

# Project Report

## AI Nutrition Assistant

Pallavi Talluri

A report submitted in part fulfilment of the certificate of  
**Artificial Intelligence Programming Assistance**  
**(2024-2025)**

**Guidance:** Mr. Sudip Kundu



**NSTIW Vidyanagar, Hyderabad**

**Date: 01/07/2025**

---

## Abstract

This project introduces an AI-powered system titled “ **AI Nutrition Assistant**”, which automates the process of identifying food items and delivering accurate nutritional data in real-time. It operates in two modes—text and image—enabling users to either type the name of a food item or upload its photograph. The system utilizes the **MobileNetV2** deep learning model for image classification and integrates the **USDA FoodData Central API** for retrieving nutrition facts. By minimizing manual effort and providing instant feedback, this assistant supports users in making healthier dietary choices through the power of artificial intelligence.

## Acknowledgement

I sincerely thank **Mr. Sudip kundu**, my project mentor, for his guidance, encouragement, and expert insights throughout this project. I am also grateful to **NSTIW Vidyanagar, Hyderabad** for providing the opportunity and the necessary infrastructure for conducting this work. This project would not have been possible without their continuous support.

## Table of content

Abstract .....	2
Acknowledgement .....	3
Table of content .....	4
Problem Statement .....	5
Literature Review .....	5
Proposed Solution .....	5
Requirements.....	6
Algorithms Used.....	6
Dataset Description.....	7
Data Preprocessing .....	7
Exploratory Data Analysis (EDA) .....	7
Model Building.....	8
Model Evaluation .....	8
Results and Discussion .....	9
Challenges Faced.....	11
Conclusions and Future Work.....	11
References.....	12
Appendix .....	13

## Problem Statement

Tracking daily nutritional intake is a challenge for many individuals in today's fast-paced environment. Existing tools and mobile applications often require users to input data manually, which is time-consuming, inconvenient, and prone to error. Additionally, these tools lack smart automation features like image recognition or live feedback. This project addresses these limitations by developing a smart AI assistant that can automatically recognize food items from text or images and return relevant nutrition details using real-time data sources.

## Literature Review

Numerous nutrition-tracking applications, such as **MyFitnessPal**, **Yazio**, and **HealthifyMe**, assist users in monitoring their diet. However, most of these rely heavily on manual data entry, which limits user experience. With advancements in **computer vision** and **deep learning**, particularly **convolutional neural networks (CNNs)**, food recognition through images has become feasible. Furthermore, publicly available APIs like the **USDA FoodData Central API** now offer precise nutritional data. This project integrates these technologies to build a practical and intelligent nutrition assistant.

## Proposed Solution

The solution is a hybrid system that combines AI-based image recognition with public nutrition databases. It consists of:

- **Text Input Mode:** Users manually enter food names; the system queries the USDA API.
- **Image Input Mode:** The MobileNetV2 model processes the uploaded image to identify food items.
- **Top-3 Prediction Display:** The model displays the top three predicted classes with confidence scores.
- **Nutritional Analysis:** The top predicted item is sent to the USDA API to retrieve and present key nutrient values.

This solution offers flexibility, real-time insights, and a more engaging user experience.

---

## Requirements

### a. Technology Stack:

- Python 3.10+
- TensorFlow and Keras
- NumPy
- Requests (for API interaction)
- USDA FoodData Central API
- Keras Image Utilities

### b. Hardware:

- A modern computer system with at least 4GB RAM

### c. Software:

- Jupyter Notebook or google collab

### d. Deployment Environment:

- Local machine for development
- Cloud or web hosting for public deployment (optional)

## Algorithms Used

This project utilizes the **MobileNetV2** model, a lightweight, efficient convolutional neural network optimized for mobile devices.

### Reasons for choosing MobileNetV2:

- Pre-trained on the **ImageNet** dataset with 1,000 classes
- Small memory footprint with high prediction speed
- Effective for real-world image classification tasks

No custom training is performed. The top prediction from MobileNetV2 is used as input to the USDA API to obtain nutritional information.

## Dataset Description

Unlike conventional machine learning projects, this system does not rely on a static dataset.

- **Image Classification Source:** The MobileNetV2 model pre-trained on **ImageNet** handles food image recognition.
- **Nutritional Information Source:** The **USDA FoodData Central API** provides real-time food nutrient data.

This architecture allows the application to remain light and up-to-date without storing or maintaining large datasets locally.

## Data Preprocessing

### For image inputs:

- Images are resized to **224x224 pixels**.
- They are converted into **NumPy arrays** and normalized using `preprocess_input()`.
- A **batch dimension** is added to prepare the data for model prediction.

### For text inputs:

- No preprocessing is required.
- The input string is directly embedded into the USDA API request URL.

