Delisio: A Multimodal State Machine for Personalized Recipe Generation

Peyman Khodabandehlouei

Motivation

Among the many fields that AI can transform, the domain of food is especially significant. As a fundamental aspect of human life, food is deeply connected to health, culture, and personal preferences. However, traditional recipe services fail to consider individuals' unique dietary needs, restrictions, and preferences, offering only generic solutions.
Delisio is an intelligent solution that overcomes this limitation by providing personalized recipes tailored to each user. It combines computer vision for dish recognition with natural language processing for recipe generation by analyzing user-specific details, such as diet type, nutritional goals, allergies, and available equipment.
Delisio achieves this through three core capabilities:

- **Dish Recognition:** Identifies dishes from user-uploaded images using a Convolutional Neural Network (CNN).
- **Personalized Recipe Generation:** Creates custom recipes that account for user preferences, allergies, and nutritional goals using a Large Language Model (LLM).
- **Stateful Workflow:** Seamlessly transitions between different models.

System Architecture

Delisio employs a state machine powered by LangGraph to manage the recipe generation process. The state machine consists of the following states:

- Image Input: Users upload dish images.
- Food Recognition (CNN): The dish is identified using a CNN.
- Dish Name Extraction: The recognized dish name is extracted as text.
- User Preferences Query: User dietary preferences are retrieved.
- Recipe Generation: The LLM generates a personalized recipe.
- Recipe Output: The final recipe is presented to the user.

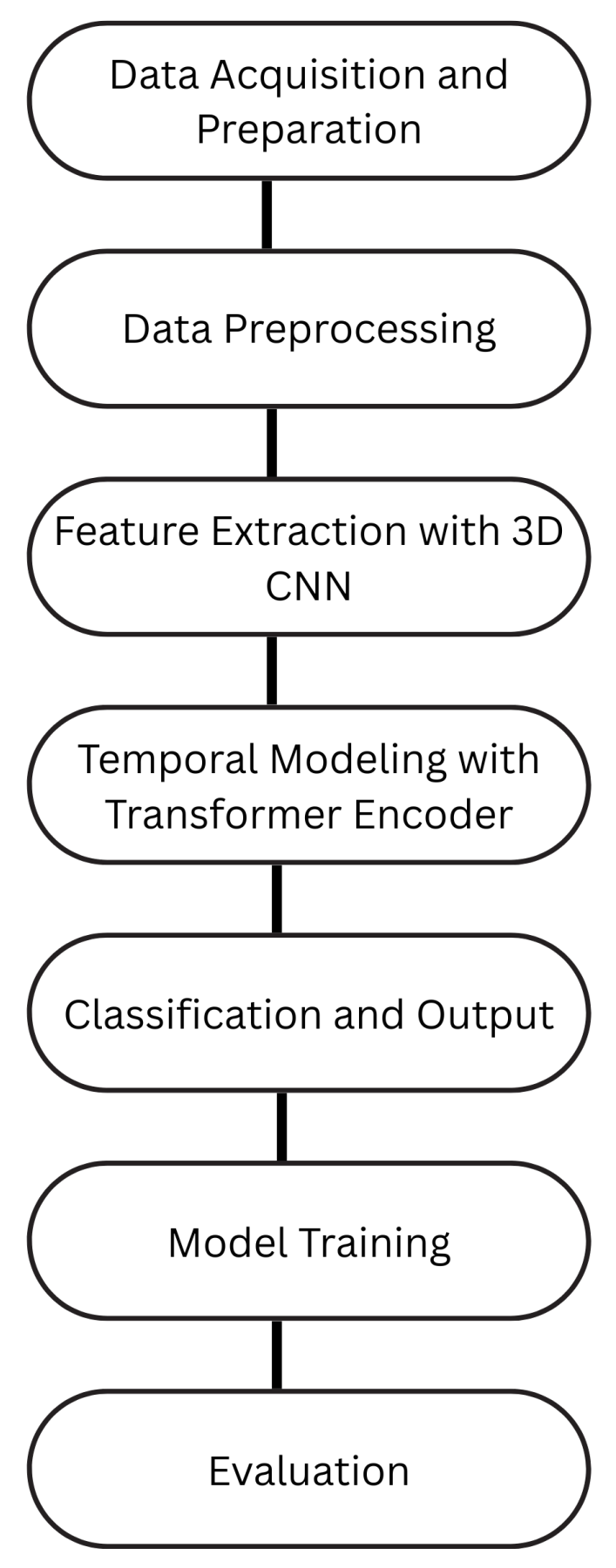
Data Acquisition and Preparation
Data Preprocessing
Feature Extraction with 3D CNN
Temporal Modeling with Transformer Encoder
Classification and Output
Model Training
Evaluation

