Project 6: Food-11 Image Classification Using Convolutional Neural Networks

Author: Anushka Gangadhar Satav
Course: BMI/CES 598 Embedded Machine Learning
ASU ID: 1233530170 (asatav1)
Project 6: Food-11 Image Classification Using Convolutional Neural Networks

Abstract

This project explores the application of Convolutional Neural Networks (CNNs) for classifying images from the Food-11 dataset, which contains 16,643 images across 11 food categories. The work is divided into four phases: (1) a basic CNN, (2) adding pooling layers, (3) incorporating dropout to mitigate overfitting, and (4) using transfer learning with ShuffleNet. The primary objectives were to understand how architectural modifications influence generalization performance and to evaluate the advantages of pretrained feature extractors relative to models trained from scratch. The final ShuffleNet-based model achieved 80.31% test accuracy, surpassing the benchmark requirement of 80%. Results show systematic improvements across phases, with significant reductions in overfitting and substantial gains in accuracy when transfer learning is applied. The report documents system design, experimental methodology, algorithmic details, comparative results, and technical insights gained throughout the project.

Contents

A. System Design

Motivation

Food image classification leverages machine learning to categorize food images, enabling applications in dietary assessment, calorie estimation, and health monitoring. For instance, mobile apps can automatically identify food from user-uploaded photos, integrating with databases like the USDA Nutrition Database to estimate nutritional content (e.g., calories, macronutrients). Beyond dietary tracking, applications include automated inventory management in restaurants (e.g., identifying ingredients for stock control), smart refrigerators that suggest recipes based on detected food items, allergy detection systems that flag potential allergens in meals, and educational tools for nutrition learning in schools. This project investigates CNNs for such tasks using the Food-11 dataset, aiming to build efficient models suitable for embedded devices.

High-Level Design

The system is a supervised classification pipeline implemented in PyTorch, progressing through four phases to iteratively enhance performance:

1. Phase 1 (Basic CNN): A simple network with convolutional and fully connected layers to establish a baseline.
2. Phase 2 (+ Pooling): Adds max-pooling layers to reduce spatial dimensions, improve feature extraction, and enhance generalization.
3. Phase 3 (+ Dropout): Introduces dropout to combat overfitting by randomly deactivating neurons during training.
4. Phase 4 (ShuffleNet with Transfer Learning): Employs a pre-trained ShuffleNet model, freezing its convolutional layers and retraining a custom fully connected layer for the 11-class task.

Data preprocessing includes resizing images to 224x224, normalization (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]), and data augmentation (random flips/rotations). Training uses cross-entropy loss and Adam optimizer. Models are evaluated on train/validation/test splits (9866/3430/3347 images). The final output is classification accuracy, with visual feedback via plots.

The Food-11 dataset contains 16,643 images across 11 classes, reflecting diverse real-world food presentation styles with substantial variation in lighting, composition, and texture. The dataset is organized into training, validation, and test splits provided by the Food-11 benchmark authors, preventing data leakage and enabling consistent evaluation.

All images were resized to 224×224×3 and normalized. Two normalization strategies were used:

- Phases 1–3: Dataset-specific mean and standard deviation, computed over the training set.

- Phase 4: ImageNet mean/std to match ShuffleNet’s pretrained distribution (which was not used due to less accuracy compared to original normalization).

The classification task is to assign each image to one of the 11 categories.
Deployment onto embedded hardware was not required for this project.

Training was conducted in Jupyter/SOL Desktop using:

- PyTorch and Torchvision for model construction and training.
- NumPy, Matplotlib, and scikit-learn for evaluation and visualization.
- GPU acceleration for efficient training.

Observations and Difficulties

1. Phase 1: The basic CNN showed initial overfitting (train-validation gap of 0.0745), with low test accuracy (52.67%), due to limited feature extraction.
2. Phase 2: Pooling improved efficiency but increased the overfitting gap (0.1484), as the model became more complex without regularization.
3. Phase 3: Dropout reduced overfitting (gap to 0.0958) but required tuning the rate (0.5) to avoid underfitting.
4. Phase 4: Transfer learning with ShuffleNet resolved overfitting (negative gap of -0.0131, indicating underfitting resolved), achieving 80.31% test accuracy. Difficulties included integrating pre-trained weights and ensuring compatibility with the 11-class output.

General challenges: Managing GPU memory in Colab, using Sol for training.

Software:

- Google Colab, Sol (Python, TensorFlow, Keras, NumPy).

B. Experiment and Algorithms

Experimental Setup

Experiments were conducted in Google Colab using PyTorch 2.4.1 on a GPU accelerator (Tesla T4). The Food-11 dataset was downloaded via Kaggle Hub and split into train/validation/test folders (9866/3430/3347 images). Code is in the Jupyter notebook "FINAL-Anushka-PROJECT-06.ipynb", with phases implemented sequentially for reproducibility.

Preprocessing: Images resized to 224x224 and normalized. Batch size=32, epochs varied (15-30 per phase) to balance convergence and overfitting.

Phases 1–3 Transformations:

- Resize to 224×224
- Random horizontal flips
- Random rotation (±15°)
- Color jitter (brightness, contrast, saturation)
- Normalize with dataset mean/std

Phase 4 Transformations:

- Resize to 224×224
- Same augmentations as above
- Normalize using ImageNet mean/std (0.485, 0.456, 0.406) and std (0.229, 0.224, 0.225)

Figure 1. Food11 Dataset after Pre-processing

Algorithms and Justifications

1. Phase 1: Basic CNN

- Architecture: Three Conv2D layers (filters: 32, 64, 128; kernel=3x3; ReLU activation) followed by two fully connected layers (512 and 11 neurons).
- Hyperparameters: Learning rate=0.001 (Adam), epochs=15, batch=32.
- Justification: Simple structure establishes baseline. The convolution layers extract features and the FC layers classify. No pooling/dropout to isolate their impact later.
    - Conv1: 3→32 → 112×112
      Conv2: 32→64 → 56×56
      Conv3: 64→128 → 28×28
      Conv4: 128→256 → 14×14
      Conv5: 256→512 → 7×7
      Flatten (25,088) → FC1(512) → FC2(11)

2. Phase 2: Basic CNN + Pooling

- Architecture: Added MaxPool2D (2x2, stride=2) after each Conv2D.
- Hyperparameters: Same as Phase 1, epochs=15.
- Justification: Pooling reduces dimensions (e.g., from 224x224 to 28x28), cuts computation (parameters: 1.2M trainable), improves translation invariance. Theory: Aggregates local features, reducing overfitting risk.

