International Journal of Scientific Research and Engineering Development-– Volume 8 Issue X, 2026

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

AI EDGE NUTRITION SCANNER

Bhavya D
(COMPUTER SCIENCE & ENGINEERING, EAST WEST INSTITUTE OF TECHNOLOGY, and Bangalore
Email:bhavya.d1511@gmail.com)

----------**********----------

ABSTRACT:

The ai edge nutrition scanner is an intelligent web-based application designed to automate food recognition and nutritional estimation from images. The system addresses the limitations of manual nutrition tracking, including the need to search food items, consult nutrition charts, and calculate portion-dependent values. The proposed solution accepts an uploaded image or live camera input and uses computer vision and deep learning to identify the food category. The report describes a transfer learning approach based on mobilenetv2 with the fruits-360 dataset, while the implemented application also documents yolo world and openclip components for food recognition. After recognition, the backend retrieves nutrition values from a structured nutrition.csv dataset and scales calories, protein, carbohydrates, fat, fibre, and sugar according to the user-entered weight. A documented banana test at 80 g achieved a 98% confidence score with calculated nutritional values, demonstrating the feasibility of the proposed workflow. The modular architecture is intended to support future multi-food detection, tensorflow lite deployment, mobile applications, portion estimation, barcode recognition, and personalized dietary recommendations.

Keywords—artificial intelligence, food recognition, nutrition estimation, computer vision, deep learning, mobilenetv2, yolo world, openclip, react, flask, firebase, edge computing

----------**********----------

I. INTRODUCTION

Nutrition monitoring is important for maintaining healthy dietary habits, yet estimating the nutritional content of everyday food can be inconvenient. Conventional approaches require users to identify food manually, search nutritional charts, estimate portion sizes, and perform calculations. Such workflows increase user effort and may introduce calculation errors. The project report identifies these limitations as a motivation for an automated image-based nutrition estimation system.

Recent progress in artificial intelligence, computer vision, and deep learning enables systems to recognize visual objects and connect predictions with structured information. In food-related applications, image recognition can reduce manual searching and provide a faster interface for dietary assessment. However, recognition performance can be affected by illumination, background, viewpoint, image quality, and occlusion. Cloud-based commercial services may also introduce subscription cost, internet dependency, and restrictions on customization.

The AI Edge Nutrition Scanner combines food image recognition with nutrition analysis in a unified web application. The user uploads or captures a food image, supplies the approximate weight, and receives a predicted food category together with nutritional values. The report describes a modular architecture involving a React frontend, Flask REST API backend, deep learning inference, a structured nutrition dataset, and Firebase-based scan-history storage.

The main objectives are to automate food recognition, apply transfer learning for image

classification, estimate nutrition according to user-specified weight, reduce dependence on paid recognition APIs, and provide an architecture that can be extended toward edge deployment.

II. RELATED WORK

The project report surveys research directions that support the proposed system. DeepFood demonstrated the application of convolutional neural networks to food image recognition and computer-aided dietary assessment, establishing the usefulness of deep learning for reducing manual dietary analysis. The Food-101 work provided a large benchmark for food image classification and highlighted the importance of representative image datasets for computer vision systems.

MobileNetV2 introduced an efficient convolutional architecture based on inverted residuals and linear bottlenecks. Its relatively lightweight design makes it appropriate for applications where computational efficiency is important, including mobile and edge-oriented scenarios. EfficientNet further demonstrated that model depth, width, and input resolution can be jointly scaled to obtain strong accuracy with efficient resource use.

Transfer learning is another important foundation. Pan and Yang described how knowledge learned from a source task can be transferred to a related target task, reducing training requirements when task-specific data are limited. AlexNet demonstrated the effectiveness of deep convolutional representations for large-scale image classification and helped establish the foundation for later CNN architectures.

The Fruits-360 dataset is used in the project as a source of labeled fruit images for model development. The report also cites work on computer-vision-based food recognition and nutrition estimation, reflecting the growing integration of visual recognition with nutritional databases. Building on these directions, the proposed system focuses on a practical end-to-end workflow rather than recognition alone: image acquisition, preprocessing, prediction, nutrition retrieval, weight-based calculation, result visualization, and scan-history storage.

III. PROPOSED SYSTEM

The proposed system is an integrated food recognition and nutrition estimation platform. It contains three principal functional layers: the user interface, the backend service, and the AI/nutrition processing layer. The user interface accepts an uploaded image or webcam capture and the food weight. The backend validates the request, preprocesses the image, invokes the AI model, accesses the nutrition dataset, performs calculations, and returns a JSON response. The frontend then presents the food name, confidence score, and nutritional values.