Figure 1:State Machine Architecture

Food Classifier Model (CNN)

To train an accurate model using a small, high-quality custom dataset, transfer learning was applied. This approach allows the model to automatically extract features from images without requiring extensive training data.
Five base models were evaluated to determine the best performing architecture:

- **MobileNetV2**
- **EfficientNetB0**
- **ResNet50**
- **InceptionV3**
- **DenseNet121**

Each model was trained on the same dataset with a standard classifier following the convolutional layers. The models were trained for 5 epochs, and their validation accuracy is shown in Figure 2.

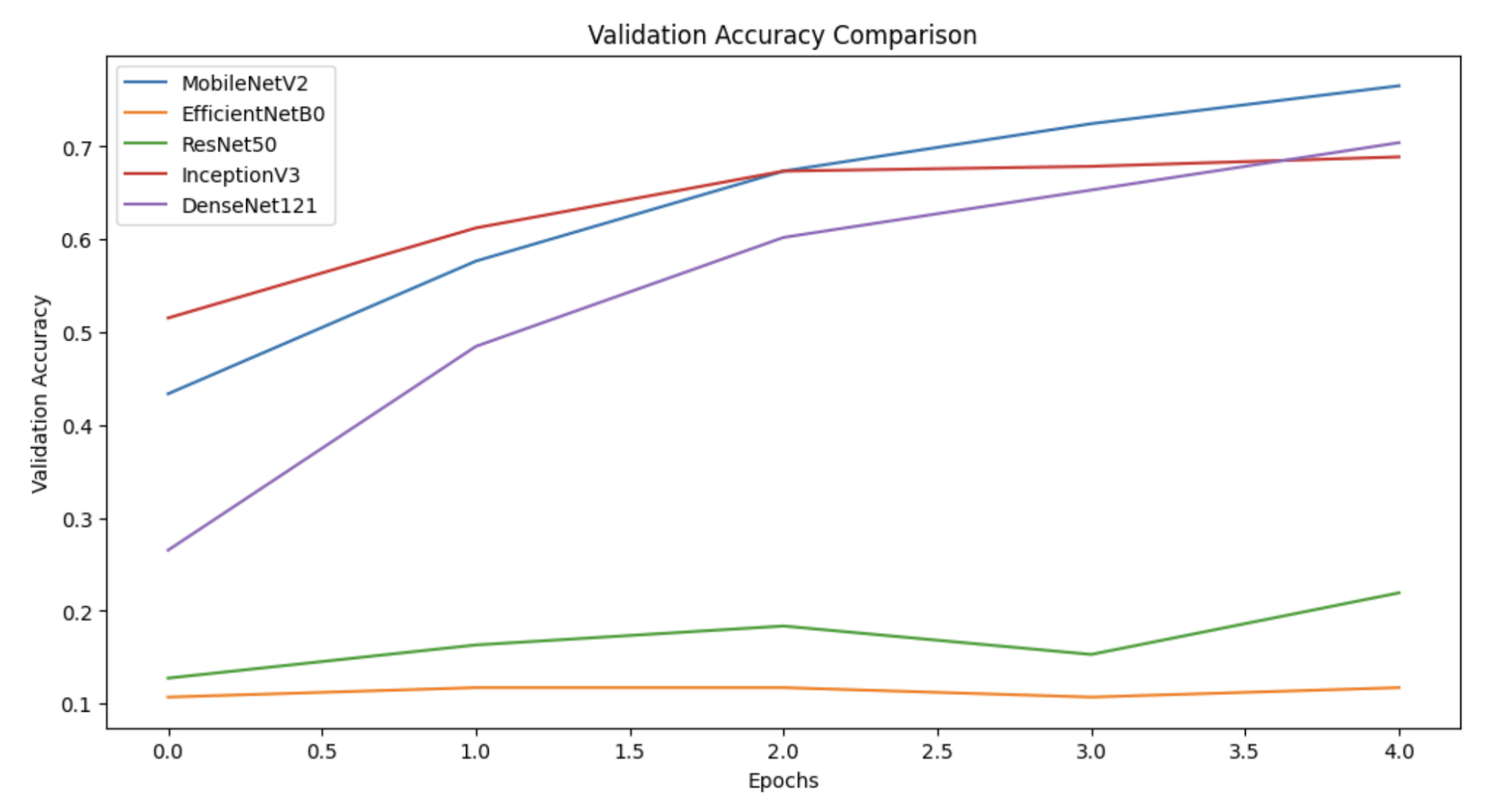

Figure 2:Validation Accuracy of Different Base Models Over 5 Epochs

As the experiment shows, the best-performing base model without epochs is **MobileNetV2** with 75% accuracy. This model was chosen as the feature extractor.
To further optimize the performance of the final model, hyperparameter tuning was performed using Keras Tuner with a Random Search strategy, systematically exploring various hyperparameters to find the best configuration. After 25 trials with 30 epochs for each trial, the best model configuration achieved a validation accuracy of **89.90%**. The optimal hyperparameters were as follows:

- **Number of Dense Layers:** 1
