applied sciences

MDPI

Article

Computer-Vision-Based Planogram Compliance Evaluation

Julius Laitala [1] and Laura Ruotsalainen [2,*]

[1] RELEX Solutions, 00230 Helsinki, Finland
[2] Department of Computer Science, University of Helsinki, 00014 Helsinki, Finland
* Correspondence: laura.ruotsalainen@helsinki.fi

Abstract: Arranging products in stores according to planograms, optimized product arrangement maps, is an important sales enabler and necessary for keeping up with the highly competitive modern retail market. Key benefits of planograms include increased efficiency, maximized retail store space, increased customer satisfaction, visual appeal, and increased revenue. The planograms are realized into product arrangements by humans, a process that is prone to mistakes. Therefore, for optimal merchandising performance, the planogram compliance of the arrangements needs to be evaluated from time to time. We investigate utilizing a computer vision problem setting—retail product detection—to automate planogram compliance evaluation. Retail product detection comprises product detection and classification. The detected and classified products can be compared to the planogram in order to evaluate compliance. In this paper, we propose a novel retail product detection pipeline combining a Gaussian layer network product proposal generator and domain invariant hierarchical embedding (DIHE) classifier. We utilize the detection pipeline with RANSAC pose estimation for planogram compliance evaluation. As the existing metrics for evaluating the planogram compliance evaluation performance assume unrealistically that the test image matches the planogram, we propose a novel metric, called normalized planogram compliance error (E_{PC}), for benchmarking real-world setups. We evaluate the performance of our method with two datasets: the only open-source dataset with planogram evaluation data, GP-180, and our own dataset collected from a large Nordic retailer. Based on the evaluation, our method provides an improved planogram compliance evaluation pipeline, with accurate product location estimation when using real-life images that include entire shelves, unlike previous research that has only used images with few products. Our analysis also demonstrates that our method requires less processing time than the state-of-the-art compliance evaluation methods.

Keywords: computer vision; object detection; planograms

Citation: Laitala, J.; Ruotsalainen, L. Computer-Vision-Based Planogram Compliance Evaluation. *Appl. Sci.* **2023**, *13*, 10145. https://doi.org/10.3390/app131810145

Academic Editors: Mukesh Prasad, Pang-jo Chun, Xian Tao and Ali Braytee

Received: 26 July 2023
Revised: 3 September 2023
Accepted: 6 September 2023
Published: 8 September 2023

1. Introduction

In the modern, highly competitive retail environment, work efficiency is essential and requires algorithmic optimization of operations. This includes optimization of the display and arrangement of products in stores: *planograms*, charts that show where each product should reside on a shelf, are often generated by specialized software, and the products are then arranged accordingly. Studies have shown that complying with optimized planograms can increase sales up to 7–8%, and that the total cost of suboptimal merchandising sales performance in the US is approximately 1% of gross product sales [1]. Over 60 percent of purchase decisions are made at the point of sale for which visibility and presentation, and thereby decisions made through a planogram, are vital for motivating sales.

As the planograms are "realized" into product arrangements by people, the possibility of human error is always present. Therefore, the arrangements need to be checked for *planogram compliance*—whether the arrangement of products in-store truly matches the desired arrangement given by the planogram—from time to time. This, too, is still a manual task, requiring costly human resources. Computers, on the other hand, have better

memories, do not make mathematical errors, and are better at multitasking than humans, and therefore a computer-based compliance method is appealing. We encourage the reader who is less familiar with the latest deep learning-based methods of computer vision to look for background information from the field's foundational sources [2,3].

Object detection is a computer vision task that could provide automation for planogram compliance evaluation. Most object detection literature deals with *generic object detection*—recognizing everyday objects in everyday environments. This includes approaches such as feature pyramid networks (FPNs) [4], region-based convolutional neural networks (R-CNNs) [5], and you only look once (YOLO) [6], as well as challenges such as ImageNet [7] and MS COCO [8]. However, recognizing retail products in the environment of racks is a mix of generic object detection and *object instance detection.* Object instance detection means detecting a specific object instead of a general class, as is carried out in general object detection. In the retail product detection problem setting, the aim is to localize and classify products from images of shelves, cabinets, racks, or bays taken in retail store aisles. The recognition process is called the *retail product detection problem* [9], and it exhibits a number of difficulties in comparison to this more general problem that has made using computer vision for planogram compliance verification infeasible for quite some time.

The two most fundamental difficulties of retail product detection—the small amount of available training data per class and the domain shift between training data and test data—were identified already by [10], who kick-started research into retail product detection in 2007. Other difficulties of retail product detection when compared to generic object detection and object instance detection include the very high number of object classes (in the order of several thousand for small shops and tens of thousands for hypermarkets); the closeness of the appearance of the object classes (for example, for different flavors of the same product from the same brand), requiring fine-grained classification; and the frequently changing assortment-making models that require slow retraining infeasible [11]. Due to these peculiarities, the results achieved by [10]—and, indeed, results achieved with any pure generic object detection or instance detection methods—were quite poor.

Retail product detection approaches that perform well enough to be utilized in practice have emerged only recently. Local invariant feature-based approaches have been utilized to solve the problem well beyond the breakthrough of CNNs for generic object detection. Notable approaches that utilize invariant features, in addition to the seminal [10], include combining corner detection, color information, and Bag of Words [12]; utilizing BRISK features and Hough transform estimation [13]; and detecting products one shelf level at a time with SURF features and dimensionality information-based refinement [14].

The first papers discussing the use of deep learning for retail product detection were published only in 2017. Initially, deep learning was utilized in hybrid approaches, that is, in combination with local invariant features. In notable hybrid models, deep learning was used for product classification [12] showing the power of deep learning over more traditional methods in the case of complex (that is, realistic) scenarios and [15] presenting a non-parametric probabilistic model for initial detection with CNN-based refinement trying to overcome the lack of training data. An attention mechanism was added by Geng et al. [16], improving the detection accuracy and adaptability to new product classes without retraining.

Starting in 2018, fully deep-learning-based product detection pipelines have started to emerge. These include combining a class-agnostic object detector with an encoder network utilized for classification into a model that, after being trained once, was able to fit new stores, products, and packages of existing products [11] and combining RCNN [5] object detection and ResNet-101 [17] classification with a specialized non-maximal suppression approach for rejecting region proposals for unlikely objects arising from overlapping products in images [18].

In addition to end-to-end retail product detection, research has been carried out towards *product proposal generation*—detecting the objects without classifying them. The main

obstacle with this research was, until 2019, the lack of quality training data, with attempts to circumvent this via, e.g., 3-D renderings [19]. In 2019, however, Ref. [20] introduced the SKU-110K dataset, including images collected from thousands of supermarket stores and being of various scales, viewing angles, lighting conditions, and noise levels and thus enabled the training of deep product proposal generators. The baseline product proposal generation approach of [20] consisted of combining RetinaNet with a Gaussian-based non-maximal suppression. Later utilizations of SKU-110K include baking the Gaussian non-maximal suppression into the product proposal generator network itself, thus resulting in a multi-task learning problem [21], and addressing object rotations and outlier training samples with engineered-for-purpose neural network models [22].

The other end of end-to-end retail product detection—that is, *retail product classification*—has also seen some research separate from the full problem. One notable approach is that of [23], which combines training an encoder for k-nearest neighbor (KNN) product classification with utilizing generative adversarial networks (GANs) for data augmentation and a distance measure that takes into account the product hierarchy.

The planogram compliance evaluation problem is not solved by just detecting the products. Additionally, the detected arrangement needs to be compared to the planogram, and any deviations noted. This is not trivial, either: the exact positions where the products given in the planogram should be in the image are rarely known.

Planogram compliance evaluation is not as well researched as retail product detection. The seminal articles [1,24] came out in 2015, with approaches consisting of the utilization of detecting subsections of shelves [1] and distance metrics of the current shelf image with an image of the shelf taken in its ideal, planogram-compliant state [24]. Other notable approaches to the problem include viewing it as a recurring pattern detection problem [25], evaluating compliance one shelf at a time with the help of directed acyclic graphs (DAGs) [14], and viewing the product arrangements as graphs and solving for a maximum common subgraph between them [13]. The evaluation methods used in existing research on planogram compliance evaluation seem to assume that the test images match the planograms and to then just calculate standard object detection performance metrics based on how many of the facings were indeed detected. We argue that for a real-world use case, having test images that do not fully match the planogram is more interesting than having only a hundred percent compliant images. To enable evaluating planogram compliance evaluation methods with datasets that correct this deficiency, we propose a novel metric called normalized planogram compliance error E_{PC}.

Some of the recent developments bordering retail product detection have not been to our knowledge yet utilized in a full retail product detection pipeline. On the one hand, the SKU-110K dataset [20] has enabled the training of product proposal generators, deep neural networks that locate the products without classifying them, on an unprecedented scale. The dataset has already been successfully utilized in networks such as the Gaussian layer network (GLN) [21] and the dynamic refinement network [22]. On the other hand, the domain invariant hierarchical embedding (DIHE) [23] introduced the idea of utilizing GANs to train an encoder for KNN classification to be used for product classification purposes.

To enable planogram compliance evaluation on large, real-world shelves and to take planogram compliance evaluation to the neural network era, we propose a novel retail product detection pipeline using GLN for product detection, DIHE for product classification, and RANSAC for pose estimation [26]. Our proposed pipeline is suitable for use in all brick-and-mortar stores regardless of the retail vertical. As the conventional metrics used for evaluating the performance of object detection are not suitable for planogram compliance detection, we propose a novel metric. We summarize our main contributions as follows:

- We propose a novel retail product detection pipeline by combining a product proposal generator [21] with a product classification approach [23], and compare the product proposal generation, product classification, and retail product detection performance of the resulting framework to existing approaches.

