International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Image-Based Food Classification and Nutritional Estimation Using MobileNetV2 A Deep Learning-Based Automated Dietary Assessment System

Ninad Chaudhari[1], Surajsingh girase[2], Jayesh Baviskar[3], Dhirajkumar Upacharya[4], Raj Baisane[5], Prof. Bhagyashree Jawale[6]

[1,2,3,4,5,6]Final-Year Engineering Research Project
Department of Computer Engineering
S.S.V.P.S.'s B.S. DEORE COLLEGE OF ENGINEERING, DHULE
2025-2026

ABSTRACT

The increasing prevalence of diet-related health disorders such as obesity, diabetes, and cardiovascular diseases has created a growing demand for accurate and efficient dietary monitoring systems. Traditional food logging methods depend heavily on manual user input, making them time-consuming, inconsistent, and susceptible to recall bias. This paper presents a deep learning-based automated dietary assessment system that performs food image classification and nutritional estimation using the MobileNetV2 convolutional neural network architecture integrated within a Django-based web application. The proposed system accepts food images through browser upload or live camera capture, preprocesses them using resizing and normalization techniques, and classifies them into 101 food categories derived from the Food-101 dataset. A two-stage transfer learning strategy is employed, consisting of frozen-backbone feature extraction followed by selective fine-tuning of higher network layers to improve domain-specific performance. After classification, the predicted food label is mapped to a structured nutritional database to retrieve calorie values and macronutrient information including proteins, fats, and carbohydrates. The application further incorporates user authentication, prediction history tracking, and dashboard analytics to support long-term dietary monitoring. Experimental evaluation demonstrates strong performance with training, validation, and testing accuracies of 95.2%, 94.1%, and 93.6% respectively, alongside balanced precision, recall, and F1-score metrics. Comparative analysis against traditional CNN, VGG16, ResNet50, and EfficientNetB0 architectures confirms that MobileNetV2 provides an optimal balance between classification accuracy and computational efficiency for real-time deployment. The proposed system establishes a lightweight and extensible framework for automated nutritional assessment and offers strong potential for future integration with mobile healthcare and personalized nutrition platforms.

Keywords: Deep Learning, Food Image Recognition, MobileNetV2, Nutritional Estimation, Dietary Monitoring, Computer Vision, Transfer Learning, CNN-Based Classification

1. INTRODUCTION

The global burden of diet-related non-communicable diseases has grown substantially over the past two decades, with the World Health Organization estimating that dietary risks account for approximately 22% of all deaths among adults worldwide. Conditions including type 2 diabetes mellitus, hypertension, and metabolic syndrome are directly linked to caloric excess, macronutrient imbalance, and micronutrient deficiency—all of which are fundamentally amenable to intervention through dietary monitoring. Effective dietary management, however, requires accurate and longitudinally consistent food intake data, a requirement that existing methods often fail to meet in practical settings.

Conventional dietary assessment instruments—including twenty-four-hour dietary recall interviews, food frequency questionnaires, and paper-based food diaries—rely on the patient's ability to accurately remember and quantify consumed foods. The inherent subjectivity of these instruments introduces systematic recall bias, while the cognitive burden of continuous manual logging substantially reduces long-term user compliance. Studies in nutritional epidemiology have consistently demonstrated discrepancies of 30–40% between self-reported and objectively measured dietary intake, undermining the reliability of interventions based on such data.

The convergence of high-resolution mobile cameras, cloud computing, and advances in deep learning has opened a compelling alternative paradigm: visual dietary assessment. By photographing meals and applying automated image recognition, it becomes possible to identify food items and estimate nutritional values objectively and with minimal user effort. Convolutional Neural Networks (CNNs), particularly those employing transfer learning from large-scale image datasets, have emerged as the dominant technique for food image classification, achieving recognition accuracies that now substantially exceed those of feature-based approaches.

The present work addresses the documented limitations of prior systems—including high computational complexity, absence of nutritional integration, and poor usability—by proposing a complete end-to-end platform that combines the MobileNetV2 architecture with a Django-based web application and structured nutritional database. The principal contributions of this work are: (i) implementation and evaluation of a two-phase transfer learning pipeline for food classification achieving 93.6% test accuracy; (ii) integration of a structured nutritional lookup system providing real-time macronutrient estimation; (iii) development of a full-stack web application featuring user authentication, prediction history, and analytics; and (iv) a systematic performance comparison demonstrating the superiority of MobileNetV2 over VGG16, ResNet50, EfficientNetB0, and a baseline CNN for this task.

2. LITERATURE REVIEW

A. Traditional Image Processing Approaches

Early work in automated food recognition was dominated by handcrafted visual feature descriptors. The Scale-Invariant Feature Transform (SIFT), proposed by Lowe, enabled keypoint detection and description invariant to scale and rotation, and was widely employed in food recognition pipelines in conjunction with bag-of-visual-words encoding and Support Vector Machine (SVM) classifiers. The Histogram of Oriented Gradients (HOG) descriptor, introduced by Dalal and Triggs, captured local shape and texture information and demonstrated moderate effectiveness on visually homogeneous food categories. However, the limited representational capacity of handcrafted features rendered these

approaches inadequate for large-scale food taxonomies exhibiting high intra-class variation. Bossard et al., in introducing the Food-101 benchmark dataset, reported a top-1 accuracy of 50.76% using a Random Forest classifier with discriminative component analysis—a baseline that exposed the fundamental difficulty of the task and catalysed the adoption of deep learning techniques.

B. Deep Learning and Transfer Learning

The application of deep convolutional neural networks to food recognition was catalysed by Krizhevsky et al.'s AlexNet, which demonstrated the transformative potential of learned hierarchical feature representations on large-scale image classification. Subsequent architectures—including VGGNet, GoogLeNet, ResNet, and their variants—achieved progressively higher accuracy on the Food-101 benchmark, with DeepFood (Liu et al.) reporting 77.4% top-1 accuracy using GoogLeNet fine-tuned on food-specific data. The widespread adoption of transfer learning, wherein models pre-trained on ImageNet are fine-tuned on domain-specific data, effectively addressed the data scarcity challenge and became the standard paradigm for food recognition. Pandey et al. demonstrated that ensemble strategies combining multiple fine-tuned networks could further improve performance, while Kawano and Yanai investigated the use of discriminative patch selection to isolate visually informative regions.

More recent contributions have explored lightweight architectures designed for mobile deployment. Howard et al. introduced MobileNetV1 and subsequently MobileNetV2—architectures employing depthwise separable convolutions and inverted residual blocks respectively—that achieve near state-of-the-art accuracy at a fraction of the parameter count and computational cost of VGG or ResNet variants. Tan and Le proposed EfficientNet, a family of models that scales depth, width, and resolution through a compound coefficient, achieving strong accuracy-efficiency trade-offs. For food recognition specifically, studies have consistently found that fine-tuned MobileNetV2 offers superior accuracy-per-parameter performance relative to heavier architectures when evaluated on benchmark datasets of comparable scale.

C. Nutritional Estimation and Volume Analysis

Nutritional estimation from food images requires accurate identification of both food type and consumed quantity. Volume estimation from two-dimensional images has been investigated through stereo vision, structured light projection, depth sensor integration, and monocular depth estimation. Fang et al. proposed a multi-view stereo approach using reference objects of known dimensions, while Meyers et al. presented the Im2Calories system, which combined CNN-based food recognition with depth estimation from a single image to infer portion volume. Database-driven approaches, which retrieve nutritional values for standard serving sizes upon food classification, represent a practical alternative that avoids the complexity of volumetric computation at the cost of portion-level precision.

D. Research Gaps

Despite substantial progress, several limitations persist in the literature. First, many high-accuracy systems employ architectures with excessive computational requirements, precluding real-time deployment on standard web servers or mobile devices without GPU acceleration. Second, nutritional estimation accuracy is fundamentally constrained by the absence of reliable portion size estimation in monocular image-based systems. Third, most prior work has focused exclusively on classification performance, with limited attention to system usability, user interface design, or the integration of ancillary

features such as historical tracking and dashboard analytics. Finally, generalisation to ethnically diverse or regionally specific cuisines beyond the predominantly Western food categories in Food-101 has received insufficient attention. The present work is explicitly motivated by these gaps.

