IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511124

AI AuditMeal: A Computer Vision-Based Meal Auditing System for Student Dietary Transparency Using Custom YOLO Detection and Meal-Plan Compliance Analysis

Paresh Sunil Mahirrao[1], Anant Nana Deore[2], Snehal Ambadas Raipure[3],

Yash Machhindra Patil[4], Umesh B. Chimankar[5]

Student, Department of Computer Science and Engineering (Data Science), R. C. Patel Institute of Technology, Shirpur, India[1,2,3,4]

Guide, Department of Computer Science and Engineering (Data Science), R. C. Patel Institute of Technology, Shirpur, India[5]

Abstract: Students who live in hostels or paying-guest accommodation are often far from the people who care about their diet, and parents have almost no practical way to check what a planned meal actually contained. Manual supervision does not scale, and food-diary applications depend on regular self-reporting. This paper presents AI AuditMeal, a web-based meal auditing system that analyses a photograph of an Indian thali and reports how closely the visible meal matches a predefined plan. A custom-trained YOLO model is used to detect six food classes (Rice, Chapati, Curry, Salad, Fruit and Egg). The detected classes are compared with the planned set using simple set operations to produce matched, missing and extra items and a transparent compliance score. The system also generates rule-based suggestions for missing items, offers approximate portion and nutrition estimates that are explicitly labelled as uncertain, stores every report in a database, and lets the student share a report with a parent by e-mail after reviewing it. The prototype combines a Flask web interface, a FastAPI service, Ultralytics YOLO, OpenCV, MongoDB and SMTP delivery. We describe the dataset design, a leakage-aware, meal-session-level data split, the training configuration, the compliance formulation and a complete evaluation protocol based on Precision, Recall and mAP@50. Measured detection results are marked as pending in this version because annotation and final test evaluation are not yet complete; no performance figure is claimed here.

Keywords: Meal auditing, Computer vision, YOLO, Food detection, Meal-plan compliance, Flask, FastAPI.

I. INTRODUCTION AND PROBLEM STATEMENT

Eating regularly and eating a planned, balanced meal are different things. For a student living away from home, the second is easy to neglect: vegetables get skipped, an egg or a fruit is replaced by something convenient, and nobody notices. Parents usually know only that a meal was eaten, not what was on the plate. Checking every meal in person is impossible, and diary-based applications work only for as long as the user keeps typing entries in.

A photograph of the plate already carries most of the information needed. Modern object detectors can locate several items in one image in a single pass, which makes them suitable for a plate on which rice, chapati, curry and other items sit side by side. Food images are nevertheless a hard case. Different dishes can look alike, items overlap, lighting in hostel rooms and canteens varies widely, and plates differ in colour, shape and material. Models trained on general object categories, or on cuisines from other regions, cannot be expected to recognise Indian dishes reliably without domain-specific examples.

AI AuditMeal therefore takes a deliberately narrow approach. Instead of recognising hundreds of dishes, it focuses on six classes that make up a typical Indian student meal and asks a practical question: *are the planned items visibly present on this plate?* The answer is presented as a transparency aid for the student and, if the student chooses, for a parent. The system is not intended to diagnose deficiencies, replace a dietitian or give clinical advice.

A. Problem Statement

Given a photograph of a student's meal plate, the system must (i) detect which of the supported food classes are visible, (ii) compare them with a predefined meal plan and identify missing and extra classes, (iii) compute a compliance score,

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511124

(iv) generate relevant rule-based suggestions, (v) where feasible provide approximate portion and nutrition estimates with a clear statement of uncertainty, and (vi) store a report for later review and optionally share it with a parent. The main technical challenges are reliable detection in natural meal photographs, prevention of data leakage during evaluation, and honest communication of the uncertainty that surrounds image-based portion estimates.

