BBM 418 - Computer Vision Laboratory

Assignment 2 Report

Yusuf Demir
2210356074

Introduction

This report presents the implementation and evaluation of convolutional neural networks (CNNs) for food image classification using the Food-11 dataset. The study is structured into two main parts. In Part 1, CNN models are designed and trained from scratch, with and without residual connections. In Part 2, a transfer learning approach is employed using a pre-trained MobileNetV2 model. The goal is to assess the effects of different design strategies on model performance, generalization ability, and training behavior.

Dataset and Tools

The dataset used is Food-11, which consists of 11 food categories including apple pie, fried rice, sushi, etc. The dataset is split into 3 parts: training (60%), validation (20%), and test (20%). Each class contains hundreds of medium-resolution RGB images. All images are resized to 256×256 and normalized using ImageNet mean and standard deviation.

PyTorch was used as the main framework for implementing and training the models. Training and evaluation were performed using Kaggle's GPU environment, which provided sufficient computational resources for both training from scratch and fine-tuning pre-trained models.

Part 1: CNN from Scratch

Model 1: Classical CNN

We implemented a standard convolutional neural network consisting of 5 convolutional layers and 2 fully connected layers. Each convolutional layer is followed by Batch Normalization, ReLU activation, and Max Pooling. Dropout is applied before the final classification layer for regularization.

The architecture is as follows:

- Conv1: $3 \rightarrow 32$ (kernel size: 3, padding: 1)
- Conv2: $32 \rightarrow 64$
- Conv3: $64 \rightarrow 128$
- Conv4: $128 \rightarrow 256$
- Conv5: $256 \rightarrow 512$
- FC1: 1024 neurons
- FC2: 11-class softmax output

Model 2: Residual CNN

This model integrates residual blocks to improve gradient flow in deeper layers. The residual structure includes skip connections that add the input of a layer to its output.

The network structure is similar to the Classic CNN, but the intermediate convolutional blocks are replaced with residual blocks as follows:

- Conv1 + BN + ReLU
- ResidualBlock(32, 64)
- ResidualBlock(64, 128)

- ResidualBlock(128, 256)
- ResidualBlock(256, 512)
- FC1 + Dropout + ReLU
- FC2 (11 outputs)

Training Setup and Parameters

- Number of epochs: 50
- Learning rates tested: 0.001, 0.0005, 0.0001
- Batch sizes tested: 32, 64
- Optimizer: Adam with weight decay $1e^{-4}$
- Loss function: CrossEntropyLoss
- Accuracy was calculated using top-1 prediction match ratio

Results: Model Comparisons

We experimented with 3 learning rates (0.001, 0.0005, 0.0001) and 2 batch sizes (32, 64), resulting in 6 training configurations for both Classic and Residual CNN models. For each configuration, training and validation accuracy and loss were recorded for 50 epochs.

The best-performing configuration for the Classic CNN was achieved with a learning rate of 0.0001 and batch size 32, reaching a validation accuracy of **66.55%**. For the Residual CNN, the best result came from using a learning rate of 0.0001 and batch size 64, yielding a validation accuracy of **61.82%**.

The training and validation curves for these best-performing models are shown in Figures 1 and 2.

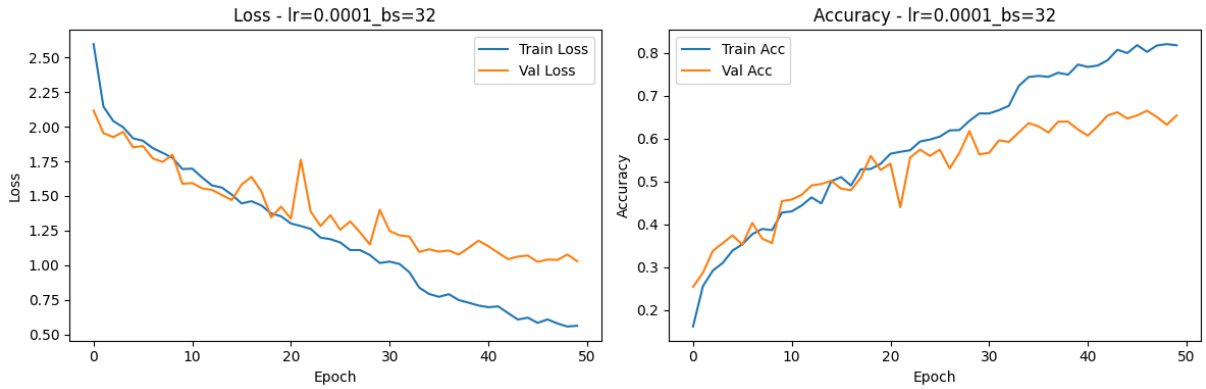

Figure 1: Training and validation accuracy/loss for best Classic CNN configuration (lr=0.0001, bs=32)

