World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:17, No:6, 2023

Effective Stacking of Deep Neural Models for Automated Object Recognition in Retail Stores

Ankit Sinha, Soham Banerjee, Pratik Chattopadhyay

Abstract—Automated product recognition in retail stores is an important real-world application in the domain of Computer Vision and Pattern Recognition. In this paper, we consider the problem of automatically identifying the classes of the products placed on racks in retail stores from an image of the rack and information about the query/product images. We improve upon the existing approaches in terms of effectiveness and memory requirement by developing a two-stage object detection and recognition pipeline comprising of a Faster-RCNN-based object localizer that detects the object regions in the rack image and a ResNet-18-based image encoder that classifies the detected regions into the appropriate classes. Each of the models is fine-tuned using appropriate data sets for better prediction and data augmentation is performed on each query image to prepare an extensive gallery set for fine-tuning the ResNet-18-based product recognition model. This encoder is trained using a triplet loss function following the strategy of online-hard-negative-mining for improved prediction. The proposed models are lightweight and can be connected in an end-to-end manner during deployment to automatically identify each product object placed in a rack image. Extensive experiments using Grozi-32k and GP-180 data sets verify the effectiveness of the proposed model.

Keywords—Retail stores, Faster-RCNN, object localization, ResNet-18, triplet loss, data augmentation, product recognition.

I. Introduction

IN this paper, we propose a Deep Learning-based solution to the problem of identifying products on racks in retail stores from an image of the rack and a database of query/product images that may be placed on the rack. The problem can be explained using Figs. 1 (a)-(d). With reference to the figure, there exists a set of query/product images (Fig. 1 (b)) and multiple instances of each of these may be placed at different positions on the rack. Our Deep Learning model will take as input the original rack image (Fig. 1 (a)) and output the same rack image with detected and identified product instances. Fig. 1 (c) shows the identified query objects with color codes specified in Fig. 1 (d). In real-life situations, there will be several such query images and large rack images due to which there is a need for the development of automated product recognition algorithms in retail stores to assist customers to find the right product.

There are several challenges associated with the problem as given in [1, 2]. Usually, only a single query/reference image per product class is available which is insufficient to train a Deep Neural Network model. Also, the appearance of the reference image may differ significantly from the rack images in terms of orientation, illumination, resolution, reflection, etc. Moreover, variants of the same product with minor visual cues are likely to pose difficulty in the correct identification of the actual product class.

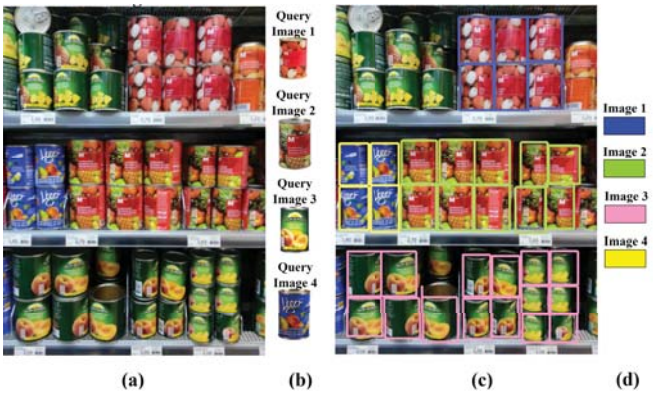

Fig. 1 (a) Sample Rack Image, (b) Query Images, (c) Rank Image with Detected Product Classes, (d) Color Codes for the Query images

Research work to date have tried to address this issue and reduce human intervention, and our work is an improvement to that proposed by Tonioni et al. in [3] that also follows a two-stage pipeline involving Deep Neural Networks for object localization and recognition. Specifically, we attempt to improve the effectiveness of the YOLO-based product localization model used in [3] by employing a Faster-RCNN with a Feature Pyramid Network to capture multi-scale features for improved localization. For object recognition also, we use a lightweight ResNet-18-based product recognition model instead of the larger VGG-16 model as considered in [3]. The underlying architecture of our proposed model is lightweight and it is expected to perform more accurately due to the use of multi-scale features for product localization as compared to that of [3], which makes it suitable for deployment on edge devices with less amount of memory. The main contributions of this paper can be summarized as follows:

- In this article, a two-stage Deep Learning-based pipeline for product recognition in retail settings is proposed which is lightweight and can be conveniently integrated with edge devices.
- We avoid template matching-based object detection as used by most existing techniques and employ multi-scale Deep Convolutional Networks for the same

Ankit Sinha and Pratik Chattopadhyay are with the Department of Computer Science and Engineering, Indian Institute of Technology (Banaras Hindu University), Varanasi, India, PIN 221005 (e-mail: ankitsinha.cse18@iitbhu.ac.in, pratik.cse@iitbhu.ac.in).

