YOLO-ASC: You Only Look Once And See Contours

Petr Hurtik
CE4I, IRAFM
University of Ostrava
Ostrava, Czech Republic
petr.hurtik@osu.cz

Vojtech Molek
CE4I, IRAFM
University of Ostrava
Ostrava, Czech Republic
vojtech.molek@osu.cz

Pavel Vlasanek
CE4I, IRAFM
University of Ostrava
Ostrava, Czech Republic
pavel.vlasanek@osu.cz

***Abstract*—YOLO is a useful, one-stage tool for object detection and classification. In this paper, we consider the application of grocery product detection. The grocery stores have a significant amount of product classes, so it is beneficial to postpone the classification into a second, specialized neural network with a higher capacity. Extracting bounding boxes for a classification network is not precise enough as the detected area includes redundant information about the background. We propose YOLO-ASC, which, for rectangular-based objects, detects bounding boxes together with object contour using a quadrangular. This approach allows detecting objects more accurately and without the background. For the quadrangular detection functionality, YOLO-ASC shares the feature maps that are already present in the network, and therefore its inference time is almost identical to the original YOLO. YOLO reaches high detection precision by using YOLO apriori knowledge, anchors extracted from data. In this work, we present two experiments where we demonstrate that YOLO-ASC training converges faster due to the symbiosis between the bounding box detection and quadrangular detection. Finally, we propose a tool for generating synthetic datasets with quadrangular labels that is helpful for transfer learning.**

***Index Terms*—YOLO, object detection, object mask**

I. Problem statement

In order to decrease money-cost or to increase effectiveness, stores are automatized. As an example, we can mention hand scanners or self-service cash desks. With the spread of machine learning-based image processing, different approaches for improvement are available. In this study, we are focusing on the automatic check of articles on shelves, where the functionality is as follows. A part-time worker puts all the goods on a shelf based on a previously-given manual, takes a picture with a mobile phone camera, and the system will automatically evaluate if the positions of the goods are correct; the automatic check may save a supervisor's time significantly. Such automatic processes already exist in literature [1] and usually consist of two phases: detection and recognition. Modern object detectors can recognize objects at the same time as they are detected, but it is better to realize classification by a separate neural network with a corresponding capacity and specialized on the specific task. This approach can be observed usually in practice and also in competitions such as Signate object detection, where the official tutorial (see signate.jp/competitions/159/tutorials/11) follows the schema. It is evident that for the correct recognition of an object, it is necessary to do not fail in the detection phase. The second issue is the most detectors are based on bounding boxes detection without additional refinement, leading to a classifier getting redundant data. For an illustration, see Figure 1, where the object covers only 65 % of the detected bounding box; the rest is a background that can be disturbing to a classifier. The second area where the precise information is needed is instance segmentation, where the goal is to distinguish between the particular instances, notwithstanding they are of the same class.

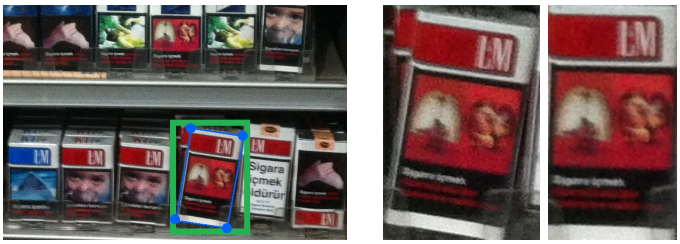

Fig. 1. An illustration of the problem. Left: a shelf image with products where a single instance is marked as detected. The green rectangle is standard detection with a bounding box, blue quadrangle is given by four vertices. Middle: the extracted instance in its original form given by the bounding box. Right: extracted and rotated instance from the blue bounding quadrangle. In comparison with the right image, the middle one includes only 65 % of information related to the object.

In our work, we use YOLOv3 [2], which is one of the fastest detectors – see Figure 2 in [3]. We propose a way how to improve YOLO bounding box detection capabilities by additionally predicting vertices of a quadrangle without a negative impact on processing speed. Assuming that a more precise object boundary improves the classification accuracy, our approach is valuable for the described problem of grocery store products detections. In this paper, we present the following contributions to the problem domain:

- with the re-use of YOLO feature maps, we expand its output vector to predict vertices of an object bounding quadrangle,
- we base the inference of vertices on apriori knowledge, anchors, that is already available in YOLO, so no other pre-preprocessing is needed,
- by the intelligent integration of our functionality into

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

YOLO, we avoid any significant addition to the size of the network, and therefore the inference time is not affected,
- we propose and implement a simple and powerful tool for generating synthetic data for training YOLO enriched with bounding quadrangle detection.

