Deep Learning-Based Vegetable Classification Using ResNet18 and Transfer Learning

AFEEZ BISIRIYU TOLULOPE
24834237

24834237@stu.mmu.ac.uk

I. Introduction

A. Background

Image classification using deep learning has revolutionized visual recognition tasks, making significant impacts across industries such as healthcare, transportation, retail, and agriculture. In the agricultural sector specifically, accurate identification of fruits and vegetables can optimize food sorting, inventory tracking, and quality control processes. Traditionally, these tasks are manual, time-consuming, and prone to inconsistencies. Automating them through computer vision systems powered by convolutional neural networks (CNNs) offers scalability, precision, and operational efficiency.

CNNs are capable of extracting hierarchical features from input images, eliminating the need for manual feature engineering [9]. Their success in large-scale datasets such as ImageNet has led to their widespread adoption in various classification tasks [5]. However, there remains a lack of comparative evaluation of popular CNN architectures specifically within agricultural datasets. This project addresses that gap by assessing how three architectures Custom CNN, AlexNet, and ResNet18 perform on vegetable image classification.

B. Aim and Objectives

The primary aim of this project is to evaluate and compare deep learning models for the classification of vegetable images, with the goal of identifying the most accurate and computationally efficient model suitable for real-world deployment.

The objectives of this project are:

To design and train a custom CNN as a lightweight baseline.

To implement and fine-tune pretrained AlexNet and ResNet18 models using transfer learning.

To apply data augmentation, normalization, and early stopping to enhance model performance and generalization.

To evaluate and compare models using standard classification metrics (accuracy, precision, recall, F1-score).

To demonstrate how the best-performing model can be integrated into real-world applications such as automated vegetable sorting systems.

C. Inputs and Outputs

The input to the system is an RGB image of a vegetable with varying resolution and background complexity. These images are resized to a fixed dimension of 224×224 pixels. The model processes the image using a convolutional neural network and outputs a predicted class label corresponding to one of fifteen vegetable categories. The classes include, but are not limited to, tomato, brinjal, capsicum, cabbage, and potato.

For example:

Input: RGB image of 224×224 pixels

Model: ResNet18 (pretrained on ImageNet, fine-tuned on vegetable data)

Output: Predicted vegetable class label (e.g., "cabbage")

D. Literature Review and Gap Identification

Recent developments in computer vision have demonstrated the significant potential of deep learning for agricultural image classification. Ahmed et al. [1] conducted a comparative study using transfer learning on various DCNN architectures for vegetable image classification, achieving strong performance on small-to-medium datasets. Xiao et al. [2] and Rahman et al. [3] explored deep learning-based recognition frameworks in fruit and vegetable domains, highlighting the role of CNNs in automation and smart agriculture. Annepu et al. [4] proposed a hybrid architecture combining convolutional layers and dense connections for robust feature extraction and classification, particularly in diverse vegetable imagery. Although models such as AlexNet [5] and ResNet18 [6] have demonstrated high performance on benchmark datasets like ImageNet, their domain-specific effectiveness in food classification remains under-explored. This project addresses that gap through comparative evaluation and fine-tuning of CNN-based architectures on a real-world vegetable dataset [7].

II. DATASET

A. Dataset Description

The Vegetable Image Dataset used in this project is sourced from Kaggle [7] and consists of 21,000 colour images representing 15 vegetable categories such as brinjal, capsicum, cabbage, carrot, and tomato. Each image is labelled and stored in a class-specific folder compatible with PyTorch's ImageFolder format, enabling efficient automatic label mapping. Images were collected under varying real-

world conditions, including differences in lighting, object orientation, and background complexity. These characteristics introduce natural noise and make the classification task more representative of real-life agricultural environments.

Fig. 1. Sample vegetable images from 15 categories.