- We utilize the product detection pipeline and RANSAC pose estimation [26] to evaluate planogram compliance on real retail data consisting of full planograms and images that are not fully planogram compliant, while previous research has focused on small fully compliant subsections of planograms
- We propose a novel metric, the mean-squared planogram compliance error, for benchmarking planogram compliance detection performance.

2. Materials And Methods

Our proposed approach consists of a two-stage product detector and a RANSAC comparison between the planogram and the detections. We choose a two-stage detector over a one-stage one due to a number of reasons related to the retail product detection problem: one-stage approaches need retraining whenever the assortment changes; one-stage detectors have been shown to fare poorly against two-stage approaches when dealing with a multitude of small objects, which is a common scenario in retail product detection [27]; and a two-stage detector allows us to choose and improve upon both of the components separately.

The proposed two-stage detector utilizes a deep product proposal generator [21] as its first stage, and classifies these proposals with an encoder-based approach [23] in the second stage. The first stage is selected due to its impressive performance with the SKU-110K dataset, which leads us to expect top-level performance in real-world use. The second stage is selected for its novel way of dealing with the domain shift, its ability to handle an arbitrary amount of classes, and its ability to classify products that were not present in the training data.

The detector outputs an empirical planogram, which is then compared to the expected planogram utilizing RANSAC-based pose estimation. The approach is visualized in Figure 1.

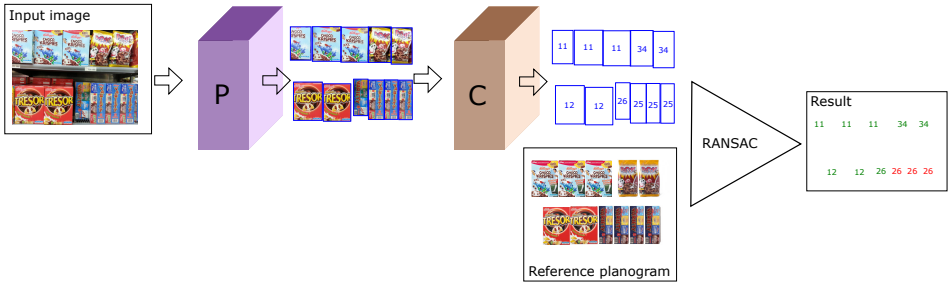
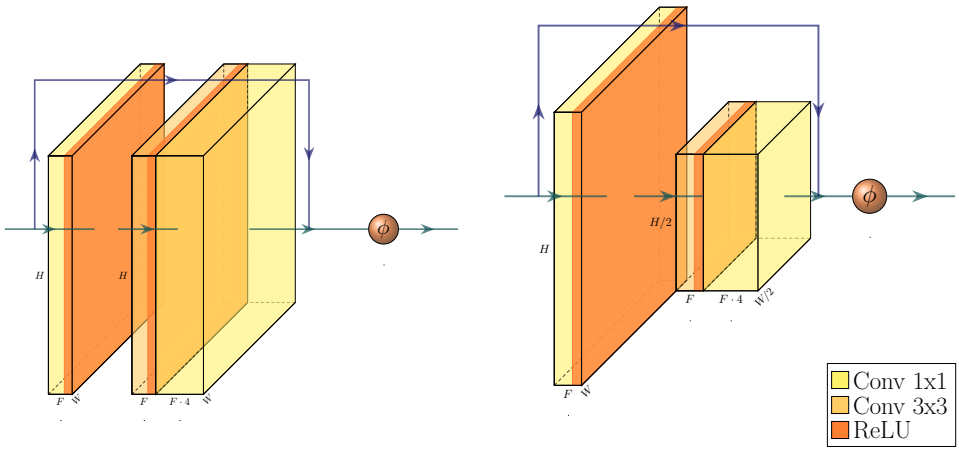


Figure 1. A visualization of the proposed approach. The inputs consist of the input image and the reference planogram. The purple box marked with *P* represents the product proposal generator, and the brown box marked with *C* represents the product classifier. The result shows which of the products expected in the reference planogram were included in the input image (green numbers) and which were not found (red numbers). The input image and the images that the reference planogram consists of are from the Grocery Products dataset of George and Floerkemeier [28].

2.1. Product Proposal Generation

We follow the example of [21] in choosing ResNet-50, a fully convolutional object classification network, pre-trained with the ImageNet dataset [7], as the backbone for our product proposal generator. The backbone architecture is visualized in Figures 2 and 3.

We build a feature pyramid network (FPN, Ref. [4]) on top of this backbone, and augment the architecture further by making our network training a multi-task learning problem via the introduction of a Gaussian layer and subnet [21]. The architecture of the resulting network is visualized in Figures 4–7. We further explore substituting the ReLU activation of the final layer of the Gaussian subnet with Tanh. Our intuition here is that as the prediction target of the Gaussian subnet is limited to the range $[0, 1]$, it makes sense to bar the neural network from producing values outside of this range. This alternative is visualized in Figure 6b.



(**a**) Basic bottleneck. (**b**) Downsampling bottleneck.

Figure 2. The bottlenecks in the PyTorch [29] implementation of ResNet-50. The input width is W, height is H, and it consists of F, $F \cdot 2$ or $F \cdot 4$ feature maps, depending on the position in the network. The output from the bottleneck consists of $F \cdot 4$ feature maps of size $H \times W$ in the case of basic bottleneck or $W/2 \times H/2$ in the case of downsampling bottleneck. The identity connection (the blue arrow from input to before the final ReLU activation) is implemented as simple summation if the input is of the same shape as the output. Otherwise a 1×1 convolution, possibly with a stride of 2, is applied before the summation. The downsampling in the downsampling bottleneck is implemented via a stride of 2 in the middle, 3×3 convolution layer. The PyTorch implementation differs from the original ResNet-50, introduced by He et al. [17], by downsampling with a stride of 2 in the 3×3 convolution layer instead of in the first, 1×1 convolution layer.

As suggested by [21], we use a weighted sum of three different loss functions, each calculated from different outputs of our product proposal generator network. Our full loss function is

$$\mathcal{L} = \alpha_{\text{focal}} \mathcal{L}_{\text{focal}} + \mathcal{L}_{L_1} + \alpha_{\text{MSE}} \mathcal{L}_{\text{MSE}} \,. \tag{1}$$

In the above equation, $\mathcal{L}_{\text{focal}}$ is *focal loss* [30]. We use it for foreground–background classification loss. It is calculated as

$$\mathcal{L}_{\text{focal}}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \,, \tag{2}$$

where

$$\alpha_t = \begin{cases} \alpha & \text{if the detection belongs to the foreground,} \\ 1 - \alpha & \text{otherwise} \end{cases} . \tag{3}$$

$\alpha \in [0, 1], \gamma \geq 0$ are hyperparameters,

$$p_t = \begin{cases} p & \text{if the detection belongs to the foreground,} \\ 1 - p & \text{otherwise} \end{cases} , \tag{4}$$

and $p \in [0, 1]$ is the estimated probability that the detection belongs to the foreground [30].

$\mathcal{L}_{L_1}$ is L_1 loss, or, in other words, mean absolute error. We use it as our bounding box regression loss. It is defined as

$$\mathcal{L}_{L_1}(x, \hat{x}) = |x - \hat{x}| \,, \tag{5}$$

where x is the ground truth, and $\hat{x}$ is the prediction.

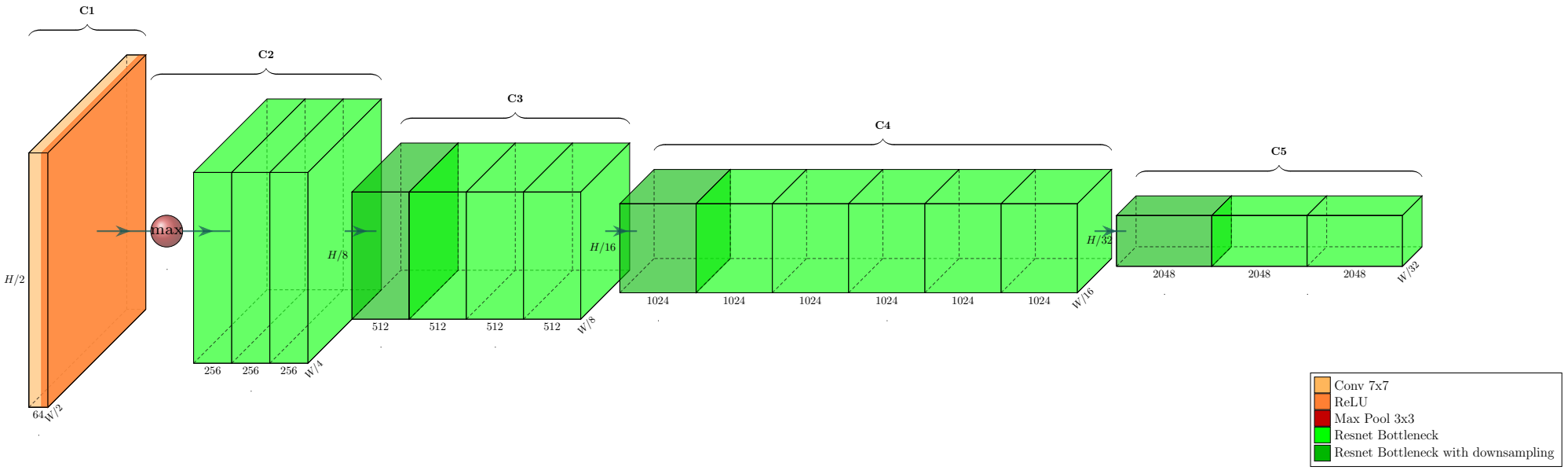
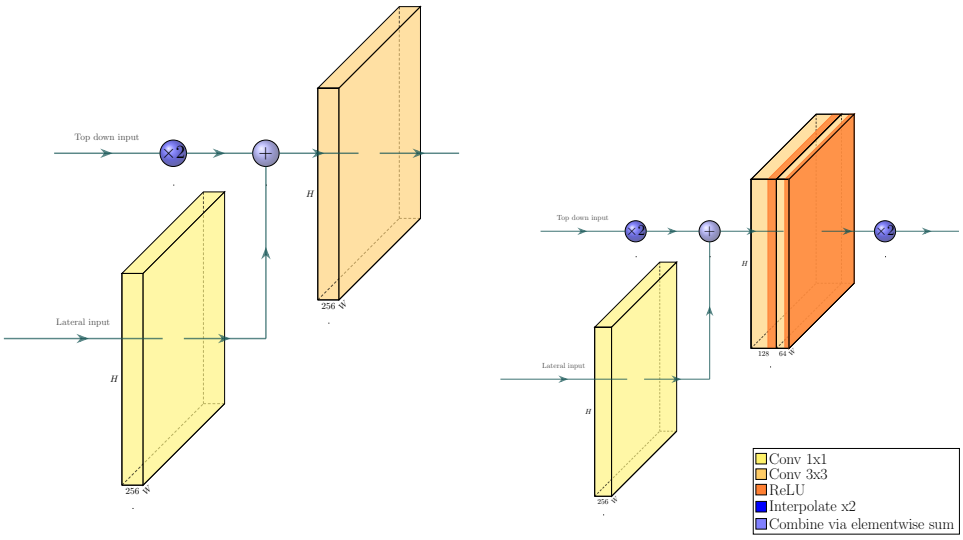
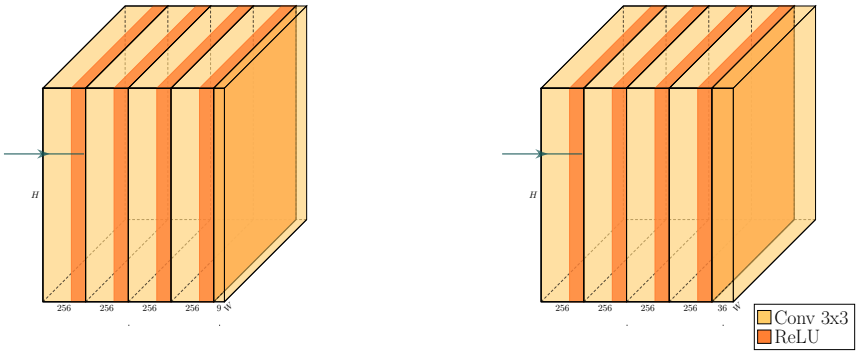