Abbreviations used in the paper

IOU	Intersect Over Union
YOLO	You Only Look Once
BB	Bounding Box
FC	Fully Connected
CNN	Convolutional Neural Network
LR	Learning Rate
LSTM	Long Short-Term Memory

II. Related work

Historically, the attempts to detect and recognize objects were based on various hand-crafted features such as those based on edges or contours [4], SIFT [5] and its derivatives [6], or histograms [7], including HOG [8]. The features were then classified using nearest neighbor in alternative space [9], SVM [10], tree/forest-based methods [11], or by shallow neural networks, to mention some of them. Let us note that the methods usually suffer from low precision and low robustness in varying conditions.

The big boom came with the development of deep neural networks. However, the first methods were based on a simple principle of sliding window [12] where each sub-part of an image was classified separately. The process was improved by generating proposal regions in R-CNN [13] and by integrating into a CNN in Faster R-CNN [14]. The current generation of neural networks, including SSD [15], YOLO [2], CornerNet [16], RetinaNet [3], or EfficientDet [17] is able to detect bounding boxes directly in a single network pass; such a functionality is called one-stage detector.

In our study, we are focusing on the quadrangle approximation of object contours in order to obtain more precise object detection. The similar (in the sense of the output) functionality is presented in Mask R-CNN [18], where additional convolutional layers are added at the end of the network to realize instance segmentation of RoI (region of interest). It is important to mention that additional functionality slows down the training/inference time. The current state-of-the-art focuses on adding the instance segmentation functionality to one-stage detectors while keeping the inference time identical. Fu et al [19] modified RetinaNet into RetinaMask, where the inference time is decreased only negligibly, mainly due to additional tricky speed-ups.

III. YOLO-ASC

YOLO (the acronym of You Only Look Once) has been developed by Joseph Redmon et al [20] in 2016, further developed by the same author in 2017 [21] and finally brought to the final v3 version in 2018 [2]. The critical fact is that the main idea has been preserved throughout the versions, and it is recalled in the following subsection.

A. YOLO v3

Let us consider a standard convolutional neural network for classification, which outputs a vector of class (e.g., dog, cat, etc.) probabilities. Such a network would work/classify well, but it will not be able to locate an object in a scene; adding a regression of the object bounding box will not solve the issue due to the network predicting the box even if the scene does not include the object. On the other hand, when the scene includes more than one object, the bounding box will be predicted somewhere between them. Adding more bounding box regressors will not solve the issue because they will be placed again somewhere in the middle between the objects due to their independence.

YOLO solves the independent bounding boxes (BBs) regression problem by splitting an image into logical sub-areas called cells, forming a grid. Each cell detects and classifies a single object. To avoid multiple detections of a single object by adjacent cells, non-maxima suppression is applied. Furthermore, together with the standard tuples (x, y, w, h) (i.e., top left corner of BB and its width and height), YOLO learns confidence whenever given cell BB contains an object or not.

In fact, YOLO can detect up to three objects in a single cell; the problem of independent regressors is solved by so-called anchors. Anchors are prototypical widths and heights of BBs extracted from the data by K-means algorithm [22] and are a form of apriori knowledge. Then, each of the three cell regressors estimates BB using one of the anchors; thus, it is guaranteed they will behave differently and try to detect different objects. Let us note that there is room for improvement by using some of the modern and more powerful clustering algorithms. To solve the problem of detecting objects with varying sizes, YOLO uses three different scales, where each scale has its own grid with its own anchors. In detail, YOLO uses scales with the grids of 13×13, 26×26, and 52×52 cells, where the input image size is 416×416px.

To be complete, the difference between the YOLO versions is that the second version [21] adds batch normalization, anchors selection, uses multi-scale prediction, and replaces the FC layers at the end of the network by convolutional layers. The current version follows the previous one and replaces the backbone Darknet-19 with a deeper one – Darknet-53 having 53 convolutional layers. This leads to a decrease in computation speed from 171FPS to 78FPS [2] on ImageNet, but on the other hand, it increases the accuracy of the network.

