# Improving Deep Learning-based Automatic Checkout System Using Image Enhancement Techniques

Long Hoang Pham, *Member, IEEE*, Duong Nguyen-Ngoc Tran, *Graduate Student Member, IEEE*,
Huy-Hung Nguyen, Hyung-Joon Jeon, Tai Huu-Phuong Tran, Hyung-Min Jeon,
and Jae Wook Jeon, *Senior Member, IEEE*,

Department of Electrical and Computer Engineering
Sungkyunkwan University, South Korea
{phlong, duongtran, huyhung91, joonjeon, taithp, hmjeon, jwjeon}@skku.edu *

## Abstract

*The retail sector has experienced significant growth in artificial intelligence and computer vision applications, particularly with the emergence of automatic checkout (ACO) systems in stores and supermarkets. ACO systems encounter challenges such as object occlusion, motion blur, and similarity between scanned items while acquiring accurate training images for realistic checkout scenarios is difficult due to constant product updates. This paper improves existing deep learning-based ACO solutions by incorporating several image enhancement techniques in the data pre-processing step. The proposed ACO system employs a detect-and-track strategy, which involves: (1) detecting objects in areas of interest; (2) tracking objects in consecutive frames; and (3) counting objects using a track management pipeline. Several data generation techniques—including copy-and-paste, random placement, and augmentation—are employed to create diverse training data. Additionally, the proposed solution is designed as an open-ended framework that can be easily expanded to accommodate multiple tasks. The system has been evaluated on the AI City Challenge 2023 Track 4 dataset, showcasing outstanding performance by achieving a top-1 ranking on test-set A with an F1 score of 0.9792.*

## 1. Introduction

The retail industry has recently experienced a shift with the incorporation of artificial intelligence and computer vision technologies, with automatic checkout (ACO) systems emerging as a significant innovation. ACO systems enable customers to scan, bag, and pay for their purchases with minimal or no assistance, resulting in cost savings and greater flexibility for retailers. Developing a vision- and deep learning-based ACO system requires addressing challenges such as object occlusion, motion blur, item similarities, and the implications of miss-detection and misclassification. Moreover, the extensive scope and complex nature of product categories, along with difficulty in acquiring realistic training images due to continuous product updates, add to the complexity of the development process. For real-world ACO systems, accuracy, stability, and efficiency are crucial.

In order to promote further progress in ACO development, AI City Challenge 2023 Track 4: Multi-Class Product Counting & Recognition for Automated Retail Checkout (AIC23) [17] has been proposed. This track focuses on automatically detecting and identifying products within a camera view to assist in retail store checkouts. When products are visible, the objective is to report information, including the product name, id, timestamp, or frame index. All products are expected to be presented within a defined region of interest (ROI) in the camera view, a condition that is typically achievable in real-world situations.

This paper presents DeepACOv2, a deep learning-based ACO system built upon [20]. DeepACOv2 employs the same detect-and-track approach, consisting of three primary steps: (1) detecting candidate objects as bounding boxes; (2) tracking objects across consecutive frames; and (3) managing tracks for various post-processing tasks such as counting. The data generation process incorporates several techniques to diversify the training data, including copy-and-paste, random placement, and augmentation. Two main issues should have been addressed in [20]: motion blur resulting from rapid scanning movements and operators'

*This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2021-0-01364, An intelligent system for 24/7 real-time traffic surveillance on edge devices)