Figure 3. The PyTorch implementation of the ResNet-50 backbone, as used in RetinaNet [17,29,30]. The input width is W, height is H, and the number of input channels is 3. ResNet bottleneck layers are presented more in detail in Figure 2. The shape of the features after the last bottleneck is $2048 \times H/32 \times W/32$. The layers are grouped to C1, C2, $\cdots$, C5; these groups are equivalent to conv1, conv2.x, $\cdots$, conv5.x in He et al. [17].



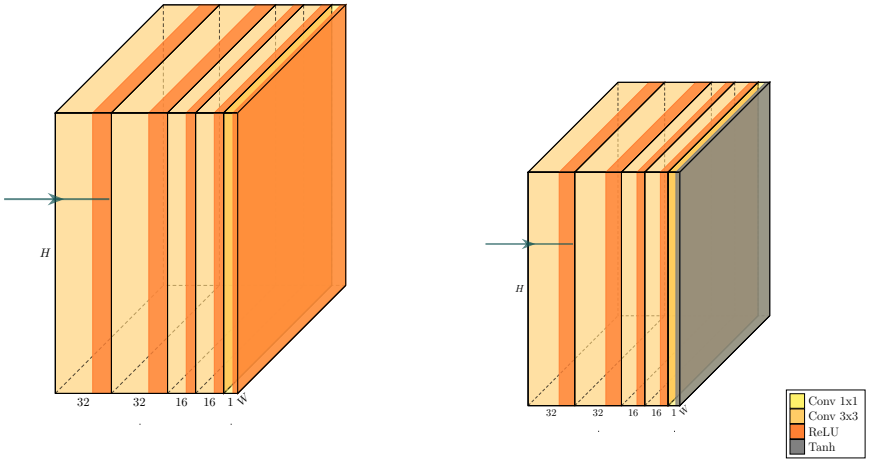
(**a**) Feature pyramid network block. (**b**) Gaussian layer.

Figure 4. Feature pyramid network block as given in the PyTorch implementation of RetinaNet [29,30] and the Gaussian layer introduced by Kant [21]. In both figures, the top-down input is of shape $256 \times H/2 \times W/2$, and the lateral input has shape $F \times H \times W$, with varying F. The output shape is $256 \times H \times W$ for the feature pyramid network block and $64 \times 2H \times 2W$ for the Gaussian layer. Interpolation is performed with the nearest neighbor method in all cases.



(**a**) Classification head. (**b**) Bounding box regression head.

Figure 5. The classification and regression heads of the PyTorch implementation of RetinaNet [29,30]. The input shape is $256 \times H \times W$. The output shape is $AK \times H \times W$ for the classification head and $4A \times H \times W$ for the regression head, where A is the number of anchors per location and K the number of classes. In our single class, the object detection case, using the default values of the PyTorch implementation, gives us an output of $9 \times H \times W$ for classification and $36 \times H \times W$ for bounding box regression.



(**a**) Gaussian subnet with final ReLU activation. (**b**) Gaussian subnet with final Tanh activation.

Figure 6. The Gaussian subnet used in Kant [21] (**a**) and our proposed modification with hyperbolic tangent activation (**b**). The input shape is $64 \times H \times W$. The output of the subnet consists of a Gaussian map, with shape $1 \times H \times W$ and each pixel representing the density of the map at the given point.

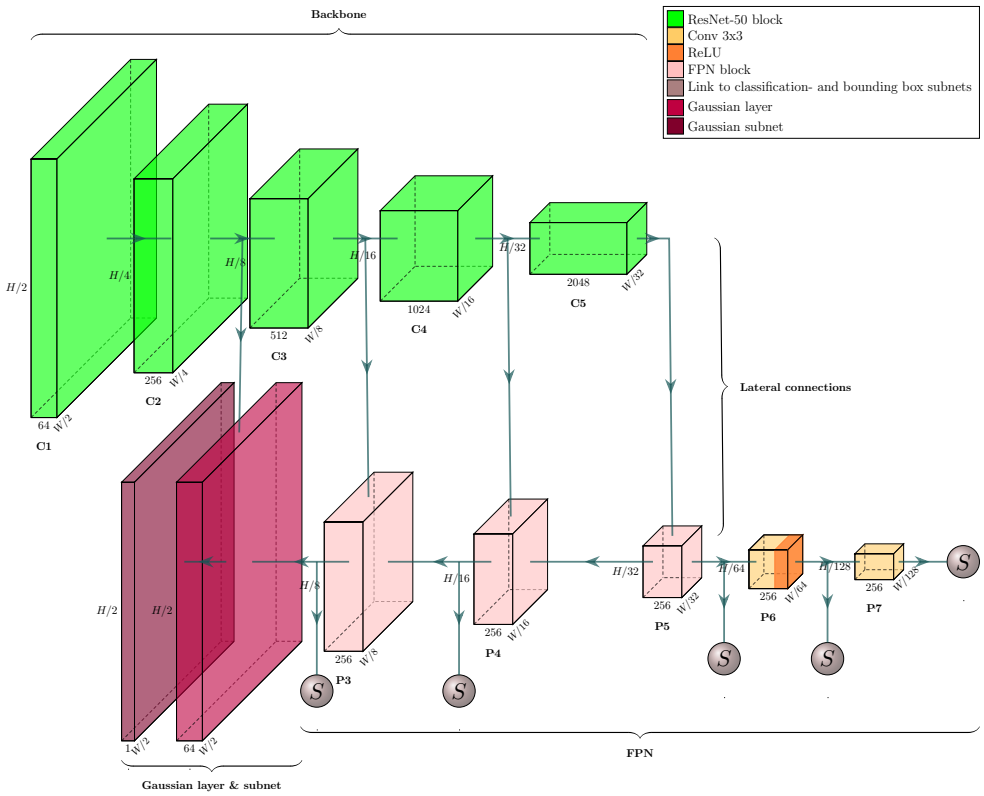


Figure 7. The Gaussian layer network, introduced by Kant [21], as implemented via extending the PyTorch implementation of RetinaNet [29,30]. The major building blocks—the backbone, the feature pyramid network (FPN), and the Gaussian layer and subnet—are annotated with braces, along with the lateral connections. The backbone blocks, labeled C1–C5, are shown in more detail in Figure 3. The FPN blocks are illustrated in Figure 4a. They are labeled P3–P7. P5 receives only lateral connections, i.e., its top-down input consists of just zeroes. P6 and P7 receive only top-down connections; they are therefore implemented as convolution layers with a stride of 2 instead of full-blown FPN blocks. The links, drawn as balls marked with *S*, all lead to the same pair of classification and bounding box regression subnets, shown in Figure 5. The Gaussian layer is illustrated in Figure 4b, and Figure 6 displays the structure of the Gaussian subnet.

Finally, $\mathcal{L}_{\text{MSE}}$ is the mean-squared error. It is used as the loss for our multi-task learning part—the loss between our generated Gaussian maps and ground-truth Gaussian maps. It is calculated as

$$\mathcal{L}_{MSE}(x, \hat{x}) = (x - \hat{x})^2, \tag{6}$$

where x is the ground-truth Gaussian map, and $\hat{x}$ is the prediction. The Gaussian loss is only calculated for coordinates where x is either 0 or larger than a hyperparameter, ω [21].

α_{focal} and α_{MSE} are weighting parameters for focal loss and MSE, respectively. We use them to tune the loss function in order to achieve more stable training and better results.

