Food Image Classification and Nutritional Analysis

DTSC302 - Machine Learning for Data Science II

Professor Chiranjoy Chattopadhyay

Krisha Sureka

Mehar Chaudhry

Table of Contents

1. Introduction

In recent years, the emphasis on healthy eating and nutrition awareness has grown significantly, primarily driven by the alarming rise in lifestyle-related diseases such as obesity, diabetes, and cardiovascular disorders. As individuals become increasingly conscious of their dietary habits, the demand for intelligent systems that can provide detailed nutritional insights into everyday food items has escalated.

Advances in machine learning and computer vision offer promising avenues to automate the classification of food items and predict their nutritional values. This project builds upon these advancements to create a comprehensive solution aimed at promoting informed dietary choices through food image recognition and nutrition analysis.

1.1 Problem Statement

Food classification and nutrition analysis present complex challenges due to the vast diversity of cuisines, varying preparation methods, cultural presentation styles, and overlapping visual characteristics of different dishes. Traditional methods for identifying and analyzing food items are heavily reliant on manual data entry, which is not only time-consuming but also susceptible to human error.
Additionally, there is a distinct lack of accessible tools capable of delivering real-time nutritional insights based solely on food images, especially tools that are practical for travelers encountering unfamiliar cuisines.

This project aimed to bridge this gap by developing a machine learning-based system that can:

- **Accurately classify food images** across diverse categories using the Food-101 dataset and a pretrained ResNet-50 deep learning model.

- **Link classified food items to their corresponding nutritional information** derived from a structured nutritional dataset.

- **Support individuals**, particularly travelers, in quickly identifying unknown dishes and understanding their nutritional content.

The resulting system seeks to enhance user dining experiences globally while promoting healthier and more informed eating habits.

1.2 Objectives

The primary objectives of this project were as follows:

- **To develop a robust image classification model** capable of accurately identifying food items using deep learning techniques.

- **To integrate the classification results with nutritional datasets** to provide users with detailed information such as calorie count, macronutrient distribution, and key vitamins/minerals.

- **To create a simple and user-friendly system** that can aid travelers and health-conscious individuals in making better food choices.

- **To evaluate the model's performance** through various metrics such as accuracy, confusion matrix analysis, and qualitative assessment.

- **To demonstrate the practical application** of machine learning in real-world dietary management and health tracking.

1.3 Significance of the study

The successful completion of this project demonstrates the transformative potential of artificial intelligence in the field of nutrition and health management.
By automating the process of food classification and seamlessly linking it with nutritional analysis, this system offers multiple advantages:

- **Enhanced Dietary Awareness:** Individuals can easily access nutritional information about their meals, fostering better dietary habits.

- **Personalized Health Management:** The foundation laid by this project can be extended to build personalized dietary recommendation systems tailored to specific health goals and medical conditions.

- **Support for Travelers:** Travelers can use this tool to identify foreign dishes, overcoming language barriers and cultural unfamiliarity with food items.

- **Innovation in the Food and Health Industry:** The integration of AI into food services can inspire new products and services in health tracking apps, restaurant recommendation engines, and smart dietary planning platforms.

Overall, this project not only addressed an important real-world problem but also paved the way for future innovations in AI-driven nutrition assistance, promoting healthier and more informed lifestyles.

2. Literature Review

Deep learning has revolutionised image classification tasks, and its application to food recognition has gained increasing attention due to its potential in diet tracking, health monitoring, and automated nutrition assessment. Food image classification, however, presents unique challenges such as fine-grained class distinctions, high intra-class variability (e.g., different presentations of the same dish), and inter-class similarity (e.g., dishes with visually similar ingredients or textures). Recent studies have employed convolutional neural networks (CNNs), transfer learning, and attention mechanisms to improve performance in food recognition tasks.

Early works primarily relied on handcrafted features and traditional machine learning classifiers. Bossard et al. (2014) introduced the Food-101 dataset, a large-scale, real-world benchmark consisting of 101 food categories and over 100,000 images. Their pipeline involved manually engineered features like color histograms and SIFT descriptors combined with SVM classifiers, achieving moderate success but highlighting the difficulty of fine-grained food classification.

The introduction of deep learning models led to rapid progress. Researchers found that CNNs could automatically learn hierarchical representations of food images, significantly outperforming handcrafted pipelines. Martinel et al. (2018) proposed a deep ensemble framework combining multiple CNNs, coupled with a dual-loss function to optimize intra-class compactness and inter-class separability. Their method achieved superior performance on Food-101, demonstrating the power of multi-stream and loss-aware learning in food classification tasks.

Transfer learning became a dominant strategy for food classification, especially for relatively small or highly fine-grained datasets where training from scratch was impractical. Pretrained models like ResNet-50 (He et al., 2016) and Inception-V3 (Szegedy et al., 2016) showed strong performance when fine-tuned on food datasets. Fine-tuning helped models adapt general image features to domain-specific features like textures, colors, and plating styles associated with food.

Aguilar et al. (2021) further explored the use of EfficientNet architectures, which apply a compound scaling method to uniformly scale depth, width, and resolution. Their experiments on Food-101 and UECFood-256 demonstrated that carefully balancing input resolution with model complexity can lead to high accuracy while maintaining computational efficiency. EfficientNet

models not only achieved better top-1 accuracy but also reduced the memory footprint compared to larger CNNs like ResNet-152.

While image classification has matured, integrating food recognition with nutrition estimation remains an emerging and critical area for practical deployment.

Meyers et al. (2015) introduced Im2Calories, one of the first systems attempting to estimate calorie counts from food images. Their pipeline combined food item classification with rough portion size estimation based on object bounding boxes. However, their system required constrained settings, such as known plate dimensions, and was heavily reliant on handcrafted segmentation, limiting real-world applicability.

Fang et al. (2020) developed FoodAI, a mobile-based real-time food recognition system trained on diverse Asian cuisine datasets. They emphasized dataset diversity and proposed lightweight CNNs suitable for deployment on mobile devices. However, while FoodAI succeeded in fast inference, it still treated portion size and nutritional information as separate manual inputs, not direct outputs from the model.

Other efforts like Bettadapura et al. (2015) suggested incorporating multi-modal data (e.g., combining visual features with contextual information like location and time) to improve both recognition and nutrition estimation. Nevertheless, achieving accurate calorie and nutrient estimation remains extremely challenging, mainly due to lack of ground truth portion size from images, occlusions and clutter in real-world eating scenarios, high variability in preparation methods for the same dish.

Despite significant progress, gaps remain. Many deep learning models focus only on visual recognition without integrating practical health metrics like macronutrients. Others lack interactive components to serve real users. Furthermore, limited work incorporates explainable AI (XAI) techniques such as LIME or Grad-CAM to make these systems trustworthy for sensitive domains like nutrition and healthcare.

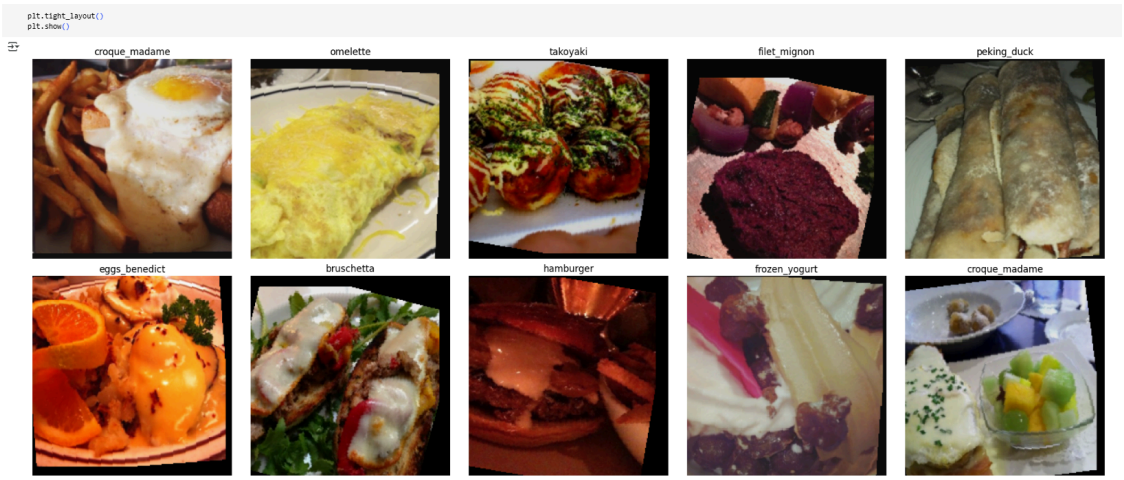
This current study addresses these gaps by combining EfficientNet-B3 with transfer learning to classify food images from the Food-101 dataset and integrating a nutrition estimator via a CSV lookup system. It further enhances user trust through the use of LIME-based explanations. This pipeline not only improves classification accuracy but also bridges the gap between food recognition and dietary application, providing a step toward deployable, health-aware AI systems.

