



# NutriVision: AI-Powered Food Recognition and Nutritional Analysis

by Malhar Sham Ghogare and Shubham Gaur

Our platform combines AI food recognition with personalized dietary guidance.

# Dataset Overview



## Food-101 Dataset

101,000 images across 101 food categories with balanced, realistic images



## USDA FoodData Central

1,841,897 entries with comprehensive nutritional information

# Dataset Overview

## FOOD 101:

Index	Food Class
1	Apple Pie
2	Baby Back Ribs
3	Baklava
4	Beef Carpaccio
5	Beef Tartare
6	Beet Salad
7	Beignets
8	Bibimbap
9	Bread Pudding
10	Breakfast Burrito

- 101 food categories with 101,000 images (1,000 per class)
- Ranges from desserts (Apple Pie) to protein dishes (Baby Back Ribs)
- Balanced class representation
- Dataset serves as comprehensive training foundation for our classification model

## USDA FoodData Central DB:

- USDA FoodData Central database: Contains over 1.8 million entries
- Structured data includes: Unique FDC\_ID identifiers, food descriptions, serving sizes, and key nutritional metrics (calories, protein, fat)
- Wide nutritional variation: From high-calorie oils (867 kcal) to low-calorie broths (4 kcal)
- Database enables detailed nutritional analysis for ideally any identified food item

FDC_ID	Description	Serving Size	Calories	Protein	Fat
1105904	WESSON Oil 1 GAL	15 ml	867	0	93.33
1105905	Swanson Beef Broth	240 ml	4	0.83	0
1105906	Clam Chowder Soup	440 g	82	2.45	5.31
1105907	Cheese Broccoli Soup	440 g	82	1.22	6.12
1105908	Swanson Chicken Broth	240 ml	4	0.83	0
1105909	Bean and Ham Soup	412 g	61	3.67	0.61

# Exploratory Analysis

## Image Preprocessing

- Resizing and normalization
- Data augmentation techniques
- Class distribution visualization

## USDA Data Exploration

- Nutritional value distributions
- Food category breakdown
- Missing value handling

