IncreACO: Incrementally Learned Automatic Check-out with Photorealistic Exemplar Augmentation

Yandan Yang[1], Lu Sheng[2]*, Xiaolong Jiang[3], Haochen Wang[1], Dong Xu[4], Xianbin Cao[1,5,6]
[1]School of Electronic and Information Engineering, Beihang University, Beijing, China
[2]College of Software, Beihang University, Beijing, China
[3]Alibaba Youku Cognitive and Intelligent Lab, Beijing, China [4]University of Sydney, Sydney, Australia
[5]Key Laboratory of Advanced Technology of Near Space Information System,
Ministry of Industry and Information Technology of China
[6]Beijing Advanced Innovation Center for Big Data-Based Precision Medicine, China
{yangyandan,lsheng,haochenwang,xbcao}@buaa.edu.cn,
xainglu.jxl@alibaba-inc.com, dong.xu@sydney.edu.au

Abstract

Automatic check-out (ACO) emerges as an integral component in recent self-service retailing stores, which aims at automatically detecting and counting the randomly placed products upon a check-out platform. Existing data-driven counting works still have difficulties in generalizing to real-world retail product counting scenarios, since (1) real check-out images are hard to collect or cover all products and their possible layouts, (2) rapid updating of the product list leads to frequent and tedious re-training of the counting models. To overcome these obstacles, we contribute a practical automatic check-out framework tailored to real-world retail product counting scenarios, consisting of a photorealistic exemplar augmentation to generate physically reliable and photorealistic check-out images from canonical exemplars scanned for each product and an incremental learning strategy to match the updating nature of the ACO system with much fewer training effort. Through comprehensive studies, we show that the proposed IncreACO serves as an effective framework on the recent Retail Product Check-out (RPC) dataset, where the proposed photorealistic exemplar augmentation remarkably improves the counting performance against the state-of-the-art methods (77.15% v.s. 72.83% in counting accuracy), whilst the proposed incremental learning framework consistently extends the counting performance to new categories.

1. Introduction

Recent self-service retailing stores are trying to release the human labors from tedious retailing product checking-out, by the means of emerging automatic check-out (ACO) system. This system functions when observed the unmanned check-out platform with cluttered products picked by the consumers; it generates a shopping list, telling the numbers of products of each category, as shown in Fig. 1. This system is analogous to object counting in the computer vision community, but it still faces challenges, especially in real-world check-out scenarios:

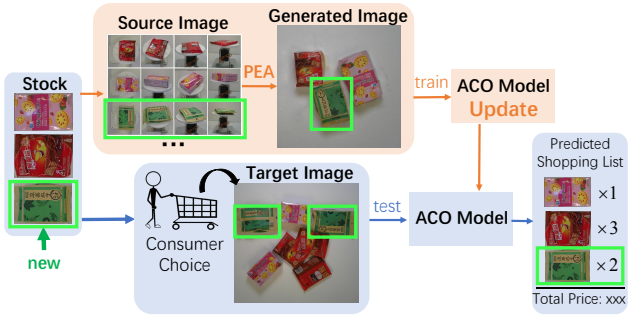

Figure 1. The incrementally update process of the automatic checkout (ACO) system with our IncreACO framework. We generate images from isolated source exemplars and incrementally update the ACO model to recognize new products.

(1) Learnable object counting models heavily rely on carefully labeled check-out datasets covering all available products and their possible poses, interactions, layouts, and *etc*. However, these images are hard to collect all at once due to the well-known combinatorial explosion and when the categories have to be updated frequently.

(2) Practical ACO systems have to be frequently updated due to the rapid updating of product categories in real ACO scenarios. However, it is time-consuming and unscalable to train a counting model from scratch with a large dataset including the old and new products.

*Corresponding Author

To deal with the first challenge, prior arts employ deep generative models to synthesize check-out images by sampling and re-arranging canonical exemplars of products captured from a set of fixed viewpoints and poses, such as Wei *et al.* [25]. Nevertheless, the produced images are far from realistic and usually ignore the physical reliability of object layouts, leading to abnormal object poses and interactions among objects. Thus we would like to explore a physically reliable data augmentation, called Photorealistic Exemplar Augmentation (PEA), that comprehensively renders unlimited novel check-out images with diverse but reliable object layouts, so as to minimize the domain gap between the real and synthesized check-out data distributions.

Furthermore, the second challenge can be alleviated by the recent advances of incremental learning frameworks, which also mitigates the dataset scale required for training the new categories. It just requires a limited amount of labeled check-out images including randomly placed products both from new and old categories. However, the ACO task is analogous to fine-grained object detection, thus simple adoption of conventional data-separated incremental learning pipeline is not applicable and would inevitably result in confusion of products with similar patterns. Therefore, an ACO-specific incremental learning framework becomes a necessity and to the best of our knowledge, we are the first to explore such a pipeline in this special application.

To this end, we propose a novel learnable framework towards the practical automatic check-out system, namely as incrementally learned automatic check-out with photorealistic exemplar augmentation (a.k.a. IncreACO), so as to resolve the aforementioned challenges simultaneously in an integrated way. The overall framework of our IncreACO is shown in Fig.1.

