A deep learning pipeline for product recognition on store shelves

Alessio Tonioni
DISI, University of Bologna
alessio.tonioni@unibo.it

Eugenio Serra
DISI, University of Bologna
eugenio.serra@studio.unibo.it

Luigi Di Stefano
DISI, University of Bologna
luigi.distefano@unibo.it

Abstract

Recognition of grocery products in store shelves poses peculiar challenges. Firstly, the task mandates the recognition of an extremely high number of different items, in the order of several thousands for medium-small shops, with many of them featuring small inter and intra class variability. Then, available product databases usually include just one or a few studio-quality images per product (referred to herein as ***reference*** *images), whilst at test time recognition is performed on pictures displaying a portion of a shelf containing several products and taken in the store by cheap cameras (referred to as* ***query*** *images). Moreover, as the items on sale in a store as well as their appearance change frequently over time, a practical recognition system should handle seamlessly new products/packages. Inspired by recent advances in object detection and image retrieval, we propose to leverage on state of the art object detectors based on deep learning to obtain an initial product-agnostic item detection. Then, we pursue product recognition through a similarity search between global descriptors computed on reference and cropped query images. To maximize performance, we learn an ad-hoc global descriptor by a CNN trained on reference images based on an image embedding loss. Our system is computationally expensive at training time but can perform recognition rapidly and accurately at test time.*

1. Introduction

Recognizing products displayed on store shelves based on computer vision is gathering ever-increasing attention thanks to the potential for improving the customer's shopping experience (*e.g.*, via augmented reality apps, checkout-free stores, support to the visually impaired . . .) and realizing automatic store management (*e.g.*, automated inventory, on-line shelf monitoring. . .).

The seminal work on product recognition dates back to [12], where Merler *et al*. highlight the peculiar issues to be addressed in order to achieve a viable approach. First of all, the number of different items to be recognized is huge, in the order of several thousands for small to medium shops, well beyond the usual target for current state-of-the-art image classifiers. Moreover, product recognition can be better described as a hard instance recognition problem, rather than a classification one, as it deals with lots of objects looking remarkably similar but for small details (*e.g.*, different flavors of the same brand of cereals). Then, any practical methodology should rely only on the information available within existing commercial product databases, *i.e.* at most just one high-quality image for each side of the package, either acquired in studio settings or rendered (see Figure 1-(b)). *Query* images for product recognition are, instead, taken in the store with cheap equipment (*e.g.*, a smart-phone) and featuring many different items displayed on a shelf (see Figure 1-(a)). Unfortunately, this scenario is

Figure 1: Given a *query* image featuring multiple products (a), our system first detects the regions associated with the individual items and then recognizes the product enclosed in each region based on a database featuring only one *reference* image per product (two examples are shown in (b)). All the products are correctly recognized in (a) with bounding boxes colored according to the recognized classes.

far from optimal for state-of-the-art multi-class object detectors based on deep learning [15, 9, 16], which require a large corpus of annotated images as similar as possible to the deployment scenario in order to provide good performance. Even acquiring and manually annotating with product labels a huge dataset of in-store images is not a viable solution due to the products on sale in stores, as well as their appearance, changing frequently over time, which would mandate continuous gathering of annotated in-store images and retraining of the system. Conversely, a practical approach should be trained once and then be able to handle seamlessly new stores, new products and/or new packages of existing products (*e.g.*, seasonal packages).

To tackle the above issues, we address product recognition by a pipeline consisting of three stages. Given a shelf image, we perform first a class-agnostic object detection to extract region proposals enclosing the individual product items. This stage relies on a deep learning object detector trained to localize product items within images taken in the store; we will refer to this network as to the *Detector*. In the second stage, we perform product recognition separately on each of the region proposal provided by the *Detector*. Purposely, we carry out K-NN (K-Nearest Neighbours) similarity search between a global descriptor computed on the extracted region proposal and a database of similar descriptors computed on the *reference* images available in the product database. Rather than deploying a general-purpose global descriptor (*e.g.*, Fisher Vectors [13]), we train a CNN using the *reference* images to learn an image embedding function that maps RGB inputs to n-dimensional global descriptors amenable for product recognition; this second network will be referred to as to the *Embedder*. Eventually, to help prune out false detections and improve disambiguation between similarly looking products, in the third stage of our pipeline we refine the recognition output by re-ranking the first K proposals delivered by the similarity search. An exemplary output provided by the system is depicted in Figure 1-(a)).