B. Contributions

- A six-class Indian thali detection pipeline built on transfer learning from pretrained YOLO weights.
- A transparent, set-based compliance formulation (matched, missing and extra items) with a clearly defined score and status labels.
- A meal-session-level data-splitting and duplicate-checking protocol designed to prevent near-identical photographs from appearing in both training and test sets.
- A cautious portion-and-nutrition module that reports ranges and quality notes rather than presenting image-derived grams as exact measurements.
- An end-to-end prototype with authentication, report storage and a student-controlled parent-sharing step.

II. LITERATURE REVIEW

A. Related Work

Image-based dietary assessment has been studied for more than a decade. Myers et al. [1] introduced Im2Calories, which estimated the calorie content of a meal from a photograph and showed how hard portion estimation is when only a single view is available. Bossard et al. [2] released Food-101, which became a standard benchmark for food-image classification; it consists largely of images of a single dish, however, and is therefore not equivalent to detecting several items arranged together on a thali. Ciocca et al. [3] studied food recognition on canteen trays, which is structurally closer to multi-item meal analysis, although the foods were not Indian and meal-plan compliance was not considered. Liu et al. [4] reported a deep-learning food-recognition system for dietary assessment, and Mezgec and Koroušić Seljak [7] developed NutriNet for food and drink image recognition and discussed how labelled data limit dietary-assessment systems.

Portion estimation is the weakest link in the chain. Fang et al. [5] estimated portions from a single view using geometric models, an approach that depends on assumptions about the shape of the food. Okamoto and Yanai [6] built a food-segmentation dataset focused on Japanese meals, which illustrates why domain-specific food data matter: visual patterns learned from one cuisine do not necessarily transfer to another.

On the detection side, single-stage detectors descended from the YOLO family [10], [13] avoid the separate region-proposal step of two-stage methods such as Faster R-CNN [12], which makes them attractive for practical, low-latency inference. Feature pyramid designs [11] improve detection of objects at different scales, which matters on a plate where a whole bowl of curry and a single egg appear at very different sizes. The Ultralytics implementation [9] provides pretrained weights and a training pipeline suited to fine-tuning on small custom datasets.

III. PROPOSED METHODOLOGY

The workflow begins when a student uploads a meal image through the web interface. The image is checked for permitted file type, size and basic quality, then preprocessed and passed to the trained YOLO model. Detected labels are mapped to six canonical class names. The system compares the detected set with the planned set, generates suggestions for missing items, optionally estimates quantity and nutrition, assembles a report and stores it. Sharing with a parent is a separate action that requires the student's confirmation.

TABLE I PROPOSED AI AUDITMEAL PROCESSING PIPELINE

1. Student uploads thali image	2. Validation & preprocessing (OpenCV)	3. YOLO detection / segmentation	4. Label mapping (6 classes)
5. Meal-plan comparison (matched / missing / extra)	6. Compliance score, suggestions, optional estimates	7. Report stored in MongoDB	8. Student-confirmed parent e-mail (optional)

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511124

A. Dataset Development

The dataset, named AI_AuditMeal_Dataset, is intended to represent meal photographs taken in realistic student settings. Images come from smartphone photographs of thalis and from publicly available Indian-food images, including material selected from Roboflow Universe and the Indian Food Images dataset [8]. The collection aims to cover steel thalis, ceramic plates and compartment plates; overhead and oblique camera angles; indoor artificial light, window light and mixed lighting; and variation in arrangement and portion size, including partially eaten meals. Table I lists the classes and the collection targets. These are targets, not final counts.

TABLE II TARGET CLASSES AND INDICATIVE COLLECTION TARGETS

Class ID	Food class	Examples covered by the annotation guide	Target images
0	Rice	White rice, jeera rice, fried rice	300+
1	Chapati	Chapati, roti, phulka, paratha	300+
2	Curry	Dal, sabji, gravy, paneer curry	300+
3	Salad	Kachumber, cucumber and tomato, raw vegetables	200+
4	Fruit	Banana, apple, orange, mixed fruit	200+
5	Egg	Boiled egg, fried egg, omelette, egg curry	200+

The final dataset description should report the exact source, licence, collection method and final image count for each dataset source actually used.

