Received Date: 12th February, 2025
Revision Date: 5th March, 2025
Accepted Date: 11th March, 2025

SmartCal: Calorie Estimation of Local Nepali Cuisine with Deep Learning-Powered Food Detection

Arnab Manandhar[1*], Looza Subedy[2], Chandan Kumar Pandit[3], Praches Acharya[4]
[1]Dept of Electronics and Computer Engineering, Thapathali campus, TU. Email: arnab.manandhar11@gmail.com
[2]Dept of Electronics and Computer Engineering, Thapathali campus, TU. Email: loozasubedy998@gmail.com
[3]Dept of Electronics and Computer Engineering, Thapathali campus, TU. Email: chandan7kp@gmail.com
[4]Associate Professor, Dept of Electronics and Computer Engineering, Thapathali campus, TU. Email: tc.pacharya@tcioe.edu.np

Abstract— Diet observation is a critical component of preventive healthcare, yet traditional manual calorie tracking methods are often inaccurate and time-consuming. This paper presents SmartCal, an automated system for calorie estimation of local Nepali cuisine using deep learning. The system focuses on foods like rice, lentils, spinach, and boiled eggs, which are staples in Nepali diets. It employs an ESP32 Camera Module to capture food images, which are processed by a Mask R-CNN (Region-Based Convolutional Neural Network) model trained specifically for Nepali cuisine. The model achieves a mean Average Precision at 50 (mAP50) of 99%, demonstrating high accuracy in food detection. Calorie estimation is performed based on standard cooking conditions and fixed portion sizes. Results are displayed via a web application, where users can set daily calorie targets and track their intake history. This system provides a practical, accurate, and user-friendly solution for dietary monitoring, particularly suited to the dietary habits of Nepali individuals.

Keywords— Calorie Intake, Convolutional Neural Network (CNN), Deep Learning, ESP32 Camera Module, mAP50, Mask R-CNN, Nepali Cuisine

Introduction

The global rise in obesity and associated health risks has underscored the urgent need for innovative solutions to promote healthier lifestyles and combat diet-related diseases. In an era marked by hectic schedules and diverse food choices, maintaining a balanced diet has become increasingly challenging. Calorie intake plays a pivotal role in health management, yet traditional methods of manual food journaling are often inaccurate, time-consuming, and prone to underreporting. According to the World Food Organization, the minimum daily calorie requirement for adults is 2,220 calories; however, 21% of Nepal's population falls short of this threshold. This highlights the critical need for accessible and accurate dietary monitoring tools tailored to local dietary habits. Motivated by these challenges, this paper presents SmartCal: Calorie Estimation of Local Nepali Cuisine with Deep Learning- Powered Food Detection, an automated system designed to simplify and enhance dietary management. The project leverages advanced deep learning techniques, specifically a Mask R-CNN (Region-Based Convolutional Neural Network) [1], to detect and classify common Nepali food items such as rice, lentils, spinach, and boiled eggs. By integrating the ESP32 Camera Module [2] for image capture and a user-friendly web interface for calorie tracking, SmartCal provides real-time, accurate calorie estimations based on fixed portion sizes and standard cooking conditions. The primary objective of this work is to address the limitations of conventional calorie tracking methods by automating food detection and calorie estimation. The system aims to [3] identify food items using Mask R-CNN and calculate their calorie values, and [4] provide a web-based interface for users to view calorie estimates, track daily intake, and set personalized calorie targets. This approach not only eliminates the inaccuracies of manual input but also encourages users to adopt healthier dietary habits through continuous monitoring and feedback. The scope of SmartCal extends to automating calorie tracking, improving accuracy, and enhancing user engagement through its intuitive web interface. Potential applications include personal dietary management, research by nutritionists, and optimization of food portions in restaurants. However, the system currently focuses on four specific food items and requires images to be captured under consistent lighting conditions and in identical bowls of known volume. Despite these limitations, SmartCal represents a significant step toward empowering individuals to take control of their nutritional well-being in an increasingly fast-paced world.

** Corresponding Author*

Literature Review

In the pursuit of addressing the challenges associated with dietary monitoring, recent advancements have introduced innovative solutions to calculate the calories present in food. One of the methods, as mentioned in [5], uses the YOLO model [6] to detect multiple types of food in the input image and calculate food calories by multiplying calories for each food by its amount. In this method, the volume of the food is not calculated and the calculation of the amount of food is based on the dataset which contains multiple images of food with different amounts, which doesn't give an accurate estimate of the portion size and is limited to only a small amount of foods.

Another method, described in [7], allows the user to snap a photo of the food using a smartphone and compare it to a pre-defined image of the same item that has its known nutrient value saved in a database. This system's primary drawback is that it ignores the food's size, which is a crucial factor.

Puri et al. [8] proposed a food volume estimation using a stereo-vision based approach, based on the multiple-view re-construction method. Here, the users were required to capture images from multiple angles and a fiducial marker (reference object) was used to recover the global scale factors of the captured food item. After scale correction, dense stereo matching was performed to obtain left-right image-based matches. Also, RANSAC [9] was used for table plane estimation. And, the volume estimation process involved Delaunay Triangulation to fit the surface of food. Here, the use of a checkerboard as a fiducial marker every time to determine the volume of food, makes it less convenient.

However, Sun et al. in [10] proposed a VR approach that uses a wearable device embedded with a camera that constantly records the person's meals. It utilized pre-built 3D- food models with known volumes and superimposed them onto the food items in the real scene. By scaling, translating, and rotating the models to fit the food items in the image, food volume was estimated. The limitation of this method stems from the fact that it might not be able to track irregular food shapes that do not resemble pre- built 3D models. In the system proposed by Yoshikazu et al. in [11], smartphones with a pair of cameras that can be treated as stereo cameras are used to estimate the volume of food items. It takes an RGB-D image of a dish, estimates categories and volumes of foods on the dish, and calculates the amount of their calories using a pre-registered calorie density of each food category. This method achieves a high accuracy and no fiducial marker is required. However, depth-sensing cameras are not always embedded in smartphones, which limits the functionality of the system.