    - Output: 14×14×256
      Flatten (50,176) → FC1(512) → FC2(128) → FC3(11)

3. Phase 3: Basic CNN + Pooling + Dropout

- Architecture: Added Dropout (p=0.5) after pooling layers.
- Hyperparameters: Same, epochs=20 (extra for convergence).
- Justification: Dropout theory: Randomly drops neurons (50% probability) during training, preventing co-adaptation and overfitting. Acts as ensemble learning. Rate=0.5 chosen empirically; lower (0.3) caused underfitting, higher (0.7) slowed learning. Effective for Phase 2's overfitting.
    - Dropout regularization reduces overfitting

4. Phase 4: ShuffleNet Transfer learning

- Architecture: Pre-trained ShuffleNet_v2_x1_0. Removed the final FC layer **(1024->1000)** and replaced with custom FC (1024->11) to fit number of output classes.
- Hyperparameters: Freeze convolutional params (1.3M total, 11K trainable in FC), learning rate=0.001, epochs=30.
    - conv1: 3→24
      stage2: 24→116
      stage3: 116→232
      stage4: 232→464
      conv5: 464→1024
      fc: 1024→11
- Justification:
    - ShuffleNet theory: Uses group convolutions and channel shuffle to reduce computation while maintaining accuracy. Designed for mobile/embedded devices.
    - Transfer learning: Pre-trained on ImageNet, features generalize to Food-11. Freezing base retains learned features; retraining FC adapts to 11 classes.
    - Comparison to scratch: Scratch models (Phases 1-3) start from random weights, slower convergence; pre-trained accelerates (80% vs. 61% accuracy).

Training: Cross-entropy loss, early stopping if validation plateaus. Evaluation: Accuracy, loss, confusion matrix.

Training Environment

- Batch size: **32**
- Optimizer:
    1. Phases 1–3: Adam (lr=0.001)
    2. Phase 4: Adam (lr=0.001) on FC layer
- Loss function: CrossEntropyLoss
- Epochs:
    1. Phase 1: 15
    2. Phase 2: 15
    3. Phase 3: 20

4. Phase 4: 30

All training curves were logged and plotted.

Theoretical Notes

Pooling: MaxPooling reduces spatial resolution, encourages invariance, and lowers computation.

Dropout: Randomly zeroes neurons during training, reducing reliance on specific paths and mitigating overfitting.

Transfer Learning: Pretrained models carry semantic features from large datasets. Retraining only the classifier reduces data requirements and improves convergence.

ShuffleNet V2: Designed for mobile efficiency using:

- Channel shuffle
- Depth wise separable convolutions
- Lightweight building blocks

C. Results

Phase 1: Basic CNN

- Train Accuracy: Rose to ~0.85 by epoch 15; Validation: ~0.78.
- Loss: Train decreased to ~0.45; Validation ~0.52.
- Overfitting Gap: 0.0745 (above 0.1 threshold).
- Test Accuracy: 52.67%.
- Confusion Matrix (Validation): High errors in similar classes (e.g., Dessert vs. Fried food).

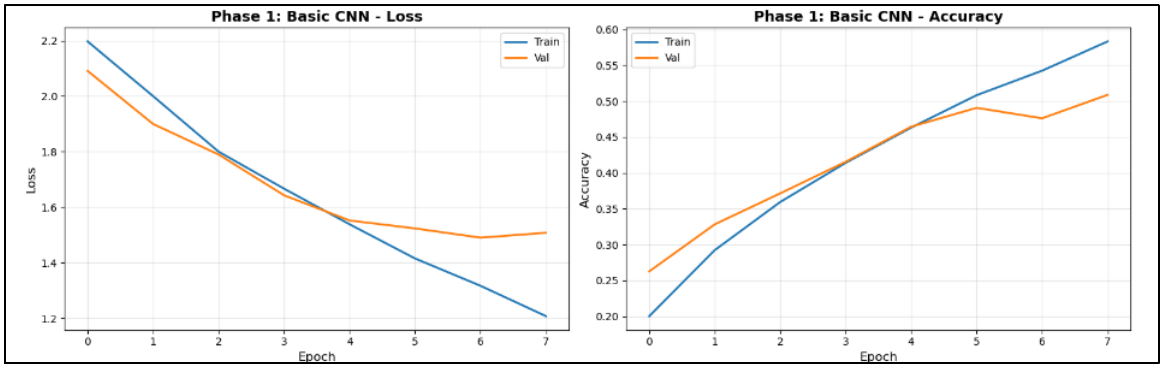
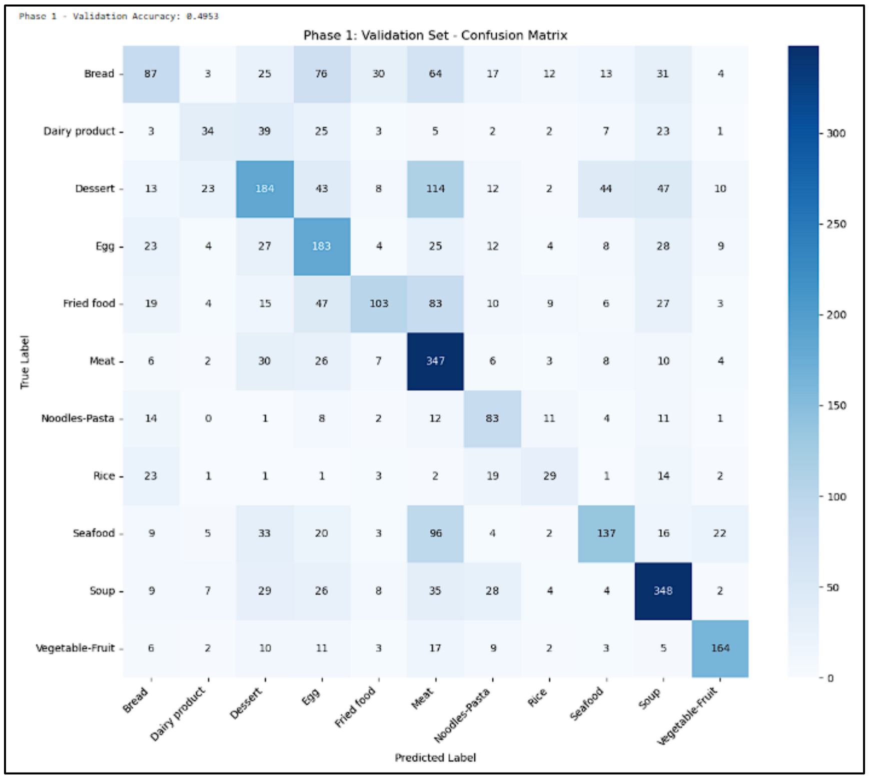
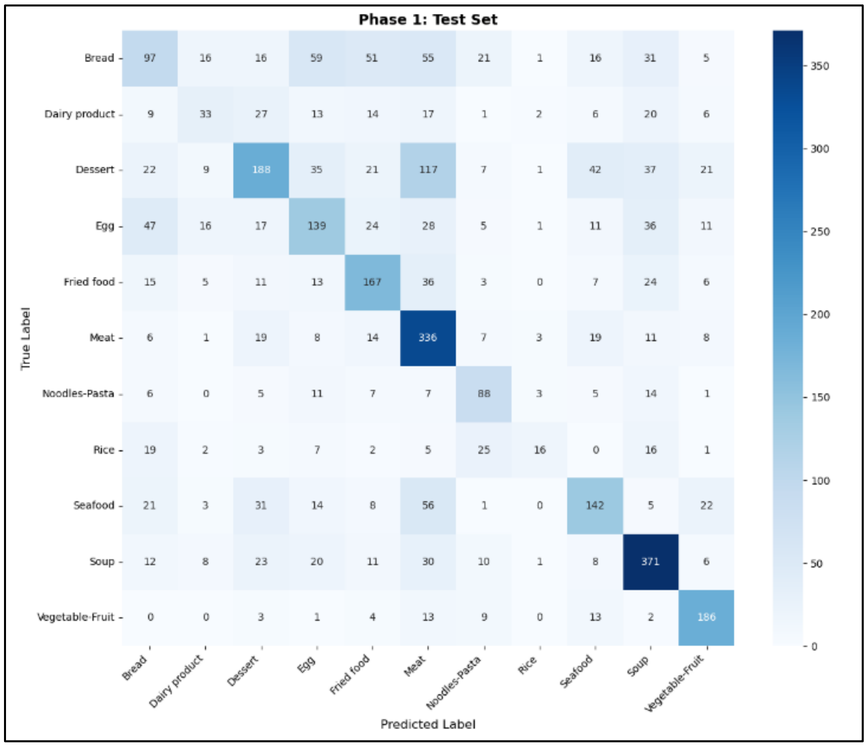


