ISSN 3085-9131 / e-ISSN 1981-1004

Embrapa Uva e Vinho

Boletim de Pesquisa e Desenvolvimento 41

Bento Gonçalves, RS / Dezembro, 2025

Proposta de modelos de aprendizado profundo para diagnóstico de doenças foliares da videira

Fabio Rossi Cavalcanti[1]

[1] Pesquisador, Embrapa Uva e Vinho, Bento Gonçalves, RS.

Resumo – Este trabalho teve como objetivo desenvolver e aplicar modelos de aprendizado profundo com o *Tensorflow* para a classificação automática de sintomas de doenças foliares em videiras, com o uso de redes neurais convolucionais (CNNs). A base de dados foi composta por imagens rotuladas em três classes: a) folhas sadias; b) folhas com manchas de óleo (míldio); e c) folhas com manchas-das-folhas (*Mycosphaerella*). Inicialmente, foi utilizado um modelo sequencial simples contendo camadas convolucionais, *pooling* e densas. Em seguida, aplicou-se a técnica de transferência de aprendizado para a arquitetura *InceptionV3* (*Tensorflow/keras*), pré-treinada com os pesos do ImageNet. Técnicas de regularização como *data augmentation* e *dropout* foram utilizadas para melhorar a generalização dos modelos. Após o treinamento, os modelos foram convertidos para o formato *Tensorflow Lite* (TFLite), visando viabilizar sua execução em dispositivos com baixa capacidade computacional. A acurácia nos dados de validação indicou bom desempenho, com potencial de uso em diagnósticos assistidos por computador em campo. Para permitir a aplicação prática, desenvolveu-se uma interface com o usuário em *Flask* (*Python*) controlada por uma <html> simples, onde o sistema recebe uma imagem foliar do usuário e exibe a classe predita. A proposta alia inteligência artificial e usabilidade, com aplicabilidade em agricultura de precisão, oferecendo uma ferramenta eficiente para o monitoramento fitossanitário em tempo real.

Termos para indexação: classificação de imagens, redes neurais convolucionais, transferência de aprendizado, sistemas especialistas.

Embrapa Uva e Vinho
Rua Livramento, n° 515
Caixa Postal 130
95701-008 Bento Gonçalves, RS
www.embrapa.br/uva-e-vinho
www.embrapa.br/fale-conosco/sac

Revisão de texto
Renata Gava

Normalização bibliográfica
Rochelle Martins Alvorcem
(CRB-10/1810)

Projeto gráfico
Leandro Sousa Fazio

Diagramação
Renata Gava

Publicação digital: PDF

Deep learning approaches for grapevine leaf diseases diagnosis

Abstract – This study aimed to develop and apply deep learning models for the automatic classification of foliar disease symptoms in grapevines using convolutional neural networks (CNNs). The dataset consisted of labeled images across three classes: a) healthy leaves; b) Downy mildew symptoms; and c) *Mycosphaerella* leaf blight. An initial sequential model was implemented with convolutional, pooling, and dense layers. Subsequently, transfer learning was applied using the InceptionV3 architecture pre-trained on ImageNet, to which a custom classification head was appended. Regularization strategies such as data augmentation and dropout were used to improve model generalization. After training, the models were converted

to the Tensorflow Lite (TFLite) format, enabling deployment on low-power edge devices. Validation accuracy demonstrated satisfactory performance, suggesting the approach's viability for assisted diagnosis in field conditions. To ensure user-friendly interaction, a Flask-based web application (with an <html> controller) was developed, allowing users to user upload leaf images and receive real-time classification predictions. The proposed solution integrates artificial intelligence with practical usability and demonstrates significant potential for precision agriculture. It offers a robust and accessible tool for phytosanitary monitoring, helping to improve disease management decisions and reduce crop losses through early and accurate detection of foliar diseases in vineyards.

Index terms: deep learning, image classification, convolutional neural networks, transfer learning, expert systems.

Introdução

A detecção precoce de doenças em plantas agrícolas é um desafio recorrente para a produtividade e sustentabilidade na agricultura. Os métodos tradicionais de diagnóstico, baseados na inspeção visual e na experiência de campo, são limitados por fatores como subjetividade, variabilidade regional e tempo de resposta. Nesse contexto, abordagens com inteligência artificial, especialmente Aprendizado de Máquina (*Machine Learning,* ML) e Aprendizado Profundo (*Deep Learning,* DL), têm demonstrado grande potencial para automatizar e melhorar a precisão do diagnóstico de doenças foliares. A literatura científica recente tem se concentrado em modelos computacionais capazes de identificar padrões sintomáticos a partir de imagens de folhas, com o objetivo de construir sistemas diagnósticos robustos e acessíveis. Um estudo influente nesse campo é o de Ferentinos (2018), que treinou redes neurais convolucionais (CNNs) com imagens de 25 culturas agrícolas afetadas por 58 doenças distintas, alcançando mais de 99% de acurácia. Esse desempenho demonstra a superioridade dos modelos DL em tarefas de classificação multiclasse baseadas em sintomatologia visual. Complementando essa perspectiva, Ahmad et al. (2023) realizaram uma revisão sistemática sobre o uso de DL no diagnóstico vegetal, destacando a importância de bases de dados bem rotuladas, o desafio da variação morfológica das folhas e a necessidade de modelos que levem em conta a progressão temporal dos sintomas. Em linha semelhante, Saleem et al. (2019) apontam que CNNs são eficazes justamente por não dependerem de extração manual de atributos, aprendendo, diretamente dos dados visuais, características de cor, textura e forma típicas de doenças foliares. Além da acurácia, a aplicabilidade prática dos modelos tem sido um foco relevante. Ahmed e Reddy (2021) propuseram um sistema diagnóstico móvel baseado em DL que funciona em *smartphones*, visando agricultores de regiões com pouca infraestrutura. Assim, os autores desenvolveram um sistema em tempo real para diagnóstico de doenças em folhas de macieiras, com desempenho equivalente a soluções em nuvem. Essas abordagens têm ampliado o alcance da inteligência artificial, tornando-a acessível a pequenos produtores. Na mesma linha, Owomugisha e Mwebaze (2016) aplicaram ML para estimar a severidade de doenças com base em imagens captadas em regiões rurais da África, reforçando a utilidade social dessas ferramentas em contextos de baixo custo e conectividade limitada.