In reference [12], the suggested method uses a multi-step procedure based on mobile device picture capture to analyze meal portions and estimate calories accurately by pre-processing the images, color and texture segmentation, feature extraction, Support Vector Machine (SVM) classification, and calorie calculation are the main processes. Furthermore, a one- time calibration procedure is utilized for size reference, utilizing the thumb of the user, guaranteeing precise measurement of food portions in pictures.

In the real world, a user may have multiple food items on their plate. Each of the items is to be labeled separately still because they may share a common space on the plate. While multi-class classifiers are used in most applications, it is assumed that the labels are mutually exclusive. The system proposed by Myers et al. [13] used a multi-label classifier, which acknowledges that food items might share a common space. It is a better method for multiple food detection and estimation. Moreover, it also reduces the discrepancies left by the CNN model by running the CRF smoothing for semantic image segmentation using the system from [14]. The food portion in the image is extracted more finely. Volume estimation is conducted by the CNN itself as it predicts the depth map for the detected food items. The depth map is then converted into a voxel representation. To do so, the system uses RANSAC to separate the food from the table, projecting each pixel of the food to 3D space to the known intrinsic parameters of the F200 sensor [15]. The table surface is then tiled with a 2d grid and the average height of all the points in each cell of the grid is computed, while a "tower" of the retrieved height is constructed. Still, an error of 50 to 400 ml was found in [13] as shown in the comparison table in [10]. The research in [10] concludes that an integrated approach provides 93.1 percent accuracy in food volume estimation, indicating, that it is better to combine multiple approaches.

In the method proposed by Hu et al [16], volume is estimated by calculating the depth map from a single image by the employment of encoder and decoder-based architecture, the model is trained on NYU Depth Dataset, which consists of RGB-D images of indoor scenes various building structures. This model has also been used for finding the depth of other unrelated images such as food items. However, the accuracy of the depth map remains questionable.

Methodology

As shown in figure 1, Firstly, the users are required to input their credentials through the web application interface which includes their name, age, and calorie intake goal for the day. Then, if the user wants to estimate the calories of the food items they are about to have, they are requested to use the proposed system which comprises a camera to capture an image of the food item. The camera is placed above the food items such that the top view is captured.

The ESP32-CAM AI Thinker module comprises the OV2640 camera [17] and a microcontroller, with Wi-Fi functionality in it while the Arduino Uno [18] facilitates its power supply. The OV2640 camera captures the image of the food item which is sent to the server by the module, through Wi-Fi, using the HTTP. The server side, created using NodeJS, receives the image and directs it into the database of the server, where it gets saved. The image captured is stored in the server is used by the trained Mask R- CNN model [19], which not only detects the food but recognizes it.

After that, as the image is segmented and the food items are recognized, the weight of the detected food items is calculated. With respect to the predicted food, the fixed volume is converted into its weight using the values of the physical density of food items.

A simple function is then implemented to calculate the calories of the recognized food items using the predicted food weight values and standard calorie values present in the Nutrition API [20]. With this, the calorie associated with the food item is estimated. Then, the calculated result is displayed on the website. The image is also uploaded to and displayed on the user interface section created using ReactJS.

Additionally, the system incorporates a database to store users' calorie consumption and foods eaten each day, providing a comprehensive overview displayed on a dedicated web page. Users can actively engage in their health management by setting personalized daily calorie intake targets through the web page. The resulting estimated calorie values, along with users' intake history and target goals, are dynamically showcased on the web page, fostering a user-centric and informed approach to dietary choices.

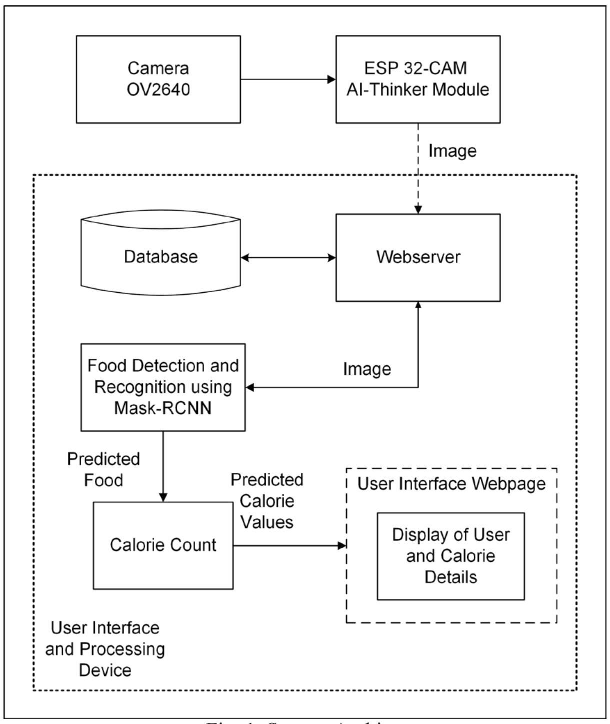

Camera OV2640
ESP 32-CAM AI-Thinker Module
Image
Database
Webserver
Food Detection and Recognition using Mask-RCNN
Image
Predicted Food
Predicted Calorie Values
Calorie Count
User Interface Webpage
Display of User and Calorie Details
User Interface and Processing Device

Fig. 1. System Architecture

Implementation Details

Hardware Implementation

The implementation of the project has been divided into hardware and software. Both portions are interdependent for the full functioning of the system. The following figure 2 shows the connection of hardware during module programming.