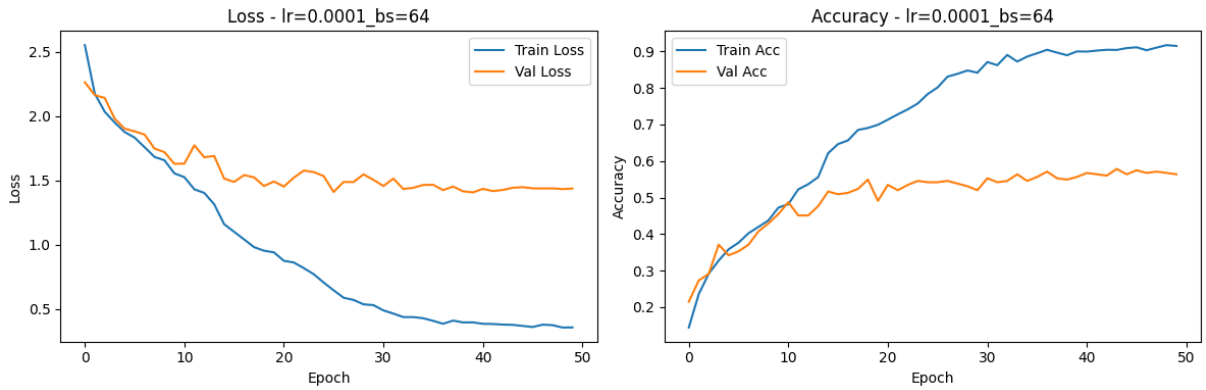

Figure 2: Training and validation accuracy/loss for best Residual CNN configuration (lr=0.0001, bs=64)

Although the Classic CNN achieved the highest validation accuracy, it also showed moderate overfitting, with training accuracy surpassing 85% while validation accuracy plateaued around 66%. The Residual CNN configuration performed slightly worse in terms of validation accuracy and showed more unstable training behavior, with larger fluctuations in validation loss and a wider gap between training and validation performance. Despite the benefits of residual connections in theory, the residual model in this case did not generalize better than the classic model.

Dropout Experiments

To mitigate overfitting, we conducted dropout experiments by modifying both Classic and Residual CNN architectures to include a dropout layer before the final classification layer. Dropout was applied right after the fully connected layer (`fc1`), before the final output layer. This design aims to prevent the model from relying too heavily on specific activations and encourages more robust feature representations.

Two different dropout rates were tested: 0.3 and 0.5. Models were retrained using the best hyperparameter configurations identified from Part 1 (i.e., optimal learning rate and batch size). Validation accuracy was monitored during training, and the best-performing model based on validation accuracy was saved and later evaluated on the test set.

The test set results are summarized below:

Model	Dropout	Test Accuracy	Test F1 Score	Precision
Classic CNN	0.3	39.27%	38.19%	40.02%
Classic CNN	0.5	30.91%	28.43%	29.69%
Residual CNN	0.3	45.82%	45.21%	45.74%
Residual CNN	0.5	43.27%	43.93%	46.15%

Interpretation: Overall, applying dropout improved generalization compared to some baseline overfitting cases, but the improvements were limited. For both architectures, the model with a dropout rate of 0.3 yielded better performance than 0.5, suggesting that excessive regularization (via high dropout) harmed learning capacity. The Classic CNN suffered more from higher dropout, with a sharp decrease in accuracy and F1 score when using 0.5.

Among all dropout configurations, the best test performance was achieved by the Residual CNN with dropout rate 0.3, which reached **45.82%** test accuracy. However, this still fell short of the best model trained without dropout (Classic CNN with **66.55%** validation accuracy), indicating that dropout was not a definitive solution for improving generalization in this setup.

Conclusion: Dropout is a useful regularization technique, especially when overfitting is evident. However, in our case, since the dataset was moderately sized and batch normalization was already applied, dropout provided limited gains. Fine-tuning other architectural elements or data augmentation might yield better results.