Por outro lado, técnicas clássicas de ML ainda apresentam vantagens em cenários com pouca disponibilidade de dados. Estudos conduzidos por Ahmed e Yadav (2023a) mostraram que algoritmos como SVM, KNN e árvores de decisão podem superar redes profundas em contextos com desequilíbrio de classes e menor volume de imagens. Em uma análise complementar, os mesmos autores (Ahmed e Yadav, 2023b) destacam que abordagens híbridas, combinando ML com técnicas de pré-processamento e seleção de atributos, ainda são altamente eficazes. A importância de soluções híbridas também é evidenciada por Wani et al. (2022), que organizaram metodologias integradas com base em sensores, classificação visual e padrões ambientais para aumentar a robustez dos diagnósticos.

Apesar dos avanços, ainda há desafios substanciais. Jackulin e Murugavalli (2022) apontam os riscos de sobreajuste (*overfitting*) e viés amostral causados por bancos de dados construídos artificialmente. Eles defendem a necessidade de dados reais, coletados em diferentes regiões agrícolas, estações do ano e estágios fenológicos da planta, como base para a construção de modelos mais generalizáveis. O futuro dos sistemas diagnósticos vegetais parece apontar para a integração com sensores Internet das Coisas (*Internet of Things*, IoT), dados climáticos e informações geoespaciais, de forma a produzir diagnósticos mais precisos, contextuais e acionáveis. Esses avanços não apenas aumentam a eficiência da produção agrícola, como também promovem sustentabilidade e redução no uso de agroquímicos, ao permitir o manejo racional e precoce das doenças (Cavalcanti, 2025).

Dessa forma, a literatura recente comprova que a Inteligência Artificial (IA) aplicada à Fitopatologia, especialmente por meio da análise sintomática de folhas, representa uma das fronteiras mais promissoras da agricultura digital. Os modelos de DL demonstram acurácia superior em classificação de sintomas visuais, mas seu sucesso depende fortemente da qualidade e da diversidade dos dados de entrada. Em paralelo, técnicas de ML tradicional ainda ocupam um espaço relevante em contextos de baixa disponibilidade de dados (Ferentinos, 2018). O desenvolvimento de sistemas portáteis, interpretáveis e de uso em campo tem sido uma estratégia para democratizar o acesso à tecnologia e viabilizar seu uso prático por agricultores. Com isso, a convergência entre agronomia, ciência de dados e computação promete transformar a forma como doenças de plantas serão identificadas, monitoradas e controladas nas próximas décadas.

O objetivo do presente trabalho foi desenvolver e comparar modelos de aprendizado profundo supervisionado para classificação de imagens de folhas de videira, tendo como modelo de estudo a discriminação entre sintomas de míldio e manchas-das-folhas, e folhas sadias assintomáticas. A partir da validação dos modelos em um conjunto de dados rotulado, buscou-se construir uma ferramenta inteligente capaz de identificar, com alta acurácia, padrões visuais de diferentes classes fitopatológicas em condições reais de campo. Neste trabalho, o modelo de classificação foi construído por meio do *framework Tensorflow* (com *Keras*), utilizando arquiteturas convolucionais do tipo *Sequential* e, em etapas mais avançadas, a técnica de transferência de aprendizado com a arquitetura pré-treinada *InceptionV3*. O *Tensorflow* é um *framework* de código aberto amplamente utilizado para aprendizado profundo, que oferece uma estrutura robusta para construção, treinamento e exportação de modelos neurais, incluindo suporte à conversão para o formato otimizado *Tensorflow Lite* voltado a aplicações embarcadas e web.

Os modelos de DL podem ser expandidos se treinados com base de dados mais ampla, substituindo a dependência da apreciação humana em sistemas especialistas de geração anterior, baseados em chaves dicotômicas e decisões de usuário.

Material e métodos

Fontes de dados

Inicialmente, um caminho de acesso aos dados foi preparado para realização de uma inspeção inicial do conjunto de imagens para o treinamento do modelo. Diferentes repositórios de imagens disponíveis ao público foram utilizados para busca de dados (imagens) para esse treinamento (Khan, 2025; Mandal, 2025; Redape, 2025).

Em um notebook .ipynb, bibliotecas pathlib e PIL foram utilizadas para inspeção inicial das imagens dos *datasets* das bases de dados públicas. Em seguida, foi feito um levantamento quantitativo das imagens com compressão .jpg, o que permitiu verificar rapidamente o volume de dados disponíveis para o treinamento de modelos. As imagens com sintomas foliares típicos de doenças da videira foram fixadas nas dimensões 256 x 256 x 24 Bpp. Para organizar, foram identificadas subpastas como classes-alvo do problema de classificação supervisionada, que foram três diferentes classes de sintomas: 1) folhas sadias; 2) folhas com míldio (*Plasmopara viticola*); e 3) mancha-das-folhas (*Mycosphaerella personata*). As coleções contaram com imagens de folhas dotando sintomas em diferentes estágios e diferentes percentuais de área foliar lesionada.

Montagem e treinamento da Rede Neural (camadas de *perceptrons*)

Para preparar os pacotes de treinamento e teste (validação) foram utilizadas a biblioteca *NumPy* e a funcionalidade *glob* de *pathlib* para percorrer todas as subpastas (classes de doenças), verificando a quantidade de imagens em cada classe. Cada classe foi organizada justamente com fotos padronizadas. Para cada classe identificada, foi carregada a primeira imagem usando a biblioteca PIL e convertida em um *array NumPy*. Em seguida, foram extraídas e impressas as dimensões da imagem, indicando altura, largura e número de canais (normalmente três, correspondendo a RGB).