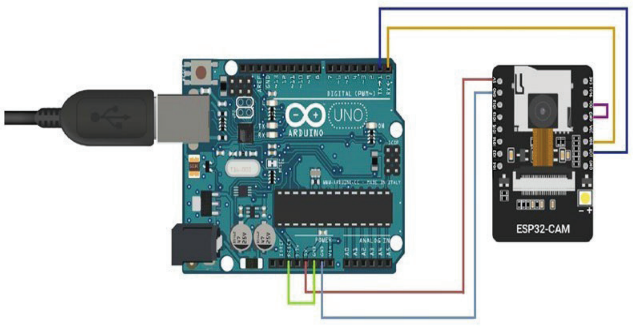


Fig. 2. Connection during Module Programming

When power is initially applied to the Arduino Uno, the microcontroller starts in an undefined state. Grounding the reset pin briefly (connecting it to GND) ensures that the Arduino Uno microcontroller is forced to reset. Similarly, the ESP32CAM Module enters the programming mode when the 'IOO' pin is grounded as in the figure. A USB A to USB B cable connects the computer to the Arduino UNO which sends it further to the ESP32 CAM Module.

The CAM Module stores the received program in its Flash Memory. And then, it only requires a power supply of 5V and grounding to capture images.

Software Implementation

Software implementation comprises using the Food Detection Model, website implementation, and counting the calories.

1) *Food Detection Model*: We utilized the Matterport Mask R-CNN repository [21], which provides a comprehensive implementation of the model, and created a separate environment using Anaconda to install the dependencies for the model as specified in the repository, and for running the model. After that, we downloaded the pre-trained weights of the Mask RCNN Model on the COCO Dataset [22] and initialized the model with these weights.

   We integrated the custom food dataset into the Matterport Mask R-CNN framework and modified the dataset class to load our annotations, images, and categories. Since our custom dataset includes classes not present in the COCO dataset, we update the configuration file. Extend the list of class names and indices to accommodate the additional classes

   As we have limited annotated data for our specific classes, we fine-tune only the top layers of the model. We freeze the backbone layers and RPN, allowing the model to adapt to our custom dataset while retaining the knowledge from COCO. Then we train the model and save the trained weights, along with the losses associated with each epoch inside our logs directory. We have supplied one image per GPU, set the number of classes to 5 including the background class, the learning rate of 0.001, the batch size as 4 with 89 steps per epoch, and a minimum detection confidence of 90% for training on our custom dataset.

   Through the Flask application, we integrate the pre-trained Mask R-CNN model to facilitate the deployment of the model for real-time food item detection. We configured the flask server to handle the POST requests made to /predict route. The images are passed to the Flask API for analysis. Upon receiving an image, it is decoded, and the initialized Mask R-CNN model executes inference to perform instance segmentation. The flask application runs on default port 5000. The API responds with a JSON containing the information about the detected instances, class names, and confidence scores, providing a user-friendly interface for using the pretrained Mask R-CNN model for food detection.

2) *Website Implementation:* The details of website implementation are divided as the Frontend and Backend portions. The frontend portion of the website was handled using React, where we created four components: Display, Home, Navbar, and Register. We used Axios to make POST and GET requests from the react components to the NodeJS server. Axios operates as a promise-based HTTP client for the browser and Node.js, allowing developers to send asynchronous requests to a server.

   The nodeJS server runs on port 4000, and it uses the Express framework to handle all the requests made to the server. It is responsible for communicating with the APIs, which include Flask API and Nutrition API to make the requests and receive the results accordingly. It is also responsible for storing the data in our MongoDB database.

Dataset Analysis

1) *Dataset Collection:* The model is initially trained on the COCO (Common Objects in Context) dataset, which is a widely used benchmark in computer vision for object detection, segmentation, and captioning tasks. It comprises a large collection of images, each annotated with object-bounding boxes and segmentation masks. The COCO dataset consists of around 118,000 images for training and 5,000 images for validation, covering a diverse range of scenes and objects, 80 common object classes such as people, animals, and everyday items. Images in the COCO dataset are densely annotated, providing detailed information about the location and shape of objects within each image, all of which we train our model on, to later be fine-tuned to our custom Food Dataset.

   To adapt the model for food-related tasks, a custom food dataset was curated by combining images from various sources, including the Indian Food dataset and Nepali Food dataset present in Kaggle, a popular platform for varieties of existing datasets. The custom food dataset consists of 354 images for training and 95 images for validation, covering some of the common food items in Nepal. The dataset includes 4 object classes: boiled egg, rice, lentils, and spinach.

Fig. 3 Custom Dataset Sample Images

2) *Dataset Sample:* Annotations: Images in the food dataset are manually annotated using the VGG Image Annotator [23], and different food items within a single image are properly classified

Result Analysis

The results include the ESP-32 CAM result, the food detection model result, and the website result.

A. ESP-32 CAM Results

An image of the dimensions 1600 x 1200 pixels, with a bit depth of 24 and file format '.jpg' was received by the server database. With proper lighting in the environment where the image was captured, the image received is of decent quality with which food items can be detected, distinguished, and recognized. The ESP32-CAM Port Transmission can be seen in figure 5.

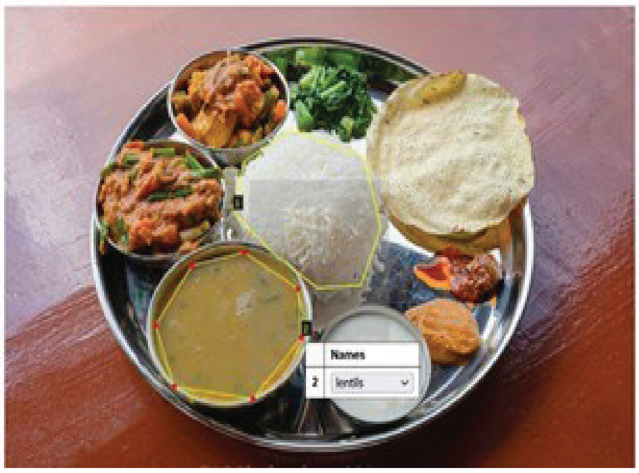
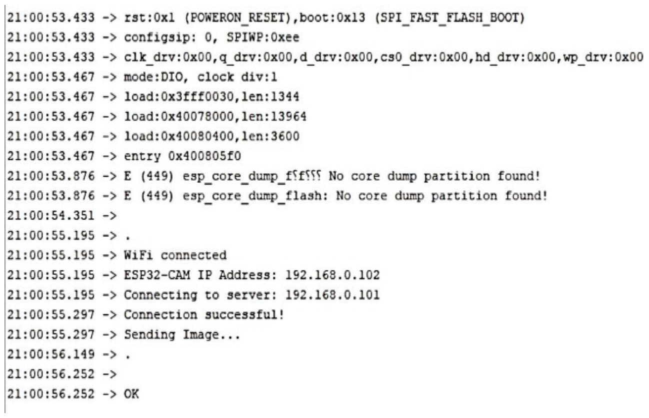