A. User Interface Layer

The React-based interface provides image upload, camera capture, weight input, result visualization, and scan-history access. The design emphasizes a simple workflow so that users do not need extensive nutritional knowledge. Supported image formats include common JPG, JPEG, and PNG files. The system also validates user inputs before prediction.

B. Backend Layer

The Flask REST API acts as the integration layer between the interface, AI model, and nutrition dataset. It validates image input and performs preprocessing operations such as resizing and normalization. After inference, it retrieves nutrition values from nutrition.csv and calculates quantity-specific values. The backend sends the result to the React frontend in structured JSON form.

C. AI Recognition Layer

The report documents two related descriptions of the recognition component. The implementation methodology identifies MobileNetV2 with TensorFlow/Keras and transfer learning as the deep-learning classifier trained using Fruits-360. The abstract and results sections additionally describe

YOLO World and OpenCLIP with PyTorch/OpenCV as recognition components. To preserve the project report faithfully, this paper records both descriptions rather than silently choosing between them.

D. Nutrition Analysis

The nutrition dataset stores values per 100 g for supported food categories. For a user-entered weight W, the system scales each nutrient from its per-100-g reference value N100 using:

$$N(W) = N100 \times W / 100.$$

The same calculation is applied to calories, protein, carbohydrates, fat, fibre, and sugar. This produces a quantity-specific nutritional estimate while keeping the dataset representation simple.

E. Scan History

Firebase is used to retain previous scan results. The history records allow users to review earlier food names, analyzed weights, calories, and confidence scores. This feature extends the system from a one-time recognition tool to a basic dietary monitoring application.

IV.SYSTEM ARCHITECTURE

The system workflow begins with image acquisition. The user either uploads a food image or captures one through a webcam and enters its weight. The React frontend validates the request and sends it to the Flask backend through a REST API. The backend validates and preprocesses the image before passing it to the trained recognition model. The model returns the predicted food category and a confidence score. The backend then searches nutrition.csv for the recognized food category, scales the per-100-g values using the entered weight, and constructs the final response. The frontend displays the result and stores relevant scan information for later review.

The logical data flow can be summarized as:

User → React Frontend → Flask REST API → Image Preprocessing → AI Model → Predicted Food + Confidence → nutrition.csv → Weight-Based Calculation → JSON Response → React Result View → Firebase Scan History.

The modular architecture separates presentation, service logic, AI inference, and nutritional data. This separation improves maintainability and allows future replacement or enhancement of individual components without redesigning the entire application. The report specifically identifies future edge deployment using TensorFlow Lite as a potential extension .

V. IMPLEMENTATION

The implementation follows a systematic pipeline. First, food images and nutritional data are gathered from the Fruits-360 dataset and nutrition.csv. During preprocessing, images are resized, normalized, and augmented to reduce noise and improve dataset diversity. The report selects MobileNetV2 because of its balance between classification capability and computational efficiency.

The model is developed using TensorFlow/Keras with transfer learning. After preprocessing, the image is passed to the model, which extracts visual features and produces a food category with a confidence score. The backend then maps the category to the corresponding nutrition record. Since the nutrition dataset is maintained per 100 g, the values are scaled to the user's supplied weight.

The documented software environment includes Python 3.11+, React 18 with Vite, Flask, TensorFlow/Keras, OpenCV, Pillow, NumPy, Pandas, Scikit-learn, Node.js/npm, Git/GitHub, and Firebase. The report also documents PyTorch, YOLO World, and OpenCLIP for the AI recognition implementation. The application is designed to run on Windows, Linux, and macOS through modern web browsers.

VI. TESTING AND EVALUATION

Testing was performed using black-box and functional testing techniques. The test plan covers

image upload, image preprocessing, AI prediction, nutrition calculation, backend API communication, frontend display, validation, performance, stress, and usability.

Input validation was tested using supported and unsupported image formats, different image sizes, blurred images, incorrect weights, and empty fields. Functional tests verified that valid images reached the AI pipeline and that the system returned a food category and confidence score. The nutrition module was checked by comparing calculated values with the values stored in nutrition.csv. API testing verified correct request and response transmission between React and Flask.

Performance and stress testing were also considered. The report states that the system was evaluated under multiple prediction requests and consistently processed images, performed recognition, retrieved nutritional information, and displayed results without data loss or significant delay. The software requirements specify a target prediction response of approximately 2–5 seconds.

A documented banana test provides a concrete example. For an 80 g banana, the report records a 98% AI confidence score and the following estimated nutritional values: 71.2 kcal calories, 0.88 g protein, 18.24 g carbohydrates, 0.24 g fat, 2.08 g fibre, and 9.76 g sugar. An apple example at 120 g is also documented with 62.4 kcal, 0.36 g protein, 16.56 g carbohydrates, 0.24 g fat, 2.88 g fibre, and 12.48 g sugar, with an AI confidence score of 77.61%.