These steps ensure compatibility with both the model and API.

## Exploratory Data Analysis (EDA)

Since the USDA API returns **live data**, traditional EDA on static datasets is not applicable. However, sample responses were reviewed to ensure consistency.

- Common nutrients retrieved: **Energy (kcal), Protein, Total Fat, Carbohydrates, Fiber**
- Data structure is JSON-based, consistent across queries
- Example queries like “banana” and “chicken” show full nutrient profiles

## Model Building

There is no training phase in this project.

### Steps:

- Load **MobileNetV2** from TensorFlow with weights='imagined'
- Preprocess uploaded food image
- Use `decode_predictions()` to extract top 3 likely classes
- Choose the top prediction with confidence
- If confidence < 30%, alert the user to upload a clearer image

This use of a pre-trained model enables fast deployment and inference.

## Model Evaluation

No training = no typical evaluation metrics. However, the following methods are used:

- **Confidence Scoring:** Display model confidence % for each prediction.
- **Manual Validation:** Confirm predictions visually with known food images.
- **Functional Testing:** Cross-check whether recognized food items return correct nutrient data.

The goal is usability and correctness rather than numerical metrics.



---

## Results and Discussion

### Was the prediction accurate?

Yes, the prediction was largely accurate for commonly known food items.

- In **text input mode**, queries such as "banana", "rice", and "chicken" produced precise and relevant nutritional data from the USDA API.
- In **image input mode**, photographs of items like "pizza", "apple", and "boiled egg" were correctly identified with **confidence scores above 80%** in most cases.

However, accuracy dropped when:

- The input image was poorly lit, blurred, or contained multiple overlapping items.
- The food item was not part of the MobileNetV2 model's training set (e.g., regional or Indian dishes).

### Which features were most important?

In the image recognition component:

- **Visual texture, shape, and color** of the food item played key roles in determining the prediction.
- The **top-3 predicted labels** with their confidence scores gave insight into how confident the model was in its output.
- The **confidence threshold** ( $\geq 30\%$ ) was critical in filtering out low-reliability predictions.

In the nutritional data component:

- The food **description** matched from the USDA database determined which nutrient breakdown was returned.
- Among the nutritional values, the most emphasized features were:
  - **Energy (kcal)**
  - **Protein (g)**
  - **Total Fat (g)**
  - **Carbohydrates (g)**
  - **Fiber (g)**

### Any surprising observations?

Yes, a few surprising insights emerged during testing:

- The model **misclassified Indian foods** like "idli", "biryani", and "samosa" due to the **ImageNet dataset's Western bias**.
- Occasionally, foods like "laddu" were predicted as "meatball" or "cheeseburger" based on visual similarity.
- Surprisingly, the system was **able to distinguish between visually similar items** like "apple" and "tomato" quite accurately, indicating strong model sensitivity to subtle visual cues.
- The USDA API sometimes returned **multiple entries** for a single food name (e.g., "chicken" as raw, cooked, grilled), which required selecting the most appropriate match.

These observations highlight the need for:

- Training on **localized datasets**
- Implementing **user feedback or confirmation** loops
- Future integration of **contextual filtering** (e.g., raw vs. cooked, processed vs. fresh)

## Challenges Faced

- **Recognition of Indian/regional foods** was limited due to ImageNet's Western-focused training set.
- **Image quality** (blur, lighting) significantly affected prediction accuracy.
- **API limitations** like network dependency and rate limits required handling fallback scenarios.
- **No offline capability** meant users needed constant internet access.

These issues will inform improvements in the next phase.

## Conclusions and Future Work

This project successfully demonstrated how artificial intelligence, when combined with real-time data access, can simplify nutrition tracking for users. The system was able to accurately classify food images and retrieve nutritional information with minimal user effort. By supporting both text and image inputs, it offers flexible interaction and practical use in daily scenarios.

### What Worked Well

- **MobileNetV2** effectively recognized common food items in well-lit and clear images.
- The **USDA FoodData Central API** provided reliable, structured, and detailed nutrient information.
- The dual-mode input system (text + image) allowed users to interact with the assistant based on their preference.
- The modular code design ensured smooth integration of image processing, API communication, and user interaction.

### What Needs Improvement

- The model showed **limited accuracy with regional or culturally specific foods**, such as Indian dishes, due to the limitations of the ImageNet training set.
- **Image quality** (blur, poor lighting) directly impacted prediction confidence and accuracy.

- There is **no offline functionality**—the system is entirely dependent on internet availability to access the API.
- The system currently lacks **voice input** and does not provide **personalized recommendations**.