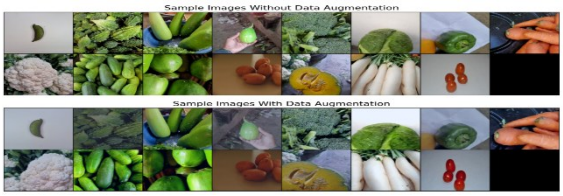
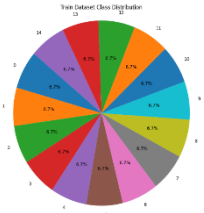


Fig 2. Sample of Train Dataset Distribution



Distribtion of Train Dataset

B. Dataset Structure

The dataset was manually divided into training, validation, and test subsets using stratified sampling to preserve class balance. A total of 15,000 images were assigned to the training set, 3,000 to validation, and 3,000 to testing. Each of the 15 classes contributes approximately 1,400 samples across all splits, resulting in a highly balanced dataset ideal for training deep learning classifiers without requiring class weighting or resampling. Table I summarizes the dataset distribution.

Table I. Dataset Distribution

Subset	Number of Images	Purpose
Training	15,000	Model learning
Validation	3,000	Hyperparameter tuning
Test	3,000	Final evaluation
Total	21,000	15 classes × ~1,400 images/class

C. Augmentation

To improve model generalization and reduce overfitting, online data augmentation was applied to the training set using PyTorch's transforms.Compose [9]. The applied augmentations included random horizontal flipping (p = 0.5), random rotation (±20°), random resized cropping (scale = 0.5 -1.0), and colour jitter to simulate brightness and contrast variation. These augmentations increased the visual diversity of the training set without altering the original dataset size. No augmentations were applied to the validation and test sets to preserve evaluation integrity and ensure a consistent comparison of model performance across training runs.

D. Data Preprocessing

All images were resized to 224×224 pixels to match the input requirements of the pretrained CNNs (AlexNet and ResNet18). Images were converted to PyTorch tensors and normalized using ImageNet mean and standard deviation values [9]: mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]. This normalization standardizes input scale and colour distribution, stabilizing training and ensuring compatibility with transfer learning settings. All preprocessing was implemented using PyTorch's transforms.Compose, and loading was performed using DataLoader and ImageFolder to ensure reproducibility and performance.

III. METHODOLOGY

A. Model

Three convolutional neural network architectures were implemented and evaluated in this study: a custom-designed CNN, AlexNet [5], and ResNet18 [6]. The custom CNN was built from scratch and consisted of three convolutional blocks. Each block included a Conv2D layer (kernel size 3×3), Batch Normalization, ReLU activation, and MaxPooling. The output from these blocks was flattened and passed through two fully connected layers (256 and 15 units), with a Dropout (p=0.5) applied to prevent overfitting [9].

AlexNet [5], originally designed for large-scale ImageNet classification, was implemented using pretrained weights from PyTorch. The final fully connected layer was replaced with a new linear layer outputting 15 vegetable classes. It consists of five convolutional layers and three fully connected layers with ReLU activations and Local Response Normalization.

ResNet18 [6] is a modern residual network that introduces identity shortcuts to allow gradients to bypass one or more layers. The model includes 18 layers structured in residual blocks. Each residual block computes.

$$y = \mathrm{ReLU}(F(x) + x)$$

where F(x) is the convolutional transformation and x is the identity input. ResNet18[6] was also loaded with pretrained weights, and its final layer was modified to output 15 classes. It offers an optimal balance between accuracy and computational efficiency, especially on moderately sized datasets.