It is worth pointing out how our approach needs samples of annotated in-store images only to train the product-agnostic *Detector*, which, however, does not require product-specific labels but just bounding boxes drawn around items. In section 4 we will show how the product-agnostic *Detector* can be trained once and for all so to achieve remarkable performance across different stores despite changes in shelves disposition and product appearance. Therefore, new items/packages are handled seamlessly by our system simply by adding their global descriptors (computed through the *Embedder*) in the *reference* database. Besides, our system scales easily to the recognition of thousands of different items, as we use just one (or few) *reference* images per product, each encoded into a global descriptor in the order of one thousand float numbers.

Finally, while computationally expensive at training time, our system turns out light (*i.e.* memory efficient) and fast at deployment time, thereby enabling near real-time operation. Speed and memory efficiency do not come at a price in performance, as our system compares favorably with respect to previous work on the standard benchmark dataset for product recognition.

2. Related Work

Grocery products recognition was firstly investigated in the already mentioned paper by Merler *et al*. [12]. Together with a thoughtful analysis of the problem, the authors propose a system based on local invariant features. However, their experiments report performance far from conducive to real-world deployment in terms of accuracy and speed. A number of more recent works tried to improve product recognition by leveraging on: a) stronger features followed by classification [4], b) the statistical correlation between nearby products on shelves [1, 2] c) additional information on the expected product disposition [22] or d) a hierarchical multi-stage recognition pipeline [5]. Yet, all these recent papers focus on a relatively small-scale problem, *i.e.* recognition of a few hundreds different items at most, whilst usually several thousands products are on sale in a real shop. George *et al*. [6] address more realistic settings and propose a multi-stage system capable of recognizing ~ 3400 different products based on a single model image per product. The authors' contribution includes releasing the dataset employed in their experiments, which we will use in our evaluation. More recently, [24] has tackled the problem using a standard local feature based recognition pipeline and an optimized Hough Transform to detect multiple object instances and filter out inconsistent matches, which brings in a slight performance improvement.

Nowadays, CNN-based systems are dominating object detection benchmarks and can be subdivided into two main families of algorithms based on the number of stages required to perform detection. On one hand, we have the slower but more accurate two stage detectors [16], which decompose object detection into a region proposal followed by an independent classification for each region. On the other hand, fast one stage approaches [15, 9] can perform detection and classification jointly. A very recent work has also addressed the specific domain of grocery products, so as to propose an ad hoc detector [14] that analyzes the image at multiple scales to produce more meaningful region proposals.

Besides, deploying CNNs to obtain rich image representations is an established approach to pursue image search, both as a strong off-the-shelf baseline [19] and as a key component within more complex pipelines [7]. Inspiration for our product recognition approach came from [17, 3]. In [17], Schroff *et al*. train a CNN using triplets of samples to create an embedding for face recognition and clustering

while in [3] Bell *et al*. rely on a CNN to learn an image embedding to recognize the similarity between design products. Similarly, in the related field of fashion items recognition, relying on learned global descriptor rather than classifiers is an established solution shared among many recent works [8, 23, 18].

3. Proposed Approach

Figure 2 shows an overview of our proposed pipeline. In the first step, described subsection 3.1, a CNN (*Detector*) extracts region proposals from the *query* image. Then, as detailed in subsection 3.2, each region proposal is cropped from the *query* image and sent to another CNN (*Embedder*) which computes an ad-hoc image representation. These will then be deployed to pursue product recognition through a K-NN similarity search in a database of representations precomputed off-line by the *Embedder* on the *reference* images. Finally, as illustrated in subsection 3.3, we combine different strategies to perform a final refinement step which helps to prune out false detections and disambiguate among similar products.

3.1. Detection

Given a *query* image featuring several items displayed in a store shelf, the first stage of our pipeline aims at obtaining a set of bounding boxes to be used as region proposals in the following recognition stage. Ideally, each bounding box should contain exactly one product, fit tightly the visible package and provide a confidence score measuring how much the detection should be trusted.

State-of-the-art CNN-based object detectors may fulfill the above requirements for the product recognition scenario, as demonstrated in [14]. Given an input image, these networks can output several accurate bounding boxes, each endowed by a confidence and a class prediction. To train CNN-based object detectors, such as [15, 9, 16], a large set of images annotated with the position of the objects alongside with their class labels is needed. However, due to the ever-changing nature of the items sold in stores, we do not train the *Detector* to perform predictions at the fine-grained class level (*i.e*., at the level of the individual products), but to carry out a product-agnostic item detection. Accordingly, the in-store training images for our *Detector* can be annotated for training just by drawing bounding-boxes around items without specifying the actual product label associated with each bounding-box. This formulation makes the creation of a suitable training set and the training itself easier and faster. Moreover, since the *Detector* is trained only to recognize *generic products* from everything else it is general enough to be deployable across different stores and products. Conversely, training a CNN to directly predict bounding boxes as well as product labels would require a much more expensive and slow image annotation process which should be carried out again and again to keep up with changes of the products/packages to be recognized. This continuous re-training of the *Detector* is just not feasible in any practical settings.