Soham Banerjee is with the Department of Computer Science and Engineering, Budge Budge Institute of Technology, Kolkata, India, PIN 700137 (e-mail: banerjeesoham524@gmail.com).

task which makes the prediction accurate with a fast response time. This scheme is also capable of effectively identifying blank regions on the rack which the template matching-based approaches fail to handle properly.
- We improve upon the work in [3] in terms of employing a more effective network that extracts multi-scale features for product localization and also using a much lighter ResNet-18 model for product recognition.
- We conduct extensive experiments and perform a comparative study with state-of-the-art approaches to evaluate the effectiveness of the proposed approach.

II. Related Work

Santra et al. have been working on several challenges in this domain [4–6]. In [4], an end-to-end annotation-free mechanism for product detection on racks is proposed. It is a multi-stage exemplar-driven approach in which the relative scale of the rack images with respect to the available product templates is estimated in the first stage. In the second stage, potential object regions are determined and refined using greedy Non-Max-Suppression (NMS). Finally, a Convolutional Neural Network (CNN) is used to perform the classification using the extracted regions to identify the product classes. In [5], the use of greedy NMS in the task of detecting object locations in rack images is analyzed in depth. In this work, the authors argue that the greedy NMS discards bounding boxes with superior geometric placement due to the overlapping of other boxes with higher confidence scores and propose a graph-based NMS to compute the potential confidence scores. In another work [6], fine-grained classification of product instances is done by extracting unique local patches around key points within an image and encoding these using Convolutional Long-Short-Term Memory (LSTM) network.

George et al. [7] proposed a per-exemplar multi-label image classification and localization approach by establishing a locality constraint linear coding [8] model using dense SIFT features of the product images. Further, a discriminative Random Forest is trained followed by a multi-class ranking of products to carry out the recognition task. Wang et al. [9] proposed a destruction followed by a construction method guided by a self-attention mechanism for end-to-end fine-grained classification tasks. Osokin et al. [10] employed a one-stage hybrid model to achieve the tasks of localization and recognition jointly. First, the local features are extracted from both the input and class images using a ResNet following which dense correlation scores between the two are computed. Next, the feature maps are semantically aligned through a trained geometric transformation model to predict the bounding boxes before finally computing the recognition scores.

In the task of planogram compliance, Saran et al. [11] presented a visual analysis framework and applied the Hausdorff metric to compute the occupancy of product shelves. Here, the authors describe a robust product counting algorithm using row detection methods with texture and color-based features. In another work, Ray et al. [12] proposed a two-layer hypothesis and verification model in which the model first predicts a set of candidate items at a certain position of the rack, and next the above hypothesis is verified by a graph-based algorithm. The proposal about candidate items is made through a combination of correlation-based and ad-hoc SURF [13] schemes. Liu et al. [14] proposed an unsupervised recurrent pattern mining strategy with a graph-based matching algorithm for planogram compliance by adopting a divide-and-conquer policy independent of product templates. In [15], the authors proposed a two-stage approach to recognize products in rack images. First, SIFT features and Hough transformation are used to find probable matches of reference product images on the shelf. Next, to determine the missing products and remove incorrect matches, the sub-graph isomorphism between the observed output and the actual output is performed.

Goldman and Goldberger [16] proposed a Deep Learning-based method to classify well-structured objects with high inter-class similarity by treating sequences of images as linear Conditional Random Fields (CRFs) to include contextual information. Baz et al. [17] proposed a hybrid classifier combining Support Vector Machines (SVM) with probabilistic graphical models like Hidden Markov Models (HMM) and CRFs by exploiting the spatial continuity in the arrangement of products on a rank. Since products of similar brands are placed adjacent to each other, the authors model the contextual information using an *HMM* or *CRF*. The classification performance of SVM has been seen to improve with the inclusion of this contextual information.

