Self-Service Check-Out

Martina Dallari, Silvia Giovanardi, Ayoub Imad
University of Modena and Reggio Emilia
{284999,284444,285322}@studenti.unimore.it

Abstract

While self-checkout systems are commonly utilized in retail, their influence on the food service sector, specifically corporate catering, remains minimal. This study suggests a viable approach by introducing a self-service checkout system aimed at optimizing self-canteen operations and improving the overall customer experience.

1 Introduction

This paper presents a self-checkout system for canteens using deep learning for automatic food recognition. It includes an object detection network to identify food items and a classification network to recognize them based on a daily menu. The system handles varying numbers of food items by generating embeddings for detected objects and matching them to a menu database, enabling the classification of different food categories without prior knowledge of the number of food categories. This combination supports the self-checkout process in real-world cafeteria settings.

2 Data Retrieval

This section describes the components and processes employed to obtain the data needed for training.

2.1 Food2k Dataset

The Food2k dataset [1] is a comprehensive resource used for tasks in food image recognition and classification. It includes 2000 different food categories. The categories cover well-known foods such as pasta and burgers as well as sweets like cheesecake and tiramisu. This dataset serves as the primary resource for training all models within the system. Every image in the dataset is accompanied by an annotation specifying its label.

Lasagna *Tiramisù* *Pizza*

Figure 1: Example of Food101 images

2.2 UNIMIB2016 Dataset

The UNIMIB2016 [2] dataset includes 1,027 images taken in a real canteen environment, featuring 73 different food categories and 3,616 individual food instances. These images show a variety of foods placed on trays, some on placemats instead of plates, often accompanied by side dishes along with the main courses. This dataset served as a test bed for the final system, reflecting real-world conditions. The dataset includes annotations for each dish, specify-

ing its bounding box and label, thereby offering ground truth for the final evaluation.

Tray 1

Tray 2

Figure 2: Example of UNIMIB2016 images

2.3 Custom Food101 Dataset

This dataset is a modified version of the Food101 [3] dataset, enriched with bounding box annotations for 101 food categories. It comprises 28,000 annotated images, divided into 19,600 images for training, 5,600 for testing, and 2,800 for validation. The inclusion of bounding box annotations enhances its suitability for food detection tasks.

Annotation 1

Annotation 2

Figure 3: Example of annotations

3 Image processing

Images naturally contain noise, which is primarily introduced by sensors and, more broadly, the acquisition systems. Training a model on noisy data can lead to decreased performance, making it a poor practice. To minimize noise and subsequently train the models with cleaner data, a bilateral filter [4] was selected. This approach determines the pixel value at the output by computing a weighted sum of the surrounding pixel values.

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

The weight factor $w(i,j,k,l)$ is determined by the product of a spatial domain kernel, $d(i,j,k,l)$, and a range kernel based on data intensity, $r(i,j,k,l)$. The domain kernel is defined as:

$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right)$$

whereas the range kernel evaluates intensity similarity relative to the center pixel:

$$r(i,j,k,l) = \exp\left(-\frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right)$$

Thus, the bilateral weight function results from the multiplication of these kernels,

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right)$$

Original

Filtered

Figure 4: Example of Bilateral Filter

The domain kernel follows a Gaussian distribution, while the range kernel assesses appearance similarity to the central pixel, leading to a bilateral filter kernel that is the product of these components, which removes noise, blurs images, and preserves edges.

Furthermore, to enhance the model's generalization abilities, we implemented

Figure 6: Example of transformed data

data augmentation methods. These included geometric transformations like rotation, translation, zooming, cropping, and horizontal flipping. Finally, random adjustments to brightness and contrast were employed to improve resilience to different lighting environments.

4 Food detection

To detect food items on the tray, we fine-tuned a Faster R-CNN [5] using the custom dataset. The model was initialized with weights pre-trained on the COCO dataset [6] and it uses a ResNet-50 architecture [7] as the backbone for feature extraction. By fine-tuning the model on the dataset, we adapted it to distinguish food items from the background.

4.1 Faster-RCNN

Faster R-CNN is a two-stage object detection model. In the first stage, a Region Proposal Network (RPN) generates region proposals that are likely to contain objects. The RPN operates on a convolutional feature map, predicting objectness scores and bounding box coordinates for anchors at various locations. In the second stage, the proposed regions are refined and classified. Each region is cropped from the feature map using a RoI Pooling layer, producing fixed-size feature representations. These features are processed by fully connected layers to predict class labels and refine the bounding box coordinates.