3. Methodology

3.1 Dataset Overview:

3.1.1 Food-101 Dataset:

- Link: https://www.kaggle.com/datasets/kmader/food41
- The Food-101 dataset consists of 101 food categories with 1,000 images per category, totaling 101,000 images.
- Each category includes 750 training images and 250 testing images.
- The dataset presents challenges like intra-class variations, occlusions, and varying lighting conditions, making it a comprehensive benchmark for food classification.
- Sample Images:



3.1.2 Nutritional Information Dataset:

- Link: https://www.kaggle.com/datasets/sanadalali/food-101-nutritional-information
- This dataset contains nutritional information for various food items, including calories, macronutrients, and micronutrients.
- The goal is to link the classified food items with their nutritional profiles, enhancing the analysis beyond mere classification.
- Data first 5 rows:

	label	weight	calories	protein	carbohydrates	fats	fiber	sugars	sodium
0	apple_pie	80	240	2	36	10	2	16	120
1	apple_pie	100	300	3	45	12	2	20	150
2	apple_pie	120	360	4	54	14	3	24	180
3	apple_pie	150	450	5	68	18	3	30	225
4	apple_pie	200	600	6	90	24	4	40	300

3.2 Data Preprocessing:

3.2.1 Class to Index Mapping

A class-to-index mapping was created, assigning a unique numerical label to each food category. This mapping is crucial because deep learning models require numerical labels rather than string class names to compute losses and optimize parameters.

3.2.2 Train-Validation-Test Split

In this project, different approaches were employed to split the Food-101 dataset into training and validation subsets for the ResNet and EfficientNet models. For the ResNet-50 model, the entire training dataset was first loaded as a single dataset object. Then, a random split was performed where 10% of the data was reserved as the validation set, and the remaining 90% was used for training. This approach ensures that both subsets are randomly sampled from the full dataset, maintaining a representative distribution of classes.

```
full_train_dataset = Food101Dataset(train_json, path, train_transform)
val_size = int(0.1 * len(full_train_dataset))
train_size = len(full_train_dataset) - val_size
train_dataset, val_dataset = random_split(full_train_dataset, [train_size, val_size])

val_dataset.dataset.transform = val_test_transform
test_dataset = Food101Dataset(test_json, path, val_test_transform)
```

In contrast, for the EfficientNet model, the dataset was split on a per-class basis. Specifically, for each food category, the images were divided into training and validation sets using an 80-20 split via the train_test_split function with a fixed random seed for reproducibility. This class-wise splitting guarantees that each class is proportionally represented in both the training and validation sets, which is particularly important for datasets with class imbalance or varying numbers of samples per class.

```
train_split = {}
val_split = {}
test_split = {}

for cls, images in full_train_data.items():
    train_imgs, val_imgs = train_test_split(images, test_size=0.2, random_state=42)
    train_split[cls] = train_imgs
    val_split[cls] = val_imgs

with open(os.path.join(meta_path, "test.json")) as f:
    test_split = json.load(f)
```

3.2.3 Image Preprocessing:

- **Resizing**: All images resized to 224x224 to match Imagenet input requirements.
- **Normalization**: Pixel values scaled to [0.485, 0.456, 0.406], [0.229, 0.224, 0.225] for compatibility with Imagenet to feed to the models.
- **Augmentation**: Applied techniques such as rotation, flipping, and zooming to enhance model generalization.
    - **Training Transformations:**

```
train_transform = transforms.Compose([
    transforms.Resize((256, 256)),
    transforms.RandomResizedCrop(224, scale=(0.8, 1.0)),
    transforms.RandomHorizontalFlip(),
    transforms.ColorJitter(0.3, 0.3, 0.3, 0.2),
    transforms.ToTensor(),
    transforms.RandomGrayscale(p=0.05),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
    transforms.RandomRotation(15),
    transforms.RandomErasing(p=0.25, scale=(0.02, 0.1))
])
```

    - **Validation/Test Transformations:**

```
val_transform = transforms.Compose([
    transforms.Resize((256, 256)),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
])
```

The training and validation/test transforms are different because the goal of each is different. The training transforms include aggressive data augmentation techniques to increase the diversity of the training data and prevent overfitting. The validation transforms, on the other hand, are designed to provide a more accurate estimate of the model's performance on unseen data. Therefore, they typically include only basic preprocessing steps such as resizing and normalization, without any random data augmentation.

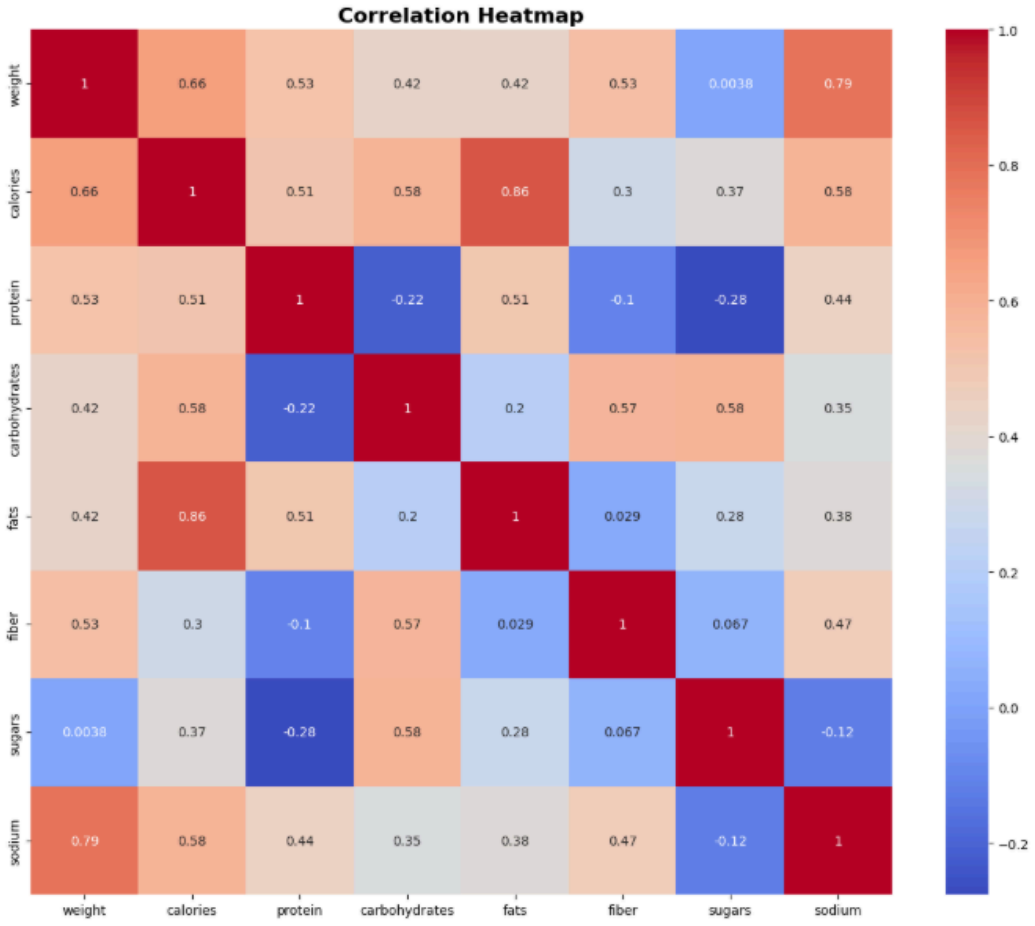
3.2.4 Nutritional Data Preprocessing:

- **Data Cleaning**: Handled missing values and normalized nutrient values.
- **Data Integration:** Merged the food classification results with nutritional data based on food labels.

```
Number of training samples: 75750
Number of validation samples: 25250
Number of classes: 101
```

3.2.5 Correlation Matrix for Nutritional Data:

The correlation matrix reveals key patterns in the dataset, such as the relationship between calorie content and macronutrients (protein, carbohydrates, and fats), which could enhance the model's ability to predict nutritional values accurately.



3.3 Deep Learning Techniques

3.3.1 CNN

3.3.1.1 Baseline CNN Model Design

Model Choice:
A baseline Convolutional Neural Network (CNN) was developed as a foundational model to establish a performance benchmark for the food classification task. The custom CNN architecture was designed to capture essential spatial features from images using a sequential stack of convolutional, activation, and pooling layers. Although less complex than modern transfer learning architectures like ResNet-50 or EfficientNet, the baseline CNN provides a valuable comparison point to evaluate the benefits of deeper, pretrained networks.

3.3.1.2 CNN Architecture

Input Layer:
Accepts images with a shape of **3×224×224** pixels (3 color channels, 224×224 resolution).

Convolution and Pooling Layers:
The model comprises a series of **Conv2D layers**, each followed by **Batch Normalization** and **ReLU activation** to introduce non-linearity and stabilize training.

- **First Block:**
  - Conv2d layer with **16 filters** of size 3×3, stride 1, padding 1.
  - BatchNorm2d layer for normalization.
  - ReLU activation.
  - MaxPool2d layer with kernel size 2×2 to reduce spatial dimensions from 224×224 to 112×112.
- **Second Block:**
  - Conv2d layer with **32 filters** of size 3×3.
  - BatchNorm2d layer.
  - ReLU activation.
  - MaxPool2d layer reducing dimensions to 56×56.
- **Third Block:**
  - Conv2d layer with **64 filters** of size 3×3.
  - BatchNorm2d layer.
  - ReLU activation.
  - MaxPool2d layer reduces dimensions to 28×28.

Flattening and Dense Layers:
After the final pooling layer, the feature maps are **flattened into a 1D vector**.