Confusion Matrix

Confusion matrices for the best models of each architecture are shown below. These matrices help visualize how well the models classified different food categories from the Food-11 dataset.

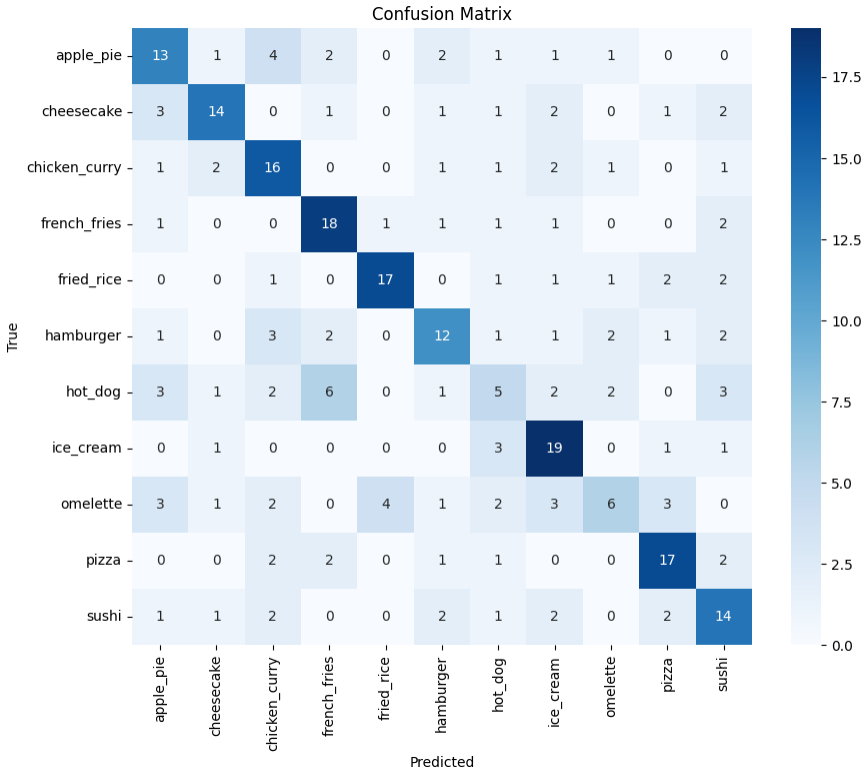

Figure 3: Confusion matrix for best Classic CNN model (lr=0.0001, bs=32)

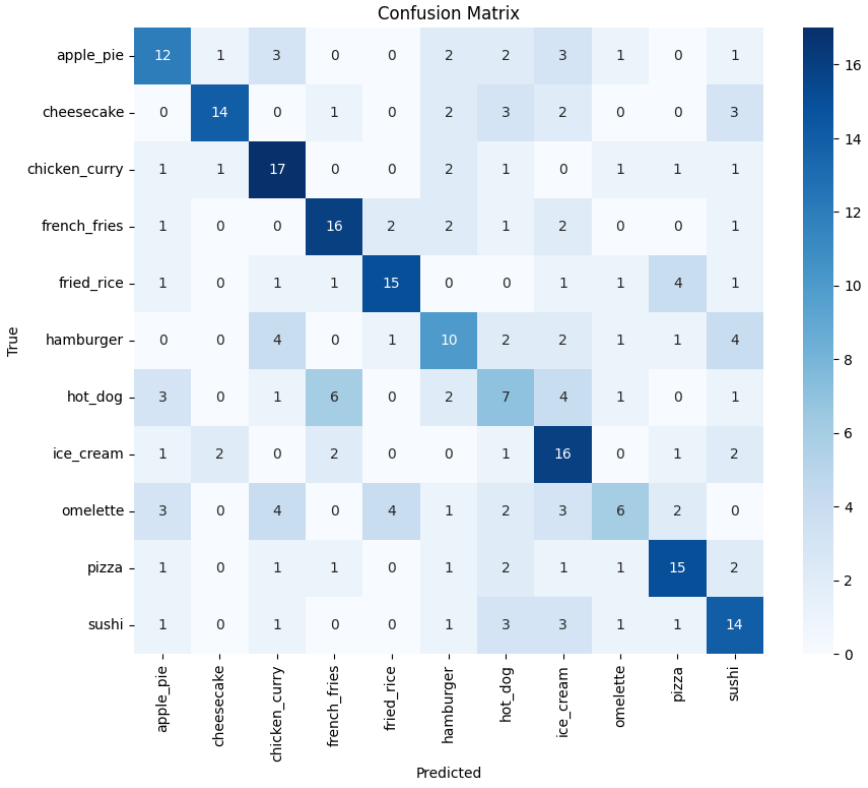

Figure 4: Confusion matrix for best Residual CNN model (lr=0.0001, bs=64)

