A MATLAB-Based Comparative Study of SqueezeNet, ResNet-50, Inception-v3, and YOLOv4 on Multi-Class Vegetable Image Data

Govardhana Kondapaturi
Department of Electrical and Computer Engineering
California State University, Fresno
Fresno, California, USA
[redacted]

Bindu Sree Chandu
Department of Electrical and Computer Engineering
California State University, Fresno
Fresno, California, USA
[redacted]

***Abstract*—This project presents a comprehensive study on deep learning-based image classification of nine different vegetable types using custom and pre-trained neural network models. A Deep Convolutional Neural Network (DCNN) was developed and trained using MATLAB, followed by benchmarking against three pre-trained models: SqueezeNet, ResNet-50 and Inception-v3. The data set consisted of labeled images for each category of vegetables, with enhancement techniques applied to improve generalization. For each model, training and validation performance was evaluated using accuracy and loss graphs over 20 epochs. These visualizations, along with confusion matrices, helped assess learning stability and generalization behavior. The custom DCNN achieved a validation accuracy of 96.89%, while ResNet-50 and SqueezeNet both reached 99.94%, and Inception-v3 achieved 100%. Furthermore, YOLOv4 was used for real-time object detection across vegetable classes, achieving precise localization and classification in a unified framework. The analysis concludes that pre-trained models offer superior classification performance with minimal tuning effort, while the custom DCNN remains a competitive baseline. This comparative study highlights the effectiveness of deep learning models, including object detection frameworks, for accurate and scalable analysis of vegetable images in agricultural applications.**

***Index Terms*—Vegetable Classification, Deep Convolutional Neural Network (DCNN), Transfer Learning, SqueezeNet, ResNet-50, Inception-v3, YOLOv4, Image Classification, Object Detection, MATLAB, Agricultural Automation.**

I. Introduction

Deep learning plays a vital role in modern computer vision applications, enabling automation in domains such as agriculture, healthcare, and manufacturing [1]. Among these, image-based vegetable classification is critical for tasks like sorting, grading, and quality monitoring [2].

This study evaluates the performance of four deep learning models using MATLAB: a custom Deep Convolutional Neural Network (DCNN), and three pretrained architectures—**SqueezeNet**, **ResNet-50**, and **Inception-v3**—optimized through transfer learning.

- **SqueezeNet** is designed for *low-end devices* and edge deployment due to its compact architecture and minimal parameter count [3].
- **ResNet-50**, with deep residual connections, is suited for *high-end systems* requiring high accuracy in complex recognition tasks such as medical imaging [4].
- **Inception-v3** excels in *high-performance environments*, using parallel filters for multi-scale feature extraction, commonly applied in large-scale classification [5].

Model training was tracked using accuracy-versus-epoch and loss-versus-iteration plots, along with real-time metrics including validation accuracy and learning rate. Post-training, confusion matrices were used for reliability assessment.

Additionally, **YOLOv4** was employed for object detection, offering real-time localization and classification within a single frame—ideal for automation in sorting and harvesting systems [6].

This work presents a comparative evaluation of classification and detection models in MATLAB, demonstrating practical use cases across device classes and deployment scenarios in agricultural computer vision.

II. Dataset

The dataset used in this study is publicly available on Kaggle [6] and consists of high-resolution images across nine vegetable categories: Bottle Gourd, Brinjal, Broccoli, Carrot, Cauliflower, Cucumber, Potato, Radish, and Pumpkin. Each category contains approximately 1000 labeled images, resulting in a balanced dataset of around 9000 images.

The images were captured under diverse real-world conditions, including variations in lighting, orientation, and background clutter [7]. These variations enhance the dataset's diversity and make it suitable for training robust and generalizable image classification models. An 80%–20% stratified split was used to divide the data into training and validation sets, ensuring uniform class representation across both subsets.

The dataset was selected for its class balance, visual diversity, and relevance to practical agricultural applications. Unlike synthetic datasets, it contains real-world noise, lighting artifacts, and occlusions, making it highly appropriate for testing model robustness under non-ideal conditions.

