**Desarrollo de un Modelo de Clasificación de Enfermedades del Tomate Utilizando Swin Transformer**

Oscar David Valencia 20192182074

Maria José Capera Firigua 20201188262

Facultad de Ingeniería, Universidad Surcolombiana

Electiva Ciencias de la computación II - Visión Artificial

Dr. Ferley Medina Rojas

16 de Noviembre de 2023

**TABLA DE CONTENIDOS**

[**1. INTRODUCCIÓN 4**](#_heading=)

[**2. PROBLEMA 5**](#_heading=h.30j0zll)

[**3. OBJETIVOS 8**](#_heading=)

[3.1 Objetivo general 8](#_heading=h.1fob9te)

[3.2 Objetivos específicos 8](#_heading=h.3znysh7)

[**4. ESTADO DEL ARTE 10**](#_heading=h.2et92p0)

[4.1 Machine Learning Approach towards Tomato Leaf Disease Classification 10](#_heading=h.vgc5l9g32ol2)

[4.2 ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network 10](#_heading=h.3dy6vkm)

[4.3 Tomato plant disease detection using transfer learning with C-GAN synthetic images 11](#_heading=h.efievltcs6b)

[4.4 Early Detection and Classification of Tomato Leaf Disease Using High-Performance Deep Neural Network 12](#_heading=h.eqamthg5nc4b)

[4.5 Tomato leaf disease classification using supervised learning techniques: contrasting analysis 13](#_heading=h.wfqg4rv69rsr)

[4.6 AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf 14](#_heading=h.qgjzpuau8k0t)

[4.7 BotanicX-AI: Identification of Tomato Leaf Diseases Using an Explanation-Driven Deep-Learning Model 15](#_heading=h.1t3h5sf)

[**5. MARCO TEÓRICO 17**](#_heading=)

[5.1 Enfermedades de la planta de tomate 17](#_heading=)

[5.1.1 Definición 17](#_heading=h.1rkp5mnqxr6t)

[5.1.2 Enfermedades comunes en la planta de tomate 17](#_heading=)

[5.1.3 Síntomas 18](#_heading=)

[5.1.4 Métodos tradicionales de detección de enfermedades 19](#_heading=)

[5.1.5 Desafíos en la detección de enfermedades 19](#_heading=)

[5.2 Seguridad alimentaria (food safety) 20](#_heading=)

[5.3 Visión artificial 20](#_heading=)

[5.3.1 Clasificación de Imágenes 21](#_heading=)

[5.3.2 Machine learning y Deep learning en visión artificial 22](#_heading=)

[5.4 Redes Neuronales Convolucionales (CNN) 22](#_heading=)

[5.5 Vision Transformer 23](#_heading=)

[5.6 Swin Transformer 24](#_heading=)

[5.6.1 Arquitectura de Swin Transformer 25](#_heading=)

[5.6.2 Ventanas desplazadas 25](#_heading=)

[5.6.3 Estructura piramidal 26](#_heading=)

[5.6.4 Detalles arquitectónicos de Swin Transformer 27](#_heading=)

[5.7 Dataset 29](#_heading=)

[5.8 Dataset de plantVillage 30](#_heading=)

[5.8.1 Clases asociadas al tomate en el dataset de plantVillage 31](#_heading=)

[5.9 Aumento de datos 31](#_heading=)

[5.10 Métricas de Evaluación 32](#_heading=)

[5.11 Matriz de Confusión 33](#_heading=)

[5.12 Interpretabilidad mediante mapas de atención 34](#_heading=)

[**6. MODELO MATEMÁTICO 34**](#_heading=)

[6.1 Detección y eliminación de imágenes duplicadas del dataset 34](#_heading=)

[6.1.1 Redimensionamiento y Conversión a Escala de Grises 34](#_heading=h.lq8q23x535p8)

[6.1.2 Transformada de Coseno Discreta (DCT) 35](#_heading=h.fxky2z1vspyg)

[6.1.3 Reducción de la DCT 35](#_heading=h.l377y41lrczo)

[6.1.4 Cálculo de la mediana 35](#_heading=h.33dnwleapk92)

[6.1.5 Binarización 36](#_heading=h.8iw3u1gogwz)

[6.1.6 Construcción del Hash 36](#_heading=h.gq8rguo6ltsm)

[6.1.7 Comparación de Hashes 37](#_heading=h.epqdc5mla3xn)

[6.2 Aumento de datos 37](#_heading=)

[6.2.1 Rotaciones 38](#_heading=h.elladdqpjkwu)

[6.2.2 Desplazamientos 38](#_heading=h.e4qiak1znmyn)

[6.2.3 Recorte 39](#_heading=h.6ri1dl188o08)

[6.2.4 Inversión Horizontal 39](#_heading=h.fqujvpycqrhy)

[6.2.5 Inversión Vertical 40](#_heading=h.r2erz77ysbqa)

[6.3 Arquitectura Swin Tranformer 40](#_heading=)

[6.3.1 División de parches 41](#_heading=)

[6.3.2 Capa de embedding lineal (Linear Embedding) 41](#_heading=)

[6.3.3 Bloques de Swin Transformer 43](#_heading=)

[6.3.4 Normalización de Capa (LayerNorm) 43](#_heading=)

[6.3.5 Autoatención Basada en Ventanas (W-MSA) 44](#_heading=)

[6.3.6 Multi-Layer Perceptron (MLP con GELU) 45](#_heading=)

[6.3.7 Autoatención Basada en Ventanas Desplazadas (SW-MSA) 46](#_heading=)

[6.3.8 Conexión Residual 46](#_heading=)

[6.3.9 Ventanas desplazadas en bloques sucesivos 46](#_heading=)

[6.3.10 Patch Merging 47](#_heading=)

[6.4 Métricas de evaluación 48](#_heading=)

[6.5 Matriz de Confusión 49](#_heading=)

[**7. METODOLOGÍA 50**](#_heading=)

[Fase 1. Diseño y construcción del dataset 50](#_heading=h.yh7gkwxaq1we)

[Paso 1. Identificación de Requerimientos de Datos 50](#_heading=h.z4wyspmvqty7)

[Paso 2. Obtención del Dataset 51](#_heading=h.qtowavlm1cw1)

[Paso 3. Limpieza del Dataset 51](#_heading=h.wjfkykic7b53)

[Paso 4. Balanceo del Dataset 51](#_heading=h.gnjv0c2a5d6w)

[Paso 5. Preprocesamiento de datos 51](#_heading=h.fw2mqm9hfgj3)

[Paso 6. Aumento de Datos 51](#_heading=h.tj95psp1mhmh)

[Fase 2: Entrenamiento del modelo 52](#_heading=h.6htm0j39wyuc)

[Paso 1. Definición de requerimientos de Hardware y Software 52](#_heading=h.p8m7kskqn4fa)

[Paso 2. Arquitectura del modelo 52](#_heading=h.n8uwkjma3d2l)

[Paso 3. Configuración de parámetros 52](#_heading=h.dsg9oycxgciu)

[Paso 4. Entrenamiento del modelo 52](#_heading=h.cmidzwif8sf3)

[Fase 3: Evaluación y Análisis del modelo 53](#_heading=h.d3a2gjhqu030)

[Paso 1. Selección de métricas 53](#_heading=h.oad9uq7zg4ky)

[Paso 2. Pruebas de rendimiento 53](#_heading=h.8p6mqis2igb7)

[Paso 3. Interpretación de resultados 53](#_heading=h.2w1k4dsc0yv8)

[Fase 4: Comparativa con otros modelos 53](#_heading=h.obroqt5o0xkv)

[Paso 1. Entrenamiento de modelos de CNN para comparación 53](#_heading=h.2p0pv4ybyahs)

[Paso 2. Evaluación de modelos 54](#_heading=h.o225q12fvobd)

[**REFERENCIAS 54**](#_heading=h.17dp8vu)

# INTRODUCCIÓN

El cultivo del tomate representa un pilar crucial para la economía agrícola global según la FAO. Además de su importancia económica, el tomate es nutricionalmente relevante por sus beneficios a la salud (Collins et al., 2022). No obstante, esta hortaliza es susceptible a diversas enfermedades causadas por diferentes patógenos (Panno et al., 2021). Factores como la limitada diversidad genética, el monocultivo y el cambio climático incrementan esta vulnerabilidad, generando enormes pérdidas económicas (Caruso et al., 2022). Dichas pérdidas tienen un impacto en la economía agrícola, la salud pública y la sostenibilidad ambiental.

La detección temprana y precisa de enfermedades en el tomate es, por lo tanto, una necesidad imperante. Los métodos tradicionales de diagnóstico tienen limitaciones en términos de costo, tiempo y susceptibilidad a errores (Blancard, 2011). Aunque la visión artificial y el Machine Learning han ofrecido alternativas, presentan sus propias restricciones. Las técnicas de Deep Learning como las redes neuronales convolucionales (CNN) han mostrado efectividad pero han llegado a un punto de estancamiento en la innovación arquitectónica y efectividad (Li, 2020).

Este estudio se propone explorar un enfoque innovador para la detección y clasificación de enfermedades del tomate mediante la arquitectura "Swin Transformer". Se abordarán preguntas críticas como la construcción de un dataset balanceado, el entrenamiento del modelo, la evaluación e interpretación de las predicciones y la comparación del modelo con técnicas basadas en CNN. A través de un análisis sistemático y una metodología rigurosa, se busca desarrollar un modelo que supere las limitaciones actuales en la detección y clasificación de enfermedades del tomate.

La estructura del trabajo incluirá secciones sobre el problema, objetivos, estado del arte, marco teórico, modelo matemático, metodología, resultados, conclusiones y trabajo futuro.

# 

# PROBLEMA

El cultivo del tomate es esencial para la alimentación y la economía agrícola global debido a que es una de las hortalizas más producidas mundialmente. De acuerdo con la FAO (2021), la producción mundial superó los 189 millones de toneladas, abarcando más de 5 millones de hectáreas cultivadas. Nutricionalmente es una fuente rica de fitoquímicos, nutrientes, antioxidantes y compuestos fenólicos (Collins et al., 2022). Se ha determinado que su consumo aporta beneficios anticancerígenos y protección contra enfermedades cardiovasculares y neurodegenerativas (Collins et al., 2022).

No obstante, el tomate es vulnerable a una amplia variedad de enfermedades causadas por hongos, bacterias, fitoplasmas, virus y viroides (Panno et al., 2021; ver Tabla 1). Esta vulnerabilidad se ve incrementada por factores como su limitada diversidad genética resultado de la selección intensiva, la tendencia al monocultivo, el intercambio de material infectado a nivel internacional y el cambio climático (Panno et al., 2021). Estas enfermedades representan pérdidas económicas que superan los 30.000 millones de dólares al año (Caruso et al., 2022). Dichas pérdidas repercuten en la economía agrícola, la salud pública y la sostenibilidad ambiental, al afectar el rendimiento y la calidad del cultivo, desestabilizar ecosistemas y elevar los costos de producción por el uso intensivo de pesticidas y otras medidas de control (Panno et al., 2021).

