Light-Weight Intelligent Egyptian Food Detector For Diabetes Management

Menna-allah Sayed[1], Aya Saafan[1], Salma Zakzouk[1], Mustafa A. Elattar[2], and M. Saeed Darweesh[1,3]

[1]School of Engineering and Applied Sciences, Nile University, Giza 12677, Egypt
[2]Center for Informatics Science (CIS), Nile University, Giza 12677, Egypt
[3]Wireless Intelligent Networks Center (WINC), Nile University, Giza 12677, Egypt

2022 18th International Computer Engineering Conference (ICENCO) | 978-1-6654-5895-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICENCO55801.2022.10032514

***Abstract*—Diabetic patients need a management tool that combines multiple features and tracks and views detailed data time-efficiently. Effective food logging is an important element of health monitoring. In this paper, we propose "Suger.ly", a lightweight mobile application with artificial intelligence food recognition for diabetes management. The system has been trained to recognize 101 distinct types of food, with a focus on Egyptian cuisine. The app can then get nutritional value and insulin calculations. The results obtained from the Single-Shot multibox Detection (SSD) MobileNet-V1 food detection model localization process, and a separate single-plates classifier achieved a mean average precision (mAP) of 0.592 and mean average recall (mAR) of 0.556. The average response time for the food object detection model in the case of a single food item detection is 74 ms.**

Keywords—Diabetes Management, Food Detection, SSD Mobilenet-V1, Egyptian Cuisine, Lightweight Mobile Application.

I. INTRODUCTION

Food is an important part of everyday life in all cultures and has a wide range of effects on people's diets and eating habits and general health. Improper eating habits frequently contribute to people being overweight or obese, which is connected to chronic illnesses such as diabetes and cardiovascular disease. This paper provides a method for food consumption tracking to not only assist people to prevent illnesses, but also to help those suffering from these disorders manage their health more effectively. Sugar.ly, the proposed mobile application, offers a solution to diabetes management based on a food-image recognition logging system along with other supporting features. The traditional approach to monitoring diets of maintaining a manual food journal can be tedious, time-consuming, and does not encourage adherence. Sugar.ly exploits the convenience of smartphones and the widespread social trend of capturing food images to overcome the limitations of traditional food logging.

Deep learning models have shown promising results for image-based food recognition and classification on a larger and more complex dataset. In response to the growing reliance on smart devices, Sahoo et al. offer a smart-food logging mobile application: FoodAI [1]. The FoodAI model was trained on the SGFOOD dataset, containing 400,000 food photos divided into 756 categories. Food products are the most popular in Singapore, and the dataset is heavily skewed. The model was created by fine-tuning pre-trained ImageNet models and applying focal loss to boost performance. SENet and ResNeXt-50 combination produces the best results, with top-1 accuracy of 80.86% and top-5 accuracy of 95.61%. The FoodAI model is available via the Healthy 365 mobile app as a RESTful API web service.

In 2019, Merchant and Pande in [2] used a CNN network on a mobile application to classify the input food and calculates its nutrients. The used dataset for food images was Food 101. they used Inception V3, and they got an accuracy of 70%. The time needed in the android application for classifying was 5s, and the size of the application was 6.9 megabytes.

In 2020, Ramesh et al. [3] created an application that automatically detects and locates food items in real-time scenarios. A dataset was gathered from different internet sources and manually annotated using label-IMG to train an SSD configuration. An XML file stores bounding box coordinates in the image for object detection. Additionally, data augmentation expands the size of the data. The model was initialized with a pretrained checkpoint trained for the COCO dataset. The object identification model and different CNN architectures were paired with SSD, with InceptionV2 partnered with SSD being the most efficient way.

Furthermore, each sample evaluated is an image with a corresponding label XML file, and the dataset is split into two sections: 80% train and 20% test. The XML files were concatenated into a single CSV file for train data and another for the test data. The images and CSV files are used to make two TFRecords and a Label-map. The object identification model and different CNN architectures were paired with SSD, with InceptionV2 partnered with SSD being the most efficient way. The trained model is exported to form an inference graph, and the weights are dumped to a checkpoint file to validate the model's accuracy and build up a prediction environment. Finally, SSD MobileNetV1 and SSD InceptionV2 are trained to convergence, with MobileNetV2 converging at 60,000 and InceptionV2 at 80,000. InceptionV2 outperformed MobileNetV1, such that the total loss for InceptionV2 and MobileNetV1 were 2.15 and 2.903, respectively. Thus, the model's accuracy reaches 97.6%.

