Capstone Project: Food Vision AI

Problem Statement

The ability to accurately classify food items from images has significant applications in various industries, including health monitoring, restaurant management, and e-commerce. However, achieving high accuracy in food image classification is challenging due to the diversity in food presentation, variations in lighting, and overlapping visual characteristics among certain dishes.

This project, **Food Vision AI**, aims to develop a deep learning-based solution capable of effectively classifying food images into 101 distinct categories using the Food101 dataset. The project leverages a pre-trained EfficientNetB7 model, fine-tuned to achieve optimal performance. Furthermore, the solution is deployed via a Flask application to ensure accessibility and practical usability.

Methodology

Dataset Preparation

The Food101 dataset, consisting of images with varying resolutions, was preprocessed to ensure uniformity. Each image was resized to 224x224 pixels, and the dataset was batched with a batch size of 32 for efficient processing. The dataset was split into training and testing sets, with 25% of the testing data further reserved as a validation set.

Model Architecture

The project utilized the EfficientNetB7 model, a state-of-the-art pre-trained convolutional neural network. To adapt the model for the Food101 classification task:

1. A GlobalAveragePooling layer was added before the final layer.
2. The last layer was replaced with a dense layer containing 101 neurons to match the number of classes in the dataset, using a softmax activation function.

Training and Fine-Tuning

Initially, all layers of the EfficientNetB7 model were frozen, and only the final layers were trained for 7 epochs using the Adam optimizer with a learning rate of 0.001. Subsequently, the entire model was unfrozen, and fine-tuning was performed using a reduced learning rate of 0.0001 to optimize model performance without overfitting. The loss function employed during training was sparse categorical crossentropy.

Mixed Precision Training

To enhance computational efficiency, mixed precision training was utilized, which combines 16-bit and 32-bit floating-point numbers during model operations. This approach significantly reduces memory usage, accelerates training by leveraging GPU Tensor Cores optimized for 16-bit computations, and enables larger batch sizes. Despite these efficiency gains, the model maintained accuracy comparable to traditional 32-bit training, demonstrating the effectiveness of mixed precision in handling resource constraints and large datasets like Food101.

Evaluation

The model's performance was evaluated using accuracy as the primary metric. Additionally, custom food images downloaded from the internet were used to test the model's real-world generalizability. Training progress was tracked by storing logs in a history variable, which was later visualized to analyze accuracy and loss trends across epochs.

Deployment

The trained model was deployed using a Flask-based web framework with HTML, CSS, and JavaScript to create a user-friendly interface. To optimize deployment, the model weights were saved separately, and the architecture was redefined in the deployment script. The saved weights were loaded to make predictions, minimizing loading time compared to saving and loading the entire model.

Results

Model Performance

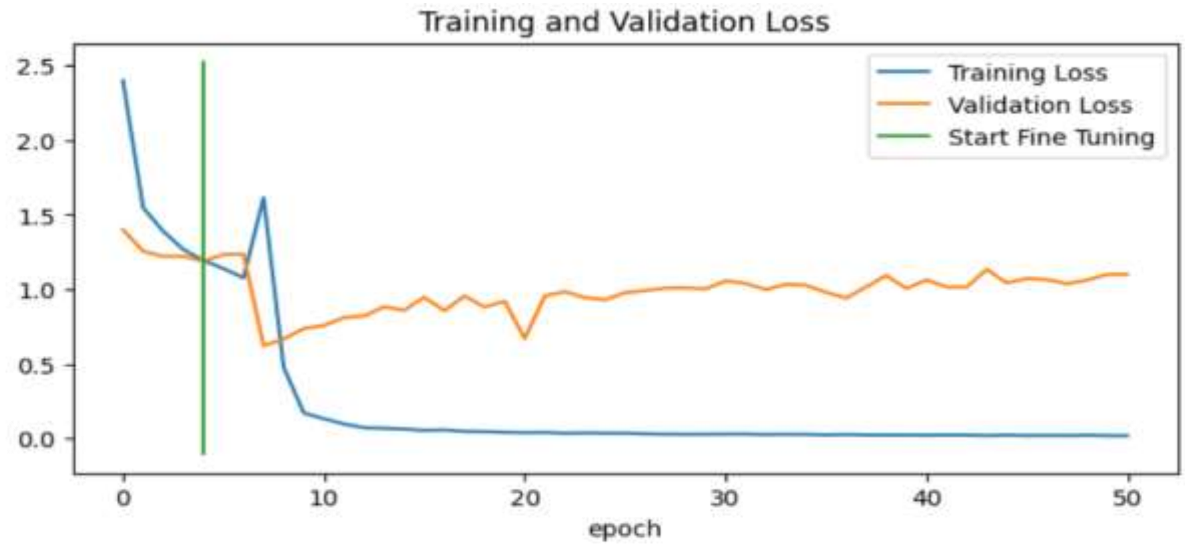
The model achieved a test accuracy of **83.35%**, demonstrating its effectiveness in classifying food images into 101 categories. Despite some overfitting observed during training (as shown in the training curves), the model performed well on custom food images downloaded from the internet, successfully predicting their labels in most cases.

