de entrenamiento y pruebas para el modelo *DCNN*. La base de datos incluye 39 clases diferentes. Para aumentar el número de imágenes y reducir el sobreajuste, les realizaron transformaciones geométricas y al final el *dataset* alojó 55,636 imágenes.

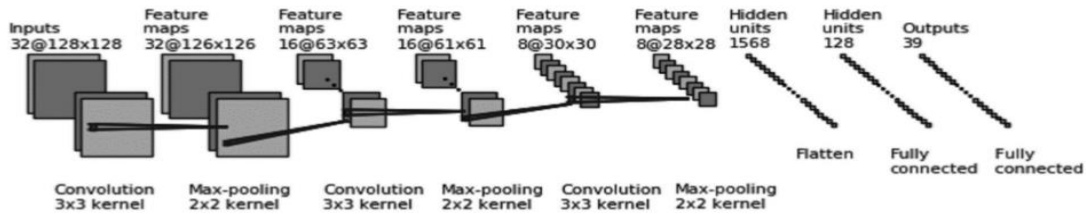


Figura 2.21. Red DCNN (Geetharamani G. et al., 2019).

Para los procesos de entrenamiento y pruebas se usaron las siguientes librerías: *the scikit-learn*, *Keras*, *pillow* y *opencv*. En cuanto al hardware se utilizó: *NVIDIA DGX-1 V100 with 8X Tesla V100 GPUs*.

El modelo propuesto DCNN es capaz de clasificar eficientemente 38 clases de plantas sanas o enfermas utilizando imágenes de hojas, el cual logra una precisión promedio de 96.46% en la clasificación, y entre 92% y 100% para la clase individual. El método de *pooling* llamado *max pooling* funciona mucho mejor que *average pooling* en comparación con otros modelos de aprendizaje automático.

Transfer Learning Based Plant Diseases Detection Using ResNet50 (I.Z. Mukti & D. Biswas, 2019)

En este artículo se busca desarrollar un modelo de CNN propio utilizando la transferencia de aprendizaje de la arquitectura *The Residual network (ResNet)*, para realizar la detección y clasificación de enfermedades en los cultivos.

Algunos de los beneficios que *ResNet* aporta es aumentar la precisión y detección según las necesidades específicas; además, *ResNet* intenta resolver las dificultades que se hacen presentes en el proceso de entrenamiento de la red CNN como: la saturación y degradación de precisión. *ResNet50* tiene un total de 50 capas. Para el desarrollo de la red propuesta se tomaron los pesos pre-entrenados en *ImageNet* de *ResNet50* y se descartó el uso de la capa *fully connected*.

Fine-tuning se utiliza para aumentar la eficiencia, esto se logra haciendo modificaciones cuidadosas en los parámetros para mejorar el resultado. Este proceso fue repetido una y otra vez para mejorar la precisión del modelo propuesto. En la Tabla 2.18 se aprecia la configuración final.

Tabla 2.18. Configuración de parámetros (I.Z. Mukti & D. Biswas, 2019).

Parameter	Value
Batch size	32
Steps per epoch	550
Epoch	25
Validation steps	1
Optimizer	SGD (stochastic gradient descent)
Learning rate	default
Decay	default
Momentum	default

El *dataset* fue elaborado a partir de la selección de 38 diferentes tipos de ejemplares de plantas, con alguna patología entre ellas y con su estado sano provenientes de los repositorios de *GitHub*, de un grupo reconocido llamado ‘*salathegroup*’. La división de imágenes del *dataset* fue de la siguiente manera: para el proceso de entrenamiento se emplearon 70,295 (80%) y 17,572 (20%) para el proceso de validación. Algunos de los cultivos que se muestran en el artículo son los siguientes: tomates, uvas, maíz, manzanas, papas, soya, fresa.

Aplicaron algunas técnicas de preprocesamiento de imágenes para aumentar el tamaño del conjunto de datos, tales como rotaciones, escalados y traslaciones, además, cada captura fue redimensionada al tamaño de 224x224 píxeles. Se realizaron varios experimentos, en una primera etapa se realizó la comparación del desempeño de *ResNet 50* contra tres modelos más, alcanzando la exactitud más alta (Ver Tabla 2.19).

Tabla 2.19. Resultados primera etapa (I.Z. Mukti & D. Biswas, 2019).

Model	Accuracy	Epoch	Time	Loss
AlexNet	93.51%	66	225s 409ms/step	0.2235
VGG19	97.95%	54	227s 413ms/step	0.0636
VGG16	97.77%	50	261s 475ms/step	0.0643
ResNet50	98.42%	4	295s 537ms/step	0.0544

En la segunda etapa se realizó la comparación de su modelo propuesto con respecto a dos modelos redes que se encuentran en el estado del arte y su modelo logró un resultado de exactitud de 99.80%, superando a la red *GooLeNet* (ver Tabla 2.20).

Tabla 2.20. Resultados segunda etapa (I.Z. Mukti & D. Biswas, 2019).

Model name	Class	Image number	Epoch	Overall Accuracy
AlexNet[12]	38	54,306	30	98.2%
GooLeNet[12]	38	54,306	30	99.34%
Proposed model	38	87,867	25	99.80%

Deep Learning based Plant Disease Detection for Smart Agriculture (Laha Ale et al., 2019)

En este artículo se utiliza una red convolucional densamente conectada (*Densely Connected Convolutional Networks, DenseNet*) basada en la transferencia de aprendizaje para la detección de enfermedades en plantas.

En este modelo de transferencia de aprendizaje, se agregó un aplanado (*flatten*) y dos capas completamente conectadas con cuatro bloques densos, con un tamaño de entrada 256x256x3 (Figura 2.22). Como resultado se tiene un total de 7,148,166 parámetros en el modelo de *DenseNet121*. El total los parámetros del modelo propuesto son menores que los necesarios para otros modelos clásicos de aprendizaje profundo, como *ResNet152* (60,

419,944 parámetros), *InceptionV3* (23,851,784 parámetros) e *InceptionResNetV2* (55,873,736 parámetros). Como el modelo es relativamente más pequeño que otros modelos se puede implementar en móviles o en dispositivos potentes del *Internet of Things* (IoT). Los pesos de *DenseNet121* se inicializan con *ImageNet*.

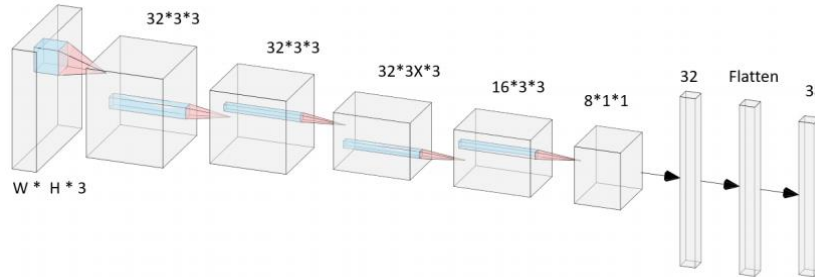


Figura 2.22. Red *DenseNet* en (Laha Ale *et al.*, 2019).

El modelo puede lograr una precisión relativamente alta y alcanzar casi la misma precisión que otras redes más complejas, el modelo converge considerablemente rápido alrededor de 50 épocas. El trabajo compara el rendimiento para el aprendizaje del modelo con diferentes resoluciones de imagen, como se muestra en la Tabla 2.21. Las bajas resoluciones comprometen el rendimiento al no tener suficientes características. Por el contrario, la alta resolución transmite más información sobre las enfermedades, pero aumenta el costo computacional exponencialmente.

Tabla 2.21. Resultados (Laha Ale *et al.*, 2019)

Input Sizes	Train Accuracy	Val Accuracy	Train Time
32x32	71.98%	71.49%	2:08:12
64x64	81.12%	74.47%	2:15:54
128x128	86.01%	85.26%	2:45:35
256x256	87.96%	87.92%	5:26:15
512x512	90.00%	89.70%	18:06:16

El proceso de entrenamiento se llevó a cabo en una PC con *Windows 10*, *hardware* que incluye *GeForce GTX 1070 Ti*, *RAM 32GB* e *Intel® CPU Core (TM) i7-8700*; y *software* que incluye *Python 3.7.3*, *CUDA 10.0* y *Tensorflow 1.13.1* y *Keras*.

Plant Nutrient Deficiency Detection Using Deep Convolutional Neural Network (Lili Ayu W. *et al.*, 2019)

En este artículo se propone el uso de una red *CNN* para diagnosticar la deficiencia de nutrientes en plantas a partir del análisis de imágenes, utilizando la arquitectura de *Inception-Resnet*.

El desafío en la detección de deficiencias de nutrientes es que en la etapa inicial, la transformación del color de las hojas no se puede distinguir visualmente, por lo que la detección temprana y el tratamiento es bastante difícil de ser diagnosticado.

Se utilizó la planta okra (*Abelmoschus Esculentus*) como información de entrada. Okra es una planta medicinal la cual es útil para pacientes con diabetes y alto colesterol. Las imágenes de la planta fueron capturadas utilizando una cámara de un teléfono inteligente con una resolución de 12MP; posteriormente, fueron redimensionadas a 299×299 píxeles. Para aumentar el número de imágenes en el *dataset* se utilizó la clase *ImageDataGenerator* proveniente de la librería *Keras*. El preprocesamiento de imágenes consistió de rotaciones, cambio de tamaño, *zooming* y voltear horizontalmente las imágenes.

Se utilizó *Inception Resnet* para detectar la deficiencia de nutrientes a partir de imágenes de la planta, cuya arquitectura se muestra en la Figura 2.23. También se implementó la red *Inception ResNet-v2* utilizando transferencia de aprendizaje del conjunto de datos *ImageNet*. Después de realizar tres experimentos, el mejor resultado de precisión fue de un 96%.

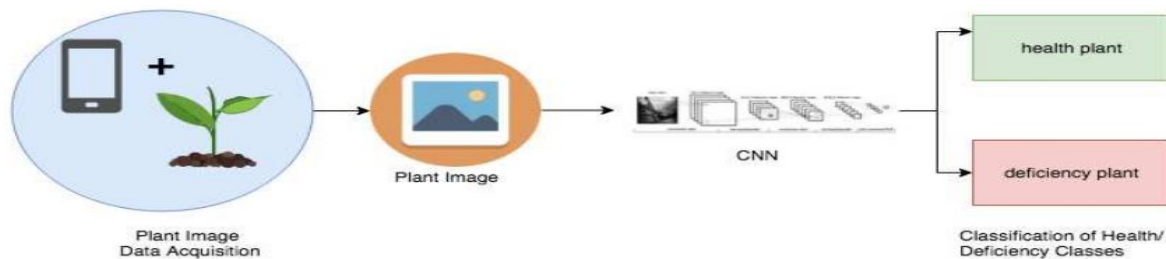


Figura 2.23. Metodología de solución en (Lili Ayu W. *et al.*, 2019).

Deep Learning-Based Segmentation and Quantification of Cucumber Powdery Mildew Using Convolutional Neural Network (Lin K *et al.*, 2019)

En este trabajo se desarrolla una red *CNN* con la finalidad de reconocer hojas enfermas causadas por hongos en los cultivos de pepinos.

El modelo propuesto en este artículo se basa en la arquitectura *U-Net*. *U-Net* es una red neuronal convolucional que ha demostrado un excelente desempeño en la segmentación de imágenes biomédicas. Entre sus características se encuentra su capa *Up-sampling* y la concatenación que hay de esta capa con la capa de activación. El proceso de la capa *Up-sampling* hace que la salida de la red neuronal tenga el mismo tamaño que la imagen de entrada, logrando una segmentación a nivel de píxeles. La arquitectura que se construyó para este trabajo se ilustra en la Figura 2.24.

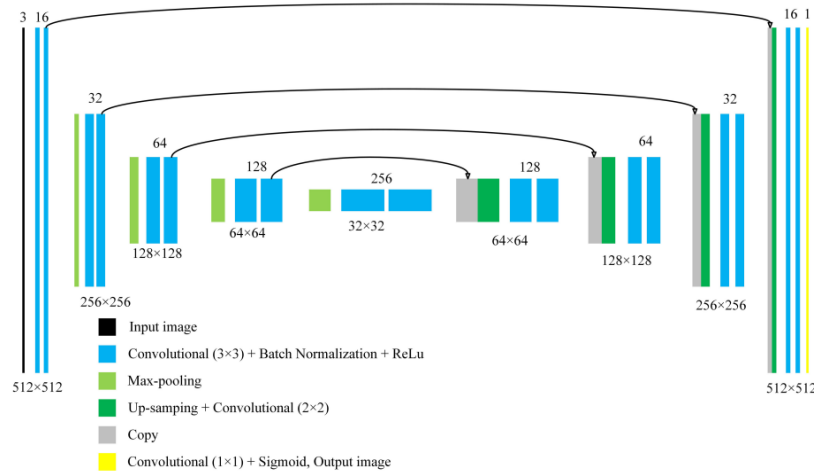


Figura 2.24. Red CNN basada en la arquitectura *U-Net* (Lin K *et al.*, 2019).

Se tomaron 50 fotografías de hojas de la planta del pepino infectadas con *powdery mildew* en una plataforma de análisis automatizada, con la finalidad de obtener una iluminación uniforme y evitar sombras en las hojas de la planta. La resolución de las capturas obtenidas fue de 2,592x1,944x3. Fue necesario resaltar las áreas enfermas de la planta (de manera manual) y redimensionar al tamaño a 512x512x3 en cada una de las capturas. Posteriormente, se seleccionaron de manera aleatoria 30 ejemplares para el entrenamiento y el resto para llevar a cabo las pruebas y evaluar el rendimiento del algoritmo.

Ya que para la etapa de entrenamiento sólo contaban con 30 imágenes, se recurrió a las técnicas de *data augmentation*: rotaciones, movimiento vertical y horizontal, *zooming in*, *zooming out*, *horizontal* y *vertical flipping*, esto con la finalidad de aumentar el conjunto de datos de entrenamiento y mejorar el aprendizaje de la red.

El *hardware* empleado en el entrenamiento fue el siguiente: una computadora con un procesador *Intel Xeon E5-2620* y una *GPU NVIDIA TESLA P100*. El *software* utilizado fue: *Keras*, *Tensorflow* y *OS Ubuntu 16.04*. El mejor resultado que reportan fue de 96.08% de precisión.

Identification of Diseases in Leaves using Convolutional Neural Networks and Boosting (Prakruti Bhatt *et al.*, 2019)

En este artículo se combinan las redes *CNN* y la técnica de *boosting* con el objetivo de detectar enfermedades y plagas en los cultivos, analizando imágenes de hojas de maíz en diferentes estados de salud provenientes del *dataset PlantVillage* que tiene imágenes de hojas sanas, moho común, tizón tardío y manchas en hojas (ver Figura 2.25). Se tomaron 500 imágenes aleatoriamente de cada clase. Las imágenes fueron sometidas a rotaciones, agregaron ruido de sal y pimienta para evitar el sobreajuste del modelo y obtener mejor precisión, dando como resultado 2,000 imágenes de cada clase. Este aumento de imágenes ayudó a crecer la calidad de los datos así como la cantidad de imágenes destinadas para el entrenamiento de clasificación. Las imágenes fueron redimensionadas en proporción a la entrada del tamaño que requiere la red neuronal, por ejemplo para *VGG-16*, *MobileNet-v1* y *ResNet-50* el tamaño de las imágenes fue de 224x224x3 y para *Inception-v2* 299x299x3.

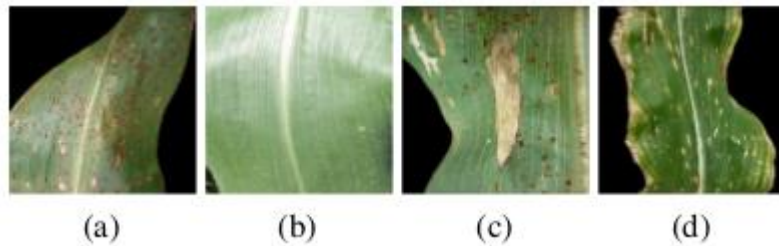


Figura 2.25. Hojas de maíz provenientes del *dataset PlantVillage*. (a) moho común, (b) hoja sana, (c) tizón tardío, (d) manchas en hojas (Prakruti Bhatt *et al.*, 2019).

Se realiza la extracción de características al pasar las imágenes a través de una red *CNN* entrenada. En caso de que el nivel de confianza del resultado de la clasificación no sea lo suficientemente alto, se utiliza un algoritmo *boosting* como apoyo para realizar una predicción confiable. El resultado final de la clasificación en las cuatro enfermedades expuestas al comienzo, con el método propuesto, es 98% de exactitud. Esto es, se mejora un 8%, aproximadamente, el rendimiento en la clasificación en comparación al uso de las redes *CNN* solamente.

A Comparative Study of CNN and AlexNet for Detection of Disease in Potato and Mango leaf (S. Arya & R. Singh, 2019)

El trabajo presentado en este artículo tiene como objetivo de utilizar una red *CNN* y un modelo pre entrenado *AlexNet* para realizar la detección y clasificación de enfermedades en los cultivos del mango y papa, y realizar la comparación del grado de precisión y eficiencia de cada arquitectura.

Las imágenes de la planta de la papa se obtuvieron del *dataset PlantVillage*, mientras que las del mango fueron capturadas de los árboles en los campos locales. Las muestras en el *dataset* fueron agrupadas en cuatro diferentes clases, de las cuales dos corresponden a enfermedades y las otras dos pertenecen a la clase saludable.

Debido a la diferente procedencia de las fotografías fue necesario ajustar sus dimensiones, 150×150 píxeles, tanto para la *CNN* como para *AlexNet*, lo cual permitió reducir el tiempo de entrenamiento en cada caso, también se realizó *data augmentation* (rotaciones, recortes, cambios en el brillo, contraste, etc.) con la finalidad de aumentar el tamaño del *dataset*.

El tamaño final del banco de imágenes fue de 4,000 ejemplares, el cual fue dividido en dos partes, el 80% fue destinado para el entrenamiento y el resto para las pruebas y validaciones. Los experimentos fueron ejecutados en una computadora con sistema operativo a 64 bits, procesador *Intel Core i3-6006U*, 4GB RAM y un disco duro de 500 GB.

En la Tabla 2.22 se muestra la configuración de los hiperparámetros correspondientes a los modelos.

Tabla 2.22. Hiperparámetros utilizados para el entrenamiento (S. Arya & R. Singh, 2019)

Hyperparameters	CNN	AlexNet
Solver type	SGD	SGD
Base Learning Rate	0.001	0.001
Momentum	0.9	0.9
Batch Size	32	32

Los resultados finales reportan que *AlexNet* alcanzó una precisión del 98.33%, mientras que la red *CNN* logró un 90.85%; sin embargo, hay que resaltar que *AlexNet* requirió mucho más tiempo en el entrenamiento, debido a que la arquitectura es más compleja comparado con *CNN*.

Deep neural networks with transfer learning in millet crop images (Solemane Coulibaly *et al.*, 2019)

Proponen el uso la red *CNN VGG 16*, con la finalidad de construir un sistema capaz de identificar la presencia de moho en los cultivos del mijo, la cual se preentrenó con *ImageNet*. La red, por defecto, toma una imagen de 224×224 píxeles como entrada y regresa un vector de 1,000 características, con la probabilidad de pertenecer a una determinada clase. Este modelo contiene 13 capas de convolución, tres capas totalmente conectadas y cinco capas *pooling*.

Para contar con información específica respecto al moho del mijo se creó un banco de datos que incluyó imágenes de internet, así como 124 imágenes propias. Aplicaron técnicas de *data augmentation* (rotación, escala, variación de color, ruido, etc.) a las imágenes de su *dataset*, dando un total de 711 imágenes. El *dataset* fue dividido en tres partes: para el entrenamiento y validación se destinó el 80%, y el resto para la fase de pruebas. Las imágenes para alimentar la red *CNN* fueron redimensionadas a un tamaño de 150×150 píxeles; además, se empleó *Stochastic Gradient Descent (SGD)* como un optimizador.

Para la etapa del experimento, se instaló el *framework Keras/Tensor Flow* en una laptop con un procesador Intel Core i5 @ 2.7 GHz y 8 GB de memoria RAM.

Se midió el rendimiento del modelo con diferentes porcentajes para el entrenamiento y validación tal y como se muestra a continuación en la Tabla 2.23.