La detección temprana y precisa de enfermedades en el tomate es primordial para mitigar su impacto, Los métodos tradicionales de diagnóstico como la detección manual realizada por técnicos expertos requiere un amplio conocimiento en múltiples disciplinas para un diagnóstico preciso. Este enfoque consume tiempo y recursos humanos y es susceptible a errores humanos. Los métodos de laboratorio, como técnicas inmunológicas, cultivo de patógenos y microscopía, ofrecen precisión pero implican costos elevados y tiempos de procesamiento largos (Blancard, 2011; Valdivieso, 2021).

A pesar de los avances significativos en visión artificial en la última década, se han presentado limitaciones. En el ámbito del Machine Learning, los métodos suelen recurrir a la extracción manual de características y algoritmos heurísticos, lo que puede ser lento, laborioso, costoso y susceptible a errores (Zahangir Alom et al., 2018). Estos enfoques han sido superados por técnicas de Deep Learning como las redes neuronales convolucionales (CNN) (Rawat et al. 2022). Sin embargo, estas últimas parecen haber llegado a un punto de estancamiento, su arquitectura presenta sesgos inherentes que consideran los elementos cercanos más importantes que los lejanos en una imagen lo cual puede llevar a sobreajustes en la red (Szeliski, 2020), tampoco han presentado innovaciones en su arquitectura recientemente lo que conlleva a un estancamiento de la efectividad en la resolución de problemas de clasificación (Li, 2020).

Es por ello, que la presente investigación sugiere un enfoque innovador basado en la arquitectura "Swin Transformer", la cual ha demostrado ser altamente efectiva en tareas de visión artificial pero que nunca ha sido aplicado en la clasificación de enfermedades del tomate.

En este contexto, se plantea las siguientes preguntas:

* ¿Cómo desarrollar un modelo de detección y clasificación de enfermedades del tomate utilizando la arquitectura Swin Transformer que supere la precisión de las redes neuronales convolucionales (CNN)?
* ¿Cómo se puede construir un dataset balanceado en las diferentes clases de las enfermedades del tomate?
* ¿Cómo evaluar e interpretar las predicciones del modelo Swin Transformer para cada clase?
* ¿Cómo se compara el accuracy (exactitud) del modelo Swin Transformer con los modelos basados en redes neuronales convolucionales?

# OBJETIVOS

## Objetivo general

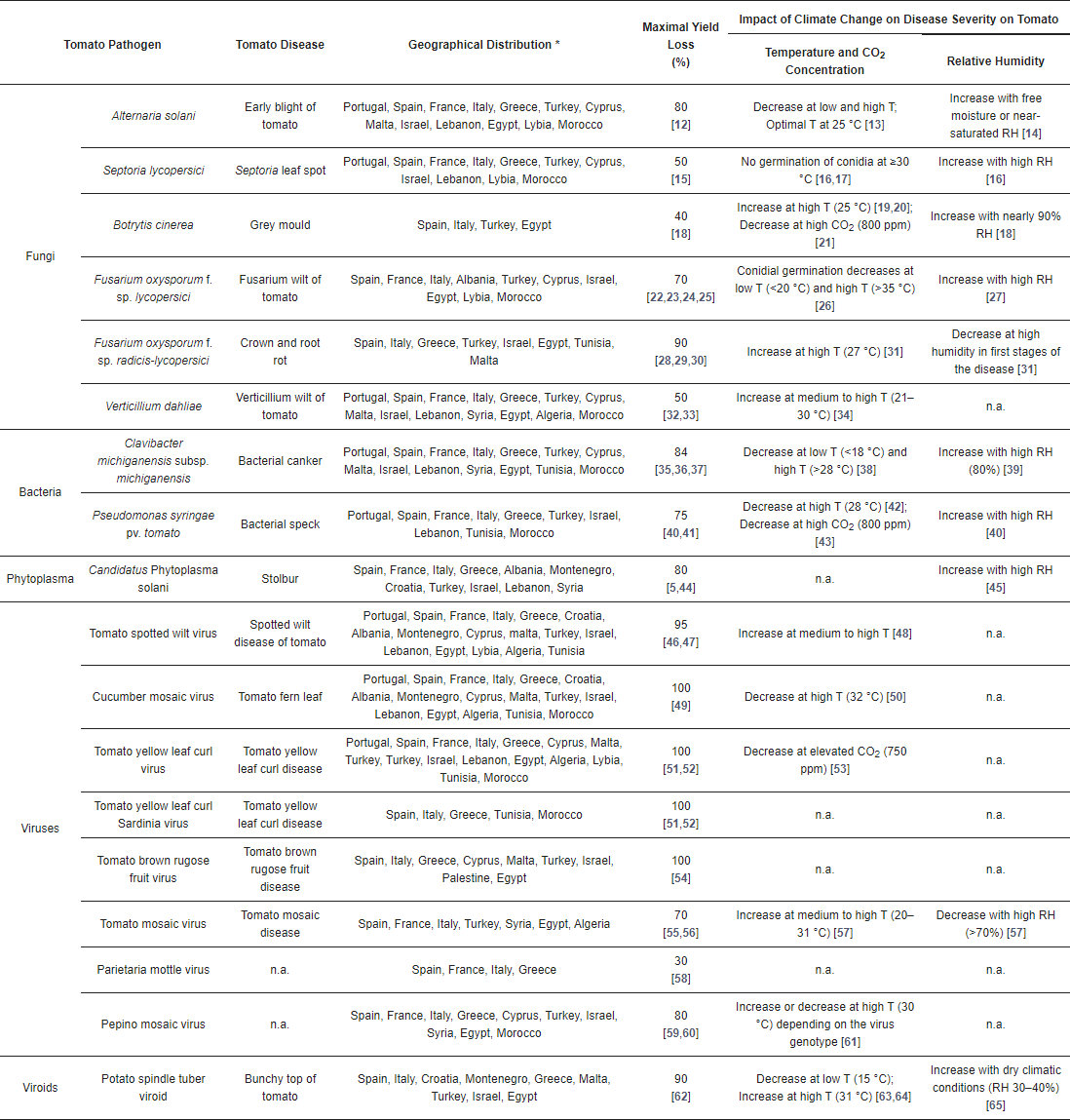
Desarrollar un modelo de detección y clasificación de enfermedades de plantas de tomate utilizando la arquitectura Swin Transformer, con el propósito de conseguir una precisión superior en comparación con las redes neuronales convolucionales.

## Objetivos específicos

* Construir un dataset balanceado en las diferentes clases de las enfermedades del tomate.
* Evaluar e interpretar las predicciones del modelo Swin Transformer para cada clase.
* Comparar y analizar la exactitud (accuracy) del modelo Swin Transformer con modelos basados en redes neuronales convolucionales.

**Tabla 1**

*Lista de patógenos y enfermedades de la planta de tomate*



*Nota*. Adaptado de "A Review of the Most Common and Economically Important Diseases That Undermine the Cultivation of Tomato Crop in the Mediterranean Basin", por Panno et al., 2021, Agronomy, 11(11), p. 2188. https://doi.org/10.3390/agronomy11112188

# ESTADO DEL ARTE

## Machine Learning Approach towards Tomato Leaf Disease Classification

En este artículo Gadade y Kirange, (2020) presentan un enfoque de machine learning para la detección de enfermedades del tomate. El proceso de investigación inició con la recolección de 9,000 imágenes de hojas de tomate del dataset de PlantVillage, enfocándose en siete clases de plantas enfermas y una clase de planta sana.

En la fase de preprocesamiento se minimizó el ruido de las imágenes utilizando un filtro de mediana, después se realizó la extracción de características mediante las técnicas GLCM (Gray-Level Co-Occurrence Matrix), Gabor y SURF (Speeded Up Robust Features). Para clasificar las imágenes en categorías normales o enfermas, se emplearon varios algoritmos de clasificación como la máquina de soporte vectorial (SVM), k-Nearest Neighbors (KNN), Naïve Bayes (NB) y árboles de decisión. Los resultados mostraron que la combinación de SVM y Gabor obtuvo métricas superiores a todas las demás con un 'accuracy' de 0,7339, 'precision' de 0,2525, 'recall' de 0,9492 y 'f1-score' de 0,3989.

Este trabajo evidencia que los métodos tradicionales de Machine Learning muestran métricas muy inferiores en comparación con las técnicas modernas de Deep Learning. Esta observación señala una posible obsolescencia de los métodos tradicionales de Machine Learning en el ámbito de la detección de enfermedades del tomate.

## ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network

En este estudio de Mohit Agarwala et al. (2020) se propone un enfoque basado en una Red Neuronal Convolucional (CNN) para la clasificación de enfermedades del tomate. El estudio utiliza el dataset de PlantVillage seleccionando imágenes de hojas de tomate afectadas por nueve tipos de enfermedades, además de una clase de hojas sanas. Para equilibrar el dataset, se aplicaron técnicas de aumento de datos resultando en 10,000 imágenes para el entrenamiento, 7,000 para la validación y 500 para las pruebas.

La arquitectura CNN desarrollada en el estudio consta de 3 capas convolucionales, 3 capas de max-pooling y 2 capas completamente conectadas. Se evaluaron también modelos CNN preentrenados a través de transferencia de aprendizaje, empleando el mismo dataset, con el objetivo de contrastar la eficacia del modelo propuesto. Las métricas resultantes mostraron un accuracy en un rango de 76% a 100% para diversas clases, con un accuracy promedio de 91.2%. Este rendimiento supera al de modelos preentrenados como VGG16, con una precisión del 77.2%, InceptionV3 con 63.4%, y MobileNet con 63.7%.

El modelo propuesto tiene un número menor de parámetros entrenables comparado al de los modelos preentrenados, sugiriendo ventajas en almacenamiento y computación. No obstante, la variabilidad en la exactitud entre clases indican que el modelo, aunque eficaz presenta deficiencias para la clasificación de algunas enfermedades.

## Tomato plant disease detection using transfer learning with C-GAN synthetic images

Este artículo de Abbas et al. (2021) aborda un enfoque de Deep Learning cuya metodología implementa dos fases, la generación de imágenes sintéticas mediante Redes Antagónicas Generativas Condicionales (C-GAN) para extender el dataset y la clasificación de enfermedades utilizando un modelo pre-entrenado DenseNet121. El desempeño del método se evaluó con el dataset de PlantVillage, que incluye nueve clases de hojas de tomate con distintas enfermedades además de una clase de hojas sanas. Las imágenes sintéticas generadas con C-GAN se añadieron al dataset existente. Luego, se entrenó el modelo DenseNet121 con el conjunto de imágenes extendido.

El primer paso del estudio se centró en el entrenamiento de C-GAN y la generación de imágenes sintéticas; el segundo, en el entrenamiento del modelo DenseNet121 utilizando el dataset ampliado. Un análisis comparativo con modelos preentrenados como VGG19, ResNet50, Inception-V3, Xception y MobileNet mostró que el modelo propuesto logró el nivel más alto de exactitud, con un 97.11%.

