Contents

FRUITS AND VEGETABLES CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS (CNN)

Tanzeela Asghar(032), Neha Amjad(024)

Department of Software Engineering, Fatima Jinnah Women University, Rawalpindi

Abstract

The classification of fruits and vegetables using Convolutional Neural Networks (CNN) is an essential task in computer vision and machine learning. This project explores the design, training, and evaluation of a CNN-based classification system that identifies 36 types of fruits and vegetables. Leveraging a dataset with 3,855 images, organized into training, validation, and testing subsets, the model achieved remarkable accuracy: 99.36% on the training set, 96.30% on the validation set, and 96.38% on the test set. The results validate the robustness and practical applicability of the model, offering potential for further research in dietary monitoring, inventory management, and agricultural applications.

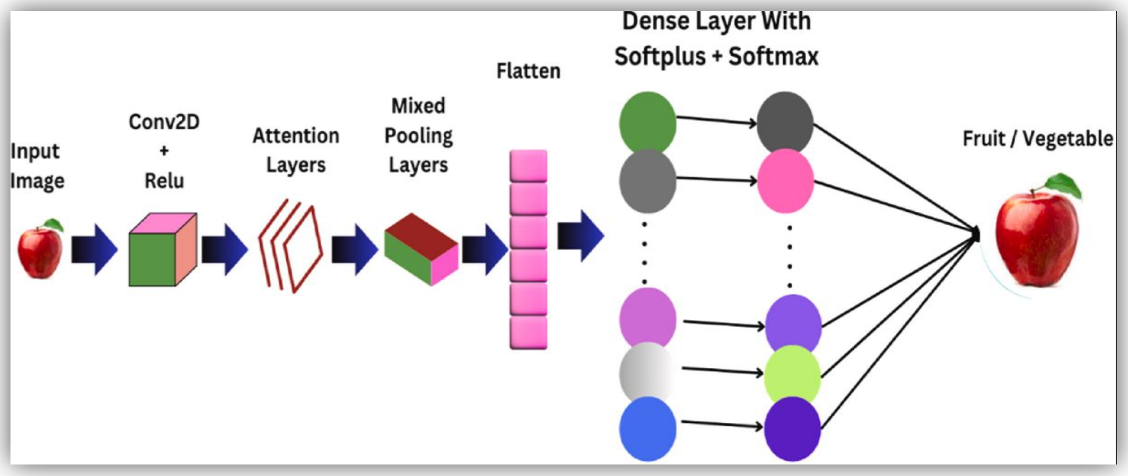
1. Introduction

Image recognition has emerged as a transformative technology across numerous industries, including retail, healthcare, and agriculture. Within these domains, the ability to automatically identify food items from images has opened up innovative solutions for dietary monitoring, grocery inventory management, recipe suggestions, and even nutritional analysis. As dietary awareness grows and automation technologies advance, food recognition systems are becoming increasingly vital in addressing global challenges related to food security, health, and sustainability.

The classification of fruits and vegetables is a particularly challenging task in food recognition due to the wide variety of items, variations in shape, color, texture, and size, as well as differences in lighting and background conditions during image capture. Traditional image recognition techniques often struggle with these complexities, leading to inaccuracies and limited applicability in real-world scenarios.

This project leverages Convolutional Neural Networks (CNNs), a class of deep learning algorithms known for their exceptional performance in image recognition tasks. CNNs can automatically learn hierarchical features from raw image data, such as edges, textures, and patterns, enabling precise classification without the need for manual feature engineering. The flexibility and robustness of CNNs make them well-suited for the intricate task of classifying 36 categories of fruits and vegetables, ranging from apples and bananas to cucumbers and tomatoes.

The primary motivation for this study is to design an efficient, accurate, and scalable system that can reliably identify food items from photographs. Such a system has significant potential for applications in:
Supermarkets: Automating the categorization of produce items for inventory management and pricing.
Healthcare: Enabling dietary monitoring tools that analyze meal compositions for calorie and nutrient tracking.
Agriculture: Assisting in quality control and grading of produce.
By utilizing a diverse and well-organized dataset, this project also addresses critical challenges, such as class variability and overfitting, which often hinder the performance of image recognition models. A detailed evaluation of model performance, including metrics such as accuracy, precision, and loss, ensures the system's reliability for real-world deployment. Through the innovative application of CNNs, this study contributes to advancing the field of automated food recognition and sets a foundation for future research and development in related areas.



2. Literature Review

Several methodologies have been proposed for food classification using machine learning and CNNs. Lee et al. (2023) utilized multispectral imaging combined with CNNs to classify food items and estimate caloric content, achieving exceptional accuracy by fusing RGB, UV, and NIR data. Similarly, recent studies such as Ege et al. (2019) explored multitask CNNs for simultaneous food classification and caloric estimation, demonstrating the model's versatility. However, systems relying solely on RGB images face limitations in distinguishing visually similar items. This study extends the application of CNNs by focusing on fruits and vegetables, addressing these challenges through robust preprocessing and data augmentation tecnique.

3. Objectives

Design a CNN-based Model to Classify Fruits and Vegetables into 36 Distinct Categories: The primary objective of this project was to develop a **Convolutional Neural Network (CNN)** model capable of classifying images of fruits and vegetables into 36 distinct categories. This involved curating a dataset containing labeled images of different fruits and vegetables and training a deep learning model to automatically recognize and categorize them.