Authors in [4] developed a modified version of the MobileNet architecture for classifying foods, using a pretrained model and removing the final three layers before adding layers for global average pooling, batch normalization, ReLU, dropout, and softmax. The suggested MobileNet architecture outperforms competing approaches when used in conjunction with data augmentation strategies. With model settings of 50 iterations, 16 batch size, 0.0001 learning rate, SGD optimizer, and transfer learning. With the sub-group of 40400 images having the highest accuracy results compared to smaller subgroups with a 72.59 %, using many food images increase the identification performance.

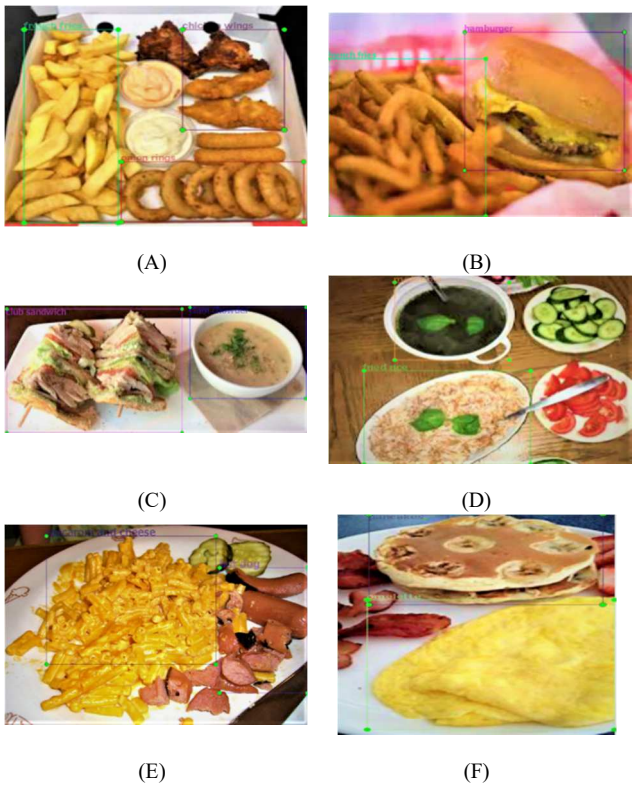
This paper is divided into: Section II presenting the methodology, dataset specifications, images preprocessing, and the SSD MobileNet-V1 architecture and hyperparameters. The results are discussed in Section III, and finally, Section IV discusses the conclusions and future work.

II. METHODOLOGY

This section discusses the detailed methodology, consisting of the dataset specification, images preprocessing, and SSD MobileNet-V1 architecture and hyperparameters.

A. Dataset specification and images preprocessing

In reality, most of the time, food images have plenty of food items, not just a single food item, especially in Egyptian cuisine. Therefore, a dataset of mixed plates was self-collected from various search engines and online sources, including social media, to ensure numerous combinations of the food items. The mixed plates dataset contains all the 101 food items from a modified Food-101 dataset. It consists of 2200 images, divided evenly into 22 labels. The images were classified into labels rather than just one folder to ensure that the dataset is balanced as much as possible and not biased towards a certain combination of food items and also has enough data to train the images on certain food combinations. The images were annotated and saved to XML files using "LabelImg" [5], a graphical image annotation tool that allows users to draw and label picture bounding boxes. The bounding box enables the detection of multiple dishes and food items in one image, as shown in Fig. 1. Not to mention, they isolate the food item from any unnecessary background (Non-food items), as shown in Fig. 2. Moreover, as SSD MobileNet-V1 requires the dataset to be in the TesnorFlow Record (TFRecord) format, The dataset had been uploaded on Roboflow for this conversion.



(A) (B) (C) (D) (E) (F)

Fig. 1. Samples of annotated mixed food plates

The dataset images are scaled to 224×224 to be consistent with the MobileNet architecture. Moreover, data augmentation techniques were implemented to increase the number of images to become 5486 images. This helps in generalizing the proposed model and make it more robust. The applied data augmentation techniques were: Horizontal flip, vertical flip, rotation, random brightness, random saturation, random exposure, and shear. These data augmentation techniques were specifically chosen because they illustrate most of the image types that a user can capture. Finally, the self-collected mixed plates dataset is divided to 75% training and 25% testing.