Tabla 2.23. Resultados del experimento (Solemane Coulibaly *et al.*, 2019)

Configuration	Accuracy	FMeasure	Precision	Recall	F1-score
80 - 20	95.0%	91.67	94.5	90.5	91.75
70 - 30	93.5%	88.66	91	87.5	88.66
60 - 40	94.0%	89.67	93	88	89.67

Diseases Detection of Various Plant Leaf Using Image Processing Techniques: A review (S.S, Kumar & B.K. Raghavendra, 2019)

El artículo hace mención y descripción de algunas enfermedades más comunes en los cultivos, además expone técnicas como la segmentación de imágenes, redes neuronales del tipo *back propagation* y *deep learning*.

Una planta se enferma cuando un virus o bacteria infecta a la planta y presenta un crecimiento anormal. Las hojas de las plantas pueden variar desde su decoloración hasta la muerte. En la Figura 2.26 se ilustran algunas enfermedades.



Figura 2.26. Tres tipos de enfermedades (S.S, Kumar & B.K. Raghavendra, 2019)

Ellos analizaron las siguientes enfermedades:

- **Enfermedad de la hoja amarilla:** Esta enfermedad es causada por patógeno de tipo fitoplasma, donde la hoja verde se vuelve de color amarilla impactando gradualmente en la producción de la cosecha.
- **Enrollamiento de la hoja:** Es causada por un hongo del género *Taphrina* o por virus.
- **Mancha en la hoja:** Es una enfermedad bacteriana grave encontrada en el chile propagada por *Xanthomonas campestris pv vesicatoria*. Presenta los siguientes síntomas pequeñas hojas verdes y manchas en las hojas.
- **Marchitez bacteriana:** El rendimiento del cultivo disminuye y eventualmente la planta cae debido a la marchitez del follaje.

Mohammed Brahim *et al*, 2017 proponen las redes *CNN* como algoritmo de aprendizaje para la detección de enfermedades. Menciona que se necesita más investigación para reducir el cálculo y el tamaño de modelos profundos para máquinas pequeñas como móviles.

Yuanyuan Shao *et al*, 2017 utilizaron el método Otsu para segmentar la región de la enfermedad y realizó una extracción de multi-características. Utilizó algoritmos genéticos para reducir los tiempos de entrenamiento de la red neuronal *back propagation* y mejorar el reconocimiento. Puede identificar (a través del modelo cliente móvil y servidor) en tiempo real la enfermedad del tabaco y hacer un diagnóstico sobre enfermedades que fueron cargadas por el usuario. Sin embargo, menciona que se necesita otro método para describir las características de la enfermedad del tabaco para mejorar la precisión.

Amar Kumar Dey utilizó el método de umbral de Otsu para segmentar enfermedades a causa de pudrición en la hoja y el algoritmo *Betel Vine* para detectar la enfermedad de pudrición de la hoja considerando la característica del color de la enfermedad. Utilizaron un escáner de cañón con resoluciones de 300 *PPI (Pixels Per Inch)* para detección. Menciona que la gravedad de la enfermedad de la hoja se puede calcular en base al porcentaje del área enferma.

Sugarcane Disease Recognition using Deep Learning (S.V. Militante *et al.*, 2019)

El objetivo en este estudio es realizar la detección y clasificación de hojas enfermas y sanas provenientes de los cultivos de caña utilizando redes *CNN*.

La Figura 2.27 muestra la arquitectura interna de la red *CNN* utilizada en este trabajo, la cual recibe una imagen como entrada y en su interior se encuentran las capas de convolución, *pooling*, *fully connected*, la función de activación *ReLU* y finalmente la capa de salida.

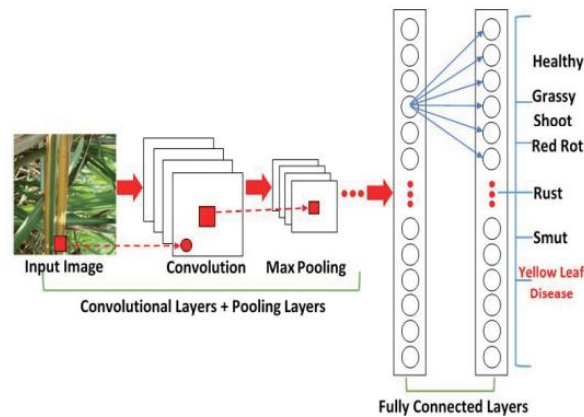


Figura 2.27. Arquitectura de la red *CNN* (S.V. Militante *et al.*, 2019).

Las imágenes para el *dataset* fueron manualmente capturadas por medio de una cámara digital, las cuales fueron redimensionadas al tamaño de 96x96 píxeles, además se les aplicaron recortes, rotaciones, *zooming* y ajustes en el brillo. Al final se obtuvieron un total de 13,842 imágenes a color en formatos *JPG* y *PNG*. Cada imagen fue clasificada conforme con las clases (seis clases de enfermedades y una clase saludable).

El modelo fue capaz de entrenar a 60 épocas con una tasa de aprendizaje de 0.001, en una computadora *Dell Inspiron 14-3476* con un procesador *Intel Core i5* y 16 GB de memoria *RAM*.

Los mejores resultados que reportaron fueron los siguientes: con una tasa de precisión del 98.50% se detectó la enfermedad *rust*, el siguiente puesto con una precisión de 99.96% se detectó la planta en estado saludable y por último, con un 98.98% se detectó la enfermedad *grassy shoot*.

On Using Transfer Learning For Plant Disease Detection (Abhinav Sagar *et al.*, 2020)

El trabajo presentado tiene como objetivo detectar enfermedades de plantas mediante la transferencia de aprendizaje utilizando cinco arquitecturas diferentes y demostrar cómo influyen los ajustes en los hiperparámetros en cada red.

La principal ventaja de la transferencia es evitar el comenzar el proceso de aprendizaje desde cero, puesto que el modelo parte de patrones que ha aprendido al resolver un problema diferente, pero de naturaleza similar al que se busca resolver. Los cinco modelos seleccionados fueron: *Inception v3*, *InceptionResNet v2* y *ResNet50*, *MobileNet* y *DenseNet169* con los pesos previamente entrenados para el trabajo propuesto.

Se utilizaron imágenes de hojas sanas y enfermas recolectadas bajo condiciones controladas provenientes del *dataset PlantVillage* de los siguientes plantíos: manzanas, arándanos, cerezas, uvas, naranjas, melocotones, pimienta, papa, frambuesas, soya, fresas y tomates. Las imágenes pueden contener alguna de las 17 enfermedades básicas, 4 enfermedades bacterianas, 2 enfermedades causadas por moho, 2 enfermedades virales y 1 enfermedad causada por un ácaro. Fue necesario aplicar técnicas de *data augmentation* como: recortar, aumentar, rotar y cambio de brillo, para pasar de 1,583 a 4,266 muestras.

Con la finalidad de tener una comparación justa entre los resultados de todas las configuraciones experimentales en las redes convolucionales, se realizó la estandarización de los hiperparámetros en todos los experimentos, de acuerdo a lo siguiente:

1. *Base learning rate: 0.001*
2. *Learning rate policy: Step*
3. *Momentum: 0.9*
4. *Weight decay: 0.0005*
5. *Gamma: 0.1*
6. *Batch size: 32*
7. *Optimizer: Adam*

La Tabla 2.24 muestra que la red *ResNet50* alcanzó el mejor resultado en las cuatro métricas utilizadas.

Tabla 2.24. Resultados (Abhinav Sagar *et al.*, 2020).

	InceptionV3	InceptionResNetV2	ResNet50	MobileNet	DenseNet169
Accuracy	0.971	0.978	0.982	0.971	0.974
Precision	0.92	0.91	0.94	0.94	0.92
Recall	0.94	0.93	0.94	0.93	0.93
F1	0.93	0.92	0.94	0.93	0.93

Design and Implementation of Unhealthy Leaves Disease Detection Using Image Processing (B. Santhikiran *et al.*, 2020)

En este artículo se diseña un sistema embebido, utilizando un vehículo integrado con una placa *Raspberry Pi 3* capaz de moverse en el campo con la finalidad de tomar videos y que estos sean enviados por *email* como archivo adjunto. El video se transforma en imágenes, las cuales se someten a un procesamiento y posteriormente pasan a ser los parámetros de entrada de la red *CNN*. En la Figura 2.28 se muestra el diagrama de flujo de la detección de enfermedades.

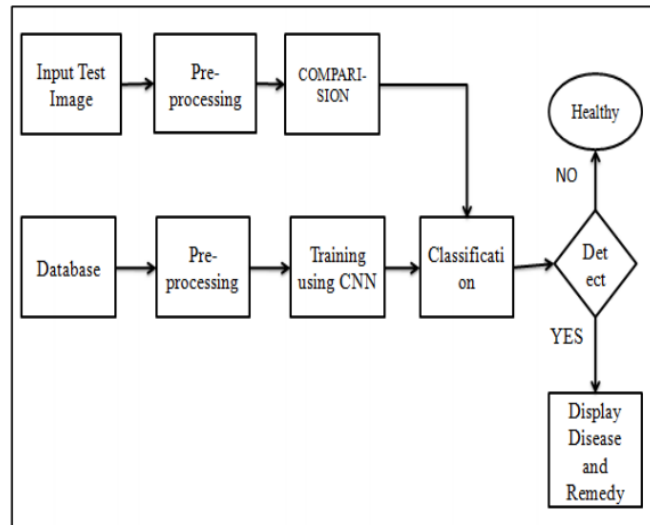


Figura 2.28. Diagrama de flujo de la detección de enfermedad (B. Santhikiran *et al.*, 2020).

Las enfermedades que se mencionan en el artículo son del tipo causadas por hongos, virus y bacterias. Como ejemplo de los resultados se muestra en la Figura 2.29, una imagen en la cual se detecta una enfermedad común (*Yellow leaf curl*) en la planta del tomate con un grado de precisión del 90%.



Figura 2.29. Resultado final (B. Santhikiran *et al.*, 2020).





An Optimized Classification Model for Coffea Liberica Disease using Deep Convolutional Neural Networks (F.J.P Montalbo & A.A. Hernández, 2020)

El trabajo expuesto en el artículo tiene como objetivo detectar y clasificar enfermedades en la planta del café utilizando el modelo *VGG16* original y ajustando los hiperparámetros con los pesos obtenidos del dataset *ImageNet*.

La arquitectura que se utilizó fue *VGG16* la cual consiste en reducir la imagen de entrada a una dimensión de 224x224 con una profundidad de 3 para *RGB*. Se usa un *kernel* de 3x3 y un método *pooling* de 2x2. Este modelo contiene una unidad lineal *ReLU* para reducir aún más los mapas de activación.

Se recolectarán 3,958 imágenes de los cultivos del café. Las clases consideradas son: *Healthy*, *Rusted*, *Infested* y *Spotted Barako leaves* (ver Tabla 2.25). Las capturas fueran clasificadas cuidadosamente por un experto en el área.

Tabla 2.25. Clases (F.J.P Montalbo & A.A. Hernández, 2020).

<i>Class</i>	<i>Image</i>	<i>Training</i>	<i>Validation</i>	<i>Testing</i>	<i>Description</i>
Healthy		804	129	10	Green, vibrant, and free from any form of damage or blemish.
Rust		1535	182	10	Presence of yellowish to orangey powdery pustules.
Infested		694	103	10	Contain dark molds with holes or chips from insect bites.
Spots		386	85	10	Contains a brownish halo-like spots.
Total	3958	3419	499	40	

Los hiperparámetros ayudan al modelo de aprendizaje profundo a producir resultados significativos incluso antes de que tenga lugar el procedimiento de entrenamiento real. Es por ello que se creó un conjunto de tres configuraciones, como se ilustra en la Tabla 2.26.

Tabla 2.26. Hiperparámetros (F.J.P Montalbo & A.A. Hernández, 2020).

Setting-1 (S1)		Setting-2 (S2)		Setting-3 (S3)	
<i>Hyperparameter</i>	<i>Value</i>	<i>Hyperparameter</i>	<i>Value</i>	<i>Hyperparameter</i>	<i>Value</i>
Epochs	100	Epochs	100	Epochs	100
FC Neurons	256	FC Neurons	512	FC Neurons	1024
Dropout Rate	0.5	Dropout Rate	0.5	Dropout Rate	0.0
Learning Rate	0.001	Learning Rate	0.0001	Learning Rate	0.001
Momentum	0.9	Momentum	0	Momentum	0
Optimizer	SGD	Optimizer	Adam	Optimizer	RMSProp
Batch Size	16	Batch Size	32	Batch Size	32

El proceso de entrenamiento fue largo, la configuración S1 tomó alrededor de 11 hrs y 58 min, S2 fue de 5 hrs y 11 min, finalmente S3 5 hrs y 22 min.

Las configuraciones de hiperparámetros diferentes, en el modelo *VGG16*, que alcanzaron el 100% de precisión son los modelos S1 y S2. El modelo S1 logró una precisión del 100% pero con una fase más lenta que la S2; además, S1 tuvo problemas de sobreajuste en épocas anteriores y posteriores. Mientras tanto, S2 también tuvo problemas de sobreajuste en las iteraciones anteriores, pero finalmente disminuyó alrededor de la época 60.

Deep learning for classification and severity estimation of coffee leaf biotic stress (José G. M. et al., 2020)

El objetivo de este trabajo es diseñar un sistema práctico y eficiente capaz de identificar y estimar el severo estrés causado por agentes bióticos en las hojas de la planta del café, basado en redes del tipo *CNN*.

Utilizaron un total de 1,747 imágenes de hojas de la especie arábica, las cuales fueron capturadas en diferentes condiciones de las estaciones del año, incluyendo hojas sanas y hojas enfermas, afectadas por uno o más de los siguientes estreses bióticos: minador de hojas, óxido, mancha de hoja marrón y mancha de hoja de cercospora. Las imágenes fueron obtenidas manualmente utilizando diferentes *smartphones* (*ASUS Zenfone 2, Xiaomi Redmi 5A, Xiaomi S2, Galaxy S8, y iPhone 6S*). Las fotos fueron tomadas del lado *abaxial* (inferior) de las hojas en condiciones parcialmente controladas y colocadas sobre un fondo blanco. El proceso de etiquetado de reconocimiento de estrés biótico fue asistido por un experto y realizado con las imágenes capturadas. A partir de las fotos obtenidas se generaron dos *dataset*, uno con las imágenes originales de las hojas enteras y una segunda que contiene sólo imágenes de hojas con síntomas.

Se seleccionaron algunas de las arquitecturas más comunes utilizadas en el problema de clasificación de enfermedades de las plantas. Algunas características de las arquitecturas utilizadas se presentan en la Tabla 2.27.

Tabla 2.27. Arquitecturas empleadas (José G. M. et al., 2020).

CNN architecture	Parameters (M)	Layers
AlexNet	61	8
GoogLeNet	6.9	22
VGG16	138	16
ResNet50	25	50
MobileNetV2	2.2	54

El tamaño de entrada de las imágenes para la red *CNN* es de $224 \times 224 \times 3$. Debido al tamaño de las imágenes, la red puede encontrar dificultades para capturar características relevantes en el conjunto de datos de hojas con síntomas muy pequeños. Para aliviar este problema, se utilizó un método de umbral fijo en el canal *S* del espacio de color *HSV*, de tal manera que la segmentación de la hoja permite colocar un cuadro delimitador para la eliminación de las regiones que no pertenecen a la región de interés.

Para la realización de todos los experimentos se utilizaron las siguientes proporciones 70% entrenamiento, 15% validación y 15% para *testing*. Se empleó *PyTorch* (*Machine learning open source library*) y una GPU *NVIDIA GeForce GTX 1060* y *CUDA 10.0*.

Los resultados del sistema propuesto, utilizando la arquitectura *ResNet50*, obtuvo una precisión del 95.24% para la clasificación de estrés biótico y 86.51% para la estimación de severidad. Además, se encontró que, al clasificar solo los síntomas, los resultados fueron superiores al 97%. Los resultados experimentales indican que el sistema propuesto podría ser una herramienta adecuada para ayudar tanto a expertos como a agricultores en la identificación y cuantificación de estrés biótico en cafetales.

Using deep transfer learning for image-based plant disease identification (Junde Chena *et al.*, 2020)

En este trabajo se hace el uso de la transferencia de aprendizaje profundo para realizar la identificación y clasificación de enfermedades en los cultivos del arroz y maíz, utilizando el modelo pre entrenado *VGGNet* con el *dataset ImageNet*.

La red *VGGNet* es un tipo de red *CNN* desarrollado por un grupo de geometría visual de la universidad de Oxford y Google *DeepMind*. En su arquitectura (ver Figura 2.30) se encuentran las siguientes capas: convolución, *pooling*, y *fully-connected*.

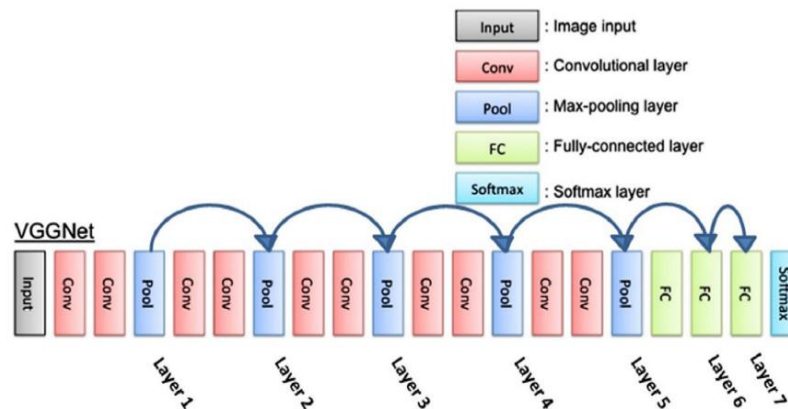


Figura 2.30. Red *CNN VGGNet* (Junde Chena *et al.*, 2020).

Se realizó la adquisición de 500 y 466 imágenes de los cultivos de arroz y maíz respectivamente, provenientes del Fujian Institute of Subtropical Botany, Xiamen, China. Las muestras fueron capturadas bajo las siguientes condiciones: iluminación no uniforme, poca intensidad lumínica y condiciones no controladas en el exterior (campos de cultivo). Las patologías obtenidas en los cultivos del arroz fueron las siguientes: *Rice Stackburn*, *Rice Leaf Scald*, *Rice Leaf Smut*, *Rice White Tip*, and *Bacterial Leaf Streak*. Con respecto a los sembradíos de maíz se encuentran: *Phaeosphaeria Spot*, *Maize Eyespot*, *Gray Leaf Spot*, and *Goss's Bacterial wilt*. Con la finalidad de obtener un estándar en las fotografías, se realizó el redimensionamiento a 224x224x3 píxeles en formato *JPG*. El *dataset* fue dividido en dos partes, 70% para el entrenamiento y el 30% restante para realizar validaciones.

Se llevó a cabo el preprocesamiento de las imágenes (escalaciones, rotaciones, ajustes en el brillo, contraste y traslaciones). Se utilizó *Anaconda3 (Python 3.6)*, *Keras-GPU library* y *OpenCV-python3 library*. El proceso de entrenamiento y pruebas se llevaron a cabo en una computadora con las siguientes características: *Intel® Core™ i7-8750 central processing unit (CPU) at 2.20 GHz with 8GB memory and NVIDIA GeForce GTX 1060 (CUDA 9.0 and 6.9 GB memory) graphics card (GeForce, 1060)*.

El mejor resultado que se logró fue al momento de realizar detecciones de enfermedades en los cultivos del arroz con una precisión del 92%; sin embargo, en el caso del maíz la precisión alcanzada fue de 80.38%. Esto se debe a que "*Phaeosphaeria Spot*" and "*Maize Eyespot*" comparten características patológicas visualmente similares en ambos cultivos, además, las condiciones no controladas de algunas imágenes comprometieron de manera no tan favorable el resultado final.

Performance analysis of deep learning CNN models for disease detection in plants using image segmentation (Parul Sharma *et al.*, 2020)

Este artículo propone una solución temprana en la detección de enfermedades en cultivos haciendo el uso de la segmentación de imágenes con redes *CNN*. Comparan el modelo *F-CNN entrenado* usando el conjunto *full images* y el modelo *S-CNN* entrenado con imágenes segmentadas. Una vez que las imágenes son leídas, porciones aleatorias de 256 x 256 píxeles de la imagen son extraídas y se les aplica ruido, distorsión, rotación.