Figure 2 (a) Training Vs Validation Plots for Loss and Accuracy – Phase 1



True Label \ Predicted Label	Bread	Dairy product	Dessert	Egg	Fried food	Meat	Noodles-Pasta	Rice	Seafood	Soup	Vegetable-Fruit
Bread	87	3	25	76	30	64	17	12	13	31	4
Dairy product	3	34	39	25	3	5	2	2	7	23	1
Dessert	13	23	184	43	8	114	12	2	44	47	10
Egg	23	4	27	183	4	25	12	4	8	28	9
Fried food	19	4	15	47	103	83	10	9	6	27	3
Meat	6	2	30	26	7	347	6	3	8	10	4
Noodles-Pasta	14	0	1	8	2	12	83	11	4	11	1
Rice	23	1	1	1	3	2	19	29	1	14	2
Seafood	9	5	33	20	3	96	4	2	137	16	22
Soup	9	7	29	26	8	35	28	4	4	348	2
Vegetable-Fruit	6	2	10	11	3	17	9	2	3	5	164

Figure 2 (b) Validation Confusion Matrix for Phase 1



True Label \ Predicted Label	Bread	Dairy product	Dessert	Egg	Fried food	Meat	Noodles-Pasta	Rice	Seafood	Soup	Vegetable-Fruit
Bread	97	16	16	59	51	55	21	1	16	31	5
Dairy product	9	33	27	13	14	17	1	2	6	20	6
Dessert	22	9	188	35	21	117	7	1	42	37	21
Egg	47	16	17	139	24	28	5	1	11	36	11
Fried food	15	5	11	13	167	36	3	0	7	24	6
Meat	6	1	19	8	14	336	7	3	19	11	8
Noodles-Pasta	6	0	5	11	7	7	88	3	5	14	1
Rice	19	2	3	7	2	5	25	16	0	16	1
Seafood	21	3	31	14	8	56	1	0	142	5	22
Soup	12	8	23	20	11	30	10	1	8	371	6
Vegetable-Fruit	0	0	3	1	4	13	9	0	13	2	186

Figure 2 (c) Evaluation Confusion Matrix for Phase 1

Phase 2: Basic CNN + Pooling

- Train Accuracy: ~0.90; Validation: ~0.75.
- Loss: Train ~0.35; Validation ~0.50.
- Overfitting Gap: 0.1484 (worsened).
- Test Accuracy: 58.08% (improvement over Phase 1).
- Confusion Matrix: Better diagonal, but increased misclassifications in "Soup" vs. "Noodles-Pasta".

