Image Classification of Fruit Using Convolutional Neural Networks

Name: Nguyen Quoc Dinh
Student ID: 300550781
Course Code: COMP309
Lecturer: Dr Qi Chen
Assignment: Image Classification

I. Introduction
II. Problem Investigation
- 1. EDA - Image Distribution in Classes
- 2. EDA - Diversity of image sizes
- 3. EDA - Color Distribution across Classes
- 4. EDA - Noisy image checking
- 5. PreProcessing - Noisy Image Removal

III. Methodology
- 1. Baseline intuition through a basic model
- 2. Building a CNN model with some optimizations
- 3. Augmentation
- 4. Transfer learning

IV. Results and Discussion
- 1. MLP model
- 2. CNN model
- 3. Comparison

V. Conclusion and Future Work
- 1. Next Steps
- 2. Future Work:

VI. References

I. Introduction

In addressing the challenge of accurately classifying cherries, strawberries, and tomatoes across diverse environmental conditions, I embarked on a comprehensive Exploratory Data Analysis (EDA) to grasp the dataset's intricacies, encompassing size, image shapes, class distribution, and potential noise patterns. Following EDA, I conducted data pre-processing, specifically targeting noisy images for removal. Subsequently, I implemented a baseline Multilayer Perceptron (MLP) and a customized Convolutional Neural Network (CNN) for image classification, training them on the refined dataset. To optimize the CNN's performance, I employed techniques like transfer learning, augmentation, hyperparameter tuning, and optimization strategies. The ultimate evaluation focused on assessing the tuned CNN model's effectiveness on the designated test set, ensuring robust classification accuracy.

II. Problem Investigation

1. EDA - Image Distribution in Classes

The class distribution analysis reveals that the dataset is well-balanced, with each class (Cherry, Strawberry, Tomato) having an equal number of samples (1500 each). This balanced distribution is beneficial for training machine learning models, as it ensures that the model receives sufficient information about each class, preventing one class from dominating the learning process.

Class	Cherry	Strawberry	Tomato
Number of samples	1500	1500	1500

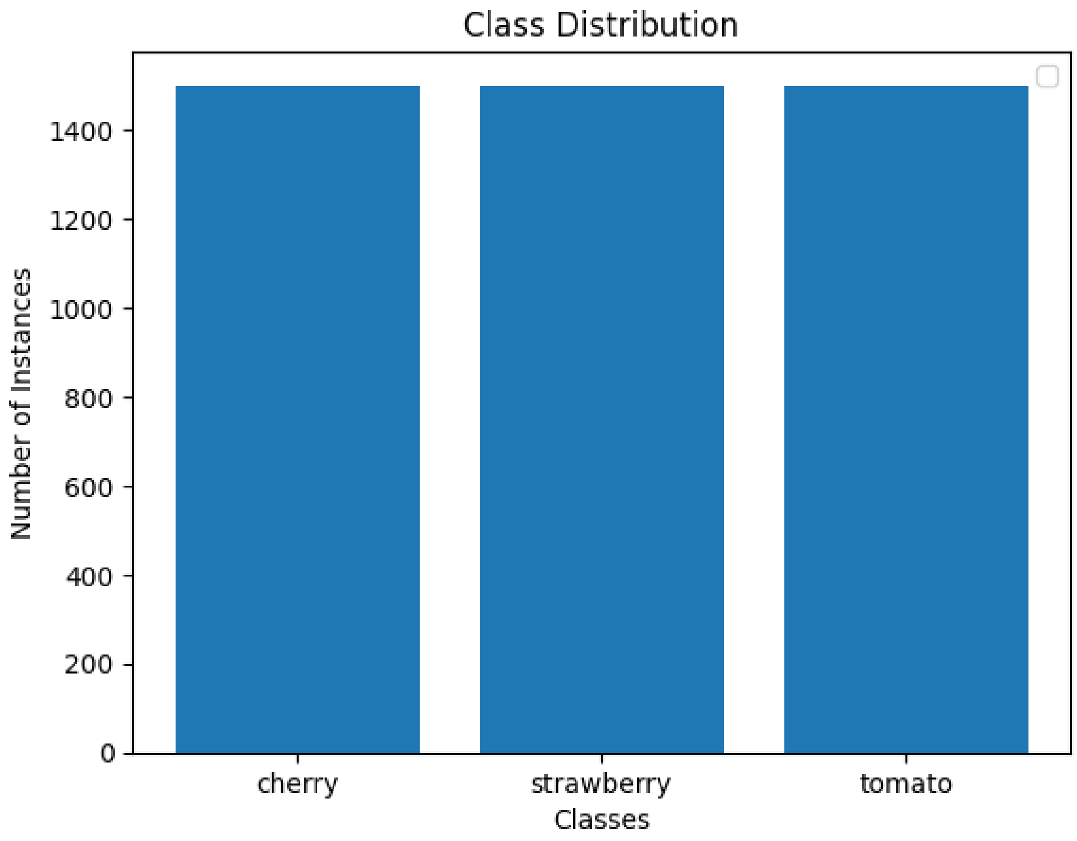

Figure 1: Image Distribution in Classes

2. EDA - Diversity of image sizes

The dataset comprises 35 unique combinations of image sizes, with varying widths and heights. The maximum image size is 900x870 pixels, and the minimum is 138x366 pixels. This diversity in image dimensions reflects real-world scenarios but necessitates preprocessing for standardization. Resizing images to a consistent size is crucial for model training, ensuring uniform input dimensions and facilitating improved convergence and computational efficiency.