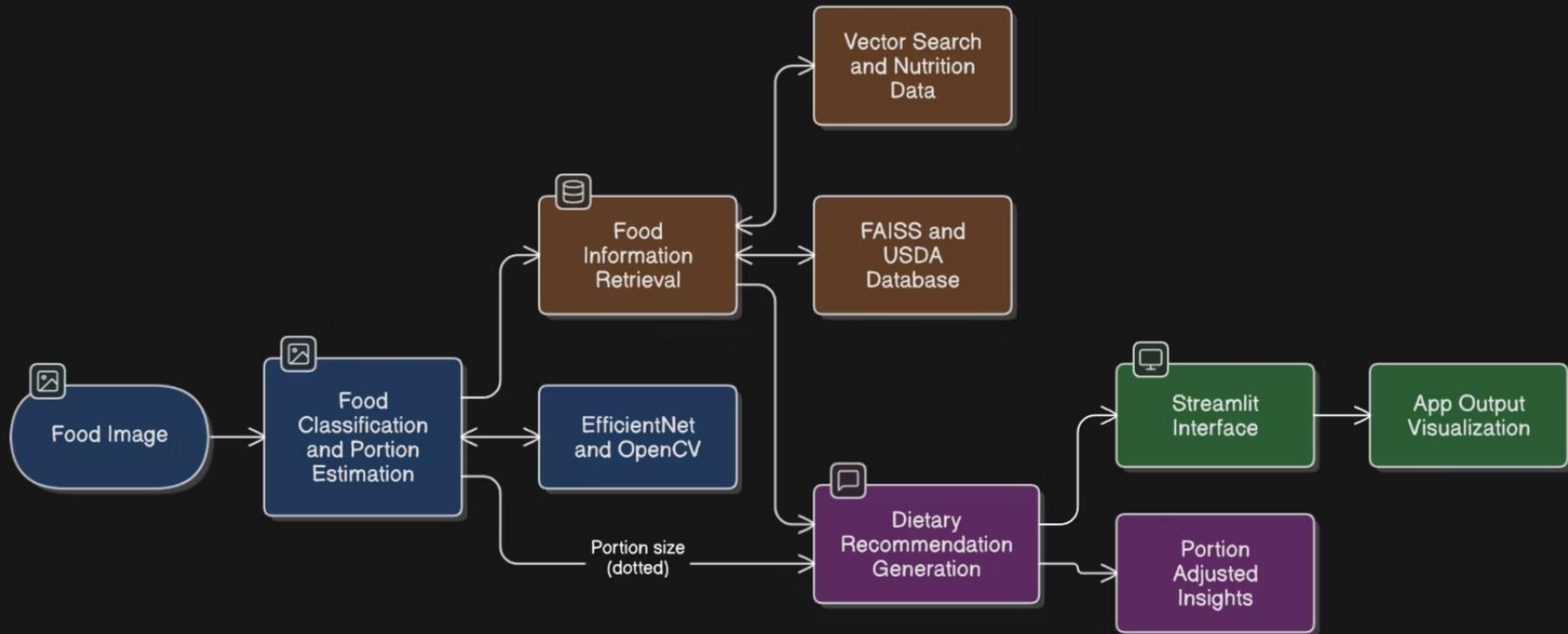
## Vector Embeddings

- Food description embeddings
- Clustering similar foods



# System Architecture

NutriVision Architecture



# System Architecture



## Modular Code Structure

app.py, vector\_db.py



## Component Interactions

User uploads food image via Streamlit UI

EfficientNet Model processes image and returns predicted food class

vector\_db.py finds semantically similar foods using FAISS

app.py queries USDA database for nutritional information

generates personalized dietary insights using OpenAI/Anthropic APIs

Results displayed to user through Streamlit interface



## Key Features

Vector-based RAG implementation

Pre-trained EfficientNet model fine-tuned on Food-101 dataset

FAISS index for efficient similarity search

Integration with generative AI for personalized recommendations



## Dependencies

Core ML: Tensorflow, Keras, FAISS

Embedding: Sentence Transformers

Data: Food 101, USDA FoodData Central - UI: Streamlit

AI: OpenAI GPT-4o, Anthropic Claude-3

# Data Processing & Feature Engineering Overview:

## **Multi-Modal Data Pipeline:**

- Image preprocessing for food recognition (224×224 RGB normalization)
- Text processing for food descriptions and embeddings
- USDA nutritional database integration (1.8M+ entries)

## **Food Classification Labels:**

- 101 food categories from Food-101 dataset (Apple Pie → Breakfast Burrito)
- Labels mapped to predicted class

## **Vector Embeddings for Semantic Food Search:**

- 384-dimensional semantic representations using SentenceTransformer
- Local FAISS index for efficient similarity search
- Streamlit caching for performance optimization

# Computer Vision Portion Size Estimation

## How It Works:

**Original Food → Processed Binary Image → Contour Detection → Area Calculation**

- **Computer Vision Pipeline:** Converts images to grayscale → enhances contrast → creates binary mask → detects food contours
- **Size Calculation:** Compares detected food area to reference database of average portion sizes
- **Nutritional Adjustment:** Automatically scales nutritional values based on detected portion size (1.0-3.0x)

**Example:** A cheeseburger detected as 1.4x average size → 40% increase in calculated calories, protein, fat, and carbs → more precise dietary guidance



# Modelling



## EfficientNetB0

Pre-trained on ImageNet, fine-tuned on Food-101

Layer (Type)	Output Shape	Param #
EfficientNetB0 (Functional)	(None, 7, 7, 1280)	4,049,571
GlobalAveragePooling2D	(None, 1280)	0
Dropout	(None, 1280)	0
Dense	(None, 1024)	1,311,744
Dropout	(None, 1024)	0
Dense	(None, 101)	103,525
Total Parameters		8,295,380 (31.64 MB)
Trainable Parameters		1,415,269 (5.40 MB)
Non-Trainable Parameters		4,049,971 (15.45 MB)
Optimizer Parameters		2,830,540 (10.80 MB)

Table 3: Model parameters for the NutriVision EfficientNetB0-based model.

### Transfer Learning Strategy

- **Feature Extraction:** EfficientNetB0 without its top layer (include\_top=False) serves as a pre-trained feature extractor that already understands visual patterns, textures, and shapes from training on ImageNet.
- **Task-Specific Classification:** The custom classification head (GlobalAveragePooling2D → BatchNormalization → Dropout → Dense) is specifically designed for 101 food classes.

```
base_model = EfficientNetB0(weights="imagenet", include_top=False,
input_shape=(224, 224, 3))
base_model.trainable = True
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    BatchNormalization(),
    Dropout(0.3),
    Dense(101, activation="softmax", kernel_regularizer="l2")
])
```

- GlobalAveragePooling2D() flattens EfficientNet’s output
- Dense(101, softmax) replaces the original classifier with your 101 food classes

