Scientific Work Short Paper Feasibility Study - SmartWarehouse

Felix Hausberger and Robin Kuck

***Abstract*— In this short paper the object detectors *You Only Look Once* and *Single Shot MultiBox Detector* are compared for precision, reactivity, training and inference behaviour and examined for their potential for industrial use. The background scenario of the Smart Warehouse offers live video data of a drone with goods in a warehouse, which are to be classffied and localized in real time. In the future, this should make it possible to carry out inventories and inventory analyses of a warehouse in a time- and cost-efficient manner conserving resources.**

The goal of this feasibility study is to find out whether the Smart Warehouse scenario is technically feasible. In addition, the focus is also on the object detectors themselves, their differences in architecture, behavior and how well they are generally suitable for industrial application scenarios.

I. Introduction

Object detection represents a major field of study in industrial fields like autonomous driving, industrial processing or even government monitoring. Especially in times of the industrial change towards Industry 4.0 such object detectors represent an optimization potential not to be neglected, e.g. in warehousing and logistics. Combined with an autonomous drone, such object detectors could make it possible to conduct inventory checks in a warehouse without human assistance. How different object detectors behave when applied to a real time industry scenario like *SmartWarehouse* should be evaluated in this short paper. As being a feasibility study, the main goal of this work also is to discuss the feasibility of the *SmartWarehouse* idea based on the performance of the two selected object detectors *You Only Look Once* (YOLO) and *Single Shot MultiBox Detector* (SSD) in terms of precision, responsiveness, training and inference behavior. In section II related work to similar industry scenarios should be introduced before explaining the approach and architecture of the *SmartWarehouse* prototype in section III. In section V the results of the feasibility study will be presented and evaluated before the interpretability of the results is discussed in section VI. section VII gives a quick coonclusion about the main achievements of this short paper.

II. Related Work

The idea of automized inventory checks with drones is not new. [1] uses a similar approach, but uses normal RFID technology or simple barcodes to identify a product. Using this approach only a small number of instances at a time can be identified, while using object detection algorithms enable many-numbered and faster processing.

The paper [2] introduces the *Single Shot MultiBox Detector* (SSD), an object detector reaching up to 76.9% mean average precision (mAP) while still keeping real time detection characteristics compared to its preceeding competitors like regional convolutional neural networks (R-CNNs). It uses a single neural network to detect objects and creates a set of default bounding boxes over different aspect ratios and scales for each feature map location instead of using an additional bounding box proposal step.

The *You only Look Once* (YOLO) algorithm was introduced in [3] and as well uses a single neural network to predict bounding boxes and class probabilities. Incremental improvements led to version three of YOLO, which is equally accurate as SSD, but three times faster [4].

III. Architecture of the SmartWarehouse scenario

A. Defining the scenario

Two goals were defined for the feasibility study of *SmartWarehouse*. First it is to be evaluated, how well existing object detectors perform in industry scenarios taking the example of *SmartWarehouse* and the two object detectors YOLO and SSD. Second, one should deal with the feasibility of the implementation of this *SmartWarehouse* scenario, which comprises the execution of an inventory for department stores with a drone.

B. Dataset

The *SmartWarehouse* is based on a large department store, where products are not packed in are not packed in cartons, but are arranged as a whole on shelves. In the construction of the training dataset, however, the feasibility study does not aim to represent such a store completely in the dataset, but rather to select a dataset that is as extensive as possible in order to prove the general feasibility of the *SmartWarehouse* scenario. In the context, the focus was therefore on beverage bottles of a department store. Nine categories were defined: *Saskia Water Small*, *Saskia Water Large*, *Pepsi Cola Small*, *Pepsi Cola Large*, *ISO*, *ACE*, *Stenger Blackcurrant Spritzer*, *Stenger Apple Spritzer*, and *Vitamalz Malt Beer*. All classes are distributed equally in the dataset. The dataset itself is 1.45 GB big and consists of 1088 manually annotated images of resolution 2112x4608 and 24 bit color depth. The annotations are given in *PascalVOC* and *YOLO* format. 75% of the images include a single instances of a beverage bottle to better train the patterns of the different classes. Furthermore, 13% of the images contain occlusions of objects to be detected, detection situations from extreme viewing positions are represented to 4%, and difficult illumination and lighting conditions are included to 8%. Only a few examples (2%) are related to different distances during detection. Care was also taken to continuously vary the backgrounds of the objects being detected. With the enrichment of the dataset with representations of such difficult recording circumstances, it is hoped to increase the generalization capability of Deep Learning models based on them. Finally, to simulate the department store where several objects are to be detected at once, in 12.5% of all images the objects are arranged on shelves, one behind the other or in beverage crates. From the total dataset of 1088 images, 90 images were extracted for later validation. The remaining 998 consist of 200 images of test data and 798 images for training. In 90% of all images, only objects of one class are shown, since commodity inventories are likewise mostly grouped by commodity. The dataset was published on Kaggle[1] for both PascalVOC format and YOLO format.

