Food Image Classification Using MobileNetV2: A Transfer Learning Approach with the Food-101 Dataset

Abdul Rafay
rafaya@bu.edu

Emily Yang
eyang4@bu.edu

Abstract

In this paper, we present a deep learning approach for food image classification using transfer learning with MobileNetV2 on the Food-101 dataset. Food image classification has numerous applications in dietary monitoring, restaurant recommendation systems, and nutritional analysis. Our approach utilizes a two-phase training strategy with initial frozen base model training followed by fine-tuning. We achieved an overall accuracy of 58% on the test set for 101 food classes, with significant performance variation across different food categories. This paper describes our methodology, presents detailed performance analyses, and discusses the challenges and future directions for improving food image classification systems from what we experimented.

1 Introduction

Food image recognition has emerged as an important area of computer vision research with applications spanning from personal health monitoring to restaurant recommendation systems. The ability to automatically identify food items from images can assist users in tracking their dietary intake, estimating nutritional content, and making informed food choices. Additionally, such systems can enhance food-related applications by adding visual recognition capabilities.

The classification of food images presents several unique challenges. Unlike general object classification, food items often have **high intra-class variability** (the same dish can look very different across restaurants) and **low inter-class variability** (different dishes may look similar). Furthermore, food appearance can be affected by numerous factors including presentation, lighting, and preparation methods.

The **Food-101 dataset** , which contains 101,000 images across 101 food categories, has become a standard benchmark for evaluating food classification algorithms. Each category contains 1,000 images, with 750 training images and 250 test images per class. This dataset represents real-world food photos with natural variations in appearance, making it an ideal testbed for developing robust food classification systems.

In this work, we implement a transfer learning approach using MobileNetV2, a lightweight convolutional neural network architecture designed for mobile and embedded vision applications. Our model employs a two-phase training strategy, first training the classification head while keeping the base model frozen, and then fine-tuning the entire network. We evaluate our model's performance across all 101 food categories and analyze patterns in classification accuracy.

2 Methodology

2.1 Dataset

We utilized the **Food-101 dataset** directly through TensorFlow Datasets (TFDS) module. For our experiments, we further split the original training set into training (80%) and validation (20%) sets, while using the original test set for final evaluation.

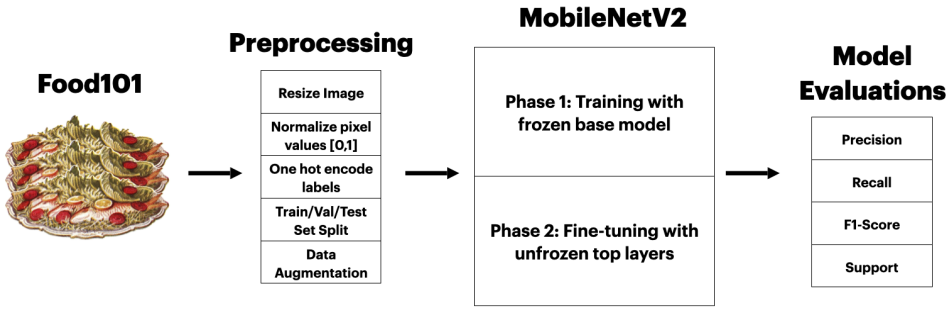


Figure 1: Model Pipeline

Various other food image datasets were explored, such as UECFOOD256, VIREO Food-172, Recipe1M+ each with their strength and weakneses, but Food101 being the standard benchmark in food image classification, and it's ease of integration into pipeline via tensorflow datasets, we chose it as our main image training source.

For relevant nutritional facts, as a baseline method we implemented a simple mapping to food item class, until we achieve satisfying performance in image classification task. For this, we referenced nutritional information from USDA FoodData Central.

2.2 Data Preprocessing and Augmentation

All images were resized to 224×224 pixels to match the input requirements of MobileNetV2. Pixel values were normalized to the range [0,1] by dividing by 255. To improve model generalization and prevent overfitting, we applied the following data augmentation techniques to the training set:

- Random horizontal flipping
- Random brightness adjustments (delta = 0.2)
- Random contrast adjustments (range 0.8 to 1.2)
- Random rotations (0°, 90°, 180°, or 270°)

These augmentations were added in anticipation that they will help the model become more robust to variations in lighting conditions, presentation angles, and food positioning; the biggest challenge in food image classification.