Within three years from the first proposal, the original paper reached more than six thousand citations and became well-accepted in the community. Excellent application of YOLO can be found in various areas such as video restoration [23], object tracking [24], or car license plate recognition [25], to name just a few.

B. Getting more

YOLO v3 backbone is a standard, fully convolutional network. The known fact is that a CNN detects the basic features such as edges in the first layers, middle layers combine them

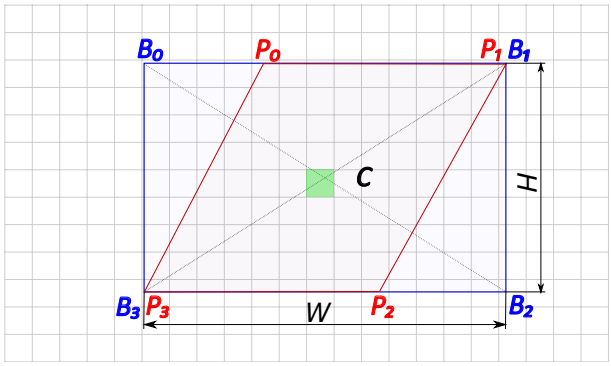

Fig. 2. An example of the bounding box and quadrangle with their properties, placed within a grid. The cell responsible for the detection of the box and the corresponding quadrangle is marked by a green color.

into intermediate representations, and finally creates abstract representations of objects in the last layers [26]. The logical assumption is that Darknet-53 combines the features in the same way as generic CNNs. We propose to leverage the learned features to estimate the object bounding quadrangle given by four vertices. It is a fact that the quadrangle differs from bounding boxes only by the level of precision. Furthermore, the vertices of the quadrangle always lie on the border of the bounding box. Because of that, it is beneficial to use the already trained Darknet-53 feature maps.

The output tensors contain the information about bounding boxes, which is extracted by 1×1 output convolutional filters. Considering that the YOLO grid includes n^2 cells, we want to predict c classes, the number of the output convolutional filters per scale is given as $3n^2(5+c)$, where 3 stands for the three anchors, 5 for predicting x, y, w, h coefficients, and the confidence p. We assume our store products to be bounded with rectangles under different transformations. We propose to describe each vertex of a quadrangular by two values so that the output vector size is $3n^2(5 + c + 8)$, i.e., the number of filters in the output layers is doubled. That may seem like a significant increase; however, **taking into account the number of trainable parameters in the Darknet-53 backbone, the increase is only negligible 0.07%** (61,523,734 vs. 61,566,814 parameters).

The question that remains unanswered is what two values per vertex we should use and how to train YOLO to learn them. The simple solution is to predict the absolute coordinates within the 416×416px image (the native YOLO input size) or to transform them into $[0, 1]$. As we mentioned above, YOLO has apriori knowledge – anchors. The goal is, therefore, to use the anchors to estimate the points of a quadrangle. For that, we consider bounding box points $\mathbf{B}$ and quadrangle points $\mathbf{P}$ as is shown in Figure 2. For such points, following holds:

1) $\mathbf{P}$ is always within the box given by $\mathbf{B}$,
2) $\mathbf{P}$ always lie on the border established by $\mathbf{B}$,

therefore, we formulate a hypothesis that the detection of bounding box given by $\mathbf{B}$ can share CNN features with detection of points $\mathbf{P}$. Furthermore, we consider alternative points $\mathbf{P}'$ where $P'_x \in [0, W]$ and $P'_y \in [0, H]$, where W and H denote width and height of a particular bounding box. The transformation $\mathbf{P} \mapsto \mathbf{P}'$ can be realized as $P'_x = C_x - P_x + 0.5W$, $P'_y = C_y - P_y + 0.5H$, so prediction of $\mathbf{P}'$ are estimated by using anchor in the same way, as is standard YOLO estimating bounding box width and height. For $\mathbf{P}'$, it also holds that the vertices are always predicted inside the bounding box and also independent of the size of an object. For that, we propose to compute the loss value L for each of the cells as

$$L = o\ell \sum_{i=1}^{4} \left(\frac{x_i}{W_A} - \hat{x_i} \right)^2 + \left(\frac{y_i}{H_A} - \hat{y_i} \right)^2,$$