	Width	**Height**
Max Size Image	900	870
Min Size Image	138	366

Example	**1**	**2**	**3**	**4**	**5**
Width	193	236	228	229	279
Height	261	214	221	220	181

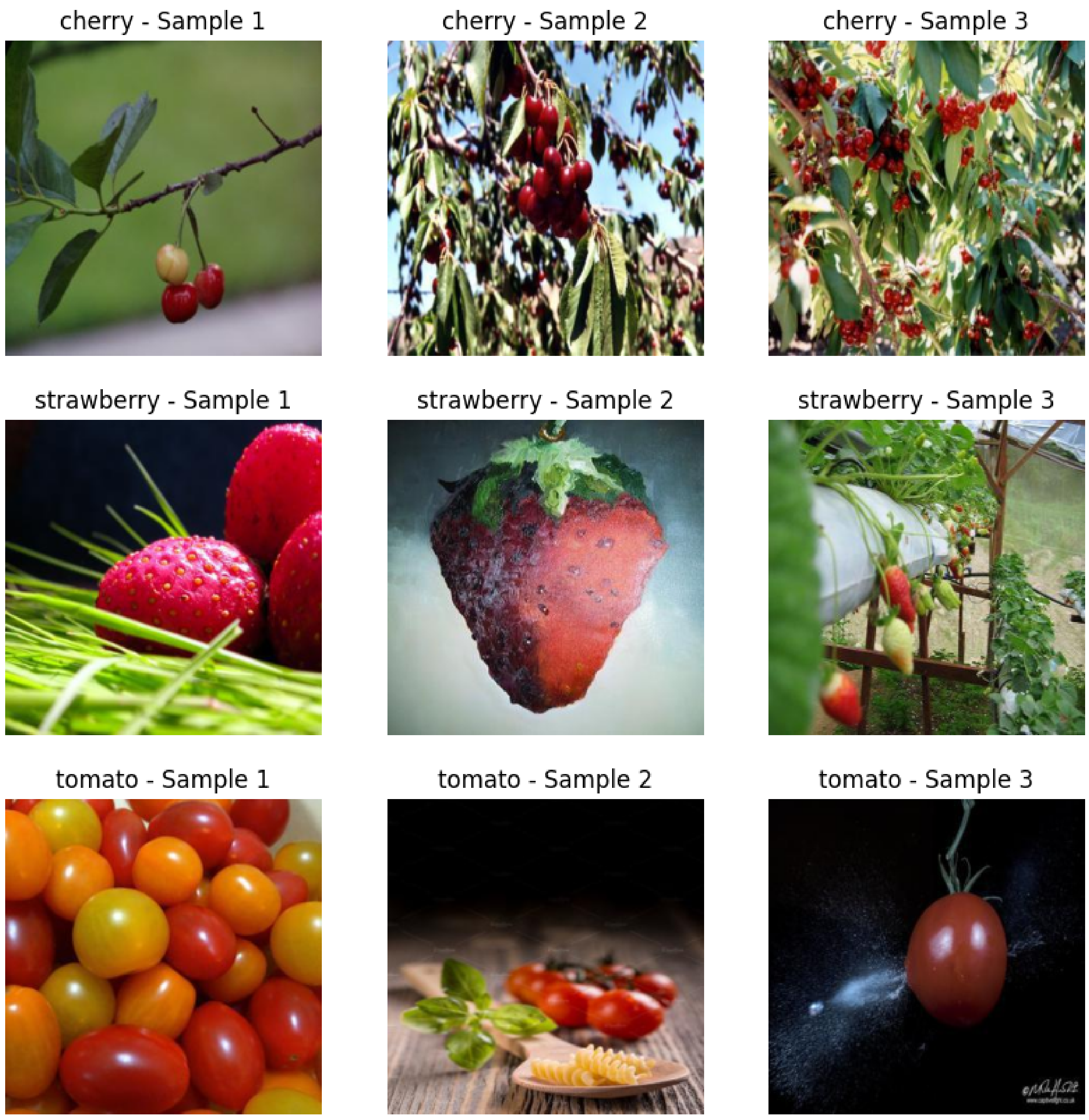

Figure 2: Different sample sizes of 3 classes

3. EDA - Color Distribution across Classes

To understand if specific color patterns characterize each class, the color distribution of three samples from each class (Cherry, Strawberry, Tomato) was analyzed. Figures 3, 4, and 5 depict the color distribution in the RGB channels. A common observation across most samples is a right-skewed distribution in all three channels (red, green, and blue). This skewness suggests the prevalence of certain color tones within the dataset. Further preprocessing may involve color normalization or augmentation to enhance model robustness to variations in color patterns.