- **Units in Dense Layer:** 256
- **Dropout Rate:** 0.5
- **Learning Rate:** 0.000369
- **Batch Normalization:** Disabled

The Keras Tuner was configured to perform a Random Search with the following settings:

- **Search Algorithm:** Random Search.
- **Objective:** Validation Accuracy.
- **Maximum Trials:** 25.
- **Executions per Trial:** 1.
- **Training Epochs per Trial:** 30.

After identifying the optimal hyperparameters through Keras Tuner, the final model was trained using the MobileNetV2 architecture with the best configuration. The final model architecture is shown below:

- **Base Model:** MobileNetV2 (pre-trained on ImageNet).
- **Global Average Pooling:** To reduce spatial dimensions.
- **Dense Layer:** 256 units with ReLU activation.
- **Dropout:** 0.5 for regularization.
- **Output Layer:** 10 units with Softmax activation for multi-class classification.
- **Optimizer:** Adam with a learning rate of 0.000369.
- **Loss Function:** Categorical Cross-Entropy.

To monitor the training process, Early Stopping and ReduceLROnPlateau were applied. This setup ensured that the model training stopped at the optimal point and that the learning rate was reduced if the validation loss plateaued. The training and validation loss curves in Figure 3 demonstrate stable training, with both training and validation losses decreasing consistently.