El entrenamiento y las pruebas se llevaron a cabo en un servidor de *Google Cloud*, con una *GPU NVIDIA*. Con la finalidad de analizar el trabajo del modelo propuesto en detalle, se seleccionó la planta del tomate, el cual incluye 10 tipos de hojas enfermas y hojas sanas. Los síntomas se muestran en la Figura 2.31. *La CNN* la cual logró una precisión superior al 93% para 15 diferentes tipos de plantas.



Figura 2.31. Síntomas de enfermedades y hoja sana. (a) Mancha bacteriana. (b) Tizón temprano. (c) Hoja sana. (d) Tizón tardío. (e) Leaf mould. (f) Mancha Septoria. (g) Ácaro de araña. (h) Target spot. (i) Virus mosaico. (j) Enrollamiento de la hoja (Parul Sharma *et al.*, 2020).

El primer modelo *F-CNN* fue entrenado utilizando un *dataset* con imágenes de hoja completa, con antecedentes y enfermedades variables. El segundo fue entrenado utilizando un *dataset* generado a partir de las imágenes usadas en el modelo *F-CNN* pero segmentadas, a fin de solo incluir regiones de interés (síntomas de enfermedades) manchas / lesiones.

713 imágenes que fueron descargadas de internet, se etiquetaron manualmente utilizando la segmentación de imágenes *random stretch*. Les aplicaron desenfoque, ajustes al brillo y contraste, tomando en cuenta las áreas de interés (lesiones). También se recolectaron manualmente 637 imágenes de los plantíos locales y fueron tratadas con los mismos criterios.

Deep Convolutional Neural Network based Detection System for Real-time Corn Plant Disease Recognition (Sumita Mishraa *et al.*, 2020)

El artículo habla de un sistema de reconocimiento de enfermedades de plantas de maíz basado en aprendizaje profundo que funciona en un dispositivo móvil independiente (Raspberry pi, teléfono inteligente) con la ventaja de no necesitar acceso a Internet.

En la Figura 2.32, se muestra el *framework* propuesto donde primero se entrena la red neuronal profunda en una computadora con GPU que usa *Keras API*, posteriormente el modelo entrenado se migra de un GPU de alto rendimiento a dispositivos móviles con una capacidad computacional limitada. El dispositivo móvil está reservado para el pre-procesamiento de imágenes cuando hay una nueva planta se capturan imágenes de hojas. El modelo *DNN* entrenado se implementa en el *Intel Movidius NCS*, un sistema en chip (SoC) usando el kit de herramientas *NSDSK*.



Figura 2.32. Metodología de solución propuesta (Sumita Mishraa *et al.*, 2020).

Después de que se entrena el modelo de aprendizaje profundo, *Movidius NCS* se conecta con el software *OpenCV*, instalado en la *Raspberry Pi* para procesar imágenes en vivo alimentadas por la cámara del teléfono inteligente. *OpenCV* fue elegido ya que el módulo *DNN* de *OpenCV* está optimizado por Intel para admitir el aprendizaje profundo.

Las imágenes para el *dataset* fueron creadas a partir de algunas capturas de plantaciones de maíz, pero la mayor parte de las imágenes se obtuvieron del *dataset* de *PlantVillage*. Las imágenes capturadas están etiquetadas por un experto en enfermedades del maíz. La Tabla 2.28 muestra la división del conjunto de datos para entrenamiento, pruebas y validación con una proporción de 70%, 10% y 20% respectivamente.

Tabla 2.28. División del conjunto de datos (Sumita Mishraa *et al.*, 2020).

Class	Common Rust	Northern Leaf Blight	Healthy
No.of Train Images	1192	986	1162
No.of Test Images	139	100	124
No.of Validation Images	227	291	161

Se utilizaron imágenes de tamaño 150×150 píxeles para alimentar la red neuronal. Durante la fase de entrenamiento los hiper parámetros de la red, como la tasa de aprendizaje y el *maximum epoch* fueron cambiados para lograr precisiones superiores al 96%.

La precisión del modelo *CNN* profundo implementado usando *NCS* en las imágenes capturadas también se calcula como el porcentaje de las imágenes que el modelo detecta correctamente. El modelo móvil *CNN* se ejecuta en 30 imágenes de las hojas experimentales de campo, con 10 imágenes que pertenecen a cada clase que aparece en la Tabla 2.29.

Tabla 2.29. Precisión de clasificación (Sumita Mishraa *et al.*, 2020)

Class	Common Rust	Northern Leaf Blight	Healthy
Accuracy of NCS Model	77.26%	88.42%	100%
Accuracy of Deep Learning model trained on GPU	96.32%	98.88%	100%

2.3 Análisis de artículos

Como se observa en la síntesis de los trabajos presentados, existe el interés de desarrollar sistemas que detecten y/o clasifiquen a través del análisis de imágenes, cuando una planta presenta síntomas de alguna enfermedad ya que, los artículos revisados no se enfocan en un solo tipo de plántíos y patologías, sino que analizan y evalúan diversos casos de estudio.

El uso de redes pertenecientes al tipo de aprendizaje profundo está teniendo un buen desempeño en este tipo de sistemas. Sin embargo, algunos factores a considerar para su uso son que se requiere: a) se requiere de hardware de alto poder, b) una gran cantidad de imágenes y c) una resolución específica según sea el tipo de red neuronal convolucional a utilizar.

El banco de imágenes más conocido y utilizado en esta área de interés es *PlantVillage* el cual, aloja una colección de imágenes de varios cultivos, ya clasificados por expertos y con condiciones aceptables; es decir, sin presencia de sombras, falta y/o exceso de iluminación, ruido inyectado por el mismo dispositivo, baja resolución etc.

También se observa que se han utilizado distintos modelos de redes convolucionales. Las más utilizadas son *AlexNet*, *GoogleNet*, *InceptionV3*, *ResNet50*, *VGG16*, *VGG19* y entre los modelos más compactos se encuentran: *MobileNet*, *MobileNetV2* y *DenseNet121*.

Se observó que la configuración correcta del banco de imágenes es destinar el mayor volumen para el entrenamiento (80% aproximadamente), un 10% para realizar validaciones y 10% para realizar las pruebas finales. No obstante, la mayoría de los trabajos optan por la configuración 80% entrenamiento y 20% validación.

CAPÍTULO 3

MARCO TEÓRICO

El capítulo se encuentra dividido en dos secciones, primero se encuentra la información relacionada con las patologías presentes en las plantas y los bancos de imágenes utilizados en el presente trabajo y como segunda sección, se detalla el conocimiento computacional de las redes neuronales artificiales, convolucionales, el modelo *MobileNetV2*, visión artificial y las métricas de evaluación.

3.1 Patologías en las plantas

Una planta se enferma cuando un virus y/o bacteria infecta a la planta y presenta un crecimiento anormal. Las hojas de las plantas pueden variar desde su decoloración hasta la muerte. Las enfermedades son causadas debido a hongos, microbios, virus y nematodos (CANNA, s.f.).

a) Hongos

Son organismos pequeños, generalmente microscópicos, que se reproducen principalmente a través de esporas. La mayoría de los hongos tiene un cuerpo vegetativo filamentososo llamado micelio. El micelio da a los hongos una apariencia algodonosa. Ésta es una característica utilizada en el campo para distinguir las enfermedades causadas por hongos de aquellas causadas por bacterias (Bdigital.zamorano.edu, s.f.), ver Figura 3.1.

A continuación alguna enfermedades comunes:

- **Tizones.** Empardecimiento general y extremadamente rápido de las hojas, ramas y órganos florales de una planta, que dan como resultado la muerte de esos órganos.
- **Antracnosis.** Lesión que se asemeja a una úlcera profunda y se produce en todos los órganos de la planta.
- **Mildiú.** Zonas necróticas que por lo común se cubren con el micelio y los cuerpos fructíferos del hongo.



Figura 3.1. Enfermedad causada por hongo (CANNA, s.f.)

b) Virus

Los virus no tienen órganos reproductivos, sino que utilizan a las plantas que infectan para replicarse, son tan pequeños que no pueden ser vistos utilizando microscopios convencionales (Bdigital.zamorano.edu, s.f.).

Las características más importantes de los virus son las siguientes:

- Son tan pequeños que se necesitan técnicas especiales para su detección.
- Pueden ser de transmisión mecánica o necesitar vectores para su diseminación y transmisión.
- Son parásitos obligados, es decir que para multiplicarse o sobrevivir necesitan de un hospedero vivo.
- Son específicos. Suelen atacar plantas con características similares: hojas anchas anuales o herbáceas, hojas angostas o zacates, plantas perennes, pero rara vez un virus es capaz de atacar más de un grupo de plantas.

Síntomas mostrados por las plantas con virosis:

- Cambios permanentes y sistémicos (se presentan en todo nuevo crecimiento) en la coloración del follaje (mosaicos verdes o amarillos, venas oscuras o aclaradas).
- Ocurrencia de malformaciones.
- Subdesarrollo u otra inhibición del crecimiento.

c) Bacterias

Las bacterias son organismos unicelulares que se reproducen por fisión binaria (una célula se parte y se convierte en dos células idénticas) y generalmente necesitan de un medio de crecimiento rico en proteínas y con ambiente de alta humedad relativa para su infección, reproducción y diseminación (Bdigital.zamorano.edu, s.f.).

Las bacterias se diferencian de los hongos por no ser capaces de penetrar directamente tejidos de las plantas, sino que necesitan de heridas provocadas por insectos, pájaros, nematodos y por los humanos durante prácticas culturales como el trasplante, poda, etc. (Bdigital.zamorano.edu, s.f.).

Las bacterias fitopatógenas tienden a atacar follaje y frutos (por ejemplo mancha y peca bacteriana en solanáceas como chile) o ser problema en raíces (por ejemplo *Ralstonia* en tomate). A diferencia de los hongos, producen lesiones (manchas, pecas, putrefacción húmeda que despiden un mal olor, chancros, moteados, roñas y costras), pero nunca presentan micelios y son predominantes únicamente bajo condiciones de alta humedad relativa o encharcamiento del suelo (Bdigital.zamorano.edu, s.f.).

3.2 Desórdenes fisiológicos en las plantas

Los desórdenes fisiológicos o abióticos se distinguen de otro tipo de trastornos por el hecho de que no son causados por organismos vivos (tales como virus, bacterias, hongos, insectos, etc.), sino que son el resultado de la influencia de factores medioambientales, de

las prácticas culturales llevadas a cabo durante el desarrollo del cultivo y de mutaciones genéticas (CANNA, s.f.).

a) Estrés ambiental

Las plantas pueden estar sujetas a numerosos tipos de estrés ambiental. Sequía, temperaturas extremas (ver Figura 3.2) y exceso de luz son factores medioambientales que pueden afectar al desarrollo y calidad de las plantas. El estrés ambiental es muy difícil de controlar en cultivos al exterior, y aunque los invernaderos suavizan los efectos de estos factores, no pueden ser eliminados por completo (CANNA, s.f.).



Figura 3.2. Estrés por temperaturas extremas (CANNA, s.f.)

b) Estrés derivado de las prácticas de cultivo

Los factores causantes de estrés en las plantas, derivados de las prácticas de cultivo, se suelen dar en cultivos específicos de una determinada planta, en un lugar y con un objetivo definido. En los invernaderos, las prácticas de cultivo están diseñadas para estimular y mantener el crecimiento y desarrollo de una planta; sin embargo, también pueden ser causa de problemas abióticos (CANNA, s.f.). Ejemplos de este tipo de estrés es el exceso o carencia de nutrientes, exceso de riego (ver Figura 3.3), daños químicos, estrés mecánico por tocar o rozar y daños físicos.



Figura 3.3. Estrés por exceso de riego (CANNA, s.f.)

3.3 Bancos de imágenes

a) *PlantVillage*

El *dataset* es una vasta colección de imágenes, las cuales se encuentran clasificadas de acuerdo a su región (origen), especie y estado (enferma o sana). Es de suma importancia contar con un *dataset* clasificado puesto que existen una gran variedad de patologías en las plantas, que pueden ser causadas por virus, hongos, y bacterias.

Conforme a la lectura de los artículos que se seleccionaron para llevar a cabo el desarrollo de la tesis, se encontró que el *dataset* más utilizado en el área de la detección de enfermedades, mediante el análisis de imágenes es *PlantVillage* ya que, la síntesis que se presentó en el capítulo 2, reporta haber utilizado de manera parcial (para complementar su(s) propio(s) *dataset*(s)) o total (tomar tal cual las imágenes y construir el *dataset*) (ver la Tabla 2.23 del capítulo 2).

Gracias a que la mayoría de las imágenes alojadas en este *dataset* fueron capturadas en ambientes controlados, brinda la confianza de poder utilizar su repertorio para construir bancos de imágenes y con ello, crear modelos de clasificación a partir de las redes *CNN*.

El conjunto de datos de *PlantVillage* aloja un total de 54,303 imágenes (en *RGB*, escala de grises y segmentadas) de hojas sanas y no saludables divididas en 38 categorías por especies y enfermedades (Plant Village, s.f.). En la Figura 3.4 se ilustran tres tipos de enfermedades y una en estado saludable de la hoja de maíz.

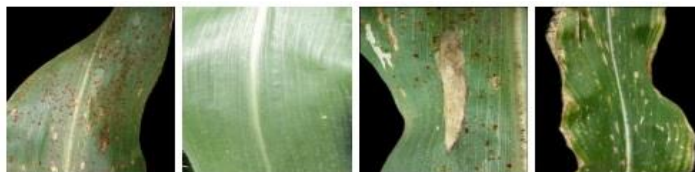




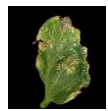




Figura 3.4. Muestras de hojas del maíz (Plant Village, s.f).

Algunas características de las fotografías son: las dimensiones de las capturas son de 256x256 píxeles; la perspectiva de captura de cada ejemplar en su mayoría muestran la hoja de manera vertical pero, existen enfoques en diagonal y horizontal en menor número. Es importante resaltar que la iluminación cambia en las imágenes ya que, en algunas se presentan sombras y cambios en el color de fondo gris, tendiendo a colores como: grises claros y oscuros, tonos blanquecinos y azul pálidos.

Revisando el conjunto *Plant village*, se seleccionaron las patologías correspondientes a las plantas de maíz y jitomate que se presentan en la Tabla 3.1. De estas imágenes, se dividieron en dos subconjuntos: el primero se integra de las fotos que ya se encontraban segmentadas y se utilizaron para construir el banco para la etapa de entrenamiento, el segundo se utilizó para la llevar a cabo la evaluación. La distribución de las imágenes se detalla en la sección 5.1.4.

Tabla 3.1. Selección de cultivos.

Cultivo	Saludable	Patologías	Agente patógeno	Muestra
Maíz	X	X	X	
Maíz		<i>Common rust</i>	Hongo (<i>Puccinia sorghi</i>)	
Maíz		<i>Northern leaf blight</i>	Hongo (<i>Exserohilum turcicum</i>)	
Jitomate	X	X	X	
Jitomate		<i>Bacterial spot</i>	Bacteria (<i>Xanthomonas vesicatoria</i> , <i>Xanthomonas euvesicatoria</i> , <i>Xanthomonas gardneri</i> , and <i>Xanthomonas perforans</i>)	
Jitomate		<i>Leaf mold</i>	Hongo (<i>Passalora Fulva</i>)	
Jitomate		<i>Tomato_yellow_leaf_curl_virus</i> -	Virus (Genus: <i>Begomovirus</i> , Family: <i>Geminiviridae</i>)	

Debido a que algunas de las muestras seleccionadas, de la colección de *PlantVillage*, para crear el segundo conjunto de fotografías presentaron irregularidades como: cambios en la iluminación, sombras y el constante cambio de los fondos los cuales tienen diferentes tonalidades de grises, blanco o azulados, fue necesario aplicar un pre procesamiento a las imágenes. Esta tarea se aborda en **4.1.1**.

b) Bancos de prueba final

Con la finalidad de realizar una prueba lo más cercana a la realidad, se buscaron dos bancos de imágenes adicionales, uno de cultivos de maíz y el otro de cultivos de jitomates. En la Tabla 3.2 se muestra la información básica de dichos conjuntos. En la sección **5.1.5** se precisa la construcción del *dataset*.

Tabla 3.2. Bancos de imágenes de prueba

Referencia	Nombre	Cultivo	Número de clases	Número de imágenes
Prajwala TM, 2018	<i>Tomato Leaf Diseases Detection</i>	Jitomate	7	6,348
Smaranji Ghose, 2020	<i>Corn Leaf Infection Dataset</i>	Maíz	4	4,188

Las características de las fotografías del *dataset Tomato Leaf Diseases Detection* son: las dimensiones varían según a cada clase (533x800, 388x800, 796x800...) píxeles; la perspectiva de captura de cada ejemplar (en su totalidad) muestran la hoja de manera vertical; la iluminación muestra cambios en las imágenes ya que, en algunas se presentan sombras y cambios en el color de fondo gris, tendiendo a colores como: grises claros y oscuros, tonos blanquecinos y café pálido. También existen en su minoría, 162 ejemplares con fondo color negro presentes en las clases: *late_blight* y *leaf_mold*.

Por otro lado en el banco de *Corn Leaf Infection Dataset* las resoluciones son múltiples (185x500, 256x256, 640x427 hasta 1500x1048) píxeles. Hay imágenes que presentan condiciones no controladas, es decir, se capturaron con diferentes perspectivas, enfoques, y diferente iluminación. El resto de imágenes presentan condiciones controladas. La perspectiva de la hoja en vertical abunda en la mayoría de las fotos; la iluminación cambia en cada muestra, en algunas se observan el fondo color: gris, gris claro, gris oscuro y blanco. El número de capturas con fondo negro es de 1,192 y pertenecen a la clase *common_rust*.

3.4 Inteligencia artificial

La inteligencia artificial (IA) es el campo de la ciencia informática dedicado a la resolución de problemas cognitivos asociados comúnmente con la inteligencia humana, como el aprendizaje, la resolución de problemas y el reconocimiento de patrones (Aws, s.f.).

La IA es una rama de las Ciencias Computacionales (CC) especializada en el desarrollo de algoritmos, cuyo objetivo es implementar dichas instrucciones en sistemas informáticos, capaces de realizar inferencias lógicas en tareas donde se requiere la percepción humana y que van más allá de la programación tradicional.

El aprendizaje automático (en inglés *Machine Learning*, ML) y el aprendizaje profundo (en inglés *Deep Learning*, DL) son ambos campos de la ciencia informática derivados de la disciplina de la inteligencia artificial (Aws, s.f.).

3.5 Redes neuronales

a) Redes neuronales artificiales

Artificial Neuronal Network (ANN) conocidas como redes neuronales artificiales (*RNAs*) son un modelo computacional de inspiración biológica que sigue el patrón de la red de neuronas presentes en el cerebro humano. Las redes neuronales artificiales también se pueden considerar como algoritmos de aprendizaje que modelan la relación entrada-salida. Las

aplicaciones de las redes neuronales artificiales incluyen el reconocimiento de patrones y la previsión en campos como la medicina, los negocios, las ciencias puras, la minería de datos, las telecomunicaciones y la gestión de operaciones (Nvidia Developer, s,f).

Una red neuronal artificial (ver Figura 3.5) transforma los datos de entrada aplicando una función no lineal a una suma ponderada de las entradas. La transformación se conoce como capa neuronal y la función se conoce como unidad neuronal. Las salidas intermedias de una capa, denominadas entidades, se utilizan como entrada en la siguiente capa. La red neuronal a través de transformaciones repetidas aprende en múltiples capas características no lineales (como bordes y formas) que luego combina en una capa final para crear una predicción (de objetos más complejos). La red neuronal aprende variando los pesos o parámetros de una red para minimizar la diferencia entre las predicciones de la red neuronal y los valores deseados. La fase en la que la red neuronal artificial aprende de los datos se denomina entrenamiento (Nvidia Developer, s,f).