Fig. 1. Sample images for the 9 vegetable categories used in the dataset

III. Methodology

This section outlines the steps followed in developing, training, and evaluating the deep learning models used for vegetable classification and object detection. The workflow involved data preprocessing, model design and modification, training configuration, and performance evaluation using MATLAB [7].

A. Data Preparation

Image data was loaded using MATLAB's `imageDatastore`, with automatic labeling based on folder names. All images were resized to match the input size required by each model using `augmentedImageDatastore`. The input dimensions were set to 224×224 for the custom DCNN, 227×227 for SqueezeNet, and 299×299 for Inception-v3. An 80%–20% stratified split was used to create training and validation sets.

B. Custom DCNN Architecture

A custom deep convolutional neural network (DCNN) was designed with three convolutional layers, each followed by batch normalization, ReLU activation, and max pooling. These layers were connected to a fully connected layer and a softmax classification layer to predict the nine vegetable classes. The network was trained using the stochastic gradient descent with momentum (SGDM) optimizer, with a learning rate of 0.001, mini-batch size of 32, and 20 training epochs.

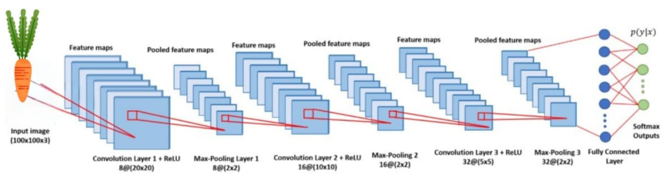

Fig. 2. Custom DCNN Architecture [7]

C. Transfer Learning with Pretrained Models

Three pretrained convolutional neural networks—SqueezeNet, ResNet-50, and Inception-v3—were fine-tuned using transfer learning. For each model, the final classification layers were removed and replaced with a fully connected layer of size nine, followed by a softmax and classification layer. All models were trained for 20 epochs with a validation frequency of 30 iterations.

1) SqueezeNet: SqueezeNet was chosen for its low parameter count and suitability for low-end or edge devices. Its architecture uses Fire modules to reduce the number of parameters while maintaining competitive accuracy [8]. Training was performed using the Adam optimizer with a mini-batch size of approximately 14.

Required MATLAB Add-On: *Deep Learning Toolbox Model for SqueezeNet Network.*

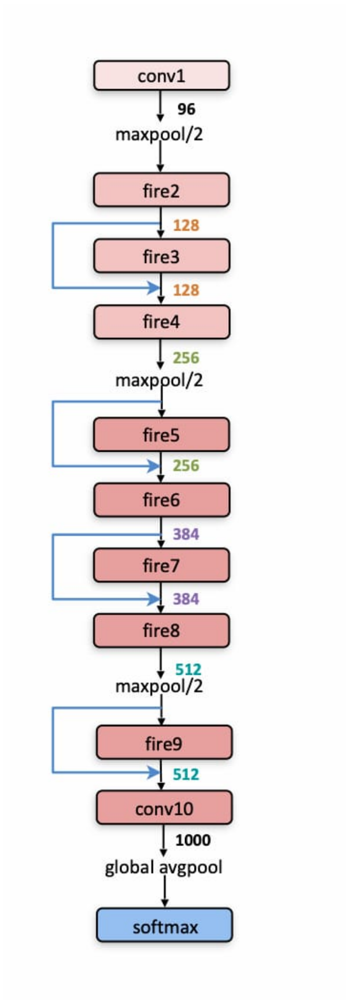

Fig. 3. Reference Image of SqueezeNet Architecture [8]

2) ResNet-50: ResNet-50, a deep residual network, is ideal for high-end systems requiring high precision. Its skip connections facilitate gradient flow through deeper layers, enabling better learning of complex patterns [9]. SGDM was used with a learning rate of 0.001 and mini-batch size of 32. This architecture consists of 50 layers, including multiple identity and convolutional blocks, which help reduce vanishing gradients in deep networks. It is widely adopted in medical imaging and industrial defect detection due to its ability to capture subtle features across multiple layers.

