Joint Training of Product Detection and Recognition Using Task-Specific Datasets

Floris De Feyter[a] and Toon Goedemé[b]
EAVISE—PSI—ESAT, KU Leuven, Sint-Katelijne-Waver, Belgium
{floris.defeyter, toon.goedeme}@kuleuven.be

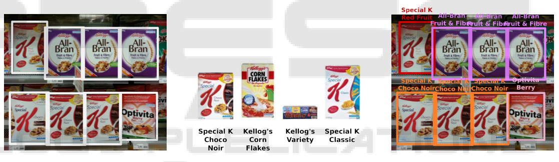
Keywords: Product Detection and Recognition, Joint Detection and Recognition, Task-Specific Training.

Abstract: Training a single model jointly for detection and recognition is typically done with a dataset that is fully annotated, i.e., the annotations consist of boxes with class labels. In the case of retail product detection and recognition, however, developing such a dataset is very expensive due to the large variety of products. It would be much more cost-efficient and scalable if we could employ two *task-specific* datasets: one detection-only and one recognition-only dataset. Unfortunately, experiments indicate a significant drop in performance when trained on task-specific data. Due to the potential cost savings, we are convinced that more research should be done on this matter and, therefore, we propose a set of training procedures that allows us to carefully investigate the differences between training with fully-annotated vs. task-specific data. We demonstrate this on a product detection and recognition dataset and as such reveal one of the core issues that is inherent to task-specific training. We hope that our results will motivate and inspire researchers to further look into the problem of employing task-specific datasets to train joint detection and recognition models.

1 INTRODUCTION

In the retail industry, *planogram compliance* refers to the compliance between the planned layout of a store rack (i.e., the *planogram*) and its true layout. These planograms are the result of negotiations between the retailer and the manufacturers. Sales representatives of the manufacturing companies have the task to regularly verify that the true shelf layout complies with the agreements that were made. This involves taking a photo of the store rack, drawing bounding boxes around each product and annotating each product with a label. Clearly, this is a time-intensive and cumbersome job. A system that could automatically recognize the products that are on the shelves of a supermarket, would make the process of verifying planogram compliance much more efficient.

In general, there are two ways to develop such a system. First, a pipeline consisting of two models could be built: one model (the *detector*) detects where there are products in the image, another model (the *encoder*) extracts an embedding for each product region that can be employed for comparison. This approach is frequently used in facial recognition (Wang et al., 2019), but has also already been proposed for retail product recognition (Tonioni et al., 2018). Second, a *single* model could be trained to jointly perform product detection and recognition. In person search literature (Munjal et al., 2019; Xiao et al., 2017; Xiao et al., 2019), this combination of detection and recognition has been achieved by adding an extra *Region of Interest* (RoI) head to a typical detector network like Faster R-CNN (Ren et al., 2015). This extra head outputs the necessary recognition embedding. Figure 2 shows an example of such a joint network architecture.



(a) (b) (c)

Figure 1: Samples from (a) a detection-only dataset; (b) a recognition-only dataset; (c) a dataset with both detection and recognition annotations.

The advantage of the *two-models* approach is that we can train on two *task-specific datasets* (see Figs. 1 (a) and (b)), i.e., one dataset that contains detection annotations and one dataset that contains labeled single products. This leads to interesting cost reductions in dataset development. Indeed, a recognition dataset containing individual products is read-

[a] https://orcid.org/0000-0003-2690-0181
[b] https://orcid.org/0000-0002-7477-8961

ily available to most retail stores, so only a dataset with bounding boxes (without a class label) would need to be developed. The disadvantage of the two-models approach is that it is very computationally expensive, since all detected products need to be passed through a second model. The *joint* approach does not have such a computational bottleneck. However, previous work has always trained such a model on a *fully-annotated dataset* (see Fig. 1 (c)), i.e., a dataset that contains a product label for each bounding box (Munjal et al., 2019; Ranst et al., 2018; Xiao et al., 2019; Xiao et al., 2017). Such a dataset is much more costly to develop.