We first propose a novel Photorealistic Exemplar Augmentation (PEA). It improves the realness of the generated check-out images by taking the perspectives, layouts, and occlusions of real check-out images into consideration, rather than just naïve pasting the exemplar product patches onto a check-out background image. Given the product patches extracted from the canonical exemplar images with a fixed set of poses [25], we at first select candidate product patches whose poses may occur within the check-out scenarios. Then we decide where to place the candidate items in the check-out platform. Finally, we calculate the degree of overlapping between candidate samples and compare them with a pre-defined threshold to avoid severe occlusions. These three steps encourage more physically realistic spatial product distribution by considering both the geometric structure of each product and the geometrical relations between products. The synthesized check-out images are then rendered by CycleGAN [26] to mimic real data. Our synthesized images are much closer to the real check-out images than those of previous works [14, 25].

We further propose an incremental learning pipeline tailored to the automatic check-out scenario, for the purpose of fast adapting to the rapid updated retail product categories and resolving the absence of labeled check-out data. The proposed photorealistic exemplar augmentation, as an effective high-quality data provider, effectively generates diverse but customized samples for the learning of novel classes. Together with the retroactivity towards the old product categories, we employ the distillation-based incremental learning framework [15, 24] to learn the object detection for the novel categories. We employ Mask R-CNN [7] as the backbone for object detection and count the number of each product accordingly. Experiment results show that the proposed photorealistic exemplar augmentation generates superior check-out samples, which can significantly boost the counting performance of our automatic check-out model on a recent large-scale retail product checkout (RPC) dataset [25]. Moreover, our ACO-specific incremental learning framework also preserves the check-out performance with progressively updated categories, which only required a fractional amount of novel samples together with a small retroactive factor.

2. Related Work

Retail Product Checkout Datasets. There are multiple datasets about retail product check-out and each of them has different settings. SOIL-47 [13] is a small-scale dataset that contains 46 categories, and each of which has 21 images. The recorded products have diverse color distributions. SKU110 [4] is a large-scale dataset, where 11762 images are collected in densely packed shelf scenarios, while no class information is provided. Meanwhile, Supermarket Produce Dataset [21] and Freiburg Groceries Dataset [9] both include thousands of images for grocery products, but they only focus on classification. Moreover, Grocery Product Dataset [3] contains 80 product categories with 8350 training images, while it tests on the shelf loaded products. Most of the mentioned datasets collect train and test sets from different domains, and none of them is developed for the check-out scenario. Recently, Wei *et. al.* [25] propose a large-scale Retail Product Dataset (RPC) designed for automatic check-out (see Sec. 4.1.1 for more details), where photorealistic data generation that transforms the training exemplars to the testing domain is vital to train a reliable automatic check-out model that works in real scenarios.

Data Augmentation. Data augmentation operations are practical solutions to enrich data to avoid overfitting and improve the generalization of the trained models [22, 18, 23]. Moreover, when the labeled real data are expensive to obtain, the data augmentation is extended to employ image synthesis to economically generate large-scale training data that match the distribution of the testing data. For instance, CDVAE [17] applies the variational autoencoder (VAE) [11] to encode the source data into the latent space and then de-

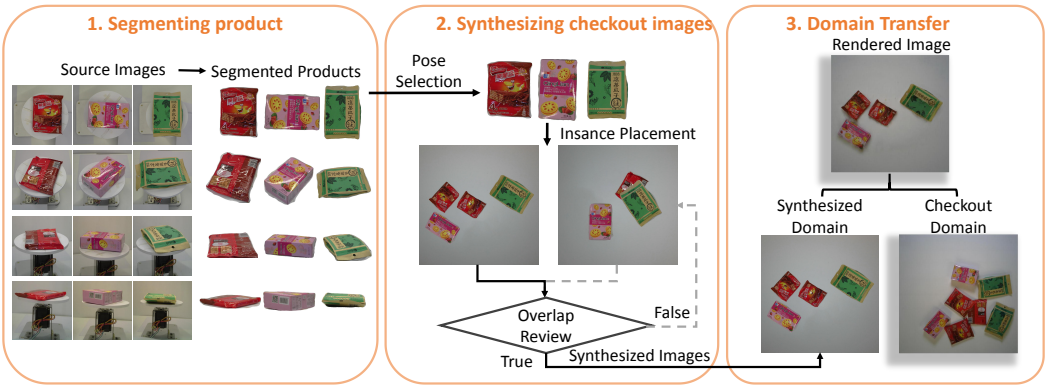

Figure 2. The proposed photorealistic exemplar augmentation.

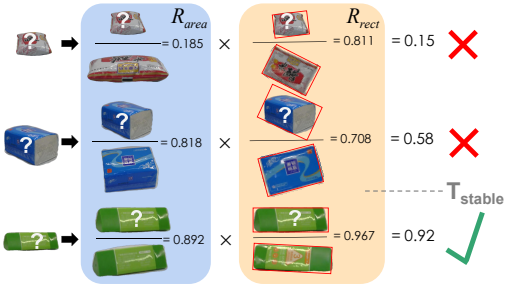

Figure 3. Samples of pose selection. The stability factor R_{stable} is the product of R_{area} and R_{rect}. The unstable poses with stability factor less than T_{stable} are removed.

code it into the target domain. More recently, Goodfellow *et. al.* [5] propose the Generative Adversarial Networks (GAN) to synthesize data in the target domain. For the product retail check-out tasks, data augmentation also follows the image synthesis pipeline. For example, Wei *et. al.* [25] propose to extract single product patches out of training images and then mix multiple patches into a single synthesized image, after which they employ CycleGAN [26] to transfer the style of real check-out images to these synthesized samples. To improve reliability, Li *et. al.* [14] propose to discard small-area patches with invalid perspective angles. However, the generated images are still lack of physical reliability as geometrical relations, are poorly considered. Compared with these methods, our method further explores geometrically reliable product layouts in real check-out scenarios and generates more photorealistic data.