C. *Introduction of evaluation criteria*

To better compare object detectors with each other the following evaluation criteria have been defined:

- Precision: The precision of an object detector measured in mAP
- Reactivity: The inference rate with the model measured in *Frames Per Second* (FPS)
- Training behavior: The speed of the gradient curve until convergence measured in training epochs
- Inference behavior: The inference behavior under special lighting conditions, extreme viewing positions, occultation of objects, and the behavior with double detected objects measured with good, medium, poor.

D. *Selection of object detectors*

For the *SmartWarehouse* scenario, a choice should be made between the four detectors Faster R-CNN, Mask R-CNN, SSD and YOLO.

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Fig. 1. Comparison of SSD on PascalVOC 2007 [2]

According to the reference results in Figure 1, it is clear that the SSD performs best in terms of mAP with 74.3% and 76.8%, respectively. Faster-RCNN can keep up with 73.2% in terms of mAP, but with only 7 FPS it is not designed for fast inference. YOLO scores worse than the SSD in both categories, it essentially achieves an mAP of 66.4% and a frame rate of 21 FPS. The SSD thus manages to maintain a good balance between precision and responsiveness. Comparing Mask-RCNN from the original paper in [5] with Faster-RCNN (RoI-Align) results in a mAP difference of 38.2% to 37.3%. Only 5 FPS could be reached.

To find two suitable object detectors to evaluate, the effort setup the implementations and to adapt them to be trained with a custom dataset was also taken into account. Whereas the SSD and YOLO implementations were easy to handle regarding these two tasks, implementations of ferster-RCNN or Mask-RCNN were either not supported on the operating system of choice or already depricated and no longer supported.

Due to these circumstances and the poorer performance in time-critical model inference, YOLO and SSD were selected as the two detectors to evaluate the capability of object detectors for industrial use using the *SmartWarehouse* scenario as an example. The official implementation of YOLOv3 in the *Darknet* framework and a custom implementation of SSD300 in the *PyTorch* framework is used.

E. *Training infrastructure*

When selecting the training infrastructure, cloud PaaS offerings were first considered. Important factors in the selection process were the lowest possible operating costs, a diverse range of hardware accelerators, and simple setup of the training infrastructure. In particular, the test versions of the respective offerings should be taken advantage of in order to achieve low operating costs.

Amazon SageMaker did not offer cloud GPUs as a hardware accelerators at all in its free trial version, which is why is was elimitated from the selection. Similarly, the not enough quota was available for the *Deep Learning VM* from the *Google Cloud Platform*. However, since Microsoft Azure's offering was only described very superficially, the decision was ultimately also made against Microsoft Azure.

The exlusion of the three hyperscalar offerings led to *FloydHub* as another deep learning platform. Here hardware accelerated GPUs were freely accessible from the very beginning until a certain time throshold was reached based on the GPU model. During the training with the NVIDIA Tesla K80, however, it was noticeable that the choice of this GPU did not bring any major performance improvements compared to local training.It is noticeable that an NVIDIA Tesla K80 can keep up with the existing local graphics cards in the degree of parallelization, but scores far worse in the number of computing operations per second. Compared to an NVIDIA GeForce GTX 1080, the NVIDIA Tesla K80 performs less than half as fast in terms of computing power. For work-related reasons, it should be noted that the two object detectors SSD and YOLO are each trained locally on different hardware, SSD on an NVIDIA GeForce GTX 1080 and YOLO on an NVIDIA GeForce RTX 2060 GPU. Therefore, it was made use of NVIDIAs' *Compute Unified Device Architecture* (CUDA) as a programming model and parallel computing platform to offload computing power to the GPUs. Also the NVIDIA *CUDA Deep Neural Net-*

[1]see `https://www.kaggle.com/fidsusj/smartwarehousessd` and `https://www.kaggle.com/fidsusj/smartwarehouseyolo`

work Library (cuDNN) was used, which offers hardware-optimized routines for convolution or pooling operations.

