An Effective Framework of Multi-Class Product Counting and Recognition for Automated Retail Checkout

Junfeng Wan1*, Shuhao Qian1*, Zihan Tian1*, Yanyun Zhao1,2†
1Beijing University of Posts and Telecommunications
2Beijing Key Laboratory of Network System and Network Culture, China
{wanjunfeng,qiansh,Tianzh,zyy}@bupt.edu.cn

Abstract

As the field of computer vision grows, Automated Retail Checkout has become a highly anticipated development goal. The key of this task is to improve the accuracy rate. If there is an error, it will bring serious losses to the business and awful experience for customers which is not our expected. This competition gives us an opportunity to simulate check-out in a real world scenario, so that we can identify problems and solve them, not only for the competition, but also for the practical application. As one of the participating teams in this task, we pursue the goal of avoiding misdetection and misclassification, and build a complete set of framework to achieve high-precise, high recall performance. In addition, there is an excessive difference between the training data and test data. How to use limited data to make up for the differences in this part is also one of the highlights of our framework. In general, our framework consists of three main parts. Firstly, the Pre-Processing module to make up for the differences between training and test data. The DTC module completes the overall process of automatic recognition. Finally the MTCR module is proposed to post-process the output of the DTC module. On the TestA data of AICITY2022 Task 4, we have achieved significant result compared to the other teams. Finally, our model is ranked 1st in AICITY2022 Task 4 [17]. The code is available at: https://github.com/w-sugar/DTC-AICITY2022.

1. Introduction

The challenges of AI CITY have promoted the research of vision problem in recent years, especially in intelligent transportation, such as traffic statistics, vehicle re-recognition, cross-camera tracking, abnormal event analysis and other application scenarios related to intelligent transportation. It is known as the "ImageNet Competition for Intelligent transportation video Analysis".

A growing application of AI and computer vision is in the retail industry. Of the various problems that can be addressed, track 4 of 2022 AI City Challenge, Multi-Class Product Counting and Recognition for Automated Retail Checkout, focuses on accurate and automatic check-out in a retail store. Participating teams will count and identify products as they move along a Retail checkout conveyor belt. The challenge stems from the real-world scenario of occlusion, movement, similarity in products being scanned, novel SKUs that are created seasonally, and the cost of misdetection and misclassification [17].

In track 4 task, we need to detect the objects, tracking for counting and classify them correctly in test videos. Image classification is one of basic vision tasks, and its goal is to classify images into corresponding categories. Automated retail checkout is more than an image classification task. Before classification, it is necessary to locate when and where products appear in test videos, so we firstly need to train object detection model with training dataset of track 4 to detect products, and then classify the detecting objects. As shown in Figure 1, the training data is shown on the left and the testing data is shown on the right. We can see that the training data and the test data [18, 28] are very different in appearance. The images of training set are low resolution and have aliasing textures, and also some areas of the object are very dark. In contrast, there are no such quality problems in testing data. The distribution difference of ap-
pearance between training and test data greatly leads to the phenomenon for excellent performance on the training set but poor performance on the test set. For example, it works well on our homemade detection training set, but there are a lot of false detections and missed detections on the test videos. Therefore, how to make up for the difference between training and test data is also an urgent problem to be solved in this task. So we exploit an image enhancement algorithm [11] to preprocess the images of training set, which decreases the difference between the two data sets, and get satisfactory performance of detection model and classification model.

In addition, each image of training set is only including one object of products, and they can be used directly for object classification network training, but not for detection network training. While the test data is untrimmed videos and each frame contains various objects in the scene, so it is necessary to locate when and where the product appears in advance. Then we construct an object detection dataset to train an effective detector with the preprocessing training data.

Tracking of objects is necessary as each product needs to be counted only once while it passes through the region of checkout. Therefore, we track the product detections in videos based on DeepSORT [27], and one tracking trajectory obtained represents a product.

In light of these challenges and the characteristic of automated retail checkout, we propose a precise and efficient framework and get an excellent evaluation result. Our main contributions are as follows:

1) Based on Track 4 automated retail checkout in AI CITY 2022 Challenge, we propose a reasonable and effective product identification and counting framework. It consists of Pre-processing, Detection-Tracking-Classification(DTC) module and Post-processing. Finally, we get the first place in test A.