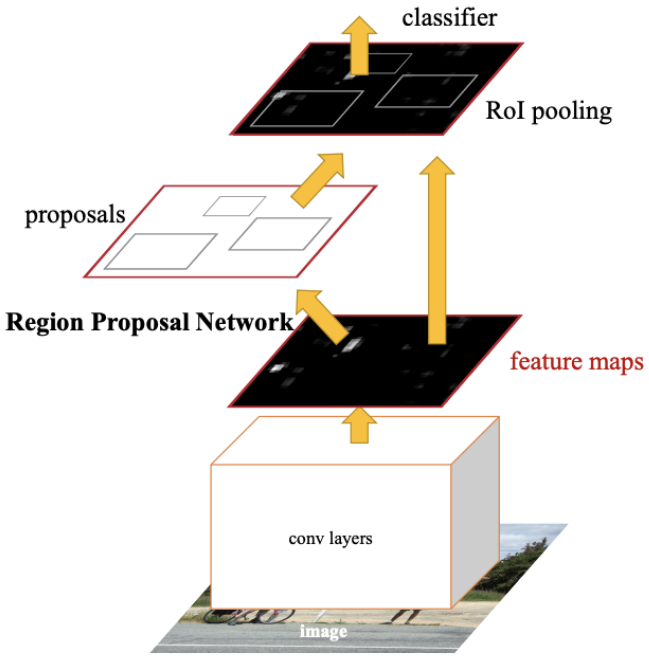


Figure 5: Faster-RCNN architecture

4.2 Fine tuning

During this phase the model was fine-tuned for a maximum of 100 epochs, with early stopping applied using a patience of 20 epochs. After early stopping, the best weights were restored based on performance on the validation set. For optimization, we used the Adam optimizer with a learning rate of 0.001, as it is widely regarded as a versatile choice for diverse problems and model complexities.

4.3 Evaluation

To evaluate the performance of the object detector, we used Average Precision (AP) [8], which is based on the precision-recall curve. This curve shows the trade-off between precision and recall at different confidence levels of the bounding boxes.

4.3.1 Intersection over union

To define what is a "correct detection" and an "incorrect detection", we used Intersection over Union (IoU) which measures how much the predicted bounding box B_p overlaps with the ground-truth bounding box B_{gt}. It is calculated as the ratio of the intersection area to the union area of the two boxes:

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$$

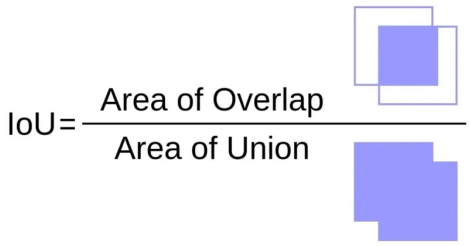


Figure 6: IoU

Precision and Recall are defined as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Where:

- **True Positive (TP)**: A correct detection of a ground-truth bounding box.
- **False Positive (FP)**: An incorrect detection of a non-existent object or a wrong detection of an existing object.
- **False Negative (FN)**: A ground-truth bounding box that was not detected.

4.3.2 Avarage precision

By comparing the IoU with a set threshold t, we can decide whether a detection is correct or incorrect.

An object detector is considered good if its precision remains high while its recall increases. This means that even if the confidence threshold changes, both precision and recall should stay high. Therefore, a large area under the precision-recall curve usually shows that the detector has both high precision and high recall. In the literature, there are two main methods for calculating the Average Precision (AP): 11-point interpolation and all-point interpolation. In the 11-point interpolation method, precision values are computed at 11 recall levels, ranging from 0.0 to 1.0 with an increment of 0.1:

$$AP_{11} = \frac{1}{11} \sum_{R \in \{0, 0.1, 0.2, \ldots, 1\}} P_{\text{interp}}(R)$$

where:

$$P_{\text{interp}}(R) = \max_{\tilde{R} \geq R} P(\tilde{R})$$

In all-point interpolation, the interpolation is performed across all recall points:

$$AP_{\text{all}} = \sum_{n} (R_{n+1} - R_n) P_{\text{interp}}(R_{n+1})$$

where:

$$P_{\text{interp}}(R_{n+1}) = \max_{\tilde{R} \geq R_{n+1}} P(\tilde{R})$$

In our experiment, we opted for this latter interpolation method, achieving average precision (AP) scores of AP50 at 0.88 and AP75 at 0.79 on the Custom Food-101 test set.