where i represents particular vertex prediction and its ground-truth value, $o \in \{0, 1\}$ is a constant expressing if an object is in the cell or not. The constant is derived from YOLO, the same as $\ell \in [1, 2]$, which is computed from a box size, and it is used to increase the importance of smaller boxes that generate a lower error than the big ones. The variables W_A and H_A stand for the width and height of the closest anchor that is found for each ground truth box separately in a preprocessing phase. The loss is added to the rest of YOLO losses (for x, y, w, h, confidence, and classes) with the same weight. For the vertices, we do not estimate confidences because they are shared with the already trained one – it is obvious that the vertices cannot be detected without a bounding box prediction.

Note, if a quadrangular vertex perfectly matches an original vertex, it is then replicated and detected twice. The special case appears where all the vertices match. That means the quadrangular object is a box itself. Here, the two parts of the loss function realize the same detection, so the original YOLO functionality is preserved. That part is helpful for model training: a standard dataset, where only boxes are labeled, is used for the model pre-training. Then, the model can be fine-tuned on a dataset with the fine annotation.

IV. Experiments

In this study, we utilize two experiments based on artificial and real data. The first one focuses on the usage of our developed tool that is helpful mainly for pretraining when one is facing a lack of data, or want to see the behavior of the training process under various circumstances. The second experiment demonstrates possible real-life usage for grocery products – cigarette boxes. In the experiments, we show that our intelligent integration leads to:

1) possibility of quadrangular border detection,
2) preserving YOLO inference speed,
3) fewer epochs for its full training,
4) improved precision of BB detection,
5) increased classifier accuracy.

A. Experiment with the synthetic data

Because the dataset used in the second experiment can be considered rather small from the number of images point of view, we implemented a tool for generating a synthetic dataset that will be used for transfer learning. The tool supports a

Fig. 3. Illustration of generated images from the synthetic dataset.

variety of shapes to be rendered with various options and a desirable amount of randomness. Its features are as follows:

- generates triangles, squares, pentagons, and stars,
- Each shape has a random position and color,
- supports randomized shape rotation,
- allows enabling/disabling occlusion,
- exports YOLO-friendly labels,
- everything mentioned above is fully customizable.

The tool is written in Python using the Jupyter platform, and it is available on our GitLab[1]. Using the tool, we generated $4 \cdot 10^3$ training images, $1 \cdot 10^3$ validation images, and $4 \cdot 10^3$ testing images. All images have a resolution of 416×416px and include 1-10 boxes per image with random size, aspect ratio, color, background, and noise. The illustration of the dataset images is shown in Figure 3. With the dataset, we trained both standard YOLO and YOLO-ASC with the following configuration: Adadelta optimizer, batch size 8, Reduce LR with factor 0.5 and patience 2, 100 epochs, and early stopping set to 10.

The training computation performance is 311s/epoch for YOLO and 314s/epoch for YOLO-ASC; inference 28.2ms/image for YOLO and 28.4ms/image for YOLO-ASC. The overhead is 0.9% for training and 0.7% for inference time, which is negligible. On the other hand, the standard

[1]https://gitlab.com/irafm-ai/yolo-asc/tree/master/tool_for_synth_data

TABLE I
COMPARISON OF THE PERFORMANCE ON THE *synthetic test set*

	Average IOU for threshold							
	.5		.6		.7		.8	
Type	□	◊	□	◊	□	◊	□	◊
box-box	.899	.914	.898	.914	.897	.913	.873	.904
box-poly	.627	–	.537	–	.389	–	.216	–
poly-poly	–	.871	–	.870	–	.862	–	.786

□ YOLO ◊ YOLO-ASC

YOLO has been early stopped at epoch 52 while the polygon version at epoch 46, so the training time is **4 hours 12 minutes** for the standard and **4 hours 0 minutes** for our proposed one. All times are measured on an RTX-2080Ti graphic card.