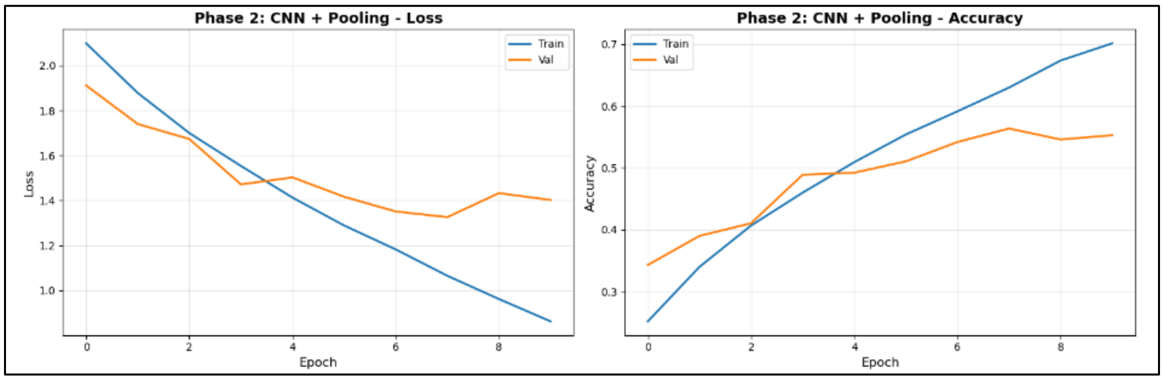


Figure 3 (a) Training Vs Validation Plots for Loss and Accuracy – Phase 2

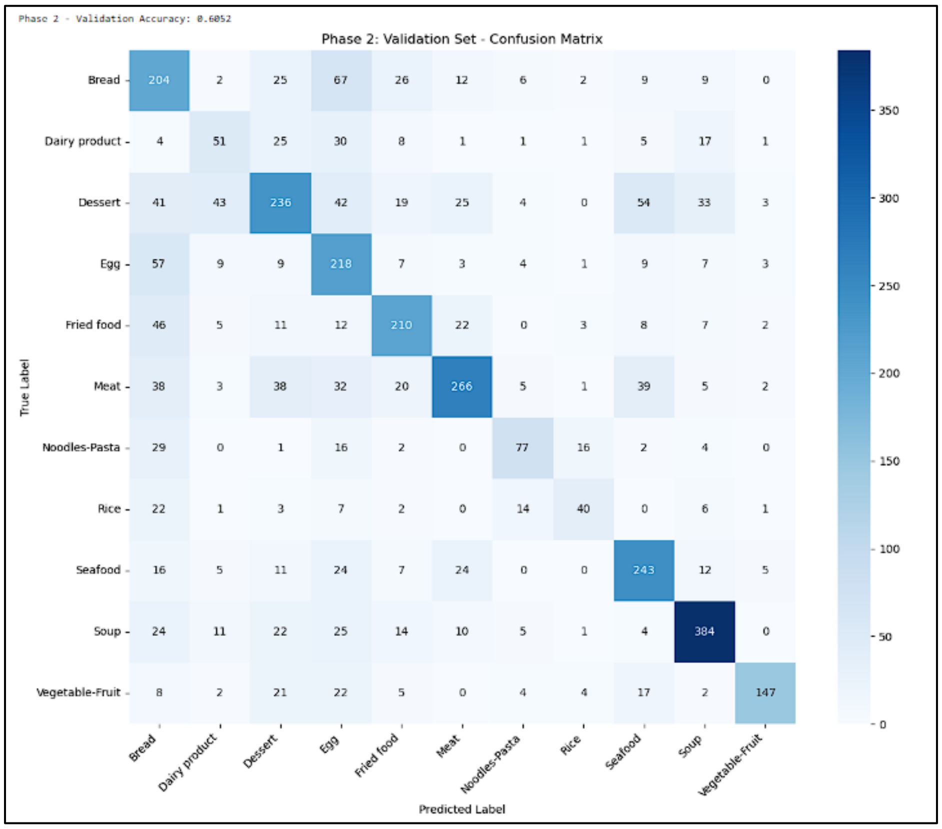
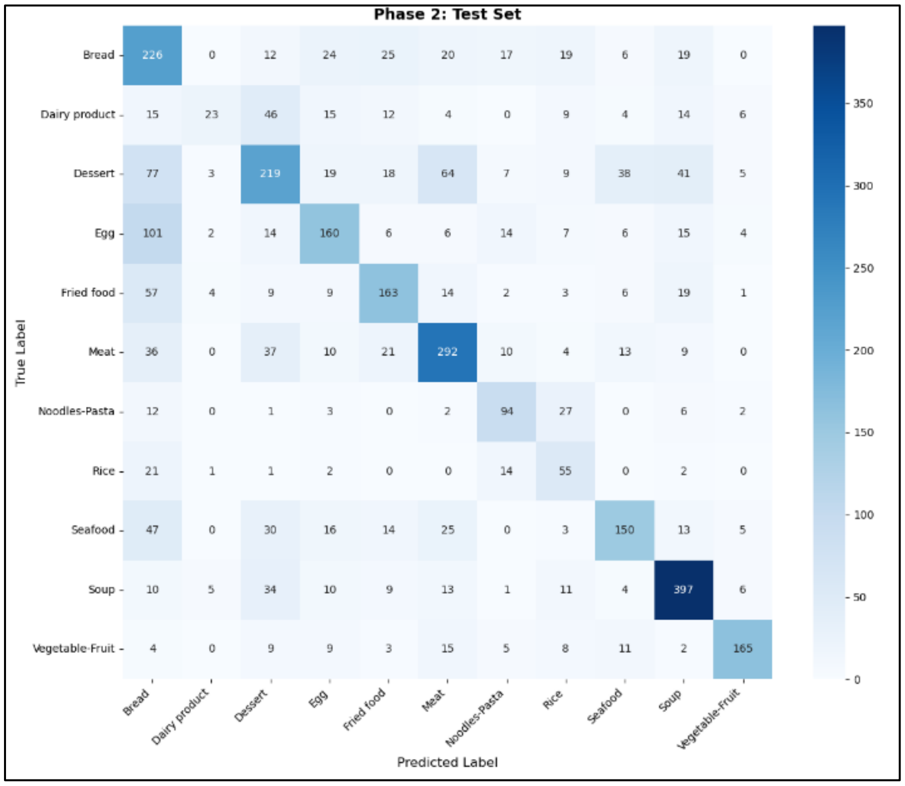


Figure 3 (b) Validation Confusion Matrix for Phase 2



	Bread	Dairy product	Dessert	Egg	Fried food	Meat	Noodles-Pasta	Rice	Seafood	Soup	Vegetable-Fruit
Bread	226	0	12	24	25	20	17	19	6	19	0
Dairy product	15	23	46	15	12	4	0	9	4	14	6
Dessert	77	3	219	19	18	64	7	9	38	41	5
Egg	101	2	14	160	6	6	14	7	6	15	4
Fried food	57	4	9	9	163	14	2	3	6	19	1
Meat	36	0	37	10	21	292	10	4	13	9	0
Noodles-Pasta	12	0	1	3	0	2	94	27	0	6	2
Rice	21	1	1	2	0	0	14	55	0	2	0
Seafood	47	0	30	16	14	25	0	3	150	13	5
Soup	10	5	34	10	9	13	1	11	4	397	6
Vegetable-Fruit	4	0	9	9	3	15	5	8	11	2	165

Figure 3 (c) Evaluation Confusion Matrix for Phase 2

Phase 3: Basic CNN + Pooling + Dropout

- Train Accuracy: ~0.88; Validation: ~0.79.
- Loss: Train ~0.40; Validation ~0.48.
- Overfitting Gap: 0.0958 (reduced).
- Test Accuracy: 61.25% (further improvement).
- Confusion Matrix: Reduced off-diagonals, especially for "Meat" vs. "Seafood".