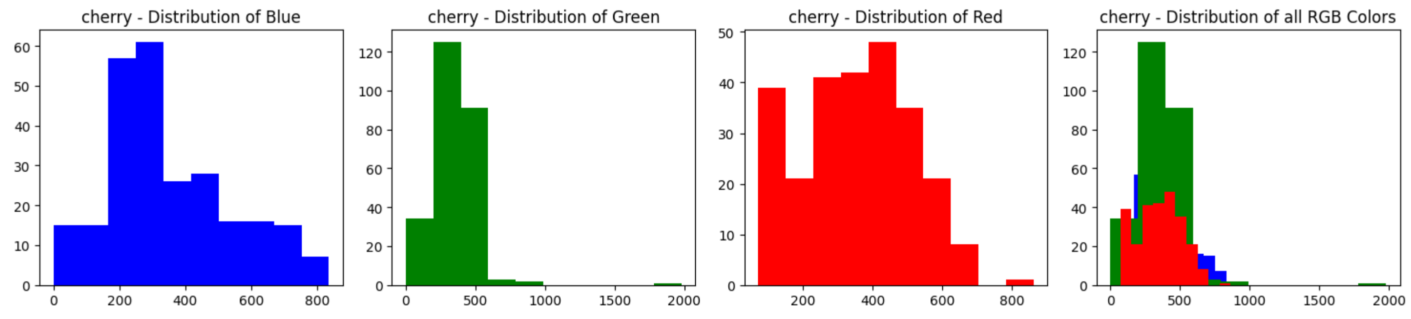

Figure 3: Color Distribution of a Cherry Sample

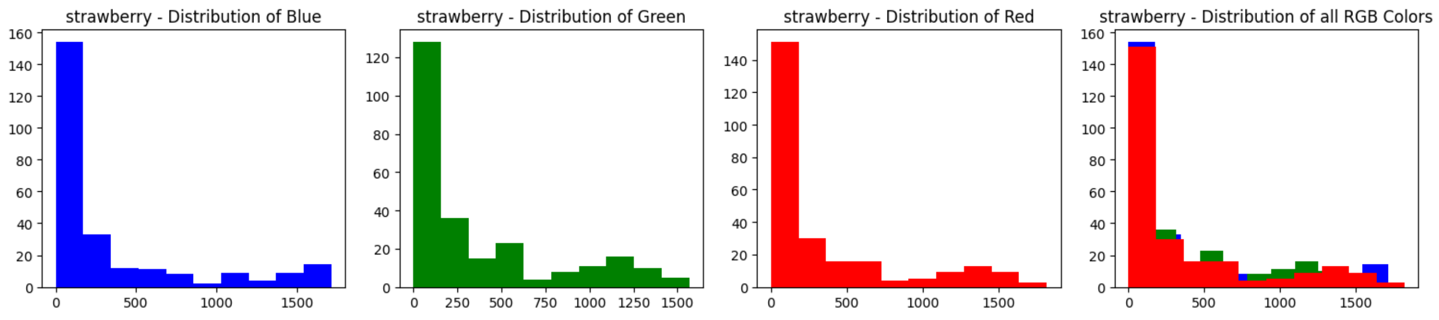

Figure 4: Color Distribution of a Strawberry Sample

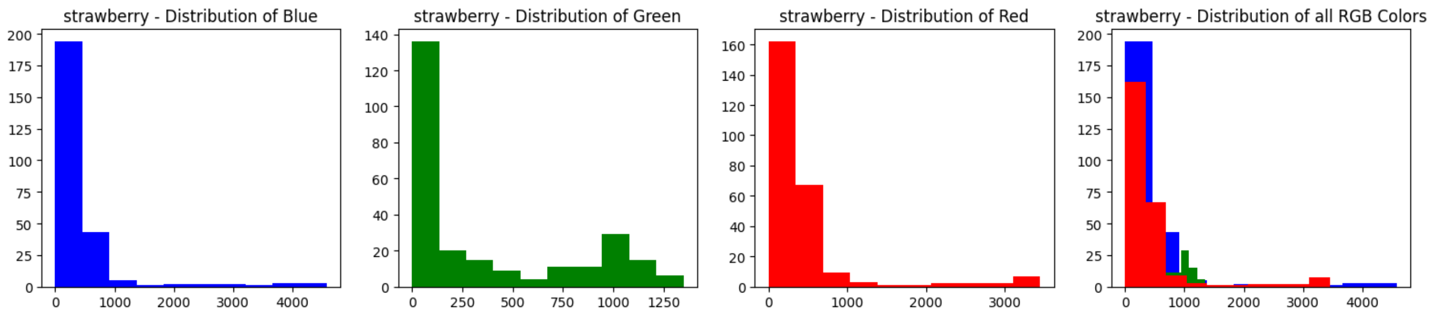

Figure 5: Color Distribution of a Tomato Sample

4. EDA - Noisy image checking

Identifying 22 noisy images with right-skewed color distributions among the dataset highlights a potential source of interference in model training. The visualization of these noisy samples informs strategies to address and mitigate their impact.