Incremental Learning Incremental learning [1] is to continuously learn new categories, tasks or domains, while avoiding tremendous performance drops, a.k.a, catastrophic forgetting, on the previous data. Most previous works focus on learning classifier. They are built on different data settings. (1) *Strict*: only new categories' data are accessible during incremental learning. LwF [15], EBLL [19], and EWC [12] apply knowledge distillation [8] and use the output of the previous frozen model as the soft labels of old classes when training with novel data. (2) *Relaxed*: a small volume of old data was allowed to help alleviate catastrophic forgetting. In incremental learning, iCaRL [20] selects the most representative samples of old classes and jointly trains the classifier with data of new classes. It is easy to separate the data for classification since each image belongs to a single category. Nevertheless, for detection, a single image might contain multiple objects, including both previous and current classes. There are some attempts on incremental learning of object detectors [24, 6] using PASCAL VOC 2007 [2] and MSCOCO 2017 [16]. However, it is somewhat hard to select reliable labeled detection data to incrementally learn the new categories. ILWCF [24] simply ignores the labels of unwanted categories during training, which would confuse the model. And CIFRCN [6] discards the images which consist of both previous and current objects, while it suffers from the shortage of training data. In our work, the photorealistic exemplar augmentation tries to solve this problem using image synthesis techniques.

3. Methodology

Our IncreACO framework would like to incrementally learn automatic check-out that is tailored to the frequently updated category list in a real retail product check-out system. This framework consists of two vital components, one is the photorealistic exemplar augmentation, and the other is the incremental learning mechanism towards the product counting model. Note that our data augmentation can be seamlessly incorporated into our incremental learning pipeline, so that the complete IncreACO framework can simultaneously resolve the aforementioned challenges with respect to data and learning, respectively.

3.1. Photorealistic Exemplar Augmentation

Based on single-product images (exemplars) captured from a fixed set of camera viewpoints, such as Retail Product Check-out (RPC) Dataset [25], the goal of the photorealistic exemplar augmentation (PEA) is to synthesize photorealistic multi-products images with reliable physical interactions and layouts, analogously to the real check-out images captured from the check-out platform.

Our PEA consists of three stages, as shown in Fig. 2. The first stage segments the product instances from these exemplar images. The second stage synthesizes multi-

product check-out images by randomly pasting the product instances via a set of geometry-aware criteria. The third stage renders the photorealistic check-out images from the synthesized images. In the first and third stages, we apply the off-the-shelf solutions proposed by Li [14] and CycleGAN [26], while our contributions lie on the second stage. The second stage has three vital geometrical criteria:

(1) Pose Selection. Given a bag of single-product exemplar images, we at first select the instances with valid poses (or called viewpoints according to the cameras) that are possible in the real check-out system. For example, in the training set of RPC dataset [25], products with some poses (such as the first two instances in Fig. 3) are rare in real check-out scenarios. Therefore, we propose a stability factor to measure how "stable" a product's pose would be:

$$R^{c,i}_{stable} = R^{c,i}_{area} \cdot R^{c,i}_{rect}, \tag{1}$$

which includes two parts, as illustrated in Fig. 3. $c \in \mathcal{C}$ means a specific class in the product category list, i denote the instance index in the class c. R_{area} measures the normalized mask area of the product instance over the largest mask area within all available product instances from this category, where a larger area indicates the placement more stable in the check-out image, *i.e.*,

$$R^{c,i}_{area} = A(m^i_c)/\max_j A(m^j_c), \tag{2}$$

where m^i_c represents the segmentation mask of a product instance in class c, and $A(\cdot)$ calculates the area of a mask. Since retail products are usually designed in regular shapes, such as cuboid or cylinder, to ease the product packaging. Thus stable placement will usually share a near regular silhouette (*e.g.* rectangles), such that

$$R^{c,i}_{rect} = \frac{A(m^i_c)/A(\text{rect}(m^i_c))}{\max_j A(m^j_c)/A(\text{rect}(m^j_c))}. \tag{3}$$

rect$(\cdot)$ computes the minimum enclosing rectangle of this mask. We remove the invalid pose whose stability factor is fewer than a threshold, $R^{c,i}_{stable} < T_{stable}$. For the candidate poses, the one with a larger stability factor takes a higher chance to be selected.