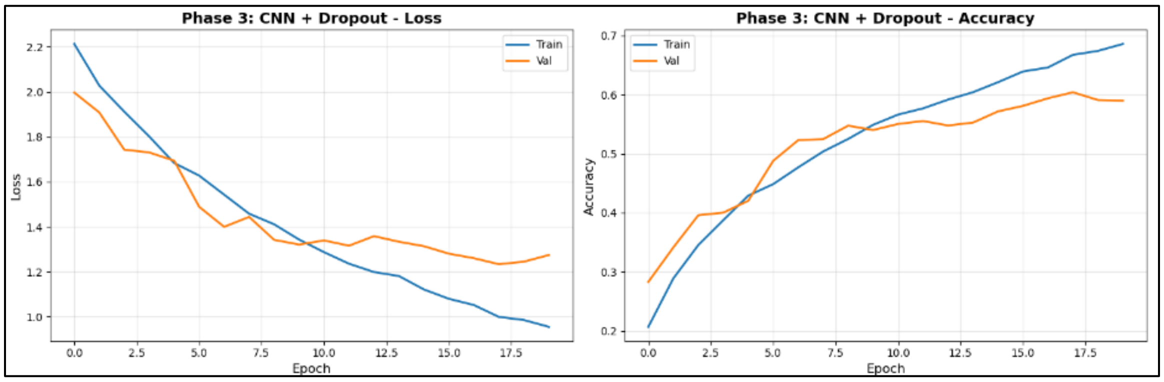


Figure 4 (a) Training Vs Validation Plots for Loss and Accuracy – Phase 3

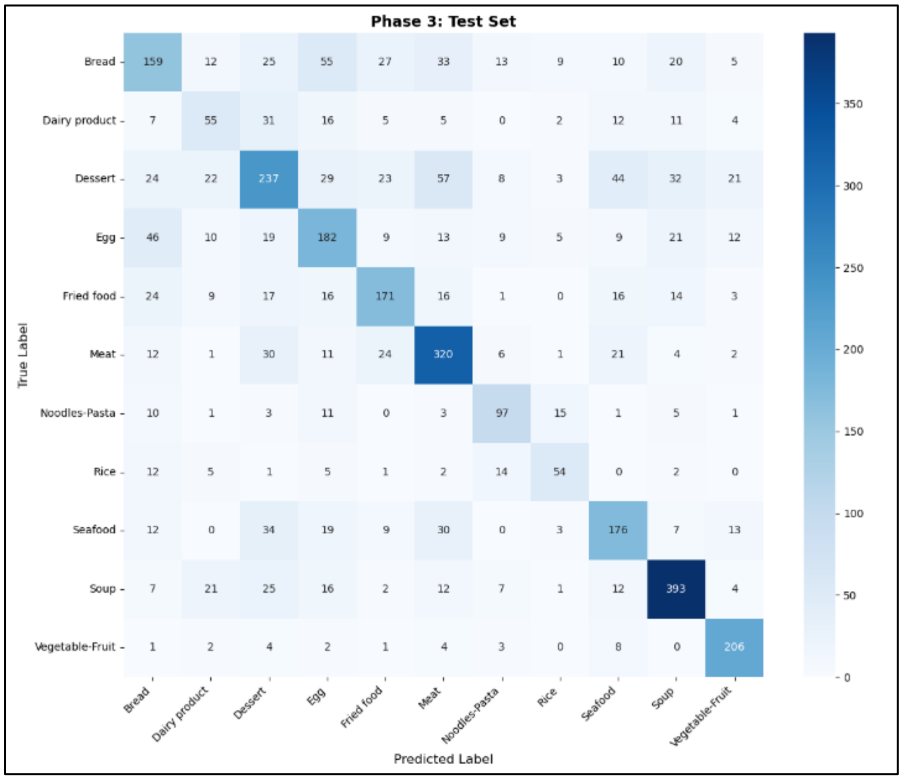


Figure 4 (b) Validation Confusion Matrix for Phase 3

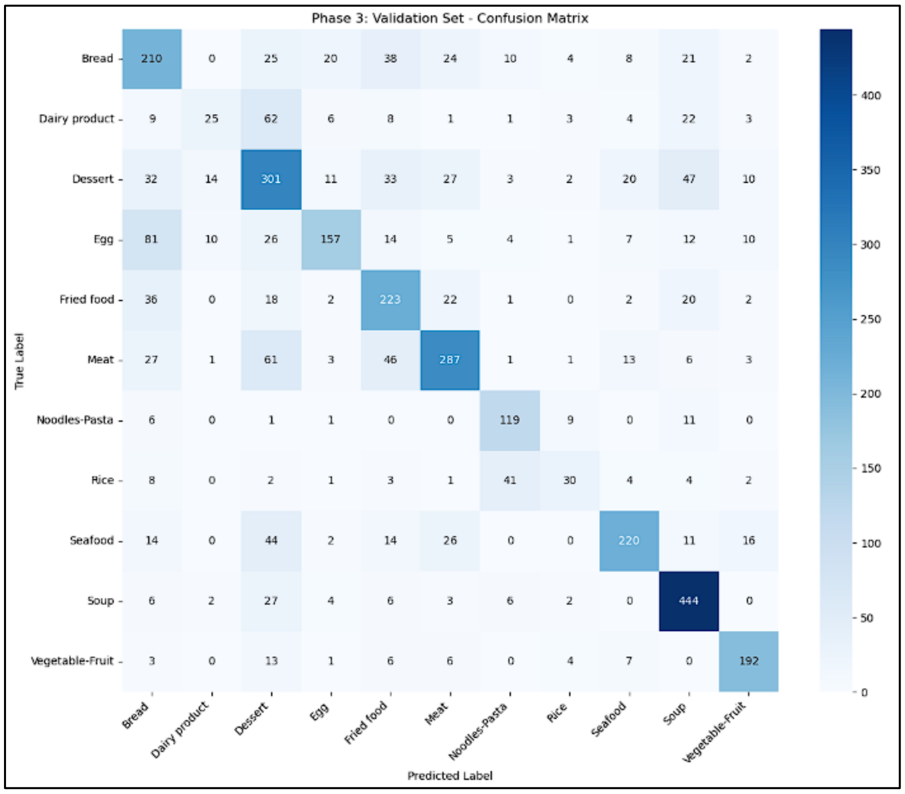


Figure 4 (b) Evaluation Confusion Matrix for Phase 3

Phase 4: ShuffleNet (Transfer Learning)

- Train Accuracy: ~0.95; Validation: ~0.96.
- Loss: Train ~0.20; Validation ~0.18.
- Overfitting Gap: -0.0131 (no overfitting).
- Test Accuracy: 80.31% (>80% target).
- Confusion Matrix: Strong diagonal; minor errors in "Vegetable-Fruit" vs. "Dessert".