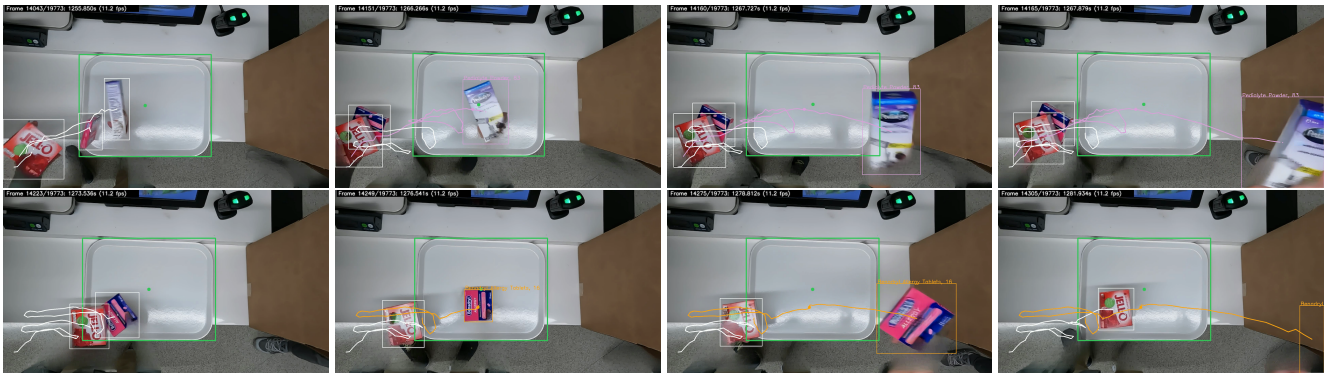


Figure 1. Examples of scanning and counting products in consecutive frames.

hands obstructing the products. To address the motion blur problem during the inference stage, NAFNet [6]—a deep learning-based image enhancement network is utilized. Additionally, the LaMa [29] model is employed to inpaint the blob where operators' hands block the products. Moreover, a new IoU-based tracing function is implemented to ensure that only objects fully within a white tray, i.e., the region of interest (ROI), are counted. Experiments on AIC23 have shown competitive results compared to other state-of-the-art (SOTA) methods, achieving a top-1 ranking with an F1 score of 0.9792.

In short, the main contributions of this paper are:

- Improve the existing DeepACO framework with a combination of image enhancement techniques in the data pre-processing step.
- Devise a new product track management scheme that guarantees to count only objects fully inside the ROI.
- Implement a versatile framework that can be easily integrated with different techniques to favor accuracy or processing speed.

The remainder of this paper is structured as follows: Section 2 presents related work. Section 3 details the proposed solution. Section 4 reports on experimental results. Lastly, Section 5 summarizes the paper's findings.

## 2. Related Work

In this section, a brief review of DeepACOv2 systems, object detection, and multiple objects tracking is presented.

### 2.1. Automatic Retail Checkout

Researchers have made significant progress in developing Automated Checkout (ACO) systems to address real-world problems. In the past, checkouts relied on scanning RFID or QR code tags on retail items as described in [5]. Recently, vision- and deep learning-based ACO systems have emerged, with the help of checkout datasets such as [18, 19, 32]. These datasets aim at generating large training sets using synthetic data and data augmentation techniques. AIC22 [18] Track 4 is the latest dataset that includes both synthetic and real-world data aimed at automatic checkout process in retail store. Based on AIC22, various studies [2, 20, 27, 28, 30] have reported encouraging results by combining different vision techniques. However, further efforts are needed to enhance ACO systems performance.

### 2.2. Object Detection

Object detection is a critical task in computer vision that involves identifying and locating objects within images or videos. Deep learning-based methods have become state-of-the-art object detection in recent years, delivering exceptional performance and accuracy. There are two primary approaches: two-stage detectors and single-stage detectors. Two-stage detectors, such as R-CNN [9], Fast R-CNN [8], and Faster R-CNN [24], first generate region proposals using a separate model or algorithm and then classify the objects within those regions. This approach tends to be more accurate but slower than single-stage detectors.

Single-stage detectors, such as YOLO [21] and SSD [14], forgo the region proposal phase, predicting object classes and bounding box coordinates in a single pass. While these models are generally faster, their accuracy may be compromised, particularly for small or overlapping objects. The YOLO model's later iterations, specifically YOLOv2-7 [4, 10, 22, 23, 31], have seen continuous improvements in accuracy, making them a favored choice for applications requiring low-latency, like robotics, autonomous vehicles, and video surveillance. Notably, YOLOv8 [11], the most recent and powerful variant, has exhibited outstanding performance in the COCO benchmark [13]. YOLOv8 has surpassed YOLOv5 in terms of accuracy, with the YOLOv8s model achieving an average precision of 51.4% on the COCO dataset and the YOLOv8m model reaching an average precision of 54.2% on the same dataset. YOLOv8 has also demonstrated superior performance in detecting small objects and addressed some limi-

tations of YOLOv5. Additionally, it features a shared backbone that can be utilized in various tasks, including object detection, semantic segmentation, and keypoint detection. Furthermore, YOLOv8 is even faster than YOLOv5. In this paper, YOLOv8 [11] object detection is employed to detect products and trays. In addition, YOLOv8 instance segmentation model is used generate human masks for the human inpainting task.