One could wonder if, instead, the joint approach could also be trained on task-specific datasets. Surprisingly, however, to the best of our ability, we were unable to find any previous work that concerned this topic. A lot of research has been done on semi-supervised object (SSOD) detection (Fang et al., 2021; Redmon and Farhadi, 2016; Zhou et al., 2022), and while parallels can be drawn, SSOD is clearly different from the approach proposed here. With SSOD, part of the data is fully annotated with bounding boxes *and* class labels, and another part of the data is only annotated on the image level with one or more labels per image (i.e., *weak labeling*). The annotations of the detection dataset should contain class labels that even partly overlap with the labels in the weakly-labeled dataset. In the set-up we wish to investigate, there is a strict boundary between both datasets: one dataset only contains bounding boxes, the other one only contains labeled images of individual products.

Our experiments suggest why we could not find any previous publications on the matter: a joint model trained on task-specific datasets clearly performs worse than the same joint model trained on a fully-annotated dataset. Due to the potential cost savings, however, we are convinced that training on task-specific datasets deserves more exposure in the literature, and that it is worth the effort to look for causes of its lower performance. Therefore, we propose a novel method to carefully evaluate the differences between both training procedures. We first train a joint model on a fully-annotated dataset and gradually apply changes to the training procedure until we end up with training on task-specific datasets. With this framework, any issues that occur during the transformation process can be pointed out and as such provide focus points to work on when attempting to close the performance gap.

2 RELATED WORK

2.1 Product Detection and Recognition

The automated recognition of products in stores is a long-standing problem with many proposed solutions. Many of the early proposed works focus on using RFID-tags, barcodes or QR-codes that are attached to each product (Kulyukin et al., 2005; López-de-Ipiña et al., 2011). Computer vision-based techniques, however, offer a potentially less intrusive and more scalable approach to product recognition. There are two tasks to be performed: *detection*, i.e., finding out *where* there are products in the image; and *recognition* (or, *classification*), i.e., finding out *which* product is present in a product region. While a lot of research has been done on employing classic computer vision techniques for both product detection and product classification (George et al., 2015; Merler et al., 2007; Tonioni and Di Stefano, 2017), most recent work focuses on using deep learning techniques. Solutions have been proposed for both deep learning-based product classification and deep learning-based product detection (Goldman et al., 2019; Qiao et al., 2017; Srivastava, 2020).

Most relevant to our work, however, are the methods that combine product detection and classification (Fuchs et al., 2019; Hao et al., 2019). Often, these models need to be (partly) retrained every time a new product class is added, limiting the scalability of the pipelines. In the product recognition literature, only Tonioni et al. (Tonioni et al., 2018) and Osokin et al. (Osokin et al., 2020) propose a product detection and recognition system that can also be used for products that were not present in the original training dataset. Tonioni et al. use a separate detector and encoder model, where each detected product is cropped out and passed through the encoder. While their results are satisfactory, the two-stage design causes a computational bottleneck at the encoder part of the pipeline (Tonioni et al., 2018). Osokin et al. propose a solution in a one-shot object detection setting (Osokin et al., 2020). Via a pairwise-correlation of the feature maps of the query (rack) image and of each of the gallery images, their model computes geometric transformation parameters that map the gallery images to matching locations in the query image. Adding new products to the gallery only requires the computation of their feature maps once. However, their method needs a training dataset that is annotated with bounding boxes and class labels, which we explicitly wish to avoid.

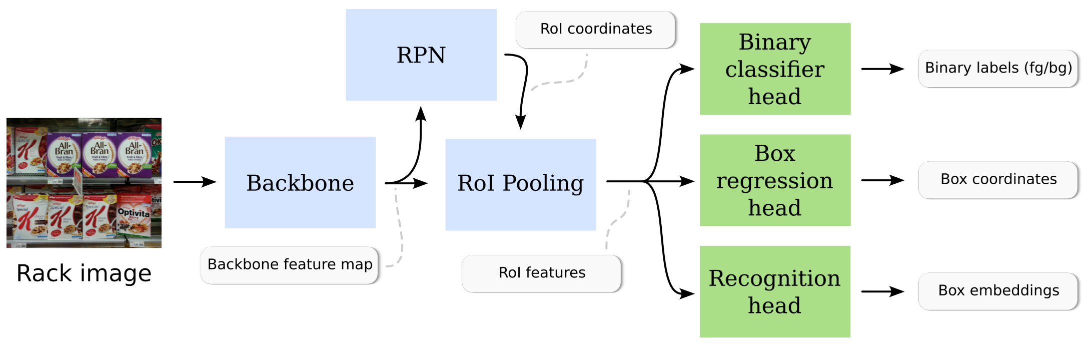


Figure 2: High-level overview of the joint architecture we use in this paper. The outputs of the different components are indicated with the light gray boxes.