(2) Instance Placement. In real check-out images, products are usually placed near to each other, but with fewer occlusions between each other. Practically, we propose to place a new product instance, *i.e.* its mask, near a previously chosen object. As shown in Fig. 4, we have already placed two candidate product instances onto the background image with mask m_{old}. Then we randomly pick a point P from the edge of m_{old} and sample a surrounding point Q according to a Gaussian distribution centered at P, which will act as a corner of the next product instance. We then compute the minimum enclosing rectangle of the previous product instance where P lies in:

$$(s_0, s_1, s_2, s_3) = \text{rect}(m^i_{old}), \text{ if } P \in m^i_{old}, \tag{4}$$

where i is the index of the previously chosen product instance. s_0, s_1, s_2, s_3 are four clock-wisely ordered vertices,

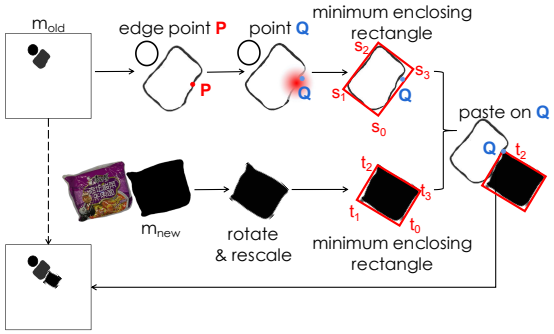

Figure 4. Instance placement. We randomly choose an edge point P from m_{old} and sample a surrounding point Q. The nearest edge of the MER of m^i_{old} of P to Q is (s_0, s_3). We compute MER for the new instance m_{new} after rotation and re-scaling and randomly choose t_2 from $\{t_1, t_2\}$ to paste on Q.

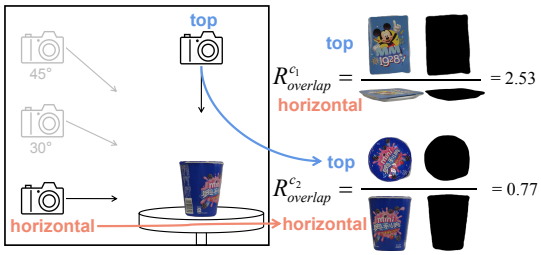

Figure 5. The overlap factor for each category is the ratio of two areas: the minimum area of mask from the top view and the maximum area of mask from the horizontal view.

and the bottom vertex is s_0. The rectangle's edge that is nearest to the point Q is marked as (u, v), $u, v \in \{0, 1, 2, 3\}$ are the indices of the four vertices, such that $(0, 3)$ means the edge whose vertices are s_0 and s_3 in Fig. 4. m_{new} is mask of the new product instance. After a random rotation and re-scaling, we also compute its MER:

$$(t_0, t_1, t_2, t_3) = \text{rect}(\text{scale}(\text{rot}(m_{new}))). \tag{5}$$

rot$(\cdot)$ is the random rotation. scale$(\cdot)$ re-scales the products from 0.5 to 0.7. t_0, t_1, t_2, t_3 are four vertexes of the minimum enclosing rectangle. Note the vertices are ordered in the same manner as the old masks. To avoid large overlap with existing objects, we choose a vertex t_k from the vertices out of t_u and t_v and let it paste on Q.

(3) Overlap Review. By the instance placement step, the previous product instances may still be occluded. Note that different product category has different geometric structure, thus they have different chances to be occluded, *e.g.* flat objects, such as notebooks, are easy to be occluded while bulging stuff, such as juice bottles, are hard to be occluded in practice. We design an overlap factor to quantify this:

$$R^c_{overlap} = \min_i A(m^i_{c,top}) / \max_i A(m^i_{c,horizontal}), \tag{6}$$

where $m^i_{c,top}$ is the mask from the top view, and $m^i_{c,horizontal}$ is the mask from the horizontal view, c is the product category. Fig. 5 indicates different overlap factors. We then convert this factor into a threshold:

$$T^c_{overlap} = \begin{cases} 0 & , \ R^c_{overlap} < 0.5 \\ (R^c_{overlap} - 0.5) \times 0.2, & R^c_{overlap} >= 0.5 \end{cases}, \tag{7}$$

which is a piecewise function, indicating to what degree the overlap caused by the instance placement can be tolerated. Since only previous instances can be covered by the new instance, the overlap degree between the new product instance and each previous instance is defined as:

$$IOU^i_{old} = A(m^i_{old} \cap m_{new})/A(m^i_{old}). \tag{8}$$

If IOU^i_{old} of each old product is lower than the threshold $T^c_{overlap}$ of its category, the newly placed object will be retained. Otherwise, we go back to the last step to re-place the instance until it satisfies the overlap criteria.

3.2. Incremental Learning Mechanism

In this section, we first present our model training strategy based on incremental learning and then introduce the retroactive factor in incremental dataset powered by the photorealistic exemplar augmentation.

3.2.1 Model training strategy

In the beginning, the model is well trained with a basic dataset, including C_o previous product categories. When C_n new product categories arrive, we generate a new dataset, named as the incremental dataset. Based on this dataset, we learn a new model via incremental learning. Referring to LwF [15], we deploy the previously well-trained model as the teacher while the newly-incremented model as the student. The teacher is frozen during the incremental learning phase and the student is first initialized as the teacher except for the new components in the last layer.