Achieve High Classification Accuracy through Optimized Network Architecture: The goal was to design an efficient CNN architecture that could achieve **high accuracy** in classifying images. This required selecting suitable layers, activation functions, and regularization techniques that balance **model complexity** and **generalization**, ensuring that the model performs well on unseen data and avoids overfitting.

Evaluate the Model's Performance Across Training, Validation, and Test Datasets: Another key objective was to evaluate the performance of the CNN model comprehensively across multiple datasets: **training**, **validation**, and **test** sets. This allowed for the assessment of how well the model learned during training, its ability to generalize to new data during validation, and its final accuracy on unseen data during testing.

Provide Rich Visualizations of Results, Predictions, and Misclassifications: To improve understanding and interpretability, the project aimed to generate clear **visualizations** of the model's performance. This included graphical representations of accuracy over epochs, confusion matrices, and visual depictions of misclassifications. These visualizations provide insights into how the model is making predictions and where it might be failing.

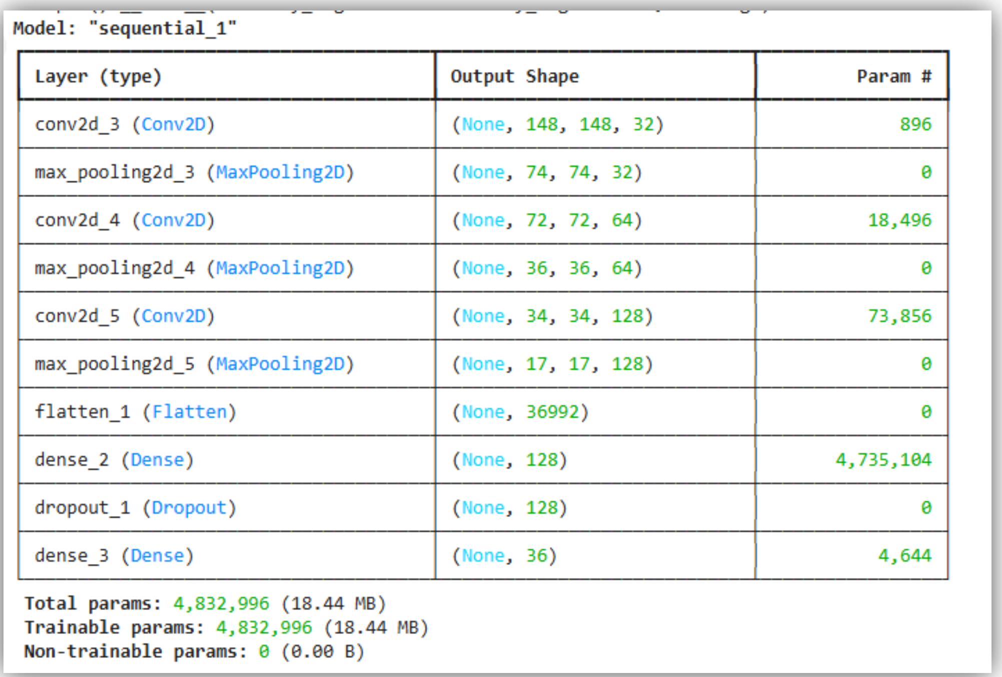
Limitation	**Description**
Limited Dataset Diversity	The dataset used for training and testing the model might not have captured the full diversity of fruits and vegetables found in real-world scenarios. Variations in lighting, backgrounds, and occlusions (e.g., fruits hidden by leaves) could affect the model's ability to generalize to real-world images. Additionally, the number of categories (36) may not have been extensive enough to handle all possible types of fruits and vegetables in real-world applications.
Overfitting Risk	While the CNN model achieved good accuracy, there was a risk of overfitting, especially when trained on a relatively small dataset. If the model becomes too complex, it could memorize the training data rather than learning to generalize to new, unseen data. Techniques such as data augmentation and dropout were applied to mitigate this risk, but it remains a limitation.
Limited Image Preprocessing	Although image resizing, normalization, and data augmentation were implemented, there

	may be additional preprocessing techniques (e.g., color space transformations, advanced noise reduction) that could improve the model’s robustness. The current preprocessing steps might not be sufficient to handle all real-world image variations.
Model Complexity and Training Time	The CNN architecture, while effective, could become computationally expensive and time-consuming, especially as the dataset size increases. Training deep learning models can require significant computational resources (such as GPUs) and time, which could be a limitation in terms of both cost and practicality for real-time applications.
Class Imbalance	The dataset could potentially suffer from class imbalance, where some categories have significantly more images than others. This could lead the model to favor the majority classes, making it less effective at recognizing underrepresented categories. Techniques such as class weighting or oversampling of minority classes could help address this, but the current model may still be biased toward the larger classes.
Lack of External Evaluation	While the model performed well on the internal datasets, its performance on external, unseen data has not been thoroughly evaluated. Real-world deployment may present additional challenges, such as differences in image quality or environmental conditions, which were not fully accounted for in the project.
Interpretability of Results	Although the project provided visualizations of results and misclassifications, deep learning models like CNNs are often criticized for their lack of interpretability. Understanding why a model makes a specific prediction is still a challenge, especially when dealing with complex models that involve many layers and

	parameters. This lack of transparency could hinder trust in the model's decisions, particularly in critical applications.