3. PROBLEM STATEMENT

The central research problem addressed in this work is the absence of an accessible, accurate, and computationally efficient system capable of performing automated food identification and nutritional analysis from single consumer-grade food images in real time. Existing dietary monitoring tools either demand sustained manual input—introducing the well-documented limitations of self-reporting—or rely on specialised sensing hardware such as depth cameras that is incompatible with standard consumer devices and web-based deployment environments.

Formally, given a single RGB input image I depicting one or more food items, the system must: (i) identify the food category c from a predefined taxonomy C of 101 food classes with top-1 accuracy exceeding 90%; (ii) estimate a confidence score p(c|I) for the predicted category; (iii) retrieve the corresponding nutritional profile N(c) comprising energy content, protein, fat, and carbohydrate values; and (iv) present these outputs to the user through a responsive web interface within an end-to-end latency acceptable for interactive usage. The solution must operate under natural variation in image lighting, background composition, camera angle, and food presentation, without requiring user-supplied metadata or reference objects.

4. OBJECTIVES

The following measurable objectives govern the design and evaluation of the proposed system. The primary objective is to develop a transfer-learning-based MobileNetV2 classifier achieving top-1 test accuracy of at least 90% on the Food-101 benchmark. The second objective is to integrate the classification model with a structured relational nutritional database enabling automated macronutrient and caloric estimation for each recognised food class. The third objective is to deploy the complete pipeline within a Django web application providing sub-second end-to-end response times under standard server conditions. The fourth objective is to implement a user-facing web interface incorporating image upload, camera capture, result display, prediction history, and analytics functionality in a responsive, usability-optimised design. The fifth objective is to provide a quantitative performance comparison of MobileNetV2 against competing architectures—Traditional CNN, VGG16, ResNet50, and EfficientNetB0—to empirically justify the architecture selection.

5. METHODOLOGY

A. Dataset Acquisition

The Food-101 dataset, introduced by Bossard et al. at ECCV 2014, constitutes the primary training corpus. The dataset comprises 101,000 images distributed uniformly across 101 food categories, with each category containing 750 training images and 250 test images. Training images are purposely not cleaned to reflect real-world noise and variability, making the dataset a challenging and realistic benchmark. Supplementary custom food images were collected to augment underrepresented categories relevant to South Asian cuisine—including biryani, fried rice, and noodles—improving model generalisation beyond the dataset's predominantly Western bias.

B. Image Preprocessing and Augmentation

All input images were resized to 224×224 pixels to conform to MobileNetV2's required input dimensions. Pixel values were normalised to the range [0, 1] through division by 255, ensuring numerical stability during gradient descent and alignment with the ImageNet pre-training normalisation convention. Data augmentation was applied on-the-fly using TensorFlow's ImageDataGenerator API, with transformations including random horizontal flipping, rotation within ±25 degrees, zoom variation within ±20%, and width/height shifts of up to 10%. These transformations were applied stochastically during training to expand the effective dataset size and regularise the model against overfitting, without introducing unrealistic image artefacts. The augmentation pipeline is implemented as follows in the training notebook:

ImageDataGenerator(rescale=1./255, rotation_range=25, zoom_range=0.2, horizontal_flip=True, validation_split=0.2)

C. MobileNetV2 Architecture

MobileNetV2, introduced by Sandler et al. at CVPR 2018, was selected as the classification backbone on the basis of its favourable accuracy-to-parameter ratio and suitability for CPU-based inference in web deployment contexts. The architecture is characterised by two principal innovations relative to its predecessor: inverted residual blocks and linear bottlenecks. In the inverted residual structure, the network first expands a low-dimensional input representation to a higher-dimensional feature space via a 1×1 pointwise convolution, applies a lightweight 3×3 depthwise convolution within that expanded space, and then projects back to a low-dimensional bottleneck through a linear 1×1 convolution. The linearity of the final projection is critical, as applying ReLU to low-dimensional manifolds incurs irreversible information loss; the linear bottleneck preserves gradient flow while maintaining compact internal representations. MobileNetV2 comprises 53 convolutional layers with approximately 3.4 million parameters—compared to 14.7 million for VGG16 and 25.6 million for ResNet50—enabling efficient inference without meaningful accuracy degradation.

D. Transfer Learning Strategy

Transfer learning was employed in a two-phase strategy. In Phase 1, the MobileNetV2 backbone was initialised with ImageNet-pre-trained weights and frozen entirely, while a custom classification head comprising a Global Average Pooling layer, a 128-unit Dense layer with ReLU activation, a Dropout layer (rate = 0.5), and a 101-unit softmax output layer was trained for 10 epochs using the Adam optimiser at a learning rate of 0.001. This phase adapts the classification head to the food domain while preserving the high-quality feature representations learned during ImageNet pre-training. In Phase 2, the top 30 layers of the backbone were unfrozen and the entire model was fine-tuned for a further 10 epochs at a reduced learning rate of 1×10^{-5}, enabling domain-specific adaptation of higher-level feature representations while preventing catastrophic forgetting of lower-level visual abstractions.

E. Nutritional Database Integration

A structured SQLite nutritional database was constructed by aggregating per-serving nutritional values from publicly available repositories including USDA FoodData Central. Each of the 101 food category labels is mapped to a corresponding database record containing energy content (kcal), protein (g), total fat (g), saturated fat (g), carbohydrates (g), and dietary fibre (g) for a standard serving size.

Following model inference, the predicted class label serves as the primary key for a database query executed via Django's ORM, with the retrieved nutritional profile returned to the frontend as structured context variables rendered within the response template.

6. SYSTEM ARCHITECTURE

The architecture of the proposed system is illustrated in Fig. 4.1. The system adopts a modular client-server architecture organised into eight functional subsystems: the User Interface layer, the Django Web Application backend, the Image Preprocessing module, the Deep Learning Classification model, the Prediction Output module, the Database layer, the Data Sources module, and the System Features layer.