Fig. 4 Annotated Images


```
21:00:53.433 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
21:00:53.433 -> configsip: 0, SPIWP:0xee
21:00:53.433 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
21:00:53.467 -> mode:DIO, clock div:1
21:00:53.467 -> load:0x3fff0030,len:1344
21:00:53.467 -> load:0x40078000,len:13964
21:00:53.467 -> load:0x40080400,len:3600
21:00:53.467 -> entry 0x400805f0
21:00:53.876 -> E (449) esp_core_dump_f⸮f⸮⸮ No core dump partition found!
21:00:53.876 -> E (449) esp_core_dump_flash: No core dump partition found!
21:00:54.351 ->
21:00:55.195 -> .
21:00:55.195 -> WiFi connected
21:00:55.195 -> ESP32-CAM IP Address: 192.168.0.102
21:00:55.195 -> Connecting to server: 192.168.0.101
21:00:55.297 -> Connection successful!
21:00:55.297 -> Sending Image...
21:00:56.149 -> .
21:00:56.252 ->
21:00:56.252 -> OK
```
Fig. 5 ESP-32 CAM Port Transmission Messages

Food Detection Model Results

The pre-trained Mask R-CNN model was fine-tuned only on the top layers of the model. We freeze the backbone layers and RPN, allowing the model to adapt to our custom dataset while retaining the knowledge from COCO. Our custom dataset consists of each class divided roughly in the ratio of 8:2, which means 80% of the total images in the dataset were used for training, while 20% of the images were used for validation. A total of 100 epochs were used for training our model with a batch size of 4, and a step size of 89.

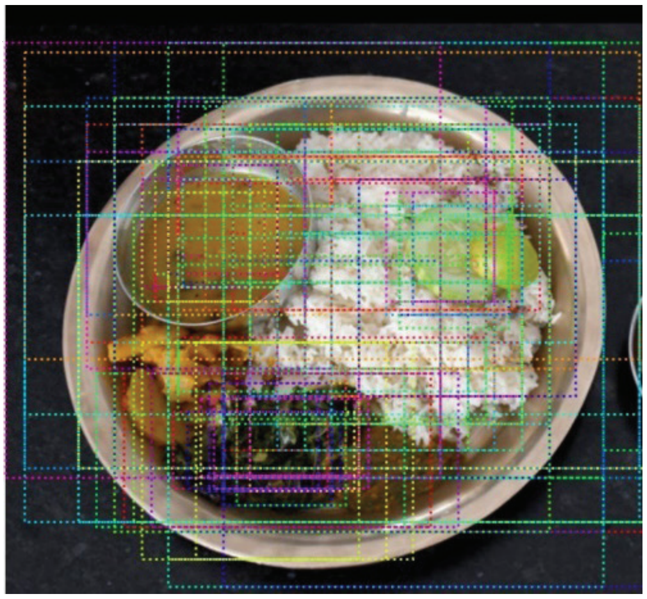


Fig. 6 Anchor Boxes Generation

As shown in figure 6 and 7, firstly, the anchor boxes are generated at various scales and aspect ratios across the feature map. These anchors serve as potential region proposals.

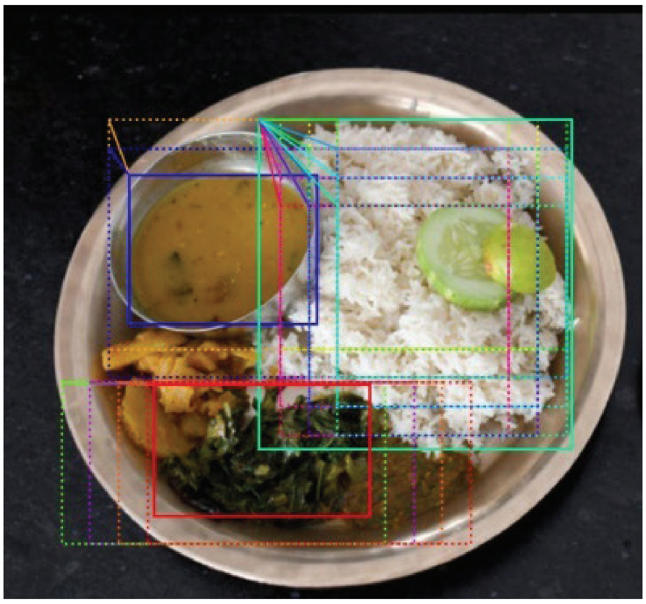


Fig. 7 Positive Anchors

Then, top anchors are selected among all the probable anchors. The dotted lines represent the positive anchors beforebounding box refinement and the solid line represents theanchor after refinement.

