CS464 - Introduction to Machine Learning Final Report

Fruit Classification Using Custom and Pre-Trained CNN Models on the Fruits-360 Dataset

Batıkan Aybey Ulu - 22103879 Mehmet Oral - 22102156 Almina Engür - 22101939
Güzinsu Yaylacı - 22102007 Ahmet Arda Keskin - 22102206

1. Introduction

The classification of fruits using image-based techniques is a relevant and practical problem in the field of computer vision and machine learning. Accurate fruit classification has numerous real-world uses, such as automated grocery store checkout systems and diet monitoring apps.

In this project, we studied the capabilities and limitations of deep learning in addressing multi-class image classification challenges. For this purpose, we both developed a Convolutional Neural Network (CNN) from scratch and leveraged pre-trained architectures, specifically ResNet and DenseNet. The objective was to construct and evaluate different CNN architectures that can accurately recognize fruit types based on their appearance in images, and compare the performance differences between the custom-designed model and state-of-the-art pre-trained models.

This final report outlines the methodologies applied, results obtained, and a comparative analysis of the models. Additionally, it highlights the background information, challenges encountered, the decisions made in the development process, and potential future directions for improving classification performance.

The methodologies applied, experimental results obtained, and a detailed comparative analysis of the models are presented in this report. The study shows that while the custom CNN model achieved an accuracy of 88.49%, significantly higher performance was obtained using pre-trained models, with DenseNet reaching 99.7% accuracy and ResNet achieving 98.38%. These results demonstrate the effectiveness of transfer learning and advanced CNN architectures in achieving high accuracy for complex image classification tasks.

2. Problem Description

Image classification is one of the core problems of computer vision, its applications vary from healthcare to agriculture. In the project, the classification of the various fruit and vegetable images using the custom and pre-trained machine learning models was the focus. Accurate fruit classification has numerous real-world uses, such as automated grocery store checkout systems and diet monitoring apps. The main question addressed throughout the project was “How effectively can custom-built and pre-trained Convolutional Neural Network (CNN) models classify fruits and vegetables from the Fruits-360 dataset?”

The Fruits-360 [1] dataset originally contains 141 classes of fruits and vegetables; however, to investigate the problem in a more time-efficient manner, the number of classes were reduced to 24 classes. The aim was to create a custom-designed Convolutional Neural Network (CNN) and to compare the results of the custom model with other widely used pre-trained machine learning models. The other pre-trained models selected were ResNet and DenseNet, since these models are popularly used in machine learning applications that are handy for image processing. The comparison was done in terms of metrics such as accuracy, precision, recall, F1-score, and confusion matrices. This analysis aims to decide whether the pre-trained models improve the performance compared to the custom-designed models.

3. Methods

The Fruits-360 dataset was used for the project. The Fruits-360 dataset is an image classification dataset containing 94110 images of 141 different fruit and vegetable classes. The image sizes are 100 x100. The dataset consists of training (70491 images) and test (23619) sets in separate folders. It contains high-resolution RGB images, each having a white background. The images are captured while the fruit rotates 360 degrees, allowing multiple viewpoints per class. Although the dataset originally contains 141 classes of fruits and vegetables, to investigate the problem in a more time-efficient manner, the number of classes was reduced to 24 classes. Total of 3640 tests and 10876 training images were used. In each class there are nearly the same number of images used for class balance (for each class approximately, 150 test and 475 training images used).

3.1. Preprocessing for Custom CNN

To prepare the data, a data loader was written to process images from the Fruits-360 by:

- converting them to grayscale,
- performing padding with a 3x3 kernel,
- and normalizing pixel values to the [0,1] range (by dividing by 255) was built.

Additionally, class labels were assigned based on folder structure, and they were initially stored as integer-encoded values. Then during the training and evaluation, one-hot encoding was performed on the class labels.

3.2. Custom CNN Architecture & Implementation

A Convolutional Neural Network (CNN) was implemented from scratch using NumPy, without using deep learning libraries such as TensorFlow or PyTorch.

CNNs utilize convolutional layers to detect local patterns such as edges, textures, and shapes; pooling layers to reduce spatial dimensions and computation; and fully connected layers to perform the final classification [2].

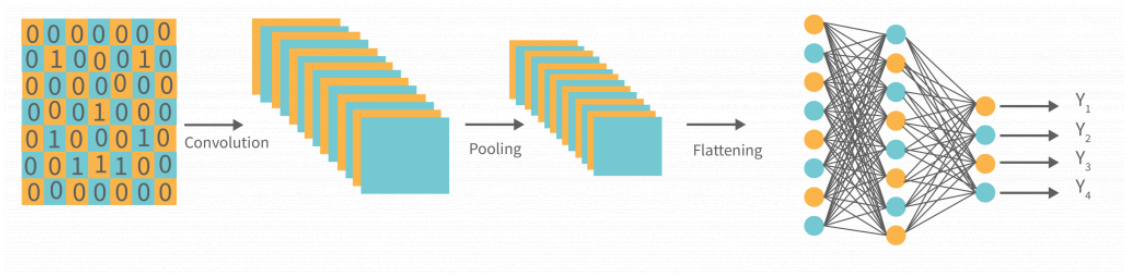

Figure 1: The diagram of the CNN architecture [2]

As can be seen in the figure above, a typical Convolutional Neural Network (CNN) processes an image through a series of layers.