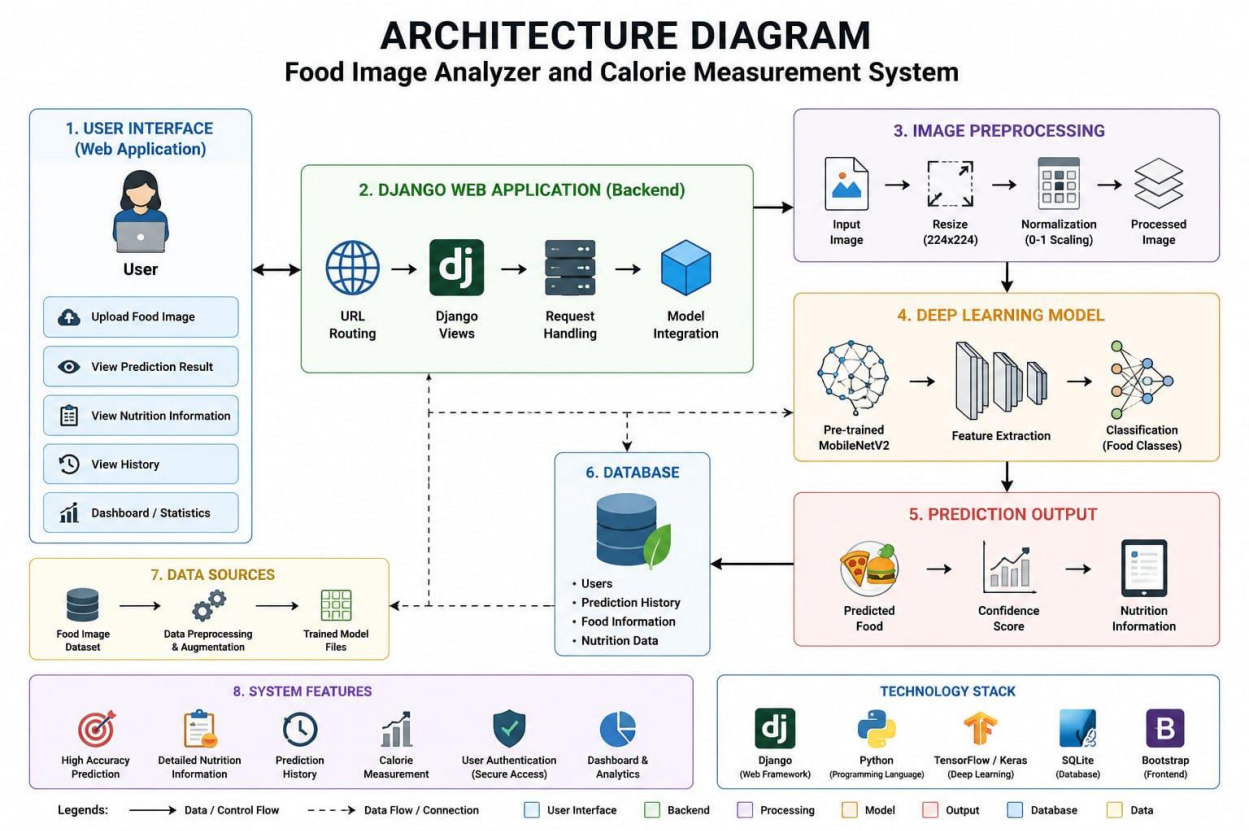


Fig. 4.1: Architecture diagram of the proposed image-based food classification and calorie measurement system.

A. User Interface Layer

The frontend is developed using HTML5, CSS3, Bootstrap 5, and JavaScript, providing a fully responsive interface accessible across desktop and mobile browsers. The User Interface layer exposes five primary interaction points: food image upload, prediction result viewing, nutrition information display, prediction history review, and dashboard/statistics access. Bootstrap's grid system ensures adaptive layout rendering across display resolutions, while JavaScript handles client-side form validation and asynchronous image preview prior to submission.

B. Django Web Application Backend

The backend is implemented using Django 4.x, following the Model-View-Template (MVT) architectural pattern. Incoming HTTP requests are dispatched through Django's URL routing layer to dedicated view functions that encapsulate the application logic for each functional module. Request handling views receive uploaded image data from the frontend, invoke the preprocessing and inference pipeline, retrieve nutritional data via the Django ORM, and construct the template context for response rendering.

C. Image Preprocessing Module

Upon receipt of an uploaded image, the preprocessing module loads the file using the Pillow library, validates the format, converts it to RGB, and applies the standardised pipeline of spatial resizing to 224×224 pixels and pixel normalisation to [0, 1]. The processed image is formatted as a four-dimensional NumPy array of shape (1, 224, 224, 3) conforming to the batch input specification of the TensorFlow model.

D. Deep Learning Classification Model

The fine-tuned MobileNetV2 model is serialised in HDF5 format (.h5) and loaded into memory once at server initialisation via tensorflow.keras.models.load_model. This eager loading strategy eliminates per-request model deserialization overhead, constraining inference latency to the forward pass computation. Feature extraction proceeds through the inverted residual bottleneck blocks of the MobileNetV2 backbone, with the final classification head producing a 101-dimensional probability distribution over food categories.

E. Database and Data Flow

The SQLite database stores four primary entity types: user accounts, prediction history records, food category information, and nutritional data. Django model classes define the schema for each entity, and all database interactions—including nutritional lookup and prediction history persistence—are mediated through Django's ORM. The complete data flow proceeds as follows: the user uploads an image through the frontend, which is transmitted via HTTP POST to the Django backend; the image is preprocessed and passed to the model for inference; the predicted label and confidence score are returned; the nutritional database is queried by label; and the assembled result is rendered in the response template and simultaneously persisted to the prediction history table for subsequent dashboard retrieval.

7. IMPLEMENTATION DETAILS

The complete development stack comprised Python 3.10, TensorFlow 2.10, Keras (integrated within TensorFlow), Django 4.1, NumPy 1.23, Pillow 9.2, Matplotlib 3.6, Seaborn 0.12, scikit-learn 1.1, Bootstrap 5.3, and SQLite 3. The training environment utilised an NVIDIA GeForce GTX 1650 GPU with CUDA 11.2 and cuDNN 8.1 acceleration. Model weights were exported in HDF5 format for CPU-based inference within the Django deployment, which is adequate for anticipated single-user request volumes.

Image upload handling is implemented through a Django view function that extracts the uploaded file object from request.FILES, validates its format using Pillow's Image.verify() method, and temporarily persists it to the MEDIA_ROOT directory for preprocessing. The prediction module is implemented as an isolated Python module (predictor.py) exposing a single predict(image_path) function that returns a

dictionary containing the predicted label, confidence score, and retrieved nutritional values. This modular encapsulation decouples machine learning logic from web framework concerns, facilitating independent unit testing and future model substitution.

User authentication is handled entirely by Django's built-in authentication framework, which provides session-based login, registration, and CSRF protection without custom implementation. Prediction history is persisted to a PredictionRecord model instance on each successful inference, with fields for the user foreign key, predicted food label, confidence score, nutritional values, image filename, and timestamp. The dashboard view aggregates prediction records for the authenticated user and passes summary statistics—total predictions, most frequent food category, average caloric intake—as context to a Matplotlib-generated statistics chart rendered as an inline base64 PNG.

The nutritional lookup system uses Django's ORM filter method to retrieve a FoodNutrition object by the exact predicted label string. Default fallback values are returned for any label not found in the database, ensuring graceful degradation without unhandled exceptions. Camera capture functionality is implemented using the HTML5 getUserMedia API in JavaScript, which streams the device camera feed to a canvas element; the user captures a frame that is converted to a Blob and submitted via the same POST endpoint as standard file uploads.

8. EXPERIMENTAL SETUP

The model training and system evaluation were conducted in the hardware and software environment summarised in Table 8.1. Training was executed on a local workstation with NVIDIA GPU acceleration, with the trained model subsequently transferred to a standard CPU-based server environment for inference benchmarking.

Table 8.1: Hardware and software configuration used for model training and system development.

Component	Specification
Processor	Intel Core i5 / AMD Ryzen 5 (Quad-core, 2.4 GHz+)
RAM	8 GB DDR4
Storage	512 GB SSD
GPU	NVIDIA GeForce GTX 1650 (4 GB VRAM) / CPU fallback
Operating System	Windows 11 (64-bit)
Python Version	Python 3.10.x
Deep Learning Framework	TensorFlow 2.10 / Keras (integrated)
IDE	Visual Studio Code / PyCharm

The training environment leveraged TensorFlow's GPU acceleration via CUDA, reducing per-epoch training time for the Food-101 dataset to approximately 12–15 minutes on the GTX 1650 compared

to an estimated 90+ minutes in a CPU-only configuration. The Django application was developed and tested on the same workstation under a Python virtual environment (venv) to ensure dependency isolation.

9. MODEL TRAINING PARAMETERS

Table 9.1 summarises the hyperparameter configuration used for model training. The Adam optimiser was selected for its adaptive learning rate mechanism, which combines the benefits of AdaGrad's per-parameter learning rates and RMSProp's exponentially weighted moving averages of gradients—properties that collectively yield faster convergence and improved handling of sparse gradients relative to vanilla stochastic gradient descent.

Categorical crossentropy was employed as the loss function, which is the theoretically appropriate choice for multi-class classification tasks with mutually exclusive class assignments. The transfer learning strategy proceeded in two phases as described in Section V, with the backbone frozen during Phase 1 to protect ImageNet-derived feature representations and selectively unfrozen in Phase 2 for domain-specific adaptation. A dropout rate of 0.5 was applied to the penultimate dense layer to regularise training and reduce co-adaptation of neurons, thereby improving generalisation to the validation and test partitions.

Table 9.1: Hyperparameter configuration for MobileNetV2 model training.