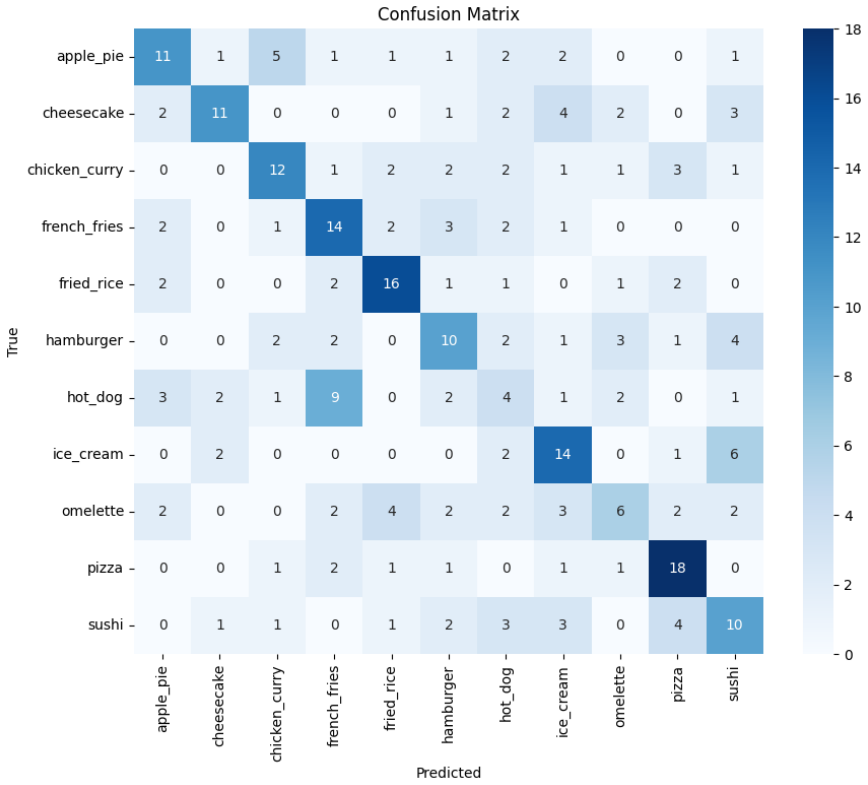

Figure 5: Confusion matrix for best Residual CNN with Dropout model (dropout=0.3)

From the matrices, we observe that:

- The Classic CNN model correctly classified most of the *french_fries*, *ice_cream*, and *pizza* instances but showed confusion between visually similar classes like *omelette*.
- The Residual CNN model provided overall slightly improved diagonal dominance, especially for *chicken_curry* and *cheesecake*, but still had difficulties with *hot_dog* and *omelette*.
- Adding dropout to the Residual CNN improved class separability in categories such as *pizza*, but some confusion remained.

Analysis and Interpretation

Across all experiments in Part 1, several key observations emerged:

- Lower learning rates (especially 0.0001) consistently produced better validation results for both Classic and Residual CNNs, helping the models converge more smoothly and avoid overshooting.
- The best Classic CNN configuration achieved a validation accuracy of **66.55%** with $(lr = 0.0001, bs = 32)$, while the best Residual CNN configuration reached **61.82%** with $(lr = 0.0001, bs = 64)$.
- Dropout was added after the first fully connected layer in both architectures to reduce overfitting. Although it did slightly improve generalization for the Residual CNN, its best dropout configuration still reached only **49.09%** accuracy, which is below the non-dropout models.
- Confusion matrices showed that all models had difficulty distinguishing between certain visually similar classes.
- Residual CNNs generally provided more consistent learning dynamics across different settings and showed more stability in training curves. However, Classic CNNs achieved higher peak accuracy when well-tuned.

In conclusion, while Residual CNNs are architecturally more robust and stable, in this specific setting the Classic CNN outperformed in terms of final accuracy. Dropout served as a regularization tool but did not surpass the performance of the original non-dropout models in this assignment.