The designed architecture includes:

- Two convolutional layers with 3x3 kernels, each followed by ReLu activation and max pooling
- He initialization - which ensures the signal does not vanish or explode as it passes through the layers during training, especially when ReLu is used as the activation function, was used for all weights to improve training convergence
- Finally, the extracted feature maps are flattened and passed through one or more fully connected layers (number of fully connected layers is a design parameter which is to be fine-tuned), followed by a final softmax activation to produce class probabilities at the final output layer.

The code is structured in a way that allows configurable design parameters such as learning rate, optimization type (SGD or GD), padding strategy (valid or same), number of hidden layers, regularization type (L1 or L2).

The implementation includes a complete forward pass, backward pass with gradient calculation and weight updates by using mini-batch gradient descent or stochastic gradient descent with momentum.

3.3. Preprocessing for CNN Models (ResNet and DenseNet)

Before training the ResNet and DenseNet models, the image data underwent several preprocessing steps to ensure compatibility with the pre-trained architectures and improve model performance. The images were first resized to 224×224 pixels, which is the standard input size required by both ResNet and DenseNet models trained on the ImageNet dataset.

Following resizing, the images were converted into tensors and normalized using the mean and standard deviation values of the ImageNet dataset [3]:

- Mean: [0.485, 0.456, 0.406]
- Standard Deviation: [0.229, 0.224, 0.225]

This normalization ensures that the input data distribution matches what the pre-trained models expect, allowing for more effective fine-tuning. The preprocessing pipeline was implemented using the torchvision.transforms module.

3.4. Pre-trained CNN Models (ResNet18 and DenseNet-121)

In addition to the custom-built CNN, we utilized two pre-trained deep learning models, ResNet and DenseNet, known for their superior performance in image classification tasks. The custom CNN is a basic model built from scratch with standard convolutional and pooling layers, requiring longer training and achieving lower accuracy. ResNet18 introduces residual (skip) connections, allowing deeper networks to train effectively by addressing the vanishing gradient problem. DenseNet121 further improves feature learning by using dense connections, where each layer receives inputs from all previous layers, resulting in better gradient flow, efficient feature reuse, and the highest classification performance among the models [4] [5].

These models were fine-tuned on our selected subset of the Fruits-360 dataset. We used the torchvision library's implementations of ResNet18 and DenseNet-121, modifying the final fully connected layers to match the 24 target classes. The models were trained for 10 epoch with a batch size of 32, using Adam optimizer and cross-entropy loss. Transfer learning allowed us to achieve higher accuracy with fewer epochs compared to training the custom model from scratch.

4. Results

Besides the main question mentioned in problem description, this project aims to answer the following key research questions regarding fruit classification using both custom-designed and pre-trained deep learning models:

- How effective is a custom-built CNN in classifying fruit images compared to pre-trained models?
- Can transfer learning with ResNet18 and DenseNet121 improve classification accuracy while reducing training time?
- What are the trade-offs between model complexity, training time, and classification performance?
- How does freezing pre-trained convolutional layers impact training efficiency and final accuracy?
- Which model demonstrates better generalization on unseen data based on evaluation metrics such as precision, recall, F1-score, and confusion matrices?

To address these questions, a custom CNN was implemented from scratch, and its performance was evaluated. Additionally, pre-trained ResNet18 and DenseNet121 models were fine-tuned using transfer learning. The final fully connected layers of the pre-trained models were modified for 24-class classification, and the models were trained and evaluated using standard performance metrics.

4.1. Custom CNN

The Convolutional Neural Network implemented in the progress report was improved by adding the functionality of updating convolutional layers. Previously, the weights of the convolutional layers were initialized randomly, and only the feedforward network was updated. Now, it is a fully functional CNN. The training takes 21 hours for 5 epochs. The convolutional filters can be seen in the figures below.