4. Proposed Network

The proposed network is a Convolutional Neural Network (CNN) implemented using TensorFlow. It consists of convolutional, pooling, and fully connected layers that extract hierarchical features from the images. Dropout layers are incorporated to mitigate overfitting, while the Adam optimizer is used for efficient training.

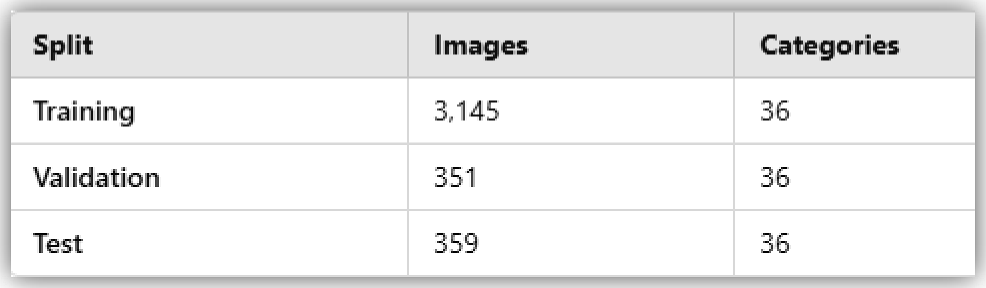


```
Model: "sequential_1"

Layer (type)                      Output Shape                  Param #
conv2d_3 (Conv2D)                 (None, 148, 148, 32)              896
max_pooling2d_3 (MaxPooling2D)    (None, 74, 74, 32)                  0
conv2d_4 (Conv2D)                 (None, 72, 72, 64)             18,496
max_pooling2d_4 (MaxPooling2D)    (None, 36, 36, 64)                  0
conv2d_5 (Conv2D)                 (None, 34, 34, 128)            73,856
max_pooling2d_5 (MaxPooling2D)    (None, 17, 17, 128)                 0
flatten_1 (Flatten)               (None, 36992)                       0
dense_2 (Dense)                   (None, 128)                 4,735,104
dropout_1 (Dropout)               (None, 128)                         0
dense_3 (Dense)                   (None, 36)                      4,644

Total params: 4,832,996 (18.44 MB)
Trainable params: 4,832,996 (18.44 MB)
Non-trainable params: 0 (0.00 B)
```

5. Proposed Methodology

a. Dataset

The dataset used for this project consists of 36 categories, with a balanced division between fruits and vegetables. The images are organized into three subsets: **Training Set**, **Validation Set**, and **Test Set**. The **Training Set** includes 3,145 images, with 100 images for each category. This provides a comprehensive dataset for the model to learn the essential features of the different fruits and vegetables. The **Validation Set**, with 351 images (10 per category), is used during the training process to evaluate model performance and adjust hyperparameters. The **Test Set**, consisting of 359 images (10 per category), is used to evaluate the final performance of the model, ensuring its generalization capability to unseen data. The dataset includes a variety of fruits, such as **Banana**, **Apple**, **Grapes**, and **Mango**, as well as vegetables like **Carrot**, **Potato**, **Tomato**, and **Cauliflower**. These images were sourced from Kaggle, which ensures high-quality samples under different lighting conditions, making the dataset highly varied and challenging for training purposes.



Split	Images	Categories
Training	3,145	36
Validation	351	36
Test	359	36

b. Image Preprocessing and Augmentation

Image Preprocessing is an essential step to prepare the dataset for training. Initially, all images are resized to 64x64 pixels to ensure uniformity in input dimensions. This is crucial because deep learning models require fixed-size inputs. Following resizing, pixel values are **normalized** to the range of [0,1]. This normalization standardizes the data and ensures that the model converges efficiently during training.

Data Augmentation plays a pivotal role in improving the generalization of the model. By artificially increasing the size of the training dataset, augmentation helps the model learn to recognize patterns that are invariant to certain transformations. The augmentations applied in this project include:

Random Rotations: The images are rotated randomly within a specified range to help the model become invariant to orientation.

Horizontal Flips: The images are flipped horizontally to allow the model to learn features that are orientation-invariant.

Zoom Adjustments: Random zooming in or out of the images helps the model recognize objects at different scales.

These augmentation techniques contribute significantly to increasing the variability of the training data, improving the model's ability to generalize to new, unseen images. By enhancing the dataset in this way, the model becomes more robust to real-world conditions, where lighting, object orientation, and scale can vary

c. CNN Architecture

The **Convolutional Neural Network (CNN) Architecture** proposed for this task is designed to efficiently process and classify images of fruits and vegetables. The architecture follows a hierarchical structure, where successive layers extract increasingly complex features from the input images. The network is comprised of the following components:

Convolutional Layers: These layers are the foundation of CNNs, used to extract spatial features from the input image. The first convolutional layer uses **32 filters**, followed by a second convolutional layer with **64 filters**. The filters are designed to capture different features such as edges, textures, and patterns, which are essential for distinguishing between the various fruit and vegetable categories.