Part 2: Transfer Learning

Selected Pretrained Model

In this part, we employed **MobileNetV2** as the pretrained model for transfer learning. MobileNetV2 is a lightweight architecture designed for efficient inference on mobile and embedded devices. It utilizes inverted residual blocks and depthwise separable convolutions to significantly reduce the number of parameters while maintaining competitive accuracy.

Fine-Tuning Explanation

Transfer learning leverages the knowledge of a model trained on a large-scale dataset (such as ImageNet) to improve performance on a smaller target dataset. Instead of training the entire network from scratch, we fine-tune the pretrained MobileNetV2 on the Food-11 dataset by experimenting with two strategies:

- **FC Only**: All convolutional layers are frozen and only the final fully connected (FC) classification layer is trained. This setup is fast and prevents overfitting but limits adaptation to the new dataset.
- **Last Two Blocks + FC**: In addition to the FC layer, the last two inverted residual blocks are unfrozen and fine-tuned. This allows the model to adapt high-level features to the new domain, improving generalization at the cost of higher computation.

Code snippet for freezing and modifying the FC layer:

```
# Case 1: FC Only
model = models.mobilenet_v2(pretrained=True)
model.classifier[1] = nn.Linear(model.last_channel, 11)
for param in model.features.parameters():
    param.requires_grad = False

# Case 2: Last 2 Blocks + FC
for param in model.parameters():
    param.requires_grad = False
for param in model.features[14:].parameters():
    param.requires_grad = True
for param in model.classifier.parameters():
    param.requires_grad = True
```

Training Strategies

In this part, we compared two fine-tuning strategies using MobileNetV2:

1. **Only FC Layer Trained:** All convolutional layers were frozen, and only the final fully connected (FC) layer was trained. This configuration achieved a validation accuracy of **89%** and a test accuracy of **80.00%**.
2. **Last Two Convolutional Blocks + FC Layer Trained:** The last two inverted residual blocks and the FC layer were fine-tuned, while the rest of the model remained frozen. This approach reached a validation accuracy of **91%** and improved the test accuracy to **85.82%**.

The following table summarizes the performance metrics:

Model	Accuracy	Precision	Recall	F1 Score
MobileNetV2 FC Only	0.8000	0.8073	0.8000	0.8004
MobileNetV2 Last 2 Blocks + FC	**0.8582**	**0.8720**	**0.8582**	**0.8591**

As observed, fine-tuning the last convolutional layers in addition to the FC layer led to significant improvements in both training and generalization. The added flexibility allowed the model to adapt more effectively to the Food-11 dataset, resulting in a higher test accuracy and better balance in the confusion matrix.

Confusion Matrix

Figure 6 shows the confusion matrix for the best transfer learning model: MobileNetV2 with last two blocks + FC layer fine-tuned. Most classes were predicted with high confidence, especially structured dishes like *french_fries*, *hamburger*, and *ice_cream*, which reached near-perfect classification. Some confusion was observed in visually similar classes such as *omelette*.

