Designing an Item Verification Application for the Supermarket Payment System using RetinaNet Algorithm

Ernest Julio Andreal1,a), Alexander Agung Santoso Gunawan2,b), Andry Chowanda3,c), and Heri Ngarianto4,d)

1Mathematics Department, School of Computer Science Bina Nusantara University, Jakarta, Indonesia

2,3,4 Computer Science Department, School of Computer Science Bina Nusantara University, Jakarta, Indonesia

b) Corresponding author: aagung@binus.edu

a) ernest.andreal@binus.ac.id

c) achowanda@binus.edu

d) heri.ngarianto@binus.ac.id

**Abstract.** Shopping in a supermarket is an activity done by people to fulfill their daily needs. In the supermarket, customers can go to the checkout counter to pay for their items with the help of the cashier. But before that payment is made, the cashier must scan all the items manually using a barcode scanner. This takes some time and can cause a queue to develop. To remedy this, some supermarkets have installed self-checkout counters. In response to this, our research proposes an item verification application designed to streamline the process. Using the RetinaNet object detection model, the application allows multiple items to be detected and verified simultaneously, reducing the time customers spend at self-checkout. The RetinaNet model has demonstrated strong performance, achieving mAP (mean Average Precision) of 92.6%. Although there may be challenges when items partially occlude each other, the model still performs well, ensuring accurate item verification.

**Keywords:** Item Verification, Payment System, object recognition, RetinaNet

# INTRODUCTION

Shopping is one of the activities people do to fulfill their daily needs. It brings advantages, both for the body and the soul. Shopping is done in a variety of places, such as malls, traditional markets, and supermarkets. When a customer shops in a supermarket, he/she usually takes items they need and keeps them wherever they go within the supermarket. Once the customer finishes taking items, he/she goes to the cashier and gives all the items to the clerk for him/her to verify. The clerk will scan the items individually, and when all the items are scanned, he/she will show the total price of the items to the customer [[1](#_ENREF_1)]. However, this manual scanning process can be time-consuming, leading to long queues at the checkout counter, which can frustrate both customers and staff [[2](#_ENREF_2)].

In the last few years, some supermarkets have installed self-checkout counters to replace traditional cashiers. On such counters, the customer can scan and pay for the items without the help of the clerk. This is an alternative for customers to verify items. Self-checkout counters have some advantages compared to traditional cashiers, such as convenient usage, and the fact that they help customers in shopping [[3](#_ENREF_3)]. Despite that, customers still have to scan the barcode on each item, then pack the items in the packing area. In other words, customer involvement remains high. Many customers use self-checkout systems with the expectation of reduced checkout times [[4](#_ENREF_4)].

Conducted research on customers’ attitude towards self-checkout systems in fast-moving consumer goods (FMCG) stores in Croatia. Respondents in their study pointed out some advantages of self-checkout systems, three of which are the lack of crowds in large retail stores, absence of queues on cashiers, and ease of managing items after scanning.

In this research, we propose an alternative approach to item verification at self-checkout counters using a camera-based system. The camera captures images of the items, and an object detection model identifies each item in the image. This model is integrated into a web application for item verification, and we will conduct experiments to test its performance in various scenarios.

The goal of this research is to make the item verification process faster and more convenient, allowing customers to spend less time at the checkout and more time on other activities. Additionally, it aims to enhance customer satisfaction, benefiting both shoppers and supermarket managers.

# Related Work

Gaol et al. [[5](#_ENREF_5)] developed a payment system that verifies items using a device equipped with a conveyor belt, a webcam, and an LCD screen. In this device, item detection is done with the help of a few algorithms, namely SURF, k-nearest neighbor, and a voting algorithm. Recognized items will be compared against the items database to determine whether those items exist in the database or not.