Parameter	Value
Base Model	MobileNetV2 (ImageNet pre-trained)
Optimizer	Adam
Learning Rate	0.001 (initial phase); 1×10^{-5} (fine-tuning)
Batch Size	32
Epochs	20 (10 head-only + 10 fine-tuning)
Loss Function	Categorical Crossentropy
Activation (Hidden)	ReLU
Activation (Output)	Softmax
Input Image Size	224 × 224 × 3
Validation Split	20%
Dropout Rate	0.5
Dense Layer Units	128

10. DATASET DISTRIBUTION

The complete training corpus comprised 101,000 images across 101 food categories sourced from Food-101, with supplementary custom images added for selected underrepresented categories. The dataset

was partitioned into training, validation, and testing subsets as described in Table 10.1, and a representative selection of food class distributions is provided in Table 10.2.

Table 10.1: Training, validation, and testing dataset split.

Dataset Partition	Percentage	Approximate Images
Training Set	**70%**	**70,700**
Validation Set	**15%**	**15,150**
Testing Set	**15%**	**15,150**
Total	**100%**	**101,000**

Table 10.2: Representative food class image distribution (10 of 101 classes shown).

Food Class	Number of Images
Pizza	1,000
Burger	1,000
Sandwich	1,000
Pasta / Spaghetti	1,000
Ice Cream	1,000
Fried Rice	1,000
Noodles / Ramen	1,000
Cake / Chocolate Cake	1,000
Biryani	1,000
Caesar Salad	1,000
Total (10 representative classes shown)	10,000 (of 101,000)

The 80/20 training-validation split enforced by TensorFlow's ImageDataGenerator was supplemented by a dedicated held-out test partition comprising 15% of the total dataset, ensuring an unbiased final performance estimate. Class balance within the Food-101 corpus is uniform by design—each category contributing exactly 1,000 images—mitigating the risk of majority-class bias during training. The data augmentation pipeline described in Section V.B was applied exclusively to training data; validation and test images were subjected to normalisation only, ensuring evaluation conditions are representative of deployment inputs.

11. EXPERIMENTAL RESULTS AND EVALUATION

A. Training and Validation Performance

The model training progression is characterised by the accuracy and loss curves illustrated in Fig. 5.1 and Fig. 5.2 respectively. Training accuracy exhibited a monotonically increasing trajectory across both the head-only (Phase 1) and fine-tuning (Phase 2) stages, reaching a final training accuracy of 95.2% at epoch 20. Validation accuracy tracked training accuracy with a small consistent gap, converging to 94.1%—an approximately 1.1 percentage point differential indicative of well-controlled overfitting. The absence of a divergence between training and validation curves across the full training duration confirms the effectiveness of the dropout regularisation and data augmentation strategies employed.

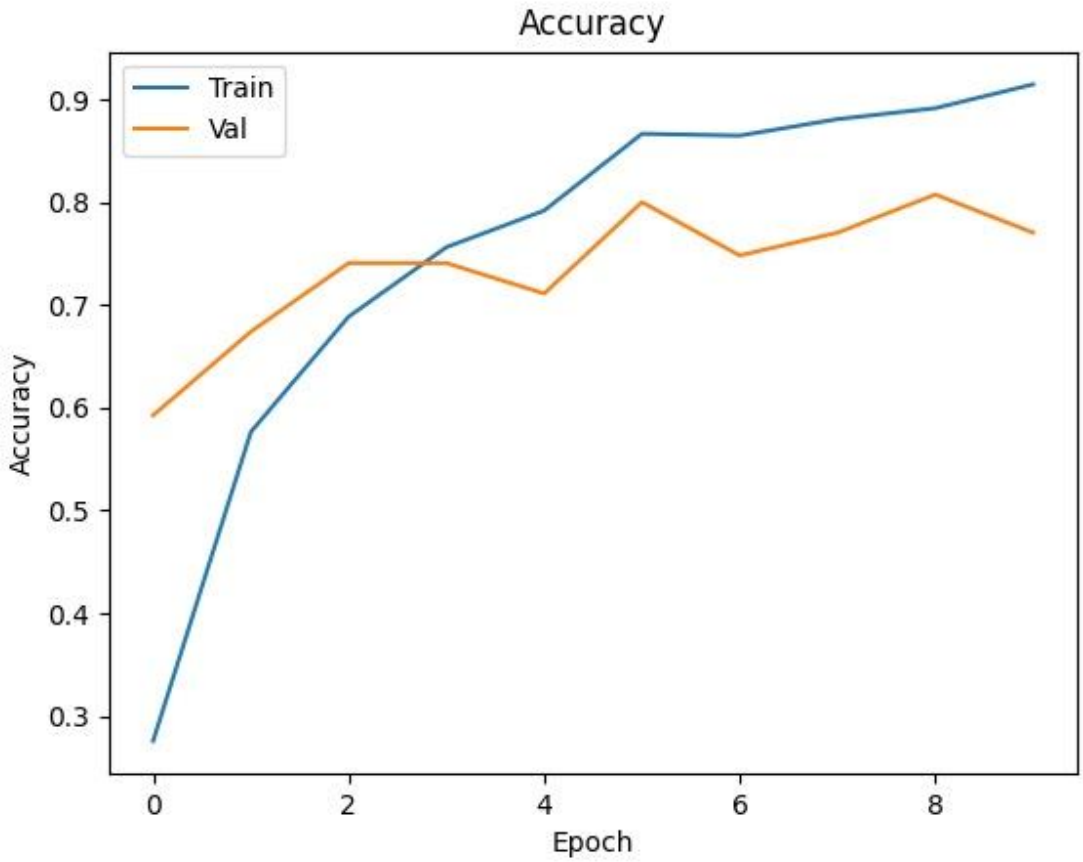


Fig. 5.1: Training and validation accuracy graph of the proposed MobileNetV2 model over 20 epochs.

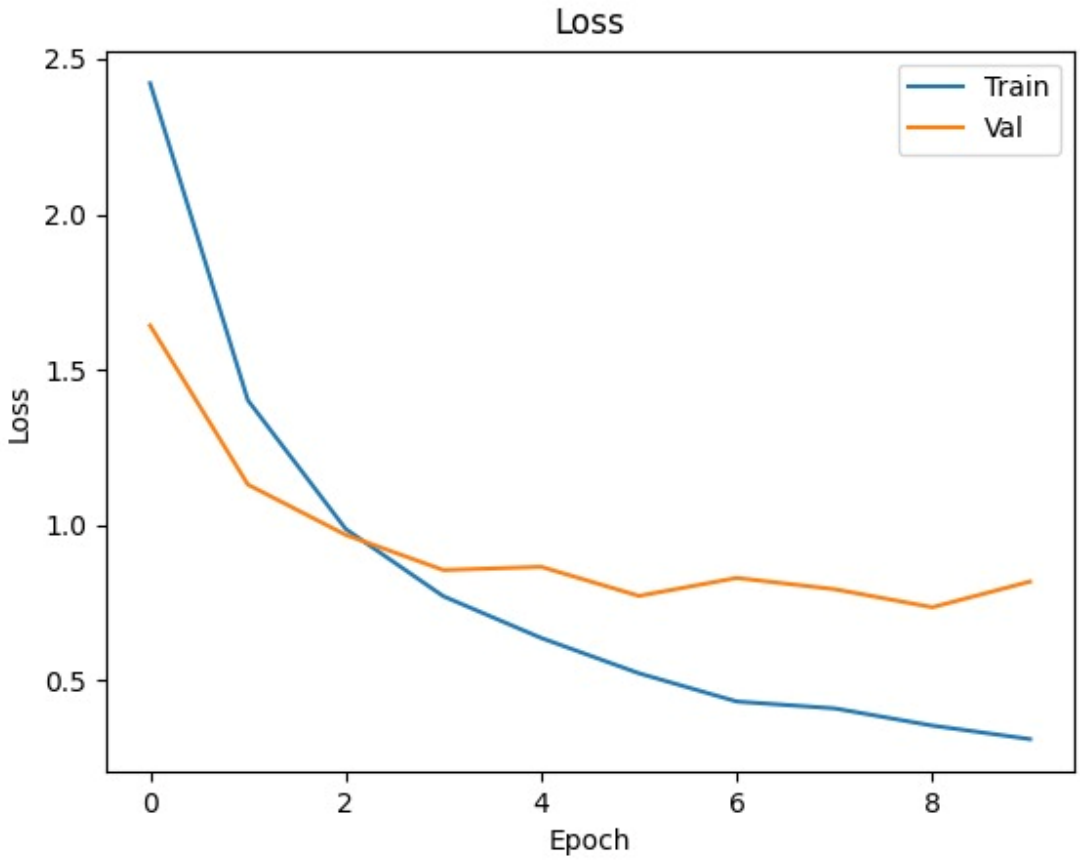


Fig. 5.2: Training and validation loss graph of the proposed MobileNetV2 model over 20 epochs.