Similar to the object detection model, the product counting model has two heads: a classifier to recognize the object and a regressor to locate it. The output of the teacher serves as the soft label for the C_o old categories to the student, which is called knowledge distillation [8]. For the classifier, the distillation loss is written as:

$$L_{old\_cls}(y_o, \hat{y}_o) = -C_o T^2 \sum_{i=1}^{C_o} y'^{(i)}_o \log \hat{y}'^{(i)}_o, \tag{9}$$

where $y_o, \hat{y}_o$ are the outputs of the teacher and student models over the old classes. y'_o and $\hat{y}'_o$ are the soften distributions by a pre-defined temperature T, i.e., $y'^{(i)}_o = (y^{(i)}_o)^{1/T}/\sum_j (y^{(j)}_o)^{1/T}$, $\hat{y}'^{(i)}_o = (\hat{y}^{(i)}_o)^{1/T}/\sum_j (\hat{y}^{(j)}_o)^{1/T}$. Here we set $T = 1$. For the regressor, we apply the smooth ℓ_1 loss between the teacher and student models for knowledge distillation:

$$L_{old\_reg}(t_o, \hat{t_o}) = \sum_{m=x,y,w,h} \sum_{i=1}^{C_o} \text{smooth}_{\ell_1}(t^{i,m}_o - \hat{t}^{i,m}_o), \tag{10}$$

where $t_o, \hat{t}_o$ are teacher and student models' bounding box regression results. For learning the new classes, the ground-truth label supervises the total $C_o + C_n$ classes with the cross-entropy loss for the classifier and the smooth ℓ_1 loss for the regressor.

3.2.2 Retroactive Factor

We apply our photorealistic exemplar augmentation to generate the basic dataset that treats each previous product category equally. In the incremental dataset, the sampling probability of each new product is p, and that of each old product is αp, where α is a retroactive factor. When $\alpha = 0$, the dataset only includes new product categories. Here we choose $\alpha = 0.05$ to preserve memory of old products and also help learn the correlation between old and new products. Additionally, each image in the incremental dataset is required to include as least one new product.

4. Implementation and Experiments

4.1. Implementation Details

4.1.1 Dataset Preparation

We adopted the Retail Product Check-out (RPC) dataset [25] to evaluate the performance of our proposed IncreACO framework. It has 200 products and 83739 images, in which the 200 fine-grained categories belong to 17 meta categories. There exists a domain gap between the source domain and the target domain. In the target domain, the testing set (24000 images) and validation set (6000 images) include multi-product images on a white check-out platform that mimics the real-world checkout scene. While in the source domain, the training set contains 53739 canonical single-product exemplar images collected in a studio in which four fixed cameras capture the product from the top, $45°$, $30°$, and horizontal viewpoints, respectively (see Fig. 5). The goal is to predict the quantity of each product in the image of the testing set.

4.1.2 Training and Testing Details

To fairly compare our method IncreACO to the previous methods, we referred to the basic pipeline of DPNet [14] and employed Mask-RCNN [7] without the mask head as the backbone to predict objects and compute the product numbers. Our model was developed on PyTorch equipped with NVIDIA RTX 2080 Ti GPU cards. The training and testing images were resized to 800×800. The learning objective was optimized by the SGD optimizer.

Photorealistic Exemplar Augmentation. We referred to the background removal method [14] in the first stage of PEA. $T_{stable} = 0.6$ for the pose selection. And the instance placement took 30 pixels as the standard deviation for the Gaussian distribution. We used CycleGAN [26] to render the final results from the synthesized results, according to [25]. The CycleGAN applied Adam [10] optimizer with an initial learning rate of 0.0002 and was trained for 200 epochs. Training images were resized to 800×800 and cropped to 256×256.

Learning from Scratch. As the oracle of the proposed method, we learned the automatic check-out model based on all available product categories. We generated $100,000$ images by the proposed photorealistic exemplar augmenta-

tion and set the batch size to 8. The learning rate started at 0.01 and decayed by the factor of 10 after 120k iterations.

Incremental Learning. Given 200 fine-grained product categories from 17 meta categories, we randomly chose one fine-grained category out of each meta category and thus obtained 17 fine-grained categories as the new coming products for incremental learning. The rest 183 categories were considered previous products and made the basic dataset. Thus $C_o = 183$ and $C_n = 17$. We set $\alpha = 0.5$ and $N_o = 91,500, N_n = 13075$. For the basic dataset, the model was trained with 180k iterations with a learning rate started at 0.01 and decayed by the factor of 10 after 120k iterations. For the new dataset, the model was trained for 18k iterations with a learning rate started at 0.001 and decayed by the factor of 10 after 12k iterations.

4.1.3 Evaluation Metrics

The testing set has three clutter modes: easy, medium, and hard, which indicate the number of products in each image. We adopt the evaluation metrics proposed by [25], such as Check-out Accuracy (cACC), Average Counting Distance (ACD), Mean Category Counting Distance (mCCD) and Mean Category Intersection of Union (mCIoU).

To meet the scenario of incremental learning, we modify the mCCD to separate the evaluations with respect to the old and new classes, named as mCCD_o and mCCD_n, respectively. We also propose the Mean Category Confidence Score (mCCS) to provide a complementary examination of the model's incremental learning ability, which is also evaluated onto new and old classes,

$$mCCS_o = \frac{1}{C_o}\sum_{k=1}^{C_o}\frac{\sum_{i=1}^{N} P_{i,k}}{\sum_{i=1}^{N} GT_{i,k}}, \tag{11}$$

$$mCCS_n = \frac{1}{C_n}\sum_{k=C_o+1}^{C_o+C_n}\frac{\sum_{i=1}^{N} P_{i,k}}{\sum_{i=1}^{N} GT_{i,k}}, \tag{12}$$

where $P_{i,k}$ and $GT_{i,k}$ are the prediction and ground-truth for the i^{th} image in the k^{th} category, and N is the number of images. A better model lets this metric approach to one.