A method for fine-grained classification of products by detecting recurring features in rack images is presented in [18]. These recurring features are compared with SIFT features from the logo regions of the reference images and assigned a rough class label. Further, fine-grained classification has been performed by training a VGG-16 [19] with an attention map policy generated using matching SURF [13] and BRISK [20] features from the product instance and the template image of the rough class label. However, such a template matching-based approach is likely to be time-intensive due to the exhaustive search required across the entire rack image and is not suitable for most practical purposes. The work of Tonioni et al. [3] is one of the few approaches that consider Deep Learning-based prediction for both object localization and prediction. Here, the authors make use of a two-stage pipeline based on YOLO-v2 [21] and VGG-16 [19], in which the YOLO-v2 object detector is fine-tuned using a privately annotated dataset to predict the bounding boxes of the product instances. Further, the VGG-16 embedder is trained with triplet loss [22] and MAC [23] features to identify the product class.

From the extensive literature survey, it has been found that classical image processing and template matching-based methods dominate Deep Learning-based techniques in the majority of the retail use cases. The scarcity of sufficient ground truth for product images is one of the primary causes of this. Template matching-based techniques suffer from high response time and are also susceptible to noise. On the other hand, Neural Network-based approaches are known for their robustness against noise and variations of input conditions, and the few existing approaches in this category use heavyweight

models that are not suitable for implementation on edge devices. In this work, we employ effective but lightweight Deep Neural Network architectures for both object localization and product detection steps and perform a rigorous analysis and comparative study with state-of-the-art approaches. Our approach is described in detail in Section III.

III. Proposed Approach

We implement a two-stage pipeline for object localization and recognition tasks. In the first stage, a faster RCNN-based Deep Neural Network [24] is employed to output the bounding box coordinates of the region proposals through a regressor head. In the second stage, another CNN-based Deep Neural Network model is used to convert these region proposals into feature descriptors in a latent space. Unlike previous models like YOLO [21, 25, 26] and SSD [27] that combine the region proposal and the recognition stages in a single model, our proposed two-stage architecture considers two separate dedicated lightweight models for the two above-mentioned tasks (as explained in the following two sub-sections), which is expected to improve the effectiveness of the overall approach further.

A. Localizing Product Instances

As mentioned before, we use the faster-RCNN architecture with ResNet-50 [28] backbone to extract features for localizing product instances. This feature extractor is followed by a Feature Pyramid Network [29] that aggregates useful multi-scale features in a top-down direction. Next, there is a region proposal network (RPN) that uses anchors of fixed sizes and aspect ratios to output high-quality region proposals. After Region-of-Interest (RoI) pooling and Non-Maximal-Suppression (NMS), there is a bounding box *regressor* head. The *classifier* head of standard RCNN architecture is discarded since our embedder serves the same objective. Then, we extract patches cropped from the input image according to the predicted bounding boxes.

The complete architecture of the localization network is shown in Fig. 2. The network consists of five convolutional

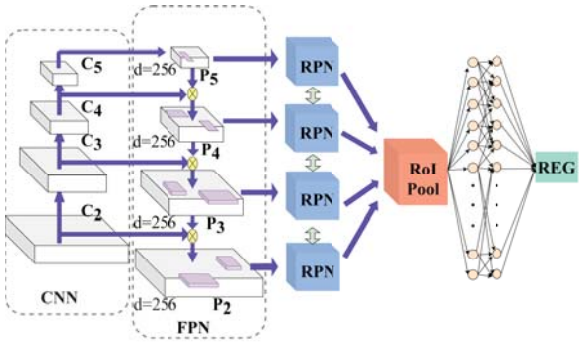

Fig. 2 The Localization Pipeline

blocks in which each block is a sub-network of multiple layers that produces feature maps of the same size. The outputs from the last layers of the second to fifth blocks are extracted and a reference set $\{C_2, C_3, C_4, C_5\}$ is formed. The feature map corresponding to the first block C_1 has not been included for further operations due to its large memory footprint. These feature maps are next used to construct a feature pyramid. The process of constructing the pyramid starts by passing C_5 through a $1 \times 1 \times d$ convolution layer to reduce the number of channels to d and obtain the first pyramid feature P_5. Except for P_5, the rest of the feature pyramids are constructed by following a similar method. So, P_5 is upsampled to twice its size using nearest-neighbor interpolation. Let us call the output from this stage $\tilde{P}_5$. Next, C_4 is convolved using a $1 \times 1 \times d$ dimensional filter to obtain $\tilde{C}_4$ which has dimensions the same as that of $\tilde{P}_5$. The feature maps $\tilde{C}_4$ and $\tilde{P}_5$ are henceforth added element-wise and this merged feature map is passed through another 3×3 dense convolution operation to compensate for the aliasing effect of upsampling and form the pyramid layer P_4. The depth of all the pyramid layers is set to $d = 256$. A similar procedure is also followed to obtain the feature pyramids P_3 and P_2. The construction of the feature pyramid P_{k-1} from P_k is mathematically expressed as follows:

$$\tilde{P}_k = \text{Upsample}_{\times 2}(P_k) \quad (1)$$

$$\tilde{C}_{k-1} = \text{Conv}_{1\times1}(C_{k-1}) \quad (2)$$

$$P_{k-1} = \text{Conv}_{3\times3}(\tilde{P}_k + \tilde{C}_{k-1}) \quad (3)$$

The resulting feature pyramids carry a good balance of semantic and fine-grained information. So anchors of a specific scale are assigned to each pyramid level instead of repetitively assigning all scales of anchors to every level. This avoids computational redundancy and ensures the simplicity of design. The pyramid levels P_2, P_3, P_4, P_5 are assigned anchors of scales 32, 64, 128, and 256, respectively. Anchors of each scale have three aspect ratios: 1:1, 1:2, 2:1.

The FPN is followed by a region proposal network (RPN) [24] which is a mini-convolutional network of 256 channels having 3×3 kernels. It further separates into two branches: *object classifier* and *object regressor*. The *object classifier* branch performs *object* vs *background* binary classification whereas the *object regressor* predicts the bounding boxes for anchors with high objectness scores. Each of the feature pyramids, P_5, P_4, P_3, and P_2, is processed using an RPN in a sliding window manner. As shown in Fig. 2, weight-sharing is done among all the RPNs, and the outputs from the different RPNs are passed through a common Region-of-Interest (RoI) pooling layer. The RoIs are next passed through two fully-connected layers and the final bounding box *regressor* head to localize the objects in the rack image.

B. Recognition of Products

The recognition pipeline shown in Fig. 3 predicts the class of each object detected within the rack image in the previous stage. A ResNet-18 embedder is used for this task which is trained offline by sampling triplets of different query images consisting of an anchor i_a, a positive i_p, and a negative i_n image. Let the embedding of these three images be denoted by x_i^a, x_i^p, and x_i^n respectively. The anchor and the positive images belong to the same category while the negative image belongs to a different class. Using the common Euclidean

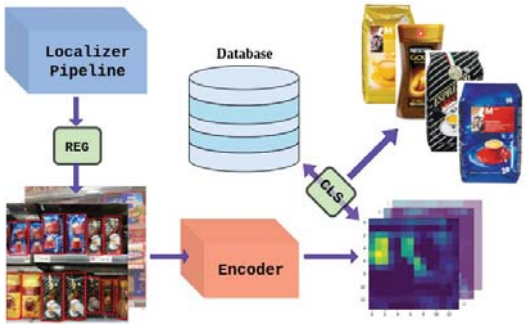

Fig. 3 The Recognition Pipeline

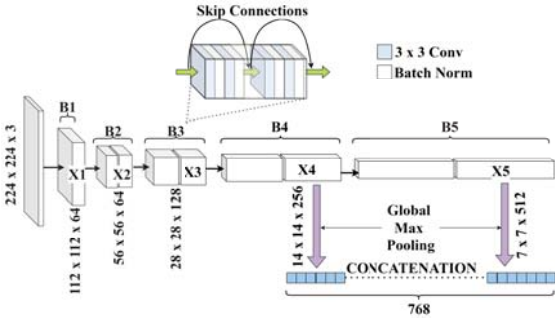

Fig. 4 Block diagram of ResNet-18-based Encoder

distance function d the network is trained to minimize the triplet loss $\mathcal{L}$ defined as

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} max(0, d(x_i^a, x_i^p) - d(x_i^a, x_i^n) + \alpha). \quad (4)$$

The objective here is to learn an embedding such that the following inequality is satisfied:

$$\|x_i^a - x_i^p\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2, \quad (5)$$

where α denotes the fixed minimum margin between the distance pairs. If a single image for every product class is present in the query image database, then we use data augmentation to generate multiple anchor images corresponding to each query image. For this, we apply standard image augmentation methods such as Gaussian blur, random crop, brightness, and saturation variations. A negative image of the triplet can be chosen in various ways. Theoretically, a huge number of triplets ($\approx N^2$) can be generated even from a small dataset, where N is the size of the dataset. Hence, to ensure faster convergence with good sample efficiency we apply the strategy of Online Hard Negative Mining (OHNM). In this approach, for every mini-batch of size b loaded during training, the *hardest* negative sample against each positive sample is chosen from the same batch. This selection scheme reduces the total computational complexity to $O(bN)$ from $O(N^2)$. Mathematically,

$$X_{bat} \subset |D|, \quad (6)$$
$$x_i^a = \tilde{f}(x_i^p), \forall x_i^p \in X_{bat}, \quad (7)$$
$$\tilde{x}_i^n = argmin_{x_j} \|x_i^a - x_j\|_2^2,\ \tilde{x}_i^n \in X_{bat}. \quad (8)$$