2) Aiming at the issues of aliasing and too high brightness in the images of the training set, we enhance the training images with Multi-Scale Retinex with Color Restoration(MSRRC) algorithm [11], makes up the quality difference between training and test images, which greatly improves the performance of object detection and classification.

3) We also present a specific post processing for the output of DTC module, and name it as MTCR algorithm which improves the accuracy of classification.

2.1. Object Detection

Object detection task is to find out all the objects of interest in images, and locate their positions. It is one of the core problems in computer vision and an essential part of our entire pipeline. There are two kinds of categories of object methods: one-stage methods [13, 15, 20, 22] and two-stage methods [4, 9, 21, 25]. One-stage means that extract features directly in the network to predict classification and location without region proposal. Many modern object detectors demonstrate outstanding performances by using the mechanism of looking and thinking twice, the first step is to give a region proposal may contain objects and then classify and locate through a CNN module, which is also the overall idea of the two-stage method. Existing detectors are trained using large-scale datasets. How to complete the detection task without training data is a difficult problem.

2.2. Tracking

The state-of-the-art methods of Multiple Object Tracking(MOT) [8, 16, 27] usually divide into Two-Step MOT and One-Shot MOT. Two-step MOT [1, 6, 7, 27] methods often treat object detection and association as two separate tasks which first use detectors to localize all objects of interest in the frame of videos, and then in a separate step they crop object regions according to the boxes and feed them to the identity embedding network to extract Re-ID features to associate objects and tracks. However, One-Shot MOT [26] simultaneously accomplish object detection and identity embedding in a single network that can reduce the inference time through sharing most of the computation. In this competition, we pay more attention to the tracking accuracy rather than the speed, so we chose DeepSORT [27] which allows users to better optimize the detector and tracker for the tracking effect.

2.3. Classification

Image classification is a fundamental task in computer vision research. Recent work such as object detection, semantic segmentation has significantly boosted image classification which serves as the backbone. In recent years, with the development of deep learning networks, a number of classification networks with good performance have emerged. Like transformer [24] leverages attention mechanism to improve model training speed which has higher accuracy and performance than the previously popular RNN [29]. Efficientnet [23] uniformly scales depth, width and resolution with a fixed set of scaling factors for improved accuracy. ResNeSt [30] is a deformation model of ResNet [10] by stacking Split-Attention block that enables across feature-map groups. For improving the accuracy of classification, data augmentation [2, 3] is an effective way. The most common data augmentation methods are random erasing [31], resize, random flip, random gray scale and so on. By data augmentation, the quality of images has obviously improved, allowing it to be processed better.
3. Method

In order to count and recognize products in Track 4 accurately, we propose a system that contains pre-processing, detecting-tracking-classifying (DTC) and Post-processing modules as shown in Figure 2, which takes frames extracted from test set A as input. Firstly, the frames are fed into pre-processing module, producing cropped and masked frames. Secondly, the processed frames are fed into detection network, outputing location bounding boxes. Then, frames with location message are fed into DeepSort and classification network, generating trajectories with category scores. Finally, counting and recognition results could be gotten by MTCR system from the trajectories.

3.1. Pre-processing Module

In order to solve the problem of image aliasing and too dark brightness in the training set, we have made many attempts to make the distribution of RBG in the training dataset and the distribution of the test dataset as close as possible to improve the detection and classification effect of this scheme.

The most important processing methods at this stage are data augmentation with MSRCR [11]. As shown in the top of Figure 3, the images in the original dataset are of poor quality, most of them are too bright or too dark, we exploit MSRCR. The key formula of MSRCR’s algorithm is:

\[ Dif f_{i,c} = log(I_i) - log(I_i * G_{c,i}), \]  

\[ MSRCR = \sum_{i=1}^{N} Dif f_{i,c} / 3, \]  

\[ MSRCR = MSRCR * (\log(125I) - \log(I_1 + I_2 + I_3)), \]  

where \( I \) is the input color image, \( c \in \{ R, G, B \} \), \( i \) is the scales, \( i = 1, 2, 3 \) means three color channels.

MSRCR has perfect performance in image enhancement by applying it on each color channel independently, so that a color balance can be effectuated. Experiments illustrate that the images enhanced by MSRCR can improve the performance of both detector and classifier. For detector, the high quality of images can be used to detect and localized well. And for classifier, the enhanced images can be easily distinguished by classification network model with high accuracy, for the reason that the enhanced training set image RGB distribution is closer to the test set RGB distribution.