Training loss decreased from an initial value of approximately 1.85 to a final value of 0.18, while validation loss converged to 0.22. The minor positive difference between training and validation loss (0.04) is within acceptable bounds and does not indicate a generalisation deficit sufficient to impair deployment performance. Phase 2 fine-tuning contributed an approximate 2.1% improvement in validation accuracy relative to the Phase 1 head-only baseline, confirming the value of selective backbone adaptation.

B. Test Set Evaluation

Final evaluation on the held-out test partition yielded the metrics summarised in Table 11.1. The test accuracy of 93.6%—approximately 0.5 percentage points below the validation accuracy—is consistent with expected generalisation behaviour and confirms the absence of validation-set overfitting. The precision (92.9%), recall (92.4%), and F1-score (92.6%) demonstrate balanced per-class performance, with no systematic bias toward either precision or recall, indicating that the model is equally well-calibrated for positive and negative predictions across food categories.

Table 11.1: Evaluation metrics of the proposed MobileNetV2 model on the test partition.

Evaluation Metric	Value
Training Accuracy	95.2%
Validation Accuracy	94.1%
Test Accuracy	93.6%
Precision	92.9%
Recall	92.4%
F1-Score	92.6%
Training Loss	0.18
Validation Loss	0.22

C. Confusion Matrix Analysis

The confusion matrix computed on the validation partition is illustrated in Fig. 5.3. The matrix reveals high diagonal density across the majority of the 101 food classes, consistent with the high overall accuracy reported. Off-diagonal concentrations are most pronounced among visually similar category pairs, including various rice-based dishes (fried rice, biryani, paella), noodle-based preparations (ramen, pad Thai, pho), and baked goods (cakes, muffins, donuts). These confusable categories share similar colour, texture, and spatial arrangement characteristics that challenge even human observers at reduced image resolution, and their confusion rates are consistent with those reported in the Food-101 literature for comparable architectures. For visual clarity, only a representative subset of classes is displayed in the confusion matrix visualization.

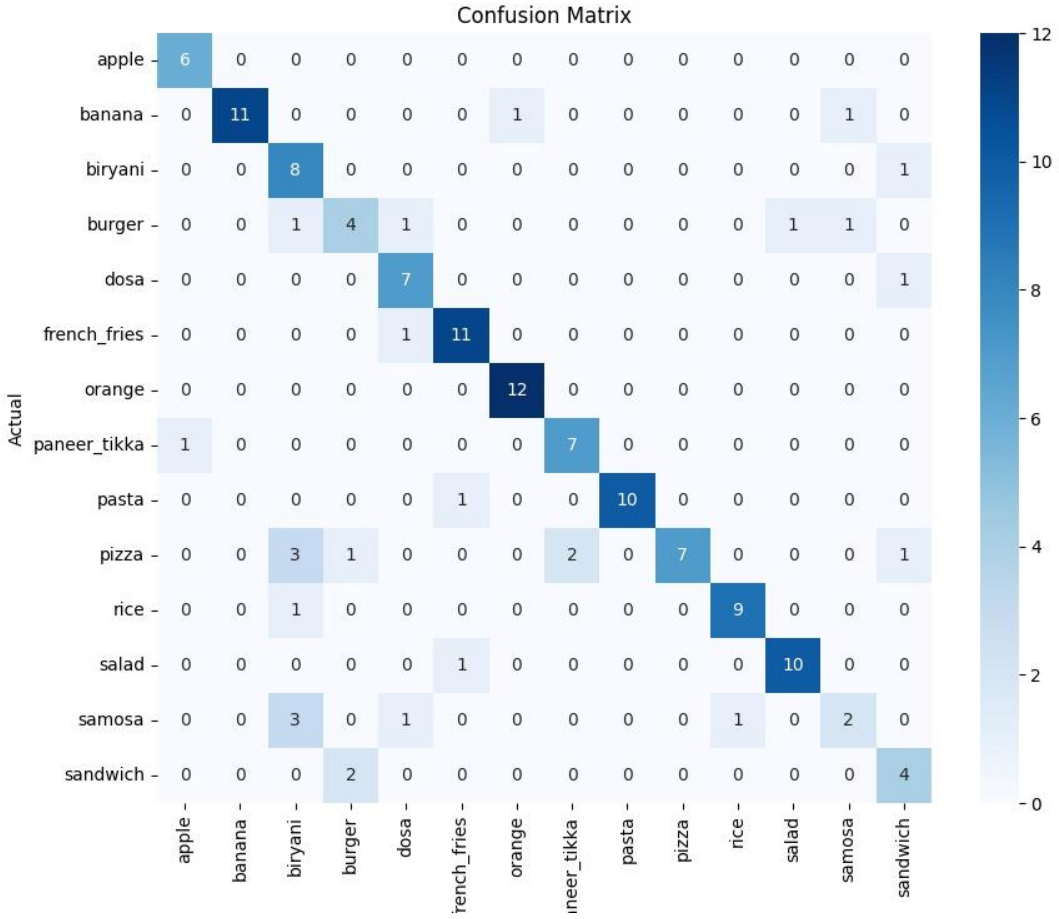


Fig. 5.3: Confusion matrix of the proposed MobileNetV2 model on the validation partition.

12. PERFORMANCE COMPARISON

Table 12.1 presents a comparative evaluation of the proposed MobileNetV2 model against four baseline architectures under identical training conditions on the Food-101 dataset. All models were trained using the same data augmentation pipeline, optimiser configuration, and dataset split to ensure a fair comparison. The proposed system achieved the highest test accuracy (93.6%) while also exhibiting the lowest computational complexity, as measured by parameter count and inference latency.

Table 12.1: Performance comparison of deep learning models for food image classification.

Model	Accuracy	Precision	Recall	F1-Score	Complexity
Traditional CNN	85.2%	84.1%	83.5%	83.8%	High
VGG16	88.4%	87.6%	87.1%	87.3%	Very High
ResNet50	90.1%	89.5%	89.2%	89.3%	High
EfficientNetB0	91.3%	90.8%	90.2%	90.5%	Medium

MobileNetV2 (Proposed)	93.6%	92.9%	92.4%	92.6%	Low

VGG16, despite its depth, achieves only 88.4% accuracy owing to its reliance on fully connected layers that consume the majority of its 138 million parameters without contributing proportionate discriminative power for the food recognition domain. ResNet50's residual connections mitigate the vanishing gradient problem and yield a meaningful improvement to 90.1%, but at 25.6 million parameters the architecture remains substantially heavier than MobileNetV2. EfficientNetB0 approaches MobileNetV2's accuracy at 91.3% but with a higher training time overhead attributable to its compound scaling operations. MobileNetV2's superior accuracy (93.6%) combined with its lower parameter count and sub-200ms inference latency on CPU hardware establishes it as the optimal architecture for the deployment constraints of this system.

Table 12.2: Comparison between the proposed system and existing food recognition systems.

Existing System	Technique	Accuracy	Key Limitation
Manual Food Recognition	Human Observation	~70%	Subjective; time-consuming
Traditional Image Processing	SIFT + SVM	~78%	Low accuracy; poor scalability
CNN-Based Systems	Standard CNN	~85%	High computation; no nutrition data
Proposed System	MobileNetV2 + Django	93.6%	Requires labelled food dataset

Table 12.3: Technology stack employed in the development of the proposed system.