2.2 Joint Detection and Recognition

An important property of the model considered in this paper, is that both product detection and product recognition are performed at once. In the area of person search and person re-identification, this has been coined as *joint detection and recognition* (Munjal et al., 2019; Ranst et al., 2018; Xiao et al., 2017; Xiao et al., 2019). All of these methods start from a standard detector (either Faster R-CNN (Ren et al., 2015) or YOLOv2 (Redmon and Farhadi, 2016)) and modify it in such a way that, for each bounding box, an embedding is returned that can be used for the recognition task. Similar architectures can also be found in the field of few-shot object detection (Kang et al., 2019; Wang et al., 2020; Zhou et al., 2022), where the task is to detect and classify an object based on only a *few* (typically < 10) examples. While product detection and recognition certainly is a valid use-case of few-shot object detection (as (Osokin et al., 2020) shows), all these methods require a fully-annotated dataset.

3 METHOD

To identify issues that arise when training a joint model on task-specific datasets, we define four procedures for training the model. The first one, Proc. 1, is simply the well-known multi-class detector training, trained on a fully-annotated dataset. The last one, Proc. 4, describes how one could train a joint architecture on task-specific datasets. Starting from Proc. 1, each next procedure is a slightly modified version of the previous one. When one of the procedures fails while the previous one worked, we have identified an issue that decreases the performance of Proc. 4.

Note that, for all except Proc. 4, we need a fully-annotated dataset. In our experiments, we used the GroZi-3.2k dataset (George and Floerkemeier, 2014) with annotations from Tonioni and Di Stefano (Tonioni and Di Stefano, 2017). This dataset is not large enough for a production-ready model (see Sec. 4 for more details), but it suffices to demonstrate how the procedures below can be applied. Also, note that the forward pass during test time is exactly the same for all the procedures: a store rack image is passed through the model, which returns a set of bounding boxes, binary class labels and recognition embeddings.

Procedure 1 (Conventional Detector Training)**.** *The model is trained with a fully-annotated dataset. A batch of product rack images is passed through the joint architecture and we obtain a set of bounding boxes, binary class labels (foreground/background) and recognition embeddings (along with region proposals and* objectness *scores from the RPN). For each of these outputs, we have an annotated ground truth, so we can compute a loss value and train the network with a gradient descent-like optimization algorithm. Fig. 3 shows an example of how these losses could be computed.*

Procedure 2 (Two-Phase Training)**.** *Unlike Procedure 1, the batch of rack images is passed through the joint architecture* twice. *Weight updates are applied after each training phase separately. During the* ***detection training phase***, *the batch goes through all the network components, except the recognition head. Losses are computed for the RPN, bounding box regression and foreground vs. background classification. During the* ***recognition training phase***, *the batch only goes through the backbone and the recognition head. We use the ground truth bounding boxes to apply RoI pooling on the feature maps that come out of the backbone. During this phase, only the recognition loss is computed. More specifically, the loss is calculated from the ground-truth product label of the bounding box used during RoI pooling. An example of the loss computation during the recognition training phase, is illustrated in Fig. 4.*