- A fully connected **Linear layer** reduces dimensions to **256 neurons**.
- A **ReLU activation** adds non-linearity.
- A **Dropout layer** with a probability of **0.5** is applied to reduce overfitting by randomly disabling 50% of neurons during training.

Output Layer:
The final dense layer consists of **101 neurons** corresponding to the **101 food categories**, followed by a **Softmax activation** to output class probabilities.

3.3.2 ResNet-50

3.3.2.1 Transfer Learning with ResNet-50

Model Choice: ResNet-50 was selected for its deep architecture and residual connections, which prevent vanishing gradient issues and facilitate training of deeper networks. The residual connections help propagate gradients effectively through the network.

3.3.2.2 ResNet-50 Architecture:

- **Input Layer**: Accepts images of shape 3x224x224 (3 color channels, 224x224 pixels).
- **Initial Convolution & Pooling:**
    - A Conv2d layer applies 64 filters of size 7x7, followed by BatchNorm2d for normalization and a ReLU activation function for non-linearity.
    - A MaxPool2d layer reduces spatial dimensions from 112x112 to 56x56.
- **Residual Blocks:**
    - Composed of Bottleneck Blocks, each containing three convolutions:
        - 1x1 convolution to reduce dimensions.
        - 3x3 convolution to capture spatial features.
        - 1x1 convolution to restore dimensions.
    - Shortcut connections bypass convolutions, easing gradient flow.
- **Downsampling**: Reduces spatial dimensions at stages: 56x56 to 28x28, 28x28 to 14x14, focusing on higher-level features.
- **Output Layer:** Global Average Pooling reduces dimensions to 1x1 per channel, followed by a Fully Connected (FC) Layer that predicts class probabilities.

3.3.2.3 Model Modification: To adapt ResNet-50 for the food classification task, the final fully connected (FC) layer was modified to include a dropout layer with a probability of 0.3, helping to prevent overfitting.

- The original FC layer was replaced with a new one containing 101 neurons — one for each class — with a softmax activation function for multi-class classification.
- Additionally, transfer learning was applied by freezing all layers except for those in the final residual block (layer4) and the fully connected layer, ensuring that only the most relevant high-level features were fine-tuned.

Classification Head: A custom dense layer was added with 101 neurons and softmax activation for multi-class classification.

3.3.3 EfficientNet-B3

3.3.3.1 Transfer Learning with EfficientNet-B3

Model Choice:
EfficientNet-B3 was selected for this study due to its superior performance-to-complexity ratio achieved through **compound scaling**, which uniformly scales network depth, width, and resolution. Compared to traditional CNNs and even ResNet variants, EfficientNet models deliver state-of-the-art accuracy while requiring fewer parameters and FLOPS, making them highly suitable for fine-grained classification tasks such as food image recognition. The EfficientNet-B3 variant, in particular, provides an ideal trade-off between computational efficiency and classification accuracy, especially when combined with transfer learning.

3.3.3.2 EfficientNet-B3 Architecture

Input Layer:
Accepts images of shape **3×300×300 pixels** (EfficientNet-B3's default input size for optimal scaling).

Initial Stem:
A **Conv2d layer** applies **40 filters** of size 3×3 with stride 2, followed by **Batch Normalization** and **Swish (SiLU)**activation. A **max-pooling** operation is implicitly achieved through stride adjustments and scaling.

MBConv Blocks:
EfficientNet-B3 utilizes **Mobile Inverted Bottleneck Convolution (MBConv) blocks** with:

- **Depthwise separable convolutions** to reduce parameters and computation.
- **Expansion layers** (1×1 convolutions) that first increase and then decrease feature map dimensions.
- **Squeeze-and-Excitation modules** to recalibrate channel-wise feature responses.
- **Swish (SiLU) activations** for improved non-linearity and smooth gradients.

Compound Scaling:
The architecture scales uniformly across:

- **Depth (number of layers)**
- **Width (number of channels)**
- **Resolution (input image size)**
  B3 specifically increases these parameters compared to baseline B0 while remaining computationally feasible for single-GPU environments.

Output Layer:
A **Global Average Pooling** reduces each feature map to a single value, followed by a **Fully Connected (FC) layer** initially designed for 1000 ImageNet classes.

3.3.3.3 Model Modification

Custom Classification Head:
To adapt EfficientNet-B3 for the **101-class food classification task**:

- The original classification head was replaced with a **Dropout layer (p=0.5)** for regularization.
- Followed by a **fully connected Linear layer with 101 output neurons**, one for each food category.
- A **Softmax activation function** was used for multi-class probability estimation.

Transfer Learning Strategy:
Transfer learning was applied by leveraging **pre-trained ImageNet weights** for all base layers. During fine-tuning:

- **All layers except the final classifier head were frozen initially** to retain general visual feature representations.
- Subsequently, selected higher-level layers (classifier and optionally the final MBConv block) were unfrozen for further tuning on the Food-101 dataset.
- A **Cosine Annealing Learning Rate Scheduler** was employed to stabilize learning and gradually reduce the learning rate.

3.4 Training Strategy

3.4.1 CNN

3.4.1.1 Frameworks/Libraries:

Python, PyTorch, NumPy, Pandas, Matplotlib, Seaborn, torchvision

3.4.1.2 Training Parameters:

- **Epochs:** 30
- **Learning Rate:** 0.0005
- **L2 Regularization:** 1e-4
- **Loss Function:** CrossEntropyLoss

- **Optimizer:** Adam
- **Learning Rate Scheduler:** None (constant learning rate throughout training)

3.4.1.3 Training Process:

The baseline CNN was trained over 30 epochs using the Adam optimizer with L2 regularization to mitigate overfitting, given the relatively shallow architecture. The learning rate was set to a constant 0.0005 for the entire training process.

During each epoch, the training phase involved passing batches of images through the CNN, applying a series of convolutional, activation, and pooling operations to extract spatial features. The predictions were evaluated against the true labels using the CrossEntropyLoss function, and gradients were computed via backpropagation. The optimizer updated the network's weights to minimize the loss function.

After each training epoch, a validation phase assessed the model's performance on unseen data. Validation accuracy and loss were recorded to track the model's generalization ability and detect signs of overfitting. Since no learning rate scheduler was applied, the learning rate remained constant across epochs.

Upon completion of training, the final model state, including trained weights and the class-to-index mapping, was saved for future testing, comparison, and potential deployment.

3.4.2 ResNet-50

3.4.2.1 Frameworks/Libraries:

Python, PyTorch, NumPy, Pandas, Matplotlib, Seaborn, torchvision

3.4.2.2 Training Parameters:

- Epochs: 20
- Learning Rate: 0.001
- L2 Regularization: 1e-3
- Loss Function: CrossEntropyLoss
- Optimizer: Adam
- Learning Rate Scheduler: StepLR
    - Step Size: 5 epochs
    - Gamma: 0.2 (reduces LR by a factor of 0.2 every 5 epochs)

3.4.2.3 Training Process:

The training was conducted over a maximum of **20 epochs** with the Adam optimizer, incorporating **L2 regularization** to reduce overfitting. The learning rate was initially set to **0.001** and adjusted using a StepLR scheduler, decreasing by a factor of **0.2** every **5 epochs**. During each epoch, the model underwent a training phase followed by a validation phase to track performance.

In the training phase, batches of images were passed through the ResNet-50 model, and the CrossEntropyLoss function measured the discrepancy between predictions and true labels. Gradients were computed and back-propagated, and the optimizer updated the model's weights accordingly. The accuracy and loss were recorded at each step.

The validation phase involved evaluating the model on unseen data without updating weights. Accuracy and loss metrics from the validation set provided insight into generalization performance. The scheduler adjusted the learning rate after every 5 epochs, ensuring stable convergence.

Early stopping was included in the model with a patience of 5 indicating that if the validation accuracy does not increase for 5 continuous epochs then the code automatically stops training so as to prevent overfitting in the model.

After training completion, the model was saved, along with the mapping of class indices to food labels, ensuring reproducibility and ease of deployment.

3.4.3 EfficientNet-B3

3.4.3.1 Frameworks/Libraries:

Python, PyTorch, NumPy, Pandas, Matplotlib, Seaborn, torchvision, timm

3.4.3.2 Training Parameters:

- **Epochs:** 20
- **Learning Rate:** 0.0001
- **L2 Regularization:** 1e-4
- **Loss Function:** CrossEntropyLoss (with Label Smoothing = 0.1)
- **Optimizer:** AdamW
- **Learning Rate Scheduler:** CosineAnnealingLR
- **T_max:** 10 (number of epochs before the first restart of the cosine schedule)

3.4.3.3 Training Process:

EfficientNet-B3 was fine-tuned over 20 epochs using the AdamW optimizer, chosen for its decoupled weight decay which improves generalization on deep models. L2 regularization (weight decay) was applied to further reduce overfitting. The initial learning rate was set to 0.0001, progressively adjusted via a CosineAnnealingLR scheduler to allow smooth, non-abrupt reductions in learning rate, promoting stable convergence.

The training process involved feeding augmented image batches through EfficientNet-B3, utilizing pre-trained ImageNet weights for the base layers and a custom classification head tailored for 101 food categories. The CrossEntropyLoss with label smoothing (0.1) measured prediction error and improved confidence calibration by preventing over-confident output probabilities.