These examples demonstrate the end-to-end connection between recognition, confidence reporting, weight input, nutritional lookup, and final visualization. However, the reported examples should be interpreted as project test results rather than as a statistically validated clinical nutrition benchmark.

VII. RESULTS AND DISCUSSION

The implemented application successfully demonstrates the intended end-to-end workflow. The report presents a home interface, image recognition screens, nutritional analysis results, scan history, and banana recognition. The application identifies food items, calculates nutritional information according to entered weight, and presents the output in organized information cards.

The banana case demonstrates high-confidence recognition at 98%, while the apple case demonstrates a lower confidence value of 77.61%. These examples also highlight an important characteristic of visual recognition systems: confidence can vary with the input image. The report recommends clear, well-lit images containing a single food item and keeping the object prominent in the frame to improve recognition reliability.

The system provides several practical advantages. First, it reduces manual nutrition lookup and calculation. Second, the locally trained recognition approach reduces dependence on paid third-party recognition APIs. Third, the modular architecture supports future model and deployment changes. Fourth, Firebase scan history allows users to revisit previous analyses.

The current scope is primarily focused on fruits and selected supported categories, and nutritional estimation depends on the correctness and coverage of the structured dataset. The report also identifies environmental image variation and the need for broader food coverage as continuing challenges. Thus, the system is best viewed as a practical nutrition-assistance prototype rather than a replacement for professional dietary assessment.

VIII. CONCLUSION

The AI Edge Nutrition Scanner demonstrates how artificial intelligence, computer vision, deep learning, web technologies, and structured nutritional data can be combined into a single nutrition-assistance platform. The system accepts food images, recognizes supported food categories, calculates nutritional values according to user-entered weight, and presents calories, protein, carbohydrates, fat, fibre, and sugar in an accessible interface.

The project successfully integrates a React frontend, Flask REST API, AI inference pipeline, nutrition dataset, and Firebase scan history. Testing confirms that the principal functional modules operate together and that the system can provide fast, organized nutritional analysis for supported food categories. The documented banana and apple cases further demonstrate the relationship between AI confidence, food recognition, weight-based nutrient calculation, and user-facing results.

The proposed architecture also provides a foundation for future edge-oriented and mobile nutrition applications. Its primary contribution is the integration of recognition and nutrition estimation into a lightweight, modular workflow that minimizes manual effort and supports future expansion.

IX. FUTURE ENHANCEMENTS

The project report identifies several future directions. These include TensorFlow Lite optimization for edge devices, mobile application development, multi-food detection, portion-size estimation using depth sensing, barcode-based food identification, personalized meal recommendations, cloud synchronization, and integration with wearable health-monitoring devices.

A broader and more diverse food dataset can also extend recognition beyond the current fruit-focused scope. Future evaluation should include larger test sets, class-wise precision and recall, confusion matrices, inference-time measurements, robustness testing under different lighting and backgrounds, and comparison of alternative lightweight architectures. Such evaluation would provide stronger evidence for deployment in real-world nutrition-monitoring environments.

REFERENCES

[1] B. D, “AI Edge Nutrition Scanner,” Mini Project Report, Department of Computer Science and Engineering, East West Institute of Technology, Visvesvaraya Technological University, 2025–2026.

[2] B. D, “AI Edge Nutrition Scanner: System Analysis, Design and Implementation,” project report sections on requirements, architecture, methodology, and algorithm, East West Institute of Technology, 2025–2026.

[3] C. Liu, Y. Cao, Y. Luo, et al., “DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment,” cited in the project report as a 2016 IEEE work.

[4] L. Bossard, M. Guillaumin, and L. Van Gool, “Food-101 – Mining Discriminative Components with Random Forests,” cited in the project report.

[5] M. Sandler, A. Howard, M. Zhu, et al., “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” cited in the project report as a 2018 work.

[6] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” International Conference on Machine Learning, 2019.

[7] S. J. Pan and Q. Yang, “Transfer Learning for Image Classification Using Deep Neural Networks,” IEEE Transactions on Knowledge and Data Engineering, 2010.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” cited in the project report, 2012.

[9] H. Mureșan and M. Oltean, “Fruits-360 Dataset: A Dataset for Machine Learning and Computer Vision Applications,” cited in the project report, 2018.

[10] “Computer Vision-Based Food Recognition and Nutrition Estimation Using Deep Learning,” cited in the project report as a 2023 IEEE work.

[11] B. D, “AI Edge Nutrition Scanner,” Testing and Evaluation section, East West Institute of Technology, 2025–2026.