$\tilde{f}$ denotes the image augmentation operator, x_i^p denotes a positive sample, x_i^n denotes a negative sample, x_i^a denotes an anchor generated from x_i^p using data augmentation, and X_{bat} is a mini-batch of data. Through the minimization of this loss function, the network learns to encode images of the same class close to each other in the encoded space while separating those belonging to different classes.

The Embedder, as shown in Fig. 4, is a ResNet-18 [28] pre-trained on the ImageNet1K [30] dataset. The ResNet-18 contains five convolutional blocks B1, B2, B3, B4, B5 that output feature maps of sizes 112, 56, 28, 14, and 7, respectively. The corresponding feature maps have been named *X1*, *X2*, *X3*, *X4*, and *X5* in Fig. 4. In our work, the final descriptors are extracted by applying MAC [23] operation on blocks B4 and B5, yielding $\tilde{X}_4$ and $\tilde{X}_5$, respectively, shown as:

$$X_4 = Conv4(X_3), \quad size = (14, 14, 256)$$
$$X_5 = Conv5(X_4), \quad size = (7, 7, 512)$$
$$\tilde{X}_4 = GlobalMaxPool(X_4), \quad size = (1, 256)$$
$$\tilde{X}_5 = GlobalMaxPool(X_5), \quad size = (1, 512)$$
$$X_{embed} = concat(\tilde{X}_4, \tilde{X}_5), \quad size = (1, 768)$$

Next, the vectors are concatenated into a single descriptor and $\ell 2$ normalized.

IV. Experiments

All experiments are conducted in Google Colab notebooks that provide free GPU and TPU support. PyTorch has been chosen as the deep learning framework. The colab virtual machines (VMs), by default, offer a RAM of 13 GB. The fine-tuning of Faster RCNN for object detection is done on an Nvidia Tesla T4 GPU with 16 GB memory. The embedder ResNet-18 is fine-tuned on a Nvidia Tesla K80 GPU with 12 GB capacity.

The Faster-RCNN FPN, pre-trained on COCO 2017 [31], is fine-tuned by Stochastic Gradient Descent (SGD) [32] with momentum 0.8 and weight decay 5×10^{-2} using a manually annotated small subset of 60 rack images from the Grozi-3.2K Food data. A learning rate schedule with warmup has been used for fine-tuning, where the number of warmup iterations is set to the total number of mini-batches $N_{batch} = \frac{\|D\|}{b}$. Here, D denotes the dataset for fine-tuning, and b (= 8) is the batch size. The warmup factor is set to 10^{-3} and the maximum learning rate is set to 5×10^{-2}. Having fine-tuned the detector for 25 epochs with the above settings, the learning rate is reduced to 5×10^{-3} for additional 15 epochs. During fine-tuning, we use the top 1000 region proposals.

The Embedder ResNet-18 is the pre-trained checkpoint on ImageNet1K [30] dataset. It is fine-tuned using the standard triplet loss with a margin of 1.0 using each product present in the query image database. In this work, we use the Grozi-3.2k data [7] that contains a total of 3235 query images, one for each product item. The fine-tuning is continued for 15 epochs with a fixed learning rate of 10^{-4} using Adaptive Gradient Descent (ADAM) [33] optimizer. For evaluation, we use 680 rack images from five different retail stores present in the

Grozi-3.2k data [7]. However, the ground truth annotations for this data set consist of identical products, that are adjacently placed, and grouped under a single bounding box. Hence, this data set is ideal for multi-label image classification tasks. We also use the GP-180 data [15] for evaluation which is a subset of the Grozi-3.2k data and contains instance-level annotations of 74 rack images.

In our first experiment, we evaluate the effectiveness of the product detection model (localizer) based on Faster-RCNN (refer to Section III-A). Fig. 5 shows two rack images from GP-180 along with the predicted bounding boxes. It can be visually observed from the results that our detector accurately localizes the different items for each of the test images and it is also able to correctly identify the blank spaces (i.e., the regions on the rack with no objects).