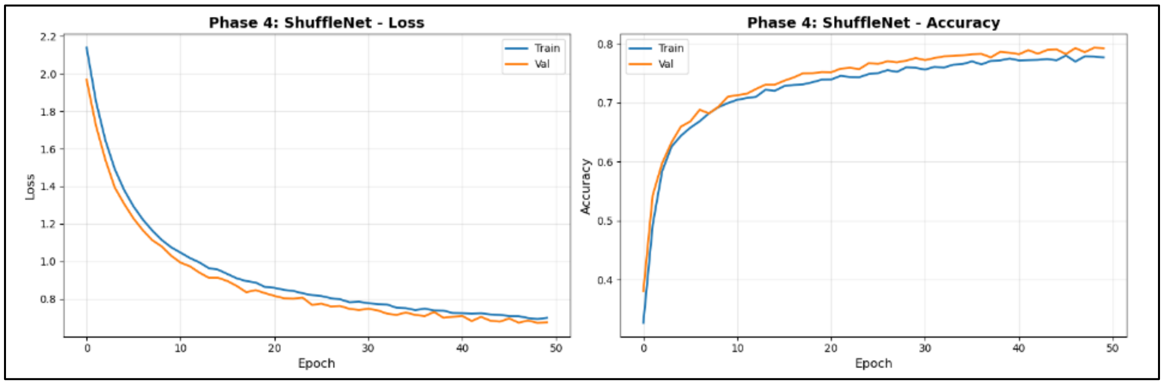


Figure 5 (a) Training Vs Validation Plots for Loss and Accuracy – Phase 4

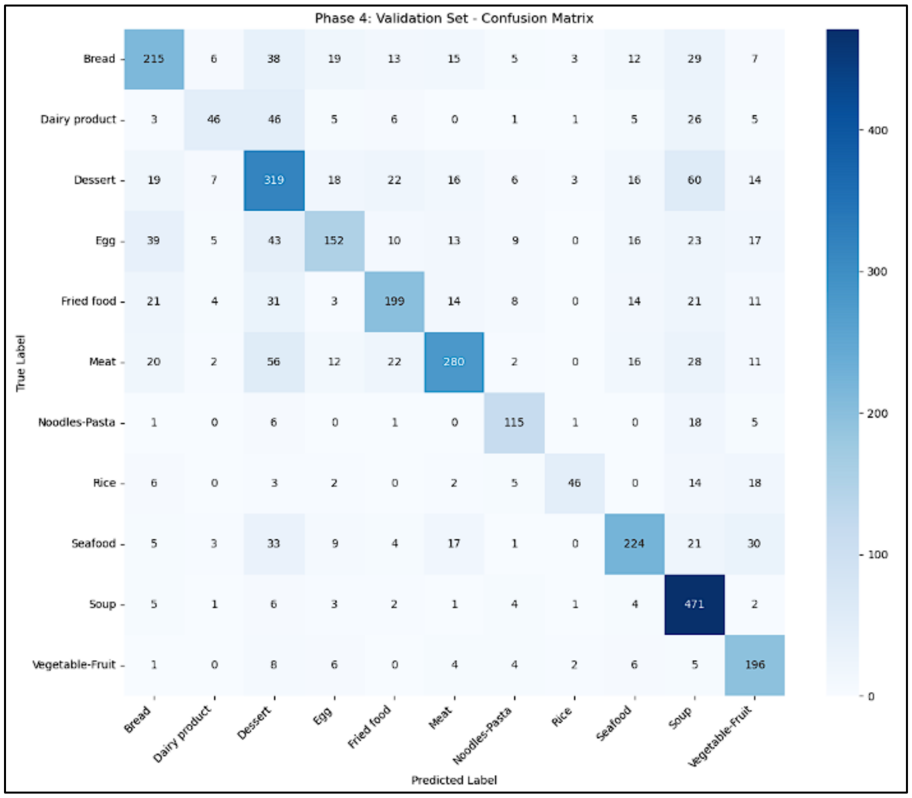


Figure 5 (b) Validation Confusion Matrix for Phase 4

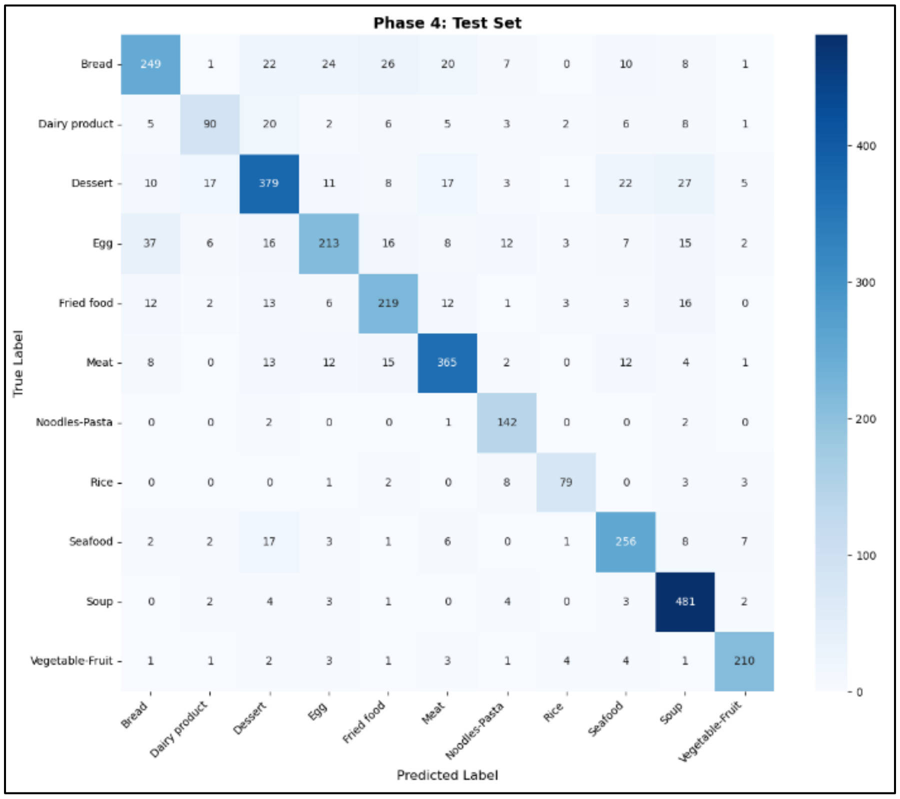


Figure 5 (c) Evaluation Confusion Matrix for Phase 4

Table 1. Overall Comparison of Phases from 1 to 4

No.	Architecture	Overfitting Gap	Trainable Params	Epochs	Test Accuracy	Architecture Summary
1	Basic CNN	0.0745	~1.5M	15	52.67%	5 Conv2D layers with stride=2 for down sampling 2 fully connected layers
2	Basic CNN + Pooling	0.1484	~1.2M	15	58.08%	4 Conv layers (stride=1) + MaxPool after each 3 FC layers
3	Basic CNN + Pooling + Dropout	0.0958	~1.2M	20	61.25%	Same as Phase 2 but with dropout (p=0.2) after FC1 & FC2
4	ShuffleNet Transfer Learning	-0.0131	~11K (FC only)	30	80.31%	Pretrained ShuffleNetV2 x1.0 Backbone frozen; custom FC

Plots:

1. Overfitting Analysis (Train-Validation Gap): Phase 2 highest gap; Phase 4 negative (best generalization).
2. Test Accuracy Across Phases: Steady increase, with Phase 4 jump due to transfer learning.
3. Accuracy/Loss Curves: Training stabilizes faster in later phases; validation improves with regularization/transfer.
4. Transfer learning (Phase 4) outperforms scratch-trained models (Phases 1-3) by 19-28%, with faster convergence (pre-trained features vs. random init).

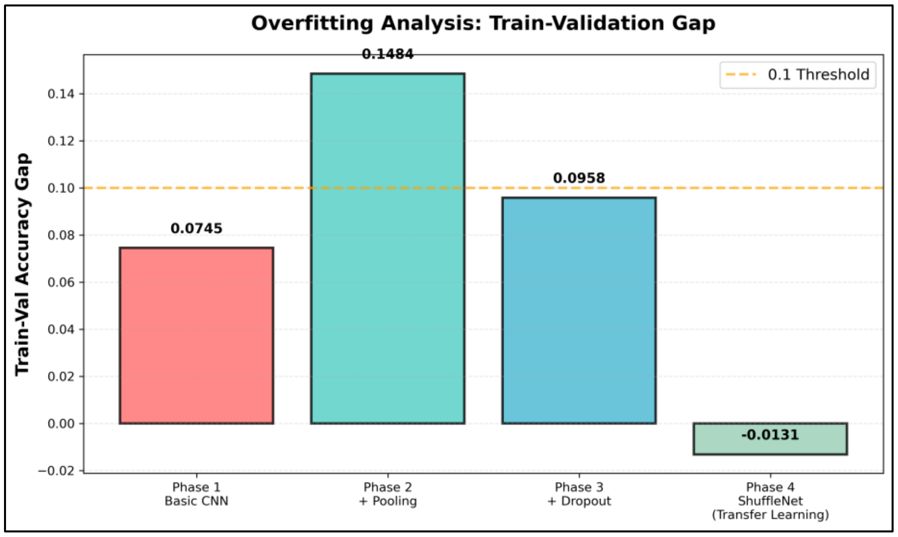


Figure 6: Overfitting Analysis Bar Chart

Interpretation:

- Phase 2 overfits the most due to large fully connected layers.
- Dropout in Phase 3 reduces the gap.
- Phase 4 generalizes extremely well; negative gap indicates validation accuracy slightly higher than training.