In a master’s thesis by Rigner [[6](#_ENREF_6)], an automatic self-checkout system based on computer vision was created using a Raspberry Pi computer and camera. Rigner compared the performance of three object detection models in detecting items using two performance indicators, namely mean average precison (mAP) with a similar format as the COCO benchmark, and also detection runtime speed. The models tested are Mask R-CNN, YOLOv3, and RetinaNet. After doing some experiments, Rigner found that the Mask R-CNN model obtained the highest mAP, while the YOLOv3 is the model with the highest detection speed. Nevertheless, he concluded that out of the three models tested, the best model is RetinaNet.

Shailaja et al. [7] proposed a checkout system incorporating an object detection feature based on the Faster R-CNN model to identify purchased items. The model is trained on a dataset consisting of 800 training images and 200 test images, divided into 10 item classes. After the model is trained, the model obtained an accurracy of 92%. The proposed system has three parts: Retail Local Server, Detection, and Response. On the Retail Local Server part, a web server takes photos of items and sends an image processing request to a machine learning model. Next, in the Detection part, the image processing request is received, and the system preprocesses the images, and at the same time, the model and labels are taken. Then, the images are sent to the model and processed by a Tensorflow session, and the system returns the name and size of the items as a response to the web server. Finally, the Response part receives the response from the model, and shows the name and quantity of the items, calculates prices, and returns the bill.

Pinto and Fernandes [8] developed an IoT-based checkout system using machine learning and RFID technology. In their system, the researchers used the YOLOv3-tiny object detection model to recognize purchased items, and they used RFID to help with the item recognition process. To support the whole process, the researchers used the Raspberry Pi 4 model B as the main computer. There are two devices plugged to the computer, namely the Pi Camera which is used to take photos of items; and the MFRC522 module which can read data stored in RFID tags. In the system, the customer placed items in a trolley so that the RFID tag attached to each item can be read, and at the same time, the camera can take pictures of the items and sends them to the YOLOv3-tiny model so that it recognizes the items. After the items are recognized, the recognition results from the two components are compared so that no items are missed, after which the bill is generated.

In a study by Gunawan et al. [9], the researchers designed a smart trolley payment system, comprising two main parts: item scanning and item verification at the store's exit. The entire system is supported by a special Android application for the Android smartphone, a computer attached to a trolley with its associated application, and a server and database as a backend of the system. To begin the item scanning part, the customer is asked to perform a check-in either to the Android application or by using a barcode scanner plugged to the trolley computer. The check-in is done by scanning the trolley number barcode stuck to the trolley so that the system knows which trolley the customer uses while shopping. After that, the customer can scan barcodes printed on item packaging, or the ones directly stuck on the items. The customer can view a list of items he/she has scanned on the smartphone screen, or the LCD screen plugged into the trolley computer. Once the item scanning process ends, the customer can start verifying items in the exit area. The items are verified with the help of a webcam that takes images of the items. Those images are then compared to item photo examples from the database inside the verification computer. The comparison is done using the structural similarity index (SSIM) which is the main component in the item verification process.

# METHODS

**FIGURE 1** shows the steps involved in this research.

A screenshot of a computer

Description automatically generated

**FIGURE 1**. Research Methodology

This study consists of four stages based on the prototyping model. The stages include:

* Planning
* Analysis
* Design
* Implementation

## PLANNING

In this stage, the initial step involves identifying the key problems outlined in the previous section: determining whether an object detection model can assist with the verification of multiple items at a self-checkout counter, assessing the model's performance in this task, and designing and developing an application capable of accurate item verification. Following this, a literature review is conducted to deepen the understanding of these challenges and to explore potential solutions.

## ANALYSIS

A dataset of images was created for use in the model training process. Each image contains items placed in various positions, orientations, and against different backgrounds. The steps involved in the creation and preprocessing of the dataset are as follows:

1. A smartphone was used to take photos with a camera resolution of 2880 × 2160 pixels to capture as much detail as possible. These photos were later downscaled to reduce their size (see step 5).
2. Items to be photographed were prepared.
3. Ten different surfaces were prepared and used as backgrounds for the photos.
4. For each surface, 20 photos were taken. Before each photo, 1 to 7 items of varying types (1 to 5 types) were placed on the surface, and a photo was taken before removing the items. This process was repeated until all surfaces had been used, resulting in 200 photos in total.
5. All photos were downscaled to a resolution of 640 × 480 pixels.
6. The downscaled photos were then randomly split into a training set and a validation set, with 160 photos in the training set and 40 in the validation set.

