DOI: 10.2195/lj_Proc_klos_en_202112_01
URN: urn:nbn:de:0009-14-54221

Using Deep Neural Networks to Measure Buffer Levels in Real-time with Edge-Computing

Verwendung von tiefen neuronalen Netzen zur Messung von Pufferpegeln in Echtzeit mit Edge-Computing

Matthias Elia Klos
Paolo Pagani

Institute for Materials Handling and Logistics (IFL)
Karlsruhe Institute of Technology (KIT)

Abstract: Technological advances and increasing data traffic in the IoT environment lead to the relