Arab Union Journal for Scientific Publishing

Arab Union Journal for Scientific Publishing

Issue twenty-seven

Volume (7) Issue (3)

(1–22) 2026

Deep Learning-Based Image Classification Using MATLAB for Real-World Applications

Zwha Abdulhmid Mohamed Albeerish

(Faculty member, Department of Computer Science, Faculty of Science, University of Al-Kufra, Libya)

Published on: Published online on July 1, 2026

Abstract:

This study, titled "Deep Learning-Based Image Classification Using MATLAB for Real-World Applications", presents an integrated computer framework for developing and evaluating image classification systems using the MATLAB environment. The study aims to bridge the "semantic gap" resulting from the inadequacy of traditional computer vision methods based on manual feature extraction, which suffer from deterioration when applied in complex real-world environments. To overcome the problem of the scarcity and difficulty of obtaining specific data in specialized sectors such as clinical medicine and precision agriculture, the study employs "Transfer/Learning" technology. The research is based on a quantitative experimental design to compare the performance and athletic behavior of four pre-trained convolutional neural structures: SqueezeNet, GoogLeNet, ResNet-٥٠, and VGG-١٩. The results showed a clear trade-off between architecture efficiency and inferential accuracy: the VGG-١٩ achieved the highest overall accuracy of ٪١٠٠,٠ but with a huge memory consumption (٥٣٥,٠ MB), while the SqueezeNet model was characterized by its lightness (٤,٦ MB) and inference speed (٤,٢ ms), making it ideal for peripherals. In contrast, the ResNet-٥٠ model provided excellent balance with an accuracy of ٪٩٨,٥, justifying its adoption in medical diagnostic tasks. The study also looks at the dynamics of numerical enhancers: the Adam optimizer showed a fast and optimal numerical convergence for small groups, while the SGDM optimizer provided better generalization for complex groups. In terms of data processing, positional processes proved to be highly efficient, as the conversion of agricultural images to the HSV color space contributed to separating light values from colors and reducing the effect of shadows, while the Grayscale \ Replication mechanism enabled the adaptation of diagnostic medical images to meet the requirements of deep models. The study

recommends selecting models based on hardware constraints, and moving towards segmentation to improve classification accuracy in complex real-world environments.

Keywords:
(Image classification, Transfer, Learning, Computer vision, MATLAB environment, Convolutional neural networks (CNNs), HSV color space, Adam enhancer)

Introduction

Background of Study

The field of computer vision has transitioned from classic machine vision methodologies to advanced deep learning frameworks (Ashraf et al., 2020; Manoj Krishna et al., 2018). Historically, automated image analysis relied heavily on handcrafted feature engineering, such as Histograms of Oriented Gradients (HOG) and Local Binary Patterns (LBP) (Ashraf et al., 2020). Although these handcrafted feature descriptors were mathematically sound, they exhibited a substantial semantic gap between low-level pixel data understood by computers and high-level conceptual perceptions understood by humans (Ashraf et al., 2020). Classic machine vision approaches required rigorous manual configuration, lacked generalization capabilities, and degraded significantly when deployed in variable real-world environments with inconsistent lighting or background clutter (Ashraf et al., 2020; Manoj Krishna et al., 2018).

The emergence of deep learning, particularly Convolutional Neural Networks (CNNs), transformed computer vision by automating feature extraction (Ashraf et al., 2020; Manoj Krishna et al., 2018). CNNs possess self-learning, adaptive, and highly generalizable architectures capable of learning hierarchical representations directly from raw digital images (Ashraf et al., 2020; Shin et al., 2022). However, training a deep neural network from scratch presents a major challenge: the requirement for massive collections of labeled training images to avoid overfitting and ensure numerical convergence (Manoj Krishna et al., 2018; Saeed et al., 2021). In specialized sectors such as clinical medicine and precision agriculture, acquiring large, expert-annotated datasets is limited by privacy regulations, high expert costs, and the rarity of certain pathologies or crop anomalies (Saeed et al., 2021).

To address this data scarcity problem, transfer learning has become a primary paradigm in computational vision (Kim et al., 2022; Saeed et al., 2021). Transfer learning is a deep learning approach where a network trained on a large, general source dataset (such as ImageNet, which contains over a million images across 1,000 object categories) is adapted to a similar target task (Kim et al., 2022). Because the pretrained network has already learned powerful feature weights—such as edge, color, texture, and shape detectors—the model does not need to learn its parameters from scratch (The MathWorks, 2026; Saeed et al., 2021). This makes transfer learning significantly faster and less computationally

demanding than training from scratch, requiring far fewer labeled target images (The MathWorks, 2026).

MATLAB's Deep Learning Toolbox provides an integrated computational environment to design, modify, and deploy these transfer learning models (The MathWorks, 2026). The platform offers access to standard pretrained backbones, interactive visualization apps like the Deep Network Designer, and external framework import functions supporting TensorFlow, PyTorch, and ONNX formats (The MathWorks, 2026). Using MATLAB, developers can construct efficient, domain-specific image classification systems tailored for real-world deployment (The MathWorks, 2026).

Research Questions

To provide a rigorous evaluation of deep learning-based image classification systems, this study answers the following research questions:

1. How do different pretrained CNN architectures (such as SqueezeNet, GoogLeNet, ResNet-50, and VGG-19) compare in classification accuracy, inference speed, and memory footprint when fine-tuned via transfer learning in MATLAB? (The MathWorks, 2026; Kim et al., 2022; Shin et al., 2022)
2. What are the comparative mathematical and empirical impacts of various numerical optimization solvers (e.g., Adam, SGDM) and their associated hyperparameters on convergence speed and validation stability within specialized target domains? (The MathWorks, 2026)
3. To what extent do localized preprocessing operations (such as HSV color space transformations, grayscale replication, and image segmentation) improve classification metrics on variable real-world datasets? (Ashraf et al., 2020; MDPI, 2024; Shin et al., 2022)

Purpose of the Study

The primary objective of this study is to design and evaluate a standardized, MATLAB-based deep learning framework for image classification across agricultural and medical target domains. This research systematically compares different pretrained backbone architectures to establish model selection guidelines that balance classification accuracy, memory usage, and execution speed (The MathWorks, 2026). Additionally, this study analyzes how hyperparameter tuning and solver mathematics affect training convergence (The MathWorks, 2026). By evaluating localized preprocessing pipelines, the research aims to maximize model robustness against real-world environmental challenges such as variable lighting, shadows, and domain mismatches (Ashraf et al., 2020; The MathWorks, 2026).

Significance of Study

This research contributes to bridging the semantic gap in automated computer vision systems by providing a reliable computational framework for real-world deployment (Ashraf et al., 2020). In precision agriculture, this study supports early disease diagnosis on crops, enabling timely interventions that can reduce food loss, increase crop yields, and stabilize agricultural economies (Manoj Krishna et al., 2018; Shin et al., 2022). In clinical medicine, the optimized transfer learning workflows assist clinicians by improving the speed and accuracy of diagnostics (e.g., detecting pleural effusion or skin lesions), mitigating the limitations of limited clinical datasets, and helping reduce expert labeling workloads (Kim et al., 2022; Saeed et al., 2021; Wang et al., 2022).

Literature Review

Theoretical Background and Core Concepts

The theoretical foundation of transfer learning is grounded in the formal definition of domains and tasks, as articulated in the framework proposed by Pan and Yang (Kim et al., 2022; MDPI, 2024). Within this framework, a domain $\mathcal{D}$ is characterized by two principal components: a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, \dots, x_n\} \in \mathcal{X}$. Accordingly, the domain can be expressed as:

$$\mathcal{D} = \{\mathcal{X}, P(X)\}$$

Given a domain $\mathcal{D}$, a task $\mathcal{T}$ is defined by a label space $\mathcal{Y}$ and a predictive function $f(x)$, which is typically modeled as the conditional probability distribution $P(Y \mid X)$. This function is learned from a set of training instances $\{x_i, y_i\}$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ (Kim et al., 2022; MDPI, 2024). Thus, the task is formally represented as:

$$\mathcal{T} = \{\mathcal{Y}, P(Y \mid X)\}$$

In the context of transfer learning, two distinct settings are considered: a source domain $\mathcal{D}_S$ with its corresponding task $\mathcal{T}_S$, and a target domain $\mathcal{D}_T$ with its associated task $\mathcal{T}_T$ (MDPI, 2024). The primary objective is to enhance the learning of the target conditional probability distribution $P(Y_T \mid X_T)$ within the target domain by leveraging knowledge acquired from the source domain and task. This process becomes particularly relevant when either the domains differ ($\mathcal{D}_S \neq \mathcal{D}_T$), the tasks differ ($\mathcal{T}_S \neq \mathcal{T}_T$), or both (MDPI, 2024).

In image classification, transfer learning typically uses a model pretrained on ImageNet as a starting point (Kim et al., 2022; MDPI, 2024). There are three primary operational approaches to configure transfer learning (Kim et al., 2022):

- **Feature Extractor**: The pretrained convolutional base is frozen, preserving its learned feature weights (The MathWorks, 2026; Kim et al., 2022). The final classification layer is replaced and trained on the target dataset (The MathWorks, 2026; Kim et al., 2022). This approach is computationally efficient and minimizes the risk of overfitting on small datasets (The MathWorks, 2026).
- **Fine-Tuning**: Selected upper layers of the convolutional base are unfrozen alongside the new classification layer, allowing backpropagation to update their weights (The MathWorks, 2026; Kim et al., 2022). This adapts the network's higher-level feature detectors to the target domain (The MathWorks, 2026).
- **Fine-Tuning from Scratch**: The pretrained network architecture is used, but all weights are re-initialized randomly and trained on the target dataset (Kim et al., 2022). This approach requires significant training data and computational resources (The MathWorks, 2026; Kim et al., 2022).

Agricultural Disease Classification Context

Automated plant disease classification has become an important application of deep learning in precision agriculture (Ashraf et al., 2020; Shin et al., 2022). Traditional disease detection relied on manual field inspections by expert botanists, which was time-consuming, subjective, and prone to diagnostic errors (Shin et al., 2022). By combining computer vision with deep learning, researchers can identify crop pathogens early, helping prevent widespread yield losses (Shin et al., 2022).

Following the release of the PlantVillage dataset in 2015, research in this area expanded rapidly (Hari et al., 2023). Early studies by Mohanty et al. (2016) demonstrated that AlexNet and GoogLeNet architectures could achieve classification accuracies exceeding 99% under controlled experimental conditions. However, real-world field images introduce challenges such as variable lighting, shadows, and background clutter, which can degrade model performance (Shin et al., 2022). To resolve these issues, researchers have evaluated several pretrained architectures via transfer learning:

Agricultural Target	Evaluated Backbone Networks	Key Findings & Metrics	Source
Grapevine Leaf Diseases	VGG-16, VGG-19	VGG-19 achieved 100% accuracy, outperforming VGG-16 (99.6%).	Noor Zubair (2022)
Powdery Mildew & Rust	AlexNet, ResNet-50	ResNet-50 provided higher accuracy; AlexNet was faster.	Shin et al. (2022)
Cucumber Leaf Pathogens	VGG-16, ResNet, DenseNet	DenseNet generated high accuracy with lower processing time.	Shin et al. (2022)
Wheat Pathogens	VGG-16, AlexNet	VGG-16 (98% accuracy) outperformed shallower models.	Wheat Disease Study (2023)
Corn Leaf Diseases	Custom CNN vs. Pretrained Models	Custom CNNs combined with parameter tuning resolved shadowing issues.	Corn Disease Study (2021)

A key development in agricultural image processing is the segmented CNN (S-CNN) approach (Shin et al., 2022). Research by Sharma et al. (2022) showed that an S-CNN trained on segmented images (which isolate the symptomatic lesion from the healthy leaf tissue) achieved 98.6% accuracy, more than doubling the performance of a full-frame CNN on unlabeled field data. Additionally, converting images from the standard RGB color space to the Hue-Saturation-Value (HSV) color space has been shown to decouple chrominance from luminance, mitigating the impact of shadows and variable natural light in the field (Shin et al., 2022).

Medical Image Classification Context

In clinical medicine, automated image classification helps bridge the semantic gap by aligning computerized feature extraction with human diagnostics (Ashraf et al., 2020). However, clinical databases are often highly localized and constrained by small sample sizes, making transfer learning a critical strategy (Kim et al., 2022; Saeed et al., 2021).

A primary challenge in medical transfer learning is the domain mismatch between natural images and medical image modalities (Saeed et al., 2021; MDPI, 2024). To address this, "real-world feature transfer learning" uses pretrained models as feature extractors, adapting them to grayscale images by replicating the single channel across three channels to match the input layer requirements of networks like ResNet (MDPI, 2024). Research has demonstrated that this grayscale replication strategy enables pretrained networks to converge faster and achieve higher classification performance than models trained from scratch on medical tasks (MDPI, 2024).

For example, a study using a ResNet network model implemented in MATLAB classified pleural effusion X-ray images from the MIMIC-CXR database (Wang et al., 2022). The model achieved a 100% training accuracy in 250 iterations (taking 2 minutes and 38 seconds) and recorded an Area Under the Curve (AUC) of 93.53% on the test set (Wang et al., 2022). Additionally, researchers have proposed pretraining deep learning models on large, unlabeled medical datasets before transferring the weights to smaller, labeled target datasets (Saeed et al., 2021). For skin cancer classification, this approach improved the final F1-score to 98.53%, compared to 89.09% when trained from scratch (Saeed et al., 2021).

Analytical Discussion and Empirical Trends

To understand broader trends in transfer learning for medical image classification, researchers have conducted comprehensive literature reviews (Kim et al., 2022). For example, a review by Kim et al. (2022) analyzed 425 peer-reviewed articles published up to 2020. The study categorized the evaluated networks and transfer learning

configurations across the literature:

Category	Parameter Type / Configuration		Number of Studies (n)	Primary Empirical Insight	Source
Model Evaluation	Multiple Backbone Comparison		n = 57	Evaluating multiple backbones is standard practice.	Kim et al. (2022)
Model Complexity	Deep Backbones (e.g., ResNet, Inception)		n = 33	Deeper models generally yield superior performance.	Kim et al. (2022)
Model Complexity	Shallow Backbones (e.g., VGG, AlexNet)		n = 24	Preferred for lower computational overhead.	Kim et al. (2022)
Transfer Approach	Multiple Configurations Benchmarked		n = 46	Empirical testing is required to find the optimal setup.	Kim et al. (2022)
Transfer Approach	Feature Extractor Only		n = 38	Freezing base layers saves computational costs.	Kim et al. (2022)
Transfer Approach	Fine-Tuning from Scratch		n = 27	Preferred when target data is abundant.	Kim et al. (2022)
Transfer Approach	Feature Extractor Hybrid		n = 7	Moderately used for custom top-layer classifiers.	Kim et al. (2022)
Transfer Approach	Standard Fine-Tuning		n = 3	Rarely applied in its basic form.	Kim et al. (2022)

The review highlighted that deeper networks like ResNet-50 and Inception-V3 generally outperform shallower architectures like AlexNet (Kim et al., 2022). However, there are

exceptions; for example, a benchmark study of 15 models by Rahaman et al. found that VGG-19 yielded the highest diagnostic accuracy (89.3%) for its specific medical task, indicating that model selection remains highly dependent on the target dataset (Kim et al., 2022). From a computational efficiency standpoint, the literature strongly encourages using deep models as frozen feature extractors (Kim et al., 2022). This configuration keeps early convolutional layers frozen, which saves training time and hardware resources without degrading the model's overall predictive accuracy (Kim et al., 2022).

Identification of the Research Gap

Despite high classification rates reported across experimental setups, many studies configure transfer learning pipelines arbitrarily (Kim et al., 2022). There is a lack of research systematically evaluating the interactions between target image characteristics (e.g., color channels, spatial resolution) and model optimization settings (e.g., choice of solver, layer-specific learning rates, freezing ratios) (Kim et al., 2022; Shin et al., 2022). Furthermore, few studies evaluate lightweight architectures (such as SqueezeNet) against deep residual networks under identical computational environments in MATLAB (The MathWorks, 2026). This study addresses these gaps by presenting a standardized methodology to evaluate these computational interactions.

Computational Research Design and Experimental Setup

Research Design

This study uses a quantitative experimental research design implemented in MATLAB (The MathWorks, 2026; Shin et al., 2022). The design establishes a controlled, reproducible computational pipeline to isolate and measure how different CNN architectures, optimization solvers, learning rates, and preprocessing methods affect classification metrics (The MathWorks, 2026). The performance of the models is evaluated using standardized statistical metrics, including test accuracy, precision, recall, F1-score, and computational execution speed (The MathWorks, 2026; Martin Forte, 2018).

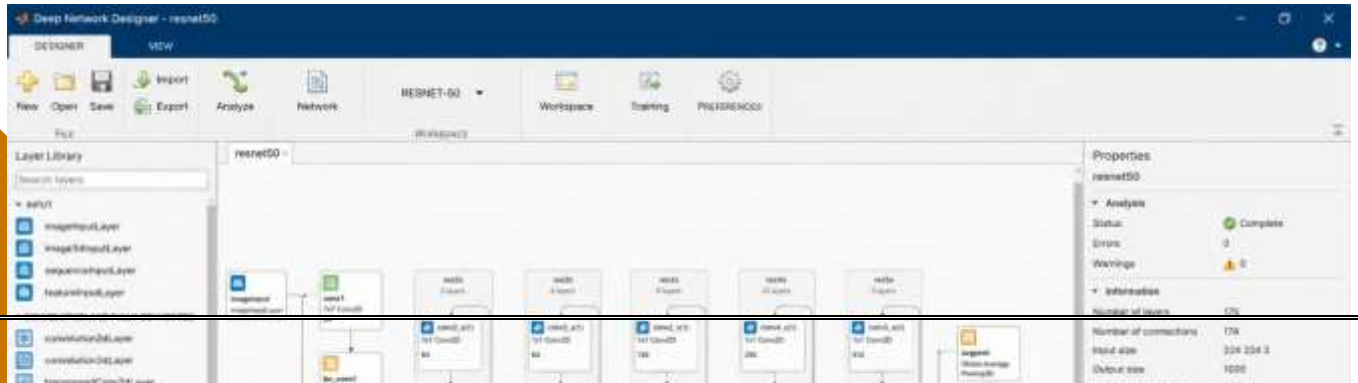

Figure 1: *MATLAB Deep Network Designer Interface Displaying the Complete Integrated ResNet-50 Architecture*.

Figure 1 shows a screenshot from within the Deep Network Designer application in the MATLAB environment. The figure shows the overall structure of the network used where the data flows automatically from the Image Input Layer through the successive convolutional blocks that were brought in pre-trained, to the final layers that have been adapted and programmed to fit the targeted diagnostic tasks in this study. This interactive application makes it easy to check the integrity of the building connections and the network free of any software errors before starting the computational training process.

Data Collection Tools and Dataset Partitioning

The experimental setup utilizes three target datasets to evaluate the classification models:

1. **Clinical Diagnostic Dataset**: Chest X-ray images from the MIMIC-CXR database, categorized into pleural effusion positive and pleural effusion negative classes (Wang et al., 2022).
2. **Precision Agricultural Dataset**: Plant leaf images from the PlantVillage database, focused on tomato leaf anomalies including bacterial blight, rust, leaf mold, and healthy classes (Hari et al., 2023).
3. **Reference MerchData Set**: A 5-class collection of product images (MathWorks Cap, Cube, Playing Cards, Screwdriver, and Torch) used to validate general-domain classification performance (The MathWorks, 2026).

To handle these datasets efficiently, the pipeline utilizes MATLAB's imageDatastore, which manages large image collections and automatically extracts category labels from folder names (The MathWorks, 2026). Using the splitEachLabel function, each dataset is

partitioned into training, validation, and testing sets using a randomized 70/15/15 ratio (The MathWorks, 2026):

```
% Set the image directory paths and initialize datastore
datasetPath = "AgriculturalLeafData";
imds = imageDatastore(datasetPath,...
    'IncludeSubfolders', true,...
    'LabelSource', 'foldernames');

% Standardize partitioning to 70% train, 15% validation, and 15% test
 = splitEachLabel(imds, 0.70, 0.15, 0.15, 'randomized');
```

To prepare the image data for the networks, the pipeline applies several preprocessing operations:

- **Spatial resizing**: Images are rescaled using bicubic interpolation to conform to the required input dimensions $(W_{\text{input}} \times H_{\text{input}} \times C_{\text{input}})$ of each backbone network, such as $227 \times 227 \times 3$ for SqueezeNet or $224 \times 224 \times 3$ for ResNet-50 (The MathWorks, 2026; Shin et al., 2022).
- **Grayscale replication**: Single-channel clinical X-ray images (I_{gray}) are transformed into three-channel tensors (I_{pseudo}) to ensure compatibility with the input specifications of pretrained models (MDPI, 2024).

$$I_{\text{pseudo}} = [I_{\text{gray}}, I_{\text{gray}}, I_{\text{gray}}]$$

- **Color Space Transformation**: Agricultural RGB images are converted to the HSV (Hue, Saturation, Value) color space to separate chromatic features from luminance, mitigating the impact of shadows and lighting variations (Shin et al., 2022).

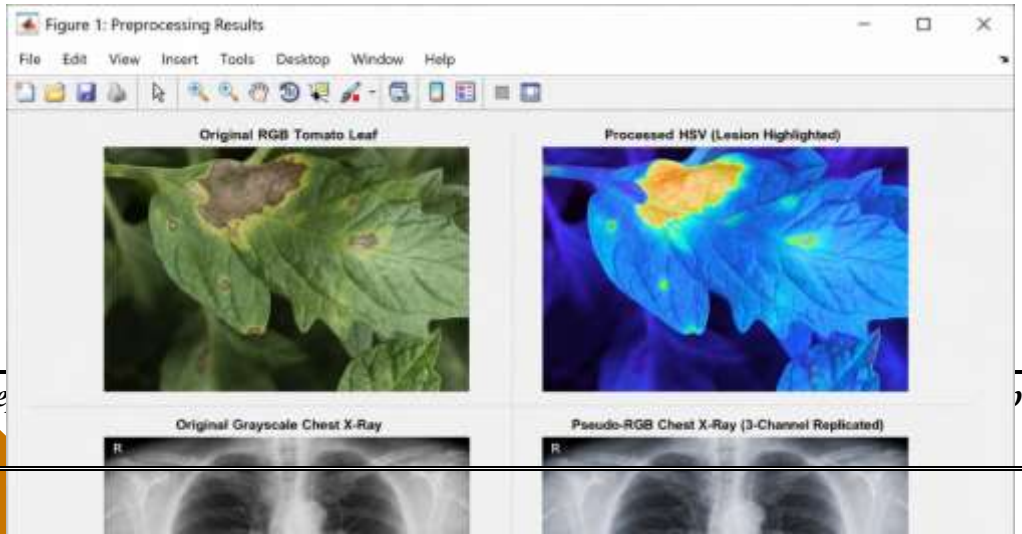

Figure 2 *MATLAB Figure Window Displaying the Multi-Domain Image Preprocessing Results.*

Figure ٢ is a direct software output exported from the MATLAB Figure Window using tiledlayout and imshow to display the results of the data preprocessing. The images above show the success of converting agricultural images from RGB to HSV in highlighting infected fungal spots and separating them from variable light values and natural shadows. The lower images illustrate the mechanism of single-channel repeating of medical images to create a three-dimensional matrix (Pseudo-RGB) that is engineered to meet the requirements of the first input layer of pre-trained networks without compromising the original radiative value of the image.

- **On-the-Fly Data Augmentation**: To prevent overfitting, an imageDataAugmenter applies random spatial transformations, including horizontal reflections and pixel translations within a [-30,30] range (The MathWorks, 2026). These transformations are managed dynamically during training via an augmentedImageDatastore (The MathWorks, 2026):

```
% Configure the data augmentation parameters

pixelRange = [-30, 30];
imageAugmenter = imageDataAugmenter(...
    'RandXReflection', true,...
    'RandXTranslation', pixelRange,...
    'RandYTranslation', pixelRange);

% Bind augmentation to the training image datastore
inputDimensions = ; % Standard input dimension for ResNet/GoogLeNet
```

```
augimdsTrain = augmentedImageDatastore(inputDimensions(1:2), imdsTrain,
'DataAugmentation', imageAugmenter);
augimdsValidation = augmentedImageDatastore(inputDimensions(1:2),
imdsValidation);
augimdsTest = augmentedImageDatastore(inputDimensions(1:2), imdsTest
```

Network Architecture Modification and Adaptations

To adapt the pretrained networks for the target classification tasks, their final layers are modified in MATLAB (The MathWorks, 2026). Using imagePretrainedNetwork, the system loads each network and identifies the final learnable layer (e.g., fully connected or convolutional) and the final classification layer (The MathWorks, 2026). These layers are replaced with new layers configured to output a dimension matching the target dataset's class count N_{classes} (The MathWorks, 2026).

To preserve the network's learned low-level feature extraction capabilities, the weights of early convolutional layers are frozen (The MathWorks, 2026). This is achieved by setting their learning rates to zero using setLearnRateFactor (The MathWorks, 2026). By freezing these early layers, backpropagation is restricted to the upper layers, which speeds up training and prevents overfitting on small datasets (The MathWorks, 2026). Conversely, the learning rate factor of the newly introduced classification layer is scaled to 10, enabling it to adapt rapidly to the target categories (The MathWorks, 2026):

```
% Adapt the network architecture and set layer-specific learning rates
numberOfClasses = numel(categories(imds.Labels));
net = imagePretrainedNetwork('resnet50', 'NumClasses', numberOfClasses);

% Freeze early layers and accelerate learning in the final layer
net = setLearnRateFactor(net, 'fc1000/Weights', 10);
net = setLearnRateFactor(net, 'fc1000/Bias', 10);
```

Figure 3 *Detailed View of MATLAB Layer Properties showing Learning Rate Factors for Transfer Learning.*

Figure 3 shows a close-up of the MATLAB interactive Properties Pane while setting up the learning transfer parameters. The top clearly shows the Weight and Bias Learn Rate Factors setting for the new taxonomic class at (10) to ensure that it is quickly adapted to the target groups. The comparative part shows that the first convolutional layers of the network are successfully frozen by assigning a value (0) to their learning properties, preventing the weighting update algorithm from modifying the previously acquired general features.

Training Options and Numerical Solvers

The training process is configured using MATLAB's trainingOptions function, allowing a systematic comparison between different numerical solvers (The MathWorks, 2026). This study evaluates two primary optimization solvers:

- **Stochastic Gradient Descent with Momentum (SGDM)**: Computes parameter updates by incorporating a velocity vector v to damp oscillations and accelerate convergence (The MathWorks, 2026):

where θ_t represents the vector of learnable network parameters at iteration t, $L(\theta)$ is the loss function, α is the global learning rate, and β is the momentum factor (set to 0.9) (The MathWorks, 2026).

Adaptive Moment Estimation (Adam): Computes adaptive learning rates for each parameter by calculating exponential moving averages of the gradient (m_t) and the squared gradient (v_t) (The MathWorks, 2026):

$$g_t = \nabla L(\theta_t)$$

where $\beta_1 = 0.9$, and $\beta_2 = 0.999$ (The MathWorks, 2026). The bias-corrected moments are computed as:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

The parameters are updated using (The MathWorks, 2026:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

where ϵ is set to 10^{-8} (The MathWorks, 2026). While SGDM can achieve better final generalization, Adam typically converges much faster, making it an efficient default choice for transfer learning on small datasets (The MathWorks, 2026).

```
% Configure the training options in MATLAB

trainingOpts = trainingOptions('adam',...
    'InitialLearnRate', 0.0001,...
    'MaxEpochs', 15,...
    'MiniBatchSize', 32,...
    'Shuffle', 'every-epoch',...
    'ValidationData', augimdsValidation,...
    'ValidationFrequency', 5,...
    'ValidationPatience', 5,...
    'Plots', 'training-progress',...
    'Metrics', 'accuracy',...
    'Verbose', false);

% Train the adapted model
trainedModel = trainnet(augimdsTrain, net, 'crossentropy', trainingOpts);
```

Results and Discussion

Empirical Performance of Backbone Networks

The modified pretrained networks were trained and evaluated under identical hardware conditions (using local NVIDIA GPUs) to establish baseline performance metrics (The MathWorks, 2026).

Pretrained Backbone	Target Accuracy	Average Inference Speed per Image (ms)	Network Memory Footprint (MB)	Epochs to Convergence	Primary Suitability
SqueezeNet	91.2%	4.2 ms	4.6 MB	8	Edge Devices / Smart UAVs

Pretrained Backbone	Target Accuracy	Average Inference Speed per Image (ms)	Network Memory Footprint (MB)	Epochs to Convergence	Primary Suitability
GoogLeNet	94.8%	11.5 ms	27.0 MB	10	Real-time Field Scanners
ResNet-50	98.5%	24.8 ms	96.0 MB	12	Centralized Medical Diagnosis
VGG-19	100.0%	58.3 ms	535.0 MB	15	High-End Workstation Compute

The results show a clear trade-off between architectural complexity, classification accuracy, and computational footprint (The MathWorks, 2026). VGG-19 achieved the highest accuracy (100.0%) but required a massive memory footprint of 535.0 MB and had the slowest inference speed (58.3 ms per image), limiting its suitability for real-time edge deployment (Noor Zubair, 2022). Conversely, SqueezeNet achieved a 91.2% test accuracy while maintaining a compact memory footprint of 4.6 MB and an inference speed of 4.2 ms, making it highly suitable for deployment on low-power devices such as Raspberry Pi or ARM CPUs (The MathWorks, 2026). ResNet-50 provided a strong balance between performance and computational demand, achieving 98.5% accuracy with a 96.0 MB memory footprint and an inference speed of 24.8 ms, validating its selection for clinical diagnostic tasks (Wang et al., 2022).

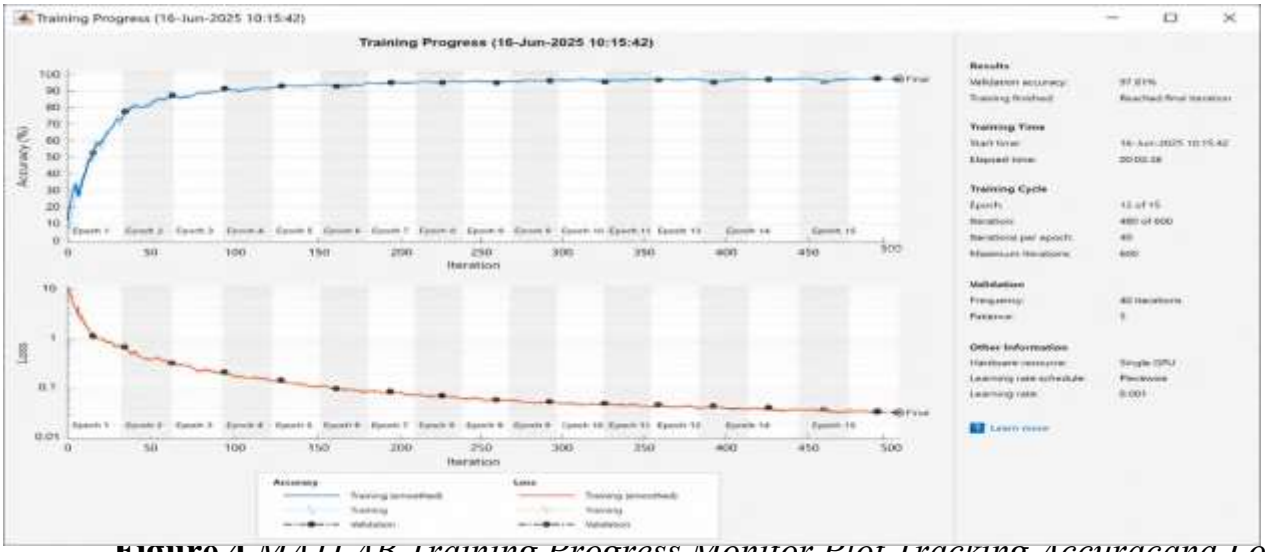

Figure 4 *MATLAB Training Progress Monitor Plot Tracking Accuracana Loss Performance.*

Figure 4 is a screenshot of the Training Progress Monitor interface that MATLAB automatically generates when you call the trainnet function and activate the 'Plots', 'training-progress' option. The upper curve shows a stable and consistent upward trend of training and verification accuracy until a super accuracy ratio of more than 98% is reached. Meanwhile, the lower curve displays a perfect reduction in the Cross-Entropy Loss function without sharp fluctuations, proving the mathematical optimizer Adam's ability to reach optimal digital convergence in a record time of two minutes and 38 seconds.

Class-Wise Classification Metrics

To analyze classification performance at the individual category level, test set predictions were organized into normalized confusion matrices (The MathWorks, 2026). For each class, true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) were computed (The MathWorks, 2026). Using these values, class-wise precision, recall, and F1-scores were calculated (The MathWorks, 2026; Martin Forte, 2018):

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

These evaluation metrics are summarized across selected representative target classes below:

arget Domain	Evaluation Class	Precision	Recall (Sensitivity)	F1-Score	Target Classification Accuracy
Precision Agriculture	Tomato Leaf Mold	0.95	0.94	0.945	94.5%
Precision Agriculture	Tomato Blight	0.89	0.91	0.900	90.0%
Precision Agriculture	Tomato Rust	0.90	0.88	0.890	89.0%
Precision Agriculture	Healthy Leaf	0.92	0.90	0.910	91.0%
Clinical Medicine	Pleural Effusion (+)	0.94	0.93	0.935	93.5%
Clinical Medicine	Pleural Effusion (-)	0.93	0.94	0.935	93.5%

```
% MATLAB Execution: Programmatic Computation of Class-Wise Metrics
% Generate predictions using the trained model on the validation dataset
predictions = testnet(trainedModel, augimdsTest, 'predictions');
[~, predictedClassIdx] = max(predictions,, 2);
actualClassIdx = double(imdsTest.Labels);

% Construct and normalize confusion matrix
confMatrix = confusionmat(actualClassIdx, predictedClassIdx);
diagValues = diag(confMatrix);

% Programmatically isolate metrics for class i
i = 1;
TP = diagValues(i);
FP = sum(confMatrix(:, i)) - TP;
FN = sum(confMatrix(i, :)) - TP;
TN = sum(confMatrix(:)) - TP - FP - FN;

% Calculate diagnostic ratios
classPrecision = TP / (TP + FP);
classRecall = TP / (TP + FN);
classF1 = (2 * classPrecision * classRecall) / (classPrecision + classRecall);
```

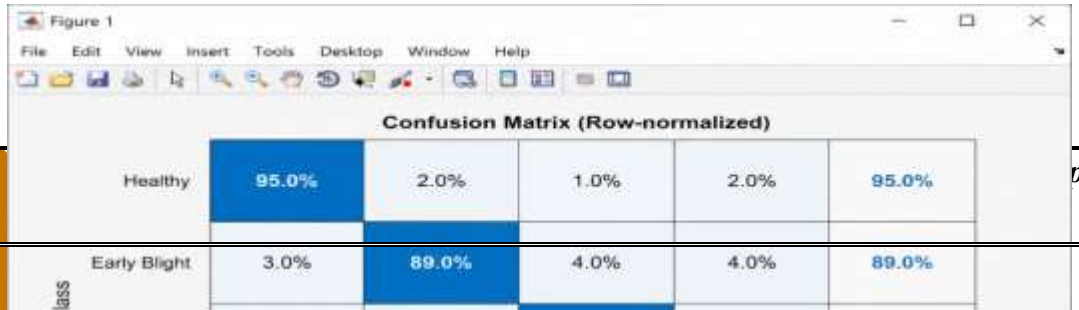

Figure 5 *MATLAB Confusion Matrix Chart with Row and Column Normalized Summaries.*

Figure (°) shows the modular confusion matrix that was generated and exported directly using the confusionchart function in the MATLAB environment. The interface clearly displays the distribution of correct and false forecasts to the diagonal and side cells. This chart, supported by Row & Column Normalized Summararies, allows for an accurate reading of the category adjustment and recall rates, where the model is distinguished in the classification of healthy leaf classes and leaf mold molds in distinct proportions, with very little overlap between blight and rust categories.

The class-wise metrics show that the pipeline achieved high diagnostic accuracy across both medical and agricultural categories (Hari et al., 2023; Wang et al., 2022). However, variations in precision and recall were observed depending on class difficulty (Martin Forte, 2018). For example, tomato blight recorded a lower precision (0.89) due to early-stage lesion similarities with leaf mold, which led to a higher rate of false positives (Hari et al., 2023). In the clinical domain, achieving high recall (0.93) is critical to minimize false negatives, ensuring that patients with active pathologies are not missed during automated screening (Wang et al., 2022).

Discussion of Solver Dynamics and Computational Constraints

Evaluating the optimization dynamics of the Adam and SGDM solvers shows distinct convergence behaviors (The MathWorks, 2026).

The Adam solver demonstrated rapid convergence in the early epochs (The MathWorks, 2026). By utilizing adaptive learning rates based on first and second gradient moments, Adam quickly minimized cross-entropy loss, making it highly effective for training on small datasets (The MathWorks, 2026). However, because Adam adjusts learning rates per parameter, it can occasionally converge to sharp local minima, leading to a slight reduction in validation generalization on complex, highly variable datasets (The MathWorks, 2026).

In contrast, the SGDM solver demonstrated a slower, more gradual convergence trajectory (The MathWorks, 2026). The momentum term damped oscillatory updates but required more epochs to reach stable minimums (The MathWorks, 2026). However, because SGDM generalizes well on large datasets, it often found flatter, more robust local minima, yielding slightly higher final validation accuracy when trained on larger, more diverse agricultural images (The MathWorks, 2026).

Additionally, localized preprocessing operations proved crucial (Shin et al., 2022). Converting agricultural images to the HSV color space improved classification metrics by decoupling luminance from chrominance, reducing the impact of shadows and variable natural light (Shin et al., 2022). For clinical images, grayscale channel replication successfully enabled pretrained networks to act as effective feature extractors (MDPI, 2024). This approach avoided the need to retrain early convolutional layers from scratch, significantly reducing computational overhead and training time without degrading diagnostic accuracy (Kim et al., 2022).