F. Drone selection

Analyzing the market for programmable drones with a freely available software development kit (SDK) and integrated camera, the supply is very low. The choice restricts to either the *Parot Bebop II* or the *Ryze Tello EDU*. When also taking into account the legal framework, only the *Ryze Tello EDU* was left for legal usage.

G. Inventory software specification

To process the drone's image data for an inventory, the drone is to be accessed by a client application. The server, acting as a client, is to connect to the drone and sends the flight signals as static flight instructions to the drone. The deep learning model is also to be integrated on the server for inference. The server's task will be to infer the video stream frames received from the drone with the model, count the detected objects for the inventory using a counting algorithm, and then forward the image data with the bounding boxes drawn in via livestream to a web application for visualization. A REST interface should provide information about the inventory data of the warehouse and provide a way to initiate the flight sequence.

IV. Realization

SSD. The base network of the SSD consists of a VGG16 pre-trained on ImageNet. The remaining convolutional layers are Xavier initialized. SSD was trainied with a batch size of 16, $1.0e^{-3}$ as learning rate and 0.9 as momentum. ReLU was chosen as its activation function and smooth L1 as its loss function. Using the mini-batch learning approach training was done with six-fold cross validation with 22 epochs each.

YOLO. YOLO was trained with a batch size of 64 and 16subdivisions. The learning rate was $1.0e^{-3}$ initially, then decreased to $1.0e^{-4}$ after 80%, and to $1.0e^{-5}$ after 90% of the training process, the momentum was also 0.9. 18.000 batches were specified in total for the mini-batch gradient descent training process. LReLU was chosen as the activation function and squared errors as the loss function.

Counting algorithm. After each inference process, the detected objects are saved so that the algorithm can access both the most recently detected objects and the newly detected objects when it is called. For all detected objects in the current run, the algorithm searches all last detected objects to find out if an object was already present in the previous image. Crucial for this is the same label as well as the distance of the bounding boxes of the currently detected object to the object of the previous round. Only if the object was not already present in the previous round according to these criteria, it is counted.

Drone connection. For control and access to the video stream, the Ryze Tello EDU offers its own WiFi network to which the control unit must connect. The communication takes place via the UDP protocol1 and consists of several communication channels. The sending of flight commands and the polling of the video stream is done via a bidirectional socket connection between the client application and the Ryze Tello EDU.

Model inference. Since the H.264 decoder can only run on Python version 2.7 and the PyTorch framework used for the SSD or the Darknet framework for YOLO can only run on Python version 3, it was not possible to allow inference of live video data from the drone. Python version 3 is generally not backward compatible with previous versions, so the H.264 decoder required by the Ryze Tello EDU SDK was not compatible with the existing infrastructure. Because of this, it was decided to simplify the problem by inferring images from a webcam instead of images from the drone.

V. Results and evaluation

SSD reached an mAP score of 83.1%, while YOLO scores almost equally high with 80.4%. To minimize double detected objects the confidence score for SSD was set to 0.7, for YOLO already 0.25 was enough. SDD struggles to detect partly hidden objects. Also detecting objects from extreme view positions was difficult, but can be improved by adding more samples for this case. For those special cases YOLO performs equally bad, for the latter even worse. SSD reliably detects objects until a three meters distance threshold, YOLOs' maximum range is only two meters. Overlit lighting conditions leads to no or wrong object detection using the SSD or YOLO algorithm. However, all detections of both models reacted invariantly to different backgrounds or image resolutions.

Both SSD and YOLO are capable to run inference in the real time scenario of *SmartWarehouse*, SSD with 30 PFS and YOLO with 28 FPS.