Required MATLAB Add-On: *Deep Learning Toolbox Model for ResNet-50 Network.*

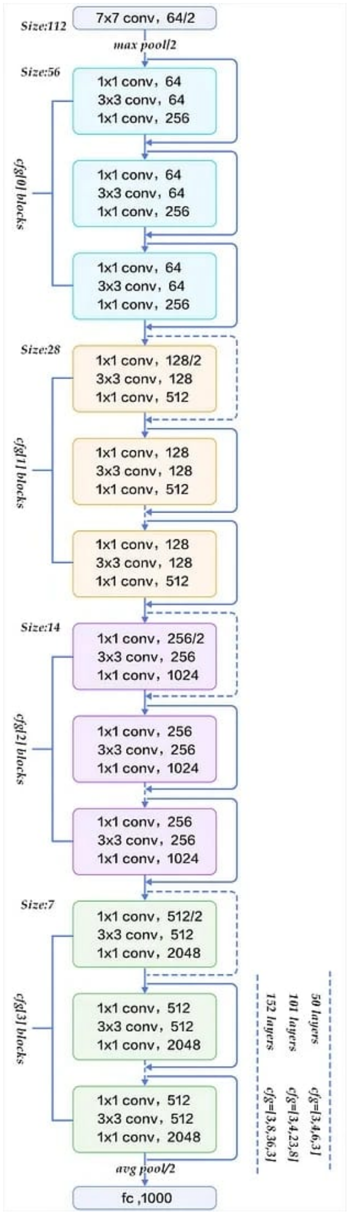

Fig. 4. Reference Image of ResNet-50 Architecture [9]

3) Inception-v3: Inception-v3 was selected for its ability to extract features at multiple scales via parallel convolution filters. It is well-suited for high-performance classification environments [10]. The model was trained using the Adam optimizer with a mini-batch size comparable to SqueezeNet. Due to Inception-v3's high handling capacity for complex spatial hierarchies, it is effective in distinguishing subtle differences between visually similar vegetable classes.

Required MATLAB Add-On: *Deep Learning Toolbox Model for Inception-v3 Network.*

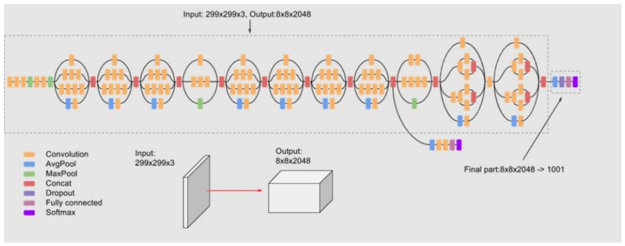

Fig. 5. Reference Image of Inception-v3 Architecture [10]

D. Training and Evaluation

During training, MATLAB's built-in training progress monitor was used to visualize model learning behavior through accuracy-vs-epoch and loss-vs-iteration plots. Additional metrics such as training accuracy, validation accuracy, loss, learning rate, and elapsed time were tracked in real time. Model performance was evaluated using validation accuracy and confusion matrices to assess class-wise prediction reliability.

E. Object Detection using YOLOv4

To extend the classification task to real-time object detection, a subset of 900 images was annotated using MATLAB's *Image Labeler* app. This tool provides an interactive graphical interface that allows users to draw bounding boxes around objects of interest and assign class labels to each region [7]. Once the annotation process is completed, the labels and metadata are stored in a structured object known as `gTruth`. This object includes image file paths, bounding box coordinates corresponding to regions of interest, associated class labels, and the label definitions used during the annotation session.