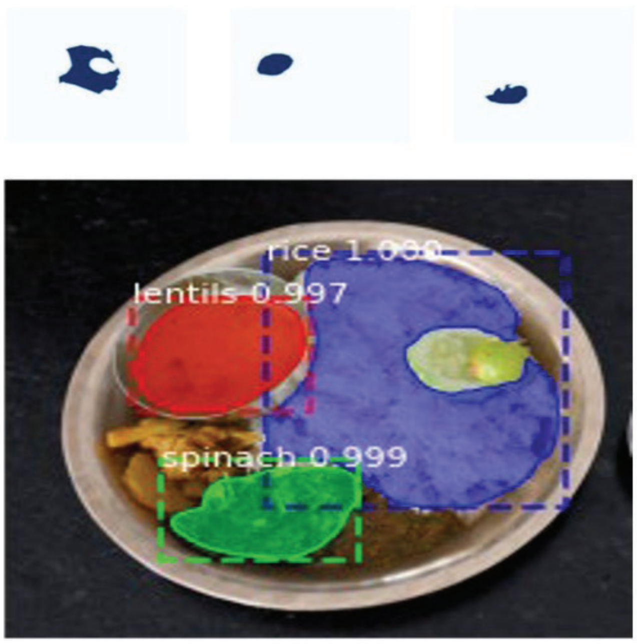


Fig. 8 Segmented Masks and Final Output

The corresponding masks for each detected instance are generated, then the final output for a given image is obtained.

Training and Validation Results

The evaluation for the food detection was done using the mAP, F1 score, precision, and recall evaluation metrics. And, the obtained curves for each of them are shown below

1) *Precision Per Epoch:* Initially, precision starts at 0.28 in the initial epoch and gradually improves to a peak of approximately 0.74 by the final epoch as seen in figure 9. Despite some fluctuations observed between epochs, particularly noticeable after epoch 10, the overall trajectory indicates a consistent enhancement in the model's ability to make accurate positive predictions. From epochs 21 to 100, precision gradually increases and stabilizes at a relatively high level, ranging between 0.65 and 0.74.

2) *Recall Per Epoch:* Initially, recall starts at around 0.43 in the first epoch and steadily climbs to approximately 0.65 by the final epoch. Despite the fluctuations, the overall pattern indicates consistent improvement in the model's ability to capture true positive instances from the dataset. At epochs 90-100, the recall stabilizes around 0.625 to 0.650. The curve can be observed in Figure 10.

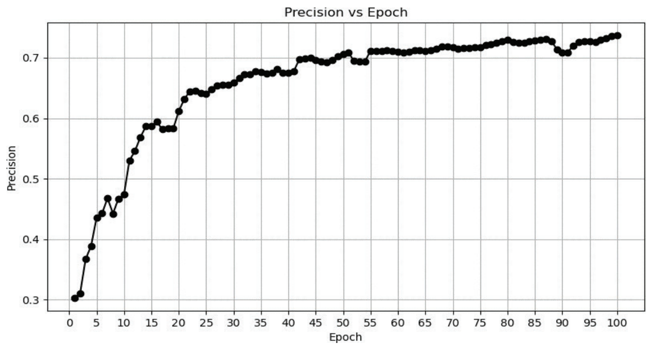


Fig. 9 Precision vs Epoch Curve

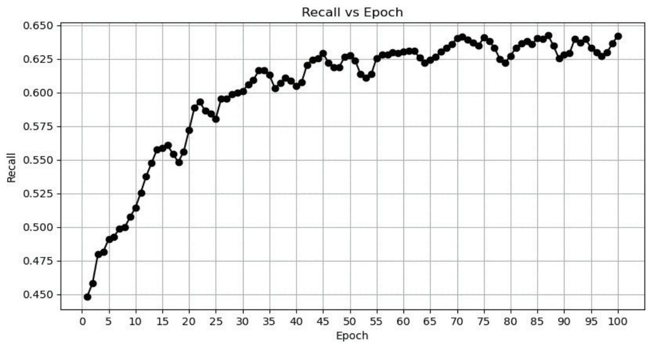


Fig. 10 Recall vs Epoch Curve

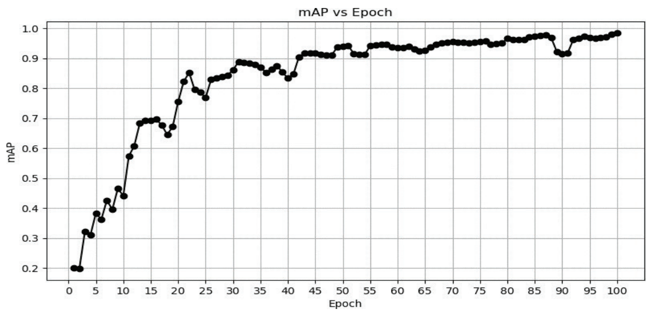


Fig. 11 mAP50 vs Epoch Curve

3) *mAP Per Epoch*: Figure 6-7: mAP50 vs Epoch curve Map50 Curve as shown in figure 11 starts at 0.13 in the first epoch, mAP shows a continuous rise, reaching its highest value of 0.99 by epoch 100. This signifies that the model becomes more proficient in precisely localizing and classifying objects within the images, resulting in high-quality detections.

4) *Score Per Epoch*: Initially, the F1 score starts around 0.25 and gradually increases over epochs, reaching approximately 0.69 by the 100th epoch. This demonstrates a significant improvement in the model's ability to balance precision and recall as training progresses. Notably, there is a notable acceleration in the F1 score improvement after the stabilization period, indicating that the model's learning becomes more effective over time.