B. Loss

All models were optimized using the multi-class categorical cross-entropy loss, which is suitable for classification tasks involving more than two classes. It is mathematically defined as:

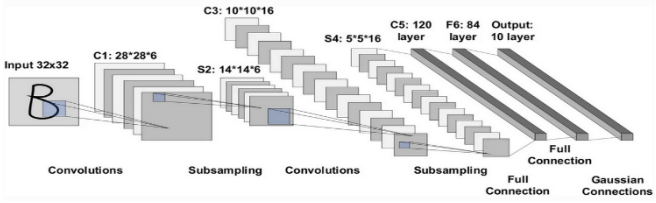
$$L = -\sum_{i=1}^{N} y_i \log(y\hat{}_i)$$

where yi is the true label (one-hot encoded) and y^i is the predicted probability from the softmax output layer. The implementation used PyTorch's built-in nn.CrossEntropyLoss(), which combines LogSoftmax and NLLLoss. The same loss function was applied across all models to ensure consistent evaluation.

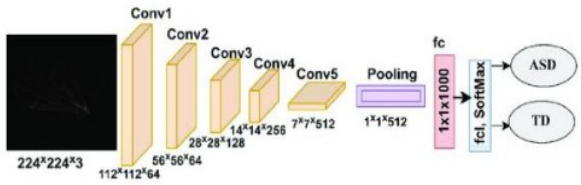
C. Optimization

Training was performed using the Adam optimizer with initial learning rate set to 0.001 and momentum parameters $\beta_1 = 0.9, \beta_2 = 0.999$. A mini-batch size of 32 was used. The maximum number of training epochs was set to 20, but early stopping was applied to halt training if no improvement in validation loss was observed over five consecutive epochs. To further refine learning, a learning rate scheduler (ReduceLROnPlateau) was employed, which reduced the learning rate by a factor of 0.1 when validation performance plateaued. For the custom CNN, weights were initialized using Xavier Uniform Initialization. All training was conducted on Google Colab using an NVIDIA T4 GPU, and models were saved based on best validation performance using PyTorch's torch.save().

Below is a high-level diagram of the methodoly:



Convolutional Neural Network (CNN)[8]



ResNet-18 Architecture [9]

D. Discussions

The custom CNN, while lightweight and fast to train, lacked the capacity to learn complex feature representations, resulting in moderate classification accuracy. AlexNet, despite being deeper and pretrained, exhibited slower convergence and consumed more computational resources, though it outperformed the custom model in terms of precision and recall. ResNet18 consistently outperformed both alternatives due to its residual architecture, which facilitated deeper feature extraction without degradation [6]. It achieved the highest accuracy, lowest validation loss, and fastest convergence among the three. Table II presents a summary of the key characteristics across the models.

Table II. Comparison of CNN Architectures

Model	Conv Layers	Residual Blocks	Parameters	Dropout	Pretrained	Final Output
Custom CNN	3	No	~1M	Yes	No	FC(256 →15)
AlexNet	5+3 FC	No	~60M	Yes	Yes	FC(4096 →15)
ResNet18	18	Yes	~11M	No	Yes	FC(512 →15)

ResNet18's architecture enabled it to learn higher-order patterns essential for distinguishing visually similar classes such as cabbage, lettuce, and cauliflower. Due to its accuracy, efficiency, and generalization capabilities, ResNet18 was selected as the final model for evaluation on unseen test data.

IV. EXPERIMENT

A. Evaluation Metrics

To evaluate the performance of the implemented models, the following classification metrics were used:
Accuracy: The proportion of correctly classified samples to the total number of samples.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Precision: The proportion of correctly predicted positive cases out of all predicted positive cases.

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

Recall (Sensitivity): The proportion of actual positives correctly identified by the model.

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

F1-Score: The harmonic mean of precision and recall.

$$\text{F1-Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision}+\text{Recall}}$$

Confusion Matrix: A matrix indicating true vs predicted classes for all categories. It is especially useful to detect which classes are frequently misclassified.
These metrics provide a comprehensive view of both global performance (accuracy) and class-wise performance (precision, recall, F1-score), allowing the strengths and weaknesses of each model to be evaluated.