Different from the training dataset with only images, the test datasets are videos, and someone is holding the product for detecting. For the purpose of eliminating this difference, people need to be detected and segmented, then use the first frame that usually does not contain people to complement. Using open source model for segmentation may leave some areas that cannot be masked.

3.2. DTC Module

For accurate identification and counting of products, our core module in this framework is DTC Module. The DTC module mainly includes three parts: detector, tracker and classifier.
3.2.2 Tracker

Object tracking is necessary since each object is only counted once while it passes through the checkout region. DeepSort [27] is one of the most typical methods of tracking-by-detection category. In our system, we apply DeepSort to associate the product detections of different frames in test videos, and get the trajectories of products. In order to extract more discriminative features for object association, we exploit the last layer features of our classifier to represent appearance of products, and then fill them to the cost matrix, and use the Hungarian algorithm [12] to match products in the DeepSort algorithm. Benefitting from image enhancement in section 3.1, we extract more expressive appearance features for object association from our classifier. Therefore, the feature matching process is more stable and robust.

3.2.3 Classifier

Accurate classifying products is a critical component of our system, so we pay more attention to designing our classifier network which is used to classify each object from one product trajectory we gotten. We choose multiple classifiers with better performance on the ImageNet [5], such as ResNeSt [30], EfficientNet [23], etc. Different models have complementary effects on different categories.

An excellent classifier could classify hard negative samples as other categories, so that they can be eliminated before subsequent processing. To depress misclassification, such as human hands, empty white trays, etc., we use false detections on our homemade detection training set and the randomly cropped region on it as the background category of our classifier training, motivating from the practice of two-stage detectors, such as Faster R-CNN [21]. Because of adding background category, we can classify the hard negative samples correctly and improve the recognition function of DTC module.

In general classification tasks, we use the hard label method to perform one-hot encoding on the label, and adding a little noise to the one-hot encoding obtained by the hard label can make the network convergence effect better.

In the classification task, the cross-entropy loss is as follows:

$$H(y, y') = -\sum_{i=1}^{K} y_i * log(y_i').$$

(4)

The $y'$ after label smooth operation is as follows:

$$y'_i = \begin{cases} 1 - \varepsilon, & i = \text{target} \\ \frac{1}{K-1}, & \text{otherwise} \end{cases},$$

(5)

where $\varepsilon$ is an artificially specified number of 0-1, $K$ is the number of categories.

In the part of classifier, we use multiple networks to train the classifiers and add background category to the training set which make our classifier works well.
3.3. MTCR Module

In order to further improve the accuracy of our results, we input the results from DTC module to MTCR module, which further improves the product counting and recognition. The entire MTCR module is as shown in Figure 4, and illuminated in Algorithm 1. Firstly results of DTC module are fused by Model fusion algorithm; secondly, problematic trajectories are split to depress mistakes by Track splitting algorithm. And then, Category voting algorithm determines the category of trajectories. Finally, the Results fusion algorithm merges the system output results.

Model fusion. To take advantage of different models, we fuse the results from 3 different models by averaging category scores. We use $s_{ijkl}$ to represent the $l$th category score in the $j$th trajectory and the $k$th frame from the $i$th model after DTC module. And we use $s_{jkl}$ to represent the $l$th category score in the $j$th trajectory and the $k$th frame after Model Fusion. Formally, the category score after Model Fusion $s'_{jkl}$ is defined as:

$$s'_{jkl} = \sum_{i=1}^{3} s_{ijkl} / 3.$$  \hspace{1cm} (6)

Comparing to weighting all model results, our approach is more universal and not just achieving recognition and counting task on the video of TestA.

Track splitting. There is a case of disconnecting two trajectory with a long time interval in DTC module. To solve the problem, we split the trajectory with a long time interval. We use $f_{jk}(f_{jk} \in T_{j})$ to represent the frame index of the $k$th frame in the $j$th trajectory, and $f_{thr}$ to represent the time threshold of long time interval. We split the trajectory $T_{j}$ when $f_{jk+1} - f_{jk} > f_{thr}$ happens.