### 2.3. Multiple Objects Tracking

Recently, SORT [3], DeepSORT [33], and ByteTrack [34] are some of the most popular and widely used multiple object tracking methods. SORT [3] follows the tracking-by-detection approach, connecting detections from previous and current frames using data association and state estimation methods based on the Kalman filter. It also accommodates object re-entry within a set time frame and handles partial occlusion. DeepSORT [33] builds on SORT [3] by incorporating a deep association metric rooted in image features. In ByteTrack [34], all detections are associated despite their low confidence scores, further improving the performance of tracking multiple objects in complex environments.

## 3. Methodology

The DeepACOv2 system consists of a central detect-track-count pipeline and a collection of pre-processing methods, referred to as a bag-of-tricks. The following subsections provide a more in-depth discussion of each of these techniques in sequential order.

### 3.1. Image Enhancement

**Enhance noisy images.** AIC23 [17] track 4 provides a dataset for multi-class product counting and recognition in automated retail checkout scenarios. This dataset contains 116,500 synthetic images of 116 distinct products captured using a 3D scanner. Due to the scanning technique, the images exhibit fine-grained noise on the objects' surfaces. Training object detection models with these images, as done in [2, 20, 27, 28, 30], may result in reduced accuracy.

NAFNet [6] is a state-of-the-art general model designed for image restoration. A key feature of NAFNet is its non-linear activation-free nature, making it computationally efficient. NAFNet is capable of handling both denoising and deblurring tasks. This study employs pre-trained NAFNet models to enhance the visualization of training images and testing videos. Fig. 2 displays the improved training images. Directly applying denoising during the experiments results in blurred details, as illustrated in Fig. 2b. Conversely, applying deblurring produces a smoother image while preserving texture sharpness, as shown in Fig. 2c. Additionally, combining denoising and deblurring does not yield better outcomes. Therefore, this paper uses a NAFNet pre-trained on the REDS dataset [16] for the enhancement process. The enhanced images are then incorporated into synthetic data generation, as described in Section 3.3.

**Remove motion blur.** AIC23 [17] track 4 testing data consists of several video recording operators scanning products where motion blur is a common issue. As shown in Fig. 3, the camera's frame rate cannot keep up with the operator's quick scanning motion, causing product details to become blurred. As a result, object detection models generate numerous false positives. Therefore, deblurring images can improve theirs visual quality, making it easier to see fine details and reducing the blurriness caused by motion or other factors. As a result, it can help to improve the accuracy of object detection. The NAFNet [6] deblurring model pre-trained on the REDS dataset [16] is also utilized to the testing videos during the inference phase. Fig. 3b depicts how NAFNet can restore blurry texture details present on the product.

### 3.2. Human Inpainting

During product scanning, the operator's hands or body may partially obstruct the item, hindering the detection and tracking process. In [20], the authors attempted to detect hand keypoints using MediaPipe [15] and employed these keypoints to identify products being handled. However, this method is susceptible to considerable error. Conversely, [2] recommended eliminating human elements using the LaMa [29] inpainting technique, which has proven to be more suitable. Thus, this paper adopts the same strategy.

The LaMa [29] method requires two inputs: an RGB image and a binary mask highlighting the area to be removed. Hence, the human mask must be generated beforehand. A YOLOv8 [11] instance segmentation model, pre-trained on the COCO [13] dataset, is utilized for this purpose. The human instance segmentation masks are visualized in Fig. 3c. The inpainting process occurs frame-by-frame. An example of the LaMa technique can be observed in Fig. 3d.