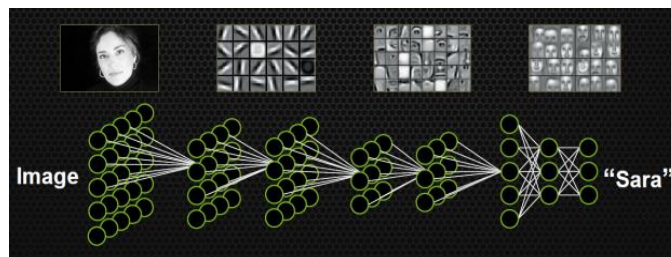


Figura 3.5. Representación esquemática de una RNA (Nvidia Developer, s,f).

b) Redes neuronales convolucionales

Una *Convolutional Neuronal Network (CNN)* o red neuronal convolucional (ver Figura 3.6) es un tipo de red neuronal artificial retroalimentada, en el cual el patrón de conectividad entre sus neuronas está inspirado en la organización de la corteza visual animal (Hanson A.J *et al.*, 2017).

Las neuronas corticales responden a estímulos en una región restringida del espacio conocida como campo receptivo. La respuesta de una neurona individual a los estímulos dentro de su campo receptivo puede ser, aproximada matemáticamente, por una operación de convolución. Las *CNN* consisten de múltiples capas de campos receptivos. Son pequeñas colecciones de neuronas las cuales procesan porciones de entrada de imagen. Las *CNN* tienen amplias aplicaciones en imágenes y reconocimiento por video (Hanson A.J *et al.*, 2017).

La capa convolucional guarda el resultado de los filtros. Esos filtros constan de pesos que contribuyen al aprendizaje. El propósito de la función de optimización es generar esos filtros que representan los datos sin error. Las capas de agrupamiento se utilizan para el muestreo descendente para reducir el tamaño de la neurona y reducir el sobreajuste. Las capas de función de activación son usadas para agregar no linealidad a la red. La función de activación más utilizada es *ReLu*. La capa *Dropout* se utiliza para prevenir el sobreajuste y apagar en cierta proporción las neuronas aleatoriamente en la red (H. Durmus *et al.*, 2017).

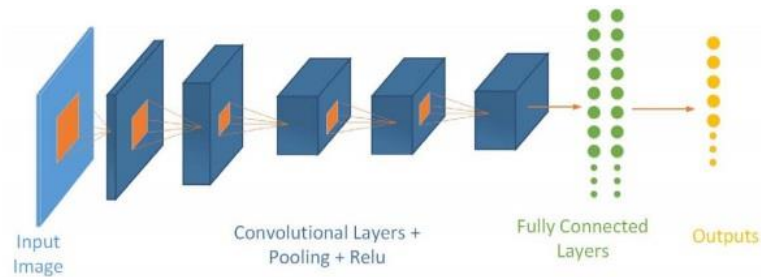


Figura 3.6. Red CNN (H. Durmus *et al.*, 2017)

En la mayoría de las redes convolucionales, cuanto más arriba está una capa, más especializada es. Las primeras capas aprenden características muy simples y genéricas que se generalizan a casi todos los tipos de imágenes. A medida que asciende, las características son cada vez más específicas del conjunto de datos en el que se entrenó el modelo (TensorFlow, s.f.).

c) *MobileNetV2*

MobileNetV2 es una arquitectura de red neuronal diseñada específicamente para los dispositivos móviles de recursos limitados (Mark Sandler *et al.*, 2019). Para poder realizar esto, se utiliza la idea de un bloque utilizado en muchas arquitecturas de redes neuronales: *depthwise separable convolutions*. Este consiste en reemplazar el operador convolucional por una versión factorizada que lo separa en varias capas. La primera capa, llamada *depthwise convolution*, realiza un filtrado aplicando un único filtro convolucional por canal de entrada, posible gracias a una convolución previa 1×1 .

La segunda capa es una convolución 1×1 lineal, llamada *pointwise convolution*, responsable de crear nuevas características a través de la combinación lineal de los canales de entrada. Se justifica la utilización de una convolución lineal ya que experimentalmente la no linealidad afecta a los resultados. Este tipo de convoluciones se denominan como *bottleneck* y se refiere al *ratio* entre el tamaño de la entrada a la convolución y el obtenido como *expansion ratio*. En la Figura 3.7 se observan los dos bloques utilizados en la red.

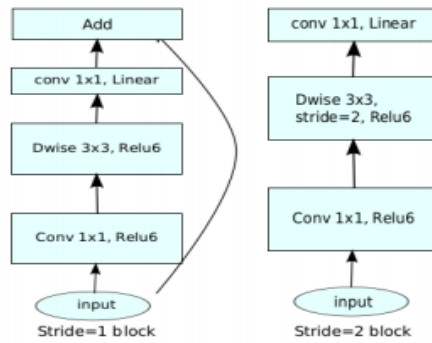


Figura 3.7. Bloques convolucionales de *MobileNetV2* (Mark Sandler *et al.*, 2019)

El primero (a la izquierda), muestra el bloque utilizado cuando se conserva el tamaño de entrada y salida de los mapas. Se emplean conexiones residuales con la finalidad de incrementar la habilidad del gradiente para ser propagado a través de múltiples capas.

El segundo bloque (a la derecha) ilustra cómo se hace la reducción del tamaño de los mapas al aplicar un *stride* de 2 píxeles. En ambos bloques se utiliza una variante de la *ReLU*, debido a su robustez cuando se emplean cálculos de baja precisión (Mark Sandler *et al.*, 2019). Por último se reduce el tamaño de los mapas a 1×1 a través de un operador *average pooling* y posteriormente, tras un *reshape*, se hace la tarea de clasificación. En la Tabla 3.3 se muestra de manera resumida otras propiedades de la red *MobileNetV2*.

Tabla 3.3. Características generales de la red *MobileNetV2* (Mark Sandler *et al.*, 2019)

Tamaño	Top-1 Acc (ImageNet)	Top-2 Acc (ImageNet)	Número de parámetros	Profundidad
14 Mbytes	0.713	0.901	3,538,984	88

3.6 Visión artificial

La visión artificial (VA) tiene el propósito de capturar imágenes y poder interpretar la información que hay en ellas, de manera similar a como lo hacen nuestros ojos y el procesamiento que hay en el cerebro (Solucioningenieril.com, s.f). Para poder simular estos procesos, los sistemas se integran de varias etapas que a continuación se listan:

Escenario a analizar: Es el área que se quiere capturar, donde se encuentra la información que se busca procesar.

Adquisición y digitalización: Es el proceso de capturar una imagen y pasarla a algún formato digital, se utiliza una cámara para capturar la escena y después se envía a una unidad donde pueda ser procesada.

Procesamiento previo: El dispositivo de captura a veces agrega ruido a las imágenes, por lo que debe de haber un pre-procesamiento para eliminarlo, para ello existen una gran

variedad de filtros. En ocasiones también es conveniente realizar transformaciones geométricas como recortes o rotaciones a los objetos para la siguiente etapa.

Segmentación: Su objetivo es localizar las regiones u objetos de interés en la imagen. Sus resultados son cruciales para todo el sistema de visión artificial.

Obtención de características: En esta etapa se extraen las características del objeto o regiones que son de interés. Las operaciones que comúnmente se realizan aquí son: detección de esquinas, colores, formas, realce de bordes, entre otros.

Reconocimiento e interpretación de información: En esta etapa se procesa la información obtenida en la fase previa, se le da una interpretación, y se aplica una acción según a lo analizado; dicha acción será la que controle a la aplicación.

3.7 Métricas de evaluación

Las métricas de evaluación permiten medir el desempeño del modelo generado de manera cuantitativa, con el objetivo de conocer realmente qué tan confiable es el producto creado durante el proceso de entrenamiento. Existen varias medidas para realizar esta evaluación, en este trabajo se seleccionaron las medias clásicas: precisión, exactitud, *recall* (*recuerdo*) y medida F1.

a) Matriz de confusión

La matriz de confusión es una tabla que describe el rendimiento de un modelo supervisado de *Machine Learning* en los datos de prueba, donde se desconocen los verdaderos valores. Se llama “matriz de confusión” porque hace que sea fácil detectar dónde el sistema está confundiendo dos o más clases (Sitiobigdata.com, s.f.).

Para efectos de la explicación y ejemplificación de los cálculos de las métricas, se asumirá que se tiene una evaluación binaria (dos clases) entonces, se tiene los siguientes valores que puede tomar cada casilla en la matriz de confusión:

- **True Positives (TP):** cuando el valor de la clase es 1 (Verdadero) y la predicha es también 1 (Verdadero)
- **True Negatives (TN):** cuando el valor de la clase es 0 (Falso) y el pronosticado también es 0 (Falso).
- **False Positives (FP):** cuando el valor de la clase es 0 (Falso) y el pronosticado es 1 (Verdadero).
- **False Negatives (FN):** cuando el valor de la clase es 1 (Verdadero) y el valor predicho es 0 (Falso).

En la Figura 3.8 se muestra una matriz de confusión binaria. La matriz de confusión en sí misma no es una medida de desempeño como tal, pero casi todas las métricas de desempeño se basan en la matriz de confusión y los números dentro de ella (Sitiobigdata.com, s.f.).

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figura 3.8. Matriz de confusión binaria (Sitiobigdata.com, s.f.)

Las métricas de evaluación son datos numéricos los cuales, permiten medir el desempeño del modelo generado esto, con el objetivo de conocer realmente qué tan confiable es el producto creado durante el proceso de entrenamiento. Existen varias medidas para realizar esta evaluación, a continuación, se explicarán las más utilizadas en el área de clasificación supervisada.

b) Exactitud

La Figura 3.9 muestra una matriz de confusión binaria y una ecuación para efectuar el cálculo de la exactitud del modelo.

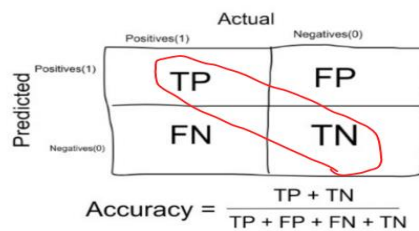


Figura 3.9. Cálculo de la exactitud (Sitiobigdata.com, s.f.)

Esta métrica es intuitiva ya que, no es apropiada en muchos casos (Sitiobigdata.com, s.f.). Por ejemplo, cuando se tienen clases desbalanceadas (demasiada diferencia de elementos entre las clases) la medida de exactitud se inclinará por la clase con más elementos. Por lo tanto, se recomienda tener las clases balanceadas para que la exactitud sea un valor fiable.

c) Recall, Sensibilidad o TPR (Tasa de True Positive Rate)

Es el número de elementos identificados correctamente como positivos del total de positivos verdaderos, en otras palabras, da información sobre el rendimiento de un clasificador con respecto a falsos negativos (cuántos fallaron) (Sitiobigdata.com, s.f.).

En la Figura 3.10 se ilustra una matriz de confusión binaria y una ecuación para efectuar el cálculo de la sensibilidad del modelo.

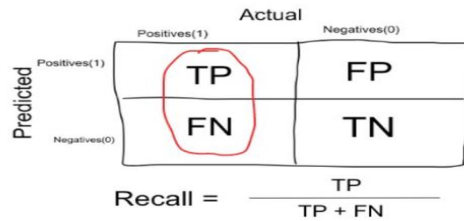


Figura 3.10. Cálculo de la sensibilidad (Sitiobigdata.com, s.f.)

d) Precisión

Es el número de elementos identificados correctamente como positivo de un total de elementos identificados como positivos, proporciona información sobre su rendimiento con respecto a los falsos positivos (cuántos capturados) (Sitiobigdata.com, s.f.). En la Figura 3.11 se expone una matriz de confusión binaria y una ecuación para efectuar el cálculo de la precisión del modelo.

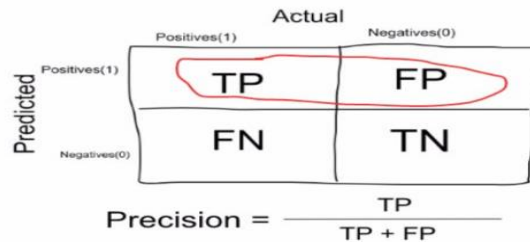


Figura 3.11 Cálculo de la precisión (Sitiobigdata.com, s.f.)

e) F1

La métrica $F1$ es la media armónica de la sensibilidad y la precisión. La media armónica se utiliza en lugar de un promedio simple porque castiga los valores extremos. Por ejemplo, si se tiene un clasificador con una precisión de 1.0 y una sensibilidad de 0.0 tiene un promedio simple de 0.5 pero con una puntuación $F1$ de 0. La puntuación $F1$ da el mismo peso a ambas medidas. La Figura 3.12 indica la ecuación del cálculo del puntaje $F1$.

$$F_1 = 2 * \frac{\textit{precision} * \textit{recall}}{\textit{precision} + \textit{recall}}$$

Figura 3.12. Cálculo de la métrica $F1$ (Sitiobigdata.com, s.f.)

3.8 Lenguaje de programación, librerías y entornos de desarrollo

a) Python

Python es un lenguaje de programación que permite trabajar más rápidamente e integrar sistemas de manera más efectiva, es poderoso, rápido, amigable y fácil de aprender. *Python* se desarrolla bajo una licencia de código abierto aprobada por OSI (Open Source

Initiative), lo que lo hace de libre uso y distribución, incluso para uso comercial. La licencia de *Python* es administrada por *Python Software Foundation* (Python, s.f).

b) JavaScript

Javascript es un lenguaje poderoso, capaz de aportar soluciones eficaces en la mayoría de los ámbitos de la tecnología. Es el único lenguaje de programación que entienden los navegadores, con el que se desarrolla la parte de la funcionalidad *frontend* en sitios *web* y aplicaciones *web* modernas (Desarrolloweb.com, s.f.). Sus usos más importantes son los siguientes:

- Desarrollo de sitios web del lado del cliente (*frontend*, en el navegador)
- Desarrollo de todo tipo de aplicaciones gracias a la plataforma *NodeJS*
- Desarrollo de aplicaciones para dispositivos móviles, híbridas o que compilan a nativo
- Desarrollo de aplicaciones de escritorio para sistemas *Windows*, *Linux* y *Mac*, permitiendo escribir un código compatible con todas las plataformas.

Por tanto, es posible considerar a *Javascript* el lenguaje universal, ya que permite diferentes tipos de aplicaciones y usos en la actualidad (Desarrolloweb.com, s.f.).

c) Kotlin

Kotlin es un lenguaje de programación expresivo y conciso que permite reducir los errores de código comunes y se integra fácilmente en las *apps* existentes. *Android Studio* brinda compatibilidad óptima con *Kotlin*. Además, viene con herramientas incorporadas para convertir el código basado en *Java* a *Kotlin*. La herramienta “Mostrar código de *bytes*” de *Kotlin* permite ver el código basado en *Java* equivalente a medida que aprendes a usar *Kotlin* (Android Developers, s.f.).

d) Keras

Keras es una *API* de aprendizaje profundo escrita en *Python*, que se ejecuta sobre la plataforma de aprendizaje automático *TensorFlow*. Fue desarrollado con un enfoque en permitir una experimentación rápida (Keras, s.f.).

En esta biblioteca se encuentra la red protagonista de esta tesis “*MobileNetV2*” la cual, fue seleccionada en base a la investigación que se realizó del estado del arte, posteriormente se estudió, se llevó a cabo el experimento y se obtuvo el modelo predictivo.

Keras permite a los ingenieros e investigadores aprovechar al máximo la escalabilidad y las capacidades multiplataforma de *TensorFlow 2*: puede ejecutar *Keras* en *TPU* o en grandes grupos de *GPU*, y puede exportar sus modelos de *Keras* para ejecutarlos en el navegador o en un dispositivo móvil (GoogleColab, s.f.).

e) TensorFlow

TensorFlow es una plataforma de código abierto para el aprendizaje automático (AA). Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que permite que los investigadores innoven con el aprendizaje automático y los desarrolladores creen e implementen aplicaciones con tecnología de AA fácilmente (Keras,

s.f.). Dentro de esta plataforma se encuentran dos bibliotecas, las cuales brindan las herramientas adecuadas para poder crear aplicaciones en diferentes entornos.

TensorFlow.js es una biblioteca de *JavaScript* para entrenar e implementar modelos en el navegador y en *Node.js* (Keras, s.f.). Por otro lado *TensorFlow Lite* es una biblioteca ligera para implementar modelos en dispositivos integrados y móviles (TensorFlow, s.f.).

f) GoogleColab

GoogleColab es una herramienta poderosa, muy conocida y utilizada en el área de la *IA* ya que, la empresa *Google* brinda de manera gratuita, acceso a máquinas virtuales con un alto poder en hardware. Principalmente, permite el uso de tarjetas *GPU* que son de gran ayuda para llevar a cabo, de manera más rápida, ciertas operaciones (extracción de características, creación de filtros, agrupación de características, reducción, etc.) que implican la creación de modelos matemáticos en un tiempo menor.

Colab permite importar un conjunto de datos de imágenes, entrenar un clasificador con dicho conjunto de datos y evaluar el modelo con tan solo unas pocas líneas de código. Los cuadernos de *Colab* ejecutan código en los servidores en la nube de *Google*, lo que permite aprovechar la potencia del hardware de *Google*, incluidas las *GPU* y *TPU*, independientemente de la potencia de tu equipo. Lo único que se necesita es un navegador (*GoogleColab*, s.f.).

Posee una interfaz muy amigable para el usuario final en donde puede organizar los proyectos por cuadernillos de trabajo, organizar el código en formato de bloques y comentar de manera muy sofisticada los bloques de código que se encuentren en dicho proyecto.

GoogleColab permite ejecutar código en lenguaje Python por lo tanto, es posible utilizar librerías escritas en este lenguaje (utilizando *Keras*, *TensorFlow*, *Pytorch*, por mencionar algunos). Además, es posible ejecutar líneas de comando *Linux* (*cd*, *rm*, *mv*, etc...).

Gracias a que *Google* pone a disposición dicha herramienta en la nube y de manera gratuita, contribuye a que la comunidad de la *IA* crezca día con día.

g) AndroidStudio

Android Studio es el entorno de desarrollo integrado (*IDE*) oficial para la implementación de *apps* para *Android* (Android Developers, s.f.). Además del potente editor de códigos y las herramientas para desarrolladores, *Android Studio* ofrece más funciones que aumentan la productividad cuando se desarrollan *apps* para *Android*, como las siguientes:

- Un sistema de compilación flexible
- Un emulador rápido y cargado de funciones
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
- Integración con *GitHub* y plantillas de código para ayudarte a compilar funciones de *apps* comunes y también importar código de muestra
- Compatibilidad con C++ y *NDK* (Native Developer Kit)
- Compatibilidad integrada con *Google Cloud Platform*,

CAPÍTULO 4

ANÁLISIS Y DISEÑO

DEL SISTEMA

En este capítulo se explica la presentación final de las aplicaciones móvil y web hacia el usuario final, así como el diseño del sistema, junto con los elementos presentes en los respectivos ambientes. También, se detallan las técnicas utilizadas en los módulos: preprocesamiento de imágenes, aumento de datos, ajuste fino y validación cruzada; fases empleadas para efectuar las dos experimentaciones que se exponen en el **capítulo 5**.

4.1 Técnicas

4.1.1 Pre procesamiento

El pre procesamiento consistió en homogeneizar el fondo de las imágenes a negro, de tal forma que las entidades visuales presentes en las mismas tengan un mayor contraste.

Para efectuar esta etapa se recurrió a una aplicación que se encuentra en la *web RemoveBackground* (Remove Back Ground, s.f.) la cual, brinda la facilidad de remover el fondo de la fotografía (ver Figura 4.1) en su totalidad hasta obtener la imagen deseada.



Figura 4.1. Segmentación transparente

La aplicación *web* proporciona una serie de herramientas: borrado, restauración manual, fondo personalizado, fondo transparente, etc. Las cuales fueron de gran utilidad ya que, como se observa en la Figura 4.2 algunas segmentaciones fueron fallidas debido a que, la iluminación en ciertas fotografías, la sombra permaneció al final del proceso.



Figura 4.2. Segmentación con sombra

Para atender este problema, se utilizaron las herramientas de borrado manual (Figura 4.3) y cambio de color de fondo (Figura 4.4).