For the trained models, we compared $IOU_{0.5}$ on the test dataset. Considering only the bounding boxes, the standard version reached the score **0.899**, while our proposed technique reached **0.914**; notwithstanding, it was trained for a shorter time due to early stopping. It is obvious that the functionality of quadrangle learning is connected with the functionality of bounding box learning so they can strengthen each other. For further analysis, we propose the following comparison based on $IOU_{0.5}$ metric. For YOLO, we compare ground truth bounding box and prediction polygon representation. For YOLO-ASC, we compare label polygon representation and prediction polygon representation. YOLO reached **0.627**, which was outperformed by **0.871** score given by YOLO-ASC. That shows that YOLO-ASC is not only better from a bounding box estimation point of view, but it is capable of detecting the precise information without a redundant background as well. The output of the inference is visualized in Figure 4 and in Table I, where we show results of the additional IOU metrics. As it is evident from the figure, the visual output corresponds with the numerical results: **YOLO-ASC detects the quadrangular borders with nearly perfect precision, and furthermore, the detection of bounding boxes is even slightly more precise than the YOLO one**.

B. Experiment with the real data

For the benchmark dataset, we have selected *Grocery Dataset* [27] that is freely available online[2]. It consists of 353 shelf images, i.e., images where the various amount of shelves with a variable number of products are captured. The variability is the advantage of the dataset: according to the authors, the images were captured in approximately 40 grocery stores by four different cameras. The images vary in resolution, aspect ratio, lightness, orientation, and level of detail. In total, the shelves' images include 13054 cigarette boxes. For an illustration, see Figure 5 that shows several images. Because the labels are given in the form of bounding

[2]https://github.com/gulvarol/grocerydataset

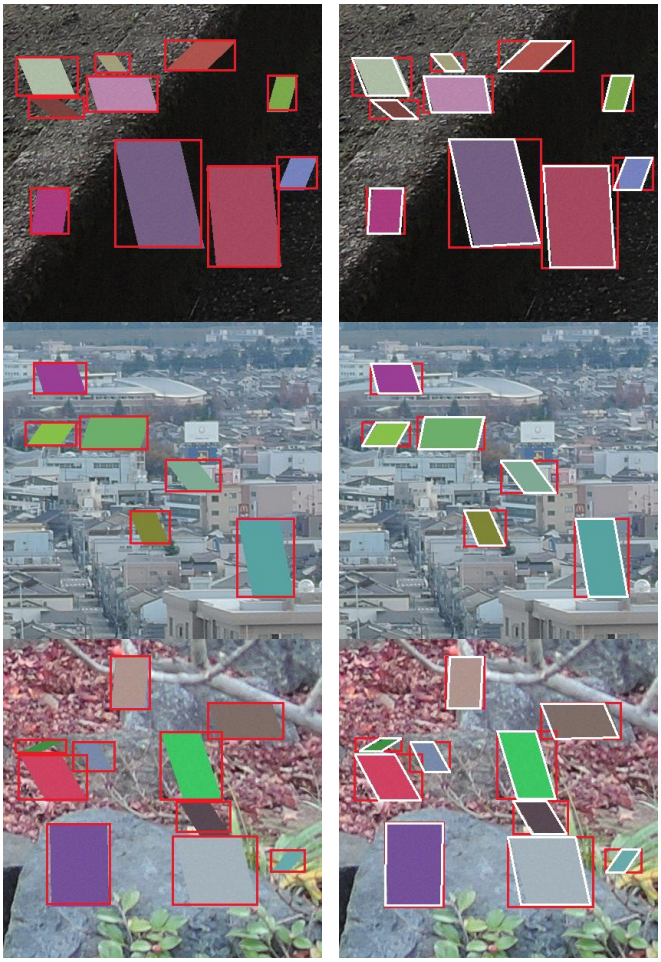

Fig. 4. An example of predicted images from the *synthetic test set* for $IOU_{0.5}$. Left: the standard YOLO. Right: YOLO-ASC.

boxes, we manually annotated the quadrangular borders of the cigarette boxes using CVAT tool[3] and put the labels online[4].

For our experiments, we extracted 30 randomly selected images to serve as a test set. The images contain 1275 cigarette boxes. Furthermore, when YOLO is trained, 10% of images from the training set are used for validation. YOLO and YOLO-ASC were trained with the same setting as in the previous experiment, i.e., we used Adadelta, batch size 8, reduce LR with factor 0.5 and patience 2, 100 epochs and early stopping set to 10. Furthermore, we created a synthetic dataset that mimics the real data using our tool described in the previous section. The synthetic dataset serves as a basis for transfer learning. The results support our assumption that YOLO-ASC learned faster due to the fact that the detection of bounding boxes and the detection of quadrangular borders are linked together and support each other. While YOLO stopped after 38 epochs, YOLO-ASC stopped after 21 epochs.