No obstante, aunque el modelo propuesto demostró una alta eficacia en la clasificación de enfermedades del tomate y la aplicación de C-GAN ayudó a evitar el sobreajuste, es importante evaluar la fidelidad de las imágenes sintéticas generadas, ya que podrían no tener el nivel de detalle que presentan las imágenes reales y, por lo tanto, afectar negativamente el rendimiento del modelo en tareas de generalización.

## Early Detection and Classification of Tomato Leaf Disease Using High-Performance Deep Neural Network

El estudio de Trivedi et al. (2021) aborda la clasificación de enfermedades de tomate mediante una CNN de arquitectura propia. Se empleó el dataset de PlantVillage con un conjunto de datos compuesto por 3000 imágenes, distribuidas en nueve clases de hojas afectadas y una clase de hojas sanas.

El proceso comprende etapas de preprocesamiento, segmentación y clasificación a través de CNN. La arquitectura CNN especificada consta de ocho capas convolucionales, ocho capas de max pooling, tres capas fully connected, una tasa de dropout de 0.5 y función de activación Relu. Se realizaron experimentos variando parámetros como la tasa de aprendizaje y el número de épocas.

Los resultados de la evaluación con la métrica de accuracy fueron del 98.49%. Los experimentos adicionales mostraron que la precisión varía levemente con diferentes tasas de aprendizaje y épocas, oscilando en general alrededor del 98.4% al 98.58%. La comparación con modelos estándar como MobileNet, VGG16 e InceptionV3 indicó que el modelo propuesto supera a estos modelos CNN.

El estudio resalta la eficacia del modelo propuesto, especialmente en comparación con otros modelos. Sin embargo requiere de un método adicional que es la segmentación de imágenes el cuál no es necesario con otros enfoques, también se omiten discusiones sobre limitaciones, aplicabilidad futura o la justificación para la elección de la arquitectura empleada.

## Tomato leaf disease classification using supervised learning techniques: contrasting analysis

El artículo presentado por Rawat et al., (2022), ofrece una contribución a este campo a través del estudio y comparación de diferentes técnicas tanto de Machine Learning como de Deep Learning. El estudio se articula en torno a cuatro componentes: selección de imágenes del dataset de PlantVillage; preprocesamiento de imágenes para la eliminación de ruido mediante un filtro de mediana; aplicación de diversos algoritmos como SVM, K-Nearest Neighbour, Naïve Bayes, Decision Tree, Feed Forward Neural Network, Back Propagation Neural Network, Deep Neural Network, Convolutional Neural Network y Multi Kernel SVM; y por último la evaluación de la efectividad de estos algoritmos mediante métricas como Precisión, F1-Score y Recall.

El estudio realiza un análisis comparativo de la eficacia de los diferentes algoritmos, la Red Neuronal Convolucional (CNN) logra una exactitud de 98,5%, superando a todos los demás algoritmos. La investigación concluye que las CNN ofrecen un rendimiento superior en la clasificación de enfermedades del tomate en comparación con métodos tradicionales de Machine Learning sin embargo no presenta un enfoque donde se balanceen las clases del problema por lo que el modelo puede presentar sesgos en algunas clases sobrerrepresentadas.

## AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf

El estudio de Chen et al.(2022) aborda la detección y clasificación de enfermedades del tomate mediante una CNN basada en la arquitectura de AlexNet. Los autores utilizaron un conjunto de datos de 18,345 imágenes de entrenamiento y 4,585 imágenes de prueba, distribuidas en nueve clases de enfermedades del tomate y una clase de hoja sana.

La modificación realizada a la arquitectura de la red AlexNet involucra tres capas de convolución, tres capas completamente conectadas y una capa de salida. Se utilizó el optimizador Adam con una tasa de aprendizaje de 0.0005, 75 épocas y un tamaño de lote de 128. La función de pérdida fue la entropía cruzada. El modelo mostró un accuracy del 96%, precision del 98%, recall del 95% y un F1-score del 97%.

Las conclusiones destacan la eficacia del modelo para la clasificación de enfermedades en hojas de tomate. Se señala que el modelo es lo suficientemente ligero como para implementarse en dispositivos móviles, lo que es relevante dadas las limitaciones de capacidad de memoria en estos dispositivos.

## BotanicX-AI: Identification of Tomato Leaf Diseases Using an Explanation-Driven Deep-Learning Model

El estudio de Bhandari et al., (2023), aborda la clasificación de enfermedades del tomate utilizando un modelo de aprendizaje profundo basado en EfficientNetB5. El dataset se construye preprocesando las imágenes mediante aumentos de datos resultando en 11,000 imágenes de hojas de tomate distribuidas en 10 clases.

La metodología consiste en aplicar transferencia de aprendizaje utilizando un modelo EfficientNetB5 al que se le agregaron capas adicionales como la normalización por lotes, capas densas y técnicas de regularización como dropout. Para incrementar la interpretabilidad del modelo se integran técnicas como GradCAM y LIME, que ayudan a identificar las regiones de la imagen que más contribuyen a la predicción.

En cuanto a las métricas el modelo propuesto alcanzó una exactitud del 99.07%, superando a otros modelos preentrenados, como MobileNet (94,00%), Xception (95,32%), VGG16 (93,35%), ResNet50 (96,03%) y DenseNet121 (96,30%). Las técnicas de GradCAM y LIME pudieron identificar las regiones más influyentes de las imágenes para la predicción. Aunque se observó que GradCAM tenía limitaciones en la identificación de ciertas regiones en algunas clases de enfermedades.

En los estudios examinados se puede evidenciar que los algoritmos de Machine Learning como SVM y KNN tienen métricas de desempeño inferiores en comparación con modelos de Deep Learning cómo ResNet50, DenseNet121, EfficientNetB5, Xception, etc. Estas arquitecturas CNN muestran un desempeño superior en todas las métricas y se destacan como el método más efectivo en la clasificación de enfermedades del tomate.

El uso de Redes Antagónicas Generativas Condicionales (C-GAN) para generar imágenes sintéticas puede mejorar el desempeño y prevenir sobreajuste, aunque la calidad de estas imágenes requiere evaluación crítica. Técnicas como GradCAM y LIME ofrecen explicabilidad de la clasificación hecha por el modelo lo cuál es relevante para interpretar cómo realiza el modelo las predicciones. Por último se evidencia que ninguna de las arquitecturas de Vision Transformer ha sido explorada para este problema específico.

# MARCO TEÓRICO

## Enfermedades de la planta de tomate

### Definición

Las enfermedades en plantas de tomate son alteraciones fisiológicas de la planta originadas por entidades biológicas conocidas como patógenos y que afectan negativamente la salud o el rendimiento de la planta. La manifestación de estas enfermedades requieren tres elementos fundamentales; presencia de agentes patogénicos (hongos, bacterias, virus, etc); un huésped (la planta de tomate) que se encuentra en un estado susceptible, debido a condiciones de estrés como sobrefertilización, salinidad del suelo, uso excesivo de pesticidas, competencia por espacio o excesiva irrigación; y por último las condiciones ambientales que facilitan la acción del agente patogénico, tales como temperatura y humedad (Flores, 2012; Bernal, 2010).

### Enfermedades comunes en la planta de tomate

A nivel mundial, más de 200 patógenos afectan a las plantas de tomate, incluyendo hongos, bacterias, fitoplasmas, virus, viroides, nematodos, insectos y ácaros (Corrales Paredes, 2021; Panno et al., 2021). Las enfermedades más habituales son el tizón temprano, tizón tardío, los marchitamientos y diversas enfermedades virales. En la región mediterránea, se enfrentan a enfermedades fúngicas como “early blight”, “vercitilium wilt”, “fusarium wilt”, “Septoria leaf Spot”, “grey mould” y “Crown and root rot”. Las bacterias causan “bacterial canker” y “bacterial speck”, mientras que los fitoplasmas provocan la “Stolbur”. Varios virus se asocian a enfermedades en el tomate como “Spotted wilt”, “Tomato yellow leaf curl”, “Tomato Brown rugose fruit” y “Tomato mosaic”, así como los viroides que causan el síndrome de "bunchy top".

Por otro lado, en regiones tropicales, las enfermedades bacterianas en tomate abarcan sarna bacteriana, moteado y cancro bacteriano, y en contextos tropicales húmedos, es común el marchitamiento bacteriano. Tanto en regiones templadas como tropicales, se observan virus como el Tomato yellow curl virus, Tomato spotted wilt virus y Pepino mosaic virus. En cuanto a las enfermedades fúngicas, están presentes patógenos como Phytophthora infestans, Alternaria tomatophila, Verticillium dahliae y Fusarium oxysporum (Blanchard, 2011).

### Síntomas

Los síntomas de las enfermedades del tomate varían considerablemente y están estrechamente ligados al tipo de patógeno involucrado y a la enfermedad específica que se manifiesta. Además, pueden verse influenciadas por factores externos como condiciones ambientales y prácticas de manejo inadecuado del cultivo (Panno et al., 2021). Las enfermedades que afectan el tomate se pueden categorizar según los síntomas que manifiestan.

El early blight y la Septoria leaf spot comparten características como defoliación y lesiones necróticas en diferentes partes de la planta. El fusarium wilt, Crown and root rot y verticillium wilt presentan síntomas de marchitamiento y pardeamiento vascular. El grey mould se distingue por síntomas como podredumbre blanda y encharcamiento de tejidos. El bacterial canker y bacterial speck muestran lesiones en diferentes partes de la planta y enrollamiento de las hojas. Stolbur phytoplasma, Tomato yellow leaf curl y Tomato brown rugose fruit presentan síntomas como amarilleamiento, atrofia y escasez de hojas. Tomato spotted wilt, Parietaria mottle y Cucumber Mosaic virus incluyen síntomas de deformación foliar y lesiones necróticas. Tomato mosaic y Pepino mosaic virus se caracterizan por síntomas como moteado de hojas y deformación de los frutos. Potato spindle tuber viroid manifiesta una variedad de síntomas, incluyendo retraso del crecimiento y deformidad de los frutos (Panno et al., 2021). Es crucial para el manejo efectivo de la enfermedad identificar correctamente los síntomas asociados con cada patógeno o enfermedad específica (Blancard, 2011).

### Métodos tradicionales de detección de enfermedades

* + - 1. **La detección manual.** Se lleva a cabo por expertos que necesitan un amplio rango de conocimientos y experiencia en diversas disciplinas. Estos conocimientos son esenciales debido a la variedad de enfermedades existentes. El diagnóstico no solo se basa en síntomas y signos, sino también en la comprensión de la biología de los agentes patógenos y el contexto en el que la enfermedad ocurre; Implica observaciones detalladas, preguntas dirigidas a contextualizar la enfermedad y razonamiento para priorizar las posibles causas. Este enfoque requiere una considerable cantidad de tiempo y recursos humanos para su implementación. Además, la evaluación manual es susceptible a errores humanos como sesgos o variables cognitivas y emocionales que afectan el juicio clínico (Blancard, 2011; Thangaraj et al., 2022; Garcia, 2016).
      2. **Los métodos de laboratorio.** Incluyen técnicas inmunológicas, cultivo de patógenos y microscopía. Aunque estos métodos son precisos, son laboriosos, costosos y requieren tiempo, lo que puede no ser viable para un diagnóstico rápido y eficaz (Verma et al., 2018; Valdivieso, 2021).