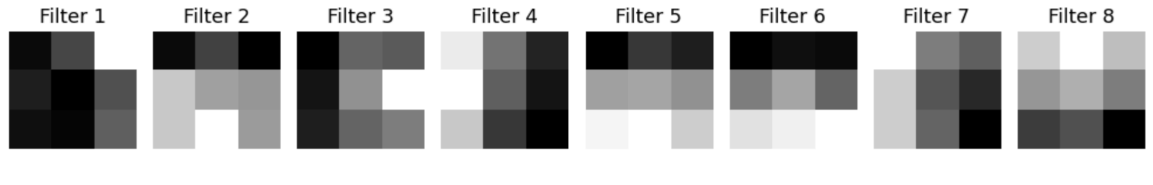

Fig. 2: First Convolutional Layer Filters

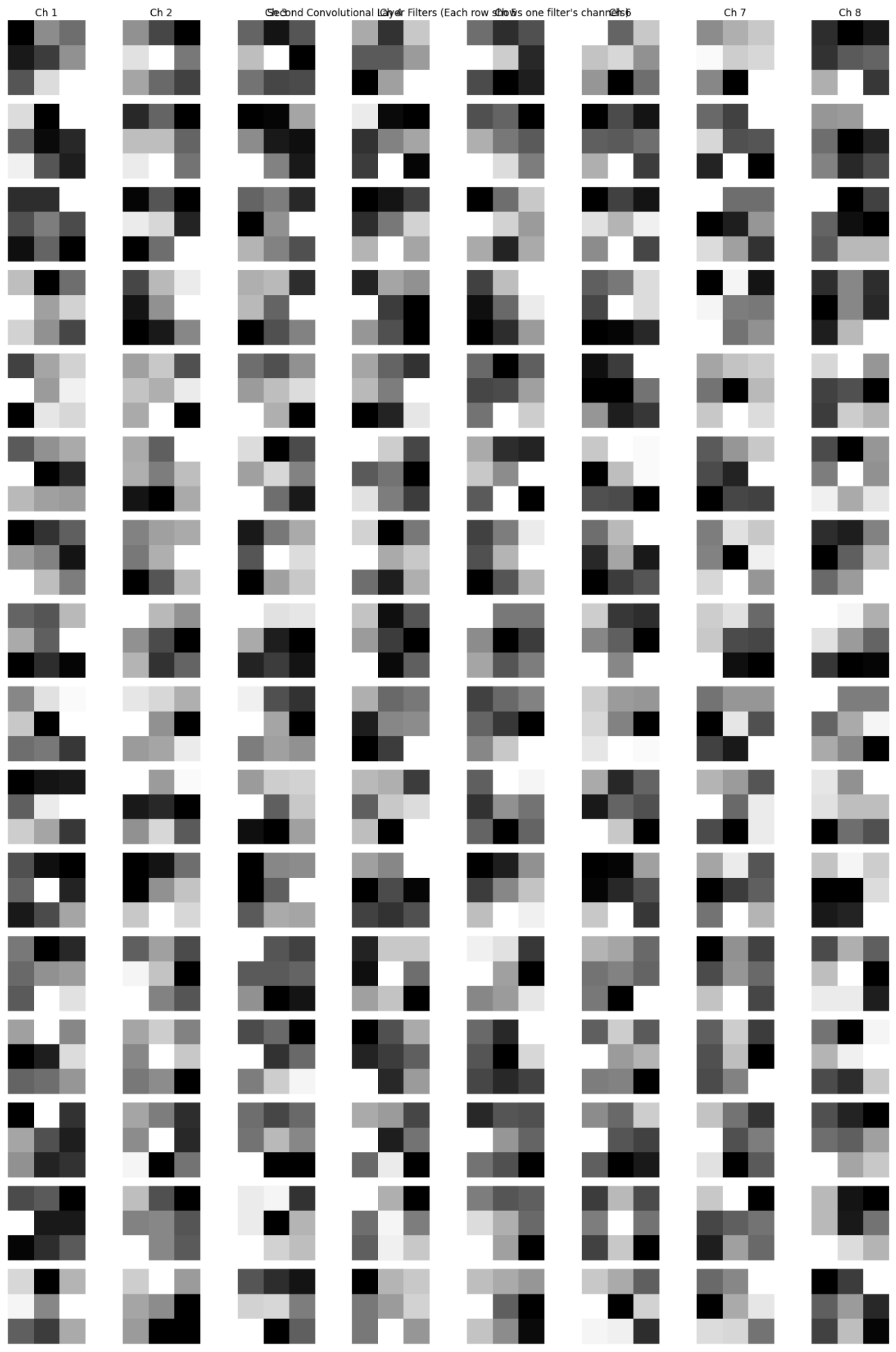
Second Convolutional Layer Filters (Each row shows one filter's channels)
Ch 1
Ch 2
Ch 3
Ch 4
Ch 5
Ch 6
Ch 7
Ch 8

Fig. 3: Second Convolutional Layer Filters

The confusion matrix can be seen below. From the figure, it can be seen that the model confuses similar classes consistently, and the diagonal values (which show the true predictions) are too dominant compared to the other predictions.