(a) (b)

Fig. 5 Sample rack images from GP-180 and the bounding boxes predicted by our Faster-RCNN-based product detector marked in red

TABLE I
PERFORMANCE OF OUR LOCALIZATION PIPELINE ON GP-180 IN THE STANDARD COCO FORMAT

Metric	maxDets	Result
AP @[IoU = 0.50]	100	0.864
AP @[IoU = 0.75]	100	0.726
AP @[IoU = 0.50:0.95]	100	0.594
AR @[IoU = 0.50:0.95]	1	0.059
AR @[IoU = 0.50:0.95]	10	0.539
AR @[IoU = 0.50:0.95]	100	0.672

Here, maxDets represents the number of region proposals used during testing.

We also study the effectiveness of our Faster RCNN-based object localization model using standard COCO metrics in Table I. The GP-180 data have been used for this experiment. In the table, *AP* denotes Average Precision, *AR* denotes Average Recall, and *maxDets* signifies the number of top region proposals considered during the test time. It is observed that our detection model is capable of precisely localizing grocery products. For IoU in the range 0.50 to 0.95 (incremented in step-size of 0.05), the AP is $\sim 60\%$. As far as the AR is concerned, it increases with the number of region proposals, as expected. However, the larger the number of region proposals the longer will be the inference time and the lesser will be the AP. In a real-time use case (using only 10 region proposals), our model accurately localizes 54% of the items on the rack, whereas, on 100 region proposals, AR satisfactorily increases to 67.2%.

Each of the following experiments deals with the evaluation of our overall product detection and recognition pipeline and a comparative study with other state-of-the-art techniques. First, we compare our work with [18] and [3] using the GP-180 [15] data. While the work in [3] describes an end-to-end Deep Learning approach, that in [18] is a hybrid method with a template matching-based bounding box detector and a class-specific CNN recognizer. For this experiment, we follow the same protocol as described in [15], i.e., we consider a prediction to be correct if the output label matches the ground truth label of the product, provided the IoU between the detected and ground-truth boxes is greater than 0.5. In Table II, we report the mean average precision mAP@0.5 and product recall PR@0.5 of our method, [18], and [3]. It is observed that

TABLE II
COMPARATIVE RESULTS OF OUR FULL PIPELINE ON GP-180 ACCORDING TO THE PROTOCOLS ADOPTED IN [3] AND [18]

Method	mAP@0.5	PR@0.5
yolo_ld+lf-mc-th [3]	76.93	86.56
SIFT + vgg16 [18]	**85.79**	-
frcnn + res50 + res18 (ours)	82.70	**89.70**

the proposed method improves over [3] by $\sim 5\%$ and $\sim 3\%$ in terms of mAP@0.5 and PR@0.5, respectively. Although [18] shows $\sim 3\%$ higher mAP@0.5 compared to that of ours, it involves the use of hand-crafted feature engineering for product localization which is significantly time-intensive and is expected to be less robust to the variation of input conditions.

We also compare our approach with two recent approaches, namely [5] and [4], on the GP-180 data set using a similar test protocol as specified in [4, 5]. Both of these compared methods are hybrid approaches that use exemplar-driven localization and CNN-based object recognition framework. The testing protocol can be described as follows: Let a product *P* be present on the rack. If the center of any detected bounding box lies within *P* in the rack and the enclosed item is predicted as *P*, the count of *true-positives* (TP) is incremented by 1. If the center of the detected bounding box lies within *P* in the rack but the enclosed item is not recognized as *P* by the object recognition model, then the count of *false-positives* (FP) of the rack is incremented by 1. Further, if the center of a detected box does not lie within any true product in the rack, the count FP is again increased by 1. Lastly, if there exists no detected box whose center lies within *P*, the count of *false-negatives* (FN) of the rack is incremented by 1. Considering the above, we compare our approach with that of [4] and [5] and present the F_1 scores after the final recognition phase in Table III. It can be seen from the table that our approach outperforms both the other two compared methods in terms of F_1 score by at least 2%.

In the next experiment, we compare our approach with state-of-the-art methods [3, 7, 34] using the Grozi-3.2k Food data set. Here, [34] is a non-ML approach based on the Hough transform, whereas [7] involves a fusion of several strategies

TABLE III
COMPARATIVE RESULTS OF OUR FULL PIPELINE ON GP-180 DATASET ACCORDING TO THE EVALUATION PROTOCOL FOLLOWED IN [4, 5]