2.2. Product Classification

We use an embedding approach, the *domain invariant hierarchical embedding* (DIHE, Ref. [23]), as our product classification model. It consists of three networks: the *encoder*, the *generator*, and the *discriminator*.

The encoder network learns an embedding, and a function from images into vectors, with the distance between two output vectors signifying the similarity between the respective images. This embedding can be used to classify images via nearest neighbor search: the query image is converted into a vector with the encoder, and it is assigned a class based on the nearest embedding of a reference image. The encoder is the only network that is used beyond the training stage.

The generator and discriminator are only used to augment the data when training the encoder. The purpose of this is to overcome the lack of data inherent with retail product image sets and to combat the problems caused by *domain shift*, that is, the training images being clear and iconic while the test images are noisy crops from an image of a retail store shelf.

We selected the VGG16 [31] as the encoder network pre-trained with ImageNet, following the example of [23]. We use *maximum activation of convolutions* (MACs)—the maximum value produced by select convolution layers—as the output of our encoder network, due to [23] achieving their best results with this approach. The encoder architecture is visualized in Figure 8.

For the generator and discriminator, we adopt the pix2pix GAN, Ref. [32] again, following the example of [23]. The GAN architecture is visualized in Figures 9–11.

Keeping with the methodology of [23], we use *domain invariant hierarchical embedding loss* ($\mathcal{L}_{\text{DIHE}}$) to train our encoder:

$$\mathcal{L}_{\text{DIHE}}(i_p, i_n, h_p, h_n) = \mathcal{L}_{\text{triplet}}(G(i_p), i_p, i_n, \alpha_{\text{hierarchial}}(h_p, h_n)), \tag{7}$$

where i_p is an image belonging to the class of interest, i_n an image belonging to a different (randomly drawn) class, h_p the product hierarchy of i_p, h_n the product hierarchy of i_n, G the generator network of the GAN, and $\alpha_{\text{hierarchial}}$ a function for calculating the triplet loss marginal. $\alpha_{\text{hierarchial}}$ is specified as

$$\alpha_{\text{hierarchial}}(h_p, h_n) = \alpha_{\text{min}} + (1 - \frac{|h_p \cap h_n|}{|h_p|}) \cdot (\alpha_{\text{max}} - \alpha_{\text{min}}), \tag{8}$$

where α_{min} and α_{max} are hyperparameters that define the minimum and maximum values of $\alpha_{\text{hierarchial}}$.

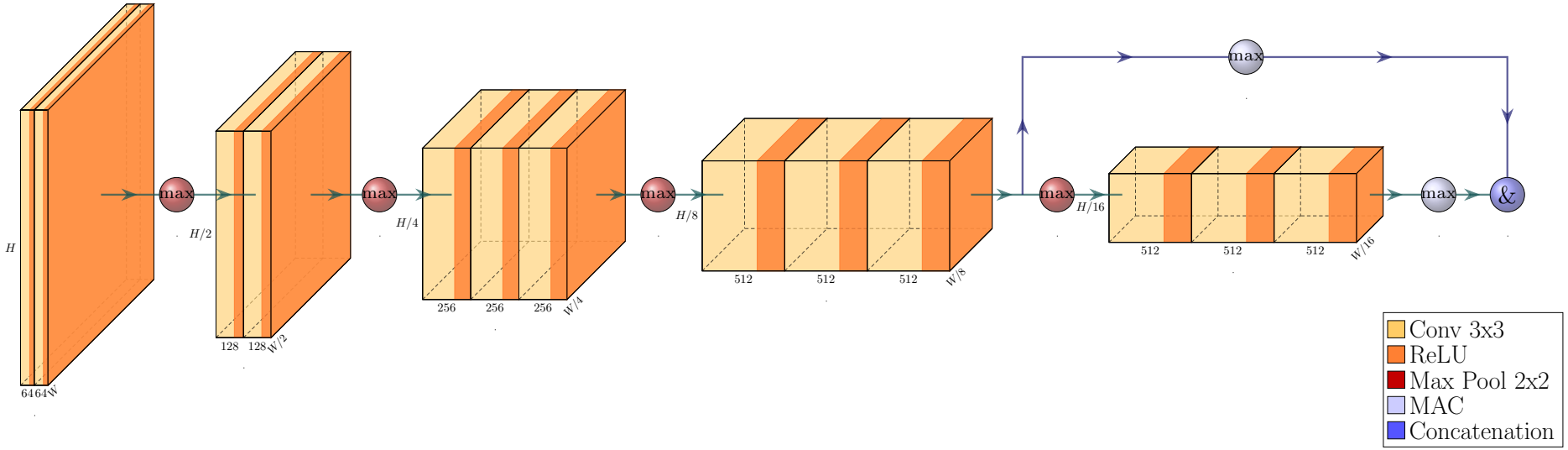
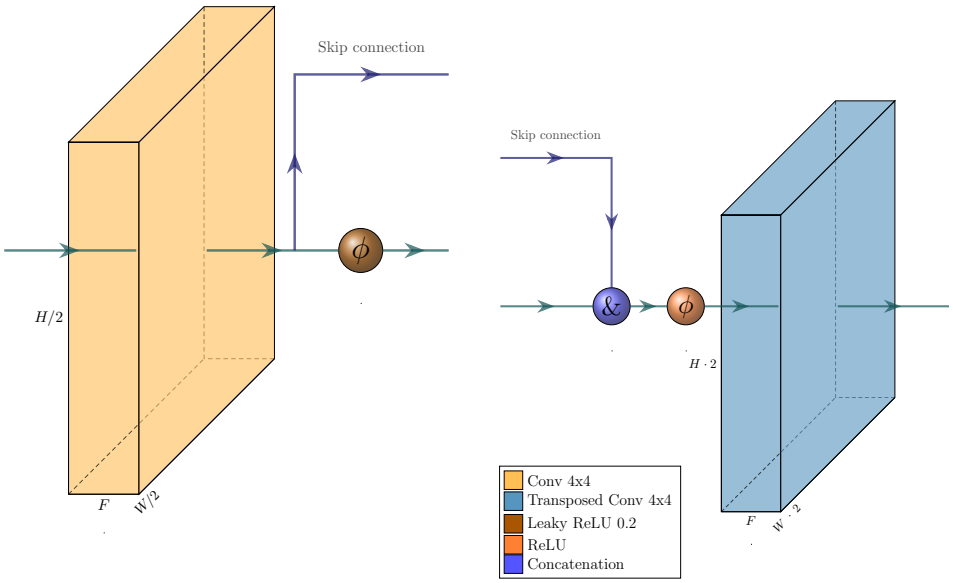


Figure 8. VGG16 encoder utilizing MAC descriptors, built from the PyTorch implementation of said network [29,31,33]. The input is an image with shape $3 \times H \times W$. The model outputs a 1024-element MAC descriptor, with the first 512 elements determined by the maximum activations of the last layer of the second-to-last convolutional block, and the remaining 512 elements determined by the maximum activations of the last convolution layer. Downsampling is implemented via stride 2 when max pooling.



(**a**) Downsampling block. (**b**) Upsampling block.

Figure 9. Down- and upsampling blocks of the PyTorch implementation of UNet, given by Isola et al. [32]. The input's shape is in both cases $A \times H \times W$, where the number of input features, A, varies depending on the location of the block in the network. Most commonly, A is $F/2$ for down- and F for upsampling blocks. Both down- and upsampling are implemented as a stride of 2 in the convolution and transposed convolution layers. The output shapes are $F \times H/2 \times W/2$ and $F \times H \cdot 2 \times W \cdot 2$, respectively. Both of the outputs of the downsampling block and both of the inputs of the upsampling block are always of the same shape.

The loss is based on the standard triplet ranking loss $\mathcal{L}_{\text{triplet}}$ [34]. Given an encoder network E, a desired marginal α, a distance function $d(a, b)$, and three images—the anchor image i_a; the positive image i_p, which belongs to the same class as the anchor image; and the negative image i_n, which belongs to a different class from the anchor image—a triplet ranking loss ($\mathcal{L}_{\text{triplet}}$) may be calculated as

$$\mathcal{L}_{\text{triplet}}(i_a, i_p, i_n, \alpha) = \max(0, \alpha + d(E(i_a), E(i_p)) - d(E(i_a), E(i_n))) . \tag{9}$$

The standard triplet ranking loss expects multiple training samples for each class. In the product detection problem setting, however, we have only one image per class. Therefore, to transform $\mathcal{L}_{\text{triplet}}$ into $\mathcal{L}_{\text{DIHE}}$ in order to overcome the domain gap between the training and test samples, we sample only the positive and negative images from the training set. The anchors are in turn generated by the generator network, G in our GAN, by passing the positive images through it and thus producing images that resemble the test samples that the network will encounter in real images.