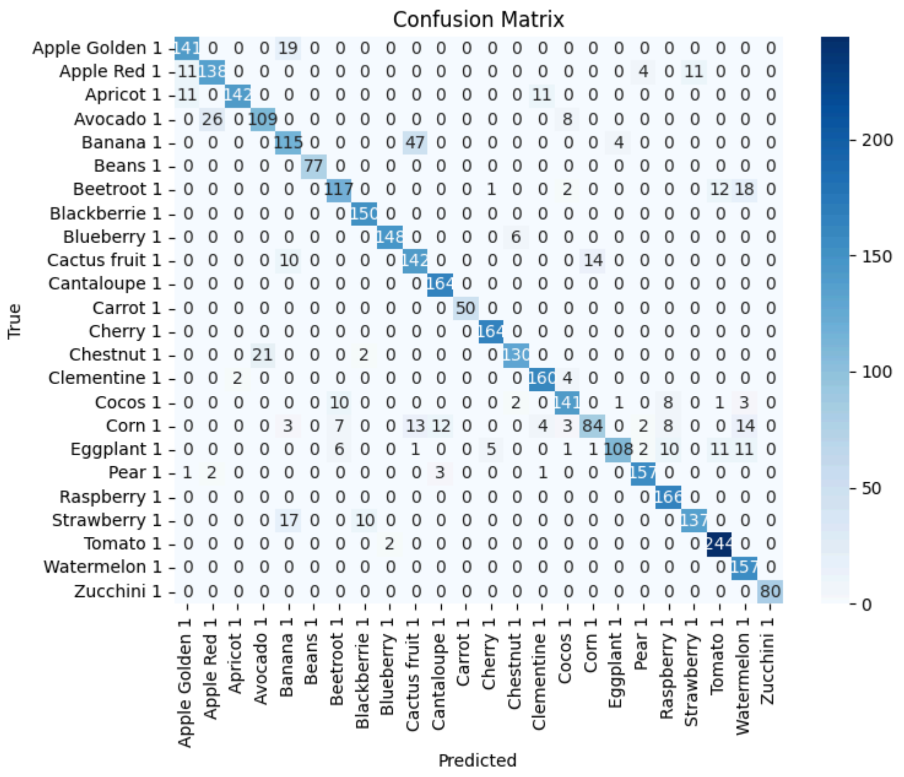

Fig. 4: Confusion Matrix of the CNN

Accuracy and test plots on the test dataset are given below. The accuracy plot starts with an accuracy of 82% at the end of the first epoch. With a number of 24 classes in the dataset, the model learned very quickly. At the end of the 3rd epoch, accuracy was decreased even if the loss was decreased. That is because the model found a local minimum between epochs 1 and 3, then left that minimum because the model found a better minimum.

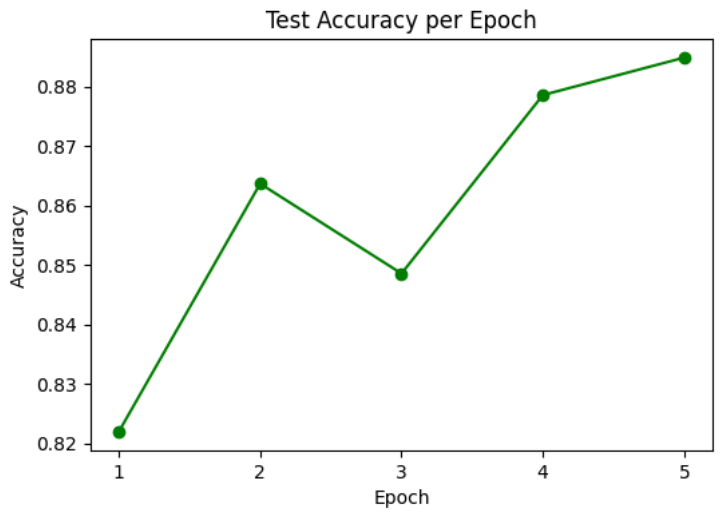

Fig. 5: Accuracy Per Epoch on the Test Dataset for Custom CNN

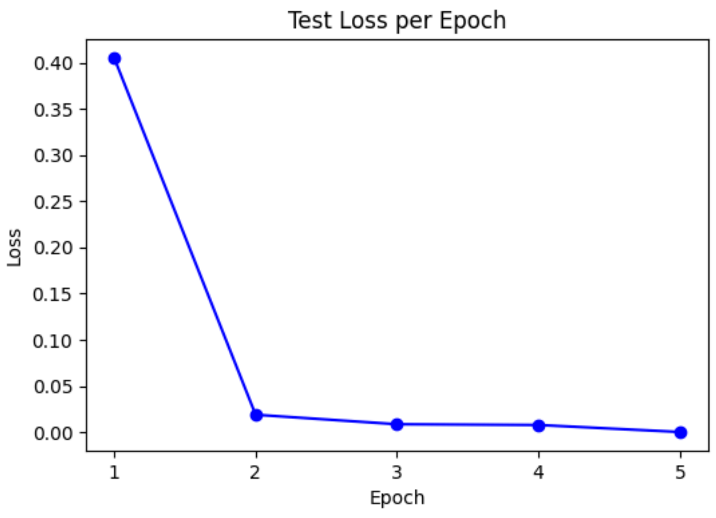

Fig. 6: Loss Per Epoch on the Test Dataset for Custom CNN

4.2. ResNet-18

To compare with the custom CNN, we implemented the ResNet18 architecture using transfer learning. ResNet18 is a widely used convolutional neural network known for its use of residual (skip) connections, which help mitigate the vanishing gradient problem and enable the training of deeper networks. In this model, the earlier layers extract general features, while the final fully connected layer was modified to classify the 24 fruit categories in our dataset. The model was trained using the Adam optimizer and cross-entropy loss. Below, the confusion matrix and training loss curve for the ResNet18 model illustrate its classification performance and training progression over epochs.