Na sequência, foi realizada a preparação dos dados para treinamento e validação de uma rede neural. Foram definidos o *batch_size* (tamanho do lote de imagens processadas em cada passo do treinamento) e as dimensões das imagens (altura e largura) para redimensionamento padronizado. O *batch_size* teve de ser ajustado para trabalho com uma GPU arquitetura Turing com 896 núcleos CUDA e 4 GB de memória GDDR5. Também, foi ativado o gerenciamento de crescimento de memória da GPU (via tf.config.experimental.set_memory_growth) — uma medida preventiva para evitar alocação antecipada de toda a memória da GPU, melhorando a compatibilidade com múltiplos processos. Por fim, os conjuntos de dados de treino e validação foram carregados utilizando a função image_dataset_from_directory do pacote de *Deep*

Learning, *Tensorflow*, com divisão de 80/20 e *seed* fixa (568) para reprodutibilidade.

Após a fixação da "semente de dados" (seed(42)) do *TensorFlow*, foi definida uma arquitetura de rede neural sequencial utilizando a API *Keras* do *TensorFlow*. O modelo foi ajustado com a camada de entrada com forma (256, 256, 3), representando imagens coloridas (RGB) de 256 x 256 pixels padronizadas para os ensaios de montagem do modelo. Na sequência, uma camada *Rescaling* normalizou os valores dos pixels da imagem, convertendo-os de um intervalo de 0–255 para 0–1, o que é uma prática comum para estabilizar o treinamento. A camada *Flatten* transforma a imagem bidimensional em um vetor unidimensional, necessário para conectar às camadas densas. A rede de *perceptrons* incluiu uma camada Dense com 128 neurônios e função de ativação ReLU para capturar padrões não-lineares, seguida por uma camada de saída com quatro neurônios e função de ativação softmax, que retorna uma distribuição de probabilidades entre as quatro classes.

Treinamento do modelo

Após a verificação da GPU, o modelo foi treinado com os dados preparados (treino e validação/teste) durante dez épocas (iteração sobre o conjunto completo de treinamento). O método modelo.fit() inicia o processo de aprendizado (ajuste), em que a rede ajusta seus pesos por meio da retropropagação para minimizar o erro entre as predições e os rótulos reais.

Adicionando camadas convolucionais e parada antecipada

Nesta etapa da sequência do desenvolvimento do *pipeline* (sequência de tarefas de modelagens em DL), um modelo sequencial de rede neural convolucional (CNN) foi definido utilizando a API tf.keras.models.Sequential. A arquitetura começa com uma camada de entrada que espera imagens com forma (256, 256, 3) (altura, largura e canais RGB). Em seguida, aplica-se uma normalização das imagens com *Rescaling* (1./255) para normalização dos pixels, o que transforma os valores de 0–255 para 0–1. O modelo foi construído com duas camadas convolucionais Conv2D, com 32 filtros e janelas de convolução de 3 x 3, cada uma seguida por uma camada de *pooling* MaxPooling2D que reduz a dimensionalidade espacial das representações. Após isso, os mapas de ativação foram achatados (*Flatten*) e submetidos a uma camada densa com 128 neurônios (*perceptrons*) e ativação ReLU. Finalmente, a camada de saída foi reduzida a quatro neurônios com ativação *softmax*, apropriada para classificação multiclasse em quatro categorias, que foi o critério do presente estudo.

Depois, o modelo foi compilado com o otimizador *Adam*, uma função de perda do tipo *sparse_categorical_crossentropy* — sugerida para problemas de classificação com rótulos inteiros (Cholet, 2021) — e com a métrica de acurácia. Em seguida, o modelo foi treinado por meio do método *fit*(), utilizando o conjunto de dados treino para o aprendizado e validação para avaliação, novamente, por dez épocas. Depois do desenvolvimento do *pipeline*, foi definida uma classe personalizada de *callback* (chamada antecipada) que herdou uma instrução do Keras, 'tf.keras.callbacks.Callback'. Nessa classe, foi escrita uma função para monitorar a acurácia ao final de cada época de treinamento de tal modo que, se a acurácia registrada no treinamento atingisse ou superasse 95%, uma mensagem seria exibida antes de um atributo de parada, interrompendo o treinamento antecipadamente. Isso, para poupar tempo e recursos computacionais.

Na sequência, o modelo convolucional foi novamente definido, incluindo camadas de normalização, convolução, *pooling*, *flattening* e densas, seguindo o modelo anterior. O modelo foi também compilado novamente com o otimizador *Adam*, função de perda 'sparse_categorical_crossentropy' e métrica de acurácia. Nessa etapa do treinamento é que o *callback* personalizado foi passado para chamar a classe e efetuar a parada.

Aprimorando o desempenho do modelo inicial

De posse de uma função de parada, o modelo de rede neural convolucional foi reconstruído utilizando a API *Sequential do Tensorflow*/*Keras*. A entrada foi novamente configurada para imagens com dimensões (256, 256, 3), RGB. Em seguida, uma camada de *Rescaling* foi aplicada para normalizar os valores dos pixels no intervalo [0, 1]. A estrutura da rede incluiu duas camadas convolucionais com 32 filtros cada, utilizando janelas (*kernels*) de 3 x 3 e função de ativação ReLU, seguidas de camadas de *MaxPooling2D* com tamanho 2 x 2 para redução da dimensionalidade espacial das ativações. A redução da dimensionalidade espacial das ativações refere-se à diminuição do tamanho das representações intermediárias (ou "mapas de ativação") que são geradas pelas camadas convolucionais em uma rede neural.

Após essas etapas, a saída é achatada com novo *flattening* e passada por uma camada densa com 128 neurônios ativados por ReLU, culminando em uma camada *Dense* final com quatro saídas

e ativação softmax, responsável por classificar a imagem em uma das quatro categorias estudadas para este trabalho. O modelo foi compilado com o otimizador *Adam* e demais indicadores de métrica e acurácia dos modelos anteriores. O treinamento foi iniciado com 50 épocas definidas, utilizando os conjuntos treino e validação (teste) previamente definidos.