4.2. Experimental results

4.2.1 Photorealistic Exemplar Augmentation

As introduced in Sec. 3.1, the counting results trained by synthesized images are denoted as *syn*, the results trained by rendered images after CycleGAN are denoted as *render*, and the results by joint training from synthesized and rendered images are denoted as *syn+render*. We compare our oracle model (*i.e.* trained from scratch and by all the categories) with two previous works, RPC [25] as the baseline of this dataset, and DPNet [14] trained with their designed generated data.

The first three rows in Fig. 6 show some samples of the synthesized data of the RPC baseline, DPNet, and our PEA sequentially. Intuitively, in images generated by our method, products are of valid poses and placed more compactly with appropriate occlusions, making the images more realistic and reliable. The fourth row demonstrates our rendered images with the same content of the synthesized images (third row) and extra-textual details, which make the images closer to the real checkout images (last row). Comparing the last two rows, it is hard to distinguish our rendered images and the real checkout images.

Figure 6. (1) The first three rows compare the synthesized images. Orange box marks images with scattered products, Red box shows invalid poses, and yellow box presents severe occlusion. Both RPC baseline (1st row) and DPNet (2nd row) suffer from invalid poses, scattered products, and severe occlusion, which are solved in our method (3rd row). (2) The fourth row demonstrates our rendered images based on the synthesized images (3rd row) and looks close to the real checkout images (last row).

Tab. 1 shows the counting results, which demonstrate the effectiveness of our PEA. Taking the average clutter mode as an example, for *syn*, our counting accuracy (cAcc) is 26.52% compared to 18.07% of DPNet and 9.27% of RPC. For *render*, the style transfer improves the performance to a great extent, such as the cAcc is increased from 26.52% (syn) to 75.3% (render) on average mode. And *syn+render* produces the best results. Specifically, our method achieves

Table 1. Results on the RPC dataset. R:RPC. D:DPNet. O:Ours.

Data	Method	cAcc ↑	ACD ↓	mCCD ↓	mCIoU ↑	mAP50 ↑	mmAP ↑
Easy Mode							
Syn	RPC	18.49%	2.58	0.37	69.33%	81.51%	56.39%
	DPNet	32.35%	1.94	0.3	76.59%	88.01%	66.71%
	Ours	43.58%	1.31	0.19	83.38%	93.45%	71.09%
Ren	RPC	63.19%	0.72	0.11	90.64%	96.21%	77.65%
	DPNet	85.38%	0.23	**0.03**	96.82%	98.72%	83.1%
	Ours	86.91%	0.23	0.04	96.65%	98.66%	86.66%
Syn + Ren	RPC	73.17%	0.49	0.07	93.66%	97.34%	79.01%
	DPNet	86.58%	**0.21**	**0.03**	**97.12%**	98.62%	83.47%
	Ours	**88.06%**	**0.21**	**0.03**	96.95%	**98.78%**	**87.28%**
Medium Mode							
Syn	RPC	6.54%	4.33	0.37	68.61%	79.72%	51.75%
	DPNet	14.48%	3.17	0.27	76.68%	87.65%	63.63%
	Ours	23.6%	2.32	0.19	82.31%	92.59%	66.90%
Ren	RPC	43.02%	1.24	0.11	90.64%	95.83%	72.53%
	DPNet	70.90%	0.49	0.04	95.9%	98.16%	77.22%
	Ours	75.36%	0.45	0.04	96.43%	98.58%	81.39%
Syn + Ren	RPC	54.69%	0.90	0.08	92.95%	96.56%	73.24%
	DPNet	73.20%	0.46	0.04	96.24%	98.19%	77.69%
	Ours	**77.31%**	**0.40**	**0.03**	**96.82%**	**98.75%**	**82.17%**
Hard Mode							
Syn	RPC	2.91%	5.94	0.34	70.25%	80.98%	53.11%
	DPNet	7.48%	4.45	0.26	77.58%	87.89%	62.34%
	Ours	12.62%	3.35	0.19	82.48%	92.25%	65.08%
Ren	RPC	31.01%	1.77	0.1	90.41%	95.18%	71.56%
	DPNet	56.25%	0.84	0.05	85.28%	97.67%	74.88%
	Ours	63.70%	0.70	**0.04**	95.97%	98.09%	78.91%
Syn + Ren	RPC	42.48%	1.28	0.07	93.06%	96.45%	72.72%
	DPNet	59.05%	0.77	**0.04**	95.71	97.77	75.45
	Ours	**66.14%**	**0.64**	**0.04**	**96.35%**	**98.29%**	**79.68%**
Average Mode							
Syn	RPC	9.27%	4.27	0.35	69.65%	80.66%	53.08%
	DPNet	18.07%	3.18	0.26	77.11%	87.74%	63.42%
	Ours	26.52%	2.33	0.18	82.63%	92.54%	66.64%
Ren	RPC	45.60%	1.25	0.10	90.58%	95.5%	72.76%
	DPNet	70.80%	0.52	0.04	95.86%	97.93%	77.07%
	Ours	75.30%	0.46	0.04	96.32%	98.22%	81.07%
Syn + Ren	RPC	56.68%	0.89	0.07	93.19%	96.57%	73.83%
	DPNet	72.83%	0.48	0.04	96.17	97.94	77.56
	Ours	**77.15%**	**0.41**	**0.03**	**96.72%**	**98.37%**	**81.82%**

Table 2. Ablation study about the photorealistic exemplar augmentation on the setting of *syn*.