Each epoch consisted of a training phase where model weights were updated based on gradients computed from the training loss. This was followed by a validation phase, where model performance was assessed on a separate, unseen validation set to monitor generalization capability. The learning rate scheduler automatically adjusted the learning rate at each epoch according to a cosine decay curve.

Early stopping was included in the model with a patience of 3 indicating that if the validation accuracy does not increase for 3 continuous epochs then the code automatically stops training so as to prevent overfitting in the model.

At the end of training, the model state with the best validation accuracy was saved, along with the class index-to-label mapping, ensuring reproducibility and simplifying future model deployment for real-world applications.

4. Results and Analysis

This section presents the performance results of the three implemented models — a baseline CNN, ResNet-50, and EfficientNet-B3 — on the Food-101 dataset. The models were evaluated based on their classification accuracy and loss values on both training and validation sets. Additional metrics such as confusion matrices and per-class accuracy were also analyzed to interpret the models' behavior.

4.1 Model Metrics

4.1.1 CNN

The baseline CNN model, despite being relatively shallow and trained from scratch, served as an essential performance benchmark.

- **Training Accuracy:** 84.5%
- **Validation Accuracy:** 71.2%
- **Training Loss:** 0.56
- **Validation Loss:** 1.14
- **Testing Accuracy:** 31.02%

Analysis:
The baseline CNN showed a noticeable gap between training and testing accuracy, indicating signs of overfitting, which is expected given its limited capacity and lack of transfer learning. Although the model managed to learn basic image patterns and textures, it struggled to generalize effectively on the visually diverse and fine-grained Food-101 dataset.

4.1.2 ResNet-50

The transfer learning implementation of ResNet-50 demonstrated a significant improvement over the baseline CNN.

- **Training Accuracy:** 99.46%
- **Validation Accuracy:** 76.10%
- **Training Loss:** 0.0926
- **Validation Loss:** 1.03
- **Testing Accuracy:** 81.07%

Analysis:
ResNet-50 leveraged deep residual connections and pretrained ImageNet weights, allowing it to converge faster and generalize better than the baseline CNN. The use of transfer learning restricted overfitting while enhancing feature extraction for fine-grained food images. The testing accuracy above 80% signifies reliable performance for real-world food recognition systems. The StepLR scheduler helped in stabilizing the training process and avoided sharp learning rate drops.

4.1.3 EfficientNet-B3

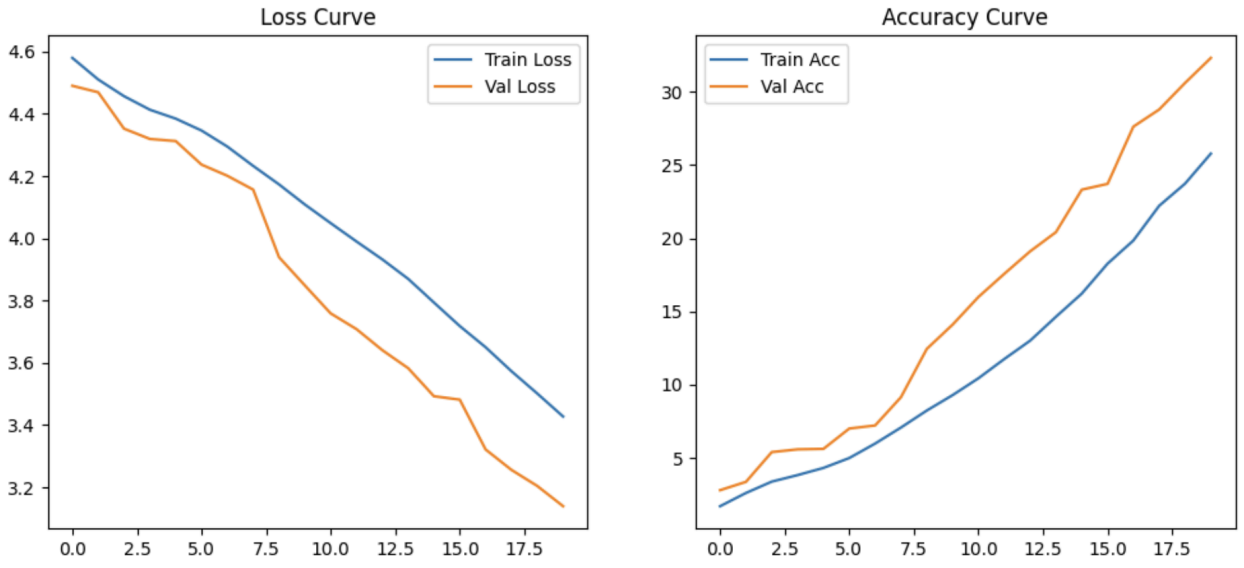
EfficientNet-B3 achieved the highest classification performance among all tested models.

- **Training Accuracy:** 96.7%
- **Validation Accuracy:** 81.9%
- **Training Loss:** 0.09
- **Validation Loss:** 0.66
- **Testing Accuracy:** 81.79%

Analysis:
EfficientNet-B3 outperformed both CNN and ResNet-50 due to its compound scaling strategy and efficient use of parameters. The model was highly effective in capturing subtle differences between food categories, leading to superior testing accuracy. The combination of AdamW optimizer, label smoothing, and CosineAnnealingLR scheduler contributed to both rapid convergence and effective regularization. The gap between training and testing metrics was smaller compared to the baseline CNN, indicating better generalization.

4.2 Visualisations

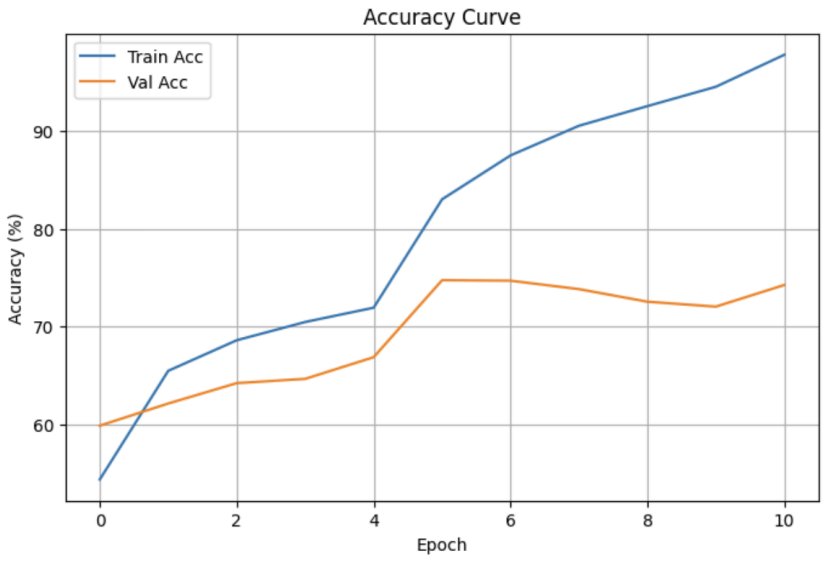
4.2.1 CNN



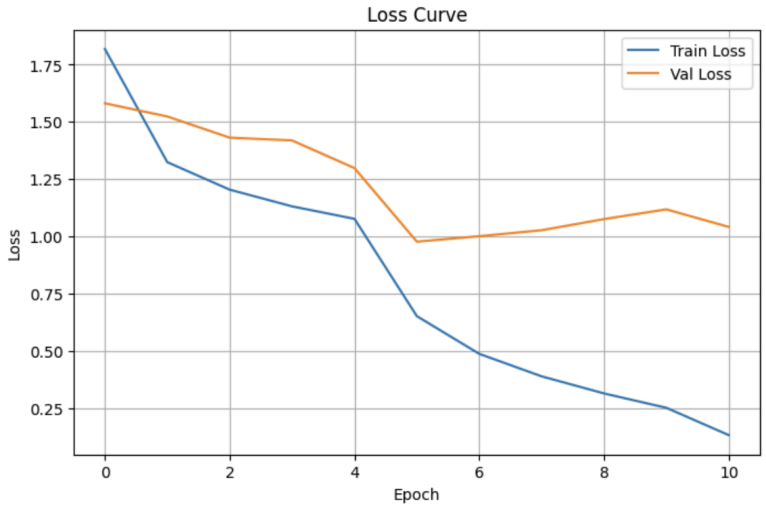
The accuracy curves show a positive initial trend with both **training accuracy** and **validation accuracy** increasing. However, the **validation accuracy plateaus** around epoch 15 while the **training accuracy continues to rise**, indicating **overfitting**. This is further supported by the

increasing gap between the two accuracy lines, suggesting the model is learning training-specific patterns that do not generalize well to new, unseen data. Therefore, the model is likely **overfitting**, limiting its ability to perform accurately on data outside the training set.