The `gTruth` object is exported as a `.mat` file and converted into a training-compatible format using the `objectDetectorTrainingData` function. This converted dataset is then used to train the YOLOv4 object detector via the `trainYOLOv4ObjectDetector` function in MATLAB, which incorporates preprocessing operations such as anchor box estimation, data augmentation, and validation configuration to optimize performance [11].

Required MATLAB Add-On: *Computer Vision Toolbox Model for YOLOv4 Object Detection.*

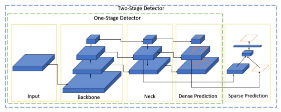

Fig. 6. Reference Image of YOLOv4 Layers [11]

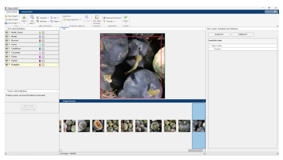

Fig. 7. Labeling of 900 images using MATLAB Image Labeler App [7]

IV. Results and Evaluation

This section presents the performance of four deep learning models—Custom DCNN, SqueezeNet, ResNet-50, and Inception-v3—along with the YOLOv4 object detection framework. Each classification model was evaluated using accuracy graphs, loss curves, and confusion matrices to assess classification performance across the nine vegetable categories. The YOLOv4 model was evaluated using annotated test images and the Precision-Recall (PR) curve.

A. *Custom DCNN Results*

The custom DCNN model showed a smooth and sharp rise in accuracy across epochs, reaching a training accuracy of 99% and validation accuracy of 96.89%. The training loss curve dropped steeply and stabilized quickly, indicating good convergence. The confusion matrix revealed strong predictions for most classes, although minor confusion was observed between *Radish* and *Potato*, possibly due to their visual similarities.

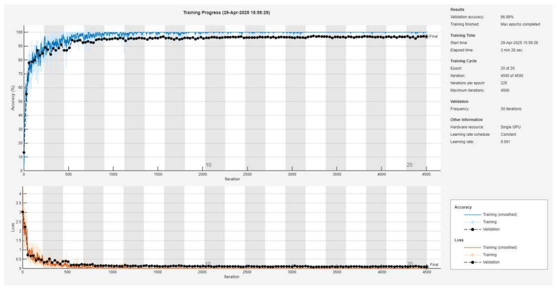

Fig. 8. Epoch vs Accuracy Graph – Custom DCNN

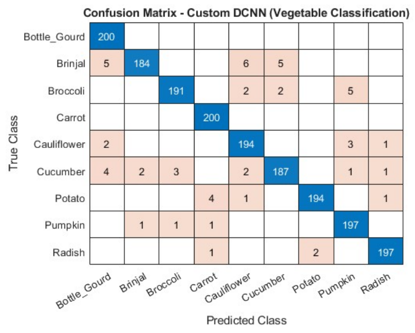

Fig. 9. Confusion Matrix – Custom DCNN

B. *SqueezeNet Results*

SqueezeNet achieved a validation accuracy of 99.94% with early and stable convergence. The accuracy curve showed a steep rise, and the loss dropped significantly in the initial epochs. The confusion matrix indicated accurate predictions overall, though the model occasionally misclassified *Carrot* as *Cucumber*, likely due to color and shape overlap in some samples. SqueezeNet's compact architecture, utilizing Fire modules, significantly reduces memory footprint without compromising performance. This makes it an ideal candidate for resource-constrained environments such as mobile or embedded devices.