B. Quantitative Results

All three models Custom CNN, AlexNet, and ResNet18 were trained using identical splits and preprocessing pipelines. Table III summarizes their final performance on the test set.
Table III. Test Set Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score

Custom CNN	0.99	0.99	0.99	0.99
AlexNet	0.89	0.88	0.87	0.875
ResNet18	0.83	0.87	0.86	0.865

The results confirm that ResNet18 outperforms both AlexNet and MyCNN by a significant margin in every metric. AlexNet improves upon MyCNN due to its pretrained depth, but ResNet18's residual architecture and fine-tuned layers yield the best overall performance.

Confusion matrix for ResNet18, Alexnet and MyCNN model showing class-wise predictions.

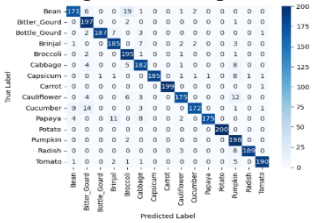


Fig 3 : MyCNN confusion matrix

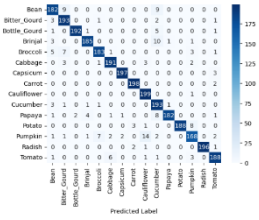


Fig 4 : Alexnet confusion matrix

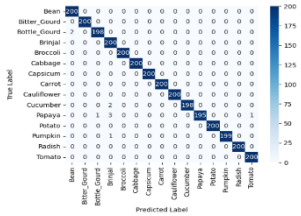
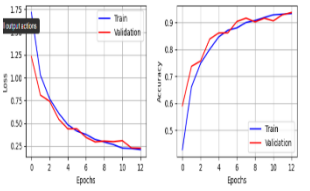


Fig 5 ResNet18 Confusion Matric

Training and Validation Performance Curves for MyCNN, AlexNet and ResNet18



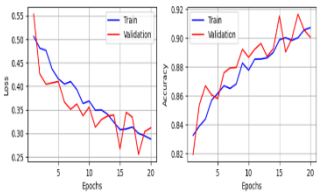


Fig. 6. Performance curve for MyCNN **Fig. 7. Performance curve for AlexNet**

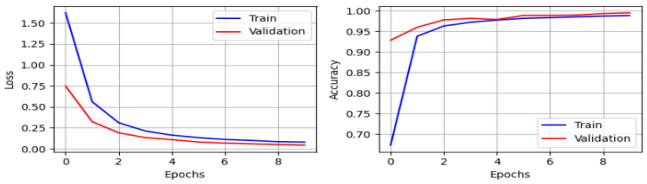


Fig. 8. Performance curve for ResNet18

The training and validation curves (Figs. 6-8) confirm that ResNet18 achieved fast and stable convergence. Both loss and accuracy curves showed smooth progression with minimal divergence, indicating excellent generalization [9]. The model reached over 99% validation accuracy by epoch 9, outperforming baseline models without any signs of overfitting. The confusion matrix in Figs. 3-5 shows nearly perfect classification across all 15 classes, with minimal misclassifications. Custom CNN showed reasonable performance but struggled on visually similar classes such as cabbage vs. cauliflower. AlexNet performed better than the custom CNN but was slower to train and less stable in convergence.

C. Qualitative Results

To visualize model behaviour, confusion matrices were generated for all three architectures. As shown in Fig. 3, the MyCNN confusion matrix reveals substantial misclassifications, especially among green vegetables with similar appearance, such as cabbage, lettuce, and cauliflower. Fig. 4, representing AlexNet, shows greater diagonal dominance and improved precision, although some off-diagonal misclassifications persist. Fig. 5 displays the confusion matrix for ResNet18, which demonstrates near-perfect class-wise accuracy, with virtually all predictions aligning on the diagonal.

Training and validation performance curves also provide insight into model convergence and generalization. As seen in Fig. 6, MyCNN showed stable convergence but plateaued at a lower accuracy ceiling. AlexNet (Fig. 7) trained faster and reached higher accuracy, but with more fluctuation in the validation phase. ResNet18 (Fig. 8) achieved the best learning behaviour, with both training and validation accuracy rapidly converging above 99%, and validation loss decreasing steadily. These trends reflect excellent generalization without signs of overfitting [9].