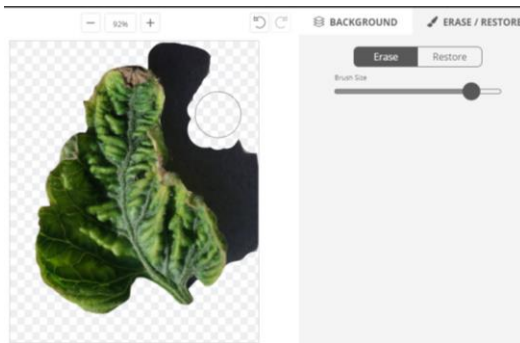


Figura 4.3. Borrado manual

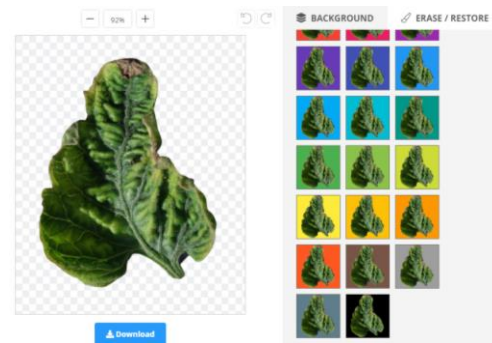


Figura 4.4. Selección de color

Finalmente, se colocó el fondo de color negro (Figura 4.5) a cada una de las muestras que conformarán el banco de imágenes.

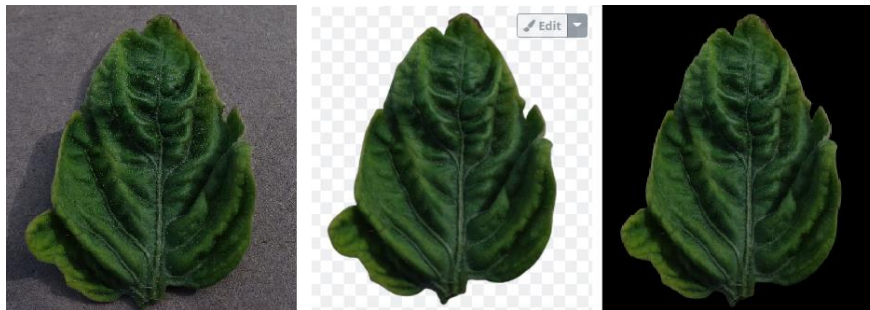


Figura 4.5. Fondo en color negro

Al llevar a cabo esta práctica, se está asegurando que la calidad de cada muestra mejora al eliminar completamente los fondos de colores irregulares y sombras (impurezas o ruido),

de esta manera se garantiza que la red *CNN* pueda extraer características de las hojas presentes en las fotografías de manera más imoluta.

4.1.2 Aumento de datos

El *Data Augmentation* o aumento de datos es una técnica que se utiliza para hacer crecer de manera artificial el número de fotografías originales, al efectuar una serie de transformaciones (rotaciones, traslaciones, cambios en la simetría, añadir ruido, etc.) con la finalidad de obtener un *dataset* que pueda representar lo más cercano posible las diferentes condiciones de los especímenes que se encuentran en el mundo real. En otras palabras, se busca obtener una mayor riqueza de ejemplares que conforman el banco de imágenes. En las Figuras 4.6 y 4.7 se aprecian dichas transformaciones para una hoja de maíz y otra de jitomate.



Figura 4.6. Hojas de maíz



Figura 4.7. Hojas de jitomate

La selección y configuración de los parámetros de cada una de las técnicas de *data augmentation* juega un papel muy importante ya que se debe vigilar que las imágenes generadas realmente representen información real y no se incluya ruido o perspectivas erróneas que provoquen el problema del sobre entrenamiento (*overfitting*); es decir, que el modelo solamente se especialice (memorice) en el conjunto de imágenes propuesto para el experimento y no sea capaz de generalizar (predecir otras fotografías) no vistas durante el entrenamiento. En la sección 5.1.2 se detallan las transformaciones aplicadas.

4.1.3 Hiperparámetros

Los hiperparámetros de un modelo son los valores de las configuraciones utilizadas durante el proceso de entrenamiento. Son valores que generalmente no se obtienen de los datos, por lo que suelen ser indicados por el científico de datos, en este caso, se tomaron los valores de referencia de los artículos seleccionados en el estado del arte. Sin embargo, durante el entrenamiento se obtuvieron distintos rendimientos, por lo cual, se experimentaron con varias configuraciones. En la sección 5.1.3, se muestran las configuraciones empleadas.

4.1.4 Validación cruzada

Esta técnica consiste en dividir los datos (*dataset*) disponibles en K particiones (normalmente $K = 4$ o 5) y entrenar cada uno en $K - 1$ particiones mientras se evalúan las particiones restantes. El *score* de validación para el modelo utilizado es entonces el promedio de las puntuaciones de validación K obtenidas (François Chollet, 2019).

4.1.5 Transferencia de aprendizaje

La intuición detrás del aprendizaje por transferencia para la clasificación de imágenes es que si un modelo (preentrenado) se entrena en un conjunto de datos lo suficientemente grande y general, este modelo servirá efectivamente como un modelo genérico del mundo

visual (TensorFlow, s.f.). No es necesario re entrenar todo el modelo. La red convolucional base (preentrenada) ya contiene características que son genéricamente útiles para clasificar imágenes. Sin embargo, la parte final de clasificación del modelo preentrenado es específica de la tarea de clasificación original (TensorFlow, s.f.). En la Figura 4.8 se muestra el proceso de la transferencia de aprendizaje.

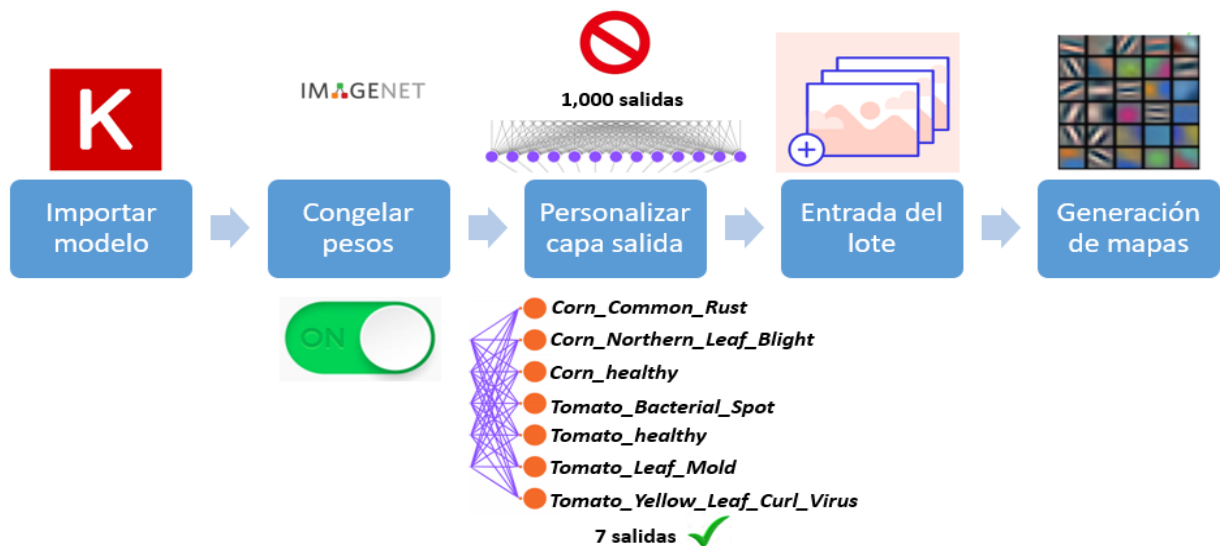


Figura 4.8. Mapa de proceso de transferencia de aprendizaje

A continuación se explican las cinco etapas que conforman el proceso de transferencia de aprendizaje:

Importar modelo: Se hace la llamada del modelo *MobileNetV2* utilizando las librerías de *Keras* (Keras, s.f.) y *TensorFlow* (TensorFlow, s.f.).

Cargar y Congelar pesos: Los pesos originales provenientes del *dataset ImageNet* son deshabilitados para que no se ven afectado durante el primer entrenamiento. Este paso es importante para efectuar la transferencia de aprendizaje de lo contrario, se estaría efectuando un entrenamiento desde cero.

Personalizar capa: Se procede a retirar la capa de salida original de mil salidas y se coloca la capa particular de siete salidas con pesos iniciados aleatoriamente, conforme a las clases a clasificar en este trabajo de tesis.

Entrada de imágenes: Ingresan los lotes de imágenes hacia la red para ser procesadas por medio de la operación convolución y con ello, crear mapas de características a partir de la partición de entrenamiento; la partición de validación se utiliza para validar el progreso de aprendizaje.

Generación de mapas: A lo largo del entrenamiento, las imágenes son transformadas y se construyen colecciones de mapas de características cada vez más elaborados. Dicho en otras palabras a medida que ocurre el entrenamiento, la red crea nuevos pesos para la última capa a partir de los mapas de características que van aumentando.

4.1.6 Ajuste fino

El ajuste fino es una técnica complementaria del proceso de entrenamiento. El objetivo de este método es adaptar estas características especializadas para trabajar con el nuevo conjunto de datos, en lugar de sobrescribir el aprendizaje genérico (TensorFlow, s.f.). Es decir, en el experimento de transferencia de aprendizaje (fase 1), solo se estaban entrenando algunas capas sobre un modelo base de *MobileNetV2* y, los pesos de esta red pre entrenada no se actualizaron durante el entrenamiento. Por lo tanto, una forma de aumentar el rendimiento es entrenar (o "ajustar") los pesos de las capas superiores del modelo previamente entrenado junto con el entrenamiento del clasificador que agregó. El proceso de entrenamiento obligará a ajustar las ponderaciones de los mapas de características genéricas a las características asociadas específicamente con el conjunto de datos (TensorFlow, s.f.).

Esto sólo se debe de llevar a cabo después de haber culminado la primera fase y previamente configurado como "no entrenable" (congelar pesos originales) de lo contrario se estarían inicializado aleatoriamente los pesos originales y todas las capas conjuntamente estarían disponibles por lo que, la magnitud de las actualizaciones del gradiente será demasiado grande (debido a los pesos aleatorios del clasificador) y el modelo previamente entrenado olvidaría de lo que aprendió (TensorFlow, s.f.). La Figura 4.9 muestra el proceso de ajuste fino.

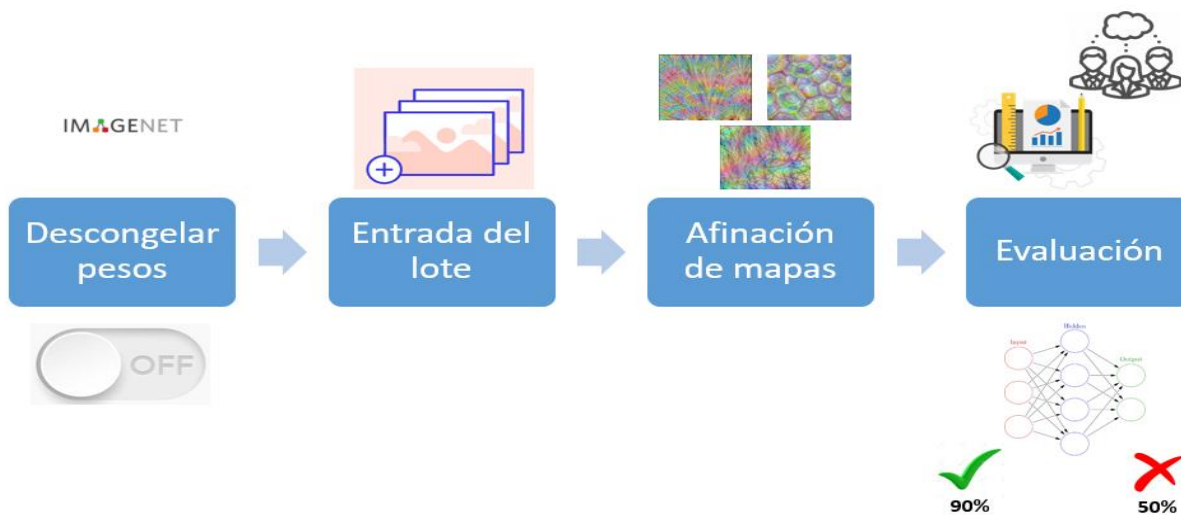


Figura 4.9. Mapa de proceso de afinación

A continuación se describen las cuatro etapas que conforman el proceso de afinación:

Descongelar pesos: Se desbloquean los pesos originales de la red que anteriormente se habían deshabilitado en la etapa anterior.

Entrada de imágenes: Ingresan las imágenes hacia la red para ser transformadas por medio de la operación convolución y crear mapas de características, al utilizar la partición de entrenamiento; la partición de validación se utiliza para validar el progreso de aprendizaje a lo largo del proceso.

Afinación de los mapas: Ya que, los pesos originales están habilitados, se efectúa un proceso de actualización del conocimiento original con respecto al creado en la etapa anterior. En este punto la red neuronal lo que hace es ajustar los pesos pero, sin alterar abruptamente los pesos originales, al tomar el conocimiento obtenido en la etapa anterior.

Evaluación: Al culminar el proceso de afinación se obtiene el producto final el cual, es sometido a las métricas de evaluación para cuantificar su desempeño y tomar las decisiones pertinentes (implementar o experimentar).

4.2 Diseño del sistema

4.2.1 Versión Móvil

El desarrollo del proyecto en su versión móvil, se utilizó el entorno de desarrollo *Android studio* versión 2020.3.1, *Windows 64-bit* (912 MiB). El lenguaje de programación empleado fue *Kotlin*.

Kotlin es un lenguaje de programación moderno de tipo estático que usan más del 60% de los desarrolladores profesionales de *Android*. *Kotlin* ayuda a aumentar la productividad, la satisfacción de los desarrolladores y la seguridad del código (Android Developers, s.f.).

En la Figura 4.10 se presenta la interfaz gráfica principal de la aplicación y el etiquetado de elementos presentes al momento de ejecutar la *app* en el teléfono.

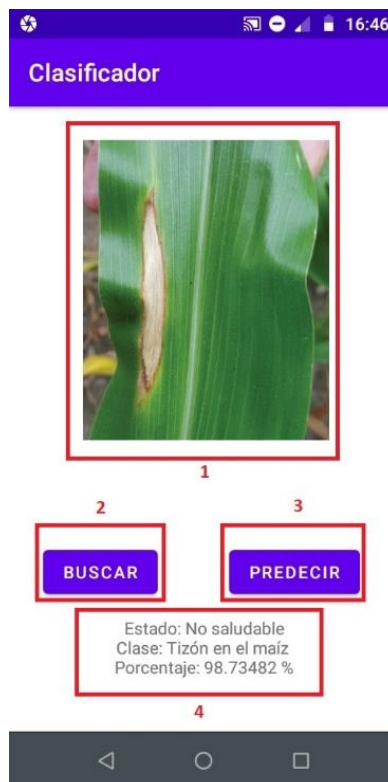


Figura 4.10. Interfaz gráfica versión móvil

Como se señala en la imagen, hay cuatro elementos presentes en la vista gráfica para el usuario:

1. **Muestra:** ilustra la fotografía que el usuario desea analizar.
2. **Buscar:** el botón buscar permite acceder a los directorios del teléfono y seleccionar la imagen que sea de interés.
3. **Predecir:** ejecuta la funcionalidad de predecir.
4. **Descripción:** imprime en el campo la predicción hecha por el modelo predictivo; se muestra en pantalla el estatus que puede ser sano o enfermo, la enfermedad, el cultivo, y el porcentaje de predicción, el cual va del 0-100 %.

4.2.2 Versión Web

La construcción para la versión web, se utilizó el lenguaje de programación *java script* e importaciones de las librerías de *TensorFlowJs*; también, se empleó el lenguaje de marcado *HTML* para generar la vista *Bootstrap* para sólo dar un poco de orden a los elementos presentes en la interfaz gráfica.

Para lograr el acceso vía *http*, se creó un servidor local simple con ayuda del módulo *SimpleHTTPServer* de *Python*.

La Figura 4.11 ilustra la interfaz gráfica principal de la aplicación y el etiquetado de elementos presentes al momento de acceder a la página web desde la computadora.

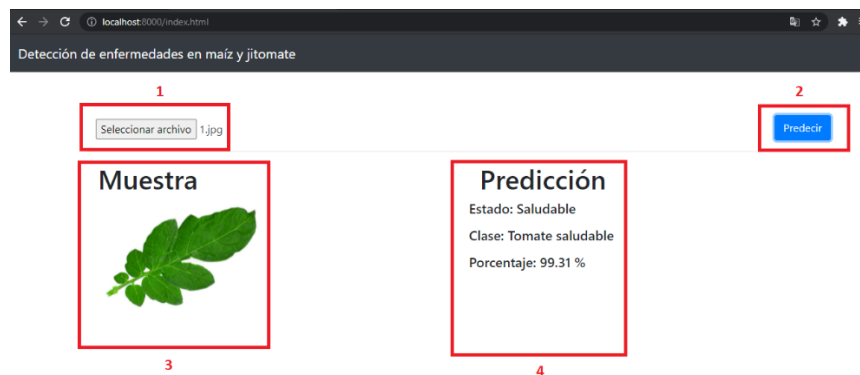


Figura 4.11. Interfaz gráfica versión web

Como se ilustra en la imagen, hay cuatro elementos presentes en la interfaz gráfica para el usuario:

1. **Seleccionar archivo:** permite acceder a los directorios de la computadora y seleccionar la imagen de interés.
2. **Predecir:** Ejecuta la funcionalidad de predecir.
3. **Muestra:** ilustra la fotografía que el usuario desea analizar.
4. **Predicción:** Imprime en el campo la predicción hecha por el modelo predictivo; se muestra en pantalla el estatus que puede ser sano o enfermo, la enfermedad, el cultivo, y el porcentaje de predicción, el cual va del 0-100%.

4.2.3 Mapa de procesos de predicción

Cabe mencionar que las dos versiones del proyecto (web y móvil) sólo se desarrollaron como una evidencia simple en donde, se demuestra la implementación del modelo predictivo generado por la red *CNN MobileNetV2*.

La Figura 4.12 detalla el proceso de predicción que se ejecuta en ambos ambientes de desarrollo. El formato, condiciones y dimensiones se detallan en la sección **5.4.3 Recomendaciones de uso del capítulo 5**.

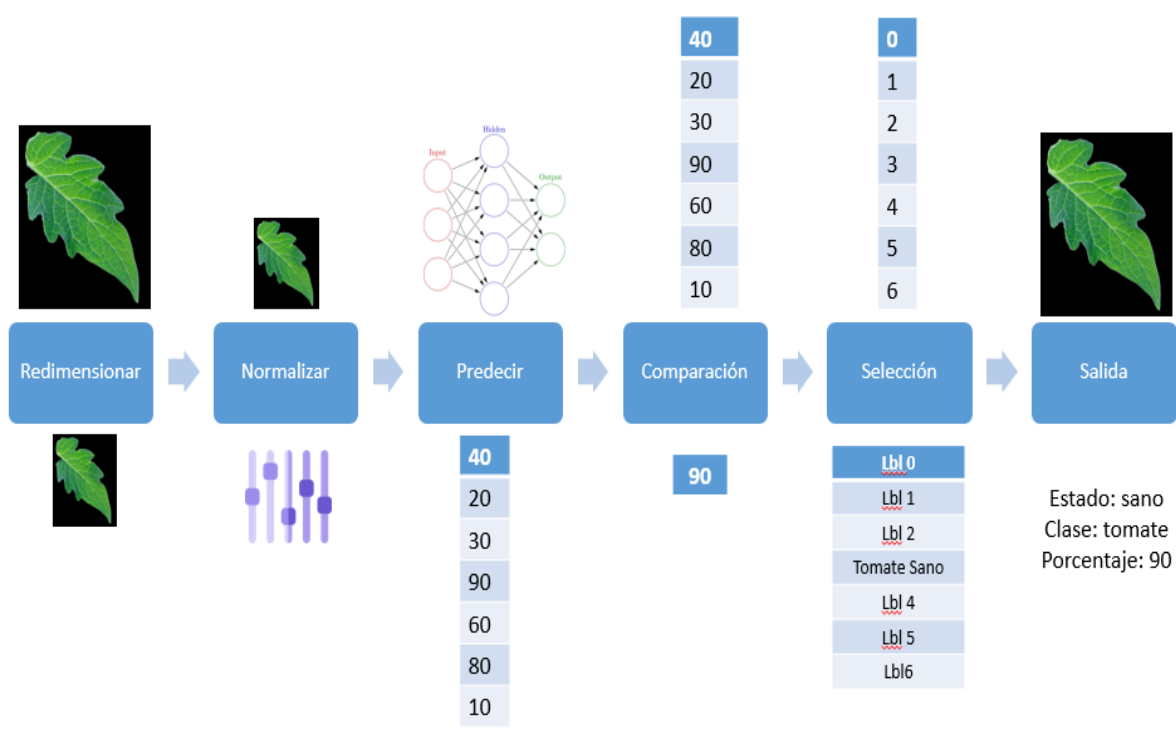


Figura 4.12. Etapas del proceso de predicción.