Em seguida, o *pipeline* foi incrementado com o treinamento da rede neural convolucional (CNN) com uma função de *data augmentation* embutido. Para isso, definiu-se uma sequência de transformações aleatórias sobre as imagens de entrada utilizando o objeto *data_augmentation*. Essas transformações incluem espelhamento horizontal (*RandomFlip*), rotação aleatória de até 5% (*RandomRotation*) e zoom aleatório de até 5% (*RandomZoom*). Essa camada foi inserida no início da arquitetura, de modo que cada imagem passada à rede durante o treinamento sofra pequenas variações, aumentando a robustez do modelo. Na sequência, a arquitetura da CNN foi reconstruída. Após o *data_augmentation*, as imagens foram reescaladas para o intervalo [0, 1], o que contribui para a estabilização do treinamento ao evitar valores extremos nos cálculos dos gradientes. Essa normalização, aliada ao uso de rótulos inteiros para representar as classes, foram adequados para tarefas de classificação multiclasse em aprendizado profundo, pois permite o uso eficiente da função de perda *sparse_categorical_crossentropy*, que espera rótulos no formato inteiro em vez de codificação one-hot.

Integração do classificador desenvolvido com o Modelo Pré-Treinado *InceptionV3* mediante transferência de aprendizado

Nesse ponto do *pipeline*, foi implementada uma abordagem de transferência de aprendizado (*transfer learning*), na qual se reaproveita uma rede neural convolucional pré-treinada, para o presente estudo, a *InceptionV3* no *Keras*, como base para um novo modelo adaptado à tarefa de classificação de imagens de folhas treinada no modelo personalizado desenvolvido. Para isso, o modelo DL treinado anteriormente foi salvo em um arquivo .h5. Em seguida, a *InceptionV3* foi carregada com os pesos da base de dados da ImageNet, sem suas camadas densas finais (include_top = False), com sua capacidade de aprendizado desabilitada (trainable = False). Isso foi feito para congelar os pesos e preservar os conhecimentos previamente adquiridos nos ajustes de modelo anteriores pelo *Keras*.

Em seguida, o *pipeline* de dados de treino e validação foi ajustado por meio de um mapeamento que aplica a normalização das imagens com *Rescaling* (1./255) para garantir compatibilidade com a entrada esperada pela rede. A arquitetura foi então expandida com camadas adicionais a partir da saída da camada intermediária 'mixed7' da *InceptionV3*. Essa saída foi achatada com *Flatten*, passada por uma camada densa com 1.024 unidades e ativação ReLU, seguida de um *Dropout* com taxa de 20% (para reduzir *overfitting*), e finalizada por uma camada *Dense* com quatro unidades e ativação *softmax* para classificação multiclasse. Mantendo o padrão, o novo modelo foi compilado e treinado por 20 épocas com otimizador *Adam* e função de perda categórica apropriada a rótulos inteiros (Figura 1).

Salvando o modelo para uso em aplicações

Como resultado final do *pipeline* de modelagens para encontrar o modelo final, o mesmo foi salvo em formato .h5 otimizado e completo, ideal para continuar o treinamento posteriormente, pois contém arquitetura, pesos e estado do otimizador. O modelo também foi salvo em formato sem o otimizador, sendo mais leve e indicado quando o modelo será apenas usado para inferência (predição), como em APIs ou apps. Também foi possível salvar o modelo no formato 'pesos.weights.h5' com somente os pesos dos neurônios, útil quando se quer carregar os pesos em outra arquitetura compatível já definida em código. Esses formatos são aplicáveis em deploys locais, nuvem ou integração em *pipelines* de MLOps.

Finalmente, o modelo foi convertido do formato *Keras* para o formato *Tensorflow Lite* (*TFLite*) usando a API *TFLiteConverter*. Essa ação abre possibilidade para implantação em dispositivos embarcados ou móveis, e para um .app simples proposto neste mesmo estudo para usuários experimentarem imagens. A conversão aplica quantização dos pesos para 16 bits, em ponto flutuante (float16), reduzindo o tamanho do modelo e melhorando a eficiência computacional. O salvamento do modelo em formato *TFLite* visa permitir implantações leves e eficientes em ambientes com recursos limitados, como dispositivos móveis, embarcados ou IoT. A conversão reduz o tamanho do modelo e otimiza sua velocidade de inferência. Isso é ideal para aplicativos baseados em sensores agrícolas com microcontroladores, ou câmeras com inteligência embarcada. Além disso, o *TFLite* permite uso offline, sem depender de servidores externos.