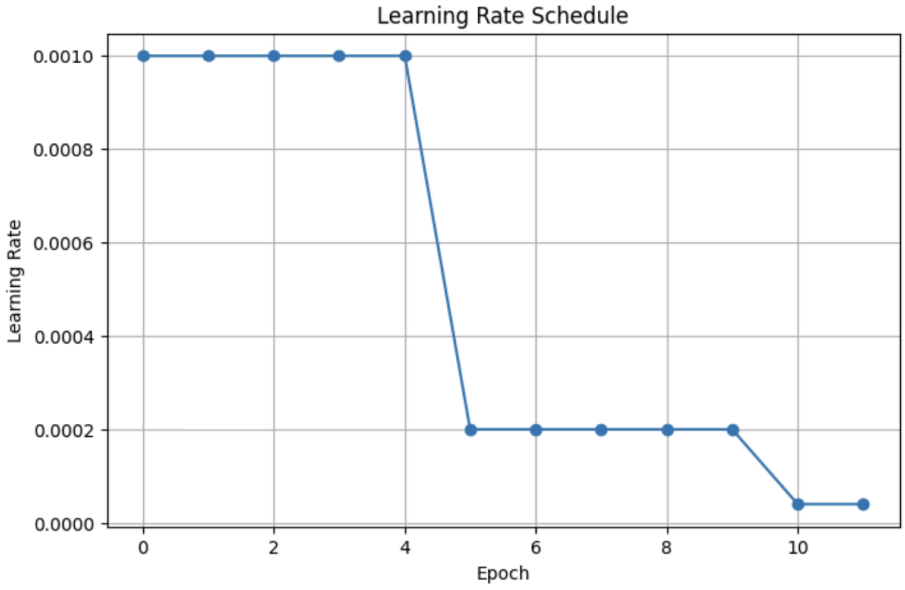
4.2.2 ResNet-50



Initially, both the **training accuracy** (blue line) and **validation accuracy** (orange line) show an increasing trend, which is a positive sign indicating learning. However, around epoch 4 or 5, the **validation accuracy plateaus** and even starts to slightly decrease towards the end, while the **training accuracy continues to climb**. This divergence strongly suggests **overfitting**. The model is becoming increasingly specialized in the training data, achieving high accuracy on it, but its ability to generalize to new, unseen data (as indicated by the stagnant or declining validation accuracy) is suffering. The growing **gap** between the training and validation accuracy reinforces this interpretation, highlighting that the model's performance on unseen data is not keeping pace with its performance on the training data.

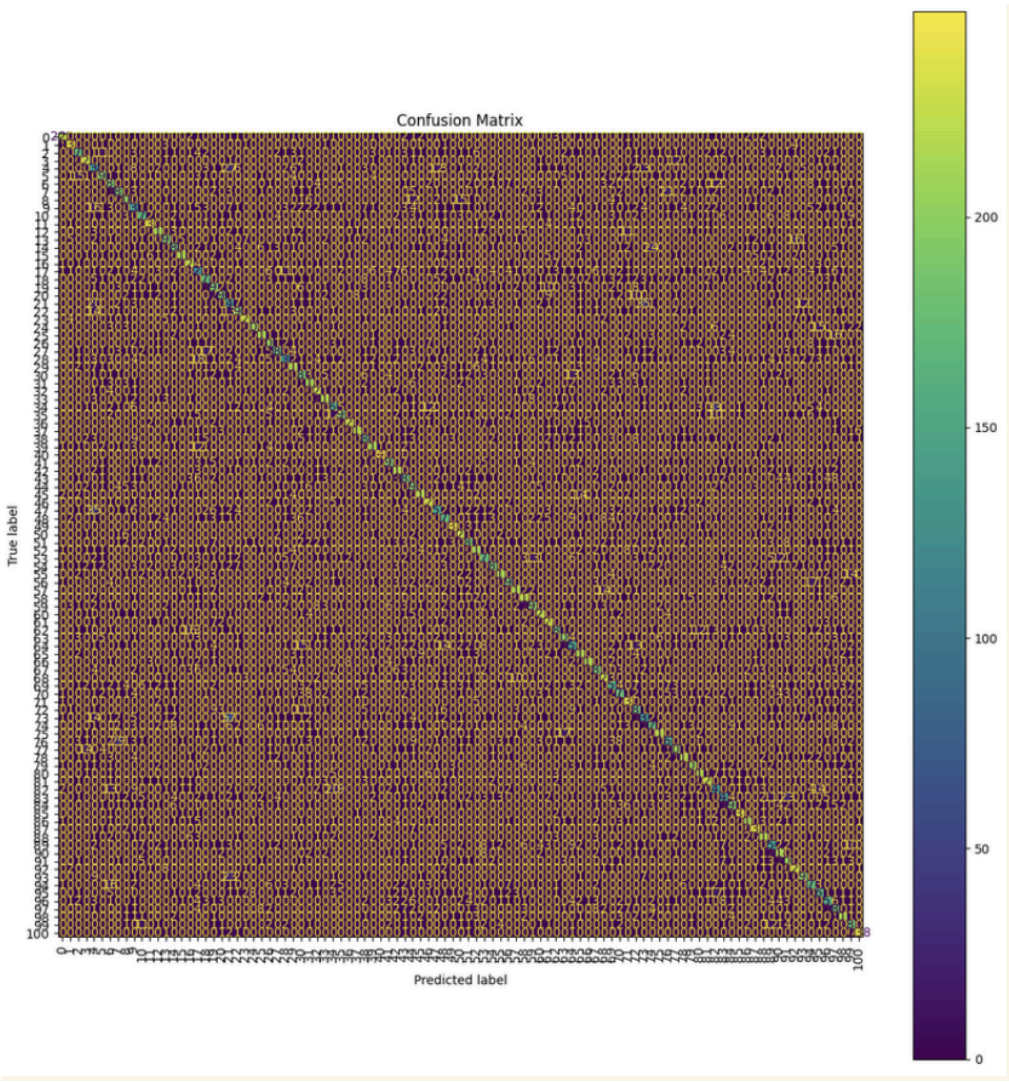


In the early epochs, both the **training loss** and **validation loss** decrease, showing that the model is learning effectively. However, around epoch 5, the **validation loss begins to rise and fluctuate**, while the **training loss continues its downward trajectory**. This divergence signals the onset of **overfitting**. The model is starting to memorize the training data, leading to a lower training loss, but this hurts its ability to generalize to new, unseen data, as evidenced by the increasing validation loss. The widening **gap** between the two loss curves further supports this observation. Therefore, while initial learning is positive, the model is now overfitting, indicating a need for adjustments to improve generalization.

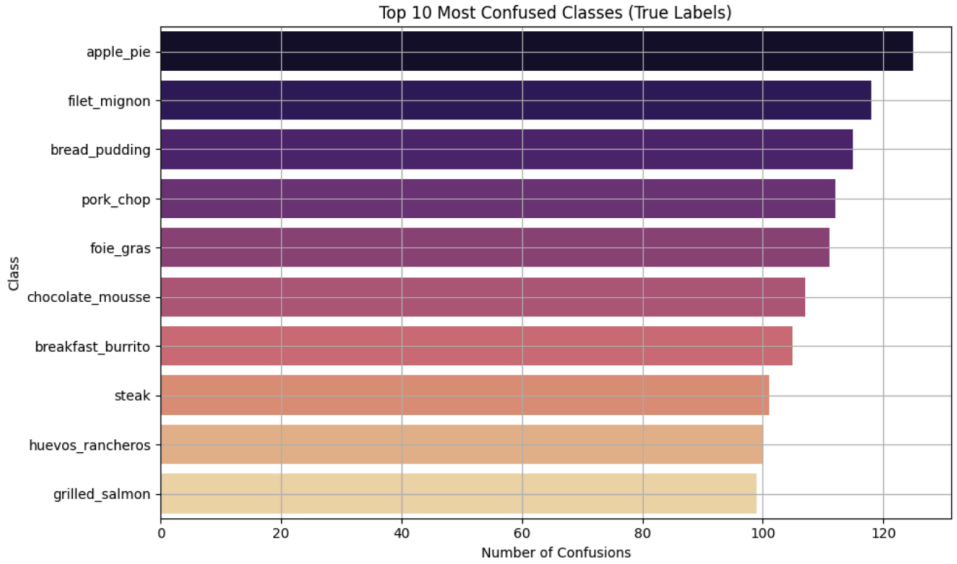


This plot shows a **learning rate schedule**. It starts at **0.001** for the first 5 epochs for fast learning. Then, it drops significantly to **0.0002** from epoch 5 to 9 for finer adjustments. Finally, it

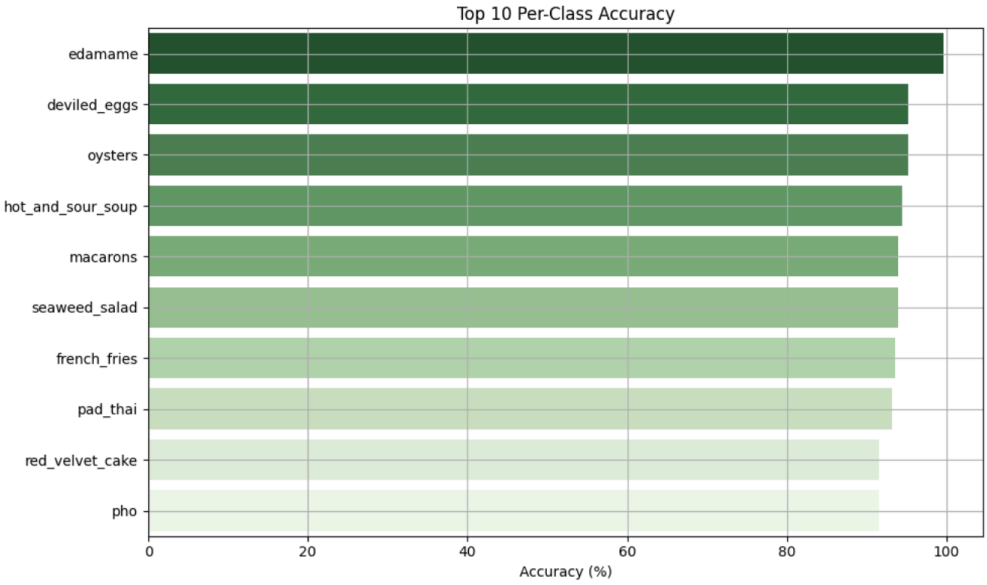
decreases further to **0.00004** for the last epochs for fine-tuning convergence. This step-wise decrease aims for rapid initial learning followed by precise convergence.