3.2. Recognition

Starting from the candidate regions delivered by the *Detector*, we perform recognition by means of K-NN similarity search between a global descriptor computed on each candidate region and a database of similar descriptors (one for each product) pre-computed off-line on the *reference* images. Recent works (*e.g*., [19]) have shown that the activations sampled from layers of pre-trained CNNs can be used as high quality global image descriptors. [23] extended this idea by proposing to train a CNN (*i.e*., the *Embedder*) to learn a function $E : \mathcal{I} \rightarrow \mathcal{D}$ that maps an input image $i \in I$ into a k-dimensional descriptor $d^k \in \mathcal{D}$ amenable to recognition through K-NN search. Given a set of images with associated class labels, the training is performed sampling triplets of different images, referred to as *anchor* (i_a), *positive* (i_p) and *negative* (i_n), such that i_a and i_p depict the same class while i_n belongs to a different one. Given a distance function in the descriptor space, $d(\mathbf{X}, \mathbf{Y})$, with $X, Y \in \mathcal{D}$, and denoted as $E(i)$ the descriptor computed by the the *Embedder* on image i, the network is trained to minimize the so called *triplet ranking loss*:

$$\mathcal{L} = max(0, d(E(i_a), E(i_p)) - d(E(i_a), E(i_n)) + \alpha) \quad (1)$$

with α a fixed margin to be enforced between the pair of distances. Through minimization of this loss, the network learns to encode into nearby positions within $\mathcal{D}$ the images depicting items belonging to the same class, whilst keeping items of different classes sufficiently well separated.

We use the same formulation and cast it for the context of grocery product recognition where different products corresponds to different classes (*e.g*. the two reference images of Figure 1-(b) corresponds to different classes and could be used as i_p and i_n). Unfortunately, we can not sample different images for i_a and i_p due to available commercial datasets featuring just a single exemplar image per product (*i.e*., per class). Thus, to create the required triplet, at each training iteration we randomly pick two products and use their *reference* images as i_p and i_n. Then, we synthesize a new i_a from i_p by a suitable data augmentation function $A : \mathcal{I} \rightarrow \mathcal{I}$, to make it similar to *query* images (*i.e*., $i_a = A(i_p)$)[1].

To perform recognition, firstly, the *Embedder* network is used to describe each available *reference* image i_r by a global descriptor $E(i_r)$ and thus create the *reference*

[1]The details concerning the adopted augmentation function are reported in section 4

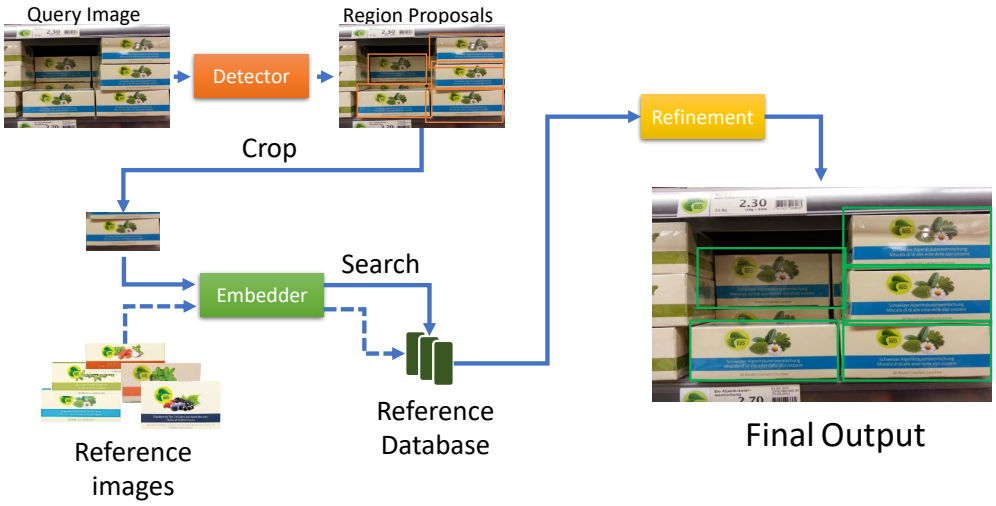


Figure 2: Schematic structure of our proposed product recognition pipeline. Dashed arrows denotes elaboration that can be performed offline just once since are not related to the *query* images.

database of descriptors associated with the products to be recognized. Then, when a *query* image is processed, the same embedding is computed on each of the candidate regions, i_{pq}, cropped from the *query* image, i_q, so to get $E(i_{pq})$. Finally, for each i_{pq} we compute the distance in the embedding space with respect to each *reference* descriptor, denoted as $d(E(i_{pq}), E(i_r))$, in order to sift-out the first K-NN of $E(i_{pq})$ in the *reference* database. These are subject to further processing in the final refinement step.