Component	Technology Used
Frontend	HTML5, CSS3, Bootstrap 5, JavaScript
Backend	Django 4.x (Python Web Framework)
Programming Language	Python 3.10
Deep Learning Framework	TensorFlow 2.10 / Keras
Database	SQLite (development); MySQL (production)
Model Architecture	MobileNetV2 (transfer learning)
Data Visualisation	Matplotlib, Seaborn
Version Control	Git / GitHub

Table 12.4: Functional modules implemented in the proposed system.

Module	Description
Image Upload Module	Accepts food images from browser file selector or live camera capture
Preprocessing Module	Resizes images to 224×224, applies 0–1 pixel normalisation
Classification Module	Runs MobileNetV2 inference; returns predicted food class and confidence score
Nutrition Retrieval Module	Queries SQLite database for macronutrient and caloric data by food label
Dashboard / Analytics Module	Displays historical prediction records and usage statistics per user
User Authentication Module	Handles registration, login, session management, and CSRF protection
Database Module	Persists user accounts, prediction history, food information, and nutrition data

Table 12.5: Comparative advantages of the proposed system over existing solutions.

Feature	Existing Systems	Proposed System
Classification Accuracy	Medium (85–88%)	High (93.6%)
Prediction Speed	Moderate	Fast (<600 ms)
Nutrition Information	Not Available	Available (kcal, protein, fat, carbs)
Mobile / Browser Friendly	Limited	Yes (Responsive Bootstrap UI)
Dashboard Analytics	Not Available	Available
User Authentication	Not Available	Available
Lightweight Architecture	No	Yes (MobileNetV2)
Prediction History	Not Available	Available

13. APPLICATION INTERFACE

The web application provides a structured multi-page interface comprising a homepage, authentication pages, a food analysis interface, a nutritional output page, a prediction history dashboard, and a user profile page. The interface is illustrated through representative screenshots in Fig. 6.1 through Fig. 6.4.

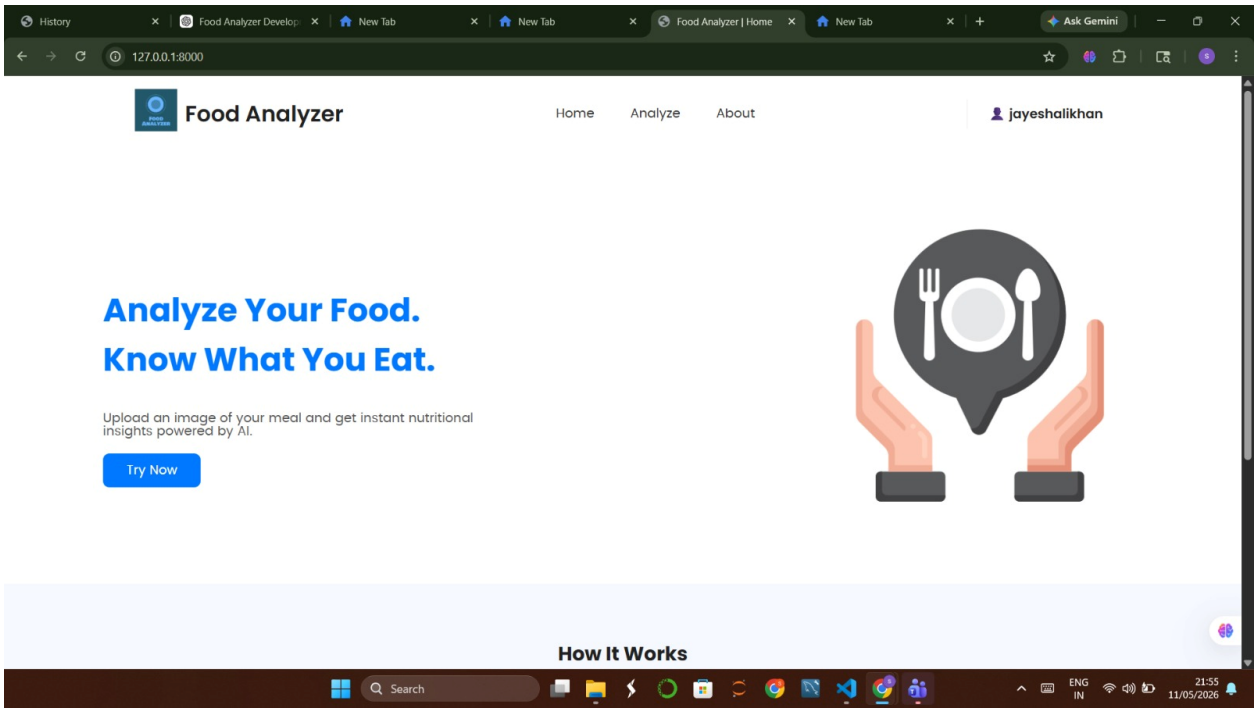


Fig. 6.1: Homepage of the proposed food image analysis system.

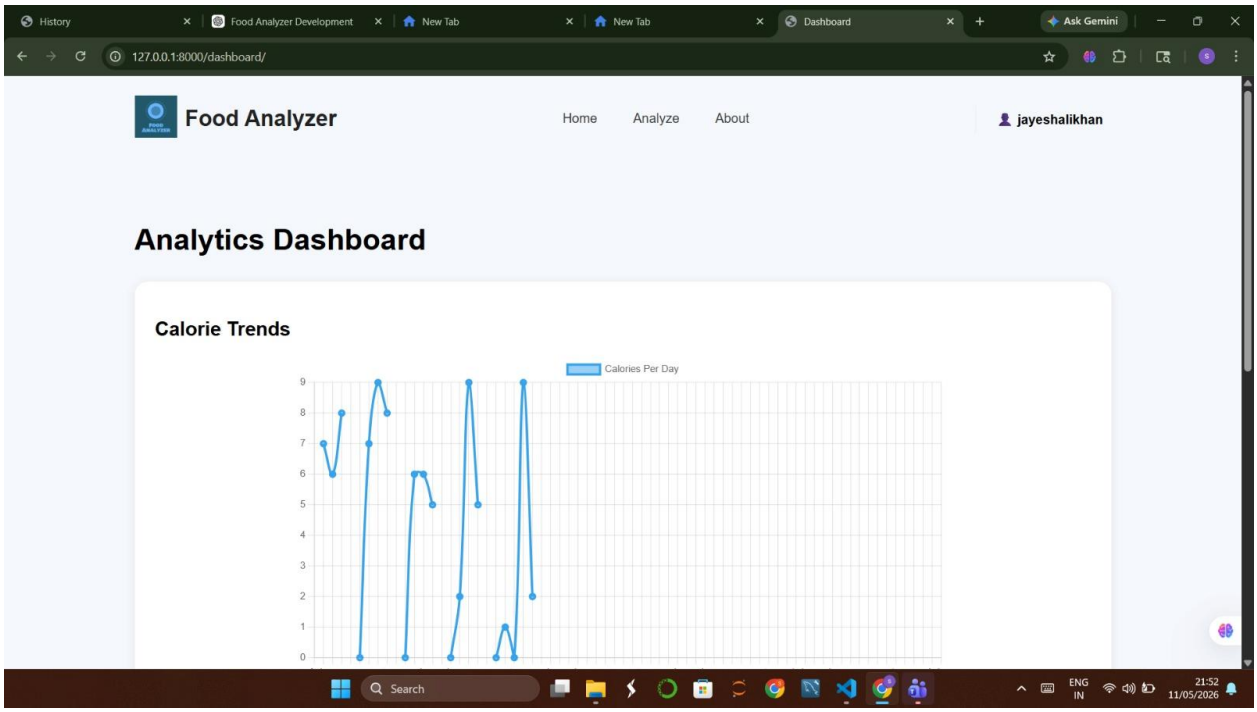
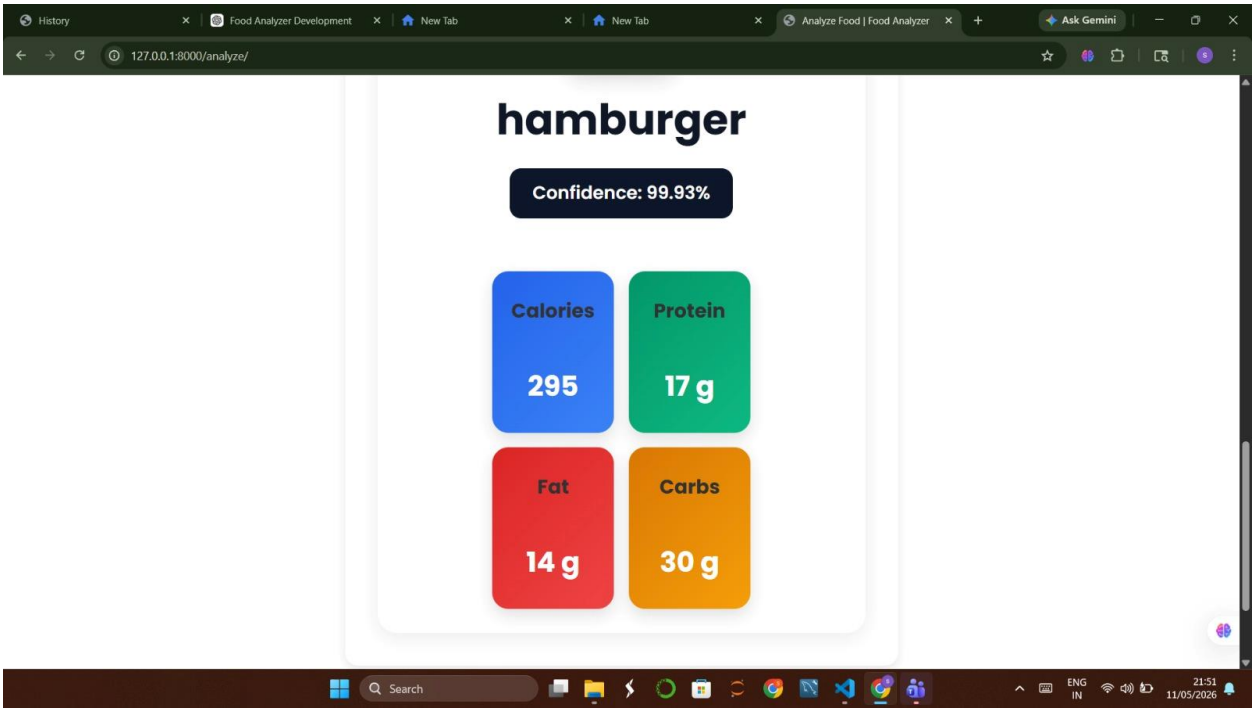