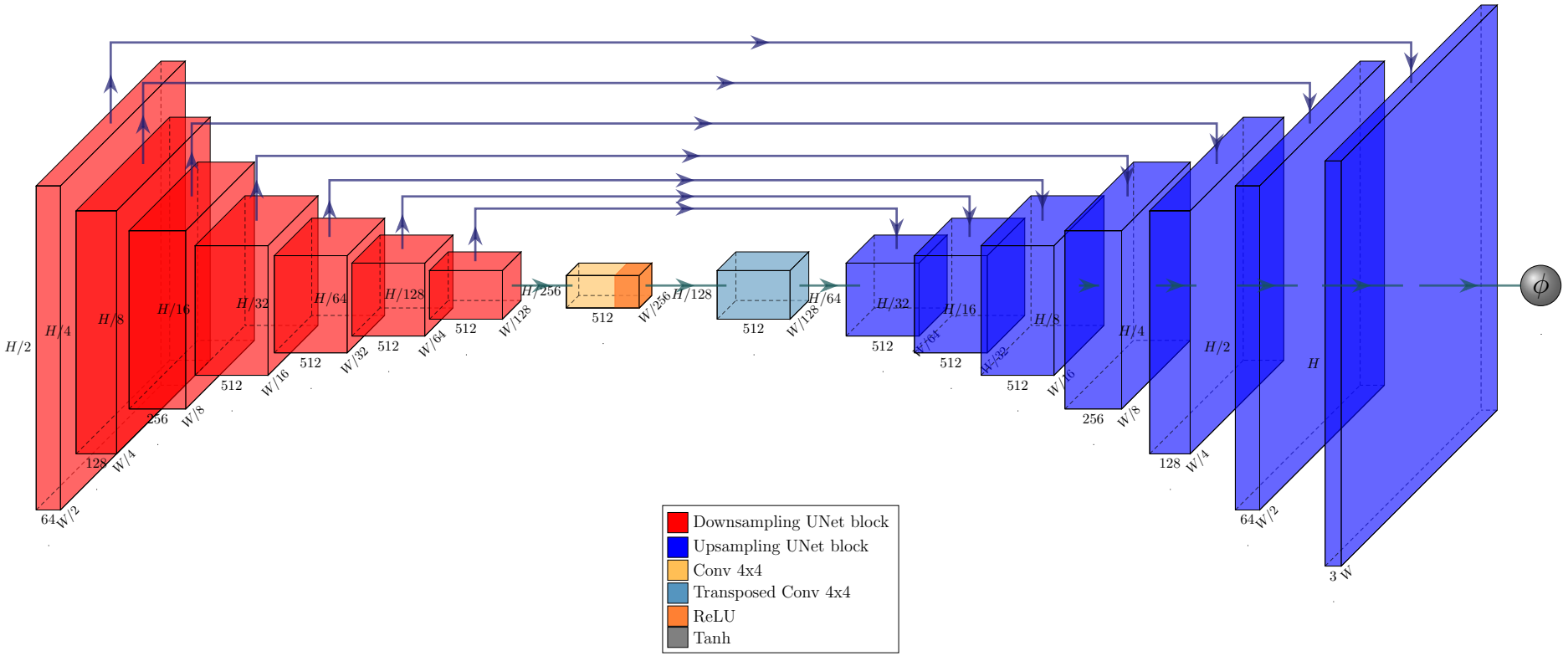


Figure 10. The PyTorch implementation of an UNet generator given by Isola et al. [32]. The input is an image with shape $C \times H \times W$, where C is 4 if our proposed product masks are used and 3 otherwise. The model outputs a modified image with shape $3 \times H \times W$. The up- and downsampling blocks are shown in more detail in Figure 9. The blue arrows show skip connections from the downsampling blocks to the upsampling blocks; the regular connections are shown in a left-to-right manner.

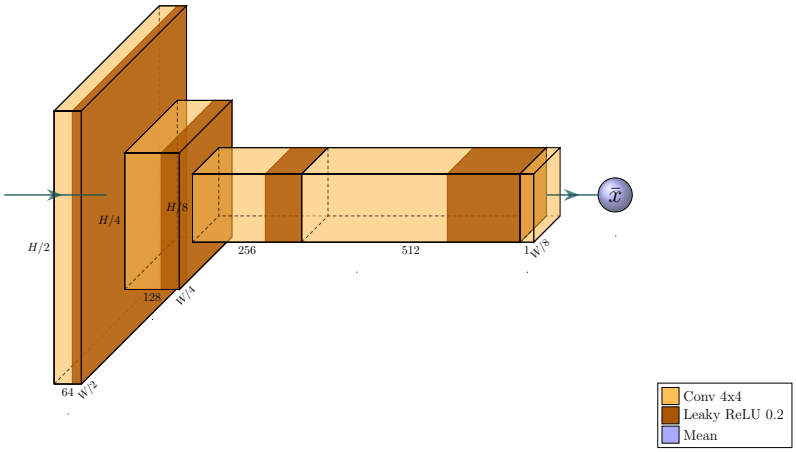


Figure 11. The PatchGAN discriminator given in the official PyTorch implementation of Isola et al. [32]. The input is an image, with shape $3 \times H \times W$. Downsampling is implemented via a stride of 2 in all but the two final convolution layers. The output is a scalar: the confidence that the input image is real, that is, not generated by the generator. This scalar is calculated by averaging the elements of the last, $1 \times H/8 \times W/8$ feature map.

Again, following [23], we train the discriminator of the GAN with the standard cross-entropy loss,

$$\mathcal{L}_{\text{discriminator}} = \log D(i_s) + \log(1 - D(G(i_p))) \,. \tag{10}$$

In this definition, D is the discriminator, G the generator, i_s an image sampled from the target domain, and i_p an image from the source domain. To train the generator, we use a weighted sum of three loss functions:

$$\mathcal{L}_{\text{generator}} = \mathcal{L}_{\text{adversarial}} + \mathcal{L}_{\text{regularization}} + \lambda \mathcal{L}_{\text{embedding}} \,. \tag{11}$$

In this loss function, $\mathcal{L}_{\text{adversarial}}$ is the standard adversarial loss,

$$\mathcal{L}_{\text{adversarial}} = -\log D(G(i_p)) \,. \tag{12}$$

$\mathcal{L}_{\text{regularization}}$ is a regularization term that ensures that the generators output does not diverge too much from its input,

$$\mathcal{L}_{\text{regularization}} = -NCC(i_p, G(i_p)) \,, \tag{13}$$

where NCC is the zero-mean normalized cross correlation discussed by, among others, Ref. [35]. $\mathcal{L}_{\text{embedding}}$ is a term that rewards the generator for creating hard-to-embed products, thus ultimately strengthening the encoder performance,

$$\mathcal{L}_{\text{embedding}} = -d(E(G(i_p)), E(i_p)) \,, \tag{14}$$

where d is a distance function, E the encoder, G the generator, and i_p a positive training image. λ is a hyperparameter used to tune the training process.

2.3. Planogram Compliance Evaluation

Initially, we tested a maximum common subgraph-based approach [13] for the planogram compliance evaluation. However, while this approach works well for images that contain only a few products, such as in the GP-180 dataset, it is computationally infeasible for the more realistic scenario where images span whole shelves. This is due to the computational complexity of the algorithm proposed by Tonioni and di Stefano [13]. If we have a reference planogram with at most r instances of a single product and an observed planogram with at most o instances of a single product, and if the intersection of the

products in these two planograms includes N products, the proposed *CreateHypotheses* routine creates $O(P^2N)$ hypotheses where $P = \max(r, o)$. With a hypothesis set this big, assuming that finding the best hypothesis always requires only constant time (which is quite optimistic) and that each product is resolved completely before moving on to the next product, *FindSolution* requires $N\sum_1^P P^2 = O(P^3N)$ iterations to clear the hypotheses related to the first product. As the first product is now cleared, the next product requires $O(P^3(N-1))$ iterations and the product after that $O(P^3(N-2))$ iterations all the way to $O(P^3)$ iterations for the final product; therefore, the full computational complexity of the first *FindSolution* run is $O(P^3\sum_1^N N) = O(P^3N^2)$. This is then repeated for each hypothesis (of which there are $O(P^2N)$) with a slowly shrinking input set. It is therefore obvious that the proposed algorithm of Tonioni and di Stefano [13] is fine for a dataset such as the planograms in GP-180 introduced by them, where the largest values for P and N are 6 and 9, respectively, but that it is not suitable for full shelf planograms with tens of different products and tens of facings for each product.

To overcome this prohibitive complexity, we started by calculating an arbitrary common subgraph instead of a maximal one and then utilizing RANSAC pose estimation [26] for determining the presence of the majority of products. We further evolved this approach with the realization that, in fact, the whole subgraph comparison is unnecessary —RANSAC is computationally performant and very resistant to outliers, so we can instead just match *all* detected products of some class to all expected products of the matching class.

Ultimately, our planogram compliance evaluation approach is as follows: We take as input the bounding boxes and classes of products in the reference planogram and the bounding boxes and classes of products detected in the image under evaluation. Then, we calculate a Cartesian product of planogram bounding boxes and detect bounding boxes for which the classification matches. Out of each pairing generated by the Cartesian product, we create nine pointwise matches: one match between each matching corner of the planogram and the detected bounding box, one match between each matching midpoint of a side of the planogram and the detected bounding box, and one match between the centers of the bounding boxes. We feed these pointwise matches to a RANSAC algorithm, and get an estimated pose as an output.

We utilize the estimated pose to transform the reference planogram's bounding boxes to image coordinates. Then, we check whether each bounding box in the reference planogram has a good enough matching detection in the image by determining whether there is a detected bounding box with a matching class and an IoU between the boxes larger than 0.5. Finally, inspired by previous work [13], we try to reclassify the part of the image under each projected planogram bounding box for which we could not find a match. We determine that each product in the planogram for which a matching detection could be found immediately or that could be determined as present by reclassification are present in the image. Products in the planogram that were not determined as present after these two steps are deemed as not present.

3. Experiments and Analysis of the Results

3.1. Datasets

We utilize the SKU-110K product proposal generation dataset [21] for training and testing the product proposal generator, and the GP-180 product detection dataset [13] for training and testing the classifier. We select these datasets due to the fact that they are the most suitable openly available datasets for the product proposal generation and product classification problems, respectively. The SKU-110K dataset provides 11,762 images with more than 1.7 million annotated bounding boxes and GP-180 instance-level annotations of 74 rack images. For more details on these datasets, the reader is encouraged to refer to the original papers.