### 3.3. Data Generation

**AIC23 Track 4 Dataset**. The AIC23 [17] Track 4 offers a dataset for multi-class product counting and recognition in the context of automated retail checkout. This dataset consists of 116,500 synthetic images $\mathbf{P}$, along with masks $\mathbf{M}$ intended for training. The images were generated using 116 unique product models obtained via a 3D scanner. To increase the dataset's diversity, random background images were selected from the COCO [13] dataset. Figure 4 presents some sample images and their respective masks for several classes. Furthermore, a video test set is also recorded. The test set is split between Test sets A and B with a ratio 40% to 60% accordingly. Test set A has been released for testing and result evaluation via the AIC23 evaluation server. Test set B is reserved for additional evaluation

(a) Input. (b) Denoising. (c) Deblurring. (d) Denoising and deblurring.

Figure 2. Illustration of denoising and deblurring on training images using NAFNet. Images are best viewed in color.

(a) Input image. (b) Deblurred image. (c) Human mask. (d) Inpainting result.

Figure 3. Illustration of motion deblurring and human inpainting on testing videos. Images are best viewed in color.

and determining the final ranking.

**Synthetic Data Generation**. Due to the strict limitations on external data usage imposed by AIC23, a custom synthetic dataset is created to train the detector. A copy-and-paste data generation pipeline is outlined in Algorithm 1 and depicted in Fig. 4. Initially, multiple clean background images $\mathbf{BG}$ are extracted from test videos. Next, $n$ product images $\mathbf{P}_n$ and masks $\mathbf{M}_n$ are chosen from $\mathbf{P}$ and $\mathbf{M}$. It is important to note that the product images $\mathbf{P}_n$ were previously refined as discussed in Section 3.1. A bitwise and operation is performed between $\mathbf{P}_n$ and $\mathbf{M}_n$ to obtain background-free product images $\mathbf{F}_n$. Following this, random rotation and scaling augmentations are applied to $\mathbf{F}_n$. Subsequently, $\mathbf{F}_n$ is randomly inserted into $\mathbf{BG}$ and bounding boxes $\mathbf{B}$ are documented. Lastly, $\mathbf{F}_n$ is merged with $\mathbf{BG}$ to generate the training image $\mathbf{I}$. A total of 54,475 images are generated, with an 80-20 split for training and validation purposes.

### 3.4. Tray Detection

During the AIC23 [17] test, the camera was placed directly above the checkout counter, facing downward. A shopping tray, positioned below the camera, served as the region of interest (ROI). However, the tray's location varied throughout the test videos, and the camera was moved at some distance. As a result, it was necessary to re-localize the white tray in every frame. In [26], the white tray are labeled in the synthetic training images to train the U-Net [25] model. In contrast, [2] suggested using a flood fill algorithm combined with the Scharr operator to identify the tray's pixels. These methods, however, assume that the tray is situated near the image's center, which can lead to errors when the camera is moved, as show in Fig. 5b. In this paper, a YOLOv8l [11] detection model is trained separately to detect the white tray automatically. The most significant challenge lies in generating training data. In the synthetic training image $\mathbf{I}$ detailed in Section 3.3, the flood fill method from [2] is employed to create pseudo-label bounding box labels. To simulate a moving camera, augmentation techniques such as random cropping, random shifting, and random rotation are applied during the training of YOLOv8l [11]. The model is run every frame, and the output bounding box coordinates are used as the ROI. The tray detection results can be found in Figs. 5c and 1.