Method	F_1 score (%)
ERP-CNN [4]	81.05
R-CNN-G [5]	80.21
Ours	**83.20**

including Fast dense pixel matching, Random Forests, and Genetic Algorithms. On the other hand, [3] follows a fully Deep Learning pipeline. The evaluation protocol followed here is similar to that described in [7]. Specifically, we use two metrics, namely, mean average precision (mAP) and mean average product recall (mAPR). Corresponding to each bounding box detected by the localization model, we fetch the top K predictions (candidate product items) by the object recognition model. If the ground truth label is present within these top K predictions, the *true-positive* count is increased by 1. Otherwise, the *false-positive* count is incremented by 1. Next, the precision and product recall are computed and averaged across all the test images to obtain the average precision (AP) and average product recall (APR) scores. The AP and APR values computed for different values of K are averaged to report the final mAP and mAPR metrics. The corresponding results are shown in Table IV for the compared methods for two values of K, i.e., 20 and 50. It can be seen from the results that our method

TABLE IV
COMPARATIVE RESULTS ON THE GROZI-3.2K FOOD DATASET USING THE PROTOCOL AS DESCRIBED IN [7]

Method	mAP (%)	mAPR (%)
FM+HO [34]	23.71	41.60
yolo_ld+lf-mc-th [3]	36.02	**58.41**
RF+PM+GA [7]	23.49	43.13
Ours	**47.77**	58.11

outperforms each of [7, 34] both in terms of mAP and mAPR. It also surpasses the mAP score provided by [3] by a large margin of 11%, but in terms of mAPR we fall short of [3] by only $\sim 0.30\%$. However, as also mentioned in Section I, the method in [3] uses VGG-16 for object recognition which has a larger memory footprint compared to our ResNet-18-based product recognition model. While the YOLO-based object localization model has 62M parameters and the VGG-16-based product recognition model has 138M parameters, our Faster RCNN-FPN-based localization model has only 42M parameters and the ResNet-18-based recognition model has only 11M parameters. Hence, in terms of mAP, mAPR, and memory space requirement, our approach can be regarded as the best among the other competing methods used in this study.

V. CONCLUSIONS

In this work, we present a two-stage Deep Learning-based pipeline to automatically detect product locations and identify the products placed on the racks in retail stores. The first stage involves the generation and refinement of region proposals using a Faster-RCNN-FPN model. In the second stage, these region proposals are passed through a ResNet-18-based embedder followed by a classification layer to predict the product class. During deployment, these two models are connected in an end-to-end manner to automatically identify which products are present on the shelves of a rack using only an image of the complete rack. We have made a thorough evaluation of both the localization and the recognition frameworks through extensive experiments and a comparative study to verify the effectiveness of our approach. Due to the use of lightweight neural network-based models for both of the localization and recognition phases, our approach is time-efficient and requires only a small amount of memory to run making it suitable for deployment on edge devices. Only a few existing approaches used in the comparative study, namely [3, 18], show performance comparable to that of our model. However, these are either time-intensive due to the use of hand-crafted features for object detection or employ heavy-weight neural network models to perform the prediction, and are hence not suitable for large-scale applications. In contrast, our lightweight model can be conveniently used to keep track of inventories in large retail stores in a time-efficient manner. In the future, network distillation may be used to reduce the complexity of our models further.

ACKNOWLEDGMENT

The authors would like to thank IIT(BHU), Varanasi for providing the necessary resources including servers, technicians, etc., to initiate research work in this area.

REFERENCES

[1] Yuchen Wei, Son N. Tran, Shuxiang Xu, Byeong Ho Kang, and Matthew Springer. Deep learning for retail product recognition: Challenges and techniques. *Computational Intelligence and Neuroscience*, 2020, Article ID: 8875910, 2020.

[2] Bikash Santra and Dipti Prasad Mukherjee. A comprehensive survey on computer vision based approaches for automatic identification of products in retail store. *Image and Vision Computing*, 86:45–63, 2019.

[3] Alessio Tonioni, Eugenio Serra, and Luigi di Stefano. A deep learning pipeline for product recognition on store shelves. In *Proceedings of the International Conference on Image Processing, Applications and Systems*, pages 25–31, 2018.

[4] Bikash Santra, Avishek Shaw, and Dipti Prasad Mukherjee. An end-to-end annotation-free machine vision system for detection of products on the rack. *Machine Vision and Applications*, 32(3):1–13, 2021.

[5] Bikash Santra, Avishek Shaw, and Dipti Prasad Mukherjee. Graph-based non-maximal suppression for detecting products on the rack. *Pattern Recognition Letters*, 140:73–80, 2020.

[6] Bikash Santra, Avishek Shaw, and Dipti Prasad Mukherjee. Part-based annotation-free fine-grained classification of images of retail products. *Pattern Recognition*, 121:108257, 2022.