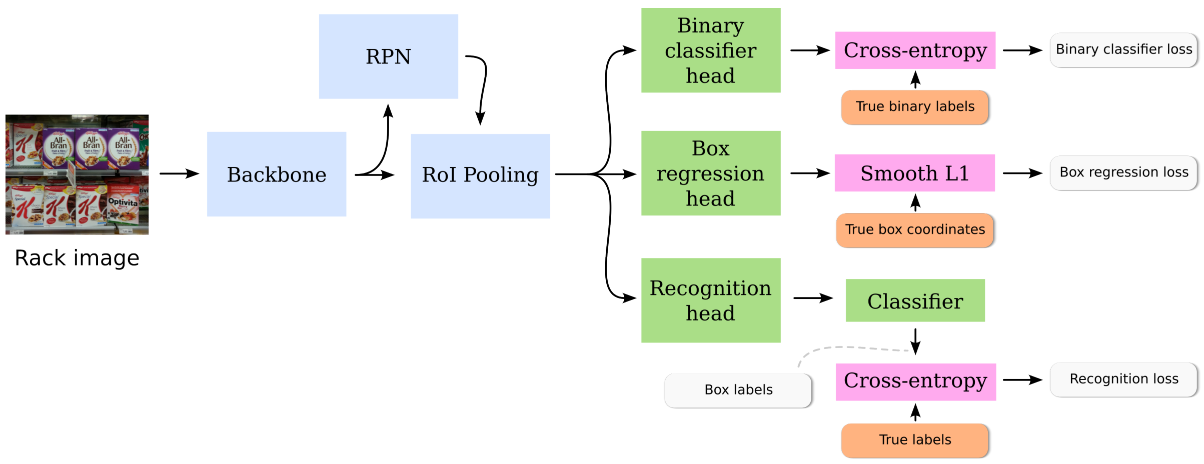


Figure 3: Computation of the detection and recognition losses of a joint architecture trained with Procedure 1 (Conventional detector training). The outputs of the model components shown in Fig. 2 are omitted for clarity.

Procedure 3 (Crop-Batch Training)**.** *The training phase for detection is the same as in Proc. 2, but during the recognition training phase, we use a batch of image crops as input, instead of the entire images. We again use the respective ground-truth bounding boxes of each crop in the batch to apply RoI pooling on the feature maps that are returned by the backbone. This is illustrated in Fig. 5. Note that we do not resize the crops, such that the absolute size of products in images during both training phases stays the same.*

Procedure 4 (Task-Specific Training)**.** *Again, the detection training phase is the same as in Proc. 2. The input of the recognition training phase, however, now consists of* individual product images. *RoI pooling is applied on the entire backbone feature map. See Fig. 6 for an example.*

4 IMPLEMENTATION

In this section, we demonstrate how the procedures described in Sec. 3 can be applied. The code, data and instructions on how to reproduce the results can be found on https://github.com/florisdf/jpdr.

For our joint model, we follow previous work (Munjal et al., 2019; Ranst et al., 2018; Xiao et al., 2017; Xiao et al., 2019) and add an extra RoI head (Girshick, 2015) to a well-studied detector architecture, in our case Faster R-CNN with a ResNet-50 Feature Pyramid Network (FPN) backbone (He et al., 2016; Lin et al., 2017; Ren et al., 2015). The extra RoI head returns a 512-dimensional recognition embedding for the corresponding region of interest by passing the RoI features through a fully-connected layer.

Each model is trained on the GroZi-3.2k (George and Floerkemeier, 2014) dataset with the improved annotations of Tonioni and Di Stefano (Tonioni and Di Stefano, 2017). This dataset consists of 123 images of store racks (similar to the input image shown in Fig. 2) with an average of 13 products per image. In total, the dataset contains 286 different products. We split up the images in the dataset into five equally-sized random folds from which four are used for training and one is used for validation. The results that we report are always the average of five runs, each with a different combination of training and validation folds.

For the RPN, the bounding box regression head and the binary classifier head, we use the same losses as described in (Ren et al., 2015). To compute a loss for the recognition head, we append an extra fully-connected layer that transforms the recognition embedding into a vector of dimension L—with L the number of product labels in the dataset—and apply softmax cross-entropy loss to that vector. We train each model for 500 epochs with a constant learning rate of 0.01 on an NVIDIA Tesla V100 GPU. The ResNet-50 FPN backbone of our model is pretrained on ImageNet (Deng et al., 2009) and frozen, except for the last layer. The layers of the RPN and the RoI heads are randomly initialized. We use stochastic gradient descent to optimize the model weights.

For Proc. 1 and the detection phases of the other procedures, the input images are resized so that their shortest size is 960 px and are then randomly cropped to a size of 800×800 px. We use a batch size of 2. The recognition phase of Proc. 2 uses the same input data, transformation pipeline and batch size. For the recognition phase of Proc. 3, we also start from the same data and, as we want the products in the crops to have the same absolute size as during the detection phase, we apply the same transformation pipeline. To create the crops, we center a box of size $s_c \times s_c$ at each