3.3. Refinement

The aim of the final refinement is to remove false detections and re-rank the first K-NN found in the previous step in order to fix possible recognition mistakes.

Since the initial ranking is obtained comparing descriptors computed on whole images, a meaningful re-ranking of the first K-NN may be achieved by looking at peculiar image details that may have been neglected while comparing global descriptors and yet be crucial to differentiate a product from others looking very similar. Thus, both the *Query* and each of the first K-NN *reference* images are described by a set of local features $F_1, F_2, ..., F_k$, each consisting in a spatial position (x_i, y_i) within the image and a compact descriptor f_i. Given these features, we look for similarities between descriptors extracted from *query* and *reference* images, to compute a set of matches. Matches are then weighted based on the distance in the descriptor space, $d(f_i, f_j)$ and a geometric consistency criterion relying on the unit-norm vector, $\vec{v}$, from the spatial location of a feature to the image center. In particular, given a match, $M_{ij} = (F_i^q, F_j^r)$, between feature i of the *query* image and feature j of the *reference* image, we compute the following weight:

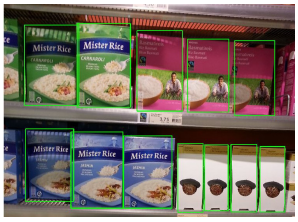
$$W_{ij} = \frac{(\vec{v}_i^q \cdot \vec{v}_j^r) + 1}{d(f_i^q, f_j^r) + \epsilon} \tag{2}$$

where $\cdot$ marks scalar products between vectors and ϵ is a small number to avoid potential division by zero. Intuitivelly W_{ij} is bigger for matching features which share the same relative position with respect to the image center (high $(\vec{v}_i^q \cdot \vec{v}_j^r)$) and have descriptors close in the feature space (small $d(f_i^q, f_j^r)$). Finally, the first K-NN are re-ranked according to the sum of the weights W_{ij} computed for the matches between the local features. In subsection 4.2 we will show how good local features can be obtained at zero computational cost as a by-product of our learned global image descriptor. This refinement technique will be referred to as ***+lf***.

A simple additional refinement step consists in filtering out wrong recognitions by the *distance ratio* criterion [11] (*i.e.*, by thresholding the ratio of the distances in feature space between the *query* descriptor and its 1-NN and 2-NN). If the ratio is above a threshold, τ_d, the recognition is deemed as ambiguous and discarded. In the following, we will denote this refinement technique as ***+th***.

Finally, as commercial product databases typically provide a multilevel classification of the items (*e.g.*, at least instance and category level), we propose a re-ranking and filtering method specific to the grocery domain where, as pointed out by [6], products belonging to the same macro category are typically displayed close one to another on the shelf. In particular, given the candidate regions extracted from the *query* image and their corresponding sets of K-NN, we consider the 1-NN of the region proposals extracted with a high confidence (> 0.1) by the *Detector* in order to find the main macro category of the image. Then, in case the

(a)-Customer[6]



(b)-Management[22]

Figure 3: In (a) the system should identify at least one instance for each product type, while in (b) it should find and correctly localize all the displayed product instances.

majority of detections votes for the same macro category, it is safe to assume that the pictured shelf contains almost exclusively items of that category thus filter the K-NN for all candidate regions accordingly. It is worth observing how this strategy implicitly leverages on those products easier to identify (*i.e.*, the high-confidence detections) to increase the chance to correctly recognize the harder ones. We will refer to this refinement strategy as to ***+mc***.

4. Experimental Results

To validate the performance of our product recognition pipeline we take into account two possible use cases dealing with different final users:

- **Customer**: the system should be deployed for a guided or partially automated shopping experience (*e.g.*, product localization inside the shop or augmented reality overlays or support to visually impaired). As proposed in [6], in this use case the goal is to detect *at least one* instance of each visible type of products displayed in a shelf picture.

- **Management**: the system will be used to partially automate the management of a shop (*e.g.*, automatic inventory and restocking). Here, the goal is to recognize *all* visible product instances displayed in a shelf picture.