Fig. 2. Isolation of single food item example "Koshary"

B. SSD MobileNet-V1 architecture & hyperparameters

Single Shot Multi-Box Detector (SSD) was presented in 2016 to implement an object detection model using a single deep neural network combining regional proposals and feature extraction [6]. The main idea of SSD is to predict class scores and box offsets for a specified set of default bounding boxes by applying small convolutional filters to feature maps.

As shown in Fig. 3 [6], SSD network architecture has a base network in the convolution layer Con4_3 which consists of a pretrained VGG16 network on ImageNet truncated before the last classification layer to extract high-level features, then passing through the next two convolutional layers Conv_6 and Conv_7. Finally, the network is extended by adding convolution feature layers of decreasing resolutions Conv8_2, Conv9_2, Conv10_2 and Conv11_2 [7]. After all, the SSD network results in 8732 bounding boxes to be followed by a non-maximum suppression step to produce the final detections.

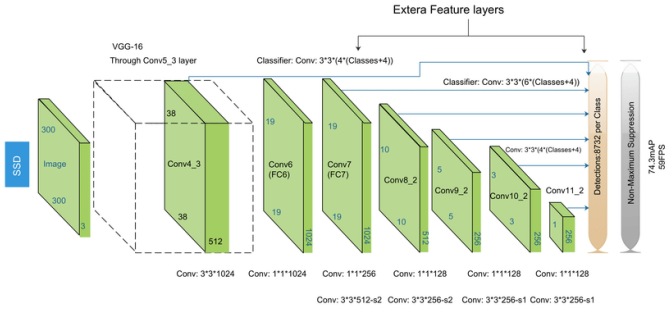


Fig. 3. SSD network basic architecture [5]

The model loss is calculated by a weighted sum between localization loss and confidence loss (1) [6]. localization loss is calculated using the Smooth L1 loss method while the confidence loss is calculated by the softmax loss method, where N is the number of matched default boxes, $Lconf$ is

the confidence loss, $Lloc$ is the localization loss, and α is the weight term with default value equals 1.

$$Loss = \frac{1}{N}\left(L_{conf} + \alpha\, L_{loc}\right) \quad (1)$$

SSD MobileNet-V1 pretrained model on COCO dataset [8] had been chosen to be the base model. The MobileNet-V1 classifier's hyperparameters used were batch size of 64, 45 epochs, initial learning rate of 0.01 and SGD optimizer was used. Furthermore, this paper used the modified MobileNet architecture in reference [4] and was trained Food-101 dataset [9], with 6 biased mixed plate dishes being replaced with Egyptian food plates. While the Single Shot Detector (SSD) hyperparameters were been as default values. The number of training steps was set to 100,000 steps. The training spent around 10 hours using GPU resources from Colab Pro.

C. Mobile Application Architecture and Deployment

Suger.ly provides image food search, reducing the difficulties of a manual food journal, along with nutrition and insulin calculations. The features include food recognition, nutritional values and recipes, carb counting, tracking blood glucose levels, giving reports on blood glucose levels over weeks, months, and even years, and keeping medical information organized.

A first-time user of the mobile application should see the boarding splash screen, followed by profile set-up screens. If the user has not registered, they should be able to fill in profile information. If the user is not a first-time user, they should be able to see the home screen directly when the application is opened. Here, users can see a summary of their glucose levels and meals. By clicking on the glucose levels summary, the user can access their glucose levels log where they can also download PDF reports. The floating action button on the home page allows the user to quickly log their insulin, glucose, and medicine and mark reminders as completed. The user can search meal nutritional values by text or image, view results, and save meals. Once the food item is identified, the mobile application will provide the needed nutritional facts by incorporating the "Nutritionix" database API [10]. The calendar screen shows all the user's reminders; the user can add more reminders, mark them as completed, or delete them. Users can also save a list of their medicine.

The application is implemented using Flutter for cross-platform design and performance. Flutter applications have a performance that is mostly indistinguishable from native app performance in most cases and even better with complex UI. The app uses the tflite plugin [11] to load the food detection model. Tflite is a flutter plugin for accessing TensorFlow Lite API that supports image classification, and object detection on both iOS and Android.

III. Results

The section evaluates the results of the implemented SSD MobileNet-V1 on the self-collected dataset to detect multiple the food items.