Category voting. The synthetic images can not include all perspective of products, which could lead to aweful scores for some frames in a predicted trajectory and matter classification. Thus, we regard the category having top score as the prediction of frame and vote for the category of a trajectory with them to avoid this extreme situation. We defined scores in the $j$th trajectory and the $k$th frame $S_{jk}$ as $S_{jk} = \{s_{ijkl} \mid l \in \{1, 2, \ldots, 117\}\}$. The index $l$ of max score $s_{ijkl}$ in $S_{jk}$ represents the category of the $k$th frame in the $j$th trajectory and we use a counter $\text{counter}_{l}(l \in \{1, 2, \ldots, 117\})$ to record the times of index $l$ appearing. And the most frequent index represents the category of a trajectory, which is assumed as temporary $c_j$. And then, we use $\hat{s}_{j}$ to represent the score of the $j$th trajectory, which is defined as:

$$\hat{s}_{j} = \sum_{k=1}^{m_j} \frac{s'_{jkl}}{m_j}, \hspace{1cm} (7)$$

where $m_j$ denotes the number of frames in the $j$th trajectory.

Comparing to firstly averaging category scores of frames in a trajectory and then selecting the category having the top score which we do in most cases, selecting firstly and voting then averaging the category elected scores avoids the effects of extreme scores, which is more robust.

Results fusion. Because of the lack of data for training detectors, we can not detect a product continuously at some time, which might lead to Cut-off error in tracking task. Thus, we fuse the results which have the same category and the neighbor timestamps. We use $p_i = (c_i, t_i)$ to represent
**Algorithm 1** The MTCR module which is post-processing of our system.

**Input:** \( A = \{s_{ijkl} | i \in \{1,2,3\}, j \in \{1,...,n\}, k \in \{1,...,m_j\}, l \in \{1,...,117\}\), \(s_{ijkl}\) to represent the \(l\)th category score in the \(j\)th trajectory and the \(k\)th frame from the \(i\)th model after DTC module. \(A\) is a set of \(s_{ijkl}\), \(m_j\) is the number of frames in the \(j\)th trajectory; \(S_{jk} = \{s'_{jk1}, s'_{jk2},...,s'_{jk117}\}\), \(S_{jk}\) is a set of category scores, \(F_{jk}\) include \(S_{jk}\) and frame index \(f_{jk}\); \(T_j = \{F_{j1}, F_{j2},...,F_{jm_j}\}\), \(T_j\) is a set of \(F_{jk}\); \(M = \{T_1, T_2,..., T_n\}\), \(M\) is a set of \(T_j\), representing trajectories in a model.

**Output:** a set of predictions, \(P\)

1: function MTCR(A)
2: \(P \leftarrow \{\}\)
3: \(M = Model\ Fusion(A)\)
4: for all \(T_j \in M\) do
5: \(ST_j \leftarrow Track\ Splitting(T_j)\)
6: for all \(T'_j \in ST_j\) do
7: \((c_j, s'_j) \leftarrow Category\ Voting(T'_j)\)
8: if \(|T'_j| > t_1\) & \(c_j < 117\) & \(s'_j > t_2\) then
9: \(t_j = average(T'_j)\) by \(f_{jk}\)
10: append \((c_j, t_j)\) to \(P\)
11: end if
12: end for
13: end for
14: \(P \leftarrow Result\ Fusion(P)\)
15: return \(P\)
16: end function

the prediction before Result fusion. Firstly, we use \(P_c\) to represent the prediction group that have the same category \(c\). Secondly, we search for \(p_i\)s which have neighbor \(t_i\) and average \(t_{i}\)s to form a new prediction. We use \(t_{thr}\) to represent the interval threshold and fuse the \(p_i\) and the \(p_j\) when \(|t_i - t_j| < t_{thr}(p_i, p_j \in P_c)\) happens.

Not only could Result fusion module make up intermittent detection, it also could compensate the mistakes caused by Track splitting, which makes our approach more robust.

Our post-processing follows: model fusion-track splitting-category voting-result fusion. The post-processing method alleviated the problems such as continuous missed detection, Mix-up error in tracking task, lack of perspectives of products data, which makes our approach more robust. The detail of four parts of MTCR Algorithm is shown in Algorithm 2.

4. Experiments

We demonstrate the effectiveness of our proposed framework on the test set \(A\), in which the customer hold the merchandise item. And train our models on training set with synthetic images. All results follows are performed on the test set \(A\).