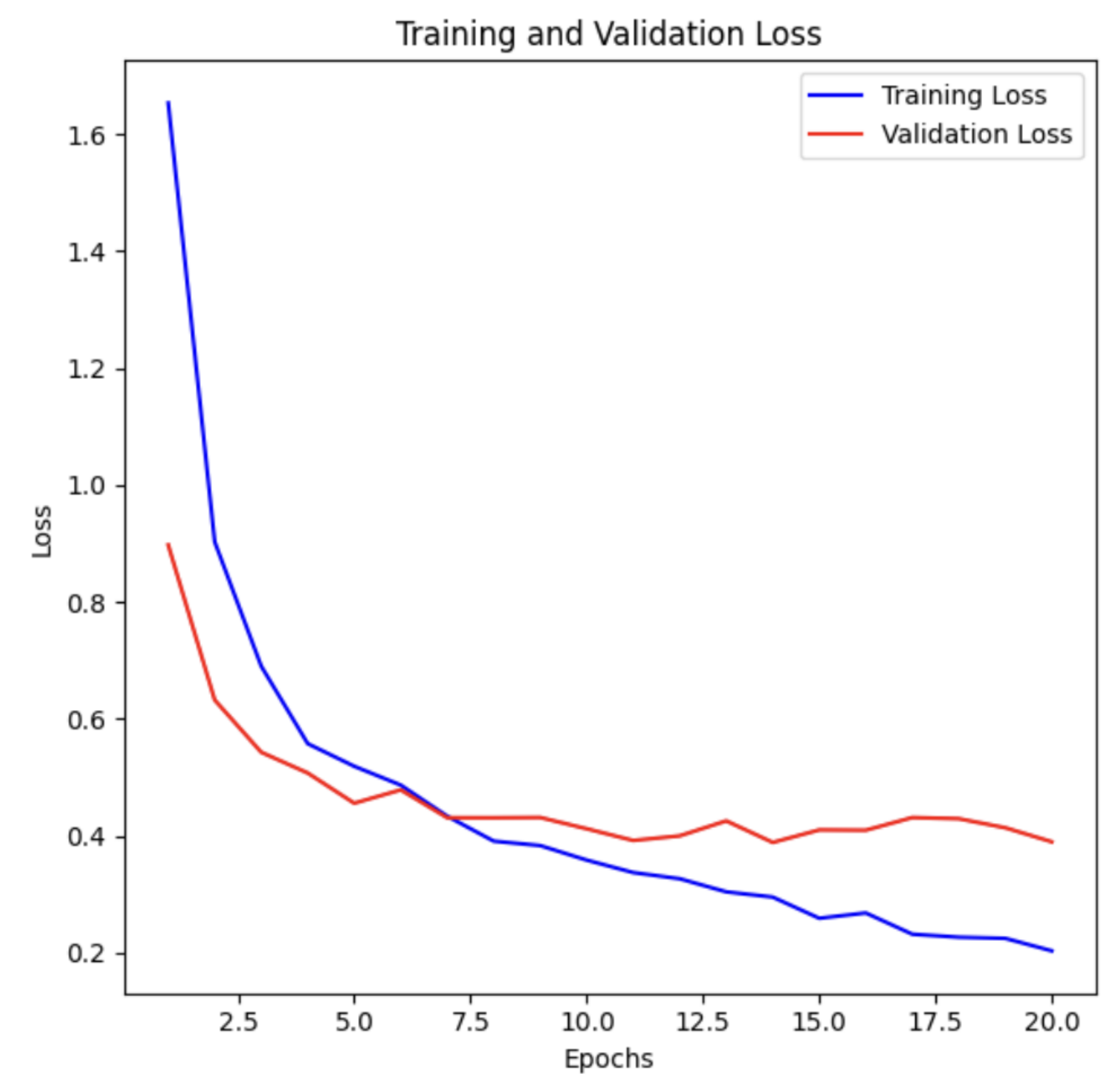

Figure 3:Figure 3: Training and Validation Loss of the Model.

The training and validation loss curves in Figure 3 show a stable learning process, with the validation loss converging close to the training loss. This indicates that the model has effectively learned without severe overfitting. The use of Early Stopping ensured that the training process was terminated at the optimal point.

Fine-Tunning

After obtaining the best frozen model, fine-tuning was performed to enhance performance. The base model (MobileNetV2) was partially unfrozen in increments of 10 layers, ranging from 10 to 100 layers. Each configuration was trained for 15 epochs with a low learning rate (1×10^{-6}) and early stopping.

- **Base Model:** MobileNetV2 (pre-trained on ImageNet).
- **Learning Rate:** 1×10^{-6}.
- **Training Epochs:** 15 per configuration.
- **Early Stopping:** Applied with patience of 5 epochs.

The highest validation accuracy (89.8%) was achieved with 70 unfrozen layers, demonstrating that selective fine-tuning significantly improved performance.