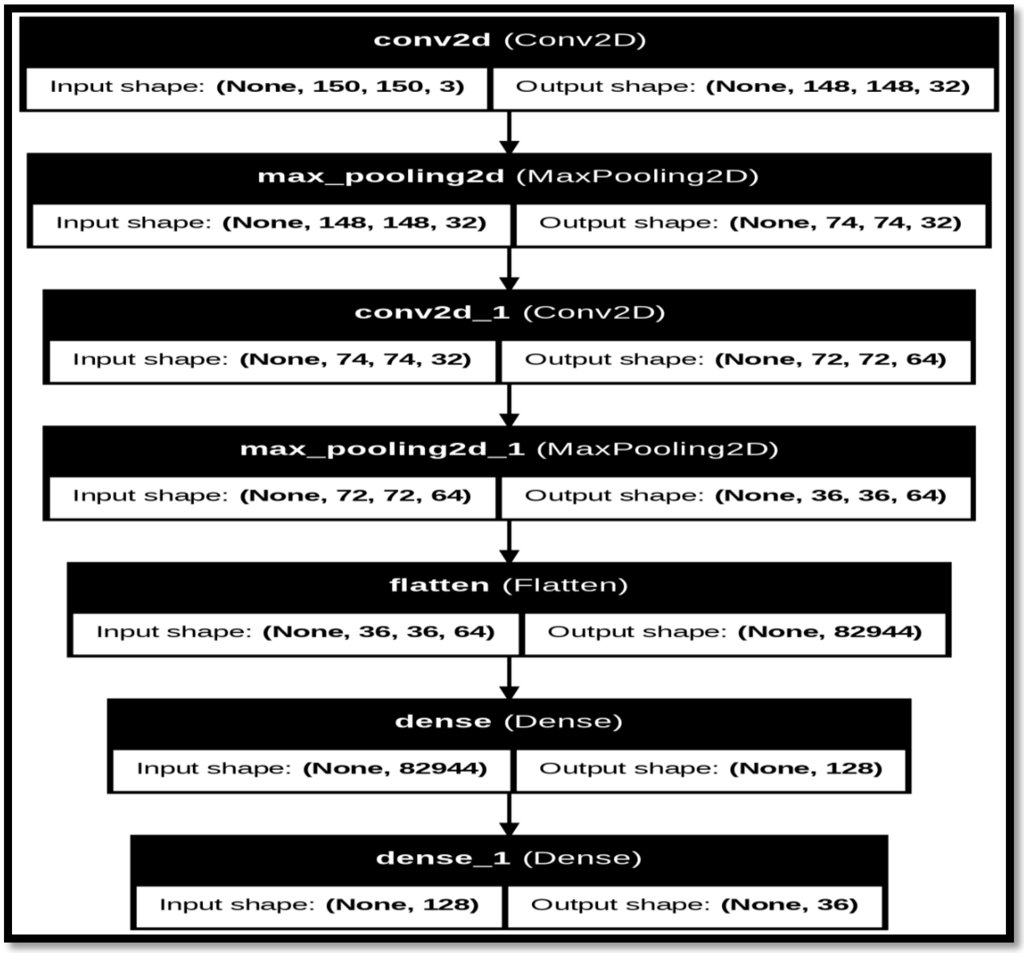
Activation Function (ReLU): The **Rectified Linear Unit (ReLU)** activation function is applied after each convolutional layer. ReLU introduces non-linearity into the model, allowing it to learn more complex patterns and better represent the underlying structure in the data.

Pooling Layers: After the convolutional layers, **max-pooling layers** are applied to reduce the spatial dimensions of the feature maps while retaining the most significant features. Max-pooling helps decrease the computational load and prevent overfitting by reducing the number of parameters in the model.

Fully Connected Layers: Following the convolutional and pooling layers, the model includes **dense (fully connected) layers**. The first dense layer consists of **512 units**, and the second dense layer has **256 units**. These layers serve to combine the features extracted by the convolutional layers and make the final decision about the class of the input image.

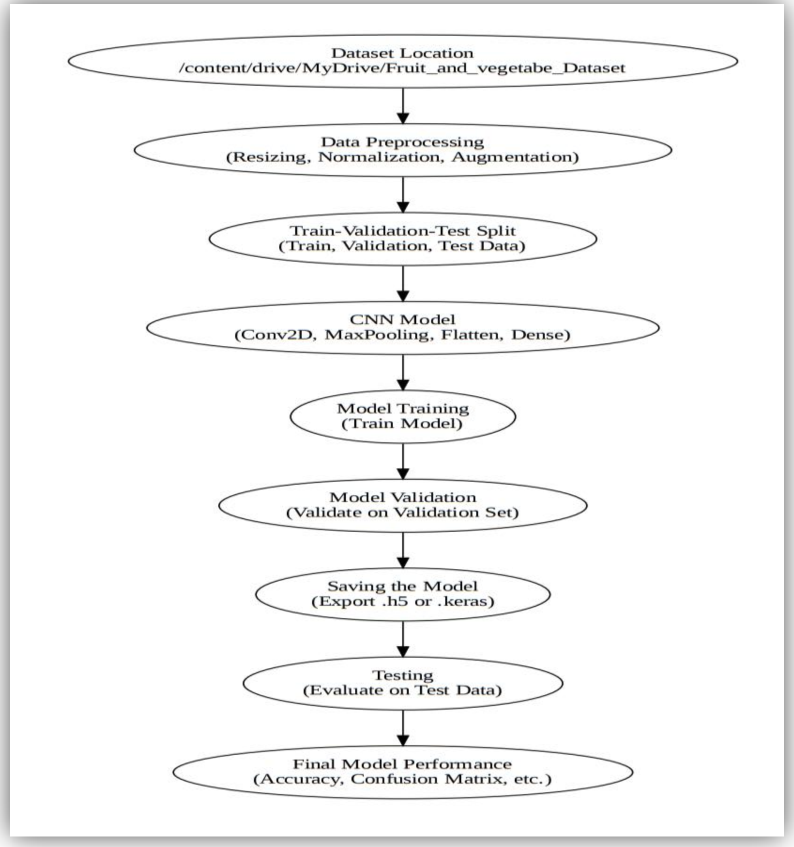
Dropout: Dropout is applied in the fully connected layers to prevent overfitting. By randomly deactivating a fraction of neurons during training, dropout encourages the model to generalize better rather than memorizing the training data.