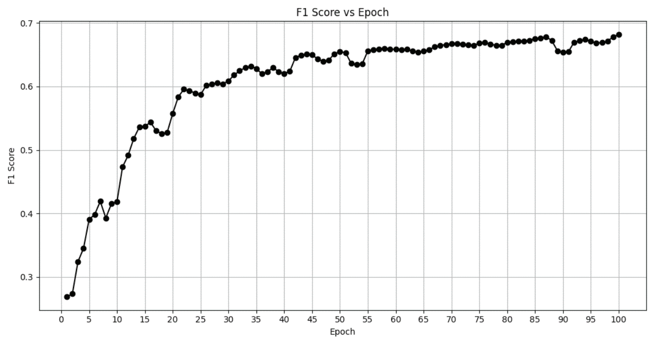


Fig. 12 F1 Score vs Epoch Curve

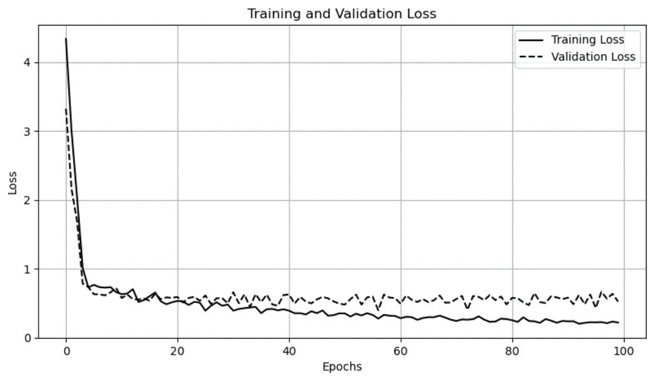


Fig. 13 Overall Losses

5) *Losses Per Epoch:* Initially, the overall loss is relatively high, around 4.34, but it decreases significantly over the first few epochs. After some fluctuations, the loss stabilizes, indicating that the model has converged to a relatively optimal solution. The mask loss, shown in figure 14 quantifies the error in pixel-level segmentation, demonstrates a decreasing trend over the epochs, indicating that the model is progressively improving its ability to generate accurate segmentation masks. It was observed that despite the stabilization of validation and mask loss around the early epochs of training, the decision to continue training the model until the completion of 100 epochs was justified by the continuous improvement in precision, recall, F1Score and mean average precision (mAP) values, which shows that the model's ability to correctly detect food items increases gradually along with the increase in epochs.

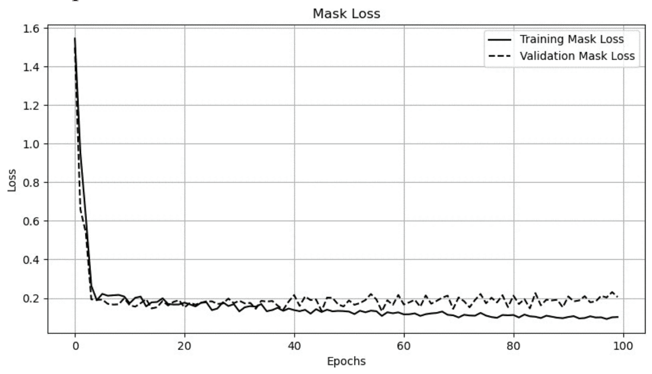
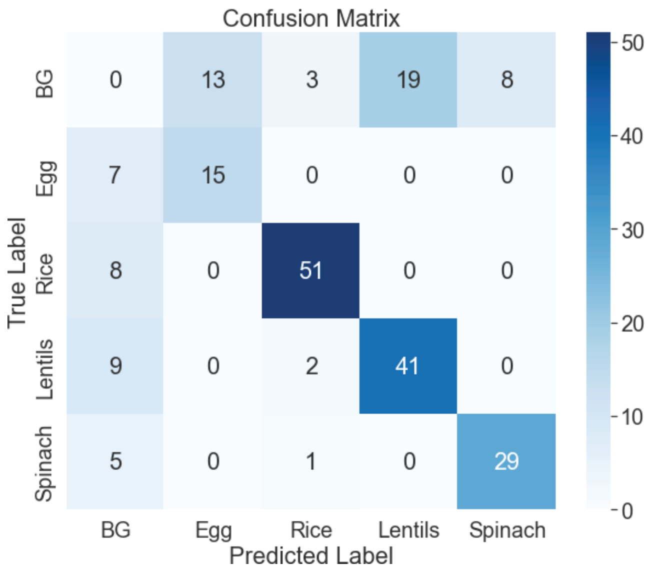


Fig. 14 Mask Losses



True \ Predicted	BG	Egg	Rice	Lentils	Spinach
BG	0	13	3	19	8
Egg	7	15	0	0	0
Rice	8	0	51	0	0
Lentils	9	0	2	41	0
Spinach	5	0	1	0	29

Fig. 15 Confusion Matrix of Trained Model

6) *Confusion Matrix:* The confusion matrix offers a comprehensive analysis of our classification model's performance in discerning food items such as "Egg," "Rice," "Lentils," and "Spinach." Impressively, the model accurately identifies these classes with minimal confusion, indicating its proficiency in distinguishing between them. However, a challenge arises with background food items being misclassified as our target classes, with "Lentils" exhibiting the most pronounced issue—where 19 background items are incorrectly labeled as "Lentils." However, there is notable consistency in the detection of existing food classes, with less confusion observed among them.

7) *Sources of Error:* The sources of errors in the food detection model encompass several factors, notably stemming from a limited dataset of 354 images. The images in our dataset include food items that are not defined in our prediction classes which increases the number of false positives detected by our model, particularly in the background food items.

Furthermore, the system assumes that all the food items are under typically normal conditions and the generalized calorie estimate for each food item is taken. Such as, within the specific class of egg, images of the inclusion of varied images depicting whole eggs alongside cut or halved eggs help the model accurately classify the food as egg, but it causes errors while accurately estimating the calories of a half or whole egg.

Final Result

For the user interface, SmartCal was designed which contains different components such as Home, Register, and Display. The users register their details through the Register component and display the results along with predicted food in the Display component.