4.1. Datasets and Evaluation Metrics

For our experimental evaluation we rely on the publicly available *Grocery Products* dataset [6], which features more than 8400 grocery products organized in hierarchical classes and with each product described by exactly one *reference* image. The dataset contains also 680 in-store (*query*) images that display items belonging to the *Food* subclass of the whole dataset. The annotations released by the authors for the *query* images allow for evaluating performance in the *Customer* use case, as they consist in bounding boxes drawn around spatial clusters of instances of the same products. To test our system also in the *Management* use case, we deploy the annotations released by the authors of [22], which consist of boxes around each product instance for a subset of 70 in-store pictures of the *Grocery Products* dataset. Figure 3 shows examples of the two kinds of annotations used to evaluate the system in the two different use cases.

To compare our work with previously published results we use the metrics proposed by the authors in [6]: mean average precision-*mAP* (the approximation of the area under the Precision-Recall curve for the detector) and Product Recall-*PR* (average product recall across all the test image). As for scenario (a), we report also mean average multi-label classification accuracy-*mAMCA*.

To train the *Detector* we acquired multiple videos with a tablet mounted on a cart facing the shelves and manually labeled a subset of sampled frames to create a training set of 1247 images. Thus, our videos are acquired in different stores with respect to those depicted in the *Grocery Products* dataset and feature different products on different shelves, vouching for the generalization ability of our system.

4.2. Implementation Details

For all our tests we have used as *Detector* the state-of-the-art one-stage object detector known as yolo_v2 (shortened in *yolo*) [15]. We choose this network as it grants real-time performance on a GPU and for the availability of the original implementation. Starting from the publicly available weights[2], we have fine tuned the network on our 1247 in-store images for 80000 steps keeping the hyperparameters suggested by the original authors.

The backbone network for our *Embedder* is a VGG_16 [20] pre-trained on the Imagenet-1000 classification task (weights publicly available[3]). From this network we obtain global image descriptors by computing *MAC* features [21] on the conv4_3 convolutional layer and applying L2 normalization to obtain unit-norm embedding vectors. To carry out the comparison between descriptors, both at training and test time, we used as distance function $d(X,Y) = 1 - X \cdot Y$ with $X, Y \in \mathcal{D}$ (*i.e.*, 1 minus the cosine similarity between the two descriptors). To better highlight the advantage of learning an ad-hoc descriptor in the following we will report experiments using general purpose descriptors obtained without fine tuning the *Embedder* with the suffix *_gd* (general descriptor), while descriptors obtained after fine-tuning as described in subsection 3.2 will be denoted by the suffix *_ld* (learned descriptor). To train the *Embedder* we use the *reference* images of *Grocery Products* dealing with the products belonging to the *Food* subclass (*i.e.*, 3288 different product with exactly one training image for each).

[2] https://github.com/pjreddie/darknet

[3] https://github.com/tensorflow/models/tree/master/research/slim

Method	mAP(%)	PR(%)	mAMCA(%)
FV+RANSAC[6]	11.26	23.14	6.41
RF+PM+GA[6]	23.49	43.13	21.19
FM+HO[24]	23.71	41.60	**32.50**
yolo_gd	21.49	47.03	13.34
yolo_ld	27.84	53.17	16.32
yolo_ld+th	30.46	37.88	28.74
yolo_ld+lf	32.34	**58.41**	17.72
yolo_ld+mc	30.15	55.25	21.09
yolo_ld+lf-mc-th **(full)**	**36.02**	57.07	31.57

Table 1: Product recognition on the *Grocery Products* dataset in the *Customer* scenario. Best result highlighted in bold. Our proposals (in italic) yields large improvements in terms of both mAP and PR with respect to previously published results.

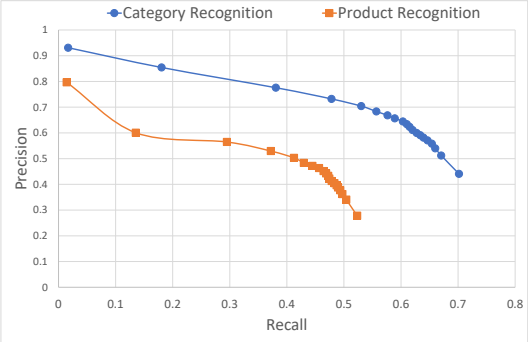


Figure 4: Precision-recall curves obtained in the *customer* use case by the *yolo_ld* system when trying to recognize either the individual products or just their category.

To enrich the dataset and create the anchor images, i_a, we randomly perform the following augmentation functions A: blur by a Gaussian kernel with random σ, random crop, random brightness and saturation changes. This augmentations were engineered so to transform the *reference* images in a way that renders them similar to the proposals cropped from the *query* images. The hyper-parameters obtained by cross validation for the training process are as follows: $\alpha = 0.1$ for the triplet loss, learning rate $lr = 0.000001$, ADAM optimizer and fine-tuning by 10000 steps with batch size 24.