Output Layer: The final layer in the architecture is a **softmax layer**, which is used for multi-class classification. The softmax function converts the raw output of the model into probabilities, where each probability corresponds to the likelihood that the image belongs to one of the 36 categories.



conv2d (Conv2D)
Input shape: (None, 150, 150, 3)
Output shape: (None, 148, 148, 32)
max_pooling2d (MaxPooling2D)
Input shape: (None, 148, 148, 32)
Output shape: (None, 74, 74, 32)
conv2d_1 (Conv2D)
Input shape: (None, 74, 74, 32)
Output shape: (None, 72, 72, 64)
max_pooling2d_1 (MaxPooling2D)
Input shape: (None, 72, 72, 64)
Output shape: (None, 36, 36, 64)
flatten (Flatten)
Input shape: (None, 36, 36, 64)
Output shape: (None, 82944)
dense (Dense)
Input shape: (None, 82944)
Output shape: (None, 128)
dense_1 (Dense)
Input shape: (None, 128)
Output shape: (None, 36)

d. Framework Diagram:


Dataset Location
/content/drive/MyDrive/Fruit_and_vegetabe_Dataset
Data Preprocessing
(Resizing, Normalization, Augmentation)
Train-Validation-Test Split
(Train, Validation, Test Data)
CNN Model
(Conv2D, MaxPooling, Flatten, Dense)
Model Training
(Train Model)
Model Validation
(Validate on Validation Set)
Saving the Model
(Export .h5 or .keras)
Testing
(Evaluate on Test Data)
Final Model Performance
(Accuracy, Confusion Matrix, etc.)

6. Experimental Results and Discussions

a. Model Training

The model for the fruit and vegetable classification task was trained for a total of **32 epochs**. An epoch refers to one complete pass through the entire dataset during training. During each epoch, the model processes all images in the training set, updating its weights to minimize the loss function. The model was trained with a **batch size of 32**, meaning that in each iteration, 32 images were passed through the network at once before the model's weights were updated. This batch size was chosen as a balance between computational efficiency and memory usage. Larger batch sizes often lead to faster training, while smaller batch sizes can result in more precise updates to the model weights but may take longer to train.

Loss Function: Categorical Cross-Entropy

For the classification task, **categorical cross-entropy** was used as the loss function. The categorical cross-entropy loss is a common choice for multi-class classification problems, where the goal is to assign each input image to one of several categories. The loss function compares the model's predicted probability distribution (output of the softmax layer) with the actual label (ground truth) and calculates the difference between them. Specifically, categorical cross-entropy computes the negative log of the

predicted probability corresponding to the true class label, penalizing the model more when it assigns lower probabilities to the correct label. The goal during training is to minimize this loss, which helps the model become more confident in its predictions.
In multi-class classification, where each input can belong to one of many categories, categorical cross-entropy is an effective loss function because it accounts for the probability distribution over all categories, ensuring that the model learns to assign higher probabilities to the correct class while learning to distinguish between all the possible categories.

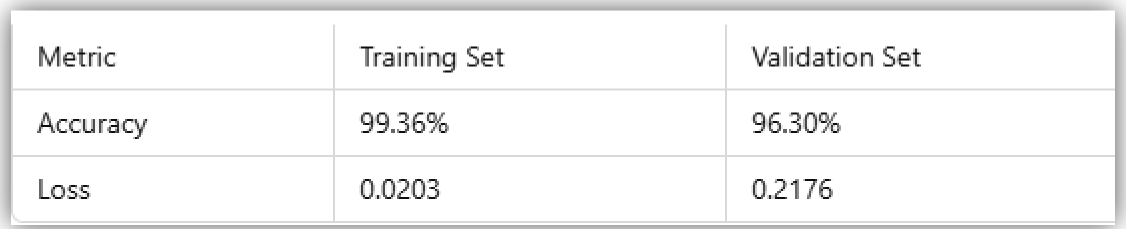
Evaluation Metric: Accuracy

The evaluation metric chosen for this project was **accuracy**, which measures the proportion of correct predictions made by the model. Specifically, it is calculated as the ratio of the number of correct predictions (where the predicted label matches the true label) to the total number of predictions made. Accuracy is a widely used metric for classification tasks as it provides an overall indication of the model's effectiveness in correctly classifying images.
For this project, accuracy was tracked on both the **training** and **validation** sets throughout the training process. The **training accuracy** reflects how well the model is learning from the training data, while the **validation accuracy** shows how well the model is generalizing to unseen data during the training process.