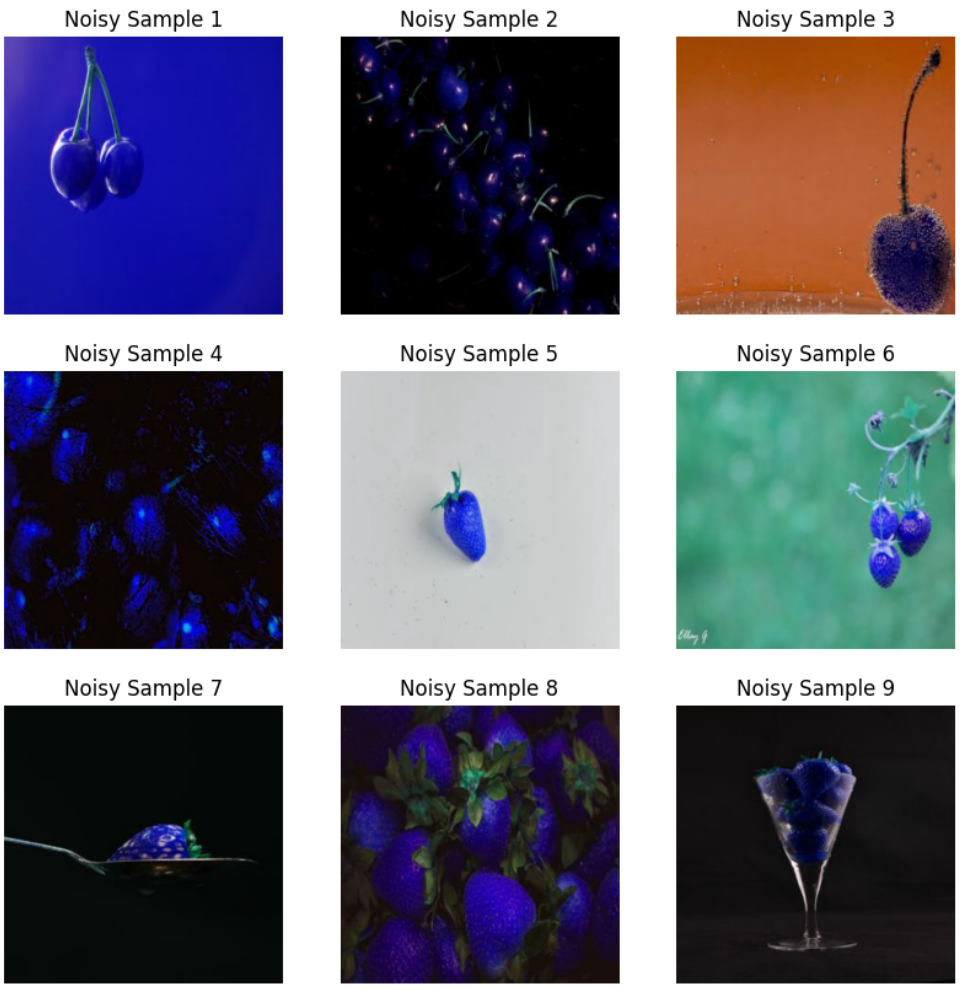

Figure 6: Noisy Samples for each Class

5. PreProcessing - Noisy Image Removal

Preprocessing involved removing 22 noisy images, improving dataset quality by reducing potential disturbances, and promoting better model generalization.

III. Methodology

1. Baseline intuition through a basic model

To have a baseline intuition on the work of image classification and the dataset, the MLP model has been built with 4 linear layers and trained the model on a small amount of dataset. The dataset involved 720 images on the training set (240 images on each class) and 180 images on the test set (60 images on each class).