### 3.5. Product Detection and Tracking

**Product Detection.** The detection module is a critical component in any surveillance system. YOLOv8x6 [11] is

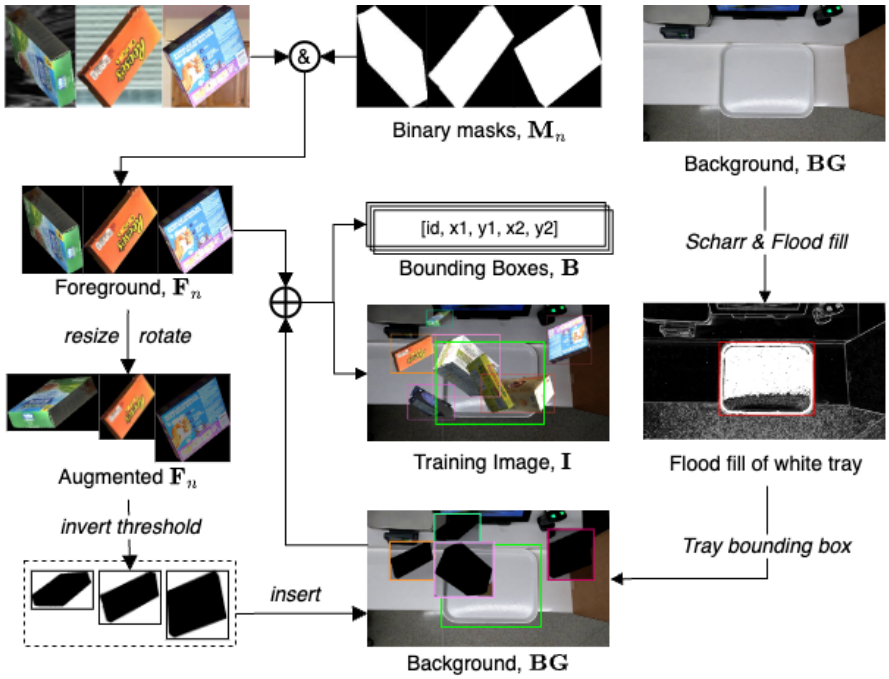



Figure 4. Illustration of the data generation process.

**Algorithm 1:** Data generation algorithm.

**Data:**
- $\mathbf{BG}$: background images
- $\mathbf{P}$: product images
- $\mathbf{M}$: binary masks

**Result:**
- $\mathbf{I}$: training images
- $\mathbf{B}$: bounding boxes

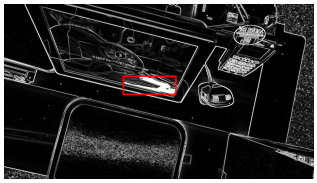
$n \leftarrow \text{random}(1, 7)$;
$angle \leftarrow \text{random}(0, 360)$;
$scale \leftarrow \text{random}(0.8, 1.2)$;
$\mathbf{I}_n, \mathbf{M}_n \leftarrow$ randomly select $n$ items from $(\mathbf{I}, \mathbf{M})$;
$\mathbf{F}_n \leftarrow \mathbf{I}_n$ & $\mathbf{M}_n$;
$\mathbf{F}_n \leftarrow \text{rotate}(\mathbf{F}_n, angle) + \text{resize}(\mathbf{F}_n, scale)$;
$\mathbf{IF}_n \leftarrow \text{invert\_threshold}(\mathbf{F}_n)$;
$\mathbf{B} \leftarrow$ randomly insert $\mathbf{IF}_n$ into $\mathbf{BG}$;
$\mathbf{I} \leftarrow \mathbf{B} + \mathbf{F}_n$;

employed for detection, as it achieves high accuracy on public datasets. The model is trained on the provided dataset of generated objects, as outlined in Section 3.3, with 116 output classes. The same training configuration as in [11] is applied. Furthermore, both basic (rotation, translation, cropping, etc.) and advanced augmentations (RandAugmentation [7] and mosaic [4]) are utilized. Adam [12] optimizer is used to train the model. The model achieves a 98.09% accuracy on the validation set during training. For each frame, the detector outputs a list of object positions, confidence scores, and class ids. Each detection result is decoded as $[idx, x1, y1, x2, y2, cls, conf]$, where $idx$ is the frame index which is subsequently used in tracks post-processing.