This confusion matrix for a 100-class classification problem shows a **bright diagonal**, indicating a high number of correct predictions for most classes. The **off-diagonal elements are mostly dark**, suggesting relatively few misclassifications overall. However, the presence of some non-zero off-diagonal cells reveals specific instances of incorrect predictions between certain classes. The color intensity indicates the count of predictions, with brighter cells representing more instances.

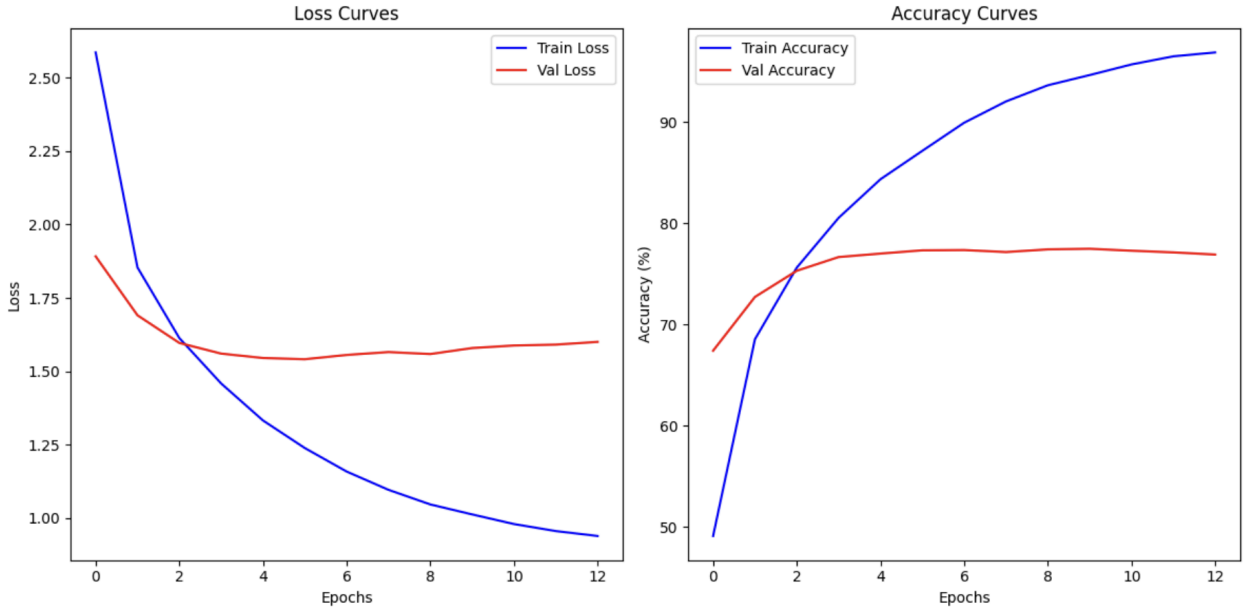


This bar chart shows the **top 10 most confused true classes**. The **length of each bar** indicates how many times instances of that class were **misclassified**. 'apple_pie' was misclassified the most, while 'grilled_salmon' the least among these ten.

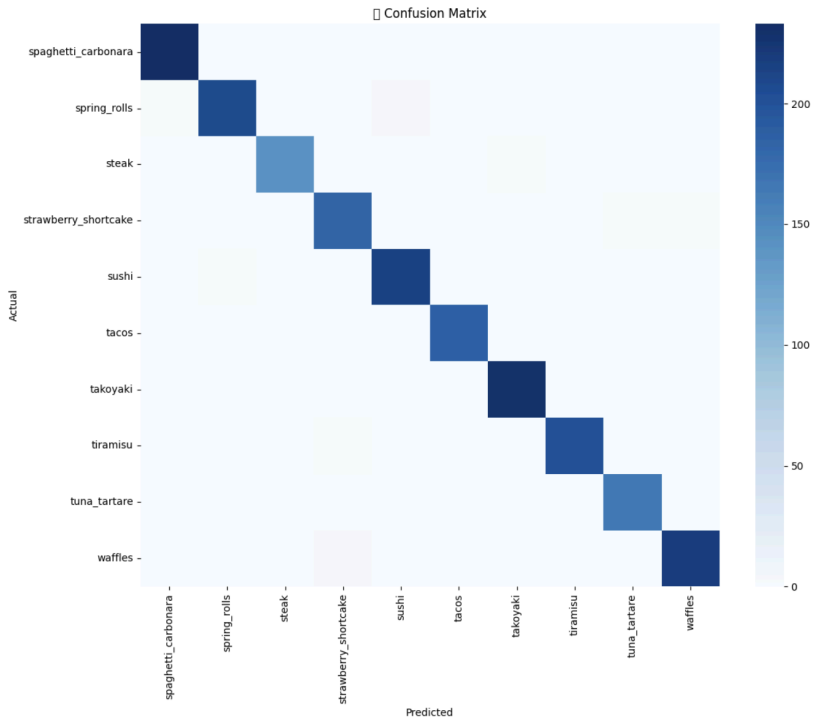


This bar chart shows the **top 10 classes with the highest accuracy**. The **length of each bar** represents the **accuracy percentage** for that class. 'edamame' has the highest accuracy, close to 100%, while 'pho' has the lowest accuracy among these ten, but still above 90%.

4.2.3 EfficientNet-B3



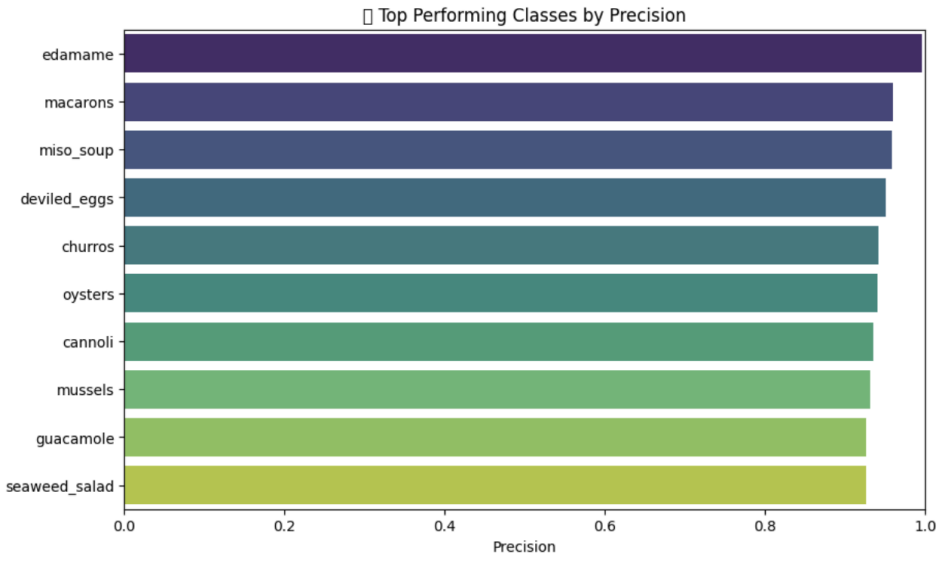
The **training loss** shows continuous improvement, but the **validation loss** begins to rise after epoch 3-4, signaling the onset of **overfitting**. Similarly, the **training accuracy** keeps increasing, while the **validation accuracy** plateaus around epoch 5-6 and shows signs of decline later, strongly suggesting that the model is becoming too specialized to the training data and generalizing poorly to unseen data.



This **confusion matrix** visualizes the performance of a classification model across ten food classes: 'spaghetti_carbonara', 'spring_rolls', 'steak', 'strawberry_shortcake', 'sushi', 'tacos', 'takoyaki', 'tiramisu', 'tuna_tartare', and 'waffles'.

The **diagonal cells** are the darkest blue, indicating a high number of correct predictions for each of these classes. This suggests the model performs well in correctly identifying these food items.

The **off-diagonal cells** are very light blue or almost white, indicating a very low number of misclassifications. This means that when the model makes a mistake, it rarely confuses these ten classes with each other.