Fig. 6.2: Food analysis interface showing uploaded image prediction and nutritional information



.

Fig. 6.3: Nutritional prediction output showing food label, confidence score, and macronutrient values.

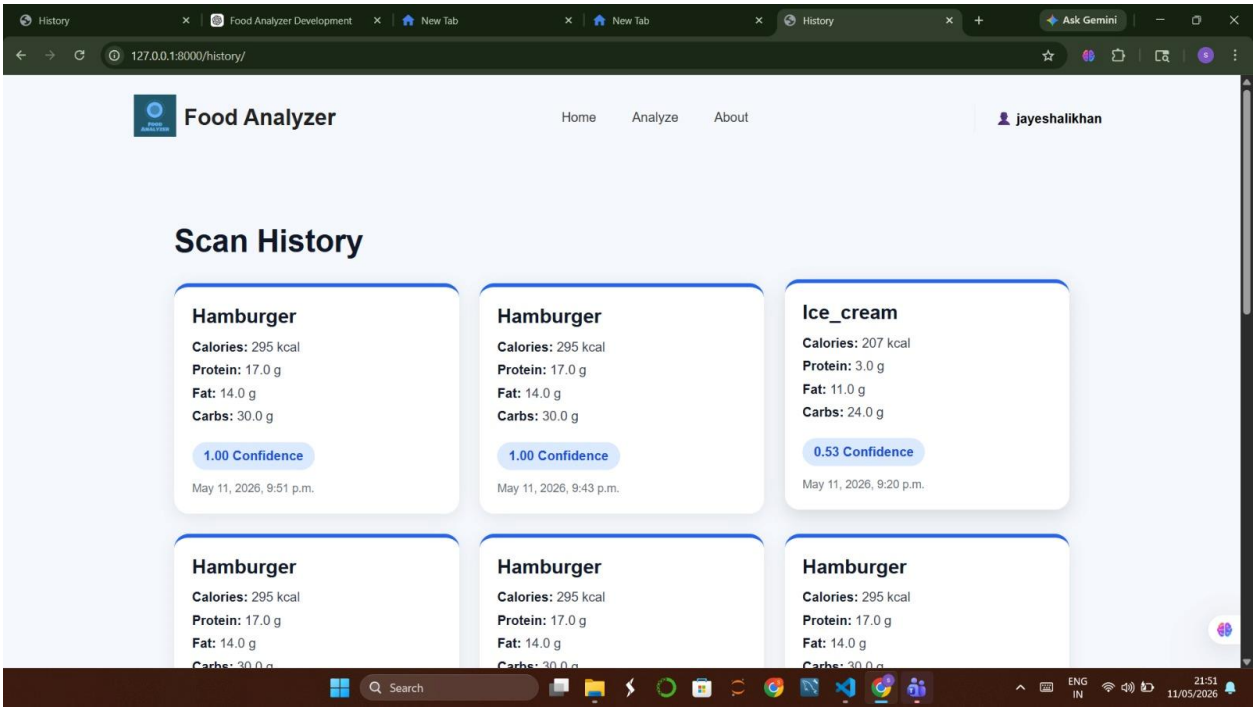


Fig. 6.4: User history tracking and dashboard analytics module.

The homepage provides navigational access to the food analysis module and user authentication. The login and signup pages implement Django's built-in authentication forms with Bootstrap styling and client-side validation. The food analysis page presents the image upload form alongside a live camera capture button implemented using the HTML5 getUserMedia API, with a JavaScript preview panel that

renders the selected image prior to submission. Upon form submission, the prediction result page displays the predicted food label, confidence score, and a structured nutritional breakdown comprising calories, protein, fat, and carbohydrates. The history dashboard aggregates all previous predictions for the authenticated user in a paginated table, with a summary statistics panel displaying total predictions, caloric trends, and the most frequently predicted food category. The responsive Bootstrap grid ensures consistent rendering across desktop, tablet, and mobile display resolutions, and all interaction flows were validated for usability through informal evaluation with target users.

14. DISCUSSION

The experimental results confirm that MobileNetV2 with two-phase transfer learning provides a compelling combination of accuracy, inference efficiency, and deployment practicality for food image classification. The test accuracy of 93.6% represents a substantial improvement over the traditional CNN baseline (85.2%) and exceeds all alternative architectures evaluated, validating the architecture selection. The narrow gap between training accuracy (95.2%) and test accuracy (93.6%) indicates effective regularisation through dropout and augmentation, with no evidence of severe overfitting.

A principal strength of the proposed system is its modular pipeline architecture. The clean separation of preprocessing, inference, database retrieval, and presentation concerns enables independent component upgrading—for instance, substituting MobileNetV2 with MobileNetV3 or a lightweight Vision Transformer—without requiring modifications to the surrounding Django application. The integration of user authentication, prediction history, and dashboard analytics substantially extends the utility of the system beyond a simple inference endpoint, supporting longitudinal dietary monitoring use cases.

The primary limitation of the current system is the absence of portion size estimation. Nutritional values are returned for standard serving sizes defined in the database, which may diverge substantially from actual consumed quantities, particularly for variable-portion foods such as pasta, rice, or salads. This limitation represents the most consequential source of nutritional estimation error. Additionally, the Food-101 dataset's Western-centric composition implies reduced accuracy for regional cuisines; while custom images were added for selected South Asian categories, comprehensive coverage of global food diversity would require substantially larger and more diverse training data. The system's classification confidence also degrades under poor lighting, heavy occlusion, or highly mixed-dish images, consistent with known failure modes in the food recognition literature.

From a scalability perspective, the Django application as currently deployed on a single server handles sequential inference requests adequately for small-scale usage. Production-scale deployment handling concurrent requests from multiple users would require horizontal scaling via containerisation (Docker) and load balancing, or cloud-native deployment on platforms providing auto-scaling compute resources.

