NutriVision+: AI-Powered Real-Time Food & Nutrition Estimation

Build a lightweight, real-time, and fully automated food recognition and nutrition estimation system

SPECIAL GRATITUDE

We would like to thank Computer Vision Lab, ETH Zurich for opensourcing Food101 dataset without which this project would not be possible. And Googl colab for generous compute credits.

AUTHORS

Siddhant Rajhans
Madhura Girish
Gauthami Nonavinakere Prakash
Abhijith Viswanathan

About

- End to end food recognition system
- Using CNN & transfer learning trained on Food-101 dataset
- Detected food is mappedwith US Agriculture Nutrition Database
- Provides calories; macros & nutritional profile instantly with instantly

Motivation & Problem

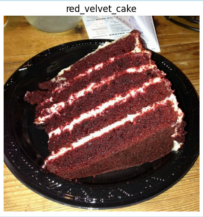
- Nutrition tracking is essential for health, fitness, and disease management.
- Manual food logging is slow, inconvenient, and often inaccurate.
- Smartphone cameras + AI now make automatic nutrition tracking possible.
- CNN-based food recognition can reduce user effort and improve tracking consistency.
- Integrating the USDA nutrition database enables reliable macro- and micro-nutrient estimates.
- Synthetic data generation (diffusion models) improves recognition accuracy for underrepresented foods.
- NutriVision+ aims to make nutrition monitoring real-time, effortless, and accessible to everyone.

Solution

- Traditional food-tracking apps rely heavily on manual input, which is time-consuming and leads to low user engagement.
- Estimating portions and identifying foods manually introduces significant inconsistencies in nutrition tracking.
- Many real-world foods have no packaging or barcodes, making automated identification difficult.
- The Food101 dataset is imbalanced, causing the model to struggle with certain underrepresented food categories.
- Current food-recognition approaches do not provide a seamless pipeline from image → detection → nutrition lookup.
- There is a need for a mobile system that can:

1. Identify food items from an image using EfficientNetB0-based transfer learning (with reasonable but not perfect accuracy)
2. Retrieve approximate nutritional information using the USDA database
3. Generate synthetic images with a diffusion model to help reduce dataset imbalance
4. Gradually improve performance through additional data and iterative retraining

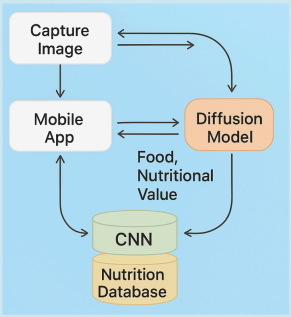
Additional motivation



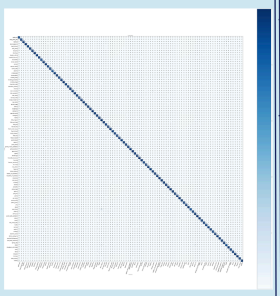
Nutritional breakdown (per slice, approximate)

- *Calories: 300–700*
- *Total Fat: 16–35 g*
- *Saturated Fat: 8–15 g (high percentage of daily value)*
- *Cholesterol: 20–60 mg (moderate percentage of daily value)*
- *Sodium: 240–570 mg (moderate to high percentage of daily value)*
- *Carbohydrates: 37–90 g*
- *Total Sugars: 29–70 g*
- *Dietary Fiber: 0–1 g (very low)*
- *Protein: 3–4 g (low)*

Layer / Module	Component Used	Role in the System	Input	Output
1. Image Acquisition Layer	Mobile Application (React + Expo)	Allows users to capture food images and initiates the prediction	Food image from camera/gallery	Image request sent to backend
2. Communication Layer	REST API Backend	Acts as the middleware between the mobile app, CNN model,	Image from mobile app	Cleaned image forwarded to CNN model
3. Food Detection Layer	CNN Food Detection Model	Performs automatic food recognition using a trained	Food image	Predicted food label
4. Nutrition Mapping Layer	Nutrition Database (US Agriculture Database)	Maps the detected food label to its nutritional values using	Predicted food label	Calories, carbohydrates, proteins, fats
5. Result Delivery Layer	REST API → Mobile App	Sends detected food name and nutritional values back to the user	Food label + Nutrition values	Displayed nutrition results
6. Data Augmentation Layer	Diffusion Model	Generates multiple synthetic food images using detected food + label	Food label + real image	New synthetic food images
7. Continuous Learning Layer	CNN Retraining Pipeline	Trains the CNN again using both real and synthetic images to	Expanded dataset (real + synthetic)	Updated higher-accuracy CNN model



Confusion Matrix(for fun!)



Components

- Mobile App (React Expo) – Captures food image and sends it to backend
- REST API Backend – Handles requests, preprocesses image, returns predictions
- Image Preprocessing Module – Resizes and normalizes image to 224×224×3
- CNN Food Detection Model – EfficientNetB0 classifier predicts food label
- Nutrition Mapping Service – Fetches calories and macronutrients from database
- Diffusion Model for Image Generation – Produces synthetic images for augmentation
- Augmented Dataset Storage – Stores real and generated images with metadata
- Model Retraining Pipeline – Periodically retrains model with expanded dataset
- Prediction & Feedback Logging – Saves outputs, timestamps, and user inputs

Frontend

- React Native (Expo) App
- Captures food image (camera / gallery).
- Sends image → backend REST API.
- Displays: predicted food label + nutrition values.
- Optionally sends feedback (correct / incorrect label).

Backend REST API Server

- Receives image from app.
- Preprocesses image to (224, 224, 3).
- Calls CNN Detection Service.
- Uses predicted label to query Nutrition Database.
- Returns JSON: {food_label, calories, carbs, protein, fat, ...}.
- Logs request, prediction, and (optional) user feedback to DB.

CNN Detection Service

- Input Layer: accepts resized food image (224, 224, 3).
- Transfer learning using EfficientNetB0: feature extractor trained on ImageNet.
- GlobalAveragePooling2D: converts feature maps → feature vector.
- Dense(len(class_names)): logits for Food-101 classes.
- softmax activation: probability distribution over food classes.

Nutrition Mapping Layer

- Underlying data: US Agriculture / USDA Nutrition Database.
- Table: food_label → {calories, carbs, protein, fat, etc.}.
- Backend queries using CNN's predicted food_label.
- Returns standardized nutrition profile for that food.

Diffusion Model Pipeline (Image Generation)

- VAE Encoder: Compresses Food-101 images into latent space
- U-Net Denoiser: Trained specifically on Food-101 to remove noise and learn food textures, colors, and shapes
- Class Conditioning (Food Labels): Guides generation using Food-101 food categories
- VAE Decoder: Reconstructs realistic food images from denoised latent space

Food-101 Image → VAE Encode → U-Net Denoising (Class-Guided) → VAE Decode → Synthetic Food Image

Data & Model Storage

- Prediction Logs DB: user id, image id, food label, nutrition values, timestamp, feedback.
- Training Dataset Store: original Food-101 + synthetic images + labels.
- Model Registry: versioned CNN models (v1, v2, ...) and diffusion checkpoints.

Training & Retraining Pipeline

Periodically:

- Load real + diffusion-generated images from dataset store.
- Train / fine-tune the CNN model (code above).
- Evaluate on validation set.
- If improved → push new model version to Model Registry.

Backend reloads latest production model for inference.