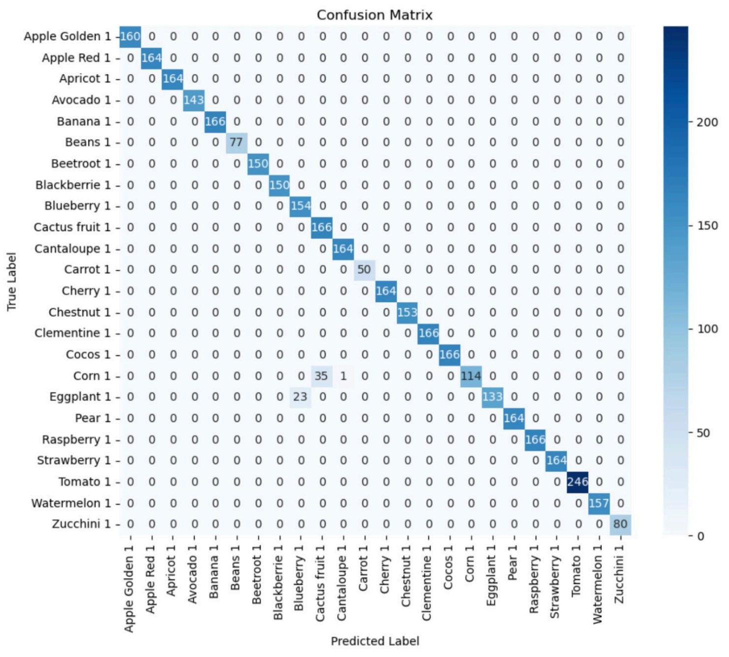

Fig. 7: Confusion Matrix of the ResNet

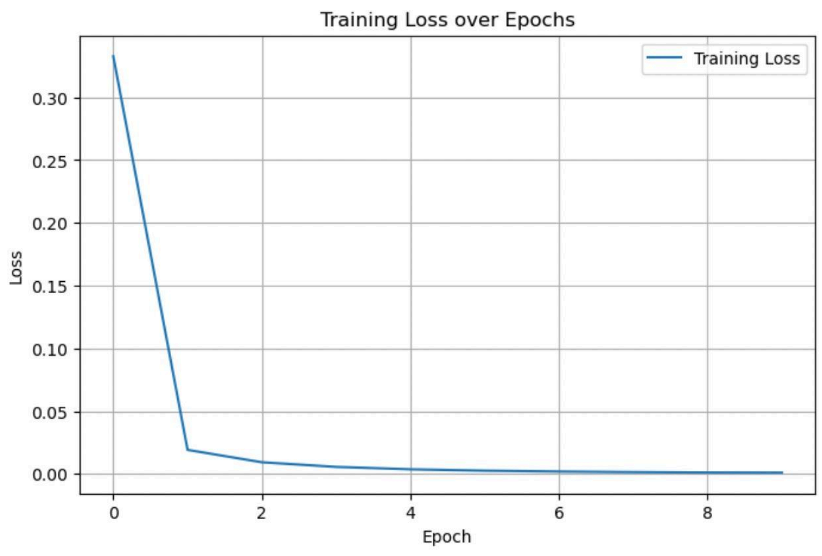

Fig. 8: Loss Per Epoch on the Test Dataset for ResNet

Within 1 hour, 10 epochs were trained. The convolutional layers were frozen (use the pretrained version), and only the final fully connected layer was updated for feature extraction. This way, the training took only 1 hour (custom CNN takes 21 hour for 5 epoch because of the convolutional layer backpropagation). The final test set accuracy was 98.38%. By comparing Figure 4 and Figure 7, it can be seen that ResNet performs slightly better but with a higher epoch size and using pretrained convolutional layers. The training loss curve for the ResNet18 model shows a rapid decrease in the early epochs, with the loss approaching near-zero values by the end of training. This indicates that the model quickly learned to fit the training data, achieving high confidence in its predictions.

4.3. DenseNet-121

The pre-trained DenseNet-121 architecture was used for the DenseNet implementation in order to take advantage of its effective feature extraction capabilities, which developed after extensive training on the ImageNet dataset. In order to adapt the model to our classification objective, a new fully connected layer set up for 24 output classes—which correspond to the chosen fruit and vegetable categories—was used in place of the final classification layer.

The final layer of the DenseNet model was optimized using our dataset, but the lower layers were kept to take advantage of their learnt hierarchical features. The Cross-Entropy loss function and Adam optimizer were used to train the model over ten epochs. To ensure adherence with the pre-trained model, batches of images were scaled to 224x224 pixels and normalized using ImageNet statistics during the training process.