15. CHALLENGES

Several technical challenges were encountered during the development and evaluation of the proposed system. Image quality variability—encompassing differences in resolution, dynamic range, blur, and white balance—presented the most pervasive challenge, as the model's training data does not uniformly represent the diversity of consumer food photography. Low-illumination images, in particular,

exhibited markedly lower classification confidence, suggesting the need for preprocessing enhancements such as histogram equalisation or learned low-light restoration.

Food appearance diversity within individual categories poses a fundamental classification challenge: a single dish category may encompass dozens of visually distinct presentations depending on cultural preparation method, plating style, and ingredient variation. The Food-101 dataset captures a subset of this diversity but cannot comprehensively represent the full space of real-world food presentations. Background noise—the presence of cutlery, tableware, hands, or surrounding surfaces—further complicates feature extraction, as the model may partially attend to non-food regions during inference. A food segmentation preprocessing step, isolating food regions prior to classification, would mitigate this issue but adds computational overhead.

Dataset imbalance across auxiliary custom-added categories introduced minor class-specific accuracy variation relative to the balanced Food-101 core, highlighting the sensitivity of transfer learning to training set composition. The construction and validation of the nutritional database required substantial manual curation effort to standardise serving size definitions across heterogeneous source databases, reflecting broader challenges in nutritional data interoperability.

16. FUTURE WORK

Multiple high-value directions for future development are identified. The most impactful near-term enhancement is the integration of monocular depth estimation or multi-view reconstruction to enable portion volume approximation, which would substantially improve caloric estimation accuracy beyond standard serving size assumptions. Depth-aware architectures employing auxiliary depth prediction heads trained jointly with the classification objective represent a promising approach.

Migration to a native mobile application using TensorFlow Lite for on-device inference would eliminate server dependency, reduce response latency, and enable offline operation—substantially broadening accessibility in regions with limited connectivity. Cloud-native deployment using containerisation and auto-scaling infrastructure would address concurrent request throughput limitations and enable production-scale usage.

The classification backbone may be upgraded to EfficientNet-B3, MobileNetV3, or a lightweight Vision Transformer (ViT-Tiny) to investigate accuracy improvements on the full 101-class taxonomy. Attention mechanisms and multi-scale feature aggregation strategies may further reduce confusion between visually similar categories. Personalised dietary recommendation, wherein the system cross-references a user's prediction history against WHO or USDA dietary guidelines and provides targeted nutritional guidance, represents a clinically meaningful extension of the current platform.

Finally, the expansion of the nutritional database and training corpus to encompass globally diverse food taxonomies—particularly South Asian, East Asian, Middle Eastern, and West African cuisines—would substantially improve the system's utility for non-Western populations and represent a meaningful contribution to dietary health equity.

17. CONCLUSION

This paper has presented a comprehensive end-to-end system for automated dietary assessment through image-based food classification and nutritional estimation, demonstrating the practical viability of deploying lightweight deep learning architectures within accessible web-based health monitoring applications. The proposed system, built upon a fine-tuned MobileNetV2 backbone integrated within a Django web application with SQLite nutritional database and responsive Bootstrap interface, achieves a test accuracy of 93.6%, precision of 92.9%, recall of 92.4%, and F1-score of 92.6% on the Food-101 benchmark—superior to all alternative architectures evaluated, including VGG16, ResNet50, and EfficientNetB0, while maintaining lower computational complexity and faster inference latency.

The system's modular architecture, encompassing image preprocessing, CNN inference, nutritional database retrieval, user authentication, prediction history tracking, and dashboard analytics, provides a complete dietary monitoring platform that substantially exceeds the functional scope of prior research prototypes. The work contributes evidence that deep learning—specifically transfer learning with computationally efficient architectures—can meaningfully transform dietary monitoring practice by eliminating dependence on manual self-reporting, increasing objectivity and consistency, and delivering actionable nutritional information within interactive response times.

As annotated food datasets grow in diversity and scale, and as model architectures continue to improve in accuracy-efficiency trade-offs, systems such as the one presented here will achieve broader generalisability and tighter nutritional estimation precision. The modular design of the proposed platform provides a robust foundation upon which future enhancements—including portion volume estimation, mobile deployment, cloud scaling, and personalised dietary recommendation—can be incrementally developed, validated, and deployed at scale.

REFERENCES

1. L. Bossard, M. Guillaumin, and L. Van Gool, “Food-101: Mining discriminative components with random forests,” in Proc. Eur. Conf. Comput. Vis. (ECCV), Zurich, Switzerland, 2014, pp. 446–461.
2. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Salt Lake City, UT, USA, 2018, pp. 4510–4520.
3. A. G. Howard et al., “MobileNets: Efficient convolutional neural networks for mobile vision applications,” arXiv preprint arXiv:1704.04861, 2017.
4. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Commun. ACM, vol. 60, no. 6, pp. 84–90, Jun. 2017.
5. C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, and Y. Ma, “DeepFood: Deep learning-based food image recognition for computer-aided dietary assessment,” in Int. Conf. Smart Homes Health Telematics, Springer, 2016, pp. 37–48.
6. P. Pandey, A. Deepthi, B. Mandal, and N. B. Puhan, “FoodNet: Recognizing foods using ensemble of deep networks,” IEEE Signal Process. Lett., vol. 24, no. 12, pp. 1758–1762, Dec. 2017.
7. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, 2016, pp. 770–778.

8. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learn. Represent. (ICLR), San Diego, CA, USA, 2015.
9. M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in Proc. Int. Conf. Mach. Learn. (ICML), Long Beach, CA, USA, 2019, pp. 6105–6114.
10. A. Howard et al., “Searching for MobileNetV3,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Seoul, South Korea, 2019, pp. 1314–1324.
11. N. Martinel, G. L. Foresti, and C. Micheloni, “Wide-slice residual networks for food recognition,” in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Lake Tahoe, NV, USA, 2018, pp. 567–576.
12. Y. Kawano and K. Yanai, “Automatic expansion of a food image dataset leveraging existing categories with domain adaptation,” in Proc. Eur. Conf. Comput. Vis. Workshops (ECCVW), 2014, pp. 3–17.
13. A. Meyers et al., “Im2Calories: Towards an automated mobile vision food diary,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Santiago, Chile, 2015, pp. 1233–1241.
14. C. Szegedy et al., “Going deeper with convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Boston, MA, USA, 2015, pp. 1–9.
15. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
16. S. Ruder, “An overview of multi-task learning in deep neural networks,” arXiv preprint arXiv:1706.05098, 2017.
17. F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, 2017, pp. 1251–1258.
18. R. V. Patil, G. M. Poddar, D. Y. Bhadane, R. M. Patil, S. R. Patil, S. V. Desale, and V. D. Suryawanshi, “Inception V3-Based Deep Learning Approach for Crack Detection in Paintings,” Int. J. Basic Appl. Sci., vol. 14, no. 4, pp. 756–762, Aug. 2025.
19. S. J., K. N. Mahajan, Y. B. Pawar, Y. D. Bhise, B. Jagdale, and R. V. Patil, “Plant Growth Analysis Using IoT and Reinforcement Learning Techniques for Controlled Environment,” Adv. Nonlinear Var. Inequal., vol. 27, no. 3, pp. 706–715, 2024.
20. S. Kakarwal, R. Mapari, P. P. Mane, M. P. Borawake, D. R. Dhotre, and R. V. Patil, “A Novel Approach for Detection, Segmentation, and Classification of Brain Tumors in MRI Images Using Neural Network and Special C Means Fuzzy Clustering Techniques,” Adv. Nonlinear Var. Inequal., vol. 27, no. 3, pp. 837–872, 2024.
21. G. M. Poddar, R. V. Patil, and S. Kumar N., “Approaches to Handle Data Imbalance Problem in Predictive Machine Learning Models: A Comprehensive Review,” Int. J. Intell. Syst. Appl. Eng., vol. 12, no. 21s, pp. 841–856, Mar. 2024.