Comparing the implemented SSD MobileNet-V1 accuracy results with previous studies is not reliable. The reason behind that is the dataset nature. In this paper, 101 classes of food had been trained by only 1000 images (about 10 images per class) due to the lack of computational power and self-annotation effort. The accuracy can be much enhanced by increasing the dataset size to represent each class more accurately. Moreover, the previous studies mentioned in the literature review did not address Egyptian cuisines which is considered the main contribution in the paper. Better evaluation metrics are *mean average precision (mAP)* and *mean average recall (mAR).*

A. Model Performance

To decide whether the predicted box with its class is a true prediction or not, there were two metrics to be considered: confidence score and Intersection over union (IoU) [11]. The confidence threshold is the probability that the predicted box contains a food item. Intersection over union (IoU) is calculated as shown in (2) [12] by dividing the intersection area of the predicted box area and the ground truth box area over the union area of the predicted box area and the ground truth box area.

$$IoU = \frac{area\left(B_p \cap B_{gt}\right)}{area\left(B_p \cup B_{gt}\right)} \quad (2)$$

By setting a threshold for the confidence score and Intersection over union, the confusion matrix of the classes is calculated. Using this confusion matrix, there are two performance metrics have been selected to determine the performance of the object detection model which were *mean average precision (mAP)* and *mean average recall (mAR).*

In (3), The prediction precision refers to the number of times that the class is really in the predicted box among the predictions that classified the class to be this food class.

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

By calculating the precision of each class on several thresholds, the average of precision records has been obtained. The calculation of the AP includes records of one class, therefore, the mean average precision in (4) [12] is calculated by getting the mean of the average precision records over all classes.

$$mAP = \frac{\sum_{i=1}^{k} AP_i}{k} \quad (4)$$

In (5), The prediction recall refers to the number of times that the class is predicted correctly among the number of times that were supposed to be obtained as found in the ground truth boxes. The same as precision, the mean average recall in (6) [12] was calculated by getting the mean of the average recall records of all classes.

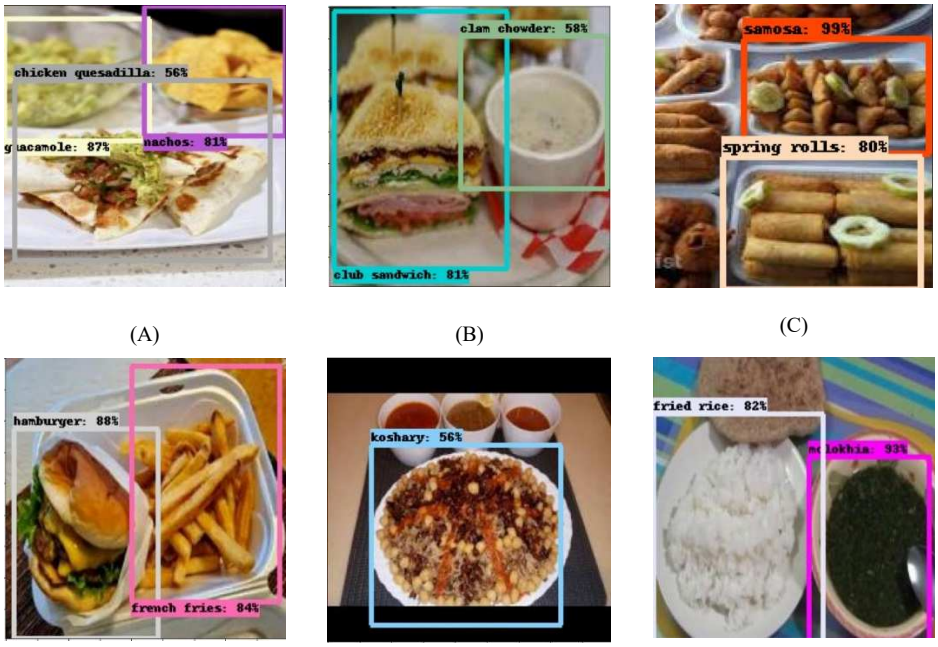
$$Recall = \frac{TP}{TP+FN} \quad (5)$$

$$mAR = \frac{\sum_{i=1}^{k} AP_i}{k} \quad (6)$$

B. Model Evaluation

By running an inference test from some images from the testing dataset, Fig. 4 shows some examples of the testing dataset with classification results.