5 Embedding Generation

Since the daily menu varies over time, it is not feasible to predetermine the number of classes and train a single object detector.

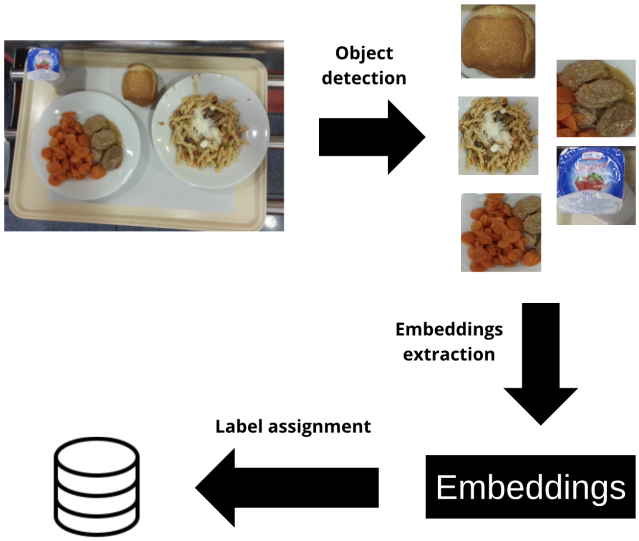


Figure 7: Pipeline

To overcome this challenge, we adopted the same strategy employed by FaceNet [9]. More in detail, we trained a deep convolutional network to generate an embedding $f(x)$ from an image x, ensuring that images from the same food category are mapped to points that are close together in the embedding space. The deep architecture consists of a Siamese network with a ResNet-50 backbone [7], followed by L2 regularization, which maps the embedding points onto a hypersphere of radius 1.

Figure 8: Architecture

The training process leverages the Triplet Loss, which minimizes the distance between an anchor and a positive sample (belonging to the same class) while maximizing the distance between the anchor and a negative sample (belonging to a different class).

Triplet loss:

$$L = \sum_{i=1}^{N} \left[\|f(x_a^i) - f(x_p^i)\|_2^2 - \|f(x_a^i) - f(x_n^i)\|_2^2 + \alpha \right]_+$$

Where:

- x_a^i is the anchor sample.
- x_p^i is the positive sample (same class as the anchor).
- x_n^i is the negative sample (different class from the anchor).
- $f(x)$ is the embedding function.
- $\| \cdot \|_2^2$ represents the squared Euclidean distance.
- α is the margin that enforces a minimum distance between positive and negative pairs.
- $[\cdot]_+$ denotes the hinge function, which outputs the value inside the brackets if it's greater than zero, otherwise zero.

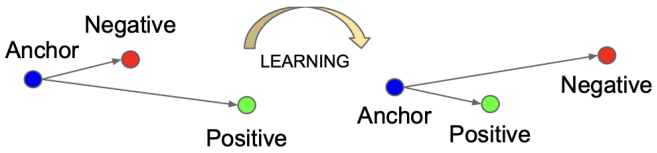


Figure 8: Triplet loss

5.1 Triplets generation and training

One potential approach to generating triplets is to create all possible combinations. However, many of these triplets would have minimal impact on training, as they are easily satisfied and do not contribute meaningfully to improving the model. This can result in slower convergence during training. To address this issue, two main strategies for triplet mining are outlined in [9]: online and offline mining. In this work, we adopted the online

triplet generation strategy, wherein meaningful triplets are dynamically created using the data within the current mini-batch.

During training, the network was trained on the Food2k dataset for a maximum of 100 epochs, employing early stopping based on the validation set. We utilized a margin of 0.85 and focused on semi-hard triplets, those where the negative sample is farther from the anchor than the positive, but still within the margin.

5.2 Evaluation

We evaluated the performance of our Siamese network using a triplet-based approach to assess its ability to differentiate between similar and dissimilar image pairs. The evaluation was conducted on a dedicated test set consisting of image triplets, where each triplet included an anchor image, a positive image from the same class, and a negative image from a different class. To ensure compatibility with the network architecture, the images were preprocessed through resizing, normalization, and transformation into triplet datasets.