Training and Validation Accuracy

The **training accuracy** and **validation accuracy** both improved consistently across the 32 epochs, indicating that the model was learning effectively and becoming better at classifying images as training progressed.

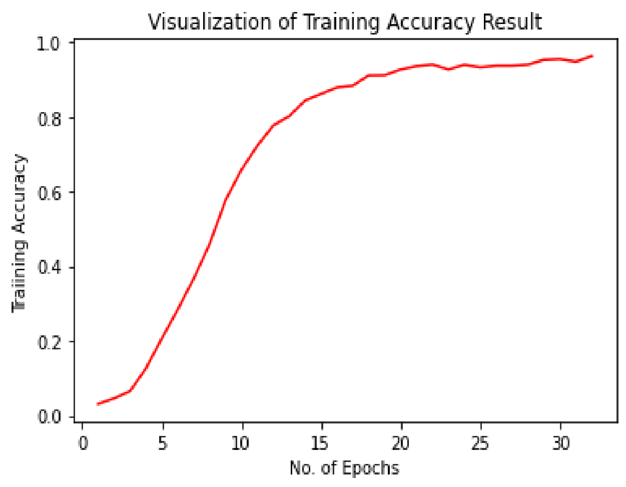
Performance Metrics

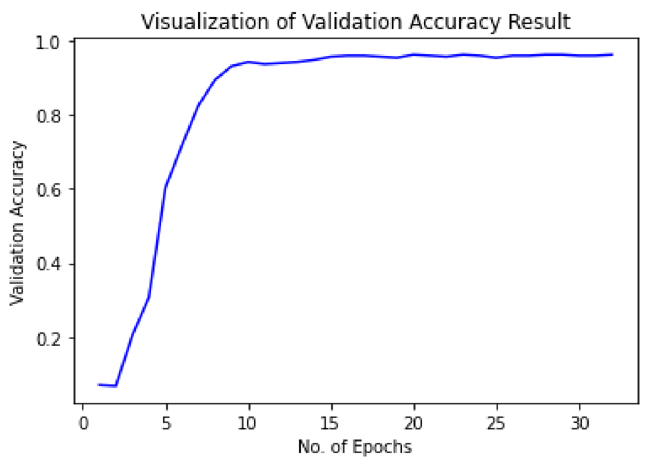


Metric	Training Set	Validation Set
Accuracy	99.36%	96.30%
Loss	0.0203	0.2176

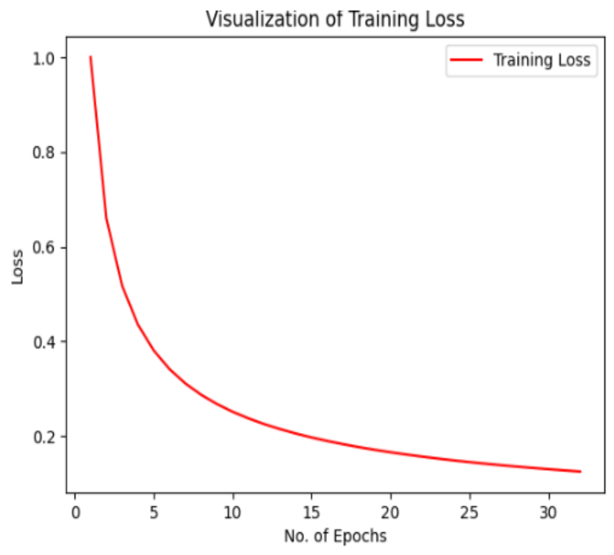
Accuracy Graphs

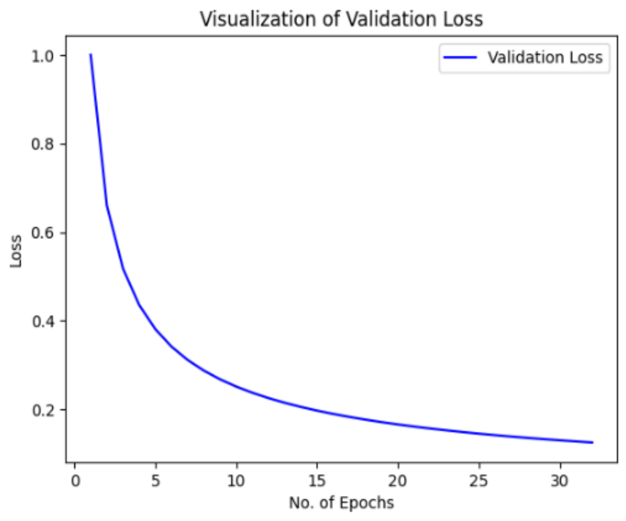
Training and Validation Accuracy Over Epochs