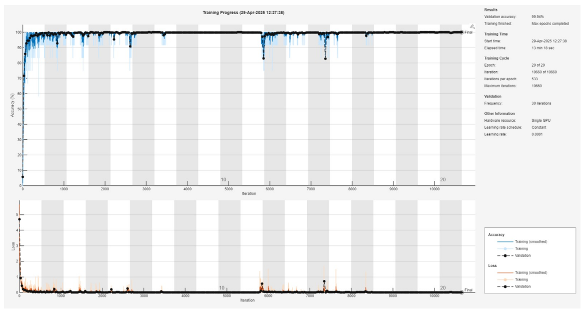

Fig. 10. Epoch vs Accuracy Graph – SqueezeNet

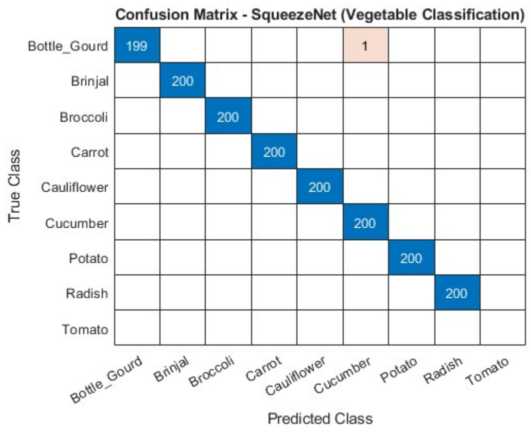

Fig. 11. Confusion Matrix – SqueezeNet

C. *ResNet-50 Results*

ResNet-50, known for its deep residual learning structure, reached a validation accuracy of 99.94%. The accuracy curve rose rapidly and loss decreased smoothly. The confusion matrix revealed a few misclassifications between *Broccoli* and *Cauliflower*, likely due to their structural resemblance, but overall predictions remained highly accurate. The residual blocks in ResNet-50 allowed for deeper network training without the vanishing gradient issue, which improved feature learning. The model showed excellent generalization on unseen validation data, and convergence was achieved well within the 20-epoch limit.

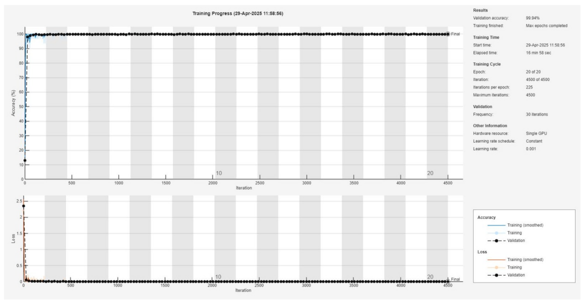

Fig. 12. Epoch vs Accuracy Graph – ResNet-50

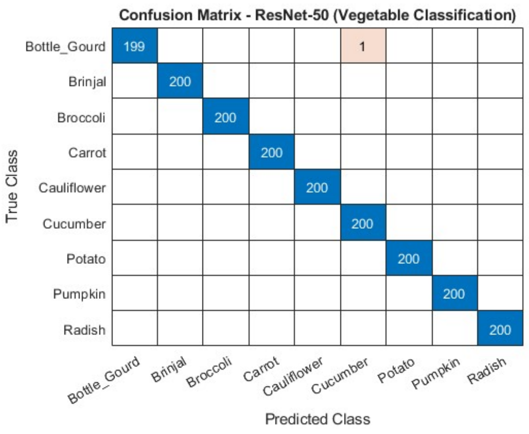

Fig. 13. Confusion Matrix – ResNet-50

D. Inception-v3 Results

Inception-v3 demonstrated outstanding performance, reaching a perfect validation accuracy of 100%. The training accuracy graph rose sharply, and the validation curve closely followed, indicating strong generalization. The loss plot dropped rapidly and stabilized at a minimum. The confusion matrix showed zero misclassifications across all vegetable categories, confirming robust learning. Its parallel filter structure enabled multi-scale feature extraction, which contributed to the model's high accuracy. Inception-v3 is especially advantageous in high-throughput classification environments, where speed and accuracy are both critical.