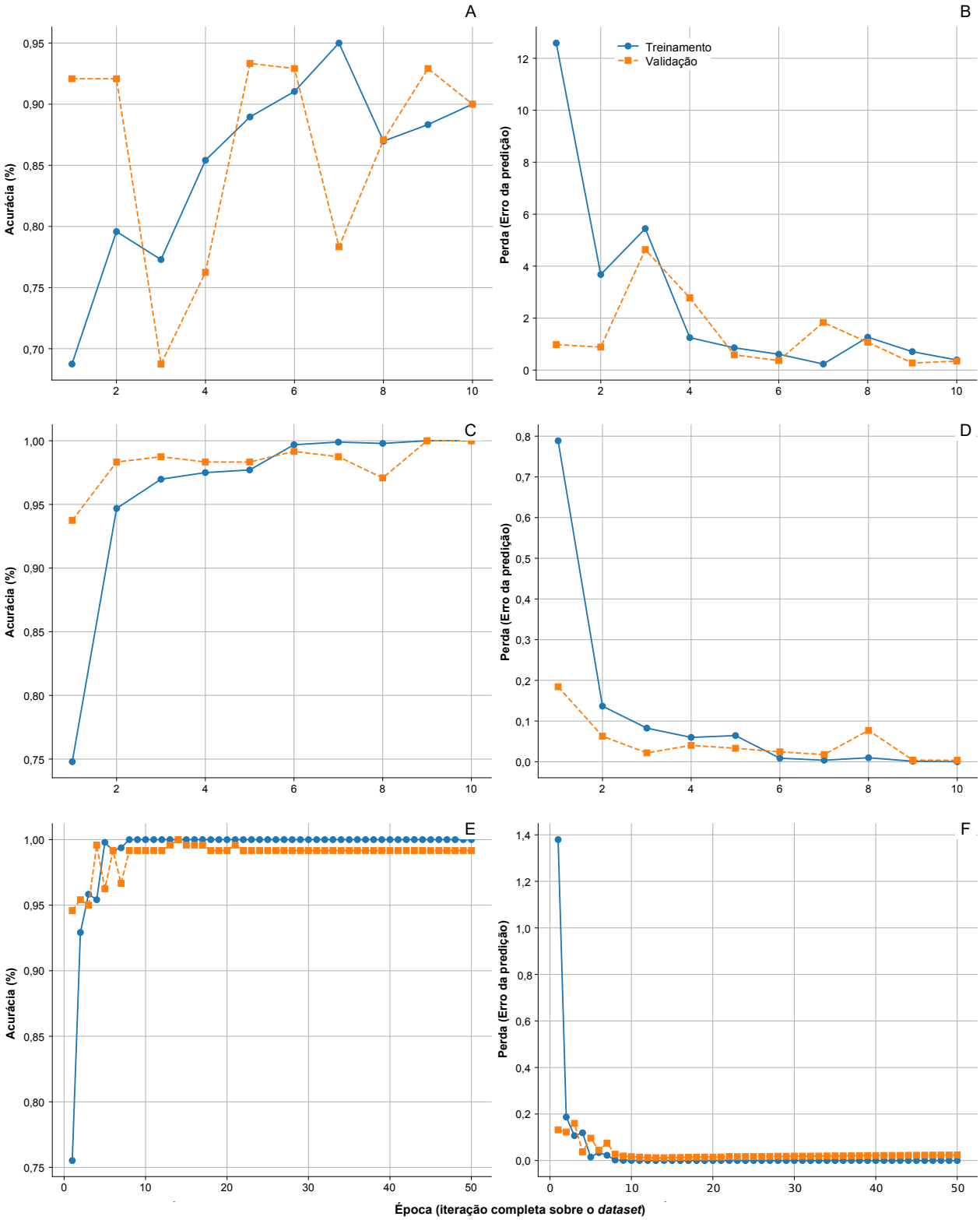

Figura 1. Sequência de perfis de métricas para modelos candidatos do *pipeline* para busca de classificação com o *Tensorflow*, na parte de treino (aprendizado) e validação (teste). (A, B) Montagem de *perceptron* com múltiplas camadas; (C, D) montagem com arranjo de rede convolucional (CNN); (E, F) com aumento da diversidade do *dataset* por aumento do número de épocas (iterações no treinamento) e transferência de aprendizado (*InceptionV3*).

Resultados e discussão

O avanço das tecnologias de Inteligência Artificial, especialmente nas áreas de *Machine Learning* e *Deep Learning*, tem impulsionado o desenvolvimento de sistemas especialistas para diagnóstico de doenças foliares em plantas. A literatura científica analisada comprova que esses sistemas são capazes de identificar e classificar doenças com precisão superior aos métodos tradicionais baseados exclusivamente em conhecimento empírico e diagnose visual. A utilização de sistemas especialistas baseados em DL revelou-se particularmente eficiente na detecção de sintomas visuais complexos, como manchas irregulares, cloroses e necroses. Wang e Liu (2024) demonstraram que o uso de CNNs aplicadas à detecção de doenças do tomate alcançou 96,5% de acurácia, mesmo sob variações de iluminação e qualidade da imagem. Os autores empregaram técnicas de *data augmentation* (rotação, inversão, escalonamento) e otimização por *Adam* para melhorar a robustez do modelo. O presente trabalho adotou estratégia similar nas etapas de treinamento do modelo de DL proposto.

Diversos autores avançaram na construção de sistemas para dispositivos móveis, permitindo diagnósticos em campo com latência mínima. O estudo de Xie et al. (2020), por exemplo, implementou uma CNN personalizada em um aplicativo baseado em um sistema operacional, com tempo de resposta inferior a 2 segundos por imagem e acurácia global de 95,7% na detecção de doenças em folhas de videira. Isso demonstra a viabilidade prática de sistemas especialistas portáteis, principalmente em regiões com acesso limitado a laboratórios ou técnicos especializados. Um modelo semelhante foi proposto por Ahmed e Reddy (2021), e testado com imagens reais capturadas por agricultores, mostrando-se resiliente a ruídos como sombras e variação climática.

Do ponto de vista metodológico, os estudos se dividem entre o uso de arquiteturas CNN “clássicas” (LeNet, AlexNet, ResNet) e modelos mais recentes, como EfficientNet e YOLOv5. Balafas et al. (2023) realizaram uma comparação sistemática entre cinco algoritmos de detecção de objetos, incluindo YOLOv5 e Faster R-CNN, para identificar sintomas em folhas de uva. Os resultados indicaram que YOLOv5 obteve a melhor combinação entre precisão e velocidade, com mAP (*mean Average Precision*) de 92% e tempo médio de inferência inferior a 60 milissegundos (ms). Essa abordagem também permitiu mapear a área afetada na folha, fornecendo uma estimativa indireta da severidade da doença, um aspecto pouco explorado na maioria dos estudos.

Neste trabalho, após a organização (em pastas) das imagens provenientes de três bases de dados repositórios publicos, foi conduzida a proposição de montagem e treinamento da Rede Neural usando o *framework Tensorflow*, disponível para o *Python*. As imagens de folhas com manchas de óleo e pulverulência de míldio pelo Redape vieram com um *background* diverso das outras bases de dados usadas, mas, isso não afetou o treinamento. Diversos estudos também abordaram o papel dos conjuntos de dados na qualidade dos modelos diagnósticos. Ferentinos (2018) alerta para o fato de que a maioria dos *datasets* amplamente utilizados (como *PlantVillage*) contém imagens capturadas em ambientes controlados, com fundo homogêneo e folhas isoladas. Esse cenário, segundo os autores, não reflete a complexidade do ambiente de campo, onde fatores como iluminação, sobreposição de folhas, e presença de múltiplos sintomas dificultam a detecção. Eles sugerem a construção de conjuntos de dados mais diversos, com anotações multilabel e segmentação de lesões, para melhorar a aplicabilidade prática dos modelos.