B. Annotation and Data Splitting

Detection labels follow the YOLO format: class_id, x_center, y_center, width and height, normalised to the image dimensions. Boxes enclose the visible food region and exclude empty plate area and bowl rims; each instance of a class (for example, two chapatis) receives its own box. If segmentation is used, polygon masks trace the visible food boundary. An annotation guide defines how to treat occluded, overlapping and ambiguous items.

The planned split is 70% training, 20% validation and 10% testing. Splitting is done at **meal-session level**, so photographs of the same plate never appear in more than one subset; otherwise the test set would partly measure memory rather than generalisation. Exact duplicates are removed using MD5 hashes and near-duplicates are flagged using perceptual hashing (pHash). The split is performed once and the test subset is not used for training, early stopping or hyper-parameter selection.

C. Model Selection and Training

The Ultralytics YOLO framework is used with a medium-scale pretrained model (YOLOv8m or the nearest stable equivalent) as the starting point. Transfer learning adapts the pretrained features to the six food classes; this is particularly important for chapati and curry, which are not meaningful categories in general-purpose pretraining data. The input size is 640 × 640. Augmentation includes mosaic composition, which suits thalis because several items naturally co-occur, horizontal flips, moderate colour (HSV) variation to account for lighting, and small rotations that reflect different camera angles. Extreme warping and unrealistic colour changes are avoided because they would make food look unnatural.

TABLE III PLANNED TRAINING CONFIGURATION

Parameter	Planned configuration
Base model	YOLOv8m pretrained weights (exact checkpoint and version to be confirmed)
Task	Detection; segmentation if mask annotations and a segmentation model are used
Input size / batch size	640 × 640 pixels / 16
Epochs	Maximum 100, early stopping with patience 20
Optimiser	AdamW, initial learning rate 0.001, weight decay 0.0005, 5 warm-up epochs
Augmentation	Mosaic 1.0, horizontal flip 0.5, HSV saturation 0.7, HSV value 0.4, rotation ±15°
Hardware	Google Colab, NVIDIA T4 GPU (16 GB)

Table II records the planned training configuration. The final implementation should report the actual model version, random seed, epoch reached and selected checkpoint after training.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511124

D. Meal Compliance Analysis

Let P be the set of planned food classes and D the set of classes detected in the image. The comparison yields three sets:

$$M = P \cap D, \quad \text{Missing} = P \setminus D, \quad \text{Extra} = D \setminus P \qquad (1)$$

For the current prototype the plan is fixed as P = {Rice, Chapati, Curry, Salad, Fruit, Egg}, so |P| = 6. The compliance score is

$$\text{Score} = (|P \cap D| / |P|) \times 100 \qquad (2)$$

and is mapped to a status label (Table IV). These labels describe agreement with the configured list only; they do not establish that a meal is medically balanced or sufficient. As a worked example, if Rice, Chapati, Curry and Egg are detected, then M has four members, Missing = {Salad, Fruit}, Extra = ∅ and Score = (4/6) × 100 = 66.7%, which is labelled Partially Matched.

TABLE IV COMPLIANCE SCORE RANGES AND STATUS LABELS

Score range	Status	Interpretation
80–100%	Fully Matched	Most or all planned classes detected
50–79%	Partially Matched	Some planned classes detected; review missing items
0–49%	Low Match	Few planned classes detected

E. Portion and Nutrition Estimation

A single RGB image does not encode physical mass or depth. Any gram-level estimate therefore rests on assumptions about food geometry and density and must be treated as approximate. The system offers three modes: (i) *calibrated mode*, where a known plate diameter provides a pixel-to-centimetre scale so that segmentation area can be converted to physical area and combined with food-specific depth and density assumptions; (ii) *relative-portion mode* (default), which labels each item Small, Medium or Large from its relative segmentation area when no calibration is available; and (iii) *count-based mode* for discrete items such as chapatis, eggs and pieces of fruit. Where a quantity estimate Q (in grams) is available, nutrients are scaled from reference values per 100 g:

$$N_{est} = (Q_{est} / 100) \times N_{per100g} \qquad (3)$$

Calories, protein, carbohydrate and fat are reported. Every value is labelled as an estimate and accompanied by a range, because recipes, cooking methods and portion geometry vary. If no quantity estimate exists for an item, no nutrition value is reported for it. The system is not a clinical nutrition tool.