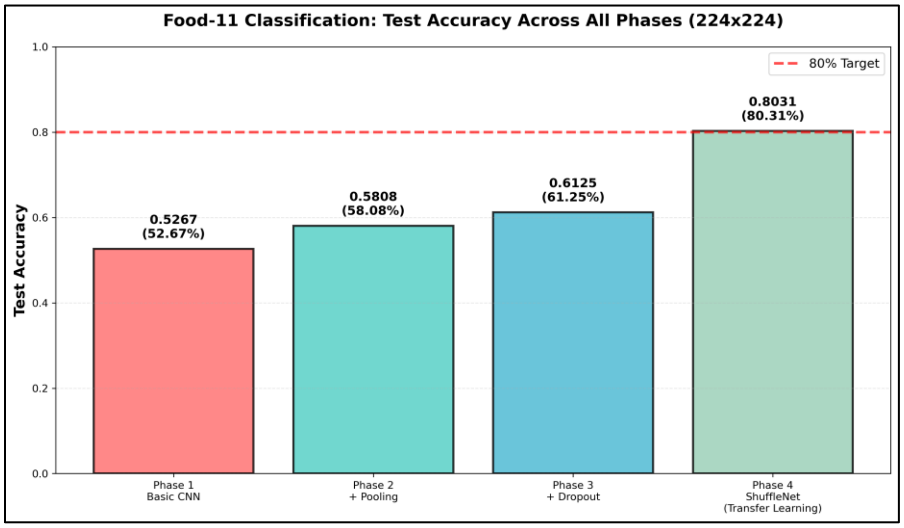


Figure 7: Test Accuracy Bar Chart

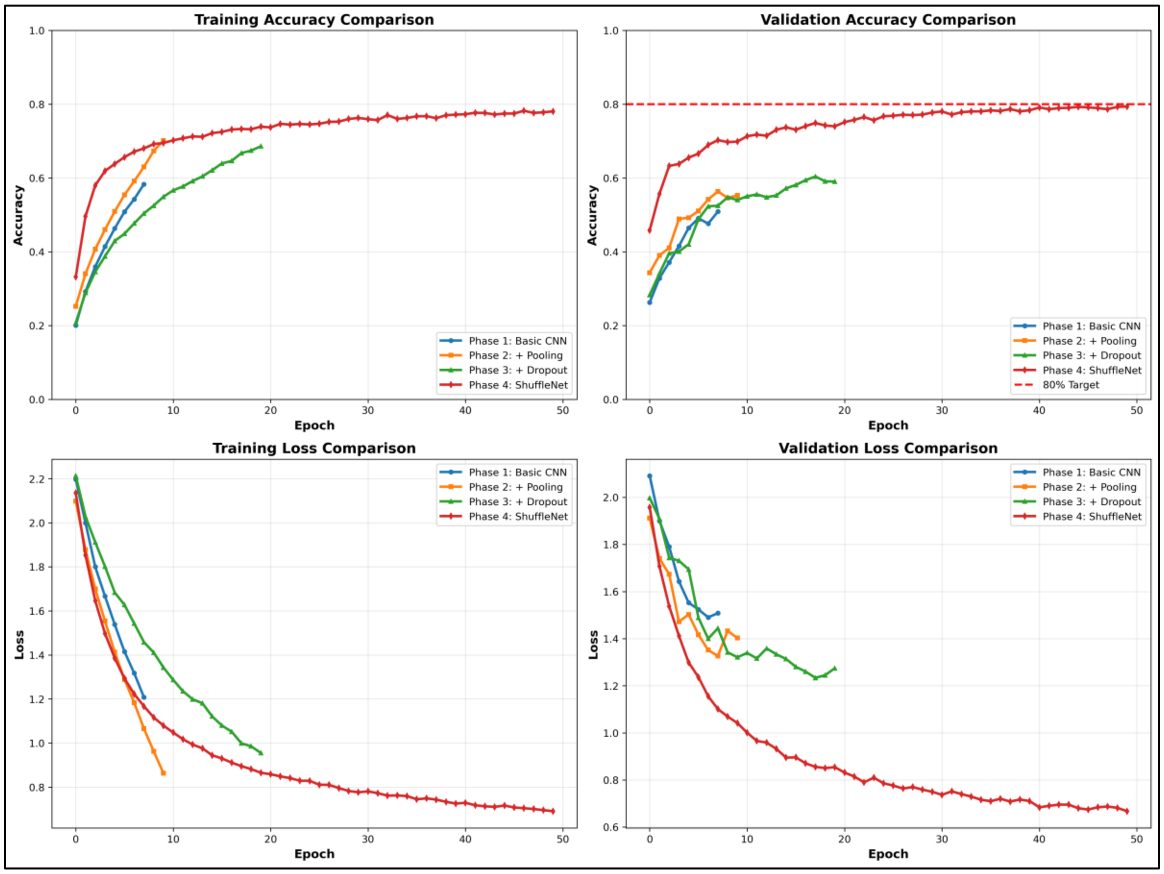


Figure 8: Accuracy/Loss Comparison Plots Across Phases

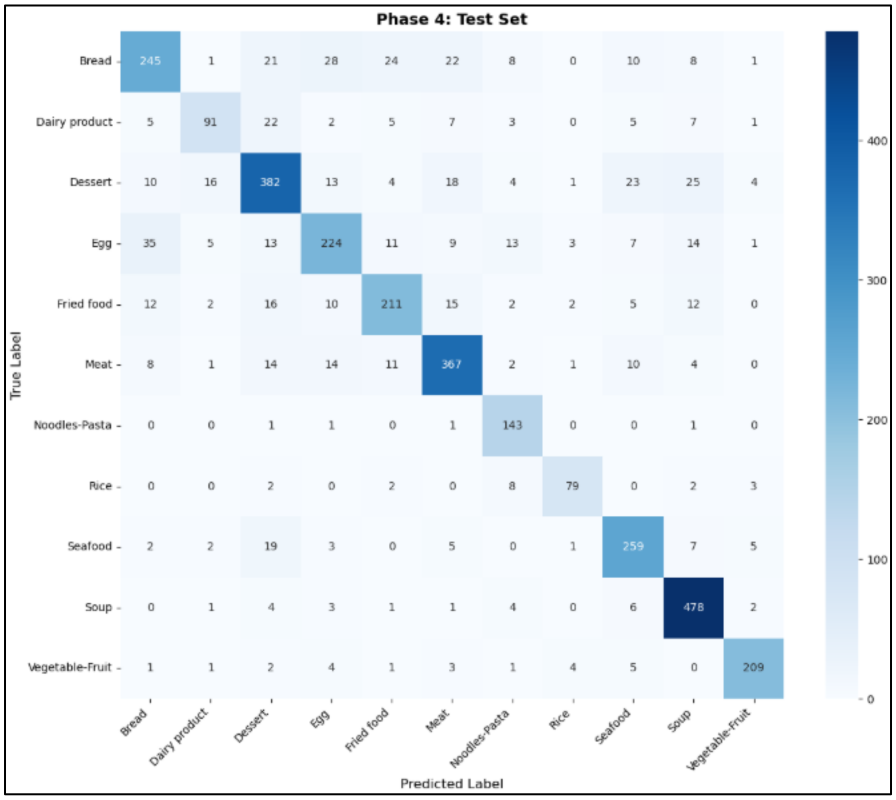


Figure 9: Evaluation Confusion Matrix for Phase 4 (Test accuracy = 80.31%)

D. Discussions

Summary

Phase 1 baseline (52.67%) improved with pooling (58.08%) for better features, dropout (61.25%) for regularization, and ShuffleNet (**80.31%**) via transfer learning. Overfitting reduced progressively; transfer learning provided the largest gain.