Para o presente modelo, a divisão estratificada (com *validation_split*) assegurou que o modelo fosse treinado em dados distintos daqueles usados para validação, permitindo monitorar o desempenho generalizável. O uso de GPU com controle de memória evitou ‘estouros’ (OOM) de memória durante o treinamento, tornando a execução mais estável. Em seguida, foi definida a estrutura básica do primeiro modelo de classificação. A normalização com *Rescaling* auxiliou o otimizador a convergir mais rapidamente ao trabalhar com valores numéricos padronizados. A estrutura simples fornecida pelo *Tensorflow* (como *Flatten* e *Dense*) foi escolhida para estabelecer um modelo de referência inicial para a modelagem. A ativação *softmax* na última camada, apropriada para tarefas de classificação multiclasse, garantiu que o modelo atingisse facilmente 93% de acurácia. O otimizador *Adam* foi selecionado por sua eficiência e adaptabilidade durante o treinamento.

Na sequência, foi adotada a adição de camadas convolucionais para captura de padrões espaciais relevantes nas imagens (como bordas, texturas e formas), com o treinamento sendo realizado mediante validação cruzada e aumento de épocas (iterações) para acompanhar o desempenho dos modelos durante os processos de aprendizado, reduzindo o risco de *overfitting* que parece ser o problema recorrente de trabalhos em DL quanto em ML. Paralelamente, foi implementada a parada antecipada, já que a acurácia do modelo (em validação

cruzada) atingia e ultrapassava um valor de 95% após apenas cinco ou seis épocas (ciclos de ajuste) (Figura 1).

Vendo a abordagem de Wang e Liu (2024) para doenças do tomateiro, foi tentado no modelo em desenvolvimento um aumento de execução de épocas para o valor de 50, mesmo arriscando aumento de *overfitting*. Por isso houve a motivação para usar a função de *data_augmentation* aumentando a variabilidade dos dados de treinamento sem aumentar o número real de imagens. Isso ajudou o modelo a generalizar melhor para novos dados e reduz o risco de *overfitting*. Ao treinar por 50 épocas, o modelo teve mais oportunidades de aprender padrões robustos nas imagens transformadas aleatoriamente, melhorando seu desempenho preditivo em situações do mundo real.

Neste trabalho, seguindo o *pipeline* de busca pelo melhor modelo, foi, por fim, implementada a integração de modelo pré-Treinado (*InceptionV3*) com classificador personalizado disponível no pacote *Keras* e deseenvolvido com imagens de repositório público de dados. O uso da *InceptionV3* como base visa aproveitar representações visuais complexas já aprendidas em um vasto conjunto de imagens (ImageNet) acelerando o treinamento e melhorando a acurácia mesmo com um conjunto de dados relativamente pequeno. Essa estratégia permite aplicar a rede a um novo domínio (como doenças foliares) sem precisar treinar desde o zero, o que é especialmente útil quando há limitação de imagens rotuladas disponíveis. Embora a *InceptionV3* tenha sido treinada originalmente com imagens do ImageNet (como gatos, carros, frutas, etc.), suas camadas iniciais aprendem padrões visuais genéricos (bordas, texturas, formas) que também são úteis para folhas de plantas. Assim, reutilizar esses filtros evita treinar tudo do zero e acelera a convergência com menos dados. Esse tipo de adaptação é muito comum em problemas com bases de dados menores, como é o caso de folhas de videira. Por exemplo, Wang et al. (2020) treinaram modelos *InceptionV3* e outros com um conjunto de imagens de insetos, alcançando desempenho médio de 94 a 97% de acurácia, com destaque para a capacidade dos modelos em generalizar entre diferentes cultivares.

Saindo um pouco do DL, modelos baseados em ML clássico continuam sendo relevantes, principalmente quando há limitação de dados ou necessidade de interpretabilidade. Barbedo (2013) enfatiza que algoritmos como SVM, Random Forest e k-NN ainda superam CNNs em contextos com conjuntos de dados reduzidos e rótulos incompletos. O autor utilizou atributos extraídos manualmente (cor, textura, forma) de folhas de soja e milho, obtendo acurácia média de 90% com *Random Forest*, sem a necessidade de treinamento profundo. Da mesma forma, Princy e Belwal (2023) compararam CNNs com técnicas baseadas em vetores de características manuais, concluindo que os modelos clássicos exigem menos tempo de treinamento e podem ser implementados com menor poder computacional.

A integração de diferentes fontes de dados também é uma tendência crescente. Raghuram e Borah (2025) propuseram um modelo híbrido CNN-LSTM para capturar a evolução temporal de sintomas, permitindo o diagnóstico não apenas baseado na imagem estática, mas na progressão dos sintomas ao longo do tempo. O sistema foi validado em um conjunto de imagens sequenciais de folhas de tomate e alcançou acurácia de 96,8% na previsão da doença com base em séries temporais visuais. Essa abordagem é especialmente útil para culturas como videira e citros, onde os sintomas podem mudar rapidamente em função de estresse hídrico ou ataque fúngico. A explicabilidade dos modelos é essencial na agricultura. Enquanto algoritmos tradicionais de ML permitem interpretações claras via regras ou árvores, redes neurais convolucionais (CNNs) operam como “caixas-pretas”. Técnicas como Grad-CAM e LIME têm sido usadas (Golhani et al., 2018) para indicar regiões da imagem que influenciam a decisão do modelo, aumentando a confiança do usuário e viabilizando validação por especialistas. Em DL, Chen et al. (2020) usaram *transfer learning* com VGG19 e ResNet50, reduzindo o tempo de treinamento em 60% e mantendo alta acurácia em culturas como manga e pimenta-do-reino. Já Sujatha et al. (2021) destacaram técnicas como quantização e poda, permitindo a execução de modelos em microcontroladores, viabilizando seu uso em drones e sensores de campo.