The only public dataset for planogram compliance evaluation that we know of is included in GP-180. However, this dataset is problematic in several ways. First, the planograms in GP-180 consist only of information on the position of products rel-

evant to each other instead of Cartesian coordinates in reference to the shelf—for example, a planogram might say that Product A should be above Product B, and that Product C should be on the right side of product B. This means that in order to use the proposed RANSAC matching, we would first need to somehow infer the Cartesian coordinates of the products based on these relative positions, which is prone to error. Second, in GP-180, all images are fully planogram compliant—a predictor that consistently gives 1 as the planogram compliance would achieve perfect accuracy with GP-180. Third, as already outlined in Section 2.3, the planograms in GP-180 encompass only small fractions of whole shelves. This makes the planograms both unrealistic and enables solving the problem with methods that are unfeasible with a realistic amount of facings and different products per image. The smallness of planograms in GP-180 also makes it unsuitable for our RANSAC-based matching, which benefits from a larger number and diversity of products that can be found in real whole-shelf images.

Due to these limitations, and in order to test the performance of our approach with actual production data, we collected an internal dataset with the help of a large retailer. The dataset consists of five test pictures encompassing whole racks in the vein of SKU-110K, corresponding planograms and iconic images of 7290 products. The test pictures were not annotated, but their planogram compliance was known.

For each of the products, there could be up to six images for the six facings (front, back, top, bottom, left, and right) and additional images for different merchandising types such as product trays. Due to this, there were a total of 27,204 product images, on average 3.7 images per product.

The product image data also included a four-level product group hierarchy in the vein of GP-180. We extended this hierarchy with two additional levels: the facing of the product and the merchandising type.

3.2. Metrics

For evaluating product proposal generation and product classification performance, we use the industry standard metrics. Our product proposal generation metrics include mean average precision (AP) calculated at intersection of union (IoU) of 0.50 and 0.75 [36], the mean of APs calculated at IoUs of 0.50 to 0.95 in steps of 0.05 (AP_{COCO}) [8], and average recall (AR) with 300 top detections [8]. The product classification metric we use is the recall at one and five top classifications.

To the best of our knowledge, there do not seem to be any established evaluation metrics for planogram compliance evaluation performance despite prior work on the subject. The existing research seems to assume that the test image perfectly matches the planogram and to then just calculate standard object detection performance metrics based on how many of the facings were detected.

We argue that for a real-world use case, having test images that do not fully match the planogram is more interesting than having only one-hundred-percent compliant images. Therefore, using the planogram as ground truth and evaluating product detection performance based on it—as has been often carried out by prior work—is not sufficient. Due to this, we propose a new metric, the *normalized planogram compliance error* E_{PC}.

Given the set of facings expected by the planogram E, and the set of facings that are actually present in their expected positions in the test image $F \subseteq E$, we can calculate *planogram compliance* PC simply as

$$PC(F, E) = \frac{|F|}{|E|}. \tag{15}$$

Let $D \subseteq E$ be the set of facings that were detected by the planogram compliance evaluation model to be in their expected positions, and $G \subseteq E$ the set of facings that are truly in their correct positions in the image, where G is determined via manual annotation. Utilising these, we can calculate the detected planogram compliance $PC(D, E)$ and the

ground-truth planogram compliance $PC(G, E)$. Further, we can set the *planogram compliance error* PE as

$$PE(D, G, E) = PC(D, E) - PC(G, E)\,. \tag{16}$$

The planogram compliance error is a value between -1 and 1, where the negative values signify underestimation by the model and vice versa. To further make the metric one-sided, we can calculate the square of planogram compliance error.

By calculating the squared planogram compliance error for all of our test images and then calculating the mean of all the squared planogram compliance errors, we arrive at the *mean-squared planogram compliance error*, MSE_{PC},

$$MSE_{PC}(R) = \frac{1}{|R|} \sum_{(D,G,E)\in R} PE(D, G, E)^2\,, \tag{17}$$

where R is a set of tuples consisting of detected facings, ground truths, and planograms for each of the test images.

The MSE_{PC} would otherwise be a good metric as it is, but its scale is somewhat awkward. Let us say we have two independent variables X and Y, both of which are uniformly random between 0 and 1. The variable X could represent a randomly guessed planogram compliance, while the variable Y represents the ground-truth planogram compliance of some random image. The expected value of the squared error between these two variables—in our example, that is, the squared planogram compliance error—is

$$\begin{aligned} E[(X - Y)^2] &= E[X^2 - 2XY + Y^2] \\ &= E[X^2] - 2E[X]E[Y] + E[Y^2]\,. \end{aligned} \tag{18}$$

The expected value of a random variable Z that is uniformly random between 0 and 1 is $E[Z] = \frac{1}{2}$, and the expected value of its square $E[Z^2] = \frac{1}{3}$. Plugging these into Equation (18), we obtain

$$\begin{aligned} E[(X - Y)^2] &= E[X^2] - 2E[X]E[Y] + E[Y^2] \\ &= \frac{1}{3} - 2 \cdot \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{3} \\ &= \frac{1}{6}\,. \end{aligned} \tag{19}$$

Therefore, assuming that the ground-truth planogram compliances in our dataset are uniformly distributed between 0 and 1, the expected MSE_{PC} is $\frac{1}{6}$ if we just randomly guess real numbers between 0 and 1 as our detected planogram compliances. The MSE_{PC} is also of similarly small magnitude with more realistic ground-truth compliance distributions: if the ground-truth compliance of all the test images is 0.8, the MSE_{PC} we expect to achieve by randomly guessing is 0.173, and for 100% compliant test images—the worst case for random guessing—the expected MSE_{PC} is $\frac{1}{3}$.

An error metric with a value smaller than $\frac{1}{3}$ sounds quite low, yet as shown, such a low MSE_{PC} can be achieved by just random guessing. Therefore, we arrive at our proposed error metric, the normalized planogram compliance error E_{PC}, by normalizing MSE_{PC} with a dataset specific factor $E_{\text{random}}[MSE_{PC}]$. The factor can be calculated with

$$\begin{aligned} E_{\text{random}}[MSE_{PC}] &= \frac{1}{3} - 2 \cdot \frac{1}{2} \cdot E[PC(G, E)] + E[PC(G, E)^2] \\ &= E[PC(G, E)^2] - E[PC(G, E)] + \frac{1}{3}\,, \end{aligned} \tag{20}$$

where $E[PC(G, E)]$ and $E[PC(G, E)^2]$ are the expected values of the ground-truth planogram compliance and its square, respectively. These can be calculated via the mean of all ground-truth planogram compliances in the dataset ($E[PC(G, E)]$) and the mean of squared ground-

truth planogram compliances in the dataset ($E[PC(G,E)^2]$). Using this factor, we arrive at E_{PC}:

$$E_{PC}(R) = \frac{MSE_{PC}(R)}{E_{\text{random}}[MSE_{PC}]}. \quad (21)$$

The normalized planogram compliance error gets a value of 0 with perfect performance, a value of 1 when randomly guessing and a value higher than 1 when performance is worse than expected via random guessing.

The $E_{\text{random}}[MSE_{PC}]$ of our internal dataset is 0.131.

3.3. Implementation Details

Our implementation was built in Python using the PyTorch library [29]. We extended the library's pre-built models with our own as follows. We built our Gaussian layer network implementation by adding the Gaussian layer and subnet, and an additional term for Gaussian loss, to the RetinaNet implementation included in PyTorch. The weights of the additional layers are initialized with Kaiming normal in the case of rectified linear unit activation, or Xavier normal in the case of our proposed hyperbolic tangent activation. A backbone pre-trained with ImageNet is included by PyTorch, and the RetinaNet implementation includes initialization for the FPN and bounding box and classification subnets.

We set the hyperparameters of focal loss to $\alpha = 0.25, \gamma = 2$ following the example set by [30]. The product proposal generator was trained with the training subset of the SKU-110K dataset. We left out a total of 19 training samples due to them being either corrupted or poorly annotated. We determined the ω of Gaussian loss and a per-epoch learning rate multiplier via hyperparameter optimization. The optimization arrived at $\omega \approx 0.6$, and at 0.995 for the learning rate multiplier. Finally, we changed the last activation of the Gaussian subnet to hyperbolic tangent activation, which required us to scale ω to 0.3. Despite switching the value of ω to a less optimal one, the resulting setup provided best performance.

We trained each product proposal generator model for 200 epochs on four NVidia Tesla V100 GPUs with a batch size of 24 (6 per GPU). We used a stochastic gradient descend optimizer with initial learning rate 0.0025, momentum 0.9, and decay 0.0001. These hyperparameters were inherited from [30], except for the learning rate, which we set to one-quarter to stabilize the training. The models were evaluated every three epochs with the validation subset of SKU-110K, and the model that achieved the greatest AP at IoU threshold 0.75 was kept as the result of the training.

We used modified PyTorch implementations of Pix2Pix UNet generator and PatchGAN discriminator [32] for our DIHE classifier's GAN, by adding an averaging operation to the output of the PatchGAN. We built the encoder part of DIHE from the PyTorch implementation of VGG16 [31] by intercepting the results from relevant layers and calculating MAC features from them. The base VGG16 was pre-trained with ImageNet.

The various loss-related hyperparameters of the classifier were set as $\alpha_{\min} = 0.05$, $\alpha_{\max} = 0.5$ and $\lambda = 0.1$ [23].

We used hyperparameter optimization to determine whether to use batch normalization in the classifier's VGG16 model and to set a learning rate and a per-epoch learning rate multiplier. The optimization arrived at no batch normalization, an initial learning rate of approximately 10^{-6} (which matches the learning rate given by [23]), and a per-epoch learning rate multiplier of approximately 0.7. As the optimization was performed on a single GPU, we used a per-epoch multiplier of $\sqrt[4]{0.7} \approx 0.9$ in the actual training to achieve a similar per-iteration learning rate multiplier, due to four GPUs resulting in four times fewer iterations per epoch. This adjustment was more important for DIHE than for GLN due to the generally very small amount of iterations per epoch.

Before training the actual encoder, the GAN was pre-trained for 100 epochs on a single NVidia Tesla V100 with a batch size of 64. GP-180 was used as the data for the generator, while cropped product images from the training subset of SKU-110K were utilized in

training the discriminator. Adam optimizers were used for both components of the GAN, with learning rates set to 10^{-5}.