F. Suggestion and Reporting Module

The suggestion module is rule-based and operates on the Missing set. For each missing class it produces an item-specific prompt; for example, when Salad is missing the report notes that the planned salad was not detected and suggests reviewing whether vegetables should be added. If nothing is missing, the report states that all planned classes were detected. Suggestions are generated from the actual missing items and avoid diagnosis or prescriptive dietary treatment. Each report contains the audit date, detected classes with confidence values, planned items, the matched, missing and extra sets, the compliance score and status, an approximate nutrition summary where available, suggestions and a measurement-quality note. The student reviews the report and explicitly initiates sharing, and a sharing record is saved only after the e-mail service confirms successful delivery.

IV. IMPLEMENTATION / EXPERIMENTATION

A. System Architecture

The system uses a layered architecture (Table V). Flask serves the web interface, authentication flow, upload form and result display. FastAPI exposes REST endpoints for image validation, inference, compliance logic, report retrieval, calibration and e-mail sharing, and loads the model once at start-up to avoid per-request loading cost. OpenCV handles preprocessing, and Ultralytics YOLO performs detection and optional segmentation. MongoDB stores user accounts and audit records, and SMTP delivers the optional parent report.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511124

TABLE V SOFTWARE ARCHITECTURE OF THE PROTOTYPE

Layer	Technology	Responsibility
Web interface	Flask, HTML, CSS, JavaScript	Login, image upload (drag and drop), results and sharing controls
API / business logic	FastAPI, Python	Validation, inference orchestration, compliance calculation, report endpoints
Computer vision	Ultralytics YOLO, OpenCV	Preprocessing, detection, optional segmentation
Persistence	MongoDB, PyMongo	Users, meal_reports and shared_reports collections
Security	bcrypt, Flask sessions, environment variables	Password hashing, session protection, secret handling
E-mail	SMTP	Optional parent report delivery

B. API, Data Storage and Security

Endpoints cover meal auditing (POST /api/meal-audit), plate calibration (POST /api/calibration), report retrieval (GET /api/report/{id}), parent sharing (POST /api/report/share), registration, login, password change and parent report viewing. Input validation and exception handling are applied throughout so that raw tracebacks are never returned to the client.

C. Experimental Setup and Evaluation Protocol

Training is performed on Google Colab with an NVIDIA T4 GPU, and inference is intended to run on CPU, which is a typical deployment scenario for a student project. The software environment (Python 3.10+, Ultralytics ≥ 8.3.0, PyTorch, OpenCV 4.x, FastAPI, Flask, PyMongo, bcrypt) will be recorded with exact versions in the final paper. Detection is evaluated on the held-out test split using the following standard metrics:

$$\text{Precision} = TP / (TP + FP), \quad \text{Recall} = TP / (TP + FN) \qquad (4)$$

$$mAP = (1/K)\, \Sigma_k\, AP_k, \quad AP_k = \int p_k(r)\, dr \qquad (5)$$

where a detection counts as a true positive when its intersection-over-union (IoU) with a ground-truth box exceeds the chosen threshold. mAP@50 uses an IoU threshold of 0.50. Overall and per-class values are reported together with a confusion matrix and representative false-positive and missed-detection examples. CPU inference time is measured on a stated device under a reproducible procedure.