SSD took 16.5 hours to train, whereas YOLO took 8 hours. Nevertheless, as both algorithms were trained on different hardware, use different frameworks and use different batch sizes, the results cannot be really compared.

VI. Discussion

Use of the object detectors in the industry. In terms of precision, both object detectors turn out better than the original reference results from the scientific publications. However, it should be noted that the results do not compare well with those of the scientific publications, since they were trained on different, significantly simpler data. With 83.1%, SSD300 is 8.8% better, while YOLOv3 even shows an improvement of 16.7%. Nevertheless, it can be concluded that both implementations scale in terms of precision in an order of magnitude that comes after the level of the comparison results from PascalVOC 2007.

Another evaluation criterion is the responsiveness. Here, SSD300 scores slightly better with an average of 30 FPS than YOLOv3 with 28 FPS, although this difference should hardly be seen as a true decision criterion to prefer SSD300 over YOLOv3.

As training for SSD and YOLO was done on different hardware, the training duration of 16.5 hours for SSD and 8 hours for YOLO cannot really be compared. Nevertheless if it were possible to use cloud infrastructure for the training process of both implementations, training duration could be scaled down tremendously. Nevertheless, YOLO was easier to adapt to a custom dataset and uses subdivisions to avoid a memory overflow by partitioning the amount of images to be loaded into graphics memory in one batch.

Regarding inference behavior both object detectors stuggle with difficult lightning conditions, extreme viewing positions, longer distances and occlusions. Some of those problems could be tackled better by enhancing the dataset with additional instances for those cases. For a close comparison see Table I.

TABLE I
INFERENCE BEHAVIOR OF SSD300 AND YOLOV3

Criterion	SSD300	YOLOv3
Lighting	0	0
Extreme view positions	0	0
Distance	0	0
Occlusion	-	-
Double detection	+	+
Background	+	+
Resolution	+	+

Overall, it can be concluded that both SSD300 and YOLOv3 are object detectors for industrial use. If many objects of different scales are to be detected in the detection environment, SSD300 is preferable to YOLOv3. However, if this is not the case and more emphasis is placed on accuracy for simple datasets such as *SmartWarehouse*, YOLOv3 is the better choice.

Feasibility of the *SmartWarehouse* scenario. For reasons of extensibility of the *SmartWarehouse* scenario, YOLOv3 was selected as the final object detector. At the moment, the scenario is only designed for beverage bags as an example. If we return to the actual idea of performing inventories in large department stores with many different products, the aspect of better precision of detection for the same objects of different scales tends to fall into the background with SSD300. Rather, it is important to enable reliable detection behavior.

Still three issues remain unsolved:

- Inferring the video stream of the drone, due to the incompatibility of the runtime environments for the H.264 decoder and the deep learning models,
- The detection of occluded objects, and
- The unique counting of objects during the inventory.

The *SmartWarehouse* scenario is therefore not feasible according to the decisions made and with the given framework conditions.

VII. CONCLUSION

The goal of this work was to evaluate how well existing object detectors are basically suited for industrial application scenarios, and to determine whether the specific application scenario for conducting an inventory of department stores using a drone can be implemented as a prototype.

Even if object detectors such as SSD and YOLO most certainly have the potential to be used industrially, the *SmartWarehouse* scenario still could not be realized end-to-end. To do so, one would need to enhance the dataset to cover certain image capturing conditions as described, implement a custom H.264 decoder to connect the drone video stream to the selected deep learning model for real time inference and find a solution to uniquely identify objects during the detection process like additionaly making use of RFID technology.

REFERENCES

[1] doks. innovation GmbH, inventairy xl (2020).
URL `https://doks-innovation.com/solutions/inventairy-xl`

[2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, Ssd: Single shot multibox detector.
URL `https://arxiv.org/pdf/1512.02325.pdf`

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, You only look once: Unified, real-time object detection.
URL `https://arxiv.org/pdf/1506.02640.pdf`

[4] A. F. Joseph Redmon, Yolov3: An incremental improvement.
URL `https://arxiv.org/pdf/1804.02767.pdf`

[5] Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick, Mask r-cnn.
URL `https://arxiv.org/pdf/1703.06870.pdf`