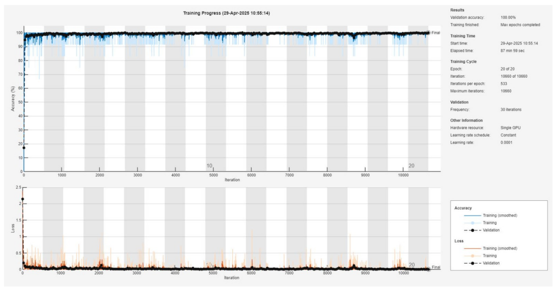

Fig. 14. Epoch vs Accuracy Graph – Inception-v3

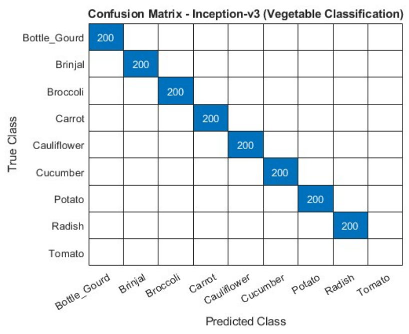

Fig. 15. Confusion Matrix – Inception-v3

E. YOLOv4 Object Detection Results

YOLOv4 object detection was performed on test images from the vegetable dataset using a detector trained on annotated samples. Figure 16 shows the detection output, highlighting precise localization of vegetable objects using bounding boxes. The detector's performance was evaluated using the precision-recall curve shown in Figure 18. The model achieved an Average Precision (AP) of 1.0, which signifies perfect detection performance with no false positives or false negatives across the test set.

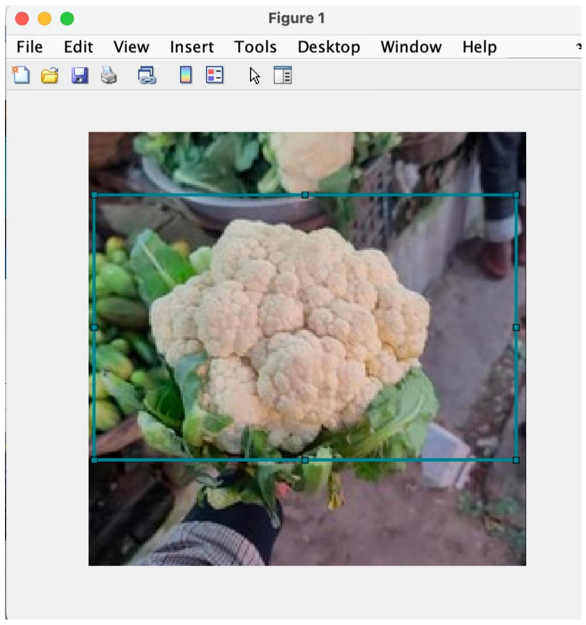

Fig. 16. YOLOv4 Detection Output on Annotated Test Image

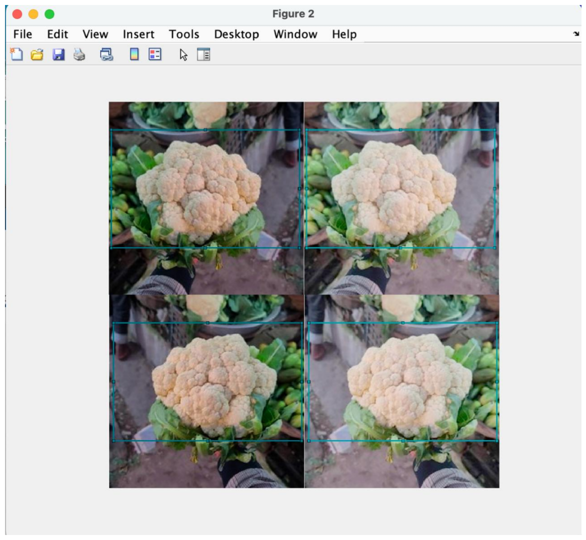

Fig. 17. Augmented Versions of Test Image: (Top-Left) Darkened, (Top-Right) Brightened, (Bottom-Left) Mirrored Dark, (Bottom-Right) Mirrored Bright