## Future Ideas

- **Train a custom CNN model** using a dataset of Indian and regional foods to improve recognition accuracy for a diverse user base.
- **Experiment with newer algorithms** like EfficientNet, ConvNeXt, or transformers for food classification.
- **Collect and build a larger image dataset** to fine-tune the model for local and diverse cuisines.
- **Integrate voice-based interaction** using libraries like SpeechRecognition or Whisper.
- **Deploy the solution as a mobile or web application** using frameworks like Streamlit, Flask, or React Native for broader accessibility.
- **Add personalized features** like meal planning, calorie goals, and diet suggestions based on user profiles.

In conclusion, the Smartest AI Nutrition Assistant lays a strong foundation for AI-based dietary support tools. With further enhancements, it can evolve into a comprehensive health companion for users across the globe.

## References

- USDA FoodData Central API: <https://fdc.nal.usda.gov/>
- TensorFlow MobileNetV2 Documentation
- Keras Image Utilities

## Appendix

### Key Code Snippets

Below are selected core code snippets that demonstrate the core functionality of the Smartest AI Nutrition Assistant system.

#### Import necessary libraries:

```
import requests
import numpy as np
from tensorflow.keras.applications.mobilenet_v2
import MobileNetV2, preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image
```

#### Loading the Pre-trained Model:

```
mobilenet_model = MobileNetV2(weights='imagenet')
```

#### Fetching Nutritional Information from USDA API:

```
def get_food_nutrition(food_item):
    api_key = "N7Lwn54DklixpmXQ6N3R3mCi8QabqpVWLjQgfw6C" # Replace with
    your USDA API key
    url =
    f"https://api.nal.usda.gov/fdc/v1/foods/search?query={food_item}&api_key=
    {api_key}"
    try:
        response = requests.get(url).json()
        food_name = response['foods'][0]['description']
        nutrients = response['foods'][0]['foodNutrients']
        top_nutrients = "\n".join([f"{n['nutrientName']}: {n['value']}"
        {n['unitName']}]" for n in nutrients[:5]])
        return food_name, top_nutrients
    except:
        return "No data found", "Nutrition details unavailable."
```

## Function to Predict Food from Image:

```
def image_food_detector(image_path):
    try:
        img = image.load_img(image_path, target_size=(224, 224))
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array = preprocess_input(img_array)

        predictions = mobilenet_model.predict(img_array)
        decoded_predictions = decode_predictions(predictions, top=3)[0]

        print("\n📷 Top predictions:")
        for i, pred in enumerate(decoded_predictions):
            print(f"{i+1}. {pred[1]} ({pred[2]*100:.2f}%)")

        best_pred = decoded_predictions[0]
        if best_pred[2] < 0.30:
            print("⚠️ Low confidence. Consider uploading a clearer image.")
            return best_pred[1]

    except Exception as e:
        print(f"❌ Error processing image: {e}")
        return "Unknown"
```

## Main Program:

```
def run_nutrition_assistant():
    print("\n🔍 Modes available: text / image")
    mode = input("Choose input mode: ").strip().lower()

    if mode == "text":
        food = input("Enter food item: ")
    elif mode == "image":
        image_path = input("Enter image path (e.g., mango.jpg): ")
        food = image_food_detector(image_path)
    else:
        print("❌ Invalid mode")
        return

    food_name, nutrition_info = get_food_nutrition(food)


    print("\n🍌 Food:", food_name)
    print("📋 Nutrition Info:\n", nutrition_info)
```

## Run the App:

```
if __name__ == "__main__":  
    run_nutrition_assistant()
```


## Sample Output - Nutrient Distribution


### Text

 Modes available: text / image

Choose input mode: text

Enter food item: chicken

 Food: CHICKEN

 Nutrition Info:

Protein: 21.4 G


Total lipid (fat): 1.79 G

Carbohydrate, by difference: 0.0 G

Energy: 107 KCAL

Total Sugars: 0.0 G


### Image

 Modes available: text / image

Choose input mode: image

Enter image path (e.g., mango.jpg): pizza.jpg

1/1  2s 2s/step

 Top predictions:

1. pizza (98.01%)

2. bagel (0.16%)

3. potpie (0.12%)

 Food: PIZZA

 Nutrition Info:


Calcium, Ca: 43.0 MG

Iron, Fe: 2.19 MG

Vitamin A, IU: 679 IU


Vitamin C, total ascorbic acid: 0.4 MG


Protein: 10.7 G


 Modes available: text / image

Choose input mode: text

Enter food item: rasam

 Food: No data found

 Nutrition Info:  
Nutrition details unavailable.

 Modes available: text / image


Choose input mode: image


Enter image path (e.g., mango.jpg): Idli-and-Sambar.jpg

1/1  1s 1s/step

 Top predictions:

1. soup\_bowl (45.38%)
2. plate (17.60%)
3. consomme (7.42%)

 Food: No data found

 Nutrition Info:  
Nutrition details unavailable.

### Input images