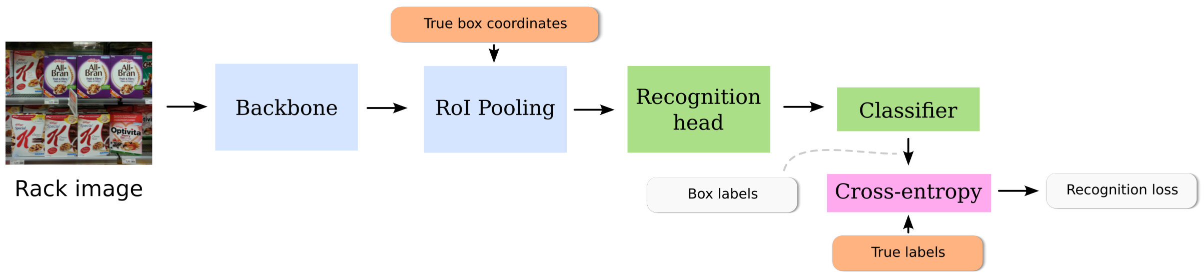


Figure 4: Computation of the recognition loss during Procedure 2 (Two-phase training). We use the ground-truth bounding boxes for RoI pooling instead of region proposals.

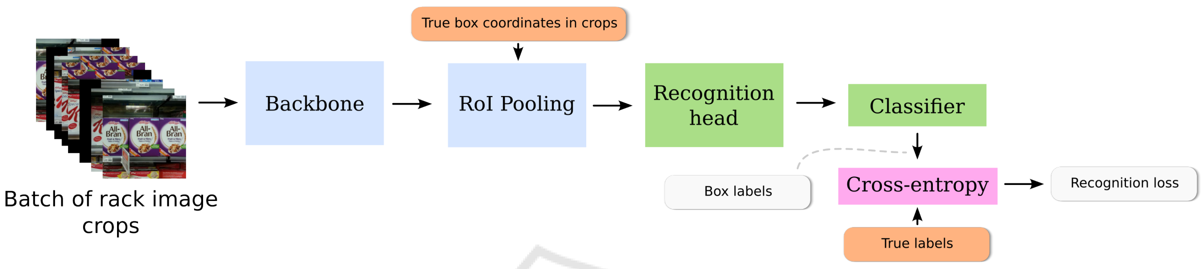


Figure 5: Computation of the recognition loss during Procedure 3 (Crop-batch training). Note that the input is a batch of crops of a rack image. We use the ground-truth bounding boxes in each crop for RoI pooling.

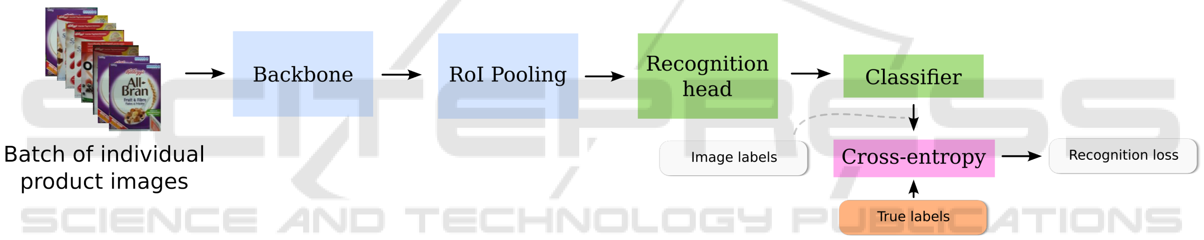


Figure 6: Computation of the recognition loss during Procedure 4 (Task-specific training). Note that the input is a batch of individual product images. The RoI pooling is now applied on the entire feature map.

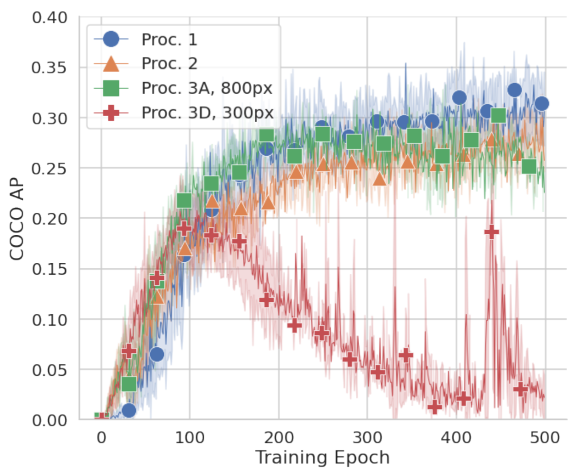
product location, with s_c a predefined crop size. When multiple boxes overlap with more than 50% IoU, only one of them will be kept. Crops that partly fall outside the image will be filled up with zeroes in that area. If more than 50% of a crop's area is outside the image, the crop is not used.