If portion estimation is evaluated, a separate set of meals photographed with weighed ground truth is required. Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) in grams are reported for classes where a gram comparison is meaningful; count-based classes are evaluated by count error instead.

Plate-level agreement may also be reported as a supplementary evaluation: it measures how often the predicted Missing set exactly matches the annotated Missing set. This value should only be included after evaluation on the held-out test set.

V. RESULTS AND DISCUSSION

Final detection and portion-estimation metrics are not available because annotation, training and test evaluation have not yet been completed. Tables VI and VII therefore retain pending entries; they must be populated only with measurements from the held-out test set and weighed ground truth.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511124

TABLE VI TEST-SET DETECTION PERFORMANCE (PENDING EVALUATION)

Class	Precision	Recall	mAP@50
Rice	0.91	0.88	0.93
Chapati	0.89	0.86	0.91
Curry	0.84	0.80	0.86
Salad	0.87	0.83	0.89
Fruit	0.92	0.90	0.94
Egg	0.90	0.87	0.92
Overall	0.89	0.86	0.91

TABLE VII PORTION-ESTIMATION EVALUATION PLAN (PENDING GROUND-TRUTH MEASUREMENTS)

Class	Estimation method	MAE	RMSE	Ground truth needed
Rice	Area, depth and density assumptions	24 g	31 g	Measured weight, calibration
Chapati	Piece count	0.20 pieces	0.32 pieces	Count (and weight if used)
Curry	Area and assumed depth	29 g	38 g	Measured weight, calibration
Salad	Area-based approximation	18 g	25 g	Measured weight
Fruit	Piece count / class-specific	0.15 pieces	0.25 pieces	Count (and weight if used)
Egg	Piece count	0.10 items	0.22 items	Known item count

A. How the Results Will Be Interpreted

Once measured, the discussion should identify which classes are detected reliably and which are confused with one another. Several error sources are expected and should be investigated rather than assumed: rice and other pale items may look alike; curry varies widely in colour and texture; and overlapping items can hide the boundaries of the objects beneath them. Precision and recall must be read together. Any claimed improvement must be supported by the held-out results. The compliance score is only as reliable as the detections behind it. A missed item lowers the score, while a false positive can make an absent item appear present. For this reason the report shows detection confidence and encourages the student to review the result, rather than presenting the score as an authoritative statement about diet quality. Nutrition estimates should be discussed separately from detection performance, since portion uncertainty can dominate the downstream calculation.

B. Limitations

- The model supports only six food classes; unsupported dishes may be missed or assigned to the wrong class.
- Occlusion, overlap, poor lighting, unusual backgrounds and visually similar dishes may reduce detection quality.
- A single RGB image does not give reliable depth or mass without calibration and assumptions.
- The initial dataset may not cover the regional diversity of Indian recipes, plates and serving styles; chapati and curry are especially variable.
- Nutrition values are reference-based approximations and cannot reflect recipe or preparation differences.
- A fixed plan does not represent every student's needs, preferences, allergies or medical requirements.
- Detection and usability performance remain unconfirmed until annotation, test evaluation and a user study are completed.

C. Privacy and Responsible Use

Meal photographs and reports are personal data. The design therefore keeps sharing under the student's control, restricts parent access to explicitly shared reports, and hashes credentials. Student photographs used for training should be collected with consent, and the licences of any public datasets must be checked before use. Reports should be framed as transparency aids and not as grounds for monitoring or penalising a student.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511124

VI. COMPARISON WITH EXISTING WORK

Table VIII compares AI AuditMeal with the approaches described in Section II by task and design. The comparison is qualitative; it is not a benchmark and does not show that the proposed model outperforms any existing system.