The trained model generated feature embeddings for each image, mapping them into a high-dimensional space. Within this space, Euclidean distances were calculated between anchor-positive and anchor-negative pairs. A prediction was considered correct if the distance between the anchor and positive embeddings was smaller than the distance between the anchor and negative embeddings. Model accuracy was computed as the ratio of correct predictions to the total number of samples, providing a measure of the network's ability to learn discriminative embeddings.

Our approach achieved an accuracy of 98% on the test set, demonstrating the model's effectiveness in distinguishing between similar and dissimilar images. Additionally, we measured the mean Euclidean distances for anchor-positive and anchor-negative pairs, which were 0.64 and 0.90, respectively, further highlighting the model's ability to separate embeddings in feature space.

6 Experiment

The final evaluation was conducted on the UNIMIB2016 dataset using a two-stage approach: object detection followed by classification. A Faster R-CNN model was employed to extract the Regions of Interest (RoIs) associated with food items on the tray. For classification, a siamese network was used as a feature extractor to generate embeddings. Each possible food class was represented by a single embedding, extracted from a reference image stored in a database that served as the day's menu. The system retrieved the correct label for each detected food item by comparing its embedding to the database using Euclidean distance.

The system's performance was evaluated using two key metrics: image-level accuracy and object-level accuracy.

- **Image-level accuracy**: An image was considered correctly classified only if all objects within it were accurately identified and categorized. In other words, a customer's tray was correctly classified if, and only if, all the food items on it were classified correctly.
- **Object-level accuracy**: This was calculated as the ratio of correctly classified objects to the total number of objects, representing the number of food items correctly classified divided by the

total number of food items.

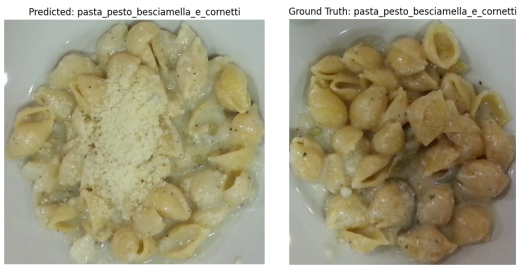
The system achieved the results showed in Table 2.

Metric	Value
Trays Accuracy	22.95%
Correct Trays	235
Total Trays	1024
Food Item Accuracy	68.99%
Correct Items	2483
Total Items	3599

Table 2: System performances

Despite the promising results for individual food item detection (68.99% accuracy), the overall tray accuracy was considerably lower (22.95%), primarily due to the consistent presence of detected items on the tray that were not part of the legitimate set of expected objects.

The following images show an example of matched food item from the tray on the left and the image retrieved from the database using Euclidean distance on the right.



7 Conclusions

This study presents a self-checkout system for canteens that leverages deep learning techniques for food recognition and classification. By combining Faster R-CNN for object detection with a Siamese network for embedding generation, we successfully built a system that adapts to varying food menus without requiring prior knowledge of the number of food categories.

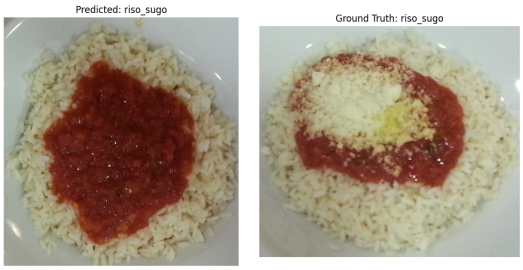


Figure 9: Matched example

In future improvements instead of using an object detection model, a segmentation model could be employed to further improve the accuracy of food recognition, particularly in complex or overlapping scenarios

References

[1] Weiqing Min, Zhiling Wang, Yuxin Liu, Mengjiang Luo, Liping Kang, Xiaoming Wei, Xiaolin Wei, and Shuqiang Jiang. Large scale visual food recognition. *arXiv preprint arXiv:2103.16107*, 2021.

[2] Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. Food recognition: a new dataset, experiments and results. *IEEE Journal of Biomedical and Health Informatics*, 21(3):588–598, 2017.

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV*

2014, pages 446–461, Cham, 2014. Springer International Publishing.

[4] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846, 1998.

[5] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[8] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, 2020.

[9] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 815–823. IEEE, June 2015.