During validation, we resize the store rack images to a shortest size of 960 px and apply a center crop of 800×800 px. The images are passed through the joint network (including all three RoI heads). We employ the classifier that was used to compute the softmax cross-entropy training loss to classify the recognition embedding of each detection. Together with the result of the bounding box regression head, these product labels and confidence scores can be employed to compute the COCO AP (Dollár and Lin, 2022; Lin et al., 2015) metric.

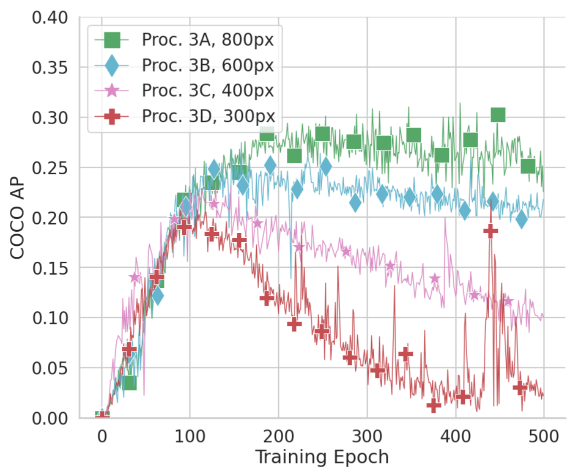
5 RESULTS

Figure 8 shows the COCO AP for Procs. 1, 2 and 3 (for $s_c = 800$ px and $s_c = 300$ px). When $s_c = 800$ px, a lot of crop boxes will overlap and/or fall largely outside the 800×800 px input image and as such be removed (see Sec. 4). In fact, only a single 800×800 px "crop" will remain, positioned around the center of the image, such that Proc. 3 becomes similar to Proc. 2. Since, after the transformation pipeline, the average size of the products in the GroZi-3.2k dataset is around 300×300 px, when $s_c = 300$ px, Proc. 3 is similar to Proc. 4. To get an idea of what these crops look like for different crop sizes, see Fig. 7.

As we can see in Fig. 8 (a), Procs. 1, 2 and 3A perform similarly. This indicates that splitting up the training in a detection and a recognition phase, does not harm the performance of the model com-



(a)



(b)

Figure 8: Validation COCO AP evaluated during training on the GroZi-3.2k dataset for (a) Procs. 1, 2 and two versions of Proc. 3; and (b) multiple versions of Proc. 3. The lines indicate the mean after five-fold cross-validation and the bands in (a) show the standard deviation. As crops fit more tightly around individual products, the validation AP starts to drop after 100 epochs.

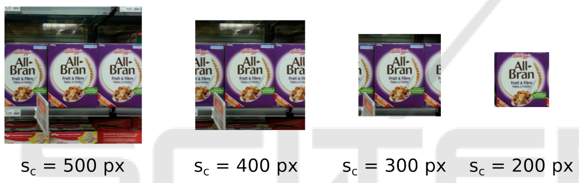


Figure 7: Example of multiple crop sizes (centered around the same product) that can be used for Proc. 3. As the crop size becomes smaller, the amount of rack context decreases.

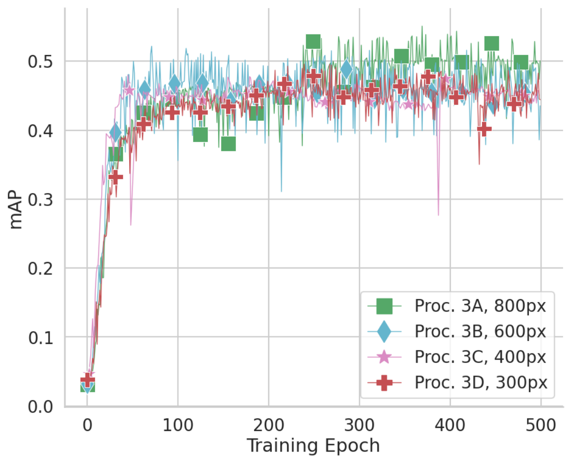
pared to training it as a conventional multi-class detector. Also, employing ground-truth bounding boxes for RoI pooling during recognition training, does not seem to be a problem. Proc. 3D, however, clearly yields inferior results. The COCO AP rises the first 100 epochs, but drops significantly after that. The maximum COCO AP achieved by Proc. 3 is clearly about ten percent points lower than the other procedures. These results suggest that separately training the recognition head on crops that tightly fit around individual products, seriously harms the performance of a joint product detection and recognition architecture. This is also confirmed by Fig. 8 (a), where we let s_c decrease from 800 px to 300 px with more intermediary crop sizes.