We propose a novel kind of local features for the *+lf* refinement: as *MAC* descriptors are obtained by applying a max-pool operation over all the activations of a convolutional layer, by changing the size and stride of the pool operation it is possible to obtain a set of local descriptor with the associated location being the center of the pooled area reprojected into the original image. By leveraging on this intuition, we can obtain in a single forward computation both a global descriptor for the initial K-NN search (subsection 3.2) as well as a set of local features to be deployed in the refinement step (subsection 3.3). For our test we choose kernel size equal 16 and stride equals 2 as to have 64 features per *reference* image.

4.3. Customer Use Case

In this sub-section we evaluate the effectiveness of our system in the *Customer* scenario. To measure performance we rely on the annotations displayed in Figure 3-(a) and score a correct recognition when the product has been correctly identified and its bounding box has a non-empty intersection with that provided as ground-truth. We compare our method with already published work tested on the same dataset: FV+RANSAC (Fisher Vector classification re-ranked with RANSAC) [6], RF+PM+GA (Category prediction with Random Forests, dense Pixel Matching and Genetic Algorithm) [6] and FM+HO (local feature matching optimized with Hough) [24]. We report the results obtained by the different methods according to the tree metrics in Table 1. As [24] does not provide the mAP figure but only the values of precision and recall, for FM+HO we report an approximate mAP computed by multiplying precision and recall.

Using our trained *yolo* network for product detection, in Table 1 we report the results obtained by either deploying a general purpose VGG-based descriptor ($yolo_{gd}$) or learning an ad-hoc embedding for grocery products $yolo_{ld}$. Moreover, we report the results achieved with the different refinement strategies presented in subsection 3.3. Table 1 shows that our pipeline can provide a higher recall than previous methods even with a general purpose image descriptor (*yolo_gd*), although with a somehow lower precision, as demonstrated by the slightly inferior mAP score. However, our complete proposal relies on learning an ad-hoc descriptor for grocery products (*yolo_ld*), which yields a significant performance improvement, as vouched by an average gain of about 6% in terms of both Recall and mAP. Wrongly classified proposals can be discarded to further improve accuracy by the threshold refinement strategy (*yolo_ld + th* - with $\tau_d = 0.9$), thereby increasing the mAMCA from 16.32% to 28.74%. Re-ranking based on the proposed local features (*yolo_ld+lf*) turns out an effective approach to ameliorate both precision and recall, as demonstrated by a gain of about 5% in mAP and PR with respect to the pipeline without final refinement (*yolo_ld*). The category-based re-ranking strategy (*yolo_ld + mc*) seems to fix some of the recognition mistakes and improve the recognition rate with respect to (*yolo_ld*) , providing gains in all metrics. Finally, by mixing all the refinement strategies to obtain our overall pipeline (*yolo_ld+lf-mc-th*), we neatly get the best trade-off between precision and recall, as vouched by the 57.07% PR and 36.02% mAP, *i.e.* about 14% and 12.5% better than previously published results, respectively, with a mAMCA turning out nearly on par with the best previous result.

Method	mAP(%)	PR(%)
FS[22]	66.37	75.0
yolo_gd	66.95	78.89
yolo_ld	74.32	84.75
yolo_ld+th	75.62	81.55
yolo_ld+lf	76.37	**86.56**
yolo_ld+mc	74.80	85.28
yolo_ld+lf-mc-th **(full)**	**76.93**	85.71

Table 2: Product recognition for *Management* use case. Our proposal highlighted (in italic), best results in bold.

Method	mAP(%)	PR(%)
FS[22]	47.32	57.0
yolo_gd	60.17	73.66
yolo_ld	67.88	80.27
yolo_ld+th	69.70	76.01
yolo_ld+lf	70.69	**82.83**
yolo_ld+mc	69.01	81.55
yolo_ld+lf-mc-th **(full)**	**73.50**	82.66

Table 3: Results in the *Management* use case performing recognition against all the items belonging to the Food subclass of the *Grocery Products* dataset ($\sim$ 3200). Our proposals highlighted (in italic), best results in bold.

We found that casting recognition as a similarity search through learned global descriptors has the nice property that even when the 1-NN does not correspond to the right product, it usually corresponds to items belonging the correct category (*i.e.* cereals, coffee,...). We believe this behavior being due to items belonging to the same category sharing similar peculiar visual patterns that are effectively captured by the descriptor and help to cluster nearby in descriptor space items belonging to the same categories (*e.g.*, coffee cups often displayed on coffee packages or flowers on infusions). To highlight this generalization property, we perform here an additional test in the Customer scenario by considering a recognition as correct if the category of the 1-NN match is the same as those of the annotated bounding box. Accordingly, we compare the performance of *yolo_ld* when trying to recognize either the individual products or their category. The results of this experiment are reported as Precision-Recall curves in Figure 4. The large difference between the two curves proves that very often the system mistakes items at product level though correctly recognizing their category. Eventually, it is worth pointing out that our method not only provides a significant performance improvement with respect to previously published results but turns out remarkably fast. Indeed, our whole pipeline can be run on a GPU in less than one second per image.