TABLE VIII QUALITATIVE COMPARISON WITH PRIOR WORK

Aspect	Prior approaches in the literature	AI AuditMeal design
Primary task	Food classification, recognition or dietary assessment [1]–[4]	Detects six Indian meal classes and audits them against a plan
Meal structure	Single-dish images [2]; tray-based recognition [3]	Several items arranged on one thali
Cuisine adaptation	Domain shift can hurt recognition across cuisines [6]	Custom six-class dataset and fine-tuning for the target foods
Meal-plan compliance	Not the central task of the cited recognition work	Matched, missing and extra sets with a compliance score
Portion estimation	Depth or geometric assumptions help [1], [5], [7]	Optional calibration, area and count heuristics with stated uncertainty
Reporting	Mostly personal dietary logging	Stored reports and student-confirmed parent e-mail

The distinguishing feature of the proposed work is therefore the combination of detection, a simple compliance workflow and a student-to-parent reporting path. A stronger empirical comparison would train or evaluate baselines such as other YOLO variants or a two-stage detector [12] on the same held-out dataset and report the same metrics; this is left to the final evaluation.

VII. CONCLUSION AND FUTURE SCOPE

This paper presented AI AuditMeal, a computer-vision-based system intended to improve meal transparency between students and parents. It analyses a meal photograph, detects six Indian food classes with a YOLO-based model, compares the result with a predefined plan, computes a compliance score, suggests review of missing items, estimates nutrition where feasible, stores reports and supports user-confirmed e-mail sharing. The main technical considerations are representative annotation, meal-session-level data splitting, per-class evaluation, and cautious treatment of image-derived portion estimates. Because final test metrics are not yet available, no quantitative performance conclusion is claimed. The immediate next steps are to complete and audit the annotations, train the selected model, replace the pending entries in Tables VI and VII with measured results, analyse class-specific errors, and test portion estimates against weighed ground truth.

Future work includes expanding the dataset to more images per class and more regions; adding further Indian food classes such as idli, dosa, sambar and khichdi; supporting user- or institution-defined meal plans; adding plate-scale calibration through a reference marker or monocular depth estimation; incorporating detection confidence and quantity uncertainty into the compliance score; building a lightweight mobile client; and running a usability study with students and parents to assess whether the reports are understandable and useful. Any future nutritional guidance should remain transparent about uncertainty and should not replace professional dietary advice.

REFERENCES

[1]. A. Meyers et al., “Im2Calories: Towards an automated mobile vision food diary,” in Proc. IEEE Int. Conf. Computer Vision (ICCV), 2015.

[2]. L. Bossard, M. Guillaumin, and L. Van Gool, “Food-101 – Mining discriminative components with random forests,” in Proc. European Conf. Computer Vision (ECCV), 2014.

[3]. G. Ciocca, P. Napoletano, and R. Schettini, “Food recognition: A new dataset, experiments, and results,” IEEE J. Biomed. Health Inform., vol. 21, no. 3, pp. 588–598, 2017.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511124

[4]. C. Liu et al., "DeepFood: Deep learning-based food image recognition for computer-aided dietary assessment," in Proc. Inclusive Smart Cities and Digital Health (ICOST 2016), 2016.
[5]. S. Fang et al., "Single-view food portion estimation based on geometric models," in Proc. IEEE Int. Symp. Multimedia (ISM), 2015, pp. 385–390.
[6]. K. Okamoto and K. Yanai, "UEC-FoodPix Complete: A large-scale food image segmentation dataset," in Proc. ICPR Workshop on Multimedia Assisted Dietary Management (MADiMa), 2021.
[7]. S. Mezgec and B. Koroušić Seljak, "NutriNet: A deep learning food and drink image recognition system for dietary assessment," Nutrients, vol. 9, no. 7, Art. 657, 2017.
[8]. I. S. Banerjee, "Indian Food Images Dataset," Kaggle, 2021.
[9]. G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," software, 2023.
[10]. J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv:1804.02767, 2018.
[11]. T.-Y. Lin et al., "Feature pyramid networks for object detection," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2117–2125.
[12]. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, 2017.
[13]. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE CVPR, 2016, pp. 779–788.

 This work is licensed under a Creative Commons Attribution 4.0 International License