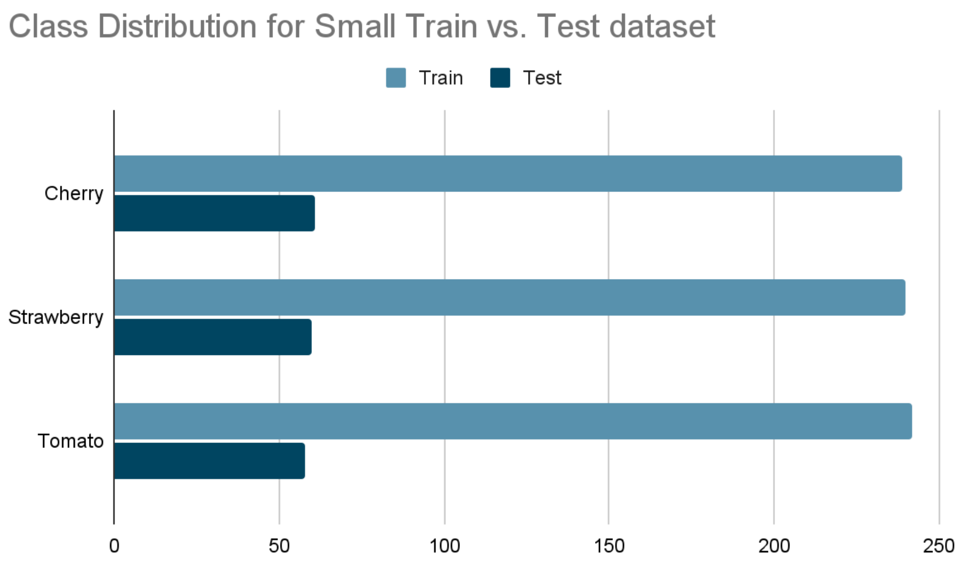

Figure 7: Train and Test Data Distribution on Classes for the MLP model

2. Building a CNN model with some optimizations

For training the SimpleCNN model, it was split at the rate of 80/20. It was implemented with a ReLU activation function, utilizing the CrossEntropyLoss as the chosen loss function for its effectiveness in multi-class classification tasks. Stochastic Gradient Descent (SGD) was employed as the optimization method with a learning rate of 0.001 and a momentum term of 0.9 to balance convergence speed and stability. L2 regularization, or weight decay, with a coefficient of 0.001 was introduced to prevent overfitting by penalizing large weights. This regularization strategy adds stability to the training process.

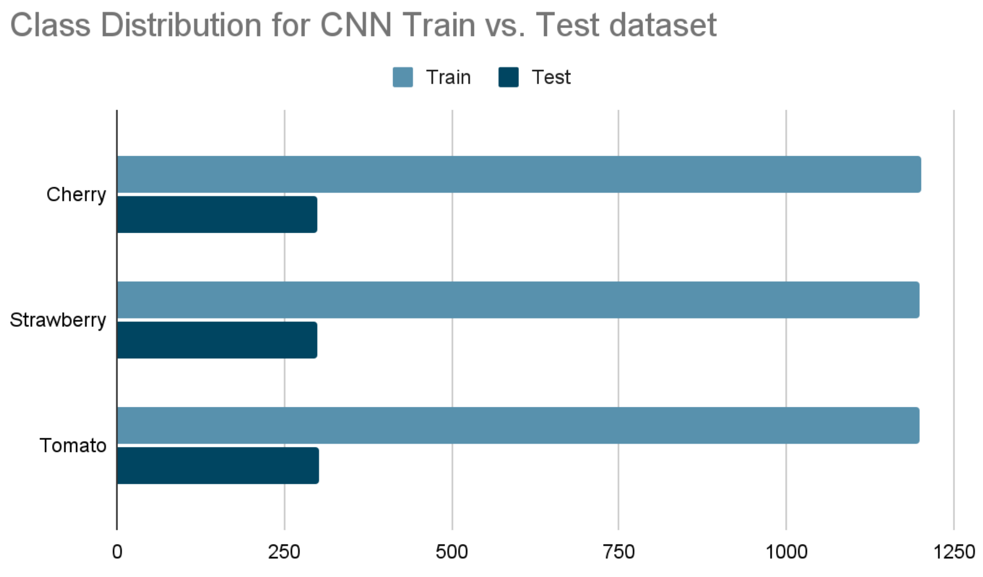

Figure 8: Train and Test Data Distribution on Classes Before Augmentation

3. Augmentation

To augment the training set, I utilized the Augmentor library to generate 300 additional images for each class—cherry, strawberry, and tomato. Various augmentation techniques, including flipping, black-and-white conversion, rotation, skewing, and zooming, were applied to introduce diverse variations. This process, resulting in 5400 images, enhances the dataset's richness and diversity. Augmentation is a widely adopted practice to prevent overfitting and improve a model's generalization by exposing it to a broader range of visual patterns and conditions, ultimately boosting its performance on unseen data.