6 DISCUSSION

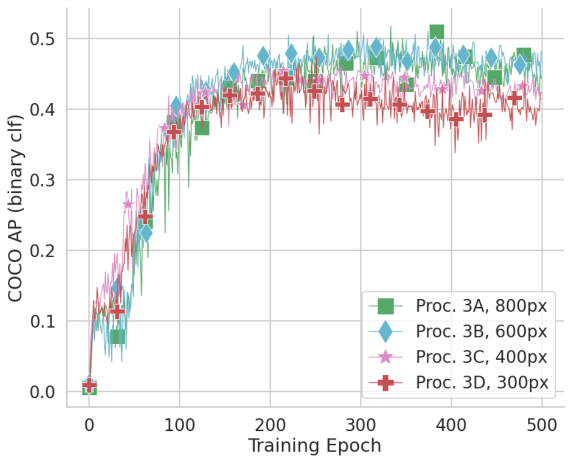
In Section 5, we ran our implementations of Procs. 1, 2 and 3. The experiments show that when $s_c = 800$ px, Proc. 3 performed similarly to the previous procedures. However, when $s_c = 300$ px, we saw a performance drop. Due to the careful definition of the procedures in Sec. 3, we can easily identify that *a smaller crop size* is causing a difference in model performance.

Why is this the case? First of all, it could be that, as the crop size becomes smaller, either the recognition or the detection task is performing worse. To investigate this, we keep track of two extra metrics during training. The first one evaluates the recognition performance. More specifically, we crop individual products out of the validation data and pass these through the backbone and recognition head. The resulting product embeddings are classified using the trained classifier from the softmax cross-entropy loss function. As such, we obtain a predicted label, along with a confidence score that can be used to compute a Precision-Recall-curve and an AP. For each validation epoch, we report the average of all image APs as the mAP. Fig. 9 (a) shows that there is no noteworthy difference in the recognition validation performance of any of the versions of Proc. 3. Second, we evaluate the COCO AP for product/no product classification during training. When this metric is low, it indicates that the detection task itself is failing. Figure 9 (b) shows a slight decrease in the COCO AP when the crop size becomes smaller, but by no means comparable to the drastic decrease we see in Fig. 8 (b).

We conclude that, for all crop sizes, the *individual* subtasks perform similar on validation data. Only when we combine both tasks and pass RoI features that come from an entire rack image to the recognition head, performance decreases. As such, we are convinced that, when trained on smaller crops, the



(a)



(b)

Figure 9: Validation (a) mAP of the recognition pipeline; and (b) COCO AP of detection pipeline during training for multiple versions of Proc. 3 on the GroZi-3.2k validation folds. Both recognition and detection seem to perform more ore less equally for different crop sizes.

recognition head does not learn to cope with the extra context that is unavoidably present in the RoI features during validation. This causes the total pipeline to perform worse.

7 CONCLUSION

In this paper, we proposed a set of training procedures to investigate the issue of employing task-specific training for a joint detection and recognition network architecture. With these procedures, we exposed an important problem that hinders task-specific training. Our experiments suggest that, by training on tightly-fit product crops, the recognition head of the joint architecture never learns to cope with the context information that is present in feature maps during inference. This inability hinders the model to achieve similar performance to a model trained on fully-annotated data.

We hope that both the proposed training procedures and our findings on the influence of context during validation will aid future research in solving task-specific training of joint detection and recognition models.

REFERENCES

Deng, J., Dong, W., Socher, R., Li, L., Kai Li, and Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Dollár, P. and Lin, T.-Y. (2022). Cocodataset/cocoapi. cocodataset.

Fang, S., Cao, Y., Wang, X., Chen, K., Lin, D., and Zhang, W. (2021). WSSOD: A New Pipeline for Weakly- and Semi-Supervised Object Detection.

Fuchs, K., Grundmann, T., and Fleisch, E. (2019). Towards identification of packaged products via computer vision: Convolutional neural networks for object detection and image classification in retail environments. In *ACM International Conference Proceeding Series*, pages 1–8, New York, New York, USA. Association for Computing Machinery.