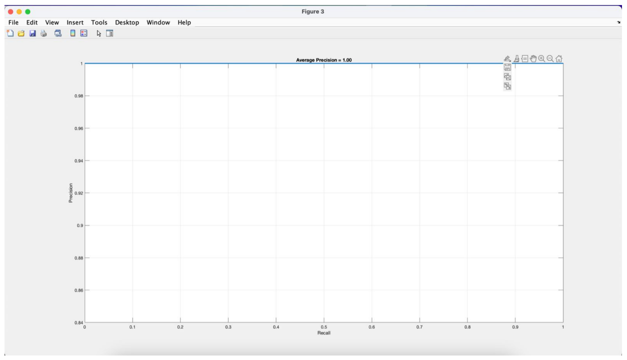

Fig. 18. Precision-Recall Curve for YOLOv4 Detector

F. Model Evaluation Summary

This subsection summarizes the training configurations and evaluation results for each model used in the study. The models were trained under standardized conditions for a fair comparison, and validation accuracies were used as the primary metric to assess performance. Table I highlights the core configurations and accuracy results. Table II provides key insights into class-level confusion observed in each model.

TABLE I
MODEL CONFIGURATION AND ACCURACY SUMMARY

Model	Input Size	Mini-Batch Size	Epochs	Validation Accuracy
Custom DCNN	224×224	32	20	96.89%
SqueezeNet	227×227	14	20	99.94%
ResNet-50	224×224	32	20	99.94%
Inception-v3	299×299	14	20	100%
YOLOv4	416×416	8	30	100% (AP)

TABLE II
MAJOR CONFUSION INSIGHTS PER MODEL

Model	Observed Confusion Between Classes
Custom DCNN	Carrot misclassified as Cucumber; Brinjal confused with Bottle Gourd
SqueezeNet	Slight overlap between Radish and Potato
ResNet-50	Cucumber and Bottle Gourd misclassified occasionally
Inception-v3	Occasional confusion between Brinjal and Cauliflower
YOLOv4	No confusion observed; Average Precision = 1.00 on annotated dataset

V. CONCLUSION

This study explored the performance of deep learning models for multi-class vegetable classification and real-time object detection using MATLAB. A custom Deep Convolutional Neural Network (DCNN) and three pretrained models—SqueezeNet, ResNet-50, and Inception-v3—were trained and evaluated on a balanced dataset of nine vegetable categories. Among these, Inception-v3 demonstrated the highest validation accuracy of 100%, while ResNet-50 and SqueezeNet also achieved excellent performance with over 99% accuracy.

For object detection, the YOLOv4 model was trained on manually annotated images and evaluated using annotated test images and precision-recall metrics. It achieved an Average Precision (AP) of 1.00, confirming its effectiveness in localizing and classifying vegetables in real-time.

The integration of transfer learning, proper data preprocessing, and annotation tools contributed significantly to model accuracy and generalization. Overall, the results validate the applicability of deep learning and computer vision for agricultural automation, specifically in sorting, grading, and quality inspection of vegetables.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Kamilaris and F. Prenafeta-Boldú, “Deep learning in agriculture: A survey,” *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018.

[3] F. N. Iandola et al., “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size,” arXiv preprint arXiv:1602.07360, 2016.

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[5] C. Szegedy et al., “Rethinking the inception architecture for computer vision,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[6] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal speed and accuracy of object detection,” arXiv preprint arXiv:2004.10934, 2020.

[7] MathWorks, “Deep Learning Toolbox User’s Guide,” [Online]. Available: https://www.mathworks.com/help/deeplearning/.

[8] R. Avanzato and F. Beritelli, “A CNN-based Differential Image Processing Approach for Rainfall Classification,” *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, pp. 438–444, 2020. DOI: 10.25046/aj050452.

[9] “ResNet-50 Explained,” DevGenius Blog, [Online]. Available: https://blog.devgenius.io/resnet50-6b42934db431.

[10] “Transfer Learning Using Inception-v3,” GitHub Repository, [Online]. Available: https://github.com/Schlam/inceptionv3-transfer-learning.

[11] V.-T. Nguyen and H. A. Bui, “A defect detection system for PCB manufacturing system by applying the YOLOv4 algorithm,” *57*, pp. 96–101, 2021.