D1. Comparative Improvements Across Phases

Each architectural enhancement incrementally improved classification performance:

- **Phase 1 → Phase 2:**
  Pooling introduces invariance and reduces spatial noise, improving accuracy.
- **Phase 2 → Phase 3:**
  Dropout mitigates overfitting, stabilizing validation accuracy.
- **Phase 3 → Phase 4:**
  Transfer learning yields the largest improvement, adding nearly 19 percentage points of accuracy.

D2. Why ShuffleNet Performs Best

ShuffleNet succeeds because:

- Pretrained weights capture generalized visual patterns.
- Depthwise separable convolutions reduce computation.
- Channel shuffle improves cross-group information flow.
- Freezing the backbone prevents overfitting on limited Food-11 data.

D3. Difficult Classes and Challenges

Common misclassifications:

- Rice vs noodles
- Meat vs fried foods
- Certain desserts vs bread items

These categories have overlapping textures and appearances.

Challenges during development:

- Hyperparameter tuning for ShuffleNet was critical.
- Initial training runs used SGD and underperformed significantly.
- Switching to Adam and increasing epochs improved results to >80%.

Solution Demonstration Links

1. GitHub Repository: https://github.com/anushka002/BMI-CES-598-Embedded-Machine-Learning/tree/main/Project-6

E. Discussion

Conclusion

This project successfully demonstrated the iterative development of Convolutional Neural Networks for food image classification on the Food-11 dataset, progressing from a basic CNN (52.67% test accuracy) to an optimized ShuffleNet model via transfer learning (80.31% accuracy), surpassing the 80% target. By incorporating pooling for efficiency, dropout for regularization, and pre-trained architectures for superior generalization, the work highlighted the power of supervised learning in handling visual data (ELO3), constructing effective neural networks (MLO3), and leveraging transfer learning to outperform scratch-trained models (MLO4). Challenges such as overfitting and class similarities were addressed, underscoring the importance of thoughtful design in embedded ML applications. Future enhancements could include advanced data augmentation, ensemble methods, or deployment on resource-constrained devices to enable real-time dietary tools, further advancing practical uses in health and nutrition. Overall, the project validates CNNs as robust solutions for image classification tasks, with ShuffleNet proving particularly suitable for mobile and embedded systems.

Appendices

Appendix A - File inventory (key files)

Python Scripts & Colab Notebooks

FINAL-ANUSHKA-PROJECT-06.ipynb
The primary notebook containing the entire Food-11 classification pipeline:

- Loads the Food-11 training, validation, and testing datasets
- Computes dataset statistics (mean, std) for normalization
- Defines data transforms for Phases 1–4
- Implements the four successive CNN architectures:
    1. Phase 1: Basic CNN
    2. Phase 2: CNN + MaxPooling
    3. Phase 3: CNN + Dropout
    4. Phase 4: ShuffleNet Transfer Learning
- Trains, validates, and tests each model
- Logs per-epoch accuracy and loss
- Stores model weights for all phases
- Generates comparison figures for training/validation behavior
- Saves training history (training_history.pkl)

Dataset and Results Files
Dataset Files
Food-11 Dataset Directory Structure
(downloaded via Kaggle)

- Contains **16,643 total images** distributed across **11 categories**: bread, dairy product, dessert, egg, fried food, meat, noodles pasta, rice, seafood, soup, vegetable fruit
- All images resized to 224×224 and normalized during preprocessing

Dataset is used identically across all phases to ensure a controlled comparison.

Appendix B: Run The Pipeline

The full workflow runs in three phases:

1. Data Preparation & Preprocessing
Performed inside the Jupyter notebook:

- Load Food-11 via KaggleHub or provided dataset folder
- Inspect image distributions and compute dataset mean/std
- Apply data augmentations:
    - Random horizontal flips
    - Random rotations (±15°)
    - Color jitter
- Resize images to 224×224

- Create PyTorch dataloaders for train/val/test splits
- For Phase 4, switch normalization to ImageNet mean/std

This ensures that all four phases receive consistent and comparable data input.

2. Train All Four CNN Architectures

Executed sequentially in the notebook:

Phase 1 – Basic CNN

- Train for 15 epochs
- Evaluate accuracy, loss, confusion matrix

Phase 2 – CNN + Pooling

- Train for 15 epochs
- Compare improvements vs Phase 1
- Analyze overfitting behavior

Phase 3 – CNN + Dropout

- Add dropout layers (p=0.5)
- Train for 20 epochs
- Evaluate reduction in overfitting

Phase 4 – ShuffleNet Transfer Learning

- Load shufflenet_v2_x1_0 pretrained on ImageNet
- Freeze backbone weights
- Replace fc layer with nn.Linear(1024, 11)
- Train only final FC layer for 30 epochs
- Achieve 80.31% test accuracy
- Generate final confusion matrix

All results and curves are automatically plotted and saved.

3. Evaluate, Compare, and Save Results

Steps:

1. Compute final test accuracies for all four phases
2. Generate side-by-side accuracy comparisons
3. Plot overfitting gaps and learning curves
4. Save all models and training history
5. Export report-ready figures

The notebook contains a dedicated section for saving all outputs for reproducibility.

Appendix C: References

1. Food-11 Dataset
   Kaggle: https://www.kaggle.com/datasets/trolukovich/food11-image-dataset
2. PyTorch Documentation
   https://pytorch.org/docs/stable/
3. Torchvision Models – ShuffleNet V2
   https://pytorch.org/vision/stable/models/generated/torchvision.models.shufflenet_v2_x1_0.html
4. Data Augmentation Guidelines
   https://pytorch.org/vision/stable/transforms.html

5. Course Materials
   BMI/CSE 598: Embedded Machine Learning (Arizona State University, 2025)

Appendix D: Use of AI Tools

The development process utilized large language models to enhance efficiency and troubleshoot complex technical challenges. **ChatGPT** and **Gemini Pro** models were specifically employed for:

Large Language Models (ChatGPT and Gemini) were used to support but not replace the development process. Assistance included:

1. Debugging & Error Resolution
   - Diagnosing why early ShuffleNet training produced low accuracy
   - Identifying issues with the optimizer choice (SGD → Adam)
   - Correcting missing imports (e.g., import pickle)
   - Debugging shape mismatches in confusion matrix generation
   - Recommending improved training hyperparameters for Phase 4
2. Code Modification Support
   - Helping structure dataloaders and transforms
   - Suggesting architecture refinements for CNN phases
   - Drafting comparison plots using Matplotlib
3. Documentation & Report Writing
   - Assisting in structuring technical explanations
   - Refining architectural descriptions
   - Helping convert the Notebook's results into a formal report
   - Generating tables and figure captions

All experiments, coding decisions, and model development were performed by the student.