### Desafíos en la detección de enfermedades

Dado los desafíos asociados con los métodos tradicionales de detección de enfermedades, como el tiempo, costo y la necesidad de expertos altamente calificados, se evidencia una necesidad imperante de desarrollar nuevas estrategias para la detección y control de enfermedades en la planta de tomate. Errores o retrasos en la detección e intervención pueden llevar a pérdida de rendimiento y calidad del producto, además de riesgos para la salud pública (Garcia, 2016; Valdivieso, 2021). El uso indebido de pesticidas, a menudo utilizado como medida preventiva ante la incertidumbre en la detección de enfermedades, puede resultar en la acumulación de residuos químicos en los alimentos, comprometiendo así la seguridad alimentaria (Corrales Paredes, 2021).

## Seguridad alimentaria (food safety)

La seguridad alimentaria se centra en prevenir enfermedades transmitidas por alimentos a través de la gestión de toda la cadena de producción, desde el cultivo hasta el consumo. Las fuentes comunes de contaminación incluyen hongos, bacterias, virus, viroides, parásitos y sustancias químicas (Guennouni et al., 2022; FAO, 2022). Esta contaminación amenaza tanto la salud individual como pública y puede comprometer la calidad nutricional de los alimentos. Por lo tanto, es necesario implementar medidas de control en cada etapa de la cadena alimentaria (Uçar et al., 2016).

Es por eso por lo que en el ámbito de la producción de plantas como el tomate la detección temprana de enfermedades cumple un rol fundamental para la seguridad alimentaria, velando por la salud tanto del cultivo como la de los consumidores. Ante esta necesidad, la tecnología de visión artificial se presenta como un método efectivo para identificar enfermedades de forma automática, no destructiva y económica. Esta tecnología contribuye a la biovigilancia al permitir una identificación más rápida y precisa de microorganismos patógenos (Khan et al., 2021).

## Visión artificial

La visión artificial contribuye a la optimización sostenible de la producción agrícola a través de una nueva tecnología de identificación y diagnóstico de plagas y enfermedades en plantas mediante el análisis de síntomas foliares (J. Gómez-Camperos et al., 2021). Esta tecnología ofrece ventajas a comparación de los métodos tradicionales tales como mayor velocidad de operación, consistencia, fiabilidad y adaptabilidad en entornos industriales (Khan et al., 2021).

Los avances en reconocimiento de patrones y aprendizaje automático han permitido el desarrollo de nuevos métodos diagnósticos. Esto ha incrementado la investigación y la implementación de nuevos modelos y técnicas de visión artificial (Nanehkaran et al., 2020). Este es un campo de la inteligencia artificial que tiene como objetivo permitir a sistemas computacionales procesar e interpretar datos visuales, para lo cual se utilizan algoritmos y modelos matemáticos que extraen información de imágenes y vídeos (Xu et al., 2021). El campo está en continuo desarrollo y tiene aplicaciones en áreas como detección de objetos, segmentación y clasificación de imágenes (Mahadevkar et al., 2022).

### Clasificación de Imágenes

Es un proceso cuyo objetivo es categorizar imágenes según su contenido a partir de un conjunto de clases ya establecido. Es relevante en diversos campos científicos e ingenieriles para la captura y tratamiento de datos visuales. En el contexto de detección de enfermedades en plantas de tomate, la clasificación de imágenes adquiere una importancia particular puesto que la correcta identificación de síntomas en las hojas permite la clasificación de la enfermedad específica que afecta al cultivo (Vocaturo, 2021). Esto proporciona una base para intervenciones de manejo más precisas y oportunas, lo que puede mitigar pérdidas de rendimiento y calidad del producto. Para la clasificación de imágenes se utilizan múltiples técnicas, incluyendo redes neuronales artificiales (ANN por sus siglas en inglés), redes neuronales convolucionales (CNN), máquinas de soporte vectorial (SVM), k-nearest neighbor (k-NN) y algoritmos de random forest. Estas técnicas facilitan la extracción de características de las imágenes para entrenar un modelo que posteriormente pueda clasificar nuevas imágenes (Sanghvi et al., 2020).

### Machine learning y Deep learning en visión artificial

La visión artificial utiliza enfoques del Machine Learning y el Deep Learning. El Machine Learning emplea una variedad de algoritmos como k-NN, Bayesian classification (Naives Bayes), Linear Regression y SVM (Szeliski, 2020). El Deep Learning es un subcampo del Machine Learning que usa grandes redes neuronales artificiales para resolver problemas complejos y ha superado las técnicas de machine learning en las diferentes tareas de visión artificial (Mahadevkar et al., 2022; Szeliski, 2020). Una distinción clave entre los dos enfoques es que en el Machine Learning, las características de las imágenes se extraen manualmente, mientras que en Deep Learning, el modelo aprende representaciones y características de manera autónoma (Szeliski 2020; Shorten & Khoshgoftaar, 2019).

El deep learning se ha destacado por su eficacia en la clasificación de imágenes. Este enfoque automatiza la optimización del modelo durante el proceso de entrenamiento (Szeliski, 2020). Las redes neuronales convolucionales (CNN) se han convertido en el estándar en las tareas de visión artificial debido a su rendimiento superior (Shah & Harpale, 2018). Sin embargo, los Vision Transformers (ViT), son una arquitectura más reciente y han surgido como competidores efectivos a las CNN, adaptando técnicas originalmente diseñadas para el procesamiento de lenguaje natural a tareas de visión artificial (Amjoud & Amrouch, 2023).

## Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales (CNN) son una variante de las redes neuronales artificiales tradicionales (ANN) optimizadas para el procesamiento de imágenes. Estas redes tienen la habilidad de aprender a identificar y categorizar distintos patrones en imágenes durante su fase de entrenamiento. (O’Shea & Nash, 2015). Szeliski (2020) describe que, estas redes se han convertido en una técnica dominante en el campo de la visión artificial (Xiao et al., 2022; Nanehkaran et al., 2020). Sin embargo, aunque las CNN son eficientes para el procesamiento de imágenes, presentan ciertas limitaciones. Una de ellas radica en los sesgos inductivos inherentes a las operaciones convolucionales, que asumen que los píxeles cercanos son más importantes que los píxeles distantes (Szeliski, 2020). Estos sesgos pueden limitar el progreso a medida que crecen los conjuntos de datos, los modelos y el poder computacional. En este contexto, los Vision Transformers (ViTs) emergen como una solución que aborda estas limitaciones.

## Vision Transformer

Los Vision Transformers (ViTs) han ganado relevancia en la clasificación de imágenes y han establecido nuevos estándares de rendimiento (Papers With Code, 2023). Estos modelos toman ideas originalmente usadas en procesamiento de lenguaje natural y las aplican a la visión artificial. En lugar de concentrarse en pequeñas áreas de una imagen, como lo hacen las Redes Neuronales Convolucionales (CNN), los ViTs consideran la imagen completa, lo que les permite entender relaciones más complejas en los datos visuales (Alexey Dosovitskiy et al., 2020). Esto los ha llevado a superar a las CNN en diversas evaluaciones de referencia en el ámbito de la visión artificial (Szeliski, 2020; Jamil et al., 2023).

No obstante, los ViTs no están exentos de desafíos. Uno de los principales obstáculos es la cantidad de recursos computacionales que requieren, especialmente cuando se manejan imágenes de gran tamaño. Esta alta demanda computacional hace que los ViTs sean menos eficientes que las CNN en algunos escenarios (Szeliski, 2020). Esto ha impulsado la búsqueda de alternativas más eficientes, como la arquitectura Swin Transformer.

## Swin Transformer

Swin Transformer, también conocido como Shifted Window Transformer, es una arquitectura que busca abordar ciertas limitaciones presentes en las arquitecturas de Vision Transformers (ViTs) y redes neuronales convolucionales (CNN). La arquitectura se caracteriza por dividir la imagen en parches y aplicar atención dentro de ventanas locales no solapadas, en contraste con la atención global empleada en ViTs. Este enfoque reduce la complejidad computacional a lineal respecto al tamaño de la imagen. Adicionalmente, la arquitectura utiliza una estructura jerárquica para procesar imágenes a diferentes resoluciones, lo que le permite capturar características en múltiples escalas. En pruebas de referencia, se ha encontrado que Swin Transformer supera a otras arquitecturas de ViTs y CNN en términos de rendimiento en tareas como la clasificación de imágenes en el conjunto de datos de ImageNet (Liu et al., 2021; Papers With Code, 2023). Este diseño le confiere aplicabilidad a una variedad de tareas en el ámbito de la visión artificial, incluida la clasificación de imágenes, la detección de objetos y la segmentación semántica.

### Arquitectura de Swin Transformer

Swin Transformer, aborda las limitaciones de los ViT introduciendo dos conceptos clave: ventanas desplazadas y estructura piramidal. Swin Transformer comienza dividiendo una imagen en parches. Cada parche se considera un "token" y sus características se establece concatenando los valores RGB de los píxeles en crudo. Estas características se proyectan posteriormente a una dimensión arbitraria (denotada como C) mediante una capa de embedding lineal.

En Swin Transformer, la resolución de la imagen se reduce sucesivamente, de la "Etapa 1" a la "Etapa 4". El término "Etapa" se refiere a las distintas fases de procesamiento de la imagen a través de la arquitectura. Esta estructura jerárquica es similar a la de las redes neuronales convolucionales en las que las características se procesan a distintas resoluciones (Liu et al., 2021).

### Ventanas desplazadas

En lugar de realizar una atención global sobre toda la imagen (lo que tendría una complejidad cuadrática con respecto al número de tokens), Swin Transformer introduce el concepto de atención dentro de ventanas locales no solapadas. Esto reduce significativamente la complejidad computacional. La atención basada en ventanas es escalable y computacionalmente más eficiente (Liu et al., 2021).

**Figura 1**

*El enfoque de ventana deslizante para calcular la autoatención en la arquitectura Swin Transformer*

Imagen de la pantalla de una ventana

Descripción generada automáticamente con confianza media

*Nota*. Adaptado de "Swin transformer: Hierarchical vision transformer using shifted windows", por Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B., 2021, In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012-10022.

Sin embargo, para no perder la capacidad de modelar las relaciones entre parches en distintas ventanas, Swin Transformer introduce la idea de ventanas desplazadas en bloques sucesivos. Esta estrategia alterna entre dos configuraciones de particionamiento en bloques consecutivos de transformers, lo que permite que cada parche se relacione con los parches de las ventanas vecinas. También se introduce un sesgo de posición relativa para tener en cuenta las relaciones espaciales entre parches dentro de una ventana. Este sesgo mejora significativamente el rendimiento del modelo y es preferible a los embeddings de posición absoluta (Liu et al., 2021).

### Estructura piramidal

Swin Transformer procesa imágenes en una serie de resoluciones, de forma similar a una pirámide. Comienza con ventanas pequeñas en una resolución más alta y agrupa gradualmente los parches mientras reduce la resolución, permitiendo que el modelo capture características en diferentes escalas (Liu et al., 2021).