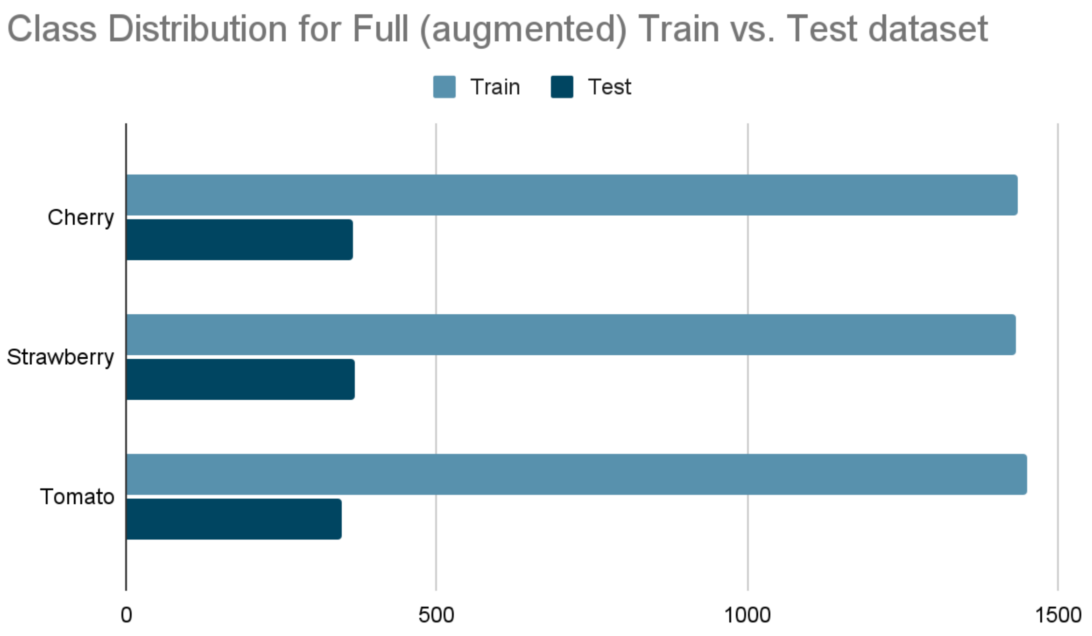

Figure 9: Train and Test Data Distribution on Classes After Augmentation

4. Transfer learning

Transfer learning employs ResNet18 pre-trained on IMAGENET1K for feature extraction. The CrossEntropyLoss handles multi-class classification, and Adam optimization with a learning rate of 0.008 is chosen. L2 regularization, with a weight decay of 0.001, prevents overfitting. Despite these optimizations, including a 25-epoch training cycle, doesn't significantly improve accuracy. Considering the trade-off between accuracy, computational cost, and complexity, transfer learning may not be worthwhile due to the extended processing time with limited accuracy gains.

IV. Results and Discussion

1. MLP model

The chosen baseline MLP consists of four layers: the input layer, two hidden layers with 900 and 120 neurons, respectively, and the output layer with three neurons corresponding to the three classes. The input layer expects flattened input data of size 3*300*300. Rectified Linear Unit (ReLU) activation functions are applied after each linear layer to introduce non-linearity. The model employs the CrossEntropyLoss as the loss function, suitable for multi-class classification tasks. Stochastic Gradient Descent (SGD) with a learning rate of 0.01 and momentum of 0.9 is used for optimization. The training is conducted over 10 epochs, aiming to strike a balance between model convergence and computational efficiency.

The training loss and accuracy curves suggest that the model has learned stably and gradually from the training data. However, the results from the test data seem to be unstable with up and down periods.