After training, the test set was used to assess the model's performance, and important performance measures such as F1-score, accuracy, precision, and recall were computed. These findings, combined with the confusion matrix, provide insight on DenseNet's classification performance for the chosen classes. In the following, the confusion matrix of the model can be observed:

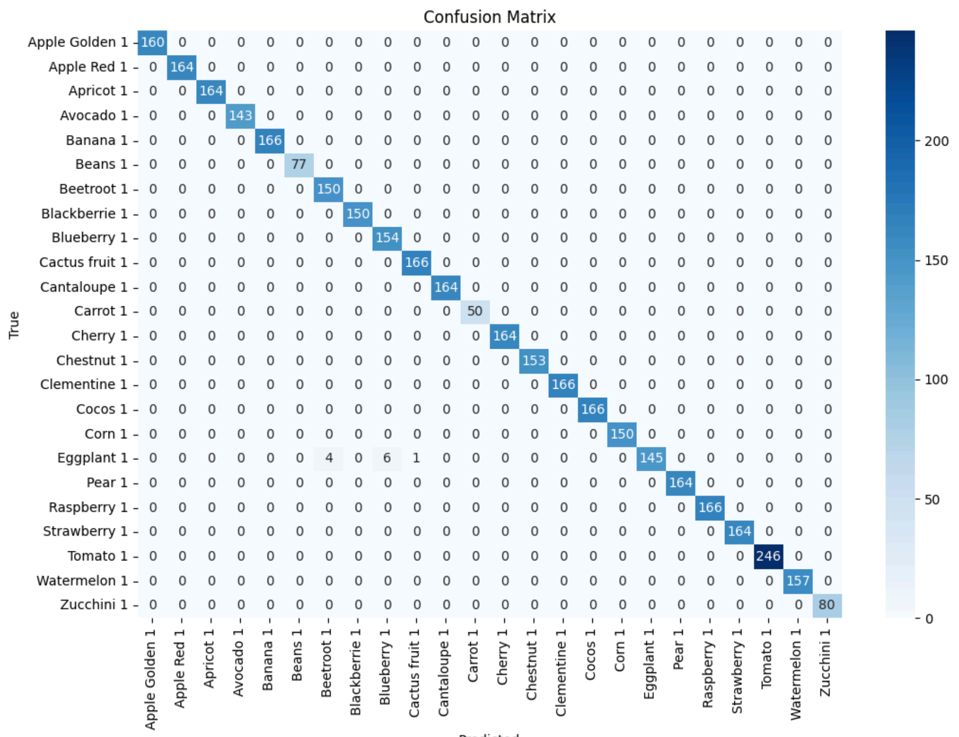

Fig. 9: Confusion Matrix for DenseNet

The confusion matrix shows that only a small portion of the classes are confused by the model. The true predictions, which lie on the diagonal axis, are higher, meaning that the true predictions are more dominant compared to the other predictions in the matrix. The test loss per epoch plot can be observed below.

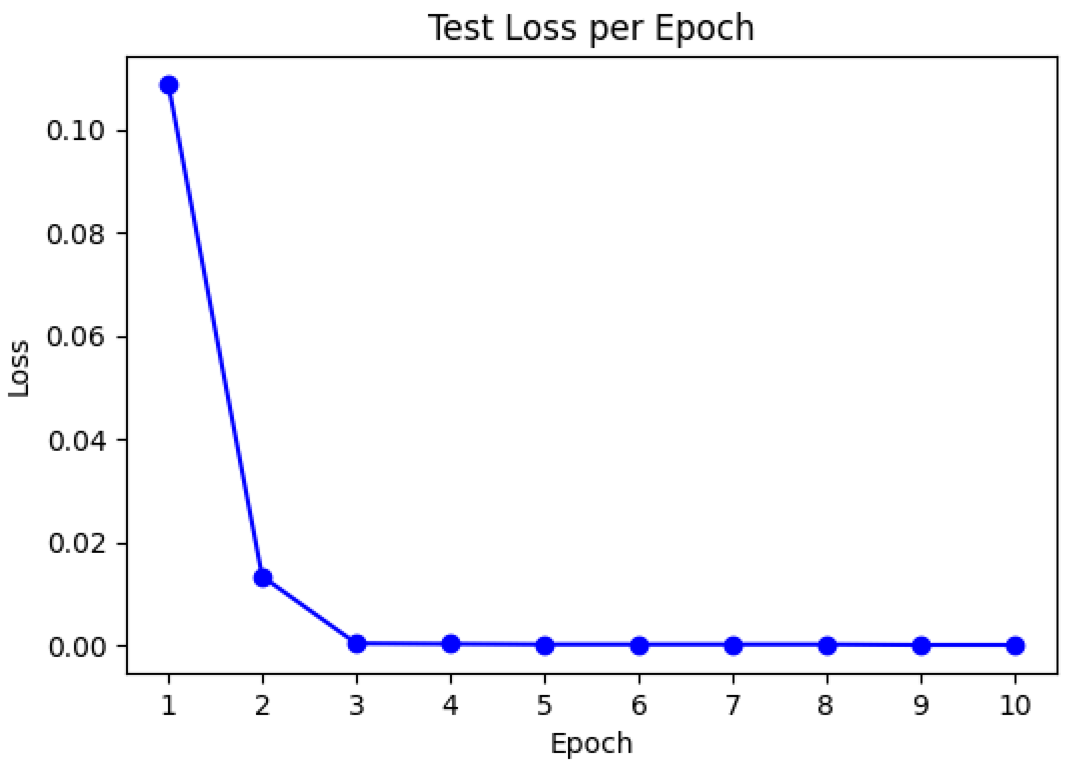

Fig. 10: Loss Per Epoch on the Test Dataset for DenseNet

The training loss curve for the DenseNet-121 model shows a rapid decrease in the first epochs and by the end of the third epoch, the loss nearly approached zero and stayed until the end of the training. This shows that the model learned to fit the training data quickly and its confidence increased in its predictions.