George, M. and Floerkemeier, C. (2014). Recognizing Products: A Per-exemplar Multi-label Image Classification Approach. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, volume 8690, pages 440–455. Springer International Publishing, Cham.

George, M., Mircic, D., Soros, G., Floerkemeier, C., and Mattern, F. (2015). Fine-Grained Product Class Recognition for Assisted Shopping. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 546–554, Santiago. IEEE.

Girshick, R. (2015). Fast R-CNN. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, Santiago, Chile.

Goldman, E., Herzig, R., Eisenschtat, A., Goldberger, J., and Hassner, T. (2019). Precise Detection in Densely Packed Scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5227–5236.

Hao, Y., Fu, Y., and Jiang, Y.-G. (2019). Take Goods from Shelves: A Dataset for Class-Incremental Object Detection. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pages 271–278, Ottawa ON Canada. ACM.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *The IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, Las Vegas, NV, USA. IEEE.

Kang, B., Liu, Z., Wang, X., Yu, F., Feng, J., and Darrell, T. (2019). Few-Shot Object Detection via Feature Reweighting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8420–8429.

Kulyukin, V., Gharpure, C., and Nicholson, J. (2005). RoboCart: Toward robot-assisted navigation of grocery stores by the visually impaired. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2845–2850.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature Pyramid Networks for Object Detection. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, Honolulu.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2015). Microsoft COCO: Common Objects in Context. *arXiv:1405.0312 [cs]*.

López-de-Ipiña, D., Lorido, T., and López, U. (2011). Indoor Navigation and Product Recognition for Blind People Assisted Shopping. In Bravo, J., Hervás, R., and Villarreal, V., editors, *Ambient Assisted Living*, volume 6693, pages 33–40. Springer Berlin Heidelberg, Berlin, Heidelberg.

Merler, M., Galleguillos, C., and Belongie, S. (2007). Recognizing Groceries in situ Using in vitro Training Data. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis. IEEE.

Munjal, B., Amin, S., Tombari, F., and Galasso, F. (2019). Query-Guided End-To-End Person Search. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 811–820.

Osokin, A., Sumin, D., and Lomakin, V. (2020). OS2D: One-Stage One-Shot Object Detection by Matching Anchor Features.

Qiao, S., Shen, W., Qiu, W., Liu, C., and Yuille, A. (2017). ScaleNet: Guiding Object Proposal Generation in Supermarkets and Beyond. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1809–1818, Venice. IEEE.

Ranst, W. V., Smedt, F. D., Berte, J., and Goedemé, T. (2018). Fast Simultaneous People Detection and Re-identification in a Single Shot Network. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6.

Redmon, J. and Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. *arXiv:1612.08242 [cs]*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 91–99.

Srivastava, M. M. (2020). Bag of Tricks for Retail Product Image Classification. In Campilho, A., Karray, F., and Wang, Z., editors, *Image Analysis and Recognition*, volume 12131, pages 71–82. Springer International Publishing, Cham.

Tonioni, A. and Di Stefano, L. (2017). Product recognition in store shelves as a sub-graph isomorphism problem. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10484 LNCS, pages 682–693. Springer Verlag.

Tonioni, A., Serra, E., and Stefano, L. D. (2018). A deep learning pipeline for product recognition on store shelves. In *2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS)*, pages 25–31.

Wang, W., Cui, Y., Li, G., Jiang, C., and Deng, S. (2020). A self-attention-based destruction and construction learning fine-grained image classification method for retail product recognition. *Neural Computing and Applications*, 32(18):14613–14622.

Wang, Z., Zheng, L., Li, Y., and Wang, S. (2019). Linkage Based Face Clustering via Graph Convolution Network.

Xiao, J., Xie, Y., Tillo, T., Huang, K., Wei, Y., and Feng, J. (2019). IAN: The Individual Aggregation Network for Person Search. *Pattern Recognition*, 87:332–340.

Xiao, T., Li, S., Wang, B., Lin, L., and Wang, X. (2017). Joint Detection and Identification Feature Learning for Person Search. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3376–3385, Honolulu, HI. IEEE.

Zhou, X., Girdhar, R., Joulin, A., Krähenbühl, P., and Misra, I. (2022). Detecting Twenty-thousand Classes using Image-level Supervision. *arXiv:2201.02605 [cs]*.