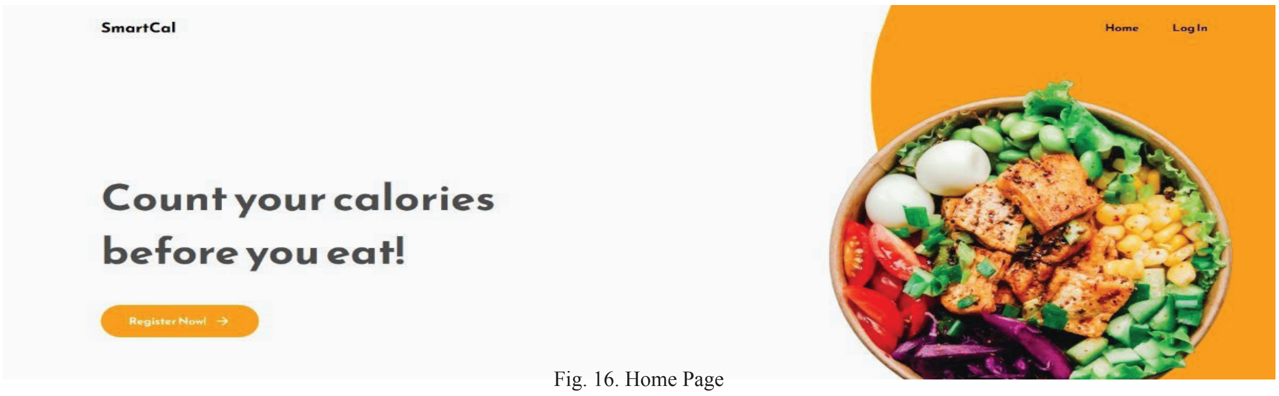
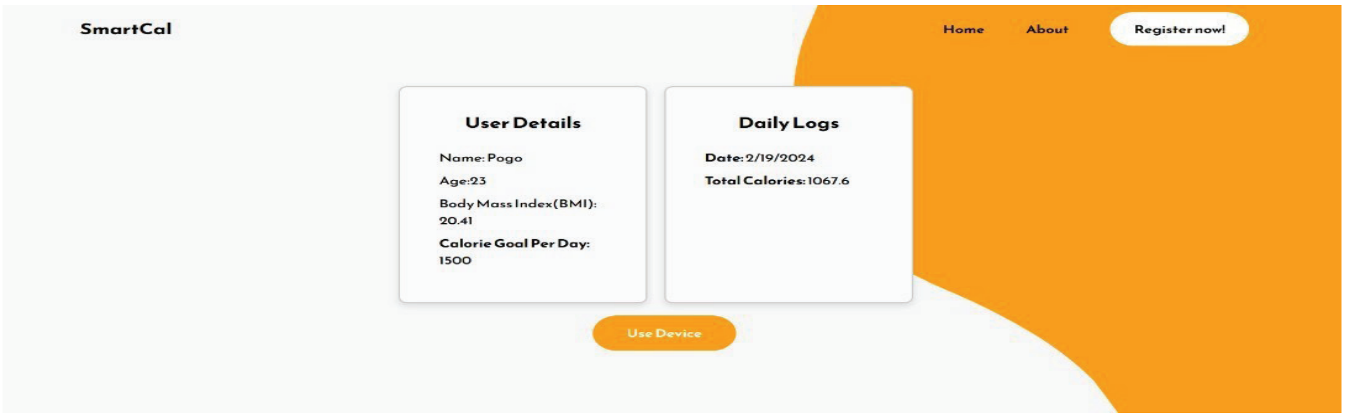


Fig. 16. Home Page



User Details

Name: Pogo

Age:23

Body Mass Index(BMI): 20.41

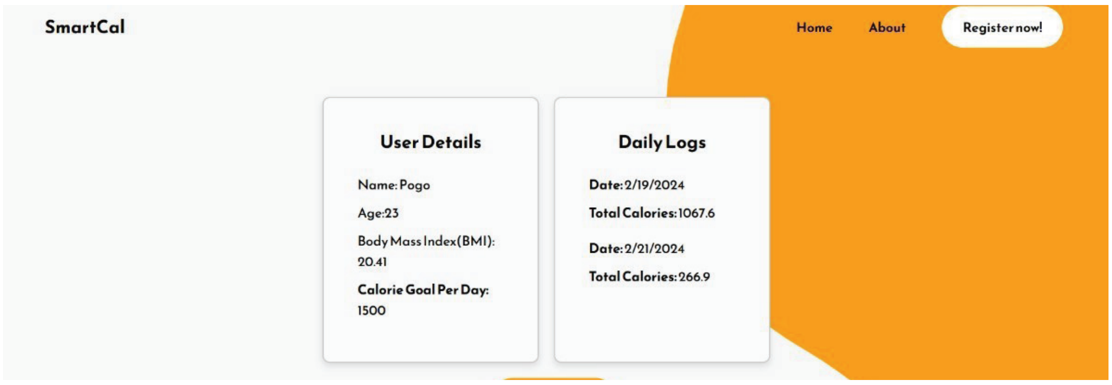
Calorie Goal Per Day: 1500

Daily Logs

Date: 2/19/2024

Total Calories: 1067.6

Fig. 17. Display the page before using the device



User Details

Name: Pogo

Age:23

Body Mass Index(BMI): 20.41

Calorie Goal Per Day: 1500

Daily Logs

Date: 2/19/2024

Total Calories: 1067.6

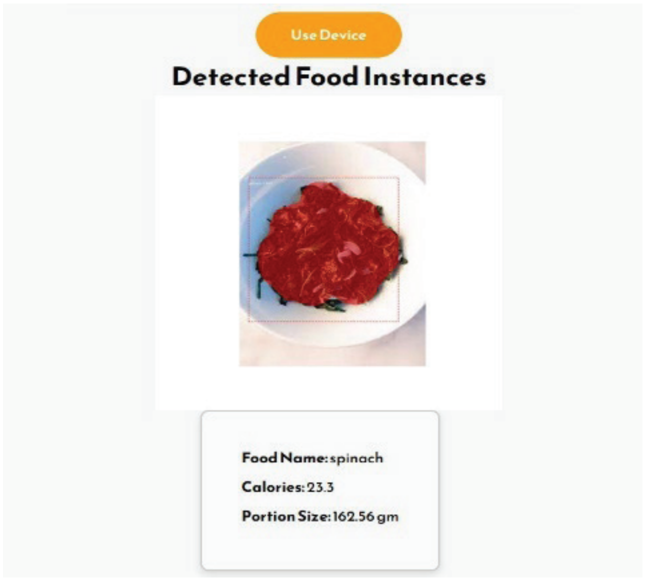
Date: 2/21/2024

Total Calories: 266.9

Fig. 18. Display the page after using the device

Future Enhancement

The future enhancements to the methodology proposed in our paper are to improve the overall performance, accuracy, and flexibility of the system. One of the primary objectives is to expand the food item dataset by including additional categories such as rice, lentils, spinach, and boiled eggs, which will improve the system's ability to detect a wider variety of foods. Moreover, we aim to incorporate a broader range of food categories, considering various geographic locations