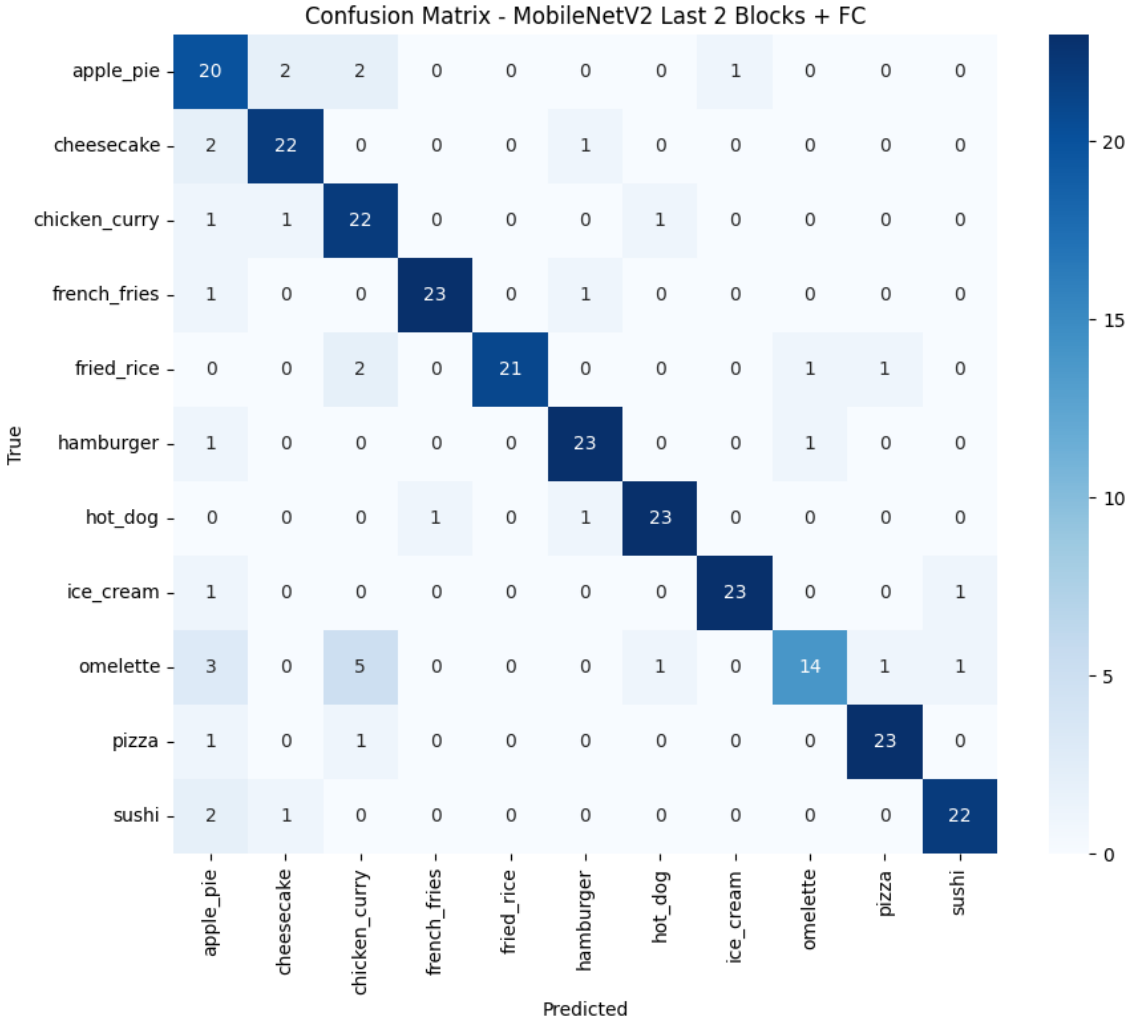

Figure 6: Confusion Matrix - MobileNetV2 Last 2 Blocks + FC Layer

Part 2 Analysis

Transfer learning with MobileNetV2 proved highly effective for food image classification. Training only the FC layer provided a fast and robust baseline. However, unfreezing the last two convolutional blocks further boosted performance by adapting mid- and high-level features to the target dataset.

Key observations include:

- Fine-tuning deeper layers significantly improved generalization and test accuracy.
- Training curves were smooth, with no signs of overfitting.
- Transfer learning outperformed all from-scratch CNN models in both accuracy and training time.

Overall, combining pretrained knowledge with targeted fine-tuning offers an excellent balance between efficiency and performance for image classification tasks with limited data.

General Comparison and Conclusion

This assignment compared two major approaches for image classification on the Food-11 dataset: training CNNs from scratch (Part 1) and applying transfer learning using MobileNetV2 (Part 2).

Comparison of Part 1 and Part 2 Results

- The best result from Part 1 (Classic CNN, lr=0.0001, bs=32) achieved a test accuracy of **66.55%**, while the best residual model reached **61.82%**.
- Applying dropout to improve generalization only helped marginally. The best dropout-enhanced residual model reached **49.09%**, which is still significantly lower than the original non-pretrained models.
- In Part 2, transfer learning with MobileNetV2 performed substantially better. The best configuration (last two blocks + FC fine-tuned) achieved a test accuracy of **85.82%**—a **20% absolute improvement** over the best CNN-from-scratch model.

Strengths and Weaknesses

CNNs from Scratch:

- **Strengths:** Full control over architecture and training; useful for domain-specific learning.
- **Weaknesses:** Requires much more data and compute power; prone to overfitting or underfitting; unstable training curves; lower generalization performance.

Transfer Learning:

- **Strengths:** Faster convergence; higher accuracy; more robust to overfitting; requires less data; better use of pretrained features.
- **Weaknesses:** Requires pretrained models; limited adaptability to very different domains; more prone to compatibility issues between architecture and dataset.

Overall, transfer learning was the superior approach in both performance and efficiency. However, understanding the fundamentals and limitations of CNN architectures built from scratch remains essential for scenarios where pretrained models are not applicable.