The encoder models were trained jointly with the GAN for 50 epochs after GAN pre-training on four Tesla V100 GPUs. We trained encoder models with both GP-180 and our internal dataset for a total of two models. The optimizers of the GAN used the same parameters as they did during the pre-training, and an Adam optimizer was used for the encoder. The encoder was evaluated every epoch with a validation split from GP-180, regardless of which dataset was used for training. This was due to our internal dataset not containing test annotations. The model with the highest validation classification accuracy was kept as the result of the training.

We utilized the homography estimation of OpenCV [37] for the planogram compliance-related pose estimation. The RANSAC reprojection threshold was set to one percent of the smaller dimension of the input image. We considered a projected planogram bounding box to match a detection if the IoU of the boxes was over 0.5.

The implementation is available at https://github.com/laitalaj/cvpce, accessed on 25 July 2023.

3.4. Results

As evident from Table 1, hyperparameter optimization gives significant improvements with the SKU-110K dataset to both the average precision and the average recall—1.8 and 4.4 percentage points, respectively—and our hyperbolic tangent activation in the Gaussian subnet gives an additional average precision boost of 0.7 percentage points for a total advantage of 2.5 percentage points over [21].

As shown in Tables 2 and 3, the performance of our models with the GP-180 dataset does not get anywhere near the performance of [21]. The suspected reason for poor GP-180 performance is differences in implementation: notably, we did not include anything like random crops or resizes of the training data in our training procedure. Nevertheless, we do not consider the sub-par performance of the model with the GP-180 set too big of a problem, as the test images in the set do not really correspond to real-life planogram compliance evaluation cases.

Table 1. Product proposal generation performance with the SKU-110K dataset of Goldman et al. [20]. The IoU threshold used for AP is given as a subscript; AP_{COCO} is average precision calculated with and averaged over COCO IoU thresholds. The best results for each metric are highlighted in bold.

Method	AP_{COCO}	$AP_{0.50}$	$AP_{0.75}$	AR_{300}
[21]	0.521	**0.891**	0.562	0.569
Our GLN, optimized hyperparameters	0.539	0.874	**0.571**	**0.613**
+ Tanh activation	**0.546**	0.874	**0.571**	0.612

Table 2. Product proposal generation performance with classless annotations of the GP-180 dataset, given by Varadarajan et al. [38]. The IoU threshold used for AP is given as a subscript; AP_{COCO} is average precision calculated with and averaged over COCO IoU thresholds. The best results for each metric are highlighted in bold.

Method	AP_{COCO}	$AP_{0.50}$	$AP_{0.75}$	AR_{300}
[21]	**0.506**	**0.862**	**0.548**	**0.634**
Our GLN, optimized hyperparameters	0.303	0.594	0.263	0.476
+ Tanh activation	0.296	0.569	0.252	0.417

Table 3. Product proposal generation performance with the GP-180 dataset [13]. The IoU threshold used for AP is given as a subscript; AP_{COCO} is average precision calculated with and averaged over COCO IoU thresholds. The best results for each metric are highlighted in bold.

Method	AP_{COCO}	$AP_{0.50}$	$AP_{0.75}$	AR_{300}
Our GLN, optimized hyperparameters	**0.287**	**0.537**	**0.271**	**0.505**
+ Tanh activation	0.233	0.469	0.202	0.415

As the two product proposal generator models are slightly better than each other in different areas—the one with hyperbolic tangent activation with SKU-110K-like data, and the one with rectified linear unit activation with Grocery Products data—we kept both of the models in the loop when validating product detection and planogram compliance evaluation performance.

During our DIHE implementation process, we noticed that achieving the classification accuracy reported by [23] is quite challenging. The DIHE training starts to overfit quite easily, and the best validation set performance is often achieved after just a few epochs of training. In addition to this, the out-of-the-box, no fine-tuning performance of the pre-trained VGG16 provided by PyTorch is five percentage points worse than that of the Tensorflow weights used by [23], meaning that more fine-tuning is necessary to achieve the same accuracy with PyTorch. Both of these facts are shown in Table 4.

Table 4. Product classification performance with the GP-180 dataset of [13]. "GP-180 all" refers to all of the test annotations in GP-180, while "GP-180 test" refers to our test split. K refers to the number of nearest neighbors used to determine classification correctness, $K = 1$ means a classification is considered correct if the correct class is the nearest neighbor of the query, while $K = 5$ means that a classification is considered correct if the set of five nearest neighbors of the query contains the correct class. Both the results achieved by [23] and the best results achieved by us are highlighted in bold.

		GP-180 All		GP-180 Test		Highest Acc.
Method		K = 1	K = 5	K = 1	K = 5	at Epoch
[23]	Untrained	0.787	–	–	–	–
	Best	**0.842**	**0.942**	–	–	–
Our DIHE, untrained		0.727	0.882	0.730	0.878	–
Our DIHE, trained with GP-180		0.809	**0.936**	0.811	0.930	9
Our DIHE, trained with internal data		**0.812**	**0.936**	**0.813**	**0.931**	3

Despite these hurdles, the DIHE trained with our internal dataset almost reached the performance of [23], with nearest neighbor classification performance 3 percentage points behind their results. When considering a classification correct when the correct class is contained within the five nearest neighbors of the query image, our classification performance got even closer to that of [23], the gap being only 0.6 percentage points. It could be argued that these accuracies are a bit skewed in our favor, as we used a part of the whole GP-180 test set as a validation set;however, as seen in Table 4, the performance using only the part of the data unseen during the validation is approximately the same as—and often even slightly better than—the full-set performance.

We used a DIHE model from each of the respective datasets in our further planogram compliance evaluation performance tests. We kept the model trained with Grocery Products as a part of our tests despite the one trained with our internal dataset outperforming it even with GP-180.

The quantitative planogram compliance evaluation performance of our approach is somewhat underwhelming, as shown in Table 5. However, this is mostly due to the classifier performance not being up to par. Figure 12 visualizes the planogram compliance evaluation performance for both the worst- and the best-performing images of our internal dataset using our best model (see Tables 5 and 6). It is evident that even with the worst performing image (Image 1, the top image in the figure) the RANSAC reprojection of the planogram matches the locations of most of the products, yet the classifier does not recognize the products correctly.

Overall, the results show that our approach is a promising option for tackling the difficult automated planogram compliance evaluation problem. However, some further research into the classifier's accuracy is needed in order to achieve an operationally acceptable level of performance.

Table 5. Planogram compliance evaluation performance with the internal dataset. The "Generator" column refers to the GLN model used—the "original" with rectified linear unit (ReLU) activation or our proposed modification with hyperbolic tangent (Tanh) activation. The "Classifier" column refers to the dataset that was used to train the classifier—either grocery products (GP) or our internal dataset. The columns "I1" to "I5" refer to the test images in the internal dataset, and the values given are planogram compliance errors. Each value is the mean of 100 runs, and the standard deviation is given in parenthesis. Best values for each image and the best E_{PC} are highlighted in bold.

Generator	Classifier	I1	I2	I3	I4	I5	E_{PC}
ReLU	GP	−0.626 (0.013)	−0.183 (0.041)	−0.415 (0.018)	**−0.285 (0.019)**	−0.506 (0.019)	1.434 (0.057)
	Internal	**−0.615 (0.014)**	−0.227 (0.051)	−0.406 (0.015)	−0.293 (0.014)	**−0.454 (0.018)**	1.362 (0.060)
Tanh	GP	−0.622 (0.011)	**−0.169 (0.038)**	−0.407 (0.022)	−0.293 (0.019)	−0.516 (0.019)	1.430
	Internal	−0.616 (0.015)	−0.201 (0.055)	**−0.405 (0.021)**	−0.292 (0.015)	**−0.454 (0.020)**	**1.346 (0.061)**

Table 6. Planogram compliance evaluation statistics for each image in the internal dataset, using the model with the lowest E_{PC} (see Table 5). The "Ground truth" and "Detected" values are planogram compliances.

Image	Ground Truth	Detected	PE	PE^2	PE^2 std
1	0.697	0.082	−0.616	0.379	0.018
2	0.812	0.612	−0.201	0.043	0.026
3	0.702	0.296	−0.405	0.165	0.017
4	0.569	0.277	−0.292	0.086	0.009
5	0.733	0.279	−0.454	0.207	0.018

Figure 12. Visualized planogram compliance evaluation performance with internal dataset images 1 (**top**) and 2 (**bottom**). Green boxes signify products originally detected at correct locations, yellow boxes products found after reprojection, and red boxes reprojected boxes that were found not to contain the expected product.

4. Conclusions

In this paper, we presented a novel retail product detection pipeline combining a Gaussian layer network product proposal generator and domain invariant hierarchical embedding (DIHE) for classification and utilized it with RANSAC pose estimation for planogram compliance evaluation. We evaluated the performance of our method with two datasets, with the only open-source dataset with planogram evaluation data, GP-180, and our own dataset collected from a large Nordic retailer. We performed the evaluation with a novel metric for evaluating planogram compliance evaluator performance in real-world setups, E_{PC}.

The only public dataset for planogram compliance evaluation that we know of is GP-180. This dataset is problematic in several ways: the planograms in GP-180 consist only of information on the position of products relevant to each other instead of in reference to the shelf, all images are fully planogram compliant, and it encompasses only small

fractions of whole shelves. This makes the planograms unrealistic. Therefore, we collected our own dataset with the help of a large retailer. The dataset consists of five test pictures encompassing whole racks, corresponding planograms, and iconic images of 7290 products with up to 6 facings each, resulting in 27,204 images in total.

The results showed that our method provided an improved planogram compliance evaluation pipeline, which resulted in accurate estimation solutions when using real-life images that included entire shelves, unlike previous research that has only used images with few products. However, the classification method requires further research to achieve comparable performance using real data as the previous research. Our analysis also demonstrated that our method requires less processing time than the state of the art.