Training Loss Over Epochs



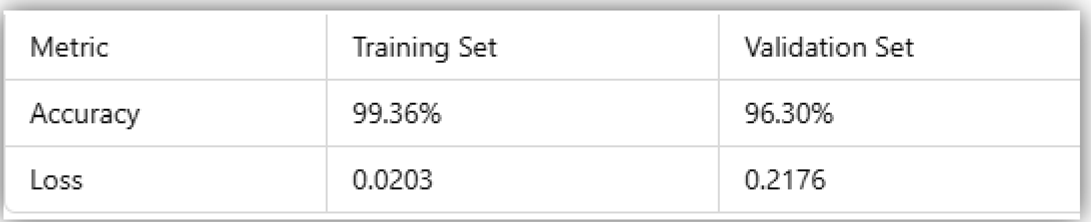


7. Test Set Evaluation

a. Test Set Evaluation

The **Test Set Evaluation** is a critical step in assessing the final performance of the model after it has been trained and validated. The test set consists of **359 images** (10 images per category) that the model has not seen during the training or validation process. It serves as an unbiased evaluation of the model's ability to generalize to new, unseen data. The performance of the model on the test set provides an estimate of how well the model will perform in a real-world scenario, where data from a similar distribution is expected but may vary in terms of lighting, angle, or other factors.

b. Performance Metrics:

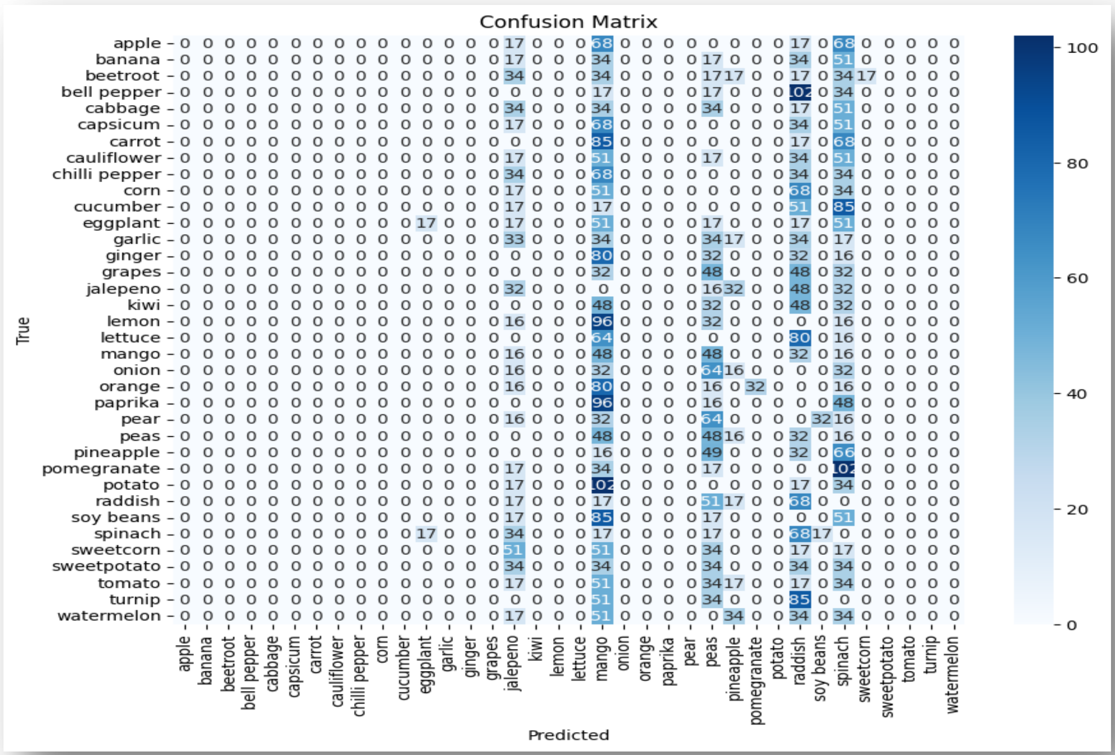


Metric	Training Set	Validation Set
Accuracy	99.36%	96.30%
Loss	0.0203	0.2176

c. Confusion Matrix

A confusion matrix was generated to visualize the classification performance across all 36 categories. Misclassifications were minimal, highlighting the model's robustness.