**Figura 2**

*Comparación del método de estructura piramidal utilizado por Swin tranformer y el método utilizado por Vit*

Gráfico, Diagrama

Descripción generada automáticamente

*Nota*. Adaptado de "Swin transformer: Hierarchical vision transformer using shifted windows", por Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B., 2021, In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012-10022.

### Detalles arquitectónicos de Swin Transformer

**5.6.4.1 Token:** se refiere a un parche de la imagen original. Cada parche se convierte en una representación vectorial o embedding que sirve como unidad básica de entrada para las capas posteriores del modelo.

**5.6.4.2 Embedding:** hace referencia a la representación vectorial de alta dimensión en la que se proyectan los tokens o parches de la imagen. Estas proyecciones vectoriales se utilizan para el procesamiento y la identificación de características en capas subsecuentes del modelo.

**5.6.4.3 Atención:** Este mecanismo permite al modelo asignar diferentes niveles de importancia a diversas partes de la entrada cuando está procesando información.

**5.6.4.4 Bloque Swin Transformer:** Un bloque Swin Transformer es una unidad de arquitectura en la red que contiene módulos específicos para la atención y la transformación de características. En general, cada bloque incluye un módulo de atención seguido de una capa de normalización y un perceptrón multicapa (MLP). Estos bloques se pueden apilar para formar la arquitectura completa del Swin Transformer.

**5.6.4.5 MSA (Multi-Head Self-Attention):** un tipo de mecanismo de atención que permite al modelo prestar atención a diferentes partes de la entrada de manera simultánea. Es esencial para la capacidad del transformer de capturar relaciones a larga distancia.

**Figura 3**

*A la izquierda está la arquitectura de Swin Transformer (Swin-T) y a la derecha están dos bloques sucesivos del Swin Transformer W-MSA y SW-MSA, que son módulos de auto-atención multi-cabeza con configuraciones de ventana regulares y deslizantes, respectivamente.*

Diagrama, Esquemático

Descripción generada automáticamente

*Nota*. Adaptado de "Swin transformer: Hierarchical vision transformer using shifted windows", por Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B., 2021, In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012-10022.

**5.6.4.6 MLP (Multi-Layer Perceptron):** Se refiere a una red neuronal completamente conectada. En el contexto de los transformers, los MLP se utilizan para transformar características después de las capas de atención.

**5.6.4.7 LN (LayerNorm):** LN, o Normalización de Capa, es una técnica de normalización que se aplica a las características en una capa específica. En Swin Transformer, LN se aplica antes de cada módulo de auto-atención multi-cabeza (MSA) y cada MLP. La normalización ayuda a estabilizar y acelerar el entrenamiento.

**5.6.4.8 Conexión Residual:** En redes profundas, las conexiones residuales ayudan a evitar el problema de desvanecimiento del gradiente al permitir que las activaciones se salten una o más capas. En Swin Transformer, se aplica una conexión residual después de cada módulo de atención y cada MLP.

**5.6.4.9 Fusión de Parches:** "Patch Merging" es una técnica utilizada en el Swin Transformer para reducir la resolución espacial del mapa de características e incrementar simultáneamente la dimensión de la característica, permitiendo que el modelo capture representaciones más abstractas a medida que se profundiza. Funciona agregando información de parches vecinos y creando un nuevo parche con una dimensión de característica más alta (Liu et al., 2021).

Después de explorar la arquitectura Swin Transformer, es necesario considerar el dataset en el que este modelo se va a entrenar y evaluar, debido a que las características propias del dataset también pueden influir en el resultado del entrenamiento del modelo.

## Dataset

Según Cappi et al. (2021) un "dataset" es un conjunto de muestras que representan una población específica. En el ámbito de Machine Learning y el Deep Learning, se utiliza para construir y evaluar modelos. Cada muestra dentro del dataset es un ejemplo concreto que refleja situaciones o condiciones que se esperan encontrar en el entorno donde se aplicará el modelo. Estos datos pueden incluir, por ejemplo, píxeles, imágenes o series temporales.

La representatividad del dataset es esencial para la evaluación precisa del rendimiento del sistema en las fases de prueba y entrenamiento. El conjunto de datos debe abarcar el espectro de situaciones que el sistema enfrentará en su operación. En procesamiento de imágenes, esto implica variedad en condiciones físicas y ambientales, objetos de interés y niveles de exposición (Cappi et al., 2021).

Mohanty et al. (2016) abordan la complejidad de crear un dataset que minimice sesgos que puedan surgir durante el proceso de recopilación de datos. Subrayan la importancia de disponer de un dataset amplio y rigurosamente verificado para el desarrollo eficaz de clasificadores de imágenes en el campo de diagnóstico de enfermedades de plantas. Esta necesidad se acentúa por la ausencia histórica de datasets de alta calidad en este ámbito específico, brecha que el proyecto PlantVillage ha logrado cubrir.

## Dataset de plantVillage

El dataset PlantVillage proporciona un conjunto de datos para entrenar modelos de detección de enfermedades en diversas plantas. Mohanty et al. (2016) afirman que, para el desarrollo de clasificadores de imágenes precisos para el diagnóstico de enfermedades de las plantas, es necesario un conjunto de datos grande y verificado de imágenes de plantas enfermas y saludables. El dataset de PlantVillage contiene 54,306 imágenes de hojas de plantas repartidas en 38 clases etiquetadas que abarcan 14 especies de cultivos y 26 enfermedades (Mohanty et al., 2016). Este dataset incluye una amplia gama de clases de enfermedades del tomate, lo que lo hace idóneo para el objeto de esta investigación.

### Clases asociadas al tomate en el dataset de plantVillage

El dataset de plantVillage contiene 9 clases para enfermedades de la planta del tomate y una clase para la planta sana.

**Tabla 2**

*Clases de las enfermedades del tomate*

| **Nombre enfermedad** | **Número de imágenes** |
| --- | --- |
| Tomato Bacterial Spot | 2127 |
| Tomato Early Blight | 1000 |
| Tomato Late Blight | 1759 |
| Tomato Leaf Mold | 952 |
| Tomato Septoria Leaf Spot | 1771 |
| Tomato Two Spotted Spider Mite | 1676 |
| Tomato Target Spot | 1404 |
| Tomato Yellow Leaf Curl Virus | 3208 |
| Tomato Mosaic Virus | 373 |
| Tomato Healthy | 1590 |
| Total | 15860 |
| Promedio | 1586 |

*Nota*. Adaptado de "Using deep learning for image-based plant disease detection", por Mohanty, S. P., Hughes, D. P., & Salathé, M., 2016, Frontiers in Plant Science, 7, p. 1419.

Aunque el dataset de PlantVillage constituye una base sólida para el entrenamiento de un modelo de clasificación de las enfermedades del tomate, también se necesita un método eficaz para minimizar el sobreajuste del modelo, este método es el aumento de datos como menciona Szeliski (2020).

## Aumento de datos

Este enfoque consiste en aplicar perturbaciones a las muestras recogidas para expandir el conjunto de datos de entrenamiento. Resulta especialmente útil dado que obtener ejemplos etiquetados puede ser costoso y las clases de las imágenes deben mantenerse constantes bajo pequeñas perturbaciones locales. Su objetivo es aumentar la cantidad y la variabilidad de los datos disponibles para el entrenamiento de modelos, lo que puede mejorar significativamente el rendimiento y la capacidad de generalización de los algoritmos de detección y clasificación.

Según Khosla y Saini (2020), el aumento de datos se segmenta en dos clases principales: transformaciones geométricas y sobremuestreo. En la categoría de transformaciones geométricas, se utilizan técnicas como el volteo, cambio en el espacio de color, recorte, rotación, traslación e inyección de ruido para alterar la apariencia y estructura de las imágenes. Estas técnicas sirven para mitigar sesgos posicionales en los conjuntos de datos.

En cuanto al sobremuestreo, se emplean métodos como la mezcla de imágenes, en la que se superponen dos imágenes seleccionadas al azar para crear una nueva. También se pueden aplicar técnicas de interpolación y extrapolación en el espacio de características de bajo nivel. Además, las Redes Generativas Antagónicas (GANs) se emplean para crear muestras sintéticas, contribuyendo así a la diversidad del conjunto de entrenamiento. El aumento de datos puede añadir variabilidad al conjunto de datos de entrenamiento, lo que es especialmente valioso cuando el conjunto de datos es limitado o desequilibrado. Este enfoque en el aumento de datos contribuirá a mejorar la precisión y la robustez de los modelos. Este proceso tiene implicaciones directas en las métricas de evaluación, ya que un modelo bien ajustado a un conjunto de datos más diverso podría mostrar una mayor eficacia.

## Métricas de Evaluación

Para evaluar la eficacia de un modelo de clasificación, Szeliski (2020) y Flach & Kull (2015) sugieren una serie de métricas, incluyendo Accuracy, Precision, Recall y F1-score.

“Accuracy” mide la proporción de predicciones correctas realizadas por el modelo sobre el total de predicciones realizadas. Sin embargo, la precisión puede ser engañosa cuando los datos están desequilibrados, es decir, cuando una clase es mucho más frecuente que las demás. En estos casos, un modelo que siempre predice la clase mayoritaria tendrá una gran precisión, pero puede no ser útil en la práctica.

“Precision” mide la proporción de verdaderos positivos de todos los casos predichos como positivos, es útil cuando el coste de los falsos positivos es elevado, como en el diagnóstico médico. “Recall” mide la proporción de verdaderos positivos entre todos los casos positivos reales y es útil cuando el coste de los falsos negativos es elevado, como en la detección de fraudes. Estas métricas son especialmente relevantes para conjuntos de datos desequilibrados.

“F1-score” es la media armónica de Precision y Recall, y equilibra la compensación entre ambas, es útil cuando tanto la Precision como el Recall son importantes, y se suele utilizar en problemas de clasificación binaria (Flach & Kull, 2015).

## Matriz de Confusión

Una matriz de confusión es el gráfico de una matriz que muestra el rendimiento de un modelo de clasificación. Utiliza cuatro categorías para mostrar las predicciones: verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. Esto ayuda a calcular diferentes medidas de rendimiento. La matriz tiene filas y columnas que representan el número de clases que se están evaluando. Si el número en la diagonal principal es alto, significa que el modelo ha hecho buenas predicciones mientras que los números fuera de la diagonal señalan errores del modelo. La matriz ofrece una forma numérica y visual de ver qué tan bien el modelo puede diferenciar entre las distintas clases involucradas (Susmaga, 2004).

## Interpretabilidad mediante mapas de atención

Los mapas de atención en visión artificial se refieren a una técnica utilizada en modelos de aprendizaje profundo para visualizar qué partes de una imagen están siendo atendidas por el modelo. Los mapas de atención son generados por el modelo y resaltan las regiones de la imagen de entrada que son más relevantes para la salida. Esta técnica se utiliza para mejorar la interpretabilidad de los modelos de aprendizaje profundo y para comprender mejor cómo realizan las predicciones (Ulutan et al., 2020).