**Product Tracking.** Tracking is an the key process in video processing. The tracking module assigns an id to a detected object, associates it in consecutive frame to ensure that the item is only identified once. Similar to [20], SORT [3] algorithm is used with some adjustments in the track management scheme. The tracking steps are described as follows:

- **New detection.** New items are identified by the detector, and each item is subsequently fed into the SORT tracker. Existing tracks are updated with corresponding items, and a new track is created for any unmatched items. If a track has not been updated for a duration of $max\_age$, it is removed.
- **Candidate.** The detected item is marked as a candidate when it's bounding box center is inside the ROI,
- **Confirmed.** A candidate item must be tracked successfully for a minimum of $min\_hit\_streak$ consecutive frames. To ensure that an item is counted only when it is entirely within the ROI, the candidate item must be at least $min\_entering\_distance$ away from the closest corner of the ROI. Once all conditions are met, the candidate item is designated as confirmed. The confirmed state represents the primary stage in the lifetime of an item.

(a) Inpainted image.



(b) Flood fill method [2].


(c) YOLOv8l [11].

Figure 5. Illustration of tray detection.

- **Counting.** For a confirmed item to be eligible for counting, it must meet several criteria. First, there may be instances where the operator moves an item in and out of the ROI without actually scanning it, which can be referred to as a "ghost"-scanning action. To avoid counting these items incorrectly, the item must remain confirmed for at least $min\_confirms$ frames. Second, some products might be too large to fit entirely within the ROI, as illustrated in Fig. 6. In such cases, the IoU value between the item's bounding box and the ROI's bounding box must be calculated and exceed the $min\_counting\_iou$ threshold. Third, since the scanning action occurs around the center of the white tray in all testing videos, an item must be at least $min\_counting\_distance$ pixels away from the ROI's center to be marked as a counting item.
- **Counted.** At the end of each processing loop, the counter searches for all counting items and logs them into the result file. Subsequently, these items are labeled as counted. All counted items remain in the tracking list until they haven't been updated for $max\_age$, at which point SORT deletes them.

### 3.6. Product Counting

The last stage in the processing loop involves counting products. Within each track, all detected instances of an item are stored. A majority voting scheme is employed to establish the item's final class, which is then reported as the definitive class for the entire track. The item is subsequently recorded in the result file. Additionally, as part of the solution, the frame index is also provided.

## 4. Experiments

### 4.1. Experimental Settings

**Implementation Details**. The DeepACOv2 system is implemented using PyTorch on an Intel Core i7-7700, an NVIDIA RTX 3090 24GB, and 32GB RAM. The training of deep learning model are described follows:

- Product detection: YOLOv8 detectors are trained using the synthetic dataset (Section **??**). Several versions (l, x, x6) of YOLOv8 are trained, and YOLOv8x6 proves to be the most effective when trained for 50 epochs with an input image size of $1920 \times 1920$. All other training parameters follow the original network settings [11]. The validation results of each model variants are summarized in Table 2.
- Tray detection: a YOLOv8x detector is trained using an input image size of $640 \times 640$ for 20 epochs. The tray's bounding boxes are derived from the test video as depicted in Section 3.5.
- Inpainting: a YOLOv8x instance segmentation model pretrained on the COCO [13] dataset is employed to extract binary masks, which are then used in the LaMa [29] inpainting process. The LaMa model is pretrained on the Places2 dataset [35].
- Image enhancement: A NAFNet [6] model pre-trained on REDS dataset [16] is used for the deblurring task.

**Pre-processing Procedure**. The procedure starts with the extraction of separate frames from videos utilizing the FFmpeg library [1] to guarantee consistent frame IDs. Following this, all images are deblurred to improve the objects' visual appearance. Next, YOLOv8 [11] instance segmentation infers the enhanced images to extract human instance masks. Upon obtaining the instance masks, LaMa [29] is applied to remove human parts. Finally, the resulting outputs are used in the main processing step.

**Main Processing Procedure**. The main processing procedure is the detect-track-count paradigm. YOLOv8 detectors are employed to identify the tray, which also serves as the ROI, as well as to detect candidate objects. The resulting outputs are then fed into the SORT [3] tracker. The track management module governs over tracks creation, update, deletion, and determines when products are ready for counting. The hyperparameters are configured as specified in Table 1.