Methods	cAcc ↑	ACD ↓	mCCD ↓	mCIoU ↑
Baseline	10.55%	4.01	0.33	70.51%
PS	12.89%	3.66	0.3	73.28%
PS+IP	15.98%	3.33	0.27	75.88%
PS+IP+OR	**21.3%**	**2.82**	**0.23**	**78.97%**

77.15%, which is 4.32% higher than DPNet and 20.47% higher than RPC. These results validate that our method surpasses the previous methods, especially in the medium and hard modes, which confirms the advantages of our photorealistic exemplar augmentation.

Ablation Study. We conduct ablation studies on our photorealistic exemplar augmentation, on the setting of *syn*. Each variant was trained on $10,000$ images. In the second stage, the baseline is randomly choosing and placing instances with a uniform intersection check. Then we replace the baseline with our pose selection (PS), instance placement (IP), and overlap review (OR) process sequentially. Tab. 2 shows the evaluation results in average mode, which indicates each of these three steps in the second stage can improve the performances and their combination achieves the best scores. To make it clear, Fig. 7 also visualizes the progress of these three steps step-by-step.

Figure 7. Visualization of three steps in the second stage of PEA. Compared to baseline, our pose selection (PS) method removes invalid poses (red box), our instance placement (IP) method sets products compactly to avoid scattered layout (orange box), and our overlap review (OR) process avoids severe occlusion (yellow box).

4.2.2 Incremental Learning

The incremental learning result is shown in Tab. 3. We first train the teacher model with the 183-class basic dataset and evaluate the 183-class test dataset, as shown in the "$183/183$" row. Then we evaluate the teacher model on the full test dataset directly, shown in the "183/200" row. Since the model might be confused by the new products, the overall metrics decrease, while the metrics for the old classes, $mCCD_o$ and $mCCS_o$, also receive a significant drop. After our incremental learning strategy with the new dataset, the student model learns the new classes as well as keeps the memory of old classes. As indicated in the "(183+17)/200" row, even the new classes are fine-grained (share the same meta categories with old classes), the model still produces good results and keeps the discriminative ability towards previous categories. Here we take the oracle model as the reference, the incrementally learned model has a comparable performance against the oracle.

We also generate the datasets with previous methods, RPC and DPNet, and incrementally update the model by our model training strategy. Comparing to previous meth-

Table 3. Results of the incremental learning mechanism. Our incremental learning strategy is also applied to two previous data generation methods provided by RPC and DPNet.

Method	Train/Test	cAcc ↑	ACD ↓	mCCD ↓	mCIoU ↑	$mCCD_o$ ↓	$mCCD_n$ ↓	$mCCS_o \rightarrow 1$	$mCCS_n \rightarrow 1$
RPC	183/183	58.74%	0.84	0.08	92.89%	0.08	-	98.08%	-
	183(+17)/200	53.08	0.99	0.08	92.29%	0.07	0.19	97.35%	102.88%
DPNet	183/183	73.37%	0.50	0.04	95.71%	0.04	-	99.02%	-
	183(+17)/200	67.57%	0.6	0.05	95.06%	0.04	0.13	98.76%	100.91%
Ours	183/183	78.76%	0.38	0.03	96.9%	0.03	-	99.53%	-
	183(+17)/200	**74.3%**	**0.44**	**0.04**	**96.51%**	**0.03**	**0.09**	**99.24%**	**100.36%**
	183/200	47.53%	2.06	0.18	84.65%	0.1	1.0	106.59%	0.0%
	200/200	77.15%	0.41	0.03	96.72%	0.03	0.04	99.31%	98.97%

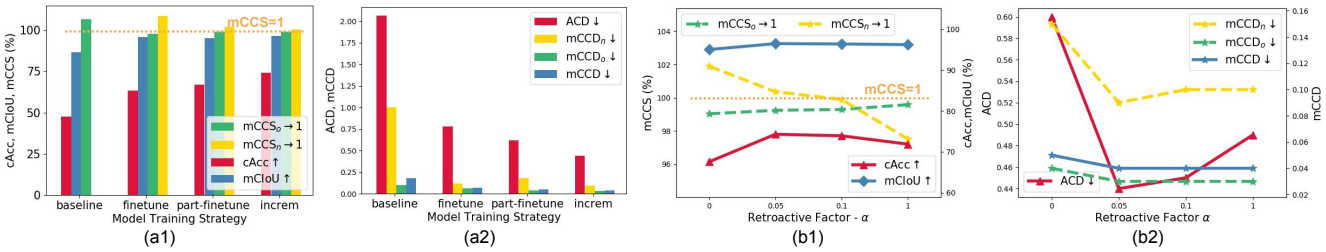

Figure 8. Comparison of (a1-a2) different model training strategies; (b1-b2) different retroactive factors.

ods, our photorealistic exemplar augmentation improves the incremental learning results, by 21.22% higher than RPC and 6.73% higher than DPNet in counting accuracy (cAcc). And our cAcc, 74.3%, after incremental learning is still higher than the cAcc of the oracle model trained using the data generated by previous methods (up to 72.83%).

Note that our incremental learning method is only incrementally trained with a small number of images (which is $13,075$), rather than training from scratch with 100k images. This saves a large number of computational budgets and meets the frequent-updating demand of the category list in real ACO scenarios.