## DESIGN

mprocessed individually. A pretrained RetinaNet model [10], sourced from the Detectron2 GitHub repository [11], was selected for training. Built on the PyTorch framework, the model uses ResNet-50 version 1 as its backbone, with an input resolution of 640 × 640 pixels. The training process was conducted on an NVIDIA GeForce RTX 3070 GPU, utilizing the training set as input while tracking the model’s detection accuracy and loss throughout the session.

Before applying the RetinaNet model to real-world item detection, validation was performed using photos from the validation set. This automatic validation occurred after several training epochs and was repeated until training was complete, with all results documented. Once training concluded, the model was tested on real items to assess its detection accuracy. In cases where performance was insufficient, the dataset was refined, and the model retrained. Upon achieving satisfactory results, the design phase of the item verification application began.

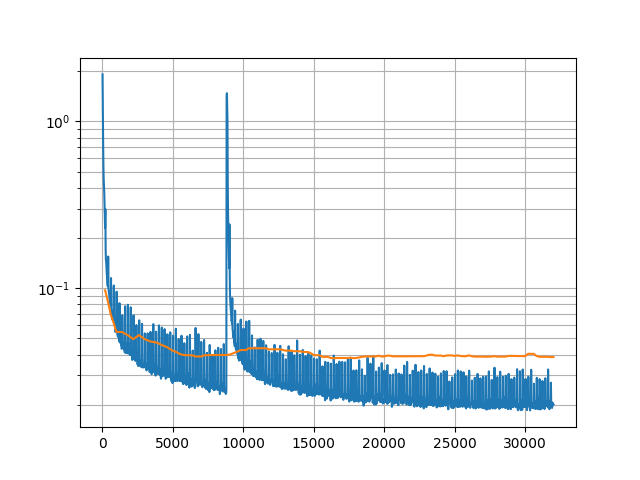
## IMPLEMENTATION

In this stage, the item verification application was designed using flowcharts to outline its structure. Mock-ups of the application's interface were also created to visualize how the final product would appear. After the design phase, the application was developed as a web-based platform. Once implemented, the application was tested to ensure its functionality. If the application did not perform as expected, it was redesigned and redeveloped according to the new design. However, if it worked as intended, a final round of testing was conducted to evaluate its suitability for practical use. During this phase, any errors or issues were identified. If the application was deemed unfit for use, the process reverted to the dataset creation step for further refinement.

# RESULTS AND DISCUSSION

## MODEL TRAINING AND VALIDATION

**FIGURE 2** shows the training and validation results of the RetinaNet model. The training lasts for 32000 iterations. The blue graph shows the training loss of the model, and the orange graph shows the validation loss. The vertical axis of the graph represents the loss value (in logarithmic scale), and the horizontal axis represents the iteration number.



**FIGURE 2**. Training and Validation Loss Graphs

The following tables show the validation results at the last validation of the model right after the last training iteration was completed. **TABLE 1** shows the overall validation result over all the validation photos, whereas Table II shows the validation metrics for each item type.