Training Progress

The training and validation accuracy/loss trends over 50 epochs are shown below. Fine-tuning began after 7 epochs, as indicated by the green line in the plots. While the training loss continued to decrease, the validation loss plateaued, suggesting overfitting.

- **Training and Validation Accuracy Plot**:

- **Training and Validation Loss Plot**:



Deployment Success

The model was successfully deployed via a Flask-based web application, providing a user-friendly interface for image upload and prediction. Screenshots of the interface are attached to demonstrate its functionality.

Comparison with Benchmarks

The model outperformed benchmarks like the **DeepFood paper (80%)**, achieving higher accuracy on the Food101 dataset. This highlights the effectiveness of leveraging a pre-trained EfficientNetB7 model and fine-tuning it for this task.

Limitations

Some limitations were observed in the model's predictions:

- It struggled with certain categories such as **apple pie**, **ravioli**, and **pork chop**, likely due to visual similarities with other classes or insufficient distinguishing features in the dataset.

Challenges Encountered

Efforts to address overfitting were constrained by the following factors:

- **Time Constraints**: Each epoch required approximately 22 minutes, and fine-tuning took nearly a week to complete.
- **Resource Limitations**: Training was conducted on Colab, with limited GPU access.
- **Unpredictable Interruptions**: WiFi and electricity outages disrupted training sessions.
- **Project Deadline**: The project deadline left insufficient time for additional training experiments.

Despite these challenges, the model met the project's practical objectives by performing well on custom data.

Future Work

1. **Addressing Overfitting**
   Future efforts will focus on addressing overfitting to improve the model's generalization capabilities. Techniques such as data augmentation, dropout, L2 regularization, and early stopping can be explored to prevent the model from overfitting to the training data.
2. **Enhancing Model Accuracy**
   Further fine-tuning and experimentation with hyperparameters, optimizers, and learning rate schedules can be conducted to improve the model's accuracy. Exploring ensemble methods or using larger, more diverse datasets may also enhance performance.
3. **Cultural Diversity in Food Classification**
   Expanding the model's scope to accurately identify foods from different cultural cuisines—such as Chinese, Japanese, Indian, Pakistani, and other regional dishes—will be a priority. This could involve curating or incorporating culturally diverse datasets to improve the model's inclusivity and robustness.

4. **Explainable AI for Predictions**
   Implementing explainable AI techniques, such as Grad-CAM or SHAP visualizations, can provide insights into why the model makes specific predictions. This could help improve trust and usability, especially in commercial or educational settings.

5. **Real-Time Prediction**
   Optimizing the model for real-time predictions on edge devices or low-resource environments could make the application more accessible and efficient. Techniques like model quantization, pruning, or conversion to formats like TensorFlow Lite can be explored.

6. **Broader Deployment Options**
   Extending the deployment to platforms like mobile apps or integrating with voice assistants could enhance accessibility. This would allow users to classify food images directly from their devices without requiring a web interface.

7. **Incorporating Nutritional Information**
   Enhancing the application by linking classified food items to their nutritional information can add value for health-conscious users, fitness applications, or diet planners.

8. **Improving Class-Specific Performance**
   Specific attention can be given to underperforming classes like apple pie, ravioli, and pork chop. This may involve collecting more representative data for these categories or refining the model's ability to distinguish similar food items.

9. **Multi-Label Classification**
   Developing the model to handle multi-label classification, where a single image can contain multiple food items, can expand its practical applications.

10. **Exploring Transfer Learning with Other Architectures**
    Experimenting with other advanced architectures, such as Vision Transformers (ViTs) or ConvNeXt, could provide insights into alternative approaches for improving performance on the Food101 dataset.