# Initial Experimentation

## Baseline Model

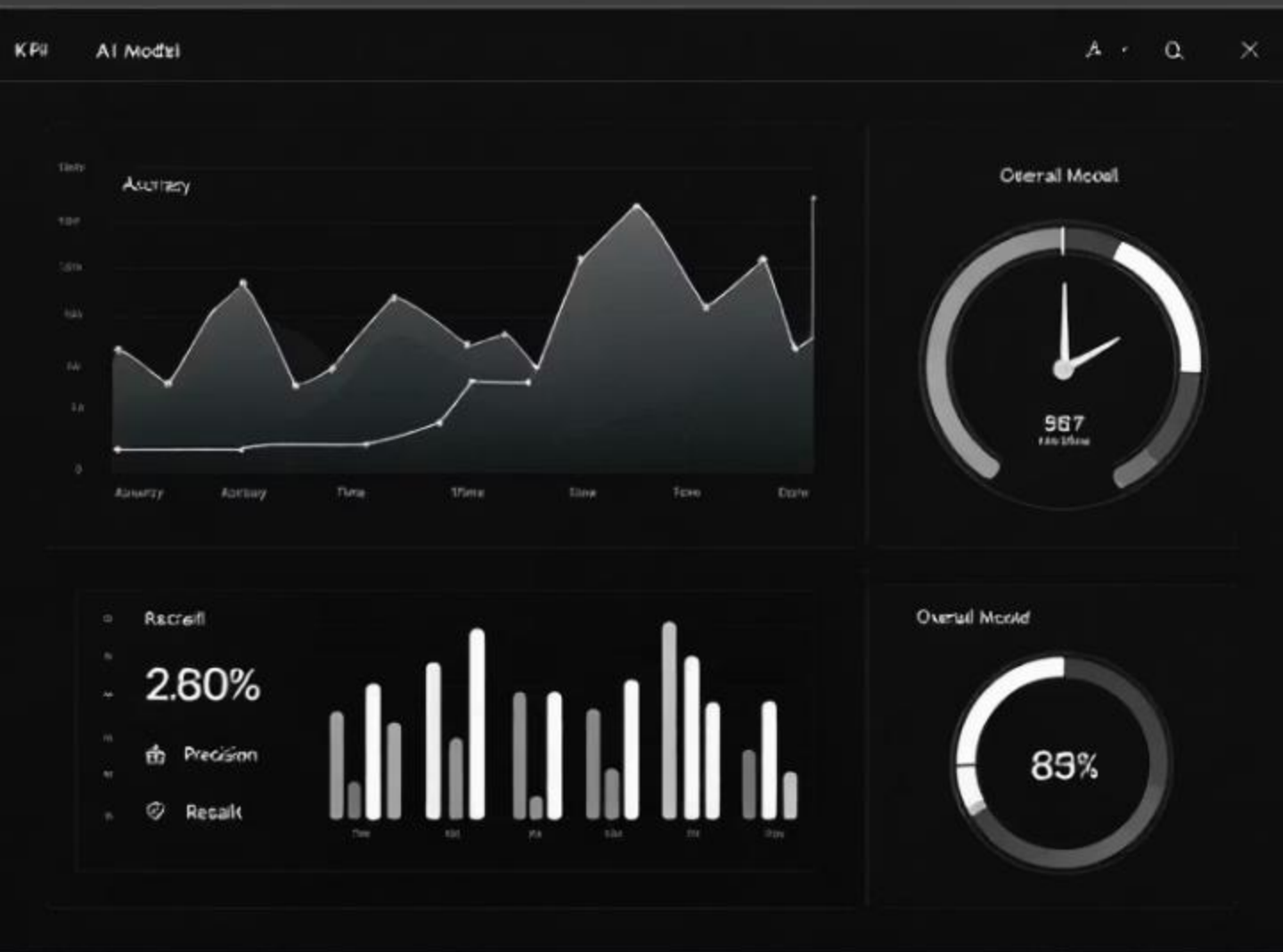
- EfficientNetB0
- Learning rate: 0.0003
- 30 epochs with dropout optimization

## Vector Search

- FAISS index
- Simple text embedding approach

## Generative AI

- Structured prompt templates
- GPT-4o vs Claude-3 comparison



# Model Evaluation

82%

Validation Accuracy

Final classification model  
performance

84%

Training Accuracy

Model learning effectiveness

# Generating AI Insights

## AI-Powered Dietary Insights Process:

1. Food classification from image (EfficientNet)
1. Nutritional data retrieval (USDA database)
1. Context formation combining food class & nutritional data
1. Prompt engineering for nutritional guidance
1. AI-generated personalized dietary recommendations

```
def generate_dietary_insight(nutrition_text,
                             model_choice="GPT-4o"):
    """Generate dietary insights using generative AI"""
    try:
        if model_choice == "GPT-4o":
            # Generate with OpenAI GPT-4o
            response = openai.ChatCompletion.create(
                model="gpt-4o",
                messages=[
                    {"role": "system", "content": "You are a helpful nutrition assistant focusing on providing concise, actionable dietary advice."},
                    {"role": "user", "content": f"Based on this data: {nutrition_text}, provide a concise, actionable dietary advice that's personalized and specific."}
                ]
            )
            return
    response['choices'][0]['message']['content'], "Generated by GPT-4o"
```

# Final Results

## **Improved Classification**

EfficientNetB7 with fine-tuned last 50 layers, advanced augmentation, early stopping

## **Enhanced RAG Implementation**

Streamlined vector search with efficient caching

Local FAISS index for fast similarity retrieval

Duplicate removal while preserving result order

## **Refined AI Prompting**

Structured system prompts for personalized dietary recommendations

## **UI Optimization**

Responsive Streamlit layout with progress indicators and error handling

# Demo



## Setup

Python 3.9+, API keys, sample food images



## Image Upload

~0.5 seconds for preprocessing, ~1-2 seconds for inference



## Search & Analysis



## AI Recommendations



# Future Work

Advanced Models  
Vision Transformers integration and  
ensemble approaches

New Features  
Multi-food detection, meal planning,  
environmental impact assessment



Optimization  
Model compression and edge device  
deployment

User Experience  
Preferences, dietary restrictions, and  
history tracking