Table 1 summarizes the mean precision and Table 2 summarizes the recall of all classes under different confidence score thresholds, and different IoU thresholds.



(A) (B) (C)

(D) (E) (F)

Fig. 4. Sample of food detection model confidence results

TABLE 1. Mean precision results of SSD-MobileNet-V2 object detection model

		Confidence Score Threshold								
		0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
IoU Threshold	**0.1**	0.195	0.222	0.223	0.243	0.266	0.265	0.296	0.299	0.352
	0.2	0.205	0.227	0.236	0.256	0.278	0.275	0.308	0.336	0.407
	0.3	0.219	0.247	0.268	0.291	0.320	0.326	0.323	0.346	0.407
	0. 4	0.225	0.257	0.288	0.3	0.323	0.336	0.323	0.344	0.401
	0.5	0.207	0.252	0.307	0.344	0.336	0.38	0.385	0.381	0.481
	0.6	0.207	0.240	0.313	0.313	0.309	0.339	0.368	0.388	0.438
	0.7	0.184	0.248	0.279	0.316	0.297	0.34	0.348	0.375	0.416
	0.8	0.264	0.196	0.190	0.139	0.167	0.188	0.167	0.222	0.5
	0.9	0.292	0.143	0.143	0.142	0.143	0.0	0.0	0.0	0.0

TABLE 2. Recall results of SSD-MobileNet-V2 object detection model

		Confidence Score Threshold								
		0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
IoU Threshold	**0.1**	0.127	0.149	0.162	0.172	0.201	0.219	0.245	0.25	0.296
	0.2	0.137	0.165	0.175	0.186	0.217	0.236	0.267	0.280	0.357
	0.3	0.153	0.186	0.193	0.201	0.23	0.248	0.28	0.293	0.377
	0. 4	0.163	0.186	0.195	0.201	0.225	0.244	0.257	0.274	0.357
	0.5	0.151	0.182	0.211	0.24	0.263	0.288	0.315	0.333	0.422
	0.6	0.199	0.225	0.266	0.288	0.291	0.308	0.347	0.360	0.417
	0.7	0.213	0.265	0.289	0.291	0.304	0.321	0.319	0.318	0.417
	0.8	0.278	0.208	0.190	0.125	0.167	0.179	0.167	0.15	0.5
	0.9	0.25	0.071	0.071	0.071	0.071	0.0	0.0	0.0	0.0

C. *Suger.ly Mobile Application*

Using TensorFlow Lite to deploy the object detection model on edge devices offers a lightweight performance, compared to other resource-intensive deep learning models. Furthermore, deep Learning models at the edge devices have low latency and optimal power consumption inferences that don't require network connectivity. Also, there is no concern for data privacy since the inferences are made on the device without shared data across the network. Fig. 5 shows an example of the food detection and recognition results in Suger.ly app.

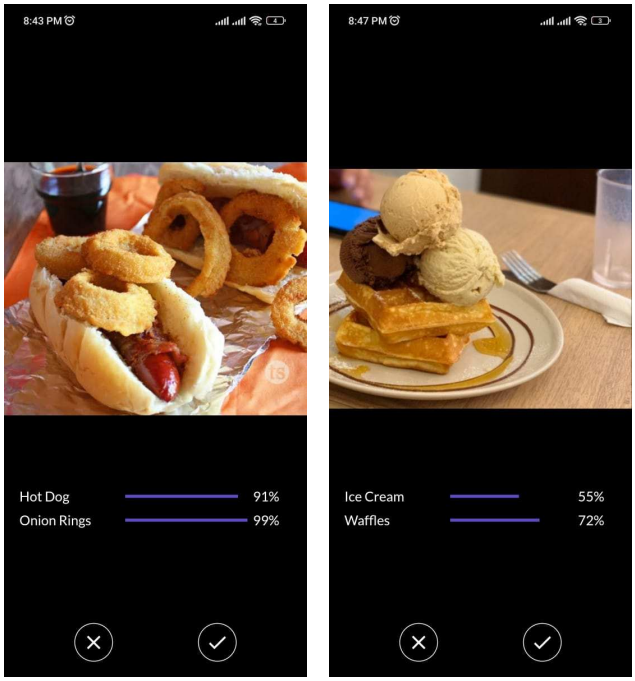


Fig. 5. Food image search in Suger.ly app

The average total response time for the bounding boxes detection is 74 ms in case of a single food item detection classification model. Not to mention, 53 ms for image preprocessing, and 69 ms for separate food recognition. Table 3 shows the performance of the proposed mobile application compared to related work. With a sample of 100 images and a benchmark response time of 2 sec, the food recognition feature gives an excellent apdex score of 0.96.