# MODELO MATEMÁTICO

## Detección y eliminación de imágenes duplicadas del dataset

El algoritmo de detección de imágenes duplicadas utiliza el método perceptual hashing para obtener un hash único (o "huella digital") de una imagen. Esta huella digital es robusta contra modificaciones leves en la imagen, lo que significa que imágenes similares o ligeramente modificadas tendrán hashes similares.

### Redimensionamiento y Conversión a Escala de Grises

El primer paso es reducir el tamaño de la imagen. Primero se escala la imagen a un tamaño de 128x128. Luego se convierte a escala de grises para trabajar solo con la luminosidad.

Donde es la imagen original e es la imagen reescalada y en escala de grises.

### Transformada de Coseno Discreta (DCT)

Se aplica la Transformada de Coseno Discreta a la imagen en escala de grises. La transforma la imagen del dominio espacial al dominio de la frecuencia. Los componentes de baja frecuencia (que representan las características generales de la imagen) están en la esquina superior izquierda de la matriz resultante, mientras que los de alta frecuencia están en la esquina inferior derecha.

Aquí es el valor de la matriz DCT en la posición , es el valor del píxel de la imagen en la posición y es una dimensión de la imagen (en este caso, 128).

### Reducción de la DCT

Se toma una submatriz de la esquina superior izquierda de la matriz . Aunque el es de 128 ⨉ 128, solo se toma la parte superior izquierda de 16 ⨉ 16 ya que esta submatriz representa las frecuencias más bajas en la imagen.

### Cálculo de la mediana

Se calcula la mediana de la . Primero, los elementos de se ordenan en una secuencia ascendente. Luego, la mediana se calcula como sigue:

Donde es el número total de elementos en (en este caso, M = 16 ⨉16 = 256) y 𝓍 representa un elemento en la secuencia ordenada de elementos en . Dado que es par (256), la mediana se calcula como el promedio de los elementos en las posiciones 128 y 129 de la secuencia ordenada.

### Binarización

Se binariza la matriz comparando cada uno de los 256 valores con la mediana calculada previamente. Si un valor en una posición específica es mayor que la mediana, se le asigna un 1 en la matriz binaria; de lo contrario, se asigna un 0.

Esta matriz binaria se denota como , y representa el valor en la i-ésima fila y j-ésima columna de la matriz.

### Construcción del Hash

La matriz binaria se convierte en el hash, se toma el arreglo multidimensional y se "aplana" a un arreglo unidimensional. Dado que bin es una matriz de tamaño al aplanar esta matriz, se obtendrá un arreglo unidimensional de tamaño que es el hash de la imagen.

### Comparación de Hashes

Se compara el hash de dos imágenes utilizando la distancia de Hamming, que cuenta el número de posiciones en las que los correspondientes bits son diferentes. Una distancia de Hamming baja indica que las imágenes son similares, mientras que una distancia alta indica que son diferentes. La distancia de Hamming entre dos hash de igual longitud es el número de posiciones en las que los correspondientes símbolos son diferentes.

La fórmula para la distancia de Hamming entre dos hash y de igual longitud es:

Donde:

y son los bits en la posición de los hash y respectivamente. es el operador XOR.

La similitud porcentual entre dos hashes se calcula como

El código de búsqueda de imágenes duplicadas recorre todas las imágenes, calcula su perceptual Hash y luego compara estos hashes para encontrar imágenes similares o duplicadas basadas en el porcentaje de similitud definido. Las imágenes consideradas similares o duplicadas se muestran juntas para una revisión visual y posterior eliminación.

## Aumento de datos

En esta sección, se aplican diversas técnicas de aumento de datos provistas por la librería Albumentations. Estas técnicas incluyen rotaciones, desplazamientos, recortes, inversión horizontal y vertical. Cada una de estas técnicas se diseña para simular diferentes condiciones de visualización y variaciones en las imágenes enriqueciendo así el conjunto de datos de entrenamiento.

### Rotaciones

Para aplicar rotaciones en imágenes, se utiliza la función albumentations.Rotate. Esta función permite rotar una imagen con un número aleatorio de grados.

Donde:

* **Limit** es el rango de ángulos para la rotación. Puede ser un único valor numérico, lo que significa que la imagen se rotará un número aleatorio de grados entre −degrees y +degrees. También puede ser una tupla de dos números, especificando el rango mínimo y máximo para la rotación. La función rota la imagen un ángulo aleatorio dentro del rango especificado.

### Desplazamientos

Para aplicar desplazamientos en imágenes, se utiliza la función albumentations.ShiftScaleRotate. Esta función permite realizar transformaciones afines que incluyen desplazamientos, rotaciones y escalado. En el contexto de desplazamientos, se especifican los máximos desplazamientos en porcentajes de la imagen original tanto en la dirección horizontal como en la vertical.

Donde:

* **Rotate\_limit** son los ángulos de rotación. Para desplazamientos, este valor se establece en 0.
* **Scale\_limit** es el rango de escalado. Para desplazamientos, este valor se establece en 0.
* **Shift\_limit** es una tupla que contiene los máximos desplazamientos en la dirección horizontal (max\_dx) y vertical (max\_dy), estos desplazamientos se aplican aleatoriamente dentro de los límites especificados. Los valores se expresan como una fracción del tamaño de la imagen.

### Recorte

Esta operación realiza un recorte de un tamaño aleatorio en una región de la imagen aleatoria, para ello se utiliza la función

Donde:

* **Height y width** son las dimensiones finales de la imagen después del recorte.

### Inversión Horizontal

Se lleva a cabo mediante la función albumentations.HorizontalFlip. Esta función acepta un único parámetro: la probabilidad de aplicar la inversión horizontal a una imagen dada. Esto se logra con la función

Donde:

* **p** es la probabilidad de aplicar la inversión horizontal. Se expresa como un valor decimal entre 0 y 1, donde 0 significa que nunca se aplicará la inversión y 1 significa que siempre se aplicará. La función invierte la imagen horizontalmente con la probabilidad especificada, generando una versión espejo de la imagen original en el eje horizontal.

### Inversión Vertical

Se realiza mediante la función albumentations.VerticalFlip. Esta función toma un único parámetro: la probabilidad de aplicar la inversión vertical a una imagen dada.

Donde:

* **p** es la probabilidad de aplicar la inversión vertical. Se expresa como un valor decimal entre 0 y 1, donde 0 significa que nunca se aplicará la inversión y 1 significa que siempre se aplicará. La función invierte la imagen verticalmente con la probabilidad especificada, creando una versión espejo de la imagen original en el eje vertical.

## Arquitectura Swin Tranformer

La arquitectura Swin Transformer inicia su proceso tomando una imagen RGB de entrada de dimensiones y la transforma en una secuencia de "tokens". Este proceso consta de dos pasos principales: la División en Parches y la capa de embedding Lineal. Aquí, y son las dimensiones de la altura y la anchura de la imagen, respectivamente, y 3 es el número de canales RGB.

### División de parches

La imagen de entrada se divide en parches no superpuestos. Cada parche se considera un "token" y sus características se establecen como una concatenación de los valores RGB de los píxeles en crudo cuya dimensión es 48, esto se debe a que dimensiones del parche se establecen en 4×4 y cada píxel tiene 3 canales (RGB). Matemáticamente, este proceso se puede representar como la función que extrae un parche específico de la imagen y se puede definir de la siguiente manera:

Donde:

* *P* es un tensor que contiene el conjunto de parches de la imagen.
* *PatchPartition* es la función que extrae los parches de la imagen.
* es la imagen original.
* Es el número total de parches *P* extraídos de la imagen.

La dimensión de la concatenación de características del parche se calcula como:

Y el número total de características de *P* es:

### Capa de embedding lineal (Linear Embedding)

Una vez que los parches son extraídos, cada uno se transforma mediante una capa de proyección lineal. Esta proyección lineal se logra utilizando una convolución 2D con un tamaño de núcleo, un paso igual al tamaño del parche, y un número de canales de salida igual a . El resultado es aplanado y transpuesto para obtener un tensor de forma

La proyección lineal se puede expresar como:

Donde:

* es el embedding correspondiente al parche .
* *i y j* son la fila y la columna de los tensores *E* y *P.*
* Es la matriz de pesos de la convolución.
* Es el vector de sesgo del embedding.
* Es la dimensión del espacio de embedding.

La operación de convolución 2D utilizada en la capa de embedding lineal es equivalente a aplicar una capa totalmente conectada a cada parche de la imagen. Esto se debe al hecho de que el tamaño del núcleo de la convolución y el paso son iguales al tamaño del parche, lo que resulta en un mapeo uno a uno de cada parche a su correspondiente embedding en el espacio de características de dimensión . La dimensión en la que se proyectan los parches se establece según el tamaño de la arquitectura:

**Tabla 3**

*Dimensiones del embedding C según el tamaño de la arquitectura Swin Transformer.*

| **Tamaño de la arquitectura** | **Dimensión de *C*** |
| --- | --- |
| Swin- T | 96 |
| Swin-S | 96 |
| Swin-B | 128 |
| Swin-L | 192 |

*Nota*. Adaptado de "Swin transformer: Hierarchical vision transformer using shifted windows", por Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B., 2021, In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012-10022.

### Bloques de Swin Transformer

Swin Transformer introduce la idea de ventanas desplazadas en bloques Swin Transformer sucesivos. Sobre los embeddings se aplican varios de estos bloques. Cada bloque contiene un módulo de atención basada en ventanas (W-MSA) seguido de un perceptrón multicapa (MLP) de dos capas con una función de activación GELU. Antes de cada instancia de W-MSA y MLP, se incorpora una capa de normalización LayerNorm (LN) y posterior a cada módulo, se añade una conexión residual. En bloques subsiguientes, el módulo W-MSA se sustituye por un módulo de atención basada en ventana desplazada (SW-MSA). La complejidad computacional se mantiene lineal al fijar el tamaño de cada ventana (7 por defecto).

### Normalización de Capa (LayerNorm)

La normalización de capas se realiza en la dimensión de las características para cada token. La fórmula matemática para la normalización de capas es:

Donde es el embedding de entrada, y son la media y la varianza a lo largo de la dimensión de las características, es un valor pequeño para evitar la división por cero, y y son parámetros entrenables. El operador denota el producto elemento a elemento.

### Autoatención Basada en Ventanas (W-MSA)

La atención se calcula solo dentro de cada ventana de embeddings. Para calcular la atención, se utilizan las matrices (query), (key) y (value), que se obtienen a partir de la matriz . Primero se ejecuta una operación de capa lineal que proyecta el tensor de entrada al espacio tridimensional con dimensión .

donde *qkv* es un tensoren el que se calculan *Q*, *K* y *V* simultáneamente, y son los pesos y sesgos entrenables, respectivamente. Después de la proyección, se reorganiza el tensor para separar las proyecciones y adaptarlo para la atención de múltiples cabezas.