D. Comparisons

The combined metric results, confusion matrices, and learning curves highlight a clear performance hierarchy across the models. MyCNN, although lightweight and efficient, lacked sufficient depth to extract high-level features. AlexNet, with pretrained weights and a deeper structure, performed better but required more resources and still showed generalization gaps. ResNet18 demonstrated superior performance by leveraging residual connections and fine-tuned layers, allowing it to learn complex visual patterns while avoiding vanishing gradients [6].

In addition to metric-based evaluation, model behaviour was analyzed visually using confusion matrices (Figs. 3-5) and training performance curves (Figs. 6-8). The confusion matrix for MyCNN (Fig. 3) highlights several misclassified classes. AlexNet (Fig. 4) shows more consistent class separation. ResNet18 (Fig. 5) achieves nearly flawless predictions across all 15 classes. Performance curves further confirm that ResNet18 learned efficiently and generalized reliably [9]. MyCNN (Fig. 6) and AlexNet (Fig. 7) showed acceptable learning patterns but did not match the consistency or accuracy of ResNet18 (Fig. 8). These combined findings confirm ResNet18 as the most effective and deployment-ready model in this study.

V. CONCLUSION

This project demonstrated the effectiveness of deep learning models in classifying vegetable images under real-world

conditions. Among the three architectures tested, ResNet18 achieved the highest accuracy and generalization performance, clearly outperforming AlexNet and the custom CNN. Its residual learning and pretrained weights enabled fast convergence and near-perfect classification. The findings confirm that with proper augmentation, preprocessing, and transfer learning, deep CNNs like ResNet18 can be deployed in practical agricultural applications for automated sorting and inventory systems [2]. Future work will focus on lightweight deployment, model interpretability, and dataset expansion to further enhance robustness.

REFERENCES

[1] M. I. Ahmed, S. M. Mamun, and A. U. Z. Asif, "DCNN-based vegetable image classification using transfer learning: A comparative study," in Proc. 5th Int. Conf. Comput., Commun. Signal Process. (ICCCSP), 2021. [Online]. Available: https://www.researchgate.net/publication/379243877

[2] F. Xiao, H. Wang, Y. Xu, and R. Zhang, "Fruit detection and recognition based on deep learning for automatic harvesting: An overview and review," Agronomy, vol. 13, no. 6, p. 1625, 2023. [Online]. Available: https://www.mdpi.com/2073-4395/13/6/1625

[3] M. M. Rahman, M. M. Hasan, A. A. Rahman, M. M. Hossain, and A. A. Mamun, "A deep CNN approach to detect and classify local fruits through a web interface," Smart Agric. Technol., vol. 5, p. 100321, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772375523001508

[4] V. Annepu, K. Bagadi, V. R. R. Chirra, Y. Prashanth, A. Nikhil, and A. K. Chaitanya, "A hybrid deep learning approach for accurate and efficient classification for vegetable recognition," in Proc. 11th Int. Conf. Comput. Sustain. Global Dev. (INDIACom), 2024. [Online]. Available: https://www.researchgate.net/publication/379936575

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in Neural Inf. Process. Syst., vol. 25, pp. 1097 - 1105, 2012. [Online]. Available: https://papers.nips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770-778. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

[7] M. Islam, "Vegetable Image Dataset," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/minhazulislam/vegetable-image-dataset

[8] L. Alzubaida, J. Zhang, A. Humaidi, M. Qaisar, and F. R. Yu, "Structure of ResNet model and its applications in object classification: A survey," ACM Comput. Surv., vol. 55, no. 10s, pp. 1 -36, Dec. 2023. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3580334

[9] S. K. Wagh and M. S. Patil, "Convolutional neural networks for image classification: A survey," Mater. Today Proc., vol. 72, pp. 2902 -2906, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785323016497