2.3 Model Architecture

Our model architecture is based on MobileNetV2 pretrained on ImageNet, with a custom classification head added on top. MobileNetV2 was selected due to its efficiency and effectiveness on mobile platforms, making it suitable for real-world deployment. We did try VGG16 for comparison with this model, as an experiment proposed in proposal, but due to it's significanlty weaker performace, we ended up focusing on MobileNetV2. The model architecture consists of:

1. Base model: MobileNetV2 (pretrained on ImageNet) with input shape (224, 224, 3)
2. Classification head:

- Global Average Pooling
- Dense layer (1024 units, ReLU activation)
- Dropout (0.5)
- Dense layer (512 units, ReLU activation)
- Dropout (0.3)
- Output layer (101 units, softmax activation)

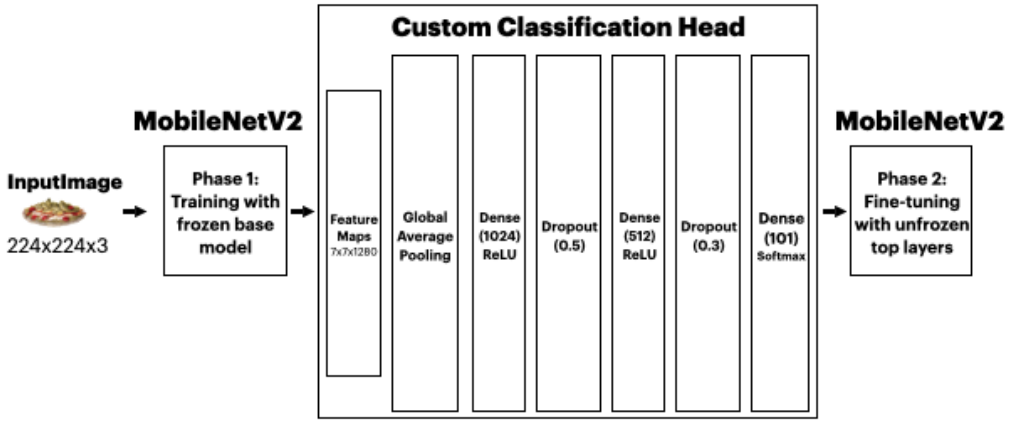


Figure 2: Model Architecture

2.4 Training Strategy

We implemented a two-phase training strategy:

Phase 1: We froze the base MobileNetV2 model and trained only the classification head for 10 epochs. This phase used an Adam optimizer with a learning rate of 0.001. By freezing all layers of the pre-trained MobileNetV2, we prevented any updates to the weights in these layers during training. This preserved the feature extraction capabilities the network originally learned from ImageNet. We only trained new layers added on top of MobilenetV2 in this phase, which are the custom classification head.

Phase 2: We then fine-tuned the entire model by unfreezing the base MobileNetV2 model and continued training for up to 20 additional epochs with a reduced learning rate of 0.0001. By making the MobileNetV2 layers trainable, we allowed the model to adjust the pre-trained weights slightly to better capture food-specific features that might differ from general ImageNet features. The reduced learning rate was intentional as the pre-trained weights already contain valuable information, so we wanted to make only small, careful adjustments.

During both training phases, we implemented the following callbacks:

- Model checkpoint to save the best model based on validation accuracy
- Early stopping with a patience of 5 epochs to prevent overfitting
- Learning rate reduction with a patience of 3 epochs (factor = 0.2, minimum learning rate = 0.00001)

The model was trained using categorical cross-entropy loss and accuracy as the evaluation metric. We used a batch size of 32 for all training and evaluation operations.

3 Results

3.1 Overall Performance

The final model achieved an accuracy of 58% on the test set with significant variances in accuracy among the different food classes. While this performance falls short of state-of-the-art results on the Food-101 dataset (which exceed 90% accuracy), it demonstrates the effectiveness of our transfer learning approach given our computational constraints and limited fine-tuning.

3.2 Performance Analysis by Category

Analysis of the classification report reveals significant performance variation across food categories:

	precision	recall	f1-score	support
apple_pie	0.00	0.00	0.00	5
baby_back_ribs	0.50	0.44	0.47	9
baklava	0.75	0.75	0.75	4
...				
tuna_tartare	0.25	0.11	0.15	9
waffles	0.78	0.64	0.70	11
accuracy			0.58	640
macro avg	0.57	0.58	0.55	640
weighted avg	0.60	0.58	0.58	640

Figure 3: Classification report of food image recognition model (truncated for brevity).