**TABLE 1.** Validation Result (All Photos)

|  |  |  |  |
| --- | --- | --- | --- |
| **COCO Metric** | **Metric Value (%)** | **COCO Metric** | **Metric Value (%)** |
| mAP | 92,607 | AR@1 | 63,909 |
| mAP@0,50 | 100 | AR@10 | 94,187 |
| mAP@0,75 | 99,333 | AR@100 | 94,187 |
| mAP (*small*) | N/A | AR (*small*) | N/A |
| mAP (*medium*) | 92,984 | AR (*medium*) | 93,125 |
| mAP (*large*) | 92,767 | AR (*large*) | 94,367 |

**TABLE 2**. Validation Result (Per Item Type)

|  |  |  |
| --- | --- | --- |
| **Item Type** | **AP (%)** | **AR (%)** |
| Instant Coffee | 93,155 | 94,5 |
| Instant Noodle | 96,473 | 97,674 |
| Instant Coconut Milk | 88,870 | 91 |
| Toothbrush | 93,217 | 94,594 |
| **Wet Tissue** | **91,322** | **93,170** |

**TABLE 2** shows that the RetinaNet model can help with the item verification process by classifying and labelling the items correctly.

### Detection Capability Test

A test was conducted to evaluate the model’s detection capability in various item placement scenarios. Twelve cases were created (labeled A to L), and each case was assessed based on the following aspects:

1. The number of items (few [1 to 9]; many [10 to 19])
2. The position of each item relative to other items (far apart; near each other; slightly occluding)
3. The number of item types (one; more than one)

Before we performed the test, we prepared some test photos for each case. Specifically, we prepared 10 photos for each of the cases where there is more than one item type (cases A to F), and 5 photos for each of the cases where there is exactly one item type (cases G to L). Shown in **FIGURE 3** and **TABLE 3** lists the explanation for each case.

A screenshot of a computer

Description automatically generated

**FIGURE 3**. 12 cases Detected

**TABLE 3**. Test Case Explanation

|  |  |
| --- | --- |
| **Case** | **Explanation** |
| A | Few items, placed far apart, more than one type |
| B | Few items, placed near each other, more than one type |
| C | Few items, placed slightly occluding, more than one type |
| D | Many items, placed far apart, more than one type |
| E | Many items, placed near each other, more than one type |
| F | Many items, placed slightly occluding, more than one type |
| G | Few items, placed far apart, exactly one type |
| H | Few items, placed near each other, exactly one type |
| I | Few items, placed slightly occluding, exactly one type |
| J | Many items, placed far apart, exactly one type |
| K | Many items, placed near each other, exactly one type |
| L | Many items, placed slightly occluding, exactly one type |

As for the test setup, we attached a camera at the end of a microphone stand, position the camera so that it is located at a distance of 80 cm above the surface of a table, and orient the camera so that it faces the table surface straight down. **FIGURE 4** shows the depiction of the setup.

A camera and a table with a video surveillance camera

Description automatically generated with medium confidence

**FIGURE 4**. Detection Capability Test Setup

For each case, the result for a test case is a “pass” if each of the items in the photo are all detected correctly (the predicted item type corresponds with the type of the item in real life and the bounding box surrounds the item correctly), and if there are not less nor more bounding boxes than the number of items appearing in the photo. Otherwise, the result for the case is a “failure”. The pass rate for a given case is calculated as the ratio of the number of passes divided by the total number of test case photos in that particular case. The results of this test are summarized in **TABLE 4**

**TABLE 4**. Detection Capability Test Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Case** | **Result** | | | **Pass Rate** |
| **Pass** | **Failure** | **Total** |
| A | 6 | 4 | 10 | 60% |
| B | 9 | 1 | 10 | 90% |
| C | 1 | 9 | 10 | 10% |
| D | 7 | 3 | 10 | 70% |
| E | 8 | 2 | 10 | 80% |
| F | 0 | 10 | 10 | 0% |
| G | 4 | 1 | 5 | 80% |
| H | 5 | 0 | 5 | 100% |
| I | 0 | 5 | 5 | 0% |
| J | 4 | 1 | 5 | 80% |
| K | 5 | 0 | 5 | 100% |
| L | 0 | 5 | 5 | 0% |

Based on the test result and after observing all the detection result photos, it is noticed that:

1. The RetinaNet model could not detect items correctly in cases C, F, I, and L, since in those cases, some of the items are slightly occluding each other.
2. The item types are correctly predicted in the cases where there is only one item type in the photo.
3. The model can easily detect items if they are placed far apart or near each other.
4. The item type could be predicted incorrectly if the item is oriented in certain ways.