* son las matrices de query, key y value.
* es la dimensión de los vectores y .
* *B* es el desplazamiento de la ventana.
* *SoftMax* es la función que convierte las puntuaciones en pesos de atención que varían entre 0 y 1, asegurando que la suma de estos pesos en un conjunto dado sea 1.Su definición matemática es: para un vector *s* de dimensión *N.*

### Multi-Layer Perceptron (MLP con GELU)

Después de la operación de atención, los tokens son procesados a través de un Perceptrón Multicapa (MLP). Este MLP consta de dos capas lineales con una función de activación GELU entre ellas. La fórmula matemática de la capa MLP se puede describir como:

Donde:

* es la entrada al MLP, es la normalización de capa y es el resultado después de la operación de atención multi-cabeza.
* es la primera capa lineal. es la matriz de pesos y es el vector de sesgos para la primera capa.
* es la salida de la primera capa lineal pasada a través de la función de activación *GELU*. *GELU* es una función de activación que introduce una no linealidad que permite al modelo aprender representaciones más complejas de los datos. Está definida matemáticamente como .
* es la salida de la segunda capa lineal. es la matriz de pesos y es el vector de sesgos para la segunda capa.
* *Dropout* es la capa que se aplica a la primera y segunda capa. Dropout es una técnica de regularización donde, durante el entrenamiento, se desactivan aleatoriamente algunas neuronas. Esto significa que en cada paso de entrenamiento, ciertas neuronas no se actualizan. Esta técnica ayuda a evitar el sobreajuste y asegura que cada neurona desempeñe un papel en el aprendizaje.

### Autoatención Basada en Ventanas Desplazadas (SW-MSA)

Para mitigar las limitaciones de *W-MSA*, la capa *SW-MSA* desplaza las ventanas para introducir conexiones entre ventanas diferentes. Cuando se aplica el módulo de atención en ventanas desplazadas, se calcula de manera similar, pero con una máscara adicional para asegurar que solo se consideren los elementos dentro de la ventana desplazada.

Es aquí donde cambia el valor de *B* debido al desplazamiento.

### Conexión Residual

Las conexiones residuales se aplican después de cada módulo, después de la capa de atención (W-MSA) y después de la capa MLP, en un bloque Swin Transformer. En las ecuaciones, la conexión residual se denota como el término  *o* .

### Ventanas desplazadas en bloques sucesivos

Las siguientes ecuaciones muestran cómo se calculan las características en bloques sucesivos utilizando diferentes configuraciones de atención basadas en ventanas (Liu et al., 2021).

**,**

**,**

,

Donde y son las características de salida de los módulos W-MSA y MLP, es decir tensores resultantes después de aplicar las operaciones de atención y *MLP* para el bloque , respectivamente.

### Patch Merging

En la arquitectura Swin Transformer, la resolución de la imagen se reduce en etapas sucesivas, desde la "Etapa 1" hasta la "Etapa 4". En cada etapa, el modelo procesa las características de la imagen a diferentes resoluciones para capturar información jerárquica. Para realizar este proceso, Swin Transformer emplea un método denominado "Patch Merging", este método se aplica para reducir el número de tokens (parches) y, por lo tanto, la resolución de la imagen. Se toma un grupo de parches vecinos de y se concatenan sus características en profundidad. Matemáticamente, este proceso se puede expresar de la siguiente manera:

Donde es la operación de concatenación a lo largo del eje de las características. Dado que cada parche tiene una dimensión , la dimensión del parche fusionado será 4C. Después de la concatenación, una capa lineal se aplica al parche fusionado para reducir su dimensión a la mitad (2C):

Donde y son la matriz de pesos y el vector de sesgo de la capa lineal, respectivamente. Esta operación reduce el número de tokens a y la dimensión de cada token a . Posteriormente, los tokens reducidos se alimentan en los bloques de Swin Transformer para transformaciones adicionales. Este proceso se puede repetir en etapas sucesivas para reducciones adicionales en la resolución y el número de tokens, permitiendo que el modelo capture características a múltiples escalas.

## Métricas de evaluación

Antes de profundizar en dichas métricas, es fundamental clasificar las predicciones del modelo en las siguientes categorías:

* **Verdaderos Positivos (TP):** representa los casos en los que el modelo predice la etiqueta positiva y la etiqueta real es también positiva. Para calcular TP, se cuentan todas las instancias del conjunto de prueba donde ambos, la predicción del modelo y la etiqueta real, son positivas.
* **Verdaderos Negativos (TN):** son los casos en los que el modelo predice la etiqueta negativa y la etiqueta real es también negativa. Para calcular TN, se cuentan todas las instancias del conjunto de prueba donde tanto la predicción del modelo como la etiqueta real son negativas.
* **Falsos Positivos (FP):** ocurren cuando el modelo predice la etiqueta positiva pero la etiqueta real es negativa. Para calcular FP, se cuentan todas las instancias del conjunto de prueba donde la predicción del modelo es positiva y la etiqueta real es negativa.
* **Falsos Negativos (FN):** son los casos donde el modelo predice la etiqueta negativa, pero la etiqueta real es positiva. Para calcular FN, se cuentan todas las instancias del conjunto de pruebas donde la predicción del modelo es negativa y la etiqueta real es positiva.

A partir de estas categorizaciones, se derivan las siguientes métricas:

## Matriz de Confusión

La matriz de confusión constituye una herramienta analítica y visual para la descomposición cuantitativa de predicciones en verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN), facilitando el cálculo de las métricas de evaluación (Susmaga, 2004). Es una matriz entera no negativa , donde y , siendo el número total de clases. El elemento representa el número de objetos de la clase que han sido asignados a la clase durante la prueba. Los elementos en la diagonal principal indican una clasificación correcta, mientras que los elementos fuera de la diagonal representan errores del clasificador. De este modo, la matriz de confusión provee un mecanismo cuantitativo para evaluar la capacidad discriminativa del clasificador respecto a las diferentes clases (Susmaga, 2004).

# METODOLOGÍA

Toda la documentación del proceso desde la construcción del dataset hasta la evaluación y comparación del modelo se llevará a cabo en un cuaderno de Jupyter y se mostrarán visualmente los cambios en el dataset, el entrenamiento del modelo y su evaluación. Con esto se llevará un seguimiento válido del progreso de la documentación del proyecto que ayudará a la reproducibilidad del mismo.

## Fase 1. Diseño y construcción del dataset

El objetivo de esta fase es desarrollar un dataset adecuado para la implementación y validación del modelo de clasificación de enfermedades en plantas de tomate utilizando la arquitectura Swin Transformer.

### Paso 1. Identificación de Requerimientos de Datos

Se llevará a cabo un análisis detallado para identificar los tipos de imágenes requeridas, incluyendo imágenes de plantas de tomate en diferentes estados de salud. Además, se identificarán los metadatos asociados que podrían ser relevantes para el proyecto.

### Paso 2. Obtención del Dataset

Se descargará el dataset plantvillage de Kaggle. Se efectuará una revisión preliminar para evaluar la calidad y la pertinencia de los datos en relación con los requerimientos identificados en el Paso 1.

### Paso 3. Limpieza del Dataset

Se identificarán y eliminarán imágenes duplicadas dentro del dataset para mejorar la calidad eliminando imágenes filtradas en el conjunto de prueba, para esto se utilizará el método de perceptual hash, el cuál crea un hash de cada imagen y los compara para determinar su similitud.

### Paso 4. Balanceo del Dataset

El dataset de PlantVillage presenta una distribución desigual de imágenes entre clases, esta característica podría introducir un sesgo en el rendimiento del modelo, para abordar este problema se realizará un análisis de la distribución de clases y se eliminarán aleatoriamente muestras de las clases sobrerrepresentadas hasta alcanzar el mismo número de elementos que la clase menos representada.

### Paso 5. Preprocesamiento de datos

Se dividirá el dataset en conjuntos de entrenamiento, validación y prueba usando dataframes de la librería pandas.

### Paso 6. Aumento de Datos

Se aplicarán técnicas de aumento de datos al conjunto de datos de entrenamiento, como rotación, desplazamiento, recorte e inversión de imágenes tanto vertical como horizontalmente, esto para incrementar la variabilidad y tamaño del dataset.

## Fase 2: Entrenamiento del modelo

### Paso 1. Definición de requerimientos de Hardware y Software

En cuanto a software se utilizará la plataforma conda la cual ofrece múltiples herramientas para entrenamiento de modelos de visión artificial, también se utilizará la herramienta jupyter notebook para escribir el código mientras se documenta todo el proceso de entrenamiento del modelo, además se utilizarán las librerías pytorch, pillow, imagehash, numpy, matplotlib, os, io, pandas y opencv. En términos de Hardware se emplearán equipos con tarjeta de procesamiento gráfico NVIDIA GTX 1080 Ti el cuál ayuda a paralelizar los cálculos y aumentar el rendimiento y velocidad del entrenamiento.

### Paso 2. Arquitectura del modelo

Se utilizará la implementación de la arquitectura Swin Transformer provista por la librería pytorch.

### Paso 3. Configuración de parámetros

Se utilizará una arquitectura Swin Transformer-L pre entrenado en ImageNet 22K; con dimensión C= 192; número de capas 2, 2, 18, 2; tamaño de imagen 384 ⨉ 384 y 197 Millones de parámetros entrenables. Se empleará un optimizador AdamW para 90 épocas con un programador de tasa de aprendizaje de decaimiento lineal con un calentamiento lineal de 5 épocas. Se usará un tamaño de lote (batch size) de 1024, una tasa de aprendizaje inicial de 0,0001 y un peso de decaimiento de 0,01. El dropout se establecerá en 0.

### Paso 4. Entrenamiento del modelo

Se entrenará el modelo con el dataset de PlantVillage y los hiperparametros previamente configurados. También se utilizará el conjunto de validación para evaluar el rendimiento del modelo durante el entrenamiento. Este proceso permite ajustar el modelo y sus hiperparámetros basándose en el rendimiento observado en el conjunto de validación, ayudando a evitar el sobreajuste al conjunto de entrenamiento.

## Fase 3: Evaluación y Análisis del modelo

### Paso 1. Selección de métricas

Dado que el dataset presenta un balance entre clases, se opta por emplear la métrica de accuracy global para evaluar el rendimiento del modelo Swin Transformer. Adicionalmente, se generará una matriz de confusión para cada clase con el fin de analizar la variabilidad en la clasificación.

### Paso 2. Pruebas de rendimiento

Se aplicará el dataset de prueba para realizar la evaluación. Se utilizará la librería PyTorch para computar la métrica de accuracy global y generar la matriz de confusión.

### Paso 3. Interpretación de resultados

Se utilizará la técnica de mapas de calor de atención para identificar las regiones de las imágenes que el modelo considera más informativas durante la fase de predicción.

## Fase 4: Comparativa con otros modelos

### Paso 1. Entrenamiento de modelos de CNN para comparación

Se entrenará modelos basados en arquitecturas CNN como ResNet, DenseNet y EfficientNet con los mismos hiperparametros y el mismo dataset para llevar a cabo la comparación con el modelo Swin Transformer.

### Paso 2. Evaluación de modelos

Se comparará el Accuracy entre el modelo basado en Swin Transformer y los modelos basados en CNN para determinar cual modelo tuvo mayor eficacia en el problema de la clasificación de enfermedades del tomate.