High-performing categories (F1-score $\geq$ 0.85):

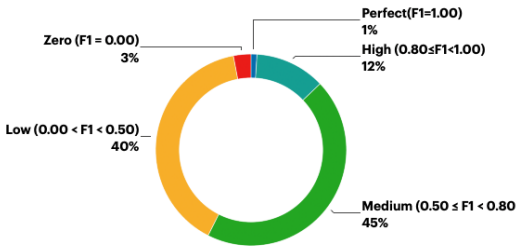
- Edamame (F1 = 1.00)
- Oysters (F1 = 0.95)
- Dumplings (F1 = 0.95)
- Sashimi (F1 = 0.93)
- Macarons (F1 = 0.92)
- Bibimbap (F1 = 0.88)
- Pho (F1 = 0.89)
- Bruschetta (F1 = 0.89)

Low-performing categories (F1-score $\leq$ 0.20):

- Apple pie (F1 = 0.00)
- Grilled cheese sandwich (F1 = 0.00)
- Pulled pork sandwich (F1 = 0.00)
- Huevos rancheros (F1 = 0.18)
- Pork chop (F1 = 0.18)
- Ceviche (F1 = 0.12)
- French onion soup (F1 = 0.15)
- Tuna tartare (F1 = 0.15)

The distribution of F1-scores across performance ranges is as follows:

- Perfect (F1 = 1.00): 1 category (1%)
- High (0.80 $\leq$ F1 < 1.00): 12 categories (12%)
- Medium (0.50 $\leq$ F1 < 0.80): 45 categories (45%)
- Low (0.00 < F1 < 0.50): 40 categories (40%)
- Zero (F1 = 0.00): 3 categories (3%)



3.3 Analysis of Classification Patterns

Looking at the performance metrics across categories, we observed several patterns:

1. **Visually distinctive foods perform better:** Foods with distinctive shapes, colors, or textures (like edamame, sashimi, and oysters) achieved higher F1-scores.
2. **Sandwich-type foods perform poorly:** Categories like grilled cheese sandwich, pulled pork sandwich, and club sandwich generally achieved lower scores, possibly due to visual similarities between different sandwich types.
3. **Precision-Recall Trade-offs:** Some categories show significant imbalances between precision and recall, suggesting model bias. For example:
   - Beef tartare has moderate precision (0.50) but high recall (0.80)
   - Cannoli has high precision (0.86) but moderate recall (0.50)
   - Nachos has perfect precision (1.00) but low recall (0.40)
4. **Support-related performance:** There appears to be some correlation between the number of test samples (support) and performance, although this is not consistent across all categories.

3.4 Proposed Experiments and Results

In our initial exploration, we focused on regression approaches to predict multiple nutritional values from food images. We experimented with different models and training strategies before transitioning to a classification framework, which ultimately yielded better performance and contributed valuable insights, hence we include them here as part of our experiments. The following are the experiments we implemented:

1. **Model Architecture Variations**

   We compared multiple architectures and output configurations under the regression framework:

   - **Architectures**: We compared a custom Keras CNN with a modified pre-trained VGG16, and noticed that the VGG16 model consistently outperformed the Keras model in both multi-output and single-output setups.
     - Multi-output (predicting protein, carbs, fat, mass, and calories together):
       Keras $R^2 \approx -1.0$, VGG16 $R^2 \approx -0.2$
     - Single-output (predicting a single nutrition category at a time):
       Keras best $R^2 \approx 0$, VGG16 best $R^2 \approx 0.44$ and worst $R^2 \approx 0.1$
   - **Multi-input Single-output**: With the VGG16 single-output configuration, we further experienced the multi-input (one overhead image and 4 side images) vs. single-input (one overhead image) structures, and find out the overall performance of multi-input model is slightly improved.

2. **Data Augmentation Strategies**

   We tested various augmentation methods to evaluate their effect on regression performance:

   - **Techniques Tested**: Shear, contrast adjustment, rotation, and width/height shift.
   - **Findings**:
     - Shear and contrast transformations degraded model performance.
     - Rotation and shift improved the accuracy.

   We concluded that certain augmentations—particularly those introducing geometric distortion or color shift—can harm pixel-sensitive regression models.

3. **Training Data Size Variation**

   To evaluate the impact of data size, we trained models on both the full dataset and a reduced 50% subset.

   - **Findings**: Reducing the training data size led to a significant drop in performance, confirming that data volume is critical to achieving stable regression results.

4. **Dropout Rate Tuning**

   We evaluated several dropout rates (0.3, 0.4, 0.5, 0.6) to explore the optimal balance between regularization and underfitting.

   - **Findings**: A dropout rate of 0.5 achieved the best validation performance. Lower values resulted in overfitting, while higher rates led to unstable training.

5. **Structure of Classification Model**

   We compared the MobileNetV2 model with one pair of top Dense-Dropout layers and the model with two pairs of top Dense-Dropout layers.

   - **Findings**: The model with two pairs of Dense-Dropout layers performs slightly better than that with only one pair of Dense-Dropout layers.