### 4.2. Evaluation

The evaluation is performed on 4 videos of test set A using F1 score:

$$F_1 = \frac{TP}{TP + 0.5 \times (FP + FN)}, \tag{1}$$

where a true-positive (TP) identification occurs when an object is accurately counted within the ROI, specifically when

Table 1. Hyperparameters for the track management process.

| **Hyperparameter** | **Value** |
|---|---|
| max_detections | 3 |
| min_entering_distance | 100 |
| min_hit_streak | 3 |
| max_age | 3 |
| min_confirms | 3 |
| min_counting_distance | 50 |
| min_counting_iou | 0.9 |

Table 2. Validation results of different YOLOv8 variants.

| Variant | Precision | Recall | mAP50 |
|---|---|---|---|
| YOLOv8l | 0.9954 | 0.9867 | 0.9867 |
| YOLOv8x | 0.9950 | 0.9863 | 0.9900 |
| YOLOv8x6 | 0.9953 | 0.9950 | 0.9910 |

the object is fully inside the white tray. A false-positive (FP) refers to an identified object that does not qualify as a TP identification. Finally, a false-negative (FN) identification takes place when a ground-truth object is not correctly identified.

During the experiments, various hyperparameter values were tested. For example, not using the $min\_entering\_distance$ resulted in items being counted before they had fully entered the ROI, leading to an F1 score of only $0.7150$. Adding $min\_counting\_iou$ and $min\_counting\_distance$ provided fine-grained control over the tracking process, increasing the F1 score to $0.9536$. Moreover, as no more than three items are scanned at once, the maximum number of detections per frame is set to 3, which helps reduce false positive detections. Additionally, increasing the image resolution to $1024 \times 1024$ improves the result to $0.9688$. The final hyperparameter values are presented in 1. In some instances, the scanning action is too fast, creating duplicate tracks and leading to the same items being counted multiple times. To address this issue, only the first of any consecutive duplicate counts of the same items is retained in the output file. The final ranking results of the challenge are displayed in Table 3, with our approach achieving first place for Test Set A and an F1 score of $0.9792$.

## 5. Conclusion

This paper introduces DeepACOv2, a deep learning-based automatic checkout system that incorporate several improvements over predecessor studies. DeepACOv2 system has a detect-track-count pipeline, which includes: (1) identifying objects in regions of interest; (2) tracking objects across consecutive frames; and (3) counting objects through a track management pipeline. Additionally, the system employs image enhancement techniques to address issues of motion blur and noisy images. Various data generation techniques—including copy-and-paste, random placement, and augmentation—are utilized to create diverse training data. Moreover, the proposed solution is designed as an open-ended framework, facilitating easy expansion to support multiple tasks. The system has been evaluated on the AI City Challenge 2023 Track 4 dataset, showcasing exceptional performance by achieving a top-1 ranking on test-set A with an F1 score of 0.9792.

Table 3. Public leaderboard of AIC23 Track 4 on test set A.

| Rank | Team ID | Team Name | F1 Score |
|---|---|---|---|
| **1** | **33** | **SKKU Automation Lab** | **0.9792** |
| 2 | 21 | BUPT_MCPRL | 0.9787 |
| 3 | 13 | Zebras | 0.8254 |
| 4 | 1 | SCU Anastasiu Lab | 0.8177 |
| 5 | 23 | Fujitsu R&D Center | 0.7684 |
| 6 | 200 | Centific | 0.6571 |
| 7 | 65 | dtb2023 | 0.4757 |
| 8 | 64 | Fu | 0.4215 |
| 9 | 9 | HCMIU-CVIP | 0.3837 |
| 10 | 68 | UTE_AI | 0.3441 |

## References


[1] Ffmpeg (http://www.ffmpeg.org). 6

[2] Vojtěch Bartl, Jakub Špaňhel, and Adam Herout. Persongone: Image inpainting for automated checkout solution. In *CVPRW*, pages 3114–3122, 2022. 2, 3, 4, 6

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *ICIP*, pages 3464–3468, 2016. 3, 5, 6

[4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv*, 2020. 2, 5

[5] M. F. M. Busu, I. Ismail, M. F. Saaid, and S. M. Norzeli. Auto-checkout system for retails using radio frequency identification (rfid) technology. In *ICSGRC*, pages 193–196, 2011. 2

[6] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *ECCV*, page 17–33, 2022. 2, 3, 6

[7] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*, volume 33, pages 18613–18624, 2020. 5

[8] R. Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015. 2

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014. 2

[10] G. Jocher. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference, 2022. 2

(a) Item is not fully inside the ROI. (b) Complex scanning action. (c) Operator moves item in and out of the ROI.