10 epochs were trained in 31 minutes. Similar to the ResNet model, the convolutional layers were frozen and only the fully connected layer was updated for feature extraction. The use of a pre-trained model significantly reduced the training time compared to the training time of the custom designed CNN model. The custom designed CNN model required 21 hours of training for only 5 epochs whereas the DenseNet model only required 31 minutes for 10 epochs. The final test set accuracy achieved was 99.70%. Comparing the results of the ResNet and DenseNet models, it is observed that DenseNet achieves higher accuracy and better overall performance metrics with less training time needed.

4.4. Final Results

Table 1: Performance Metrics Comparison of Custom CNN, ResNet18, and DenseNet121 Models

	Accuracy	Precision	Recall	F1-Score
Custom CNN	88.49%	89.64%	88.92%	88.88%
DenseNet	99.7%	99.71%	99.71%	99.7%
ResNet	98.38%	~99%	~98%	~98%

The performance metrics comparison table clearly highlights the superiority of the pre-trained models over the custom-designed CNN.

The custom CNN model has achieved an accuracy of 88.49%, which is a moderate result considering it was built from scratch. The precision (89.64%) and recall (88.92%) values are balanced, but the F1-score (88.88%) is relatively lower. These results indicate that the custom CNN struggles to maintain high predictive performance consistently across all classes. It struggles to generalize as well as the pre-trained ResNet and DenseNet models. This may be due to the model complexity being limited and the absence of the pre-learned feature representations. The custom CNN model has limited feature extraction capabilities and lacks the architectural optimizations found in modern CNN designs. Additionally, the training time is significantly longer (21 hours for 5 epochs), which suggests inefficiency compared to the pre-trained models.

ResNet shows a considerable performance improvement with an accuracy of 98.38%, since it benefits from transfer learning and residual connections. Its high precision (99%) and recall (98%) values suggest that it is both highly accurate and reliable in detecting all classes. While ResNet achieves very good results, it still slightly underperforms compared to the DenseNet model.

The DenseNet121 model stands out as the best performer with an accuracy of 99.7%. It also achieved perfectly balanced precision, recall and F1-score, all at 99.7%. This is because its dense connectivity structure allows better gradient flow and feature reuse. This not only improves accuracy but also significantly reduces training time (31 minutes for 10 epochs). The results show this model's very good performance with an excellent generalization capability and minimal confusion between classes.

5. Discussion

The experimental results demonstrate the comparative strengths and limitations of the three models—Custom CNN, ResNet, and DenseNet—used for fruit classification on the Fruits-360 dataset.

5.1 Custom CNN

The custom CNN, developed entirely from scratch using NumPy, provided valuable insights into the fundamental workings of convolutional networks. While the model achieved moderate accuracy (88.49%) after the first epoch, its performance was constrained by limitations in feature extraction and generalization; it struggled to match the performance of the pre-trained architectures. It gave lower precision and recall compared to the state-of-the-art models due to its limited feature extraction capabilities and lack of architectural optimizations.

Additionally, the model's training process was computationally expensive, requiring 21 hours for only 5 epochs, and it exhibited overfitting in later epochs. This highlights the challenges of designing and optimizing deep learning models from scratch, without hardware acceleration or optimized libraries, especially when compared to pre-trained architectures. The overfitting signs in the later epochs suggested poor generalization to unseen data. These observations aligned with the hypothesis that advanced architectures with pre-trained knowledge significantly improve classification performance.

5.2 ResNet18

ResNet, a pre-trained model fine-tuned for this task, demonstrated significant improvements in both training efficiency and accuracy. With a training time of just 1 hour for 10 epochs, ResNet achieved a test accuracy of 98.38%. Its lower training time was due to the use of transfer learning and frozen pre-trained convolutional layers. The model's superior performance can be attributed to its ability to leverage pre-trained convolutional layers, which effectively extracts hierarchical features. Its residual (skip) connections helped with getting rid of the vanishing gradient problem, enabling the model to extract robust hierarchical features. Additionally, the confusion matrix showed fewer misclassifications, underscoring the model's robustness.

While the ResNet model exhibited high precision (~99%) and recall (~98%), it underperformed slightly when compared to DenseNet in terms of accuracy and training time. This suggests that while ResNet is very effective compared to the custom CNN, its feature reuse and connectivity are not as comprehensive as DenseNet, which affects its final generalization ability.