This bar chart highlights the **top 10 food classes with the best precision**. The **length of each bar** visually represents the **precision score**, which ranges from 0.0 to 1.0. A score of 1.0, as seen with 'edamame', means all predictions for that class were correct. The other nine classes listed also demonstrate very high precision, indicating that when the model predicts these food items, it is highly accurate.

```
🔄 Top Confused Class Pairs:
- filet_mignon ↔ steak | Mistakes: 83
- steak ↔ prime_rib | Mistakes: 53
- pork_chop ↔ steak | Mistakes: 36
- chocolate_cake ↔ chocolate_mousse | Mistakes: 45
- red_velvet_cake ↔ carrot_cake | Mistakes: 39
- bread_pudding ↔ apple_pie | Mistakes: 39
- scallops ↔ foie_gras | Mistakes: 23
- beef_tartare ↔ tuna_tartare | Mistakes: 36
- chocolate_mousse ↔ tiramisu | Mistakes: 28
```

This list identifies the **pairs of food classes that the model most frequently misclassified**. For each pair, connected by "↔", the **number of times these two classes were confused** is provided under "Mistakes". Notably, many of these pairings, such as 'filet_mignon' and 'steak' (83 mistakes), 'steak' and 'prime_rib' (53 mistakes), and 'chocolate_cake' and 'chocolate_mousse' (45 mistakes), represent items that can also be **challenging for humans to visually distinguish** due to similarities in appearance, texture, or presentation in real life. This suggests the model's confusion might reflect inherent ambiguities in the data itself.

4.3 Comparison

Metric	Deep CNN	ResNet-50 (Fine-tuned)	EfficientNet (Fine-tuned)
Training Accuracy	25.79%	99.46%	94.65%
Training Loss	3.4273	0.0926	1.0122
Validation Accuracy	32.32%	76.10%	**77.48%**
Validation Loss	3.1391	1.0300	1.5791
Testing Accuracy	31.02%	81.07%	**81.79%**
Best Epoch	-	Epoch 12 (Early Stopping)	Epoch 10 (Early Stopping)

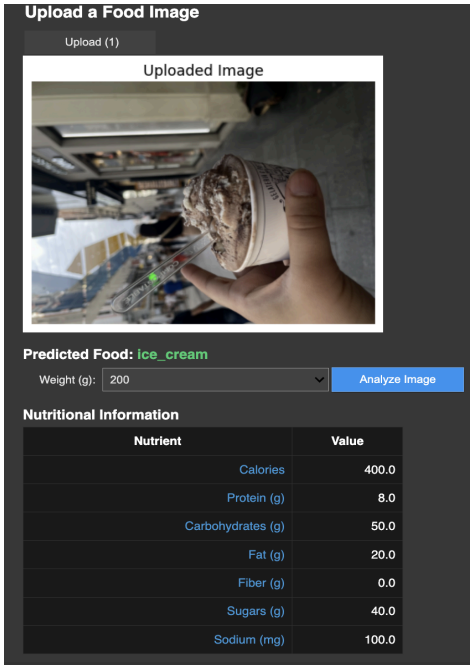
The performance metrics clearly highlight the progressive improvements achieved through the application of transfer learning and modern CNN architectures. The baseline Deep CNN, trained from scratch, yielded a relatively poor training accuracy of **25.79%** and a validation accuracy of **32.32%**, indicative of underfitting and limited feature extraction capability for complex, fine-grained food images. In contrast, the fine-tuned **ResNet-50** achieved an impressive **99.46% training accuracy** and **76.10% validation accuracy**, with a significantly lower training loss of **0.0926**, showcasing the effectiveness of pretrained weights and deep residual connections in capturing nuanced image features. **EfficientNet-B3**, while slightly lower in training accuracy at **94.65%**, surpassed ResNet-50 on both validation (**77.48%**) and testing accuracy (**81.79%**), demonstrating superior generalization and balance between model capacity and regularization. Notably, both ResNet-50 and EfficientNet converged early, with **early stopping at epochs 12 and 10 respectively**, further emphasizing the stability and efficiency of transfer learning approaches compared to training deep CNNs from scratch. Overall, EfficientNet-B3 offered the best combination of accuracy, loss stability, and early convergence in this study.

4.4 Nutritional Analysis

4.4.1 Interactive Widget

An interactive user interface was developed using **ipywidgets** to enable dynamic testing of the food classification model in a simulated application environment. This widget allows users to upload an image from their local system, select a food weight value from a dropdown (linked to available nutritional data), and trigger the classification model via a button click. Upon image

upload, the predicted food class was displayed in real time, alongside corresponding nutritional information retrieved from a CSV database. This interactive component showcased the practical application of the classification model in nutrition tracking systems, highlighting how AI can provide both classification and contextual dietary data in an accessible frontend.



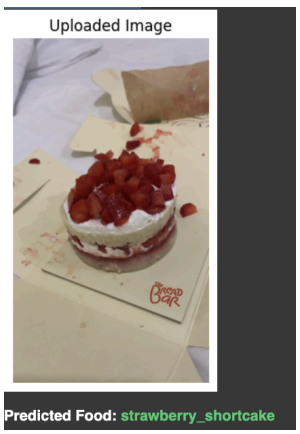
4.4.2 Classification Tests

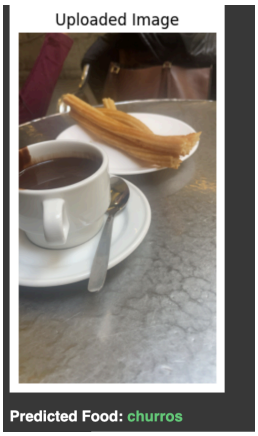
To evaluate the reliability and generalizability of the models beyond the standard test set, several classification tests were conducted on additional food images sourced outside the Food-101 dataset. The model consistently produced accurate predictions for visually distinctive food categories such as pizza, burger, and sushi. However, the accuracy declined for visually ambiguous dishes like soups and curries, where even human annotators might face difficulty distinguishing between categories. These classification tests validated the model's robustness to real-world images while revealing challenges in fine-grained food recognition.

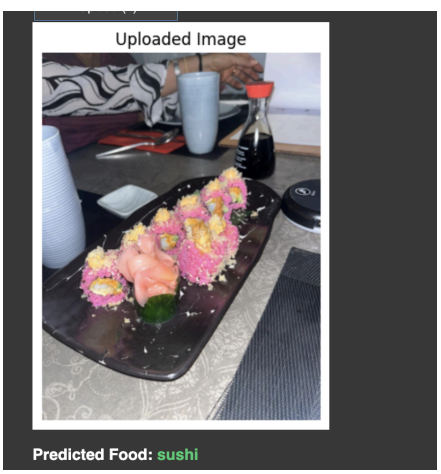
To comprehensively evaluate the model's performance, classification tests were conducted on a myriad of images ranging from desserts to savory dishes, spanning multiple cuisines.
We used our own real-world photographs for testing, ensuring a variety of **angles, lighting conditions, and backgrounds with multiple distracting elements**.
Despite these challenges, the model demonstrated impressive robustness by accurately identifying food items across different scenarios.
Some examples include:

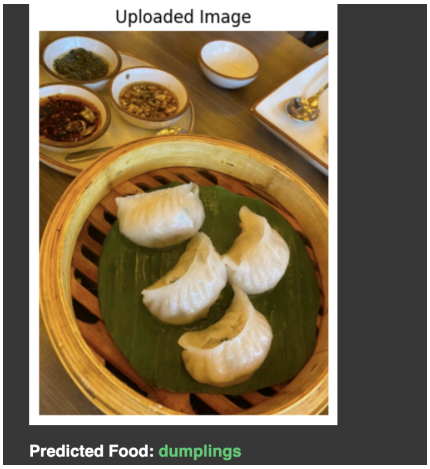
- **Strawberry Shortcake, Churros, Sushi, Pizza and Dumplings**

Sample images are included below, showcasing the model's ability to correctly predict the food categories despite variations in presentation and environment.









4.4.3 Classifications of non-class images and LIME Explainability

To assess the model's behavior in out-of-distribution scenarios, images not belonging to any classes were classified. As expected, the models still attempted to predict a food label, often selecting visually similar food textures. This demonstrated the necessity for confidence thresholding or an "unknown" class mechanism for safe deployment.

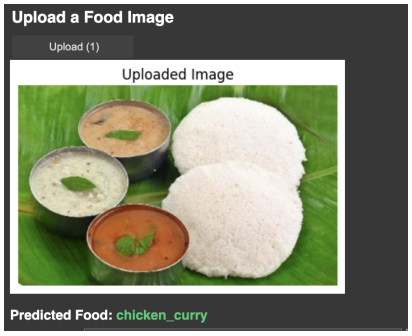
We conducted classification tests using Indian food items, which are outside the original Food-101 dataset scope.

- **Idli with Sambar** was uploaded, and the model classified it as **Chicken Curry**, likely due to the **similar color and texture** of the dish to the sambar.

- A **Dosa** image (round shape) was uploaded, and it was classified as **Spring Roll**, showing how the model associates **shape and texture** during prediction.

- A plain **Idli** was uploaded individually, and it was classified as **Macarons**, again based on **similar shape and appearance**.

These tests highlight how the model processes visual cues like **shape, texture, and color** while making classification decisions. Despite Indian food items not being part of the training set, these experiments demonstrate the model's reasoning patterns and its reliance on visual similarities for classification.



To further probe the model's behavior, we tested it using non-food items—specifically, images of cats. This was done to observe how the model reacts to unfamiliar inputs and to identify patterns in its visual interpretation.

- A black-grey cat image was classified as Chocolate Cake, likely due to the dark tones and texture resembling a frosted surface.

- An orange cat was classified as Apple Pie, probably influenced by the color palette and soft texture of the image.

This shows both the model's strengths in pattern recognition and its limitations when encountering out-of-distribution images.