Figure 6. Examples of difficult scanning actions. Images are best viewed in color and zoom in.


[11] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Yolo by ultralytics, 1 2023. 2, 3, 4, 5, 6

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, page 13, 2015. 5

[13] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, volume 8693, pages 740–755, 2014. 2, 3, 6

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, volume 9905, pages 21–37, 2016. 2

[15] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann. Mediapipe: A framework for building perception pipelines. *arXiv:1906.08172*, 2019. 3

[16] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *CVPRW*, 2019. 3, 6

[17] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M. Chang, Y. Yao, L. Zheng, M. Shaiqur Rahman, A. Venkatachalapathy, A. Sharma, Q. Feng, V. Ablavsky, S. Sclaroff, P. Chakraborty, A. Li, S. Li, and R. Chellappa. The 7th ai city challenge. In *CVPRW*, 2023. 1, 3, 4

[18] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M.-C. Chang, Y. Yao, L. Zheng, M. S. Rahman, A. S., Q. Feng, V. Ablavsky, S. Sclaroff, and R. Chellappa. The 6th ai city challenge. In *CVPRW*, 2022. 2

[19] Jingtian Peng, Chang Xiao, Xun Wei, and Yifan Li. Rp2k: A large-scale retail product dataset for fine-grained image classification. *arXiv*, 2020. 2

[20] Long Hoang Pham, Duong Nguyen-Ngoc Tran, Huy-Hung Nguyen, Tai Huu-Phuong Tran, Hyung-Joon Jeon, Hyung-Min Jeon, and Jae Wook Jeon. Deepaco: A robust deep learning-based automatic checkout system. In *CVPRW*, pages 3106–3113, 2022. 1, 2, 3, 5

[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016. 2

[22] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, pages 6517–6525, 2017. 2

[23] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018. 2

[24] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, page 91–99, 2015. 2

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015. 4

[26] Md. Istiak Shihab, Nazia Tasnim, Hasib Zunair, Labiba Rupty, and Nabeel Mohammed. Vista: Vision transformer enhanced by u-net and image colorfulness frame filtration for automatic retail checkout. In *CVPRW*, pages 3182–3190, 2022. 4

[27] Md. Istiak Hossain Shihab, Nazia Tasnim, Hasib Zunair, Labiba Kanij Rupty, and Nabeel Mohammed. Vista: Vision transformer enhanced by u-net and image colorfulness frame filtration for automatic retail checkout, 2022. 2, 3

[28] Maged Shoman, Armstrong Aboah, Alex Morehead, Ye Duan, Abdulateef Daud, and Yaw Adu-Gyamfi. A region-based deep learning approach to automated retail checkout, 2022. 2, 3

[29] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *WACV*, pages 3172–3182, 2022. 2, 3, 6

[30] Junfeng Wan, Shuhao Qian, Zihan Tian, and Yanyun Zhao. An effective framework of multi-class product counting and recognition for automated retail checkout. In *CVPRW*, pages 3282–3290, 2022. 2, 3

[31] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Scaled-yolov4: Scaling cross stage partial network. In *CVPR*, pages 13029–13038, 2021. 2

[32] Xiu-Shen Wei, Quan Cui, Lei Yang, Peng Wang, Lingqiao Liu, and Jian Yang. Rpc: A large-scale retail product checkout dataset. *SCIS*, 65:1869–1919, 2022. 2

[33] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, pages 3645–3649, 2017. 3

[34] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *ECCV*, 2022. 3

[35] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. 40(6):1452–1464, 2018. 6