Enseguida se detallan las seis etapas presentes en el proceso de predicción:

Redimensionar: se hace un ajuste al tamaño de $224 \times 224 \times 3$ a la imagen puesto que, el modelo requiere que la fotografía de entrada cumpla con dicha resolución.

Normalizar: enseguida se aplica una normalización al mapa de píxeles con la finalidad de obtener valores $[-1, 1]$ en cada píxel.

Predecir: se llama al método *predict(img)* el cual, recibe como parámetro la foto ya normalizada y como resultado, devuelve un arreglo de dimensión 1×7 , el cual contiene las predicciones generadas por el modelo.

Comparación: se selecciona el valor máximo de la tupla y se guarda la posición del elemento mayor en una variable.

Selección: con el índice máximo, se accede a la posición de la lista que aloja las clases y se obtiene la etiqueta.

Salida: se crea el mensaje y se asigna en una variable para su impresión en pantalla.

CAPÍTULO 5

EXPERIMENTACIÓN

Y RESULTADOS

En este capítulo se detallan las configuraciones previas al experimento (hiperparámetros, configuración y distribución de las particiones). Se realiza la experimentación y el análisis de los resultados obtenidos recurriendo a las métricas de evaluación y en base a ello, se opta por el mejor modelo predictivo y se exporta para su uso en los respectivos ambientes de ejecución.

5.1 Experimentación

Las experimentaciones que se realizaron en este trabajo de tesis fueron dos: caso A y caso B. A continuación se detallan los objetivos, sus configuraciones respectivamente.

El objetivo del caso de experimentación A es: crear tres modelos que puedan predecir el estado de salud en los cultivos del maíz y jitomate utilizando la validación cruzada.

El objetivo del caso de experimentación B es: generar cinco modelos capaces de pronosticar el estado de salud en el maíz y jitomate empleando la validación cruzada.

5.1.1 Aumento de datos

A continuación, en la Tabla 5.1 se enlistan las técnicas que se aplicaron respectivamente en cada experimento, las imágenes consideradas fueron solamente las que se utilizan en la partición de entrenamiento; la partición de validación quedó sin transformaciones.

Tabla 5.1. Configuraciones de aumento de datos.

CASO A	CASO B
1. Rotaciones: 45°	1. Rotaciones: 45°
2. Acercamiento (<i>zooming</i>): 30%	2. Acercamiento (<i>zooming</i>): 30%
3. Efecto espejo horizontal (<i>horizontal_flip</i>)	3. Efecto espejo horizontal (<i>horizontal_flip</i>)
4. Efecto espejo vertical (<i>vertical_flip</i>)	4. Efecto espejo vertical (<i>vertical_flip</i>)
	5. Cambios en la iluminación (20% obscurecimiento , 20% iluminación)

5.1.2 Hiperparámetros

En la Tabla 5.2, se presenta la configuración final de los hiperparámetros empleados durante la experimentación.

Tabla 5.2. Configuraciones de hiperparámetros.

CASO A	CASO B
<ul style="list-style-type: none"> • Número de clases: 7 • Número de épocas (<i>transfer learning</i>): 10; Número de épocas (<i>fine tuning</i>): 50 • Función de activación: <i>softmax</i> • <i>Rate-learning</i>: 0.0001 (<i>transfer learning</i> y <i>fine tuning</i>) • <i>Batch size</i>: 16 • Resolución de imágenes: 224x224x3 	<ul style="list-style-type: none"> • Número de clases: 7 • Número de épocas (<i>transfer learning</i>): 10; Número de épocas (<i>fine tuning</i>): 50 • Función de activación: <i>softmax</i> • <i>Rate-learning</i>: 0.0001 (<i>transfer learning</i> y <i>fine tuning</i>) • <i>Batch size</i>: 32 • Resolución de imágenes: 224x224x3

5.1.3 Creación de particiones

Con la finalidad de buscar un modelo matemático predictivo confiable y corroborar que las técnicas que se aplicaron en los bancos de imágenes (ver Tabla 5.3) fueron las idóneas para obtener un producto fiable, se aplicó la técnica de validación cruzada.

Tabla 5.3. Bancos de imágenes

Cultivo	Saludable	Patologías	Banco de imágenes 1	Banco de imágenes 2	Suma total
Maíz	X	-	1,161	1,162	2,323
Maíz		<i>Common rust</i>	1,189	1,192	2,381
Maíz		<i>Northern leaf blight</i>	982	1,018	2,000
Jitomate	X	-	1,148	1,150	2,298
Jitomate		<i>Bacterial spot</i>	1,162	1,120	2,282
Jitomate		<i>Leaf mold</i>	1,100	1,019	2,119
Jitomate		<i>Tomato_yellow_leaf_curl_virus</i>	1,166	1,142	2,308
			Total: 7,908	Total: 7,803	Total: 15,711

Para el caso A la configuración de la validación cruzada quedó con un $K=3$; las divisiones de las particiones se muestran en la Tabla 5.4. Ya que la división entre 3 generó valores irracionales, el porcentaje de entrenamiento se tuvo que redondear a 67% (66.66) y el porcentaje de validación se truncó en 33% (33.33).

Para el caso de las imágenes ocurrió exactamente lo mismo, se tuvo que redondear (cuando la décima fue mayor o igual a 5) y truncar (cuando la décima fue menor a 5) para trabajar con valores enteros en los lotes de imágenes (Ver la Tabla 5.5).

Tabla 5.4. Referencias caso A.

Referencia	Rol	Distribución
E	Entrenamiento	67%
V	Validación	33%

Tabla 5.5. Tamaño de cada lote de imágenes del caso A.

Clase	Tamaño de entrenamiento	Tamaño de validación
Corn_healthy	1,548	774
Corn_common_rust	1,588	794
<i>Northern leaf blight</i>	1,344	667
Tomato_healthy	1,532	766
Tomato_bacterial_spot	1,522	761
Tomato_leaf_mold	1,412	706
Tomato_yellow_leaf_curl_virus	1,538	769
Total de imágenes	10,484	5,237

Por otro lado, en el caso B la configuración de la validación cruzada quedó con un $K=5$; la Tabla 5.6 muestra los porcentajes de datos considerados para el entrenamiento y la validación.

Nuevamente al momento de crear los lotes de imágenes se produjeron valores irracionales por lo tanto, se tuvo que redondear y trunca para trabajar con valores enteros (Ver Tabla 5.7).

Tabla 5.6. Referencias caso B

Referencia	Rol	Distribución
E	Entrenamiento	80%
V	Validación	20%

Tabla 5.7. División de la partición de entrenamiento y validación, caso B.

Clase	Tamaño de Entrenamiento	Tamaño de Validación
Corn_healthy	1,858	465
<i>Common rust</i>	1,905	476
<i>Northern leaf blight</i>	1,600	400
Tomato_healthy	1,838	460
<i>Bacterial spot</i>	1,826	456
<i>Leaf mold</i>	1,695	424
<i>Tomato_yellow_leaf_curl_virus</i>	1,846	462
Total de imágenes	12,568	3,143

5.1.4. Conjunto de imágenes para la prueba final

Se consideró pertinente realizar una prueba cercana a la realidad utilizando fotografías que no pertenecen a los conjuntos de entrenamiento y validación antes mencionados; es decir, que tienen características completamente diferentes. De esta manera se garantiza que los valores que obtienen las métricas de evaluación son datos de confianza. Para poder llevar a cabo esta etapa, se buscaron imágenes de otros *datasets* públicos. En la Tabla 5.8 se da a conocer el número de muestras utilizadas, así como, la causa de la enfermedad y fuente de consulta (*dataset*). Esta configuración se emplea en las secciones **5.2.5 y 5.2.8**.

Tabla 5.8. Banco de imágenes prueba final.

Fuente	Cultivo	Saludable	Patologías	Número de imágenes
Smaranji Ghose, 2020	Maíz	X		35
Smaranji Ghose, 2020	Maíz		<i>Common rust</i>	34
Smaranji Ghose, 2020	Maíz		<i>Northern leaf blight</i>	36
Prajwala TM, 2018	Jitomate	X		35

Tabla 5.9. Banco de imágenes prueba final *continuación*.

Fuente	Cultivo	Saludable	Patologías	Número de imágenes
Prajwala TM, 2018	Jitomate		<i>Bacterial spot</i>	36
Prajwala TM, 2018	Jitomate		<i>Leaf mold</i>	35
Prajwala TM, 2018	Jitomate		<i>Tomato_yellow_leaf_curl_virus_</i>	38

5.2 Configuración de los Experimentos

Tanto en el caso A como en el B los pasos 1 y 2 son idénticos e indispensables para comenzar con el proceso de transferencia de aprendizaje.

1. Se cargaron los pesos originales de la red *CNN* provenientes del conjunto de datos *ImageNet*. Es importante congelar los pesos originales para cumplir con la técnica de transferencia de aprendizaje. En caso contrario, desde el inicio se estaría perdiendo el conocimiento original y por consecuencia, se estaría haciendo un entrenamiento tradicional con pesos inicializados de manera aleatoria.
2. Se retiró la última capa del modelo original debido a que en la capa final hay 1,000 salidas. Evidentemente lo hace incompatible con el proyecto puesto que, en el trabajo de tesis sólo se contemplan siete salidas. Por lo tanto, se elimina y se coloca una capa particular en función al número de clases.

5.2.1 Transferencia de aprendizaje

La transferencia de aprendizaje consta de las siguientes etapas: se extrae las propiedades de cada imagen que la red recibe, se generan mapas de características y a su vez se comienzan a crear nuevos pesos en la última capa modificada. Conforme van transcurriendo las épocas definidas, dichos pesos se van actualizando sin afectar los pesos originales debido a la configuración previa que se realizó.

La Figura 5.1 muestra las curvas de aprendizaje obtenidas en la etapa de transferencia de aprendizaje. Como se observa el comportamiento es el deseado porque, las curvas convergen suavemente; es decir, no se aprecian cambios abruptos; sin embargo, la pérdida es alta y por lo tanto, se decide aplicar el ajuste fino.

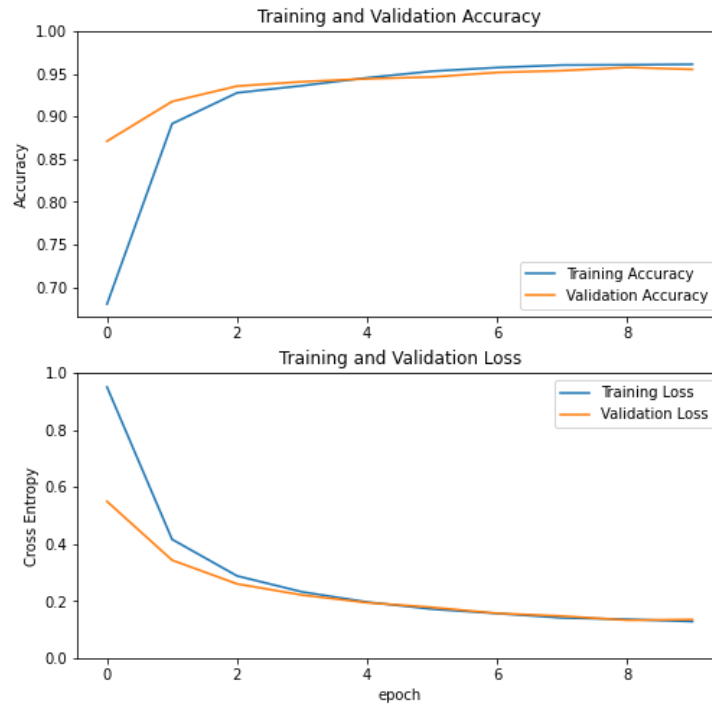


Figura 5.1. Curvas de exactitud y pérdida obtenidas en la transferencia de aprendizaje caso A.

5.2.2 Ajuste fino

El *fine tuning* o ajuste fino es una técnica en donde se descongela una parte de los pesos originales del modelo, esto con la finalidad de actualizar los pesos originales con los nuevos pesos obtenidos durante la etapa anterior, dicho en otras palabras, se hace una actualización de conocimiento.

Dentro de este proceso, es importante resaltar que, se necesita una condición de paro para ello, se empleó la parada temprana (*Early Stopping*), este mecanismo detiene el proceso de entrenamiento cuando una métrica monitoreada haya dejado de presentar mejoras. En este caso la métrica bajo observación es la pérdida (*Loss*) ya que, el objetivo del entrenamiento es minimizar la pérdida (Keras, s.f.).

El efecto positivo del *fine tuning* se puede apreciar en la Figura 5.2 donde, se ilustra que las curvas de aprendizaje muestran una mejora favorable, alcanzando una convergencia de exactitud cercana al 100% y una pérdida aproximada al 0; la línea vertical verde indica el comienzo de la etapa de ajuste fino.

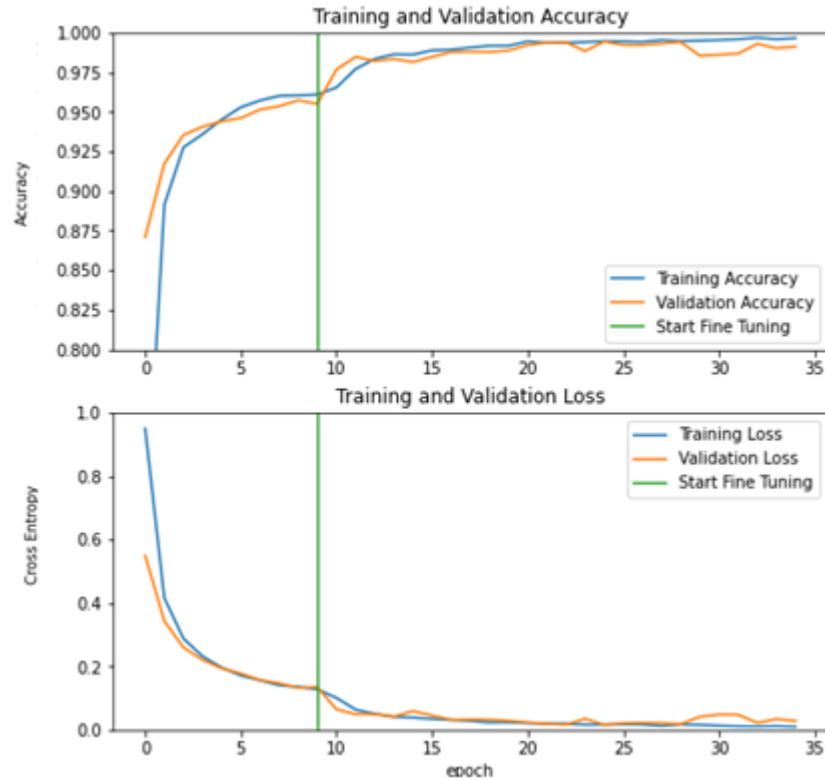


Figura 5.2. Curvas de exactitud y pérdida obtenidas en el ajuste fino caso A.

Posteriormente, se realizó una prueba con el conjunto de *test* al modelo generado, sin aplicación de métricas, sólo se llamó al método *evaluate()* función propia de la red neuronal, el cual arrojó una precisión de 0.9944

Con la finalidad de hacer legible la interpretación de los resultados de las matrices de confusión, la Tabla 5.10 muestra los valores correspondientes a cada categoría (patología).

Tabla 5.10. Clases consideradas en el modelo.

Valor	Clase
0	<i>Common rust (maíz)</i>
1	<i>Corn health</i>
2	<i>Northern leaf blight (maíz)</i>
3	<i>Tomato bacterial spot</i>
4	<i>Tomato health</i>
5	<i>Tomato leaf mold</i>
6	<i>Tomato yellow leaf curl virus</i>

5.2.3 Validación del caso A

En la Figura 5.3 se muestra la matriz de confusión creada a partir del conjunto de validación durante la etapa del entrenamiento.

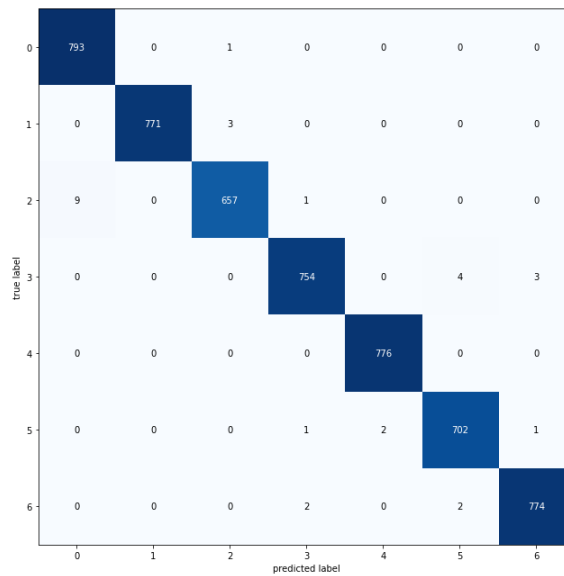


Figura 5.3. Matriz de confusión caso A.

Como puede verse en la matriz de confusión, todas las clases tienen un buen rendimiento. Las métricas que se utilizaron para medir el desempeño del modelo obtenido se ilustran en la Figura 5.4. Como ya se mencionó la métrica *F1-score* muestra el balance entre los resultados obtenidos por las métricas precisión y recall y, como se observa el modelo

En función de los valores obtenidos en la experimentación se observó que el valor mínimo de exactitud obtenido en la validación con el ajuste fino fue de 0.9867 en el *Fold 2*, mientras que el valor máximo de exactitud obtenido en la validación con el ajuste fino fue de 0.9912 en el *Fold 3*; el promedio de exactitud de los tres *Folds* es de 0.9929 , con una desviación estándar de 2.17×10^{-6} .

A continuación en la prueba final, se expone la mejor experimentación, tomando como referencia los datos estadísticos del *Fold 3*.

5.2.5 Caso A: Prueba final

El objetivo de esta evaluación fue para realizar una prueba cercana a la realidad utilizando fotografías que no pertenecen a los conjuntos de entrenamiento y validación; es decir, que tienen características diferentes a las presentes en el entrenamiento. De esta manera, se garantiza que los resultados que obtienen las métricas de evaluación son valores de confianza. Para poder llevar a cabo esta etapa, se buscaron imágenes de otros *datasets* públicos, como se pueden ver sección 5.1.4.

En la Figura 5.5 se ilustra la matriz de confusión con la finalidad de conocer la precisión del modelo generado.

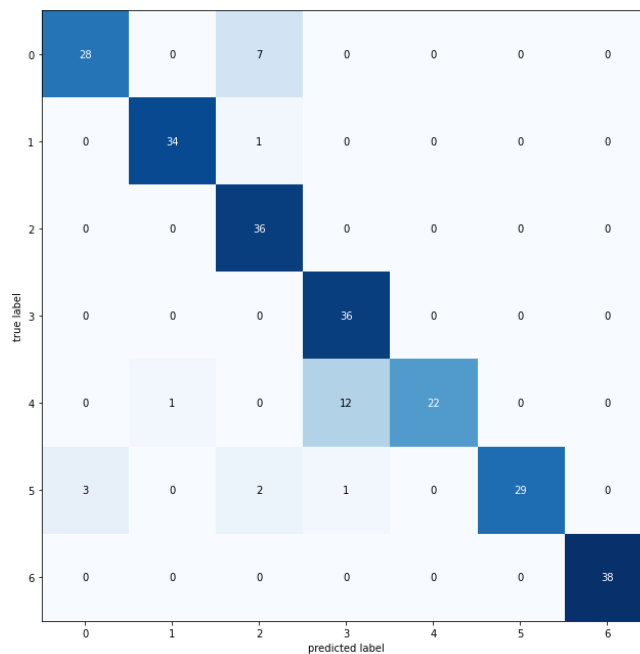


Figura 5.5. Matriz de confusión caso A prueba final.

Las métricas que se utilizaron para medir el desempeño del modelo se ilustran en la Figura 5.6. Como se observa, el modelo alcanzó una precisión del 0.8920 . El sistema tiene un buen desempeño con la mayoría de las imágenes, aun cuando presentan condiciones completamente diferentes a los datos existentes en el entrenamiento.