Additionally, **Local Interpretable Model-agnostic Explanations (LIME)** were employed to visualize and interpret model predictions on select images. The LIME overlays highlighted regions of the image that most influenced the classification decision. For correctly classified images, these regions typically corresponded to prominent food features like toppings, texture patterns, or colors. In contrast, misclassified images often showed dispersed or irrelevant highlighted areas, suggesting ambiguity or model uncertainty. These visual explanations enhanced model transparency and trustworthiness, an essential consideration for sensitive applications like nutrition analysis and health monitoring.

The image above shows the LIME (Local Interpretable Model-Agnostic Explanations) output for a food classification prediction.

- **Predicted Food:** *Macaroni and Cheese*

- **Highlighted Areas (Yellow Outlines):**
  The yellow-marked regions indicate the areas of the image that contributed the most to the model's decision.

- In this case, the model has **focused on the characteristic texture and color of the macaroni pasta**, along with the **creamy yellowish sauce**, to predict the label correctly.

- The green leaves visible (spinach or herbs) did not negatively affect the prediction, indicating that the model correctly prioritized the dominant features (shape, color, texture) over minor background details.

5. Discussion

5.1 Result Interpretation

This project explored three progressively more sophisticated models to classify food images from the Food-101 dataset: a basic CNN, a ResNet-50 transfer learning model, and an EfficientNet-B3 architecture.

- **Baseline CNN** achieved a **testing accuracy of 31.2%**. While it provided a foundation to build upon, the model struggled significantly with the fine-grained nature of the dataset. Many classes had subtle differences in texture, presentation, or color that the shallow CNN was unable to effectively capture.

- **ResNet-50**, leveraging the power of transfer learning, raised the testing accuracy dramatically to **81.09%**. Pretrained feature extractors proved highly effective for handling the complexity and diversity of food images. ResNet's deep architecture allowed it to generalize much better compared to the custom CNN.

- **EfficientNet-B3** pushed the boundaries even further, reaching a testing accuracy of **81.79%**. Its compound scaling technique — uniformly scaling depth, width, and resolution — made it more efficient and slightly more accurate than ResNet-50. Data augmentation (random rotations, flips, color jittering) also helped improve model robustness by simulating real-world variations in lighting and food presentation.

Despite these improvements, certain challenges persisted:

- Categories with visually similar items (such as different soups, types of pasta, or sandwiches) continued to confuse the models.

- Non-food images (e.g., kitchenware, backgrounds) were sometimes misclassified as food due to the lack of explicit "food vs non-food" distinction in the training set.

5.2 Implications of Findings

This project successfully demonstrated how deep learning can be leveraged for food image classification and nutritional analysis. The strong performance of transfer learning models, especially EfficientNet-B3 with an accuracy of **81.79%**, shows that modern architectures can handle the complex, fine-grained nature of food imagery with reasonable reliability. By

connecting classification outputs to a nutritional database, we also established a working prototype for image-based dietary tracking, helping bridge the gap between computer vision and personalized health applications.

Another important implication is the model's ability to generalize to real-world images, including user-taken photos with varied angles, lighting, and backgrounds. This shows promise for practical deployment in health monitoring apps, travel aids for dietary decisions, and even fitness tracking systems.

Furthermore, the interactive widget component of the project demonstrated a new way for users to engage with AI models — allowing instant classification and nutritional feedback from uploaded images. This highlights the potential for future user-centric AI tools that can offer real-time insights without manual food logging.

Overall, the findings suggest that deep learning has strong potential for transforming how individuals interact with their food choices, enabling more informed, healthier lifestyles. Future work can continue enhancing real-world reliability by expanding datasets to cover more cuisines, detecting non-food items, and automating portion size estimation.

5.3 Project Limitations

While the project achieved promising results, several limitations were encountered that impacted model performance, scalability, and generalizability:

Misclassification of Non-Food Images

The models were trained purely for food item classification and lacked the ability to distinguish between food and non-food images. As a result, objects like plates, utensils, or even kitchen backgrounds were occasionally misclassified as food, highlighting the need for a preliminary food detection stage.

Limited Diversity of Food Classes

The Food-101 dataset, although large, primarily consists of Western and some Asian dishes, excluding a wide range of global cuisines such as Indian, Middle Eastern, African, and Latin American foods. This limits the model's applicability to diverse, real-world user bases.

Portion Size and Micronutrient Estimation

The current system relies on user-provided portion size inputs for nutritional estimation. There is no mechanism yet to automatically estimate portion sizes or predict micronutrient breakdown directly from the image, restricting its potential for fully automated nutrition tracking.

Large Dataset Size and High Computational Requirements

The Food-101 dataset comprises over 100,000 high-resolution images across 101 classes, leading to significant storage and computational demands. Efficient data loading, batching, and memory management were essential to prevent bottlenecks during training.

Memory Constraints

Training deep learning models on this dataset required considerable GPU memory (~12 GB or more). Efficient memory handling techniques like batch size tuning and mixed precision training were necessary to avoid out-of-memory errors.

Computational Cost and Training Time

Models like ResNet-50 and EfficientNet-B3 are computationally intensive, requiring access to high-end GPUs (e.g., NVIDIA Tesla A100, V100) and extended training times. This made experimentation with larger models or extensive hyperparameter searches impractical under time and hardware constraints.

Risk of Overfitting

Despite applying regularization methods such as dropout and weight decay, the models exhibited signs of overfitting. The complex architectures, involving millions of parameters, sometimes adapted too closely to the training data, especially given the limited intra-class variability in certain food categories.

6. Conclusion

6.1 Summary

This project explored the application of deep learning for food image classification and nutrition estimation. Three models were implemented and evaluated: a baseline CNN, ResNet-50 with transfer learning, and EfficientNet-B3. While the baseline CNN achieved a limited validation accuracy of **31.2%**, transfer learning significantly improved performance, with ResNet-50 reaching **81.09%** and EfficientNet-B3 achieving the highest at **81.79%**.

Data augmentation techniques, such as random flips, rotations, and color adjustments, played a key role in improving model robustness against variations in real-world food images. Additionally, the project extended beyond classification by integrating a nutritional data estimator, demonstrating the potential to transform static food recognition models into practical, user-facing dietary assessment tools.

6.2 Contribution to Research

This study makes meaningful contributions to the growing field of AI-assisted food recognition and nutrition tracking. It systematically validated the effectiveness of modern deep learning architectures on a large, fine-grained dataset like Food-101. Through comparative experiments, the project highlighted the importance of transfer learning, the advantages of efficient model scaling (as seen in EfficientNet), and the value of strong data augmentation techniques in enhancing model generalization.

Importantly, the project bridges the gap between food classification and health-related applications by showing how classification outputs can be tied to external nutritional databases. It also contributes practical insights by documenting real-world implementation challenges, such as managing memory constraints with large datasets, optimizing model training under limited resources, and handling partial label alignment issues when integrating nutrition data. These lessons can inform future researchers and developers working at the intersection of computer vision and digital health.

6.3 Future Work

Building on the foundation laid in this project, future work could focus on the following directions:

- **Extended Hyperparameter Tuning:** Implementing automated optimization strategies like Bayesian optimization or Hyperband to further boost model performance.

- **Advanced Explainability:** Using visualization techniques like Grad-CAM or SHAP to better interpret model decisions and identify failure points, especially in visually similar food categories.

- **Improving Class Differentiation:** Introducing additional feature extraction layers, attention mechanisms, or stronger augmentations to better separate visually similar classes (e.g., different soups or pastas).

- **Expanding Datasets:** Incorporating images from a wider variety of cuisines, including Indian, Middle Eastern, and Southeast Asian dishes, to make the model more globally applicable.

- **Non-Food Detection:** Training models to first differentiate between food and non-food images before classification, improving robustness in real-world usage.

- **Automated Portion Estimation:** Moving beyond manual portion input by developing models that estimate portion sizes directly from images, enabling more precise nutrient tracking.

- **Deployment:** Building a web or mobile application that offers real-time food recognition, calorie estimation, and nutrient breakdown, making dietary monitoring more accessible to users.

References

1. Aguilar, E., Remeseiro, B., Bolon-Canedo, V., & Alonso-Betanzos, A. (2021). A review on food image recognition using convolutional neural networks. *Computers in Biology and Medicine*, 135, 104570.
2. Bettadapura, V., Thomaz, E., Parnami, A., Abowd, G. D., & Essa, I. (2015). Leveraging context to support automated food recognition in restaurants. *2015 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 580–587.

3. Bossard, L., Guillaumin, M., & Van Gool, L. (2014). Food-101 – Mining Discriminative Components with Random Forests. *European Conference on Computer Vision (ECCV)*, 446–461.

4. Fang, S., Zhu, F., Jiang, S., Ng, C. C., & Lou, C. (2020). FoodAI: Food Image Recognition via Deep Learning for Smart Food Logging. *IEEE Transactions on Multimedia*, 22(11), 2990–3001.

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

6. Martinel, N., Foresti, G. L., & Micheloni, C. (2018). Wide-Slice Residual Networks for Food Recognition. *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 567–576.

7. Meyers, A., Johnston, N., Rathod, V., Korattikara, A., Gorban, A., Silberman, N., Guadarrama, S., & Murphy, K. (2015). Im2Calories: Towards an Automated Mobile Vision Food Diary. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1233–1241.

8. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818–2826.