Ainda neste trabalho, para testar a aplicabilidade do modelo proposto, um app com *back-end* em *Flask* e um pequeno front-end em <html> foi sugerido para que o usuário pudesse disponibilizar e testar o modelo final de DL aqui desenvolvido, convertido e salvo para o formato T*ensorflow Lite* (*TFLite*) com suas próprias fotos de folhas lesionadas. O <html> pode ser rodado em qualquer *browser* após chamada simples (Figura 2).

A conversão para o *TFLite* garante bom desempenho do modelo em *front-end*, mesmo em ambientes com restrições de recursos computacionais (sem GPU), como dispositivos móveis ou embarcados. Ao integrar o modelo *TFLite* ao *Flask*, foi possível criar um protótipo de API simples e funcional para realizar testes de classificação de imagens

Back-end <Flask>para carregamento do modelo:

```
from flask import Flask, request, render_template
import numpy as np
import tensorflow as tf
from PIL import Image
import io

app = Flask(__name__)
interpreter =
tf.lite.Interpreter(model_path='mquantizado.tflite')
interpreter.allocate_tensors()

@app.route('/', methods=['GET', 'POST'])
def index():
    prediction = None
    if request.method == 'POST' and 'imagem' in
request.files:
        image_file = request.files['imagem']
        image = Image.open(image_file).resize((256, 256))
        image = np.expand_dims(np.array(image,
dtype=np.float32) / 255.0, axis=0)

        input_details = interpreter.get_input_details()
        output_details = interpreter.get_output_details()
        interpreter.set_tensor(input_details[0]['index'],
image)
interpreter.invoke()
        output_data =
interpreter.get_tensor(output_details[0]['index'])[0]

      # Lista atualizada com apenas 3 classes
      classes = ['chocolate', 'manchas', 'sadias']
      prediction = {cls: f"{prob*100:.2f}%" for cls, prob
in zip(classes, output_data)}

    return render_template('index.html',
prediction=prediction)

if __name__ == '__main__':
    app.run(debug=True)
```

Front-end <HTML> para submissão de imagens

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8">
<title>Classificação de Folhas de
Videira</title>
</head>
<body>
<h1>Classificador de Doenças em Uvas </h1>
<form method="POST" enctype="multipart/form-
data">
<input type="file" name="file"
accept="image/*" required>
<input type="submit" value="Classificar">
</form>