En general, el modelo tiene un bajo desempeño con 7 clases. Sin embargo, un análisis más detallado en la clase 4 se puede notar que el *recall* es bajo, obteniendo un 0.6286 , siendo la clase con mayor problema. También se observa que la *clase 0* se presenta un *recall* bajo,

esto quiere decir que el modelo es poco sensible ya que presenta dificultades de predecir casos positivos, en este caso, en detectar la patología (7 fallos). En cuanto a la precisión de la *clase 0* el porcentaje indica el número de predicciones correctas, en este caso el valor obtenido es alto, lo que significa que la mayoría de las predicciones pertenecen correctamente a la clase (28 aciertos). Este tipo de evaluaciones son difíciles de encontrar reportadas en el estado del arte.

	precision	recall	f1-score	support
0	0.9032	0.8000	0.8485	35
1	0.9714	0.9714	0.9714	35
2	0.7826	1.0000	0.8780	36
3	0.7347	1.0000	0.8471	36
4	1.0000	0.6286	0.7719	35
5	1.0000	0.8286	0.9062	35
6	1.0000	1.0000	1.0000	38
accuracy			0.8920	250
macro avg	0.9131	0.8898	0.8898	250
weighted avg	0.9129	0.8920	0.8901	250

Figura 5.6. Métricas de evaluación II caso A.

5.2.6. Validación del caso B

En la Figura 5.7 se muestra la matriz de confusión obtenida con el conjunto de validación.

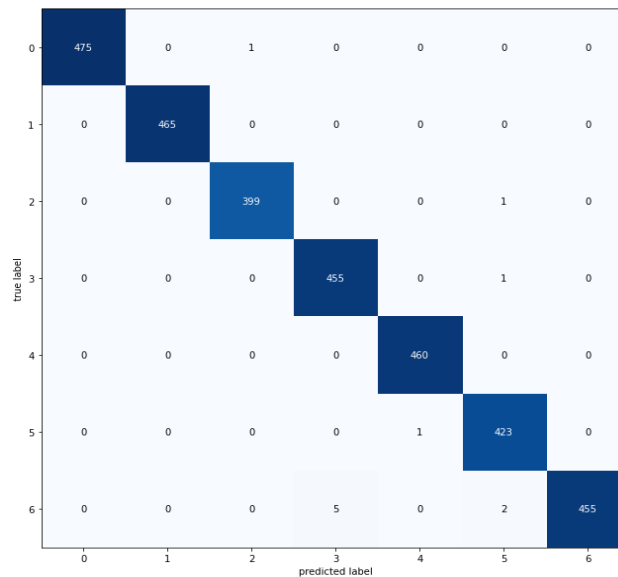


Figura 5.7. Matriz de confusión caso B.

Las métricas que se utilizaron para medir el desempeño del modelo obtenido se ilustran en la Figura 5.8. Como se observa el modelo alcanzó una precisión promedio del 0.9965%; hubo una ligera mejora de 0.0020 de exactitud comparado con el caso A, considerando que se trata de imágenes diferentes ya que se añadió una nueva transformación.

	precision	recall	f1-score	support
0	1.0000	0.9979	0.9989	476
1	1.0000	1.0000	1.0000	465
2	0.9975	0.9975	0.9975	400
3	0.9891	0.9978	0.9934	456
4	0.9978	1.0000	0.9989	460
5	0.9906	0.9976	0.9941	424
6	1.0000	0.9848	0.9924	462
accuracy			0.9965	3143
macro avg	0.9964	0.9965	0.9965	3143
weighted avg	0.9965	0.9965	0.9965	3143

Figura 5.8. Métricas de evaluación I caso B.

5.2.7 Análisis de resultados caso B

La configuración de validación cruzada para el caso B considera que, se generan cinco modelos predictivos. En cada uno de los entrenamientos se tomaron en cuenta, que las curvas de aprendizaje fueran adecuadas; es decir, que las curvas de exactitud del entrenamiento y validación mostraran una convergencia suave ascendente y una convergencia suave descendente de pérdida, en ambas curvas, a lo largo del experimento. Los valores que se obtuvieron de las métricas de evaluación también fueron considerados ya que, de acuerdo con el puntaje conseguido, se determina qué tan confiable es el modelo creado.

A continuación, en la Tabla 5.12 se muestran los valores promedio obtenidos de los cinco modelos generados, empleando la técnica de validación cruzada. En negritas se encuentra el modelo matemático más confiable en relación con las estadísticas calculadas.

Tabla 5.12. Resultados de la validación cruzada caso B

		Transferencia de aprendizaje		Ajuste fino		Exactitud promedio	Pérdida promedio
		Entrenamiento	Validación	Entrenamiento	Validación		
Fold 1	Exactitud	0.9564	0.9526	0.9968	0.9946	0.9946	0.0195
	Pérdida	0.1484	0.1653	0.0081	0.0225		
Fold 2	Exactitud	0.9558	0.9548	0.9966	0.9940	0.9961	0.0161
	Pérdida	0.1483	0.1692	0.0107	0.0253		
Fold 3	Exactitud	0.9563	0.9465	0.9962	0.9914	0.9927	0.0207
	Pérdida	0.1521	0.1680	0.0129	0.0319		
Fold 4	Exactitud	0.9563	0.9593	0.9967	0.9927	0.9968	0.0090
	Pérdida	0.1506	0.1391	0.0093	0.0225		
Fold 5	Exactitud	0.9541	0.9551	0.9974	0.9936	0.9965	0.0083
	Pérdida	0.1529	0.1445	0.0062	0.0201		

En relación con los valores obtenidos en la experimentación se observó que el valor mínimo de exactitud fue de 0.9927 en el *Fold 3*, mientras que el valor máximo de exactitud fue el obtenido por el *Fold 4 con un 0.9968*; el promedio de exactitud de los cinco *Folds* es de 0.99534 con una desviación estándar de 0.0009.

5.2.8. Caso B: Prueba final

En la Figura 5.9 se ilustra la matriz de confusión obtenida a partir de las imágenes ajenas al proceso de la experimentación. Este ejercicio se hizo con la intención de conocer el porcentaje de mejora del modelo seleccionado, en función de las configuraciones que se hicieron para el caso B.

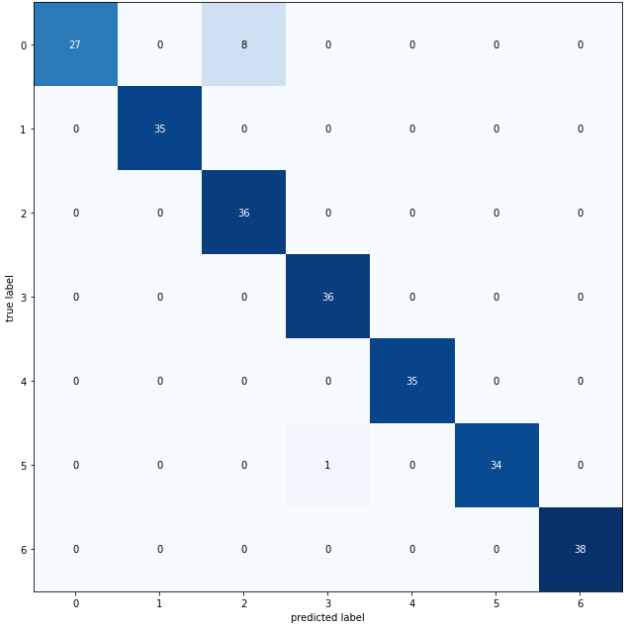


Figura 5.9. Matriz de confusión prueba final.

Las métricas que se emplearon para cuantificar el rendimiento del sistema se ilustran en la Figura 5.10. Como se observa el modelo alcanzó una precisión del 0.9640 lo cual, representa una mejora en comparación del caso A en donde, se reporta una exactitud del 0.8920, el modelo logró mejorar la clasificación de imágenes en la mayoría de las clases. Sin embargo, se observa que la clase 0 continúa teniendo traslape con la clase 2.

	precision	recall	f1-score	support
0	1.0000	0.7714	0.8710	35
1	1.0000	1.0000	1.0000	35
2	0.8182	1.0000	0.9000	36
3	0.9730	1.0000	0.9863	36
4	1.0000	1.0000	1.0000	35
5	1.0000	0.9714	0.9855	35
6	1.0000	1.0000	1.0000	38
accuracy			0.9640	250
macro avg	0.9702	0.9633	0.9633	250
weighted avg	0.9699	0.9640	0.9635	250

Figura 5.10. Métricas de evaluación II caso B.

5.3 Comparación de resultados

Concluidas las experimentaciones de los casos A y B, se tomó como mejor experimentación el caso B, seleccionando el modelo número cinco (*fold 5*) como el mejor, en concordancia con el análisis presentado anteriormente.

Dicho modelo fue seleccionado porque, en el caso B, la configuración de las particiones de entrenamiento y validación fue de 80-20 es decir, 80% para el entrenamiento y 20% para validación; en este caso los cinco modelos generados aprovecharon una mayor cantidad de imágenes, lo cual significa que cada modelo del caso B extrajo nuevas características que no estuvieron presentes en los modelos del caso A. Además, el añadir cambios en la iluminación (20% atenuaciones y 20% iluminación) contribuye a generar aún más nuevas propiedades, por lo tanto, los mapas de características de los modelos del caso B son superiores al caso A.

A continuación, en la Tabla 5.13 se comparan los resultados de exactitud obtenidos en este trabajo con respecto a los resultados que reportan algunos artículos que se estudiaron durante el desarrollo de esta tesis. Los resultados presentados en dicha tabla, corresponden al modelo número cinco del caso B, tomando de referencia los resultados obtenidos durante la experimentación (ver Tabla 5.12). Como se puede observar, los resultados logrados son comparables a los reportados por otros trabajos.

Tabla 5.13. Comparación de exactitud con diferentes trabajos del estado del arte.

Referencia	Exactitud	Red neuronal	Núm. Clases
Ocampo y Dadios, (2018)	89%	<i>MobileNet</i>	Cinco
Laha Ale <i>et al.</i> , (2019)	89.7%	<i>DenseNet</i>	N/D
Sagar, Dheeba, (2020)	98.2%	<i>ResNet50</i>	9
Aravind Krishnaswamy <i>et al.</i> , (2018)	97.47%	<i>AlexNet</i>	6
Alexandre Pereira <i>et al.</i> , (2019)	95%	<i>CNN</i>	1
Serawork Walleign <i>et al.</i> , (2018)	99.32%	<i>LeNet</i>	4
Andre da Silva Abade <i>et al.</i> , (2019)	99.59%	<i>AlexNet</i>	38
Dikdik Krisnandi <i>et al.</i> , (2019)	98.09%	<i>Inception-ResNet-v2</i>	2
Propuesto	99.65%	<i>MobileNetV2</i>	7

5.4 Aplicaciones

Una vez que se tiene un modelo predictivo confiable conforme a la información de las métricas de evaluación, es posible desarrollar aplicaciones en diferentes entornos de ejecución. En este trabajo de tesis se presentan dos ejemplos como evidencias de las utilidades y alcances que tiene la IA hoy en día.

Para cada caso se recurrieron a las librerías correspondientes que *TensorFlow* aloja y se utilizaron para realizar transformación al modelo predictivo que se generó una vez que finalizó el proceso de experimentación ya que, los formatos son distintos y cada entorno de trabajo requiere su respectivo formato.

5.4.1 Versión móvil

Al utilizar la librería *TensorFlowLite* (TensorFlow, s.f) se genera un modelo predictivo ligero capaz de ser alojado y ejecutado en dispositivos móviles en este caso, se hizo una aplicación para teléfonos con sistema operativo *Android*.

Las librerías empleadas para poder efectuar este cambio de formato fueron:

- *TensorFlow-1.15.2*
- *Keras-2.3.1*

Cabe recalcar que, *GoogleColab* gestiona las versiones disponibles de las librerías y lenguajes de programación en el sitio para poder trabajar.

Dentro de TensorFlow existe un convertidor (*TensorFlow Lite*) el cual, toma un modelo de TensorFlow (*modelo.h5*) y genera un modelo de TensorFlow Lite (*modelo.tflite*); el tamaño del modelo que se obtuvo a raíz de esta transformación fue de: 8.49 Mbytes.

Eventualmente, se desarrolló la aplicación móvil y se importó dicho modelo desde *AndroidStudio*; el tamaño de la aplicación ya instalada en el dispositivo móvil es de: 25.07 Mbytes; el teléfono donde se realizaron las pruebas fue un *Motorola e5 Play* con sistema operativo *Android Oreo 8.1.0*, memoria RAM de 2Gbytes, memoria interna de 16Gbytes, expandible a 32 Gbytes y una cámara de 8 Megapíxeles.

Durante las pruebas en el dispositivo móvil se observó un buen desempeño; la mayor resolución que se puso a prueba fue la de la misma cámara del *smartphone*, el tiempo de lectura de la imagen desde el directorio interno del teléfono hacia la aplicación demoró 4.7 segundos y el tiempo de predicción de la misma fue de 2.6 segundos.

En la Figura 5.11 se muestra la interfaz gráfica de la aplicación móvil realizando una predicción de una enfermedad presente en la hoja del tomate.



Figura 5.11. Predicción en la aplicación móvil.

Como primer paso, se accede a los directorios para buscar la imagen de interés, se selecciona y carga en la aplicación, finalmente se da clic en el botón “predecir”, se debe esperar unos pocos segundos para obtener la predicción que el modelo imprime en pantalla.

5.4.2 Versión web

TensorFlow posee una librería llamada *TensorFlowJs* que facilita utilizar el lenguaje *java script* y poder crear modelos predictivos que puedan ser ejecutados en una aplicación web, sin necesidad de utilizar lenguaje *Python* (TensorFlow, s.f).

En la Figura 5.12 se ilustra la interfaz gráfica de la aplicación web haciendo una predicción de una enfermedad presente en la hoja del maíz.

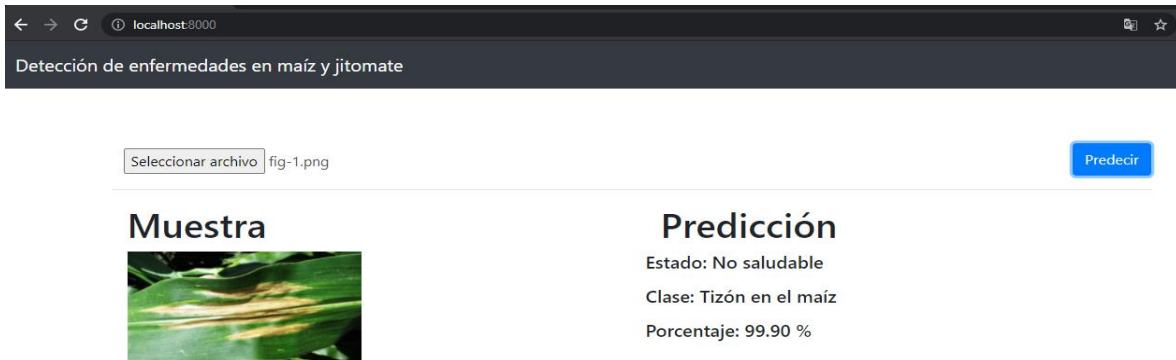


Figura 5.12. Predicción en la versión WEB.

Al igual que en la aplicación móvil, se procede acceder a los directorios para buscar la imagen a analizar, cargar en la aplicación web y finalmente dar clic en el botón “predecir”, esperar unos pocos segundos para obtener la predicción que el modelo imprime en pantalla.

5.4.3 Recomendaciones de uso

A continuación, se enlistan algunas recomendaciones para obtener predicciones más precisas al momento de querer realizar un diagnóstico.

- La imagen que se desea diagnosticar debe de ser a color (formato RGB), se recomienda que el formato de entrada de imagen sea en formato *.png* (*portable network graphic*) pero, se puede utilizar también los formatos: *jpg* y *jpeg* siempre y cuando sean de calidad es decir, las fotografías no deben estar difuminadas.
- En el caso de las resoluciones para la versión móvil, se pide que no supere los cinco megapíxeles puesto que, dependiendo de los recursos disponibles (*hardware*) del dispositivo, este puede demorar en el tiempo de carga y predicción.
- Evitar las fotografías que presenten un exceso o carencia de iluminación o sea, las capturas deben de realizarse en ambientes donde la hoja sea visible.
- Evitar capturas en donde la hoja aparezca de manera parcial o casi nula. La hoja debe estar centrada en la imagen.
- Se recomienda que en las muestras no figuren: manos, dedos, o algún accesorio sobre la zona a analizar.
- Se aconseja colocar la hoja sobre una superficie de color blanco, gris, negro o algún color que haga contraste con respecto al color de la hoja.
- Se sugiere que los ejemplares no presenten: gotas de agua, tierra, o alguna partícula no propia de la zona que se desea examinar.
- No utilizar en la etapa de germinación o con plántulas.

CAPÍTULO 6

CONCLUSIONES

En esta sección, se exponen las conclusiones generales del proyecto de tesis, la justificación de los objetivos y alcances logrados. También se anexan las aportaciones y actividades académicas adicionales hechas durante el programa de posgrado.

6.1 Conclusiones generales

Se propuso e implementó la creación de un sistema de clasificación de imágenes, que permite llevar a cabo la clasificación de patologías a partir de fotografías de hojas provenientes de los cultivos del jitomate y maíz, empleando el enfoque de aprendizaje profundo.

La arquitectura que se empleó para generar dicho modelo corresponde a la familia de redes *convolucionales* que se encuentran alojadas en *Keras*. En este caso particular, se optó por la red *MobileNetV2*. Las técnicas a las que se recurrió para crear dicho modelo fueron la transferencia de aprendizaje y el ajuste fino.

Se llevaron a cabo dos experimentaciones, en el primero se obtuvieron tres modelos y en el segundo se generaron cinco modelos predictivos. En el segundo se lograron mejores resultados al añadir una nueva transformación (cambios de iluminación) y se aumentó el número de muestras para llevar a cabo el entrenamiento (de 50% a 80%) lo que contribuyó a extraer mejores características. Los resultados obtenidos son ligeramente superiores a los reportados en el estado del arte revisado.

Debido a los resultados de exactitud que brinda esta arquitectura es posible desarrollar aplicaciones de escritorio (*software*) o aplicaciones móviles, brindándole al usuario final información de interés, lo cual permite al agricultor actuar de manera temprana en dado caso de que, se presente alguna patología pueda aplicar el pesticida y cuidados pertinentes y así evitar una pérdida total de las cosechas.

De igual manera para las personas expertas en el área puede ser empleada como una herramienta de apoyo y así reducir el tiempo de trabajo empleado, comparado con las técnicas manuales.

a) Objetivos específicos

La descripción de los objetivos logrados, se describen en la Tabla 6.1.