We tested the quality of prediction for four various IOU, see Table II. YOLO includes its own IOU threshold for training

[3]https://github.com/opencv/cvat

[4]https://gitlab.com/irafm-ai/yolo-asc/tree/master/real_dataset/annotations

Fig. 5. Illustration of randomly selected images from the *real dataset*.

TABLE II
COMPARISON OF THE PERFORMANCE ON THE *real data test set*

	Average IOU for threshold							
	.5		.6		.7		.8	
Type	□	◊	□	◊	□	◊	□	◊
box-box	.876	.878	.876	.877	.871	.871	.816	.822
box-poly	.855	–	.854	–	.839	–	.750	–
poly-poly	–	.860	–	.859	–	.847	–	.777

□ YOLO ◊ YOLO-ASC

that is set to value 0.5. Therefore, we selected $IOU_{0.5}$ as the minimum one in our comparison; lowering it does not bring any significant difference. On the other hand, $IOU_{0.8}$ (or bigger) is selected only rarely; thus, we stopped here. As it is obvious from the table, we fulfill our assumption that YOLO-ASC **can reach higher or the same accuracy** for the BB prediction as to the original one, even though it was **trained for the fewer epochs**. Comparing BB to polygon prediction (YOLO) or polygon to polygon prediction (YOLO-ASC), we reached higher detection accuracy, especially for the higher IOU thresholds.

In order to verify the claim that a more precise detection would lead to a better classification, we trained a MobileNetV2 classifier. Then, we fed the trained classifier with two types of data. The first are images extracted as standard boxes, while the second are images where YOLO-ASC removed the background according to its precise detection. **The classifier accuracy is 0.897 vs. 0.890, i.e., the quadrangular detection leads to a slight increase of the classifier accuracy.**

To put the results into the state-of-the-art context, we trained Mask R-CNN [18] with ResNet50 backbone in the same way, i.e., we pre-trained it on artificial data and then tuned on the real one. Considering $IOU_{0.5}$ and mask comparison, Mask R-CNN achieved a great score of 0.897. On the other hand, the training took approx 1450s per epoch (YOLO-ASC 130s) and prediction approx 11s per image, which is more than **300× slower than YOLO-ASC**.

Fig. 6. Cropped results from the *real data test set* for $IOU_{0.5}$. Left: YOLO. Right: YOLO-ASC.

V. Dead ends and limitations

During the development and experiment realization, we observed that there are several issues to be mentioned and can lead to further work.

The major dead-end of our original proposition that is the motivation for our final proposed technique is that direct prediction of vertices in the meaning of absolute coordinates without their anchor estimation does not work properly. In this case, the whole machinery achieves worse results than the original YOLO.

The second issue we found is the problem of extending the original YOLO functionality without deeper analysis. In YOLO-ASC, we estimate box width and height by anchors the same as in YOLO. Following the code, width and height are used in their logarithmic form to reach a more accurate result. However, applying log into quadrangle vertices leads to the collapse of loss function into not-a-number (NaN) because the vertices are from $[0, 1]$, including zero. The application of different preprocessing function is still open.

YOLO-ASC is based on YOLO; therefore, it inherits original YOLO limitations. The noteworthy one is that considering the big, medium, and small objects, the detection of big (extremely big) ones is the least precise. Actually, YOLO tends to miss such objects completely. This is mentioned in the original YOLO paper [2]: *With the new multi-scale predictions, we see YOLOv3 has relatively high APS performance. However, it has comparatively worse performance on medium and larger size objects. More investigation is needed to get to the bottom of this*.

The current most significant limitation of YOLO-ASC, which gives direction to our future work, is how quadrangle vertices are handled. Firstly, vertices have to be in the same order across the whole dataset. It means that, e.g., left top vertex has to be always placed on the same position in the label vector. That is because the prediction of vertices is realized without a knowledge of the previous vertex. For the same reason, YOLO-ASC cannot be, in its current form, generalized into the detection of a variable number of vertices. Such functionality would require an LSTM-based approach that would be time demanding and, therefore, not fitting our requirement to retain the original YOLO speed. For illustration, see [28], where the same LSTM extension is applied to semantic segmentation.