4 Discussion

4.1 Analysis of Results and Challenges

Our main model achieved 58% accuracy on the Food-101 test set, which demonstrates the effectiveness of transfer learning with MobileNetV2 while falling short of state-of-the-art results (>90%). From our experimentations we learned that this performance could be attributed to several interconnected factors:

Model Architecture Constraints:

- While MobileNetV2 is efficient and suitable for mobile applications, it has fewer parameters than larger architectures like ResNet or EfficientNet, potentially limiting its representational capacity. So, our prioritization of model efficiency for potential real-world deployment in the form of webpage came at the expense of maximum accuracy.
- Also our goal of conveying nutritional information was constricted, with more complex model needed for an amount accurate nutritional information. This would require capability to process and train on multi angle images for each food item.

Dataset Challenges:

- Food-101 contains real-world images with significant variations in lighting, presentation, and quality. This effects intra-class variability substantially, making the same dish appear dramatically different depending on preparation style and presentation.
- Visual similarity between different food categories (particularly sandwiches) created classification difficulties.
- Some categories likely have more consistent or higher quality images than others, impacting performance.
- Data set was also a constraint for our goal to convey a amount accurate nutrition information. For this to function, we would need significantly larger dataset with five images per food item accounting for images from different angles. This in turn would have introduced much complexity to model as well.

Resource Limitations:

- Computational constraints limited our hyperparameter tuning and use of more complex architecture exploration for a given timeframe.
- The training time constraints prevented more extensive fine-tuning experiments.

It is again important to notice that the performance disparity across food categories reveals clear patterns in the model's capabilities. Categories with distinctive visual features (like edamame or sashimi) achieved high performance (F1-scores >0.85), while visually similar categories were frequently confused. This suggests that the model struggles most with fine-grained visual discrimination between similar food types.

4.2 Future Work

Based on our findings, we propose several directions for future work:

1. **Model architecture improvements:** Experimenting with larger, more powerful architectures like EfficientNet or Vision Transformers could improve performance.
2. **Advanced data augmentation:** Implementing more sophisticated augmentation techniques like mixup or CutMix could enhance model robustness.
3. **Hierarchical classification:** Implementing a hierarchical approach that first classifies food into broad categories (e.g., desserts, sandwiches) before fine-grained classification.
4. **Attention mechanisms:** Incorporating attention mechanisms could help the model focus on discriminative regions of food images.
5. **Ensemble methods:** Combining predictions from multiple models could improve overall accuracy and robustness.

5 Conclusions

In this project, we presented a transfer learning approach using MobileNetV2 for food image classification on the Food-101 dataset. Our model achieved 58% accuracy on the test set, with performance varying significantly across food categories. Analysis of the results revealed that visually distinctive foods were classified more accurately, while categories with similar visual appearances posed greater challenges.

This work demonstrates both the potential and limitations of transfer learning for food classification. While our approach provides a solid foundation, achieving state-of-the-art performance would require more sophisticated models, extensive hyperparameter tuning, and advanced training techniques.

Food image classification remains a challenging problem with substantial real-world applications. Future work in this area could focus on enhancing model architectures, implementing more effective data augmentation strategies, and developing hierarchical classification approaches to improve performance on visually similar food categories.

***Note that Claude 3.7 Sonnet was used to improve grammatical clarity of this paper.**

A Resources and References

A.1 Research Papers

- Jiang, M. (2019). Food Image Classification with Convolutional Neural Networks. Stanford University. https://cs230.stanford.edu/projects_fall_2019/reports/26233496.pdf

A.2 Datasets

- **Food-101**
  A dataset containing 101,000 images of 101 food categories.
  https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/
- **Food-101 TensorFlow Datasets**
  Pre-processed version of Food-101 available through TensorFlow Datasets.
  https://www.tensorflow.org/datasets/catalog/food101
- **USDA Food Central**
  Comprehensive food and nutrient data provided by the U.S. Department of Agriculture.
  https://fdc.nal.usda.gov/

A.3 CNN Implementation Resources

- **Machine Learning Mastery: CNN for Image Classification**
  Tutorial on developing a convolutional neural network for image classification.
  https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/
- **PyTorch ResNet Models**
  Pre-trained ResNet models for image classification tasks.
  https://pytorch.org/hub/pytorch_vision_resnet/

A.4 Transfer Learning Tutorials

- **TensorFlow Transfer Learning with EfficientNet**
  Tutorial on implementing transfer learning with EfficientNet in TensorFlow.
  https://www.tensorflow.org/tutorials/images/transfer_learning

A.5 Data Augmentation Libraries

- **Albumentations**
  Powerful image augmentation library with easy PyTorch and TensorFlow integration.
  https://albumentations.ai/docs/

A.6 Object Detection Frameworks

- **Detectron2 (Facebook AI)**
  High-performance object detection toolkit for multiple food detection per image.
  https://github.com/facebookresearch/detectron2
- **YOLOv5 (Ultralytics)**
  Pre-trained object detection models, lightweight and fast.
  https://github.com/ultralytics/yolov5