1. RESULTADOS

# REFERENCIAS

Collins, E.J.; Bowyer, C.; Tsouza, A.; Chopra, M. Tomatoes: An Extensive Review of the Associated Health Impacts of Tomatoes and Factors That Can Affect Their Cultivation. Biology 2022, 11, 239.

Agarwal, Mohit, et al. “ToLeD: Tomato Leaf Disease Detection Using Convolution Neural Network.” *Procedia Computer Science*, vol. 167, no. 293–301, 2020, pp. 293–301, https://doi.org/10.1016/j.procs.2020.03.225.

Abbas, A., Jain, S., Gour, M., & Vankudothu, S. (2021). Tomato plant disease detection using transfer learning with C-GAN synthetic images. Computers and Electronics in Agriculture, 187(106279). https://doi.org/10.1016/j.compag.2021.106279

Chen, H.-C., Widodo, A. M., Wisnujati, A., Rahaman, M., Lin, J. C.-W., Chen, L., & Weng, C.-E. (2022). AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf. Electronics, 11(6), 951. https://doi.org/10.3390/electronics11060951

Bhandari, M., Shahi, T. B., Neupane, A., & Walsh, K. B. (2023). BotanicX-AI: Identification of Tomato Leaf Diseases Using an Explanation-Driven Deep-Learning Model. Journal of Imaging, 9(2), 53. https://doi.org/10.3390/jimaging9020053

FAO. (2021). Crops and livestock products. Www.fao.org. https://www.fao.org/faostat/en/#data/QCL

Caruso, A. G., Bertacca, S., Parrella, G., Rizzo, R., Davino, S., & Panno, S. (2022). Tomato brown rugose fruit virus: A pathogen that is changing the tomato production worldwide. Annals of Applied Biology, 181(3), 258–274.

Panno, S., Davino, S., Caruso, A. G., Bertacca, S., Crnogorac, A., Mandić, A., Noris, E., & Matić, S. (2021). A Review of the Most Common and Economically Important Diseases That Undermine the Cultivation of Tomato Crop in the Mediterranean Basin. Agronomy, 11(11), 2188. https://doi.org/10.3390/agronomy11112188

Singh, V. K., Singh, A. K., & Kumar, A. (2017). Disease management of tomato through PGPB: current trends and future perspective. 3 Biotech, 7(4). https://doi.org/10.1007/s13205-017-0896-1

Zahangir Alom, Taha, T. M., Yakopcic, C. G., Westberg, S., Sidike, P., Nasrin, Mst Shamima, Esesn, V., Abdul, & Asari, V. K. (2018). The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. ArXiv E-Prints, 2. https://doi.org/10.48550/arxiv.1803.01164

Boulent, J., Foucher, S., Théau, J., & St-Charles, P.-L. (2019). Convolutional Neural Networks for the Automatic Identification of Plant Diseases. Frontiers in Plant Science, 10. https://doi.org/10.3389/fpls.2019.00941

Li, E. Y. (2020, August 1). 10 Papers You Should Read to Understand Image Classification in the Deep Learning Era. Medium. https://towardsdatascience.com/10-papers-you-should-read-to-understand-image-classification-in-the-deep-learning-era-4b9d792f45a7

Rawat, V., Singh, N., Kaur, B., & Bora, S. (2022). Tomato Leaf Disease Classification Using Supervised Learning Techniques: Contrasting Analysis. 2022 International Conference on Advances in Computing, Communication and Materials (ICACCM), 1(2642-7354). https://doi.org/10.1109/icaccm56405.2022.10009617

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021, pp. 9992-10002, doi: 10.1109/ICCV48922.2021.00986.

Boulent, J., Foucher, S., Théau, J., & St-Charles, P. L. (2019). Convolutional neural networks for the automatic identification of plant diseases. Frontiers in plant science, 10, 941.

Gadade, H. D., & Kirange, D. D. (2020). Machine learning approach towards tomato leaf disease classification. International Journal of Advanced Trends in Computer Science and Engineering, 9(1), 490-495.

Abbas, A., Jain, S., Gour, M., & Vankudothu, S. (2021). Tomato plant disease detection using transfer learning with C-GAN synthetic images. Computers and Electronics in Agriculture, 187, 106279.

Uppada, R., & Kumar, D. R. (2023). Computer-aided fusion-based neural network in application to categorize tomato plants.Springer-Verlag London Ltd., part of Springer Nature 2023, 1-9.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021, July). Training data-efficient image transformers & distillation through attention. In International conference on machine learning (pp. 10347-10357). PMLR.

Flores, C. (2012). Enfermedades de tomate: Guía de consulta (S. Buono & S. Giorgioni, Eds.; 1st ed.). Ediciones Instituto Nacional De Tecnología Agropecuaria De Argentina INTA.

Bernal, R. (2010). Enfermedades de tomate (lycopersicum esculentum mill.) en invernadero en las zonas de salto y bella unión. Serie Técnica. INIA. Montevideo, Editorial Hemisferio Sur SRL, 181, 1–71.

Thangaraj, R., Anandamurugan, S., Pandiyan, P., & Kaliappan, Vishnu Kumar. (2022). Artificial intelligence in tomato leaf disease detection: a comprehensive review and discussion. *Journal of Plant Diseases and Protection*, *129*, 3.

Garcia, J. (2016). A review on the main challenges in automatic plant disease identification based on visible range images. Biosystems Engineering, 144, 52–60.

Valdivieso. (2021). Estudio comparativo de las técnicas de inteligencia artificial para el diagnóstico de enfermedades en la agricultura.

Blancard, D. (2011). Enfermedades del tomate. Mundi-Prensa Libros.

Verma, S., Chug, A., & Singh, A. P. (2018). Prediction models for identification and diagnosis of tomato plant diseases. IEEE, 1557–1563.

Corrales Paredes, J. (2021). *Uso de depredadores, parasitoides y entomopatógenos para el control biológico de plagas y enfermedades de hortalizas con énfasis en tomate (Solanum lycopersicum l.)* (p. 105) [MSc Thesis].

Gómez-Camperos, J., Jaramillo, H., & Guerrero-Gómez, G. (2022). Técnicas de procesamiento digital de imágenes para detección de plagas y enfermedades en cultivos: una revisión. Ingeniería Y Competitividad, 24, 1.

Guennouni, M., Admou, B., Bourrhouat, A., El Khoudri, Noureddine, Zkhiri, W., Talha, I., Hazime, R., & Hilali, A. (2022). Knowledge and practices of food safety among health care professionals and handlers working in the kitchen of a moroccan university hospital. Journal of Food Protection, 85, 4.

Uçar, A., Yilmaz, M. V., & Çakiroglu, Funda Pinar. (2016). Food safety–problems and solutions. Significance, Prevention and Control of Food Related Diseases, 3.

Nanehkaran, Y., Zhang, D., Chen, J., Tian, Y., & Al-Nabhan, N. (2020). Recognition of plant leaf diseases based on computer vision. In Journal of Ambient Intelligence and Humanized Computing (pp. 1–18). Springer.

Khan, R., Kumar, S., Dhingra, N., & Bhati, N. (2021). The use of different image recognition techniques in food safety: a study. Journal of Food Quality, 2021, 1–10.

Mahadevkar, S. V., Khemani, B., Patil, S., Kotecha, K., Vora, D. R., Abraham, A., & Gabralla, L. A. (2022). A Review on Machine Learning Styles in Computer Vision—Techniques and Future Directions. IEEE Access, 10, 107293–107329. https://doi.org/10.1109/access.2022.3209825

Sood, S., & Singh, H. (2021). Computer vision and machine learning based approaches for food security: A review. Multimedia Tools and Applications, 80, 18.

Jamil, S., Md. Jalil Piran, & Kwon, O.-J. (2023). A Comprehensive Survey of Transformers for Computer Vision. Drones, 7(5), 287–287. https://doi.org/10.3390/drones7050287

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1). https://doi.org/10.1186/s40537-019-0197-0

Xiao, Z., Wang, J., Han, L., Guo, S., & Cui, Q. (2022). Application of machine vision system in food detection. *Frontiers in Nutrition*, *9*, 888245.

Papers With Code. (2023). *Papers with Code - ImageNet Benchmark (Image Classification)*. Paperswithcode.com. https://paperswithcode.com/sota/image-classification-on-imagenet. (Accessed on September 2023).

Xu, S., Wang, J., Shou, W., Ngo, T., Sadick, A.-M., & Wang, X. (2021). Computer vision techniques in construction: a critical review. *Archives of Computational Methods in Engineering*, *28*, 3383–3397.

Ouhami, M., Hafiane, A., Es-Saady, Y., El Hajji, Mohamed, & Canals, R. (2021). Computer vision, IoT and data fusion for crop disease detection using machine learning: A survey and ongoing research. *Remote Sensing*, *13*, 13.

Szeliski, R. (2020). *Computer Vision: Algorithms and applications.* (2nd ed.). Springer Nature.

Vocaturo, E. (2021). Image classification techniques. In *Handbook of Research on Disease Prediction through Data Analytics and Machine Learning* (pp. 22–49). IGI Global.

Sanghvi, K., Aralkar, A., Sanghvi, S., & Saha, I. (2020). A survey on image classification techniques. In *Available at SSRN 3754116*.

Shah, U., & Harpale, A. (2018). A review of deep learning models for computer vision. In *2018 IEEE Punecon* (pp. 1–6). IEEE.

Amjoud, A. B., & Amrouch, M. (2023). Object detection using deep learning, CNN and vision transformers: A review. *IEEE Access*, *11*(35479-35516).

Cappi, C., Chapdelaine, C., Gardes, L., Jenn, E., Lefevre, B., Picard, S., & Soumarmon, T. (2021). Dataset definition standard (dds). In *arXiv preprint arXiv:2101.03020*.

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using Deep Learning for Image-Based Plant Disease Detection. *Frontiers in Plant Science*, *7*. https://doi.org/10.3389/fpls.2016.01419

Khosla, C., & Saini, B. S. (2020). Enhancing performance of deep learning models with different data augmentation techniques: A survey. *IEEE*, 79–85.

Yang, S., Xiao, W., Zhang, M., Guo, S., Zhao, J., & Shen, F. (2022). Image data augmentation for deep learning: A survey. In *arXiv preprint arXiv:2204.08610*.

Dyk, V., & Meng, X.-L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, *10*, 1.

Flach, P. A., & Kull, M. (2015, December). Precision-recall-gain curves: PR analysis done right. *NIPS*. https://api.semanticscholar.org/CorpusID:937625

Susmaga, R. (2004). Confusion matrix visualization. In A. Kłopotek, Mieczysław, T. Wierzchoń, Sławomir, & K. Trojanowski (Eds.), *Springer, Berlin, Heidelberg* (pp. 107–116). Springer Berlin Heidelberg.

Ulutan, O., Rallapalli, S., Srivatsa, M., Torres, C., & Manjunath, B. (2020). Actor conditioned attention maps for video action detection. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 527–536.