VI. Summary

Following the increasing trends in automatization, we have recalled the problem of grocery store goods localization and recognition. Here, we keep the two-stage schema, where objects are localized and extracted first and then recognized by a separate system. In this paper, we have chosen YOLOv3 to be the detection algorithm due to its excellent computation speed. We noticed that with the schema, the disadvantage is that object extraction is based on BB and therefore includes redundant information such as the background. Consequently, we propose YOLO-ASC that is able to detect precise quadrangular shape. It extends the original YOLO by only several convolution filters in output tensor, which increases the number of neural network parameters only by 0.07%. Therefore, the inference speed remains the same in YOLO-ASC as in the original YOLO. The inner logic of the precise detection relies on the transformation of points of quadrangle into an interval that can be estimated using anchors, i.e., YOLO apriori knowledge. Such quadrangular detection is then linked with BB detection that leads to the possibility of sharing feature maps.

In the paper, we presented two experiments based on synthetic and real data sets. Results show YOLO-ASC converges faster, training time is reduced, the precision of BB detection is increased, and that quadrangular detection eliminates the useless background. Comparing the results with Mask R-CNN, we demonstrated that YOLO-ASC is significantly faster. Finally, YOLO-ASC is available freely on our GitLab repository[5].

Acknowledgment

The work was supported from ERDF/ESF "Centre for the development of Artificial Intelligence Methods for the Automotive Industry of the region" (No. CZ.02.1.01/0.0/0.0/17_049/0008414)

References

[1] Y. Zhu, C. Zhang, D. Zhou, X. Wang, X. Bai, and W. Liu, "Traffic sign detection and recognition using fully convolutional network guided proposals," *Neurocomputing*, vol. 214, pp. 758–766, 2016.

[2] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[3] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[4] B. Leibe and B. Schiele, "Analyzing appearance and contour based methods for object categorization," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2. IEEE, 2003, pp. II–409.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] https://gitlab.com/irafm-ai/yolo-asc

[6] A. E. Abdel-Hakim and A. A. Farag, "Csift: A sift descriptor with color invariant characteristics," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. Ieee, 2006, pp. 1978–1983.

[7] F. Porikli, "Integral histogram: A fast way to extract histograms in cartesian spaces," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 829–836.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005.

[9] Y. Tang, C. Zhang, R. Gu, P. Li, and B. Yang, "Vehicle detection and recognition for intelligent traffic surveillance system," *Multimedia tools and applications*, vol. 76, no. 4, pp. 5817–5832, 2017.

[10] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for svms," in *Advances in neural information processing systems*, 2001, pp. 668–674.

[11] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[12] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012, pp. 3304–3308.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

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

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[16] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 734–750.

[17] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *arXiv preprint arXiv:1911.09070*, 2019.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[19] C.-Y. Fu, M. Shvets, and A. C. Berg, "Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free," *arXiv preprint arXiv:1901.03353*, 2019.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[21] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[22] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[23] S. Ali, F. Zhou, A. Bailey, B. Braden, J. East, X. Lu, and J. Rittscher, "A deep learning framework for quality assessment and restoration in video endoscopy," *arXiv preprint arXiv:1904.07073*, 2019.

[24] G. Ning, Z. Zhang, C. Huang, X. Ren, H. Wang, C. Cai, and Z. He, "Spatially supervised recurrent convolutional neural networks for visual object tracking," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[25] S. M. Silva and C. R. Jung, "Real-time brazilian license plate detection and recognition using deep convolutional neural networks," in *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2017, pp. 55–62.

[26] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Unsupervised learning of hierarchical representations with convolutional deep belief networks," *Communications of the ACM*, vol. 54, no. 10, pp. 95–103, 2011.

[27] G. Varol and R. S. Kuzu, "Toward retail product recognition on grocery shelves," in *Sixth International Conference on Graphic and Image Processing (ICGIP 2014)*, vol. 9443. International Society for Optics and Photonics, 2015, p. 944309.

[28] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic object parsing with graph lstm," in *European Conference on Computer Vision*. Springer, 2016, pp. 125–143.