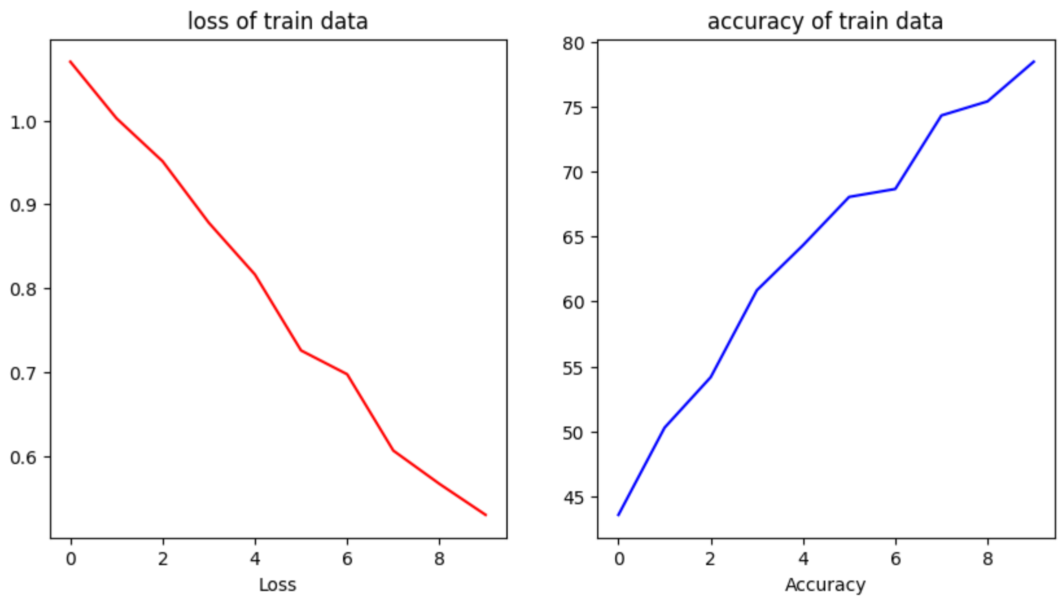

Figure 10: Loss and Accuracy of Train Data from the MLP model

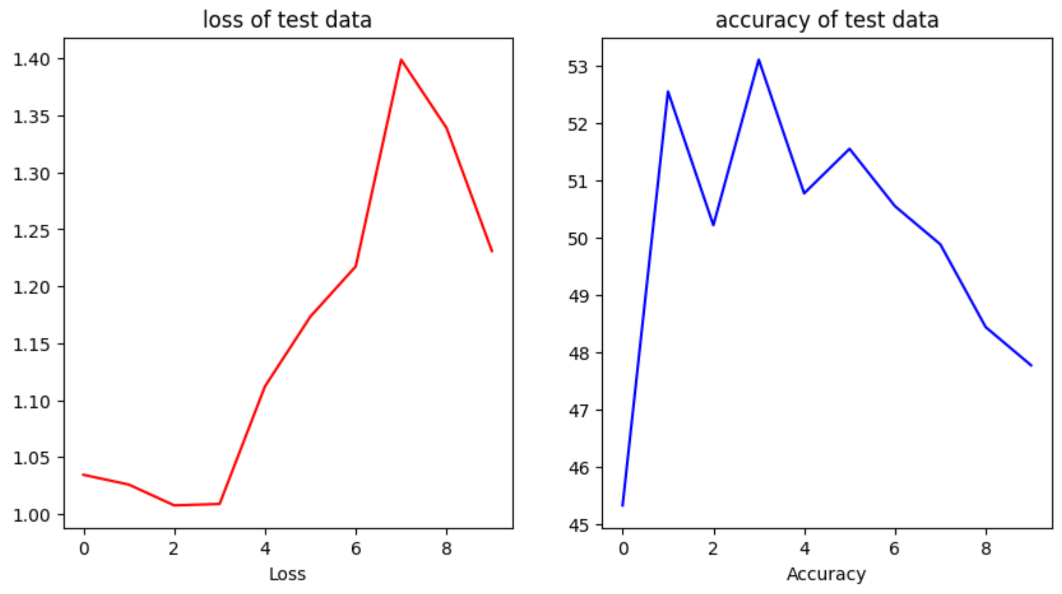

Figure 11: Loss and Accuracy of Test Data from the MLP model

2. CNN model

The best CNN model, named SimpleCNN, is designed with two convolutional layers followed by max-pooling operations to extract hierarchical features. The convolutional layers have 3 input channels, and the first layer outputs 6 channels, while the second layer outputs 16 channels. Rectified Linear Unit (ReLU) activation functions are applied after each convolutional layer to introduce non-linearity. Max-pooling layers reduce the spatial dimensions. The fully connected layers consist of three linear layers, gradually decreasing the number of neurons from 9*96*96 to 3. The ReLU activation function is again applied between these layers. The output layer, with three neurons, corresponds to the three classes of the classification task. For training, the CrossEntropyLoss function is used as the loss criterion, and the Adam optimizer with a learning rate of 0.008 and L2 regularization (weight decay) of 0.001 is employed. The training is conducted over 25 epochs to ensure sufficient convergence.

The training loss and accuracy curves suggest that the model has learned stably and gradually from the training data. This happens the same as it does in the MLP model. However, the results from the test data seem to be unstable with up and down periods. This is indeed more severe as compared with the relevant results on the MLP model. The performance goes well in the first 6 epochs before shocking waves appear through the following epochs.