    {% if predictions %}
<h2>Resultados da Classificação:</h2>
<ul>
          {% for classe, prob in predictions
%}
<li><strong>{{ classe }}</strong>:  {{ prob
}}%</li>
              {% endfor %}
</ul>
    {% endif %}
</body>
</html>
```

Figura 2. Segmento de *script* em *Python* (*Flask*) e em <html> como uma sugestão para usuário testar o modelo desenvolvido (salvo em *TFLite*, 'modelo_quantizado16bits.tflite') usando suas próprias imagens.

com dados novos. Essa API pode servir de ponto de partida para ferramentas práticas no campo, como aplicativos diagnósticos rápidos de doenças em folhas de videira, sem exigir estruturas complexas de *back-end* nem intervenção de "achismos" por parte do usuário.

Por fim, além do presente esforço, estudos recentes envolvendo um sem número de culturas e situações demonstram que os sistemas especialistas baseados em DL são hoje o estado da arte para diagnóstico de doenças foliares. Oferecem alto desempenho, escalabilidade e adaptabilidade. No entanto, os modelos de ML continuam sendo alternativas viáveis, especialmente em contextos com limitação de dados e necessidade de interpretabilidade. O campo caminha para soluções cada vez mais portáteis, explicáveis e multimodais, unindo imagem, tempo, localização e conhecimento agronômico em ferramentas de apoio à decisão acessíveis ao vitivinucultor.

Conclusões

1) Modelos de aprendizado profundo foram desenvolvidos com base em Redes Neurais Convolucionais (CNNs) e transferência de aprendizado, utilizando a biblioteca *Tensorflow/Keras*, para classificar automaticamente sintomas foliares de doenças em videiras com base em imagens.
2) Um modelo inicial foi treinado com imagens organizadas em três classes (folhas sadias, com míldio e com manchas), alcançando acurácia superior a 93% no conjunto de validação, demonstrando alta capacidade discriminativa. O modelo pode ser expandido para outras doenças facilmente, desde que haja base de dados para treinamentos.
3) A aplicação de técnicas de aumento de dados (*data augmentation*) e a posterior utilização de transferência de aprendizado com a arquitetura *InceptionV3* melhoraram a robustez e reduziram a necessidade de grandes volumes de imagens rotuladas.
4) Como primeira versão funcional, o modelo foi convertido para o formato *TFLite* e implantado em um aplicativo interativo via *Flask*, mostrando potencial real de uso em campo por técnicos e produtores em aplicações ao usuário.
5) Há perspectivas claras de evolução, incluindo: a) treinamento com bases de dados mais diversas e desbalanceadas para melhorar a generalização; b) uso de explicabilidade (como *Grad-CAM*)

para aumentar a confiança dos usuários; c) integração com sensores e sistemas de monitoramento agrícola; 4) adaptação do modelo para dispositivos móveis ou drones com restrições computacionais; e 5) desenvolvimento de *pipelines* automáticos para atualização contínua do modelo com dados reais coletados em campo.

6) O modelo final pode servir como base para *upgrades* em sistemas especialistas de gerações anteriores, baseados em chaves dicotômicas simples, dependentes da imprecisão e opiniões pessoais de usuários humanos

Referências

AHMAD, A.; SARASWAT, D.; EL GAMAL, A. A survey on using deep learning techniques for plant disease diagnosis and recommendations for development of appropriate tools. **Smart Agricultural Technology**, v. 3, p. 100083, Feb. 2023. DOI: https://doi.org/10.1016/j.atech.2022.100083.

AHMED, A. A.; REDDY, G. H. A mobile-based system for detecting plant leaf diseases using deep learning. **AgriEngineering**, v. 3, n. 3, p. 478-493, July 2021. DOI: https://doi.org/10.3390/agriengineering3030032.

AHMED, I.; YADAV, P. K. A systematic analysis of Machine Learning and Deep Learning based approaches for identifying and diagnosing plant diseases. **Sustainable Operations and Computers**, v. 4, p. 96-104, 2023a. DOI : https://doi.org/10.1016/j.susoc.2023.03.001.

AHMED, I.; YADAV, P. K. Plant disease detection using machine learning approaches. **Expert Systems**, v. 40, n. 5, e13136, 2023b. DOI: https://doi.org/10.1111/exsy.13136.

BALAFAS, V.; KARANTOUMANIS, E.; LOUTA, M.; PLOSKAS, N. Machine learning and deep learning for plant disease classification and detection. **IEEE** Access, v. 11, p. 114352-114377, Oct. 2023. DOI: 10.1109/ACCESS.2023.3324722.

BARBEDO, J. G. A. Digital image processing techniques for detecting, quantifying and classifying plant diseases. **Springer Plus**, v. 2, n. 660, Dec. 2013. DOI: https://doi.org/10.1186/2193-1801-2-660.

CAVALCANTI, F. R. Modelo de aprendizado supervisionado para validações a campo do algoritmo Embrapa/MAHM. Bento Gonçalves: Embrapa Uva e Vinho, 2025. (Embrapa Uva e Vinho. Boletim de Pesquisa e Desenvolvimento, 36). Disponível em: https://www.infoteca.cnptia.embrapa.br/infoteca/bitstream/doc/1176016/1/BolPD-36.pdf. Acesso em: jul. 2025.

CHEN, J.; CHEN, J.; ZHANG, D.; SUN, Y.; NANEHKARAN, Y. Using deep transfer learning for image-based plant disease identification. **Computers and Electronics in Agriculture**, v. 173, p. 105393, June 2020. DOI: 10.1016/j.compag.2020.105393.

EMBRAPA. **Redape**: repositório de dados de pesquisa da Embrapa. Disponível em: https://www.redape.dados.embrapa.br/. Acesso em: 17 nov. 2025.

FERENTINOS, K. P. Deep learning models for plant disease detection and diagnosis. **Computers and Electronics in Agriculture**, v. 145, p. 311–318, 2018. DOI: https://doi.org/10.1016/j.compag.2018.01.009.

GOLHANI, K.; BALASUNDRAM, S. K.; VADAMALAI, G.; PRADHAN, B. A review of neural networks in plant disease detection using hyperspectral data. **Information Processing in Agriculture**, v. 5, n. 3, Sept. 2018. DOI : https://doi.org/10.1016/j.inpa.2018.05.002.

JACKULIN, C.; MURUGAVALLI, S. A comprehensive review on detection of plant disease using machine learning and deep learning approaches. **Measurements**: Sensors, v. 24, p. 100441, Dec. 2022. DOI : https://doi.org/10.1016/j.measen.2022.100441.

KHAN, M. A. **Figshare**: repositório de dados de pesquisa. Disponível em: https://figshare.com/authors/Murtaza_Khan/16014491. Acesso em: 17 nov. 2025.

MANDAL, R. **Augmented grapevine disease dataset**. Kaggle. Disponível em: https://www.kaggle.com/datasets/rm1000/augmented-grape-disease-detection-dataset. Acesso em: 17 nov. 2025.

OWOMUGISHA, G.; MWEBAZE, E. Machine Learning for plant disease incidence and severity measurements from leaf images. In : INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND APPLICATIONS, 16., 2016, Anahein. **Anais**... Anahein : ICMLA, p. 18-20, Dec. 2016.

PRINCY, K. T. M.; BELWAL, M. Plant Disease Detection: a comparative study of deep learning approaches. In : INTERNATIONAL CONFERENCE ON INVENTIVE RESEARCH IN COMPUTING APPLICATIONS 5., 2023. **Anais**... ICIRCA, 2023. DOI : 10.1109/icirca57980.2023.10220943.

RAGHURAM, K.; BORAH, M. D. A hybrid learning model for tomato plant disease detection using deep reinforcement leraning with transfer learning. **Procedia Computer Science**, v. 252, p. 341-354, 2025. DOI : https://doi.org/10.1016/j.procs.2024.12.036.

SALEEM, M. H.; POTGIETER, J.; ARIF, K. M. Plant disease detection and classification by deep learning. **Plants**, v. 8, n. 11, p. 468, Oct. 2019. DOI: https://doi.org/10.3390/plants8110468.

SUJATHA, R.; CHATTERJEE, J. M.; JHANJHI, N. Z.; BROHI, S. N. Performance of deep learning vs machine learning in plant leaf disease detection. **Microprocessors and Microsystems**, v. 80, n. C, p. 103615, Feb. 2021. DOI: https://doi.org/10.1016/j.micpro.2020.10361.

WANG, J.; YANE, L.; FENG, H.; REN, L.; DU, X.; WU, J. Common pests image recognition based on deep convolutional neural network. **Computers and Electronics in Agriculture**. v. 179, p. 105834, Dec. 2020. DOI: 10.1016/j.compag.2020.105834.

WANG, X.; LIU, J. An efficient Deep Learning model for tomato disease detection. **Plant Methods**, v. 20, n. 61, May 2024.DOI : 10.1186/s13007-024-01188-1.

WANI, J. A.; SHARMA, S.; MUZAMIL, M.; AHMED, S.; SHARMA, S.; SINGH, S. Machine learning and deep learning based computational techniques in automatic agricultural diseases detection: Methodologies, applications, and challenges. **Archives of Computational Methods in Engineering**, v. 29, p. 641-677, April 2022. DOI: 10.1007/s11831-021-09588-5.

XIE, X.; MA, Y.; LIU, B.; HE, J.; LI, S.; WANG, H. A Deep-Learning-based real-time detector for grape leaf diseases using improved Convolutional Neural Networks. **Frontiers in Plant Science**, v. 11, June 2020. DOI: https://doi.org/10.3389/fpls.2020.00751.