TABLE 3. Performance of the proposed mobile application compared to related work.

Application	Dataset	Response time	App size
ConvFood [2]	Food-101	5 sec	6.9 MB
Healthy 365 **Error! Reference source not found.**]	SGFOOD	-	53.9 MB
The proposed mobile application "Sugar.ly"	Modified Food-101	337 ms	255 MB

IV. CONCLUSION & FUTURE WORK

Patients with diabetes require a management solution that integrates several functions, tracks detailed data, and allows for quick data viewing. A crucial part of health monitoring is accurate dietary logging. In this research paper, "Suger.ly," is suggested, a simple mobile application for managing diabetes that uses artificial intelligence to identify foods. With an emphasis on Egyptian cuisine, the system has been trained to recognize 101 different types of food. A separate single-plates classifier and the SSD MobileNet-V1 food detection model localization procedure combined to produce final findings with a mAP and mAR of 0.592 and 0.556, respectively. When only one food item is detected, the food object detection model typically responds in 74 ms. For future work, it is decided to improve the food detection and recognition feature by collecting more images for the dataset, not to mention integrating more Egyptian plates for the proposed mobile application. Also, new features such as: Multilingual version of the application and portion size estimation for the food recognition features are planned to be added.

REFERENCES

[1] Sahoo, Doyen & Hao, Wang & Ke, Shu & Xiongwei, Wu & Le, Hung & Achananuparp, Palakorn & Lim, Ee-Peng & Hoi, Steven, "FoodAI: Food Image Recognition via Deep Learning for Smart Food Logging," in *arXiv:1909.11946v1*, 2019.

[2] Kaiz Merchant and Yash Pande. "ConvFood: A CNN-Based Food Recognition Mobile Application for Obese and Diabetic Patients". In: Emerging Research in Computing, Information, Communication and Applications. Ed. by N. R. Shetty et al. Singapore: Springer Singapore, 2019, pp. 493–502.ISBN: 978-981-13-5953-8.

[3] Abhinaav Ramesh, Aswath Sivakumar, and S Sherly Angel. "Real-time Food-Object Detection and Localization for Indian Cuisines using Deep Neural Networks," in *IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT)*, pp. 1-6, 2020. doi:10.1109/ICMLANT50963.2020.9355987.

[4] S.Phiphitphatphaisit and O.Surinta, "Food Image Classification with Improved MobileNet Architecture and Data Augmentation," in *3rd International Conference on Information Science and Systems (ICISS 2020)*, 2020.

[5] Tzutalin, "Tzutalin/labelimg: LabelImg is a graphical image annotation tool and label object bounding boxes in images," *GitHub*. [Online]. Available: https://github.com/tzutalin/labelImg. [Accessed: 31-May-2022].

[6] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C., "SSD: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision*, pp. 21–37, October 2016.

[7] Salma Zakzouk, Aya Saafan, Menna-Allah Sayed, Mustafa A. Elattar and M. Saeed Darweesh, "Light-Weight Food/Non-Food Classifier for Real-Time Applications," in *the 4th Novel Intelligent and Leading Emerging Sciences Conference (NILES 2022)*, Cairo, Egypt, 2022.

[8] "Object detection: tensorflow hub," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/hub/tutorials/object_detection. [Accessed: 17-Jun-2022].

[9] Kaggle, "Food 101", 2022, [online] Available: https://www.kaggle.com/dansbecker/food-101

[10] "Nutrition & Exercise API," Nutritionix. [Online]. Available: https://www.nutritionix.com/business/api. [Accessed: 07-Jan-2022].

[11] "Tflite: Flutter Package," *Dart packages*, 13-Apr-2021. [Online]. Available: https://pub.dev/packages/tflite. [Accessed: 21-Jan-2022].

[12] Tensorflow, "Models/ssd_mobilenet_v1_coco.config at master · Tensorflow/models," *GitHub*, 13-Jun-2018. [Online]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/samples/configs/ssd_mobilenet_v1_coco.config. [Accessed: 18-Jul-2022].