**Algorithm 2** algorithm functions of the MTCR module.

1: function MODEL\ FUSION(A)
2: for \(s_{ijkl} \in A\) do
3: Formula 6
4: update \(s_{ijkl}\) in \(M\)
5: end for
6: return \(M\)
7: end function
8: 9: function TRACK\ SPLITTING(T)
10: Sort \(T_j\) by \(f_{jk}\)
11: \(ST \leftarrow \{\}\), \(f_{id} = f_{j1}, new.T = \{F_{j1}\}\)
12: for \(i = 2 \rightarrow m_j - 1\) do
13: if \(|f_{j1} - f_{id}| < f_{thr}\) then
14: append \(F_{ji}\) to \(new.T\), \(f_{id} = f_{ji}\)
15: else
16: append \(new.T\) to \(ST\)
17: \(f_{id} = f_{ji}\), \(new.T = \{F_{ji}\}\)
18: end if
19: end for
20: append \(new.T\) to \(ST\)
21: return \(ST\)
22: end function
23: 24: function CATEGORY\ VOTING(T_j)
25: \(counter = \{0\} * 117\)
26: for \((f_{jk}, S_{jk}) \in T_j\) do
27: \(c = \text{MaxIndex}(S_{jk})\), \(counter(c)++\)
28: end for
29: \(c_j = \text{MaxIndex}(counter)\)
30: Formula 7
31: return \(c_j, s_j\)
32: end function
33: 34: function RESULT\ FUSION(P)
35: \(P_{new} = \{\}\)
36: Group \(P\) by Category, get \(P_c\)
37: Sort \(P_c\) by \(t_{ci}\)
38: for \(c \in c\) do
39: \(P_t \leftarrow \{(c', t_{c'i})\}, f_{id} = t_{c'i}\)
40: for \(i = 2 \rightarrow |P_t|\) do
41: if \(|t_{c'i} - f_{id}| < t_{thr}\) then
42: append \((c', t_{c'i})\) to \(P_t\), \(f_{id} = t_{c'i}\)
43: else
44: \((c', t')\) = average \(p_i\) by \(t\)
45: append \((c', t')\) to \(P_{new}\), \(f_{id} = t_{c'i}\)
46: end if
47: end for
48: \((c', t')\) = average \(p_i\) by \(t\)
49: append \((c', t')\) to \(P_{new}\), \(f_{id} = t_{c'i}\)
50: end for
51: return \(P_{new}\)
52: end function
### Table 1
Comparison of the scores with different teams, our team ID is 16, and we get 1.0000 in the test A, means that all the merchandise items are classified correctly.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team ID</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>1.0000</td>
</tr>
<tr>
<td>2</td>
<td>94</td>
<td>0.4783</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
<td>0.4545</td>
</tr>
<tr>
<td>4</td>
<td>165</td>
<td>0.4400</td>
</tr>
<tr>
<td>5</td>
<td>66</td>
<td>0.4314</td>
</tr>
<tr>
<td>6</td>
<td>76</td>
<td>0.4231</td>
</tr>
<tr>
<td>7</td>
<td>117</td>
<td>0.4167</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>0.4082</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>0.4000</td>
</tr>
<tr>
<td>10</td>
<td>55</td>
<td>0.4000</td>
</tr>
</tbody>
</table>

### 4.1. Datasets
The dataset contains a total of 116,500 synthetic images from 116 different merchandise items for training and 5 video clips appearing these items for testing in test set A. The synthetic images are created from 3D scanned object models. The perspective of the test video clips is above the checkout counter and facing straight down while a customer is pretending to perform a checkout action by “scanning” objects in front of the counter in a natural manner. And there is a shopping tray placed under the camera to indicate where the model should focus.

### 4.2. Implementation details

**Training phase.** We train detection models with input size 500x500 for 5 epochs with the learning rate multiplied by 0.1 after 4 epochs. We train classification models with size 224x224 after crop for 20 epochs with the learning rate multiplied by 0.1 after 5, 10, 15 epochs. To expand the dataset, we load the pre-training model parameters on MS-COCO [14] and ImageNet [3].

**Test phase.** Firstly, we extract frames from video clips in test set A with FFmpeg library. Secondly, using model pretrained on MS-COCO, we detect the white tray and mask person with frame in which its IOU with the white tray is 0. We crop the region of white tray in masked images where we inference our model.

### 4.3. Experimental results
Through a series of iterative optimizations, we gradually improve the performance of our framework. On the official test set A, we try to evaluate the performance of our framework. Incredibly, our method achieves a score of 1.0000, outperforming the performance of all other team methods. The results of all teams are shown in Table 1.