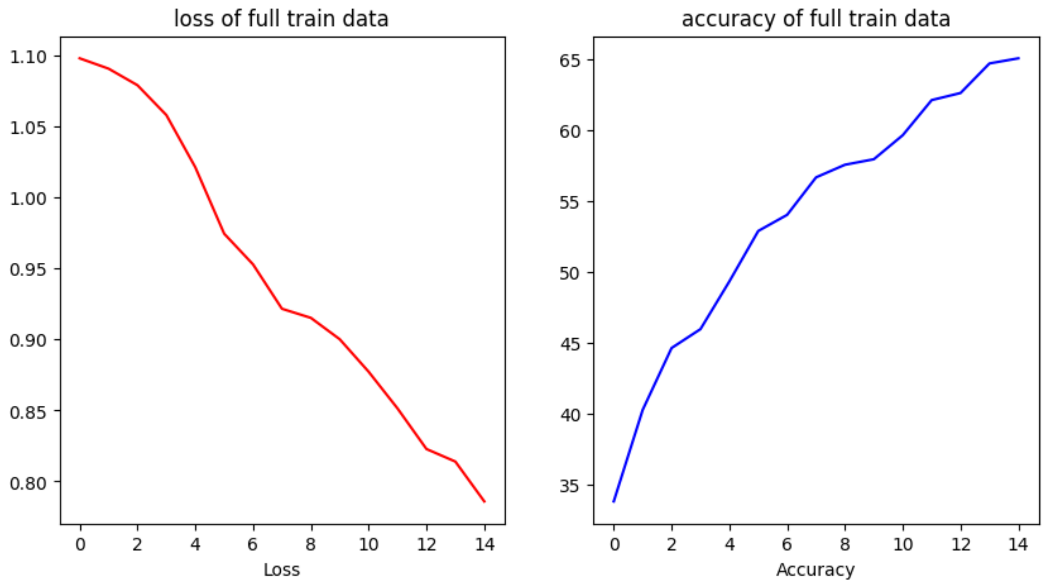

Figure 12: Loss and Accuracy of Test Data from the CNN model

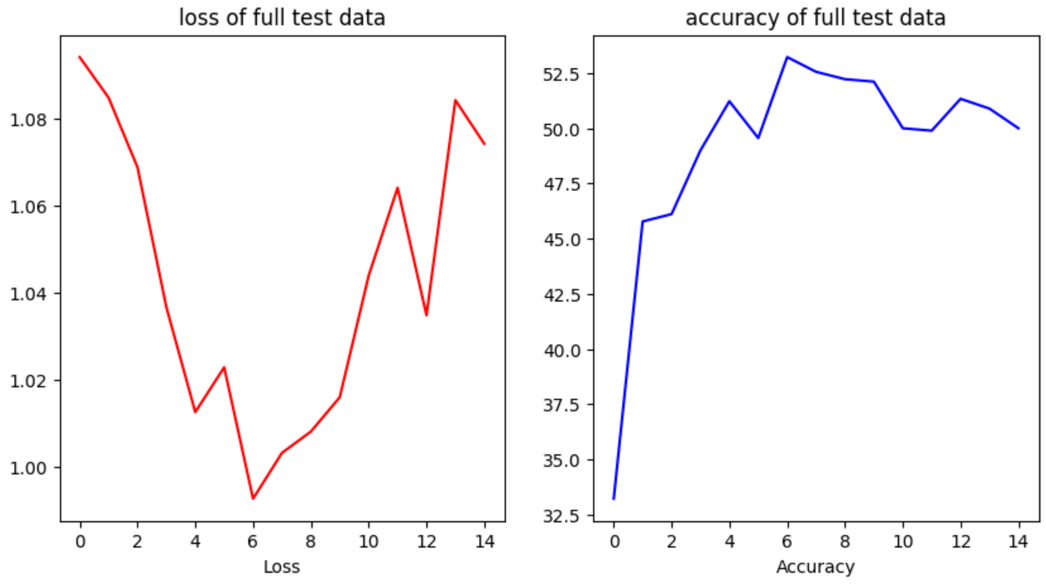

Figure 13: Loss and Accuracy of Test Data from the CNN model

3. Comparison

The MLP model shows a consistent increase in both training accuracy and test accuracy over epochs, reaching 78.47% and 47.78%, respectively. On the other hand, the CNN model achieves lower training accuracy (65.06%) but comparable test accuracy (50.00%) after 15 epochs. In terms of training time, MLP converges faster, completing 10 epochs, while the CNN model requires 15 epochs. The MLP prioritizes quicker convergence, resulting in a better classification performance on the validation/test data compared to the CNN within the specified epochs.

In comparing the MLP and CNN models, the CNN exhibited superior classification performance, with an average test accuracy ranging from 46.11% to 53.22% over 15 epochs, surpassing the MLP's range of 45.33% to 47.78% over 10 epochs. Despite a longer training time, the CNN's capacity to capture spatial dependencies in images contributed to its consistently higher accuracy. This highlights the trade-off between model complexity and training time, emphasizing the significance of selecting an architecture aligned with task requirements.

V. Conclusion and Future Work

In the comparison of three approaches, the MLP model achieved a test accuracy ranging from 45.33% to 47.78% over 10 epochs, demonstrating its effectiveness in handling tabular data. The Simple CNN model, despite a longer training time, consistently outperformed the MLP with an average test accuracy of 50.78% to 53.22% over 15 epochs, showcasing its ability to capture spatial dependencies in image data. The transfer learning approach using ResNet18 exhibited fluctuations in accuracy, reaching up to 54.63% but with increased volatility. Pros of the MLP include simplicity and efficiency for tabular data, while CNN excels in image data but requires more computational resources. Transfer learning leverages pre-trained models but does not bring out significant improvement in this case because only ResNet18 is applied instead of another advanced pre-trained model such as ResNet50.

1. Next Steps

Hyperparameter Tuning: Conduct a thorough search to optimize hyperparameters for both the MLP and CNN models, considering learning rates, batch sizes, and regularization parameters.

Data Augmentation: Implement diverse data augmentation techniques to enrich the dataset, improve model generalization, and enhance robustness.

Ensemble Learning: Explore the benefits of ensemble methods by combining predictions from multiple models, possibly including different CNN architectures or incorporating MLP in an ensemble.

Transfer Learning Refinement: Experiment with different pre-trained models, considering freezing specific layers during transfer learning for task-specific feature learning.

2. Future Work

Advanced Architectures: Investigate more sophisticated CNN architectures (e.g., ResNet, VGG) or advanced models like transformers for improved feature extraction and representation.

Examine Misclassifications: Analyze misclassified samples to understand model weaknesses, guiding targeted interventions like more data collection or specific data augmentation strategies.

Learning Rate Schedules: Implement learning rate schedules or dynamic adjustments to adapt learning rates during training, potentially accelerating convergence.

Regularization Techniques: Explore diverse regularization techniques, such as dropout or batch normalization, to mitigate overfitting and improve model generalization.

Model Interpretability: Integrate tools for model interpretability, such as SHAP values or Grad-CAM, to gain insights into feature importance and enhance trust in model predictions.

Deployment and Integration: Develop a strategy for model deployment, considering integration into real-world applications or systems, ensuring seamless interaction and efficient performance.

VI. References

Ottoni, A. L. C., Novo, M. S., & Costa, D. B. (2023). Hyperparameter tuning of convolutional neural networks for building construction image classification. The Visual Computer, 39(3), 847-861.

Krishna, S. T., & Kalluri, H. K. (2019). Deep learning and transfer learning approaches for image classification. International Journal of Recent Technology and Engineering (IJRTE), 7(5S4), 427-432.

Zheng, Q., Yang, M., Tian, X., Jiang, N., & Wang, D. (2020). A full-stage data augmentation method in deep convolutional neural network for natural image classification. Discrete Dynamics in Nature and Society, 2020.