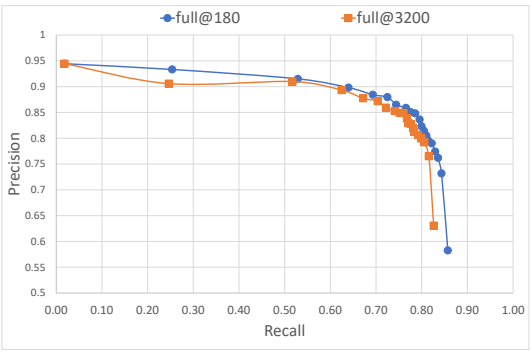


Figure 5: Precision-Recall curves for our full pipeline in the *Management* use case. full@180 denotes performing recognition on the small *reference* database of [22] ($\sim$ 180 entries), full@3200 against all the products in the *Food* category of *Grocery Products* ($\sim$ 3200).

4.4. Management Use Case

The experiments presented in this Section concern the *Management* use case, a task requiring correct recognition of all the individual products displayed on shelves. Thus, we rely on the annotations shown in Figure 3-(b) and consider a recognition as correct when the item has been correctly recognized and the intersection over union (IoU) between the predicted and ground truth bounding boxes is higher than 0.5. To the best of our knowledge, the only published results on the *Grocery Products* dataset that deals with recognition of all the individual products are reported in [22]. However, in [22] the authors address a specific task referred to as *Planogram Compliance*, which consists in checking the compliance between the actual product disposition and the planned one. Accordingly, the pipeline proposed in [22] includes an initial unconstrained product recognition stage, which addresses the same settings as our *Management* use case, followed by a second stage that deploys the exact knowledge on the planned product disposition in order to improve recognition accuracy and detect compliance issues (*e.g.*, missing or misplaced products). Therefore, we compare our proposal to the most effective configuration of the first stage of the pipeline presented in [22], referred to hereinafter as *FS*, which is based on matching BRISK local features[10] followed by Hough Voting and pose estimation through RANSAC.

To compare our pipeline with respect to *FS*, we use the annotations provided by the authors and perform recognition against the smaller *reference* database of 182 products used in [22]. The results are reported in Table 2. Firstly, it is worth pointing out how, despite the task being inherently more difficult than in the *Customer* use case, we record higher recognition performance. We ascribe this mainly to the smaller subset of in-store images used for testing, (*i.e.*,

(a) (b) (c)

Figure 6: Examples of correct product recognitions in *query* images from *Grocery Products*.

70 vs. 680) as well as to these images featuring mainly rigid packaged products, which are easier to recognize than deformable packages. Once again, in our pipeline, the use of a learned descriptor (*yolo_ld*) provides a substantial performance gain with respect to a general purpose descriptor (*yolo_gd*), as the mAP improves from 66.95% to 74.32% and the PR from 78.89% to 84.75%. The different refinement strategies provide advantages similar to those discussed in subsection 4.3, the best improvement yielded by re-ranking recognitions based on the local features extracted from the *Embedder* network (*yolo_ld+lf*). The optimal trade-off between precision and recall is achieved again by deploying together all the refinement strategies (*yolo_ld+lf-mc-th*), which provided a mAP and PR as high as 76.93% and 84.75%, respectively (*i.e.*, both about 10% better than the previously published results on this dataset).

Table 3 reports results aimed at assessing the scalability of the methods with respect to the number of products in the *reference* database. We carried out an additional experiment by performing the recognition of each item detected within the 70 *query* images against all the 3200 products of the "Food" category in *Grocery Products* rather than the smaller subset of 182 products proposed in [22]. By comparing the values in Table 2 and Table 3, we can observe how, unlike *FS*, our method can scale nicely from few to thousands of different products: our full method *yolo_ld+lf-mc-th* looses only 3.43% mAP upon scaling-up the *reference* database quite significantly, whilst the performance drop for *FS* is as relevant as 19.05%. In Figure 5 we also plot the precision-recall curves obtained by our full pipeline (*yolo_ld+lf-mc-th*) using the smaller (full@180) and larger (full@3200) sets of reference products. The curves show clearly how our pipeline can deliver almost the same performance in the two setups, which vouches for the ability of our proposal to scale smoothly to the recognition of thousands of different products. As far as recognition time is concerned, our pipeline can scale fairly well regardless of the size of the *reference* database, due to the NN search, even if extensive, amounting to a negligible fraction of the overall computation: the difference in inference time between recognizing 180 and 3200 product is less than a tenth of a second.