### 4.4. Ablation study
In order to prove the effectiveness of each module of our system, we analyze the preprocessing part, the DTC module and the MTMR module separately.

#### 4.4.1 Pre-processing
This part we use EfficientNet-b0 as the classifier and use the same results for detection and tracking. As shown in the Table 2, the first row uses the original images to train classifier, which has poor quality, obviously the result is not satisfactory. Therefore we use MSRCR to enhance the original image to adjust those pixels that are too bright or too dark. We can see image quality has been significantly improved, and it is more suitable for processing. After the fusion training of the two types of data, the results have been improved to a certain extent.

#### 4.4.2 DTC Module
In this stage, our main purpose is to compare the effects of DTC part. And in detection, we visualize the detection results using three different data as shown in Figure 5. The first set of data uses the original images to train detector, which have poor quality. Obviously the detection effect is not satisfactory. Therefore we use MSRCR to enhance the original image. We can see that many products are detected, but in many cases, human hands are also detected by mistake. Taking into account the actual situation during the test, using the masked image is a good idea and a nice boost.

Table 3 shows the results of the comparison. We also use EfficientNet-b0 as the classifier. The first row uses the original images to train detector. After we use MSRCR to enhance the original image, the detection effect is satisfactory obviously. Then after we use classifier’s feature to track, the precision and recall have been improved. We can see in the test video, someone is holding the products for detection, in order to eliminate the interference of the background, focus as much as possible on the products itself, we also mask the hand. The results are best when both classifier’s feature and mask are used.

#### 4.4.3 MTMR Module
As shown in Table 4, we train multiple single models and fuse the models based on their scores on this task. As can
Figure 5. Strong contrast when the same type of product is used for detection. Image after segmentation can be best detected, the only error-checked class is the background class while original image are difficult to be detected and the image after augmentation has many false positive boxes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Ori</th>
<th>MSRCR</th>
<th>Track</th>
<th>Mask</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effic-b0</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.8620</td>
<td>0.8620</td>
<td>0.8620</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>0.8436</td>
<td>0.9310</td>
<td>0.8852</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>0.8710</td>
<td>0.9310</td>
<td>0.9000</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.9310</td>
<td>0.9310</td>
<td>0.9310</td>
</tr>
</tbody>
</table>

Table 3. Comparison with different methods in the DTC part. Track and Mask are used to enhance detection accuracy. “✓” means this method is used. “-” means this method is not used.

<table>
<thead>
<tr>
<th>Method</th>
<th>MF</th>
<th>TCR</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>-</td>
<td>✓</td>
<td>0.9032</td>
<td>0.9655</td>
<td>0.9333</td>
</tr>
<tr>
<td>M2</td>
<td>-</td>
<td>✓</td>
<td>0.9355</td>
<td>1.0000</td>
<td>0.9667</td>
</tr>
<tr>
<td>M3</td>
<td>-</td>
<td>✓</td>
<td>0.9867</td>
<td>1.0000</td>
<td>0.9831</td>
</tr>
<tr>
<td>Fusion</td>
<td>✓</td>
<td>✓</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Table 4. Comparison with different methods in the MTCR algorithm. M1 is EfficientNet-b2, M2 is ResNet50, M3 is ResNet101 and Fusion means all the three models are used together to get a robust result. MF means Model Fusion, TCR means the mix of Track splitting, Category voting and Results fusion. “✓” means this method is used. “-” means this method is not used.

be seen from the results, each model has its own advantages. After the MTCR algorithm fuses the results of multiple models, the results have been greatly improved.

5. Conclusion

In this paper, we analyze the key issues that need to be addressed urgently in the AICITY2022 Task 4 and propose an effective and excellent solution framework, which include Pre-Processing, DTC and MTCR modules. Our Pre-Processing strategy has well enhanced the quality of training images and makes up the data difference between training set and test set, which improves the accuracy of classification highly. DTC module gives a clear process flow that inputs are all frames of the test video and gets the trajectory of each product and the multi-class score of the trajectory on each frame. And the MTCR module as post-processing makes robust recognition and counting of products, and it’s also suitable for other vision tasks. Extensive experiments demonstrate the effectiveness of our system. The whole framework have achieved significant improvement compared to the other teams. In the future, we would mainly focus on how to improve the speed of our method.

6. Acknowledgements

This work is supported by Chinese National Natural Science Foundation under Grants (U1931202, 62076033).

References