5.3 DenseNet121

DenseNet outperformed both the custom CNN and ResNet, achieving the highest test accuracy of 99.70% with a training time of 31 minutes for 10 epochs. The other metrics; precision, recall and F1-score are also balanced in the DenseNet model. The model's efficient feature extraction and compact layer connections contributed to its superior performance and strengthened its ability of generalization. The loss curve for DenseNet showed the rapid decrease and the loss reached nearly zero at the end of the third epoch. It also remained stable, demonstrating the model’s learning capacity. DenseNet's ability to achieve high accuracy while maintaining efficient training makes it the most effective model in this study.

5.4 Key Insights

- **Transfer Learning Advantage:** Both ResNet and DenseNet showcased the benefits of transfer learning, enabling efficient training and high accuracy by leveraging pre-trained feature extraction layers.
- **Custom CNN Challenges:** The custom CNN, while functional, struggled with high training time and overfitting, emphasizing the challenges of building deep learning models from scratch.
- **DenseNet Superiority:** DenseNet emerged as the best-performing model, combining high accuracy, efficient training, and robust generalization.

5.5 Limitations and Future Considerations

- The custom CNN's performance was constrained by limited computational resources and the absence of advanced design optimizations such as regularization, data augmentation, hyperparameter optimization. The use of these methods could have improved the performance of the model.

- The dataset was reduced to 24 classes for the project to be more computationally feasible. The expansion of the study to include all 141 classes and all images contained by thee classes may provide additional insights into the scale of each model.

- This comparative analysis underscores the effectiveness of pre-trained models like ResNet and DenseNet for image classification tasks. DenseNet's superior performance establishes it as the most suitable model for fruit classification in this study.

6. Conclusion

The experiments showed that the custom-built CNN is capable of classifying fruit images but it is less effective than pre-trained models. Transfer learning with ResNet18 and DenseNet12 increases classification accuracy and greatly reduces training time. It shows that pretrained models are both efficient and effective for image classification.

ResNet18 is a good balance between model complexity, training speed, and performance. Beside that, DenseNet121 achieved the highest accuracy and evaluation performance due to its high connectivity and efficient feature handling. Freezing the convolutional layers of the pretrained models allowed faster training without sacrificing accuracy. It shows the practicality of fine-tuning only the final layers.

From these methods, we learned that pre-trained models, especially DenseNet121, offer significant advantages in both accuracy and computational efficiency compared to training models from scratch. Transfer learning allows that models to converge faster and generalize better.

For future work, experimenting with more advanced architectures like EfficientNet or Vision Transformers (ViT) could further improve classification performance. Applying advanced data augmentation techniques and hyperparameter tuning could enhance the performance of the custom CNN, potentially narrowing the gap between it and the pre-trained models. Expanding the study to larger and more diverse datasets could also help assess the scalability and robustness of these models.

References

[1] M. M. Oltean and R. A. Rusu, *"Fruits-360: A dataset of images for classification"* [Online]. Available: https://www.kaggle.com/datasets/moltean/fruits

[2] "CNN Architecture - Detailed Explanation," InterviewBit, https://www.interviewbit.com/blog/cnn-architecture/ [Accessed: May. 5, 2025].

[3] J. Deng, W. Dong, R. Socher, L. Li, K. Li and L. Fei-Fei, *"ImageNet: A large-scale hierarchical image database,"* 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[4] K. He, X. Zhang, S. Ren, and J. Sun, *"Deep Residual Learning for Image Recognition,"* in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[5] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, *"Densely Connected Convolutional Networks,"* in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4700–4708.

Appendix

Group Members Contribution:

- Almina Engür: was responsible for the design and implementation of the original custom CNN architecture and the selection of the Fruits-360 dataset subset used in the experiments. She also developed the preprocessing pipeline, including image resizing, grayscale conversion, normalization, and label encoding. In addition to the custom model, she implemented and fine-tuned the pre-trained ResNet-18 model using transfer learning. She actively contributed to the preparation of both the final report and the project presentation.
- Ahmet Arda Keskin: was responsible for adjusting the dataset to resolve the class imbalance and choosing class numbers such that the training was feasible on a computer within 1 day. He trained to model. He implemented the training pipeline with Batıkan Aybey Ulu and processed the results.
- Güzinsu Yaylacı: was responsible for implementation of the testing and the evaluation phase, including accuracy, precision, recall, and F1-score calculations from the confusion matrices. She was also responsible for leading the dataset selection, formatting the reports, writing the background information, selecting and inserting appropriate figures, preparing the references, constructing confusion matrices, and integrating one-hot encoding into the evaluation pipeline. She also led the finalization and analysis of the results based on the performance metrics, forming the result analysis and discussion in the final report.
- Batıkan Aybey Ulu: was responsible for implementing the training pipeline of the CNN model using NumPy with Ahmet Arda Keskin. They also worked on the forward pass, backpropagation, gradient updates, regularization, and parameter tuning for efficient learning. After the progress report, he helped the implementation of the DenseNet121 model and helped to write the final report of the project.
- Mehmet Oral: worked on convolution layer updating and back propagation. He implemented the testing and evaluation phase. Beside these, he worked on accuracy measurement, precision/recall analysis, confusion matrix construction, and integrating one-hot encoding logic into the evaluation process. After the Custom CNN, he implemented the DenseNet.