[7] Marian George and Christian Floerkemeier. Recognizing products: A per-exemplar multi-label image classification approach. In *Proceedings of the European Conference on Computer Vision*, pages 440–455, 2014.

[8] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas S. Huang, and Yihong Gong. Locality-constrained linear coding for image classification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3360–3367, 2010.

[9] Wenyon Wang, Yongcheng Cui, Guangshun Li, Chuntao Jiang, and Song Deng. A self-attention-based destruction and construction learning fine-grained image classification method for retail product recognition. *Neural Computing and Applications*, 32(18):1–10, 2020.

[10] Anton Osokin, Denis Sumin, and Vasily Lomakin. Os2d: One-stage one-shot object detection by matching anchor features. In *Proceedings of the European Conference on Computer Vision*, pages 635–652, 2020.

[11] Anurag Saran, Ehtesham Hassan, and Avinash Kumar Maurya. Robust visual analysis for planogram compliance problem. In *Proceedings of the IAPR International Conference on Machine Vision Applications*, pages 576–579. IEEE, 2015.

[12] Archan Ray, Nishant Kumar, Avishek Shaw, and Dipti Prasad Mukherjee. U-pc: Unsupervised planogram compliance. In *Proceedings of the European Conference on Computer Vision*, pages 586–600, 2018.

[13] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision*, 2006.

[14] Song Liu, W. Li, Stephen J. Davis, Christian Ritz, and Hongda Tian. Planogram compliance checking based on detection of recurring patterns. *IEEE MultiMedia*, 23(2):54–63, 2016.

[15] Alessio Tonioni and Luigi di Stefano. Product recognition in store shelves as a sub-graph isomorphism problem. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 682–693, 2017.

[16] Eran Goldman and Jacob Goldberger. Large-scale classification of structured objects using a crf with deep class embedding. *arXiv preprint arXiv:1705.07420*, 2017.

[17] Ipek Baz, Erdem Yörük, and Müjdat Çetin. Context-aware hybrid classification system for fine-grained retail product recognition. *Proceedings of the Image, Video, and Multidimensional Signal Processing Workshop*, pages 1–5, 2016.

[18] Wei dong Geng, Feilin Han, Jiangke Lin, Liuyi Zhu, Jieming Bai, Suzhen Wang, Lin He, Qiang Xiao, and Zhangjiong Lai. Fine-grained grocery product recognition by one-shot learning. *Proceedings of the ACM International Conference on Multimedia*, pages 1706–1714, 2018.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, abs/1409.1556, 2014.

[20] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. *Proceedings of the International Conference on Computer Vision*, pages 2548–2555, 2011.

[21] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[23] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. *arXiv preprint arXiv:1511.05879*, 2015.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[25] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[26] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[27] W. Liu, Dragomir Anguelov, D. Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37, 2016.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[29] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[31] 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 *Proceedings of the European Conference on Computer Vision*, 2014.

[32] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the International Conference on Computational Statistics*, pages 177–186, 2010.

[33] Diederik P. Kingma and Jimmy Ba. Adam: A method for

stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[34] Erdem Yörük, Kaan Taha Oner, and Ceyhun Burak Akgül. An efficient hough transform for multi-instance object recognition and pose estimation. *Proceedings of the International Conference on Pattern Recognition*, pages 1352–1357, 2016.

Ankit Sinha is currently a Software Developer (IC-2) with Oracle Cloud Infrastructure. He received his B.Tech degree in Computer Science and Engineering from the Indian Institute of Technology (BHU), Varanasi. His research interests include Computer Vision, Machine Learning, and Reinforcement Learning.

Soham Banerjee received his B.Tech. degree in Computer Science and Engineering from the Budge Budge Institute of Technology, Kolkata. His areas of interest include Software Development, Machine/ Deep Learning Applications.

Dr. Pratik Chattopadhyay is currently an Assistant Professor in the Department of Computer Science and Engineering, IIT (BHU) Varanasi, India. He received his B.Tech. in Computer Science and Engineering from the Institute of Engineering and Management, Kolkata, India, in 2009, his M.E. in Computer Science and Engineering from Jadavpur University, Kolkata, in 2011, and his Ph.D. from IIT Kharagpur, India, in 2015. Post Ph.D., he worked as a Visiting Scientist at the Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata for six months and at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore for one year before joining IIT(BHU), Varanasi in 2017. His current research interests include Machine/Deep Learning, Image Processing, and Data Mining.