5. Qualitative Results

Figure 6 reports some qualitative results obtained by our pipeline. Picture (a) shows the recognition results on an image taken quite far from the shelf and featuring a lot of different items; (b) deal with some successful recognitions in a close-up *query* image, where only a few items are visible at once. Finally, (c) refers to recognition of products featuring deformable and highly reflective packages, which are quite challenging to recognize due to the appearance of the items within the *query* images turning out significantly different than in the available *reference* images. Yet, in (c) our system was able to find at least one item for each product type (*i.e.*, as required in the *Customer* use case).

6. Conclusion

In this paper we have proposed a fast and effective approach to the problem of recognizing grocery products on store shelves. Our proposal addresses the task by three main steps: class agnostic object detection to identify the individual items appearing on a shelf image, recognition through K-NN similarity search based on a global image descriptor, final refinement to further boost performance. All the three steps deploy modern deep learning techniques, as we detect items by a state-of-the-art CNN (*Detector*), learn the image descriptor by another CNN trained to disentangle the appearance of grocery products (*Embedder*) and extract local cues key to refinement as MAC features computed alongside with the global embedding.

The experiments prove that our pipeline compares favourably to the state-of-the-art on the public dataset available for performance assessment while being remarkably fast. Yet, we plan to investigate how to further improve the speed at test time (*e.g.*, to enable execution on a low-cost and/or mobile device). Purposely, we envisage devising a unified CNN architecture acting as both *Detector* and *Embedder*. Furthermore we are currently investigating on the use of generative models (*e.g.*, GANs) to augment the number of samples per product to train the *Embedder*. A generative model could also be trained to render *reference* more similar to proposals cropped from *query* images in order to shrink the gap between the training and testing domains.

References

[1] S. Advani, B. Smith, Y. Tanabe, K. Irick, M. Cotter, J. Sampson, and V. Narayanan. Visual co-occurrence network: using context for large-scale object recognition in retail. In *Embedded Systems For Real-time Multimedia (ESTIMedia), 2015 13th IEEE Symposium on*, pages 1–10. IEEE, 2015.

[2] I. Baz, E. Yoruk, and M. Cetin. Context-aware hybrid classification system for fine-grained retail product recognition. In *Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), 2016 IEEE 12th*, pages 1–5. IEEE, 2016.

[3] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 34(4):98, 2015.

[4] M. Cotter, S. Advani, J. Sampson, K. Irick, and V. Narayanan. A hardware accelerated multilevel visual classifier for embedded visual-assist systems. In *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, pages 96–100. IEEE Press, 2014.

[5] A. Franco, D. Maltoni, and S. Papi. Grocery product detection and recognition. *Expert Systems with Applications*, 2017.

[6] M. George and C. Floerkemeier. Recognizing products: A per-exemplar multi-label image classification approach. In *Computer Vision–ECCV 2014*, pages 440–455. Springer, 2014.

[7] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *European Conference on Computer Vision*, pages 241–257. Springer, 2016.

[8] M. Hadi Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg. Where to buy it: Matching street clothing photos in online shops. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3343–3351, 2015.

[9] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3296–3297, 2017.

[10] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011.

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[12] M. Merler, C. Galleguillos, and S. Belongie. Recognizing groceries in situ using in vitro training data. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[13] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3384–3391. IEEE, 2010.

[14] S. Qiao, W. Shen, W. Qiu, C. Liu, and A. Yuille. Scalenet: Guiding object proposal generation in supermarkets and beyond. In *2017 IEEE International Conference on Computer Vision, ICCV*, pages 22–29, 2017.

[15] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition,CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6517–6525, 2017.

[16] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[17] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[18] D. Shankar, S. Narumanchi, H. Ananya, P. Kompalli, and K. Chaudhury. Deep learning based large scale visual recommendation and search for e-commerce. *arXiv preprint arXiv:1703.02344*, 2017.

[19] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[21] G. Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of cnn activations. 2016.

[22] A. Tonioni and L. Di Stefano. Product recognition in store shelves as a sub-graph isomorphism problem. In *International Conference on Image Analysis and Processing*, pages 682–693. Springer, 2017.

[23] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.

[24] E. Yörük, K. T. Öner, and C. B. Akgül. An efficient hough transform for multi-instance object recognition and pose estimation. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 1352–1357. IEEE, 2016.