Confusion Matrix Heatmap



True \ Predicted	apple	banana	beetroot	bell pepper	cabbage	capsicum	carrot	cauliflower	chilli pepper	corn	cucumber	eggplant	garlic	ginger	grapes	jalepeno	kiwi	lemon	lettuce	mango	onion	orange	paprika	pear	peas	pineapple	pomegranate	potato	raddish	soy beans	spinach	sweetcorn	sweetpotato	tomato	turnip	watermelon
apple	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	68	0	0	0	0	0	0	0	0	17	0	68	0	0	0	0	0
banana	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	34	0	0	0	0	17	0	0	0	34	0	51	0	0	0	0	0
beetroot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	34	0	0	0	0	17	17	0	0	17	0	34	17	0	0	0	0
bell pepper	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	17	0	0	0	102	0	34	0	0	0	0	0
cabbage	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	34	0	0	0	0	34	0	0	0	17	0	51	0	0	0	0	0
capsicum	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	68	0	0	0	0	0	0	0	0	34	0	51	0	0	0	0	0
carrot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85	0	0	0	0	0	0	0	0	17	0	68	0	0	0	0	0
cauliflower	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	51	0	0	0	0	17	0	0	0	34	0	51	0	0	0	0	0
chilli pepper	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	68	0	0	0	0	0	0	0	0	34	0	34	0	0	0	0	0
corn	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	51	0	0	0	0	0	0	0	0	68	0	34	0	0	0	0	0
cucumber	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	17	0	0	0	0	0	0	0	0	51	0	85	0	0	0	0	0
eggplant	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	17	0	0	0	51	0	0	0	0	17	0	0	0	17	0	51	0	0	0	0	0
garlic	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	34	0	0	0	0	34	17	0	0	34	0	17	0	0	0	0	0
ginger	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	80	0	0	0	0	32	0	0	0	32	0	16	0	0	0	0	0
grapes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	48	0	0	0	48	0	32	0	0	0	0	0
jalepeno	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	16	32	0	0	48	0	32	0	0	0	0	0
kiwi	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	32	0	0	0	48	0	32	0	0	0	0	0
lemon	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	96	0	0	0	0	32	0	0	0	0	0	16	0	0	0	0	0
lettuce	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	64	0	0	0	0	0	0	0	0	80	0	16	0	0	0	0	0
mango	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	48	0	0	0	0	48	0	0	0	32	0	16	0	0	0	0	0
onion	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	32	0	0	0	0	64	16	0	0	0	0	32	0	0	0	0	0
orange	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	80	0	0	0	0	16	0	32	0	0	0	16	0	0	0	0	0
paprika	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96	0	0	0	0	16	0	0	0	0	0	48	0	0	0	0	0
pear	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	32	0	0	0	0	64	0	0	0	0	32	16	0	0	0	0	0
peas	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	48	16	0	0	32	0	16	0	0	0	0	0
pineapple	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	49	0	0	0	32	0	66	0	0	0	0	0
pomegranate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	34	0	0	0	0	17	0	0	0	0	0	102	0	0	0	0	0
potato	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	102	0	0	0	0	0	0	0	0	17	0	34	0	0	0	0	0
raddish	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	17	0	0	0	0	51	17	0	0	68	0	0	0	0	0	0	0
soy beans	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	85	0	0	0	0	17	0	0	0	0	0	51	0	0	0	0	0
spinach	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	34	0	0	0	17	0	0	0	0	17	0	0	0	68	17	0	0	0	0	0	0
sweetcorn	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	51	0	0	0	51	0	0	0	0	34	0	0	0	17	0	17	0	0	0	0	0
sweetpotato	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	34	0	0	0	0	34	0	0	0	34	0	34	0	0	0	0	0
tomato	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	51	0	0	0	0	34	17	0	0	17	0	34	0	0	0	0	0
turnip	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	51	0	0	0	0	34	0	0	0	85	0	0	0	0	0	0	0
watermelon	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	51	0	0	0	0	0	34	0	0	34	0	34	0	0	0	0	0

d. Performance metrics:

To assess the efficacy of the suggested framework, we use many statistical metrics such as precision, recall, accuracy, sensitivity, and F1- score. True positive (TP), true negative (TN), false positive (FP), and false negative (FN) characteristics are determined by the confusion matrix's conclusion. The word "TP" refers to a successful detection of true instants. The abbreviation "TN" denotes a result in which the proposed system correctly recognized the kind of fruit that was misclassified. The term "FP" refers to a situation in which the suggested framework incorrectly identifies a positive detection. The term "FN" refers to a case in which the suggested framework incorrectly determined the sort of negative detection.

These metrics are calculated below:

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

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

$$F1 - score = 2 \times \frac{Pr \times Re}{Pr + Re}$$

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

---

8. Predictions

Example: Corn Classification

Input Image:

Prediction Output

Predicted Class: Corn
Confidence: 86.8%

---

9. Conclusion and Future Work

This project demonstrates the successful application of **Convolutional Neural Networks (CNNs)** for classifying fruits and vegetables, achieving high accuracy across the training, validation, and test datasets. However, there are several ways to improve the model's performance in future iterations.

a. Expanding the Dataset

Expanding the dataset to include more categories and diverse conditions would improve the model's generalization. More fruit and vegetable types, along with images captured in varied lighting, angles, and backgrounds, would help the model better handle real-world scenarios. This would make the model more robust to challenges such as occlusion and poor lighting.

b. Implementing Advanced Architectures

Adopting more advanced architectures, such as **EfficientNet**, could enhance model performance. EfficientNet is known for achieving high accuracy with fewer computational resources by scaling depth, width, and resolution. Other architectures like **ResNet** and **Inception** can also be explored for better feature extraction and deeper learning capabilities, improving overall accuracy.

c. Exploring Transfer Learning

Using **transfer learning** with pre-trained models like **VGG16**, **ResNet50**, or **InceptionV3** can improve model performance. These models, pre-trained on large datasets like **ImageNet**, can be fine-tuned for the specific task, leading to faster training and better generalization. Transfer learning helps the model learn efficiently from smaller, specialized datasets while maintaining high accuracy.

References

[1] Kritik Seth, "Fruits and Vegetables Image Recognition Dataset," Kaggle 2020. Dataset Link.

[2] TensorFlow Documentation. https://www.tensorflow.org.

[3] Springer, *Advanced Food Classification Techniques Using CNNs*, 2023