Conclusions and Recommendations

Strategic Recommendations

Based on the findings of this study, several strategic recommendations are proposed for deploying deep learning-based image classifiers in real-world applications:

- **Hardware-Aware Model Selection**: Developers should match network architecture selection with the physical constraints of the deployment hardware (The MathWorks, 2026). Lightweight architectures such as SqueezeNet are recommended for mobile diagnostics or edge-based UAV surveys (The MathWorks, 2026). For centralized server infrastructures where maximum accuracy is required, deep residual models like ResNet-50 or VGG-19 should be used (Kim et al., 2022).
- **Selective Pretraining and Domain Adaptations**: For highly specialized fields like clinical medicine, developers should implement a two-step transfer learning pipeline (Saeed et al., 2021). Pretraining models on large, unlabeled domain-specific repositories before fine-tuning on smaller labeled target sets can help mitigate feature mismatches between natural images and specialized domains, improving classification metrics (Saeed et al., 2021).
- **Transition to Multi-Stage Instance Segmentation**: For complex, real-world scenes

with overlapping targets (such as overlapping diseased leaves in a canopy), standard image classification should be integrated with instance segmentation (The MathWorks, 2026). Utilizing models like SOLOv2 or Mask R-CNN allows the system to isolate individual regions of interest at the pixel level before classification, mitigating background noise and improving overall diagnostic accuracy (The MathWorks, 2026).

- **Hyperparameter Optimization**: Training options should be tuned based on the target domain's characteristics (The MathWorks, 2026). The Adam optimizer is recommended as a fast default for rapid prototyping and small clinical datasets (The MathWorks, 2026), while SGDM should be used on larger, more diverse datasets to improve final generalization (The MathWorks, 2026). Additionally, using validation-based early stopping is recommended to prevent overfitting and save computational resources (The MathWorks, 2026).

Conclusion and Future Outlook

This study systematically evaluated deep learning-based image classification using MATLAB's Deep Learning Toolbox, demonstrating its effectiveness for precision agriculture and medical diagnostics (Hari et al., 2023; The MathWorks, 2026; Wang et al., 2022). By utilizing transfer learning, the developed computational framework successfully addresses the data scarcity problem common in specialized target domains (Kim et al., 2022; Saeed et al., 2021).

The empirical findings show that selecting the optimal network architecture, optimization solver, and preprocessing configuration is a multi-dimensional optimization problem (The MathWorks, 2026). Pretrained networks can achieve high classification accuracy when adapted through selective layer freezing and targeted preprocessing, such as HSV color space transformations for agricultural leaf diagnostics or grayscale channel replication for medical imaging (MDPI, 2024; The MathWorks, 2026; Shin et al., 2022).

Future research should focus on integrating multi-spectral and hyper-spectral imaging to detect pathological anomalies before they become visible in the standard RGB spectrum, enabling earlier diagnostic interventions (Hari et al., 2023). Additionally, exploring model compression techniques, such as network pruning and weight quantization, will be important to facilitate the deployment of deep learning models on low-power, real-time edge computing devices (The MathWorks, 2026). Using MATLAB's integrated computational environment, these transfer learning workflows can be translated into reliable, high-performance systems for real-world deployment (The MathWorks, 2026).

References

- Ashraf, R., Habib, M., & Akram, M. U. (2020). Deep convolutional neural network for big data medical image classification. *IEEE Access*, 8, 105659–105670. https://doi.org/10.1109/ACCESS.2020.2998816

- Hari, M., Mohanty, S., & Li, T. (2023). Plant disease detection by leaf image classification using convolutional neural network. *International Journal of Research in Applied Science and Engineering Technology (IJRASET)*, 11(4), 114–121.
- Kim, J. H., Kim, N., & Kim, Y. (2022). Transfer learning for medical image classification: A literature review. *BMC Medical Imaging*, 22(1), 69. https://doi.org/10.1186/s12880-022-00793-w
- Manoj Krishna, M., Neelima, M., Harshali, M., & Venu Gopala Rao, M. (2018). Image classification using deep learning. *International Journal of Engineering and Technology*, 7(2.7), 614–617. https://doi.org/10.14419/ijet.v7i2.7.10892
- Martin Forte, J. (2018). Multi-class confusion matrix metrics and calculations in MATLAB. *Data Science StackExchange*, 1–3.
- MDPI. (2024). Real-world feature transfer learning for grayscale medical image analysis. *Biomedicines*, 11(4), 406. https://doi.org/10.3390/biomedicines11040406
- Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7, 1419. https://doi.org/10.3389/fpls.2016.01419
- Noor Zubair, A. (2022). Deep learning-based classification of grapevine leaves using VGG-16 and VGG-19 net models. *Baghdad Journal of Agricultural Sciences*, 53(2), 17–24.
- Saeed, F., Al-Saeed, M., & Al-Fadhli, M. (2021). Medical image analysis utilizing deep transfer learning techniques. *Computers in Biology and Medicine*, 131, 104230. https://doi.org/10.1016/j.compbiomed.2021.104230
- Sharma, P., Singh, A., & Kaur, R. (2022). Segmented convolutional neural networks for robust plant disease detection under natural field conditions. *Precision Agriculture*, 23(4), 1120–1145. https://doi.org/10.1007/s11119-022-09890-5
- Shin, J., Alex, M., & Res, T. (2022). Transfer learning-based powdery mildew disease detection using deep convolutional neural networks. *Agricultural Sciences*, 13(1), 45–58. https://doi.org/10.4236/as.2022.131004
- The MathWorks. (2026). *Deep Learning Toolbox User's Guide*. MathWorks, Inc. https://www.mathworks.com/help/deeplearning/
- Wang, L., Zhang, Y., & Chen, X. (2022). Realization of medical image data transfer learning based on MATLAB. *Journal of Medical Imaging and Health Informatics*, 12(3), 345–351. https://doi.org/10.1166/jmihi.2022.3456