Lighting can affect the detection result.

# CONCLUSIONS

Based on the results of the research we conducted, we conclude that the RetinaNet object detection model can indeed help with the verification of more than one item on a self-checkout counter (except when the items slightly occlude each other). The model performs well in detecting items, achieving a strong mAP (mean Average Precision) of 92.6%, and the application successfully verifies items as intended. However, there is potential for further research to improve the model's ability to handle occlusions and enhance its accuracy in more complex real-world scenarios. Future work could also explore optimizing the system for faster processing and integrating additional features, such as weight verification or product categorization, to further streamline the self-checkout experience.

# Author’s Contribution

The study was conceived and designed by Ernest J Andreal and Alexander AS Gunawan. The experiments are performed by Ernest J Andreal. The writing is guided by Alexander AS Gunawan, Andry Chowanda and Heri Ngarianto. All authors read and approved the manuscript.

# Availability Data and Materials

This study utilized a public image dataset from the Detectron2 library, developed by Facebook AI Research. Detectron2 is a cutting-edge platform that offers state-of-the-art detection and segmentation algorithms. The dataset and algorithms are publicly available on GitHub at https://github.com/facebookresearch/detectron2.

# References

1. K. Lalitha, M. Ismail, S. K. Gurumurthy, and A. Tejaswi, "Design of an intelligent shopping basket using IoT," *International Journal of Pure and Applied Mathematics,* vol. 1, no. 151, 2017.
2. P. Bhandekar, C. Tomar, D. Kasewar, and A. Sheikh, "A Survey on Smart Trolley System Based on Android Application," *Engineering and Technology,* vol. 4, no. 4, pp. 55-56, 2018.
3. E. Korchagina, "Consumers Attitudes to Innovations in Retail: A Case of Introducing the Self-Scanning Technology in Russia," *Eurasian Journal of Business and Management,* vol. 7, no. 3, pp. 34-43, 2019.
4. H. K. Leng and K. N. L. Wee, "An examination of users and non-users of self-checkout counters," *The International Review of Retail, Distribution and Consumer Research,* vol. 27, no. 1, pp. 94-108, 2017.
5. J. L. Gaol, M. A. Haq, S. Armiyanto, H. Kusuma, and T. Tasripan, "Automatic self-checkout system using surf, brute force matcher, and rfid for payment process optimization at supermarket," *IPTEK Journal of Proceedings Series,* no. 1, pp. 140-144, 2019.
6. A. Rigner, "Ai-based machine vision for retail self-checkout system," *Master's Theses in Mathematical Sciences,* 2019.
7. K. Shailaja, J. Vasundhara, N. B. Meghana, and E. Lalith, “Smart Retail Checkout,” *Purakala*, vol. 31, no. 31, pp. 317-322, 2020.
8. R. Pinto and A. Fernandes, “IoT based Quick Retail Checkout using Machine Learning and RFID Technology,” *Journal of Emerging Technologies and Innovative Research*, vol. 8, no. 8, pp. 919-923, Aug. 2021.
9. A. A. S. Gunawan, E. J. Andreal, W. Budiharto, H. Ngarianto, M. Attamimi, and H. Tolle, "Item Verification on the Smart Trolley System using Object Recognition based on the Structural Similarity Index," in *6th International Conference on Computer Science and Computational Intelligence (ICCSCI)*, Jakarta, 2021.
10. K. He, G. Gkioxari, P. Dollár, and R. Girshick, "RetinaNet: A Dense Object Detector," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 11, pp. 2980-2998, Nov. 2020.
11. Y. Wu, A. Kirillov, F. Massa, W. Y. Lo, and R. Girshick, "Detectron2," Facebook AI Research, 2019. [Online]. Available: https://github.com/facebookresearch/detectron2