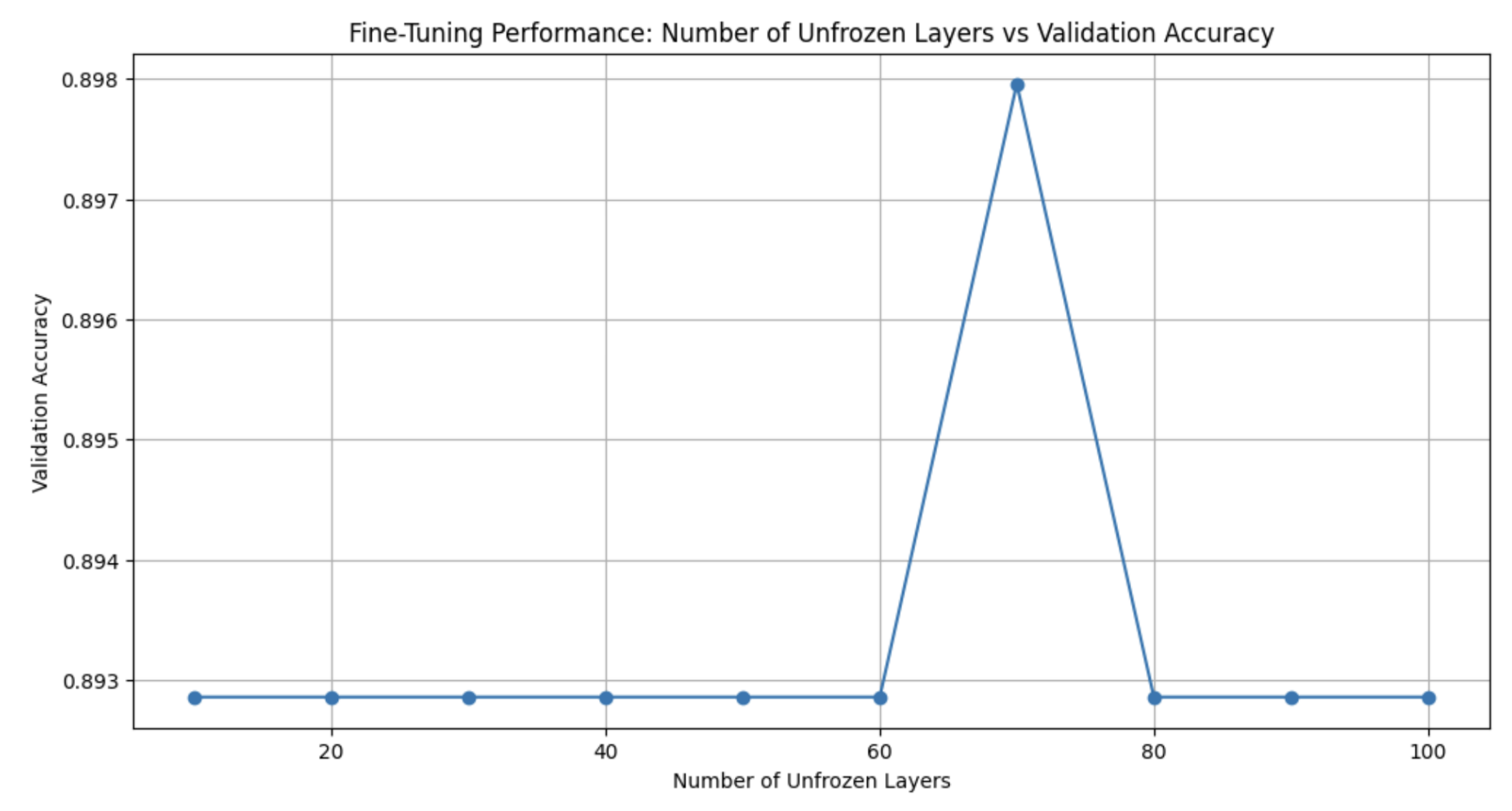

Figure 4:Figure 5: Fine-Tuning Results - Validation Accuracy by Number of Unfrozen Layers

Data

The success of this project relies heavily on high-quality data. A custom dataset of 10 food classes (Burger, Fried Chicken, Kebab, Paella, Pasta, Pizza, Ramen, Steak, Sushi, and Tacos) was collected using web scraping from reliable sources. Each image was manually reviewed to ensure quality and accurately labeled for training. The dataset was split into training (80%), validation (10%), and test (10%) sets, with all images resized to (224, 224) for model compatibility. The final dataset is available as an open-source resource.

Figure 5:Figure 6: Sample Images from the Custom Dataset