Food Name: spinach
Calories: 23.3
Portion Size: 162.56 gm

Fig. 19. Positive Anchors

and cultural cuisines, to account for staple diets as well as more extravagant dishes. Another significant enhancement is to integrate advanced food volume estimation techniques, eliminating the need for fixed containers during calorie detection, which would increase the system's adaptability. We also plan to explore the integration of physical weight detection modules to minimize errors associated with calorie estimation, ensuring more accurate results. Lastly, the adoption of more advanced cameras will be considered to enhance image capture, facilitating more accurate and reliable recognition of food items. These future enhancements will collectively strengthen the methodology, making it a more comprehensive and effective food analysis tool for users.

Conclusion

SmartCal is a project that revolutionizes calorie tracking by automating the process through cutting-edge technologies like Mask R-CNN. The system, incorporating the ESP32-CAM AI Thinker module and NodeJS/ReactJS, streamlines user interaction. It eliminates manual input, making dietary tracking more accurate and convenient. The project finds practical use in personal diet management and holds potential for research and optimizing food portions in restaurants. Our project successfully captures food images, utilizes Mask R-CNN with the F1 score of 0.69 and mAP50 score of 0.99 for food identification, calculates calories, and integrates a user-friendly web interface to display them along with the feature to set personalized daily user calorie goals in real- time.

Acknowledgement

We sincerely thank the Department of Electronics and Computer Engineering, IOE, Thapathali Campus, for providing us with the opportunity to apply our knowledge and skills. We deeply appreciate the equal contributions of every individual to "SmartCal: Calorie Estimation of Local Nepali Cuisine with Deep Learning-Powered Food Detection." Our gratitude extends to the authors of research papers and developers of programming libraries whose invaluable contributions have supported this work. We also acknowledge and respect the dedication of researchers who have shared their knowledge with the community.

We are grateful to our supervisor, Er. Praches Acharya, for his guidance and to everyone who directly or indirectly contributed to this study.

References

[1] K. He, G. Gkioxari, P. Dolla´r, and R. Girshick, "Mask r-cnn," *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[2] AI-Thinker, *ESP32-CAM Datasheet*, 2019. [Online]. Available: https://docs.ai-thinker.com/en/esp32-cam

[3] W. H. Organization, "Obesity and overweight," Jun. 2021, published on 09 June 2021. Accessed: 28 December 2023. [Online]. Available: https://www.who.int/news-room/fact- sheets/detail/obesity-and-overweight

[4] N. B. A. P. Ltd., "New business age," Oct. 2023, accessed: 29 December 2023. [Online]. Available: https://www.newbusinessage.com/Articles/view/19345

[5] F. Romadhon, F. Rahutomo, J. Hariyono, S. M. E. Sulistyo, M. H. Ibrahim, and S. Pramono, "Food image detection system and calorie content estimation using yolo to control calorie intake in the body," in E3S Web of Conferences, vol. 465, 2023.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[7] C. Gao, F. Kong, and J. Tan, "Health aware smart phone systems," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2009.

[8] M. Puri, Z. Zhu, Q. Yu, A. Divakaran, and H. S. Sawhney, "Recognition and volume estimation of food intake using a mobile device," in *IEEE Workshop on Applications of Computer Vision (WACV 2009),* Snowbird, UT, USA, 2009.

[9] M. A. Fischler and R. J. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[10] F. Lo, Y. Sun, J. Qiu, and B. Lo, "Food volume estimation based on deep learning view synthesis from a single depth map," *Nutrients*, vol. 10, no. 12, p. 2005, 2018.

[11] Y. Ando, T. Ege, J. Cho, and K. Yanai, "Depthcaloriecam: A mobile application for volume-based food calorie," in *MADiMa '19: Proceedings of the 5th International Workshop on Multimedia Assisted Dietary Management*, 2019, pp. 76–81.

[12] P. Pouladzadeh, S. Shirmohammadi, and R. Almaghrabi, "Measuring calorie and nutrition from food image," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 8, pp. 1947–1956, 2014.

[13] A. Myers, N. Johnston, V. Rathod, A. K. Balan, A. N. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. Murphy, "Im2calories: Towards an automated mobile vision food diary," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[14] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[15] I. Corporation, "Intel realsense f200 depth camera," 2015. [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and- technology/realsense/overview.html

[16] J. Hu. (2022, 10) Revisiting single image depth estimation toward higher resolution maps. CSDN Blog. [Online]. Available: https://doi.org/10.48550/arXiv.1803.0867

[17] I. OmniVision Technologies, "Ov2640 camera module," 2008. [Online]. Available: https://www.ovt.com/products/OV2640

[18] A. LLC, "Arduino uno," 2021. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno

[19] M. Z. J. C. X. W. Z. Zhou, "Detection and classification of multi- magnetic targets using mask-rcnn," IEEE Access, vol. 8, pp. 187 202– 187 207, 2020.

[20] Nutritionix, "Nutritionix api," 2021. [Online]. Available: https://developer.nutritionix.com/

[21] Matterport, "Mask r-cnn," 2017. [Online]. Available: https://github.com/matterport/Mask RCNN

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dolla´r, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[23] V. I. A. Team, "Vgg image annotator," 2021. [Online]. Available: https://www.robots.ox.ac.uk/ vgg/software/via/