Our approach paves the way for further research that can, eventually, enable utilizing computer vision for the daily in-store planogram operations, thus increasing retail efficiency. The limiting factor of our approach—the product classifier—could be further improved via more sophisticated representation learning methods or by improving the domain adaptation capabilities via substituting the GAN with a latent diffusion model [39], as the latter approach has proven to be superior to GANs in tasks such as image generation and style transfer.

Author Contributions: Conceptualization; methodology; software; validation; formal analysis; writing—original draft preparation, J.L. Resources; writing—review and editing; supervision, L.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by Research Council of Finland's Flagship program: Finnish Center for Artificial Intelligence FCAI. Open access funding provided by University of Helsinki.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: 3rd Party Data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Marder, M.; Harary, S.; Ribak, A.; Tzur, Y.; Alpert, S.; Tzadok, A. Using image analytics to monitor retail store shelves. *IBM J. Res. Dev.* **2015**, *59*, 3:1–3:11. . [CrossRef]
2. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 25 July 2023).
3. Davies, E.; Turk, M.A. *Advanced Methods and Deep Learning in Computer Vision*; Academic Press: Cambridge, MA, USA, 2022. [CrossRef]
4. Lin, T.Y.; Dollár, P.; Girshick, R.B.; He, K.; Hariharan, B.; Belongie, S.J. Feature Pyramid Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: New York, NY, USA, 2017; pp. 936–944. [CrossRef]
5. Girshick, R.B.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, 23–28 June 2014; IEEE Computer Society: New York, NY, USA, 2014; pp. 580–587. [CrossRef]
6. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; IEEE Computer Society: New York, NY, USA, 2016; pp. 779–788. [CrossRef]
7. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.S.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
8. Lin, T.Y.; Maire, M.; Belongie, S.J.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Lecture Notes in Computer Science, Proceedings of the Computer Vision—ECCV 2014—13th European Conference, Zurich, Switzerland, 6–12 September 2014, Proceedings, Part V*; Fleet, D.J., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8693, pp. 740–755. [CrossRef]
9. Santra, B.; Mukherjee, D.P. A comprehensive survey on computer vision based approaches for automatic identification of products in retail store. *Image Vis. Comput.* **2019**, *86*, 45–63. [CrossRef]

10. Merler, M.; Galleguillos, C.; Belongie, S.J. Recognizing Groceries in situ Using in vitro Training Data. In Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MI, USA, 18–23 June 2007; IEEE Computer Society: New York, NY, USA, 2007. [CrossRef]
11. Tonioni, A.; Serra, E.; di Stefano, L. A deep learning pipeline for product recognition on store shelves. In Proceedings of the IEEE International Conference on Image Processing, Applications and Systems, IPAS 2018, Sophia Antipolis, France, 12–14 December 2018; IEEE: New York, NY, USA, 2018; pp. 25–31. [CrossRef]
12. Franco, A.; Maltoni, D.; Papi, S. Grocery product detection and recognition. *Expert Syst. Appl.* **2017**, *81*, 163–176. [CrossRef]
13. Tonioni, A.; di Stefano, L. Product Recognition in Store Shelves as a Sub-Graph Isomorphism Problem. In *Lecture Notes in Computer Science, Proceedings of the Image Analysis and Processing—ICIAP 2017—19th International Conference, Catania, Italy, 11–15 September 2017, Proceedings, Part I*; Battiato, S., Gallo, G., Schettini, R., Stanco, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10484, pp. 682–693. [CrossRef]
14. Ray, A.; Kumar, N.; Shaw, A.; Mukherjee, D.P. U-PC: Unsupervised Planogram Compliance. In *Lecture Notes in Computer Science, Proceedings of the Computer Vision—ECCV 2018—15th European Conference, Munich, Germany, 8–14 September 2018, Proceedings, Part X*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11214, pp. 598–613. [CrossRef]
15. Karlinsky, L.; Shtok, J.; Tzur, Y.; Tzadok, A. Fine-Grained Recognition of Thousands of Object Categories with Single-Example Training. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: New York, NY, USA, 2017; pp. 965–974. [CrossRef]
16. Geng, W.; Han, F.; Lin, J.; Zhu, L.; Bai, J.; Wang, S.; He, L.; Xiao, Q.; Lai, Z. Fine-Grained Grocery Product Recognition by One-Shot Learning. In Proceedings of the 2018 ACM Multimedia Conference on Multimedia Conference, MM 2018, Seoul, Republic of Korea, 22–26 October 2018; Boll, S., Lee, K.M., Luo, J., Zhu, W., Byun, H., Chen, C.W., Lienhart, R., Mei, T., Eds.; ACM: New York, NY, USA, 2018; pp. 1706–1714. [CrossRef]
17. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; IEEE Computer Society: New York, NY, USA, 2016; pp. 770–778. [CrossRef]
18. Santra, B.; Shaw, A.K.; Mukherjee, D.P. Graph-based non-maximal suppression for detecting products on the rack. *Pattern Recognit. Lett.* **2020**, *140*, 73–80. [CrossRef]
19. Qiao, S.; Shen, W.; Qiu, W.; Liu, C.; Yuille, A.L. ScaleNet: Guiding Object Proposal Generation in Supermarkets and Beyond. In Proceedings of the IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, 22–29 October 2017; IEEE Computer Society: New York, NY, USA, 2017; pp. 1809–1818. [CrossRef]
20. Goldman, E.; Herzig, R.; Eisenschtat, A.; Goldberger, J.; Hassner, T. Precise Detection in Densely Packed Scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 16–20 June 2019; Computer Vision Foundation/IEEE: New York, NY, USA, 2019; pp. 5227–5236. [CrossRef]
21. Kant, S. Learning Gaussian Maps for Dense Object Detection. In Proceedings of the 31st British Machine Vision Conference Virtual Event, UK, 7–10 September 2020; The British Machine Vision Association and Society for Pattern Recognition: London, UK, 2020.
22. Pan, X.; Ren, Y.; Sheng, K.; Dong, W.; Yuan, H.; Guo, X.; Ma, C.; Xu, C. Dynamic Refinement Network for Oriented and Densely Packed Object Detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, 13–19 June 2020; IEEE: New York, NY, USA, 2020; pp. 11204–11213. [CrossRef]
23. Tonioni, A.; di Stefano, L. Domain invariant hierarchical embedding for grocery products recognition. *Comput. Vis. Image Underst.* **2019**, *182*, 81–92. [CrossRef]
24. Saran, A.; Hassan, E.; Maurya, A.K. Robust visual analysis for planogram compliance problem. In Proceedings of the 14th IAPR International Conference on Machine Vision Applications, MVA 2015, Miraikan, Tokyo, Japan, 18–22 May 2015; IEEE: New York, NY, USA, 2015; pp. 576–579. [CrossRef]
25. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.E.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Lecture Notes in Computer Science, Proceedings of the Computer Vision—ECCV 2016—14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016, Proceedings, Part I*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9905, pp. 21–37. [CrossRef]
26. Fischler, M.A.; Bolles, R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
27. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.W.; Chen, J.; Liu, X.; Pietikäinen, M. Deep Learning for Generic Object Detection: A Survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [CrossRef]
28. George, M.; Floerkemeier, C. Recognizing Products: A Per-exemplar Multi-label Image Classification Approach. In *Lecture Notes in Computer Science, Proceedings of the Computer Vision—ECCV 2014—13th European Conference, Zurich, Switzerland, 6–12 September 2014, Proceedings, Part II*; Fleet, D.J., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8690, pp. 440–455. [CrossRef]

29. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019; Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E.B., Garnett, R., Eds.; 2019; pp. 8024–8035.
30. Lin, T.Y.; Goyal, P.; Girshick, R.B.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 318–327. [CrossRef] [PubMed]
31. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015; Conference Track Proceedings; Bengio, Y., LeCun, Y., Eds.; 2015.
32. Isola, P.; Zhu, J.Y.; Zhou, T.; Efros, A.A. Image-to-Image Translation with Conditional Adversarial Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: New York, NY, USA, 2017; pp. 5967–5976. [CrossRef]
33. Tolias, G.; Sicre, R.; Jégou, H. Particular object retrieval with integral max-pooling of CNN activations. In Proceedings of the 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2–4 May 2016; Conference Track Proceedings; Bengio, Y., LeCun, Y., Eds.; 2016.
34. Wang, J.; Song, Y.; Leung, T.; Rosenberg, C.; Wang, J.; Philbin, J.; Chen, B.; Wu, Y. Learning Fine-Grained Image Similarity with Deep Ranking. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, 23–28 June 2014; IEEE Computer Society: New York, NY, USA, 2014; pp. 1386–1393. [CrossRef]
35. Briechle, K.; Hanebeck, U.D. Template matching using fast normalized cross correlation. In Proceedings of the Optical Pattern Recognition XII. International Society for Optics and Photonics, Orlando, FL, USA, 19 April 2001; Volume 4387, pp. 95–102.
36. Everingham, M.; Gool, L.V.; Williams, C.K.I.; Winn, J.M.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
37. Bradski, G. The OpenCV Library. *Dr. Dobb'S J. Softw. Tools* **2000**, *25*, 120–123.
38. Varadarajan, S.; Kant, S.; Srivastava, M.M. Benchmark for Generic Product Detection: A Low Data Baseline for Dense Object Detection. In *Lecture Notes in Computer Science, Proceedings of the Image Analysis and Recognition—17th International Conference, ICIAR 2020, Póvoa de Varzim, Portugal, 24–26 June 2020, Proceedings, Part I*; Campilho, A., Karray, F., Wang, Z., Eds.; Springer: Berlin/Heidelberg, Germany, 2020; Volume 12131, pp. 30–41. [CrossRef]
39. Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; Ommer, B. High-Resolution Image Synthesis With Latent Diffusion Models. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 10684–10695.