Model Training Strategies. We compare four different model training strategies, as shown in Fig. 8(a1-a2). When new products arrive, the student model is initialized as an identical copy of the teacher except for the weights for C_n new classes in the last layer. The *baseline* is tested on this student model without training. $mCCS_n$ shows the base model has no confidence on new classes. Then we finetune the whole student model on the new dataset, denoted as *finetune*, where the model becomes over-confident on new products. To deal with this, we also experiment on *part-finetune*, which only finetunes the weights for new classes in the last layer with other parameters fixed, but the new classes seem not well learned since $mCCD_n$ is large. Finally, we try our incremental learning strategy instead of fintune, named as *increm*, in which both $mCCS_n$ and $mCCS_o$ are close to 1 and the $mCCD$ scores are in low values. It also achieves the best results on the rest metrics, showing the advantage of our incremental learning strategy.

Retroactive Factor. We also analyze on the effectiveness of retroactive factor αs 0, 0.05, 0.1 and 1. The results are shown in Fig. 8(b1-b2). Benefiting from our incremental training strategy, the model performs well on these four experiments ($cAcc > 65\%$). When $\alpha = 0$, the incremental dataset only includes new products, thus the model has high confidence scores on new products. With the raising of α, $mCCS_n$ decreases and $mCCS_o$ increases. When $\alpha = 1$, where each old and new product takes the same ratio in the new dataset, the model is under-confident on new products.

Note in this task, the model is supposed to also learn the interrelation between different categories when they show up in the same image. We find the model performs best when $\alpha = 0.05$. In this circumstance, it does not only keep good discriminative ability about the previous products but also learn the new products rationally. Moreover, the retroactive setting enables the model to learn the interactions between old and new products.

5. Conclusion

In this paper, we introduce IncreACO to solve the practical challenges in the automatic checkout system (ACO). Our developed photorealistic exemplar augmentation improves the realness of generated images by considering the perspectives, layouts, and occlusions of real checkout images. Results show PEA performs better than previous methods by 4.32% checkout accuracy in average mode. Meanwhile, our incremental learning pipeline for the first time matches the incremental updating nature of ACO, in which a small ratio of retroactivity is proved to alleviate the forgetting of old products. Powered by PEA, the model is incremented with much less time and data than the normal training process and shows great performance on both old and new classes.

Acknowledgemets

This work was supported in part by the National Key Scientific Instrument and Equipment Development Project under Grant 61827901, and the Guangxi Municipal Science and Technology Project under Grant 31062501.

References

[1] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2019.

[2] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[3] Marian George and Christian Floerkemeier. Recognizing products: A per-exemplar multi-label image classification approach. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 440–455, Cham, 2014. Springer International Publishing.

[4] Eran Goldman, Roei Herzig, Aviv Eisenschtat, Jacob Goldberger, and Tal Hassner. Precise detection in densely packed scenes. In *Proc. Conf. Comput. Vision Pattern Recognition (CVPR)*, 2019.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Xu Bing, and Yoshua Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3:2672–2680, 2014.

[6] Yu Hao, Yanwei Fu, Yu-Gang Jiang, and Qi Tian. An end-to-end architecture for class-incremental object detection with knowledge distillation. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2019.

[7] Kaiming He, Gkioxari Georgia, Dollar Piotr, and Girshick Ross. Mask r-cnn. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[9] Philipp Jund, Nichola Abdo, Andreas Eitel, and Wolfram Burgard. The freiburg groceries dataset. *ArXiv*, abs/1611.05799, 2016.

[10] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computer Science*, 2014.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[12] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[13] D. Koubaroulis, J. Matas, and J. Kittler. Evaluating colour-based object recognition algorithms using the soil-47 database. In *in Asian Conference on Computer Vision*, pages 840–845, 2002.

[14] Congcong Li, Dawei Du, Libo Zhang, Tiejian Luo, Yanjun Wu, Qi Tian, Longyin Wen, and Siwei Lyu. Data priming network for automatic check-out. In *2019 ACM Multimedia Conference on Multimedia Conference*. ACM, 2019.

[15] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[16] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[17] Jiajun Lu, Aditya Deshpande, and David A. Forsyth. Cdvae: Co-embedding deep variational auto encoder for conditional variational generation. *ArXiv*, abs/1612.00132, 2016.

[18] Jiang Jing Lv, Xiao Hu Shao, Jia Shui Huang, Xiang Dong Zhou, and Xi Zhou. Data augmentation for face recognition. *Neurocomputing*, 230(MAR.22):184–196.

[19] Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1320–1328, 2017.

[20] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[21] Anderson Rocha, Daniel C. Hauagge, Jacques Wainer, and Siome Goldenstein. Automatic fruit and vegetable classification from images. In *Science Direct -Computers and Electronics in Agriculture 70*, pages 96–104, 2010.

[22] Grégory Rogez and Cordelia Schmid. Mocap-guided data augmentation for 3d pose estimation in the wild. In *Neural Information Processing Systems (NIPS)*, 2016.

[23] J. Andrew Royle. Analysis of capture–recapture models with individual covariates using data augmentation. *Biometrics*, 65(1), 2009.

[24] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3400–3409, 2017.

[25] Xiu-Shen Wei, Quan Cui, Lei Yang, Peng Wang, and Lingqiao Liu. Rpc: A large-scale retail product checkout dataset. *ArXiv*, abs/1901.07249, 2019.

[26] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.