Tabla 6.1. Objetivos específicos

Objetivo	Actividad
Estudiar las características visuales de las patologías y desórdenes fisiológicos presentes en las hojas de las plantas.	Durante la elaboración del estado del arte, se tomaron en cuenta artículos que describían las características de algunas enfermedades causados por agentes patógenos, en este caso: bacterias, hongos y virus. También se dio lectura trabajos relacionados con los desórdenes fisiológicos (estrés hídrico, estrés por calor, carencia de nutrientes y daños químicos) con la finalidad, de conocer las principales características visuales.
Estudiar las técnicas de aprendizaje profundo e implementar la mejor para el problema propuesto con la finalidad de poder reconocer si la planta presenta una enfermedad u otra anomalía como estrés fisiológico.	Se optó emplear una <i>CNN</i> ya que, este tipo de redes se especializa en extraer las características de una imagen de manera automática y de acuerdo con lo reportado en el estado del arte, logran buenos resultados en las tareas de reconocimiento.
Evaluar el desempeño del sistema desarrollado de manera cuantitativa a través de las métricas tradicionales en el reconocimiento de patrones.	Una vez que se creó el modelo predictivo, se recurrieron a las métricas de evaluación (precisión, exactitud, recuerdo, y medida <i>F1</i>) con el objetivo de determinar de manera cuantitativa si el producto generado es confiable para, posteriormente, implementarse en el sistema final.

b) Alcances

Los alcances estipulados hechos, se muestran en la Tabla 6.2

Tabla 6.2. Alcances

Alcance	Actividad
El sistema de visión artificial debe ser robusto a cambios en la escala, perspectiva y diferente resolución presentes en las imágenes.	En este caso, se recurrió a una técnica llamada aumento de datos, la cual consiste en aumentar el número de ejemplares en el banco de imágenes, por medio de transformaciones (rotaciones, efectos espejos, acercamientos y cambios en la iluminación). Con la finalidad de evitar problemas en las resoluciones, se lleva a cabo una normalización y un redimensionamiento a la imagen antes de que, sea procesada por la red neuronal.
El sistema debe ser capaz de identificar síntomas de enfermedad en al menos cuatro diferentes tipos de patologías.	Se trabajaron con cinco enfermedades, tres provenientes del cultivo de jitomates y dos relacionados con el maíz.
El sistema debe ser capaz de reconocer si una planta presenta síntomas de enfermedad debido a virus, bacterias u hongos.	Se llevó a cabo la recolección de imágenes de los cultivos afectados por los agentes patógenos: bacterias, hongos y virus. También se sumaron muestras sanas de los cultivos. En total, el modelo predictivo puede clasificar siete clases.
Se requiere <i>hardware</i> de alto poder, puesto que los algoritmos de aprendizaje profundo demandan demasiado costo computacional para el entrenamiento.	Se adquirió la versión <i>Pro</i> de <i>GoogleColab</i> ya que, en este plan de paga, se pudo acceder el hardware de alto poder y poder utilizar la máquina virtual por más tiempo porque, el proceso de transferencia de aprendizaje demandó demasiado tiempo.

c) Aportaciones

Las contribuciones que se lograron al llevar a cabo esta tesis fueron las siguientes:

- Se entrega dos bancos de imágenes segmentadas de cultivos de maíz y jitomate; cada *dataset* corresponde a las configuraciones de la validación cruzada ($K=3$ y $K=5$); el total de imágenes es de 15,711 categorizadas en siete clases.
- Tres modelos predictivos de cada entorno de ejecución (*python*, *web* y *móvil*).
- Dos sistemas de clasificación de imágenes en sus versiones: *móvil* y *web*.
- Scripts de códigos fuentes (*python*, *java script* y *kotlin*)
- Documentación.

d) Actividades académicas adicionales

A continuación, se anexan dos participaciones que se realizaron durante la estancia en el posgrado; de esta manera, se cumple con el requisito que el programa “maestría en ciencias de la computación” estipula para obtener el grado profesional.

Anexo A: Participación en el evento SAIBOOT 2020 celebrado del 12 al 17 de octubre



Artículo redactado

Aprendizaje profundo aplicado en el sector agrícola

Andros Meraz Hernández, Andrea Magadán Salazar,
Gerardo Vela Valdés
*Tecnológico Nacional de México/Centro Nacional de Investigación y
Desarrollo Tecnológico (CENIDET)
Cuernavaca, Morelos, México.
{m20ce040, andrea.ms, luis.vv}@cenidet.tecnm.mx*

Jorge Alberto Fuentes Pacheco
*Laboratorio de Visión por computadora
Centro de Investigación en Ciencias
Universidad Autónoma del Estado de Morelos
Cuernavaca, Morelos, México.
jorge.fuentes@ueam.mx*

Anexo B: Participación en el evento SAIBOOT 2021 celebrado del 21 al 25 de junio



Artículo redactado

Detección de patologías en cultivos empleando aprendizaje profundo

Andros Meraz Hernández, Andrea Magadán Salazar, Raúl Pinto
Elías, Nimrod González Franco
Tecnológico Nacional de México/Centro Nacional de Investigación y
Desarrollo Tecnológico (CENIDET)
Cuernavaca, Morelos, México.
{m20ce040, andrea.ms, raul.pe, nimrod.gf}@cenidet.tecnm.mx

Jorge Alberto Fuentes Pacheco
Laboratorio de Visión por computadora
Centro de Investigación en Ciencias
Universidad Autónoma del Estado de Morelos
Cuernavaca, Morelos, México.
jorge.fuentes@ueam.mx

Anexo C: Participación en el evento 3 minutos sobre mi tesis, octubre 2021



Referencias

A.A. Pedroza Lagunas. 2017. Sistema de visión artificial para determinar enfermedades provocadas por bacterias en plantas de jitomates. Cuernavaca: CENIDET.

Abhinav Sagar, Dheebea J. 2020. On Using Transfer Learning For Plant Disease Detection. Vellore Institute of Technology. Vellore Tamil Nadu. India. DOI: <https://doi.org/10.1101/2020.05.22.110957>

Android Developers. (s.f.). Recuperado 8 de febrero de 2022, de <https://developer.android.com/kotlin?hl=es-419>

A. Hidayatuloh, M. Nursalman and E. Nugraha, "Identification of Tomato Plant Diseases by Leaf Image Using Squeezenet Model," 2018 International Conference on Information Technology Systems and Innovation (ICITSI), 2018, pp. 199-204, doi: 10.1109/ICITSI.2018.8696087.

Alexandre Pereira Marcos, Natan Luis Silva Rodovalho, André R. Backes. Coffee Leaf Rust Detection Using Convolutional Neural Network. IEEE XV Workshop de Visão Computacional (WVC). São Bernardo do Campo, Brazil. DOI: <https://doi.org/10.1109/WVC.2019.8876931>

Alvaro Fuentes, Sook Yoon, Sang Cheol Kim, Dong Sun Park, 2017, A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition, Department of Electronics Engineering, Chonbuk National University, Jeonbuk 54896, Korea, <https://doi.org/10.3390/s17092022>

Andre da Silva Abade, Ana Paula G. S. de Almeida, Flavio de Barros Vidal. 2019. Plant Diseases Recognition from Digital Images using Multichannel Convolutional Neural Networks. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, ISBN 978-989-758-354-4; ISSN 2184-4321, pages 450-458. DOI: 10.5220/0007383904500458

Anton Louise P. de Ocampo and Elmer P. Dadios. 2018. Mobile Platform Implementation of Lightweight Neural Network Model for Plant Disease Detection and Recognition. IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM). DOI: <https://doi.org/10.1109/HNICEM.2018.8666365>.

Aravind Krishnaswamy Rangarajan, Raja Purushothaman, Anirudh Ramesh. 2018. Tomato crop disease classification using pre-trained deep learning algorithm. Proceedings of International Conference on Robotics and Smart Manufacturing (RoSMa). DOI: <https://doi.org/10.1016/j.procs.2018.07.070>

Aws. (s.f.). Recuperado 8 de febrero de 2022, de <https://aws.amazon.com/es/machine-learning/what-is-ai/>

Balzhoyt Roldán Ortega, Rajesh Roshan Biswal, Eddy Sánchez Delacruz, Detección de enfermedades en el sector agrícola utilizando Inteligencia Artificial, 2019, Instituto Tecnológico Superior de Misantla, Departamento de Posgrado, Misantla, Veracruz, México, https://rcs.cic.ipn.mx/2019_148_7/Deteccion%20de%20enfermedades%20en%20el%20sector%20agricola%20utilizando%20Inteligencia%20Artificial.pdf

Belal A. M. Ashqar, Samy S. Abu-Naser, 2018, Image-Based Tomato Leaves Diseases Detection Using Deep Learning, Department Information Technology, Faculty of Engineering and Information Technology, Al-Azhar University, Gaza, Palestine, International Journal of Academic Engineering Research (IJAER), <http://dstore.alazhar.edu.ps/xmlui/handle/123456789/278>

B, Santhikiran and Badam, Raghuram Reddy and Chirunjeevi, M and Kumar, D Manish, Design and Implementation of Unhealthy Leaves Disease Detection Using Image Processing (January 19, 2020). International Conference of Advance Research & Innovation (ICARI) 2020, Available at SSRN: <https://ssrn.com/abstract=3643583> or <http://dx.doi.org/10.2139/ssrn.3643583>

Bdigital.zamorano.edu. (s.f.). Recuperado 8 de febrero de 2022, de <https://bdigital.zamorano.edu/bitstream/11036/1354/2/02.pdf>

CANNA Research. (s.f.). Los desórdenes fisiológicos en las plantas. Recuperado 8 de febrero de 2022, de http://www.canna.es/los_desordenes_fisiologicos_en_las_plantas

CALS. (s.f.). Recuperado 8 de febrero de 2022, de <https://fieldcrops.cals.cornell.edu/corn/diseases-corn/northern-corn-leaf-blight/>

Delegación SADER Morelos. (7 abril del 2020). Recuperado 8 de febrero de 2022, de <https://www.gob.mx/agricultura/morelos/articulos/produccion-de-maiz-blanco-en-el-estado-de-morelos?idiom=es>

Desarrolloweb.com. (s.f.). Recuperado 8 de febrero de 2022 de <https://desarrolloweb.com/home/javascript>

Dikdik Krisnandi, Hilman F. Pardede, R. Sandra Yuwana. 2019. Diseases Classification for Tea Plant Using Concatenated Convolution Neural Network. Research Center for Informatics (P2I) - Indonesia Institute of Sciences (LIPI) Bandung 40135, Indonesia. DOI: <https://doi.org/10.21512/commit.v13i2.5886>

Emebo, Onyeka & Fori, Barka & Geteloma, Victor & Abayomi-Zannu, Temidayo. (2019). Development of Tomato Septoria Leaf Spot and Tomato Mosaic Diseases Detection Device Using Raspberry Pi and Deep Convolutional Neural Networks. Journal of Physics: Conference Series. 1299. 012118. 10.1088/1742-6596/1299/1/012118.

E. López Martínez. 2018. Algoritmos para la detección y cuantificación de defectos en manzanas por inspección visual. Cuernavaca: CENIDET.

El Sol de Cuernavaca. (3 febrero del 2019). Recuperado 8 de febrero de 2022, de <https://www.elsoldecuernavaca.com.mx/local/aumenta-produccion-de-los-jitomates-3010787.html>

François Chollet, Deep Learning with Python, Shelter Island, NY 11964, Manning Publications, 2018, <https://tanthiamhuat.files.wordpress.com/2018/03/deeplearningwithpython.pdf>

F. J. P. Montalbo and A. A. Hernandez, "An Optimized Classification Model for Coffea Liberica Disease using Deep Convolutional Neural Networks," 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), 2020, pp. 213-218, doi: 10.1109/CSPA48992.2020.9068683.

F. Jakjoud, A. Hatim, and A. Bouaaddi. 2019. Deep Learning application for plant diseases detection. In Proceedings of the 4th International Conference on Big Data and Internet of Things (BDIoT'19). Association for Computing Machinery, New York, NY, USA, Article 45, 1–6. DOI:<https://doi.org/10.1145/3372938.3372983>

Geetharamani G., Arun Pandian J., Identification of plant leaf diseases using a nine-layer deep convolutional neural network, Computers & Electrical Engineering, Volume 76, 2019, Pages 323-338, ISSN 0045-7906, <https://doi.org/10.1016/j.compeleceng.2019.04.011>.

GoogleColab. (s.f.). Recuperado 8 de febrero de 2022, de <https://colab.research.google.com/notebooks/intro.ipynb#recent=true>

Hanson, A.J., Joy, A., & Francis, J.N. (2017). Plant Leaf Disease Detection using Deep Learning and Convolutional Neural Network, <https://www.semanticscholar.org/paper/Plant-Leaf-Disease-Detection-using-Deep-Learning-Hanson-Joy/8213f2f1d34e5da9f96089aaf6e792eeb3a99c82>

H. Durmuş, E. O. Güneş and M. Kırıcı, "Disease detection on the leaves of the tomato plants by using deep learning," 2017 6th International Conference on Agro-Geoinformatics, 2017, pp. 1-5, doi: 10.1109/Agro-Geoinformatics.2017.8047016.

ImageNet. (s.f.). Recuperado 8 de febrero de 2022, de <http://image.net.org/challenges/LSVRC/2012/>

I. Z. Mukti and D. Biswas, "Transfer Learning Based Plant Diseases Detection Using ResNet50," 2019 4th International Conference on Electrical Information and Communication Technology (EICT), 2019, pp. 1-6, doi: 10.1109/EICT48899.2019.9068805.

José G.M. Esgario, Renato A. Krohling, José A. Ventura, Deep learning for classification and severity estimation of coffee leaf biotic stress, Computers and Electronics in Agriculture, Volume 169, 2020, 105162, ISSN 0168-1699, <https://doi.org/10.1016/j.compag.2019.105162>.

Junde Chena, Jinxiu Chena , Defu Zhang. 2020. Using deep transfer learning for image-based plant disease identification. School of Informatics, Xiamen University, Xiamen 361005. China. DOI: <https://doi.org/10.1016/j.compag.2020.105393>

Keras. (s.f.). Recuperado 8 de febrero de 2022, de <https://keras.io/about/>

Laha Ale, Alaa Sheta, Longzhuang, Ye Wang, Ning Zhang. 2019. Deep Learning based Plant Disease Detection for Smart Agriculture. IEEE Globecom Workshops (GC Wkshps). DOI: <https://doi.org/10.1109/GCWkshps45667.2019.9024439>.

Lili Ayu Wulandhari, Arie Qurania, Prihastuti Harsani , 2019, PLANT NUTRIENT DEFICIENCY DETECTION USING DEEP CONVOLUTIONAL NEURAL NETWORK, Jl. K. H. Syahdan No. 9, Kemanggisan, Palmerah, Jakarta 11480, Indonesia, <http://www.icicel.org/ell/contents/2019/10/el-13-10-13.pdf>

Lin Ke, Gong Liang, Huang Yixiang, Liu Chengliang, Pan Junsong, Deep Learning-Based Segmentation and Quantification of Cucumber Powdery Mildew Using Convolutional Neural Network, Frontiers in Plant Science , Vol. 10, 2019, Pages 155, URL: <https://www.frontiersin.org/article/10.3389/fpls.2019.00155>, DOI: 10.3389/fpls.2019.00155

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. 2019 "MobileNetV2: Inverted Residuals and Linear Bottlenecks", Google Inc. https://openaccess.thecvf.com/content_cvpr_2018/papers/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf

M. Sardogan, A. Tuncer and Y. Ozen, "Plant Leaf Disease Detection and Classification Based on CNN with LVQ Algorithm," 2018 3rd International Conference on Computer Science and Engineering (UBMK), 2018, pp. 382-385, doi: 10.1109/UBMK.2018.8566635.

M. S. Arya, K. Anjali and D. Unni, "Detection of unhealthy plant leaves using image processing and genetic algorithm with Arduino," 2018 International Conference on Power, Signals, Control and Computation (EPSCICON), 2018, pp. 1-5, doi: 10.1109/EPSCICON.2018.8379584.

MSU. (s.f.). Recuperado 8 de febrero de 2022, de <http://extension.msstate.edu/publications/common-diseases-tomatoes>

Nvidia Developer. (s,f). Recuperado 8 de febrero de 2022, de <https://developer.nvidia.com/discover/artificial-neural-network>

Parul Sharma, Yash Paul Singh Berwal, Wiqas Ghai, Performance analysis of deep learning CNN models for disease detection in plants using image segmentation, Information Processing in Agriculture, Volume 7, Issue 4, 2020, Pages 566-574, ISSN 2214-3173, <https://doi.org/10.1016/j.inpa.2019.11.001>

Plant Village. (s.f.). Recuperado 8 de febrero de 2022, de <https://plantvillage.psu.edu/>

Prakruti Bhatt, Sanat Sarangi, Anshul Shivhare, Dineshkumar Singh and Srinivasu Pappula, Identification of Diseases in Corn Leaves using Convolutional Neural Networks and Boosting, 2019, TCS Research and Innovation, Mumbai, India, <https://pdfs.semanticscholar.org/5c38/771389d4d7dd171d65ebc19a336d33571f41.pdf>

P. K. Kosamkar, V. Y. Kulkarni, K. Mantri, S. Rudrawar, S. Salmpuria and N. Gadekar, "Leaf Disease Detection and Recommendation of Pesticides Using Convolution Neural Network," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-4, doi: 10.1109/ICCUBEA.2018.8697504.

Priyanka G. Shinde, Ajay K. Shinde, Ajinkya A. Shinde, Borate S. P, 2017, Plant Disease Detection Using Raspberry PI By K-means Clustering Algorithm , Dept (E&TC) SVPM's College Of Engineering Malegaon(Bk),Baramati, http://www.irdindia.in/journal_ijeecs/pdf/vol5_iss1/24.pdf

Python. (s.f.). Recuperado 8 de febrero de 2022, de <https://www.python.org/about/>

RemoveBackGround. (s. f.). Recuperado 8 de febrero de 2022, de <https://www.remove.bg>

S. V. Militante, B. D. Gerardo and R. P. Medina, "Sugarcane Disease Recognition using Deep Learning," 2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), 2019, pp. 575-578, doi: 10.1109/ECICE47484.2019.8942690.

S. S. Kumar and B. K. Raghavendra, "Diseases Detection of Various Plant Leaf Using Image Processing Techniques: A Review," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), 2019, pp. 313-316, doi: 10.1109/ICACCS.2019.8728325.

Serawork Wallelign, Mihai Polceanu, Cedric Buche. 2018. Soybean Plant Disease Identification Using Convolutional Neural Network. The Thirty-First International Florida Artificial Intelligence Research Society Conference (FLAIRS-31). DOI: <https://hal.archives-ouvertes.fr/hal-01807760>

Sitio big data. (s.f.). Recuperado 8 de febrero de 2022, de <https://sitiobigdata.com/2019/01/19/machine-learning-metrica-clasificacion-parte-3/>

Solemane Coulibaly, Bernard Kamsu-Foguem, Dantouma Kamissoko, Daouda Traore, Deep neural networks with transfer learning in millet crop images, Computers in Industry, Volume 108, 2019, Pages 115-120, ISSN 0166-3615, <https://doi.org/10.1016/j.compind.2019.02.003>.

Solución Ingenieril. (s.f.). Recuperado 8 de febrero de 2022, de http://solucioningenieril.com/vision_artificial/etapas_de_un_sistema_de_vision

Sumita Mishra, Rishabh Sachan, Diksha Rajpal, Deep Convolutional Neural Network based Detection System for Real-time Corn Plant Disease Recognition, Procedia Computer Science, Volume 167, 2020, Pages 2003-2010, ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2020.03.236>.

S. Arya and R. Singh, "A Comparative Study of CNN and AlexNet for Detection of Disease in Potato and Mango leaf," 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), 2019, pp. 1-6, doi: 10.1109/ICICT46931.2019.8977648.

Shanwen Zhang, Haoxiang Wang, Wenzhun Huang, Zhuhong You, Plant diseased leaf segmentation and recognition by fusion of superpixel, K-means and PHOG, Optik, Volume 157, 2018, Pages 866-872, ISSN 0030-4026, <https://doi.org/10.1016/j.ijleo.2017.11.190>.

Smaranjit Ghose. (2020 11 de noviembre). Banco de imágenes de maíz. Recuperado 8 de febrero de 2022, de <https://www.kaggle.com/qramkrishna/corn-leaf-infection-dataset>

T. Kodama and Y. Hata, "Development of Classification System of Rice Disease Using Artificial Intelligence," 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2018, pp. 3699-3702, doi: 10.1109/SMC.2018.00626.

TensorFlow. (s.f.). Recuperado 8 de febrero de 2022, de <https://www.tensorflow.org/>

Portal Morelos. (2 febrero del 2019). Recuperado 8 de febrero de 2022, de <https://morelos.gob.mx/?q=prensa/nota/visita-el-titular-de-sedagro-invernaderos-de-jitomate>

Prajwala TM. (2018 13 de marzo.). Banco de imágenes de jitomates. GitHub. Recuperado 8 de febrero de 2022, de <https://github.com/PrajwalaTM/tomato-leaf-disease-detection/blob/master/README.md>

Vijai Singh, A.K. Misra, Detection of plant leaf diseases using image segmentation and soft computing techniques, Information Processing in Agriculture, Volume 4, Issue 1, 2017, Pages 41-49, ISSN 2214-3173, <https://doi.org/10.1016/j.inpa.2016.10.005>.

Wang J., Chen L., Zhang J., Yuan Y., Li M., Zeng W. (2018) CNN Transfer Learning for Automatic Image-Based Classification of Crop Disease. In: Wang Y., Jiang Z., Peng Y. (eds) Image and Graphics Technologies and Applications. IGTA 2018. Communications in Computer and Information Science, vol 875. Springer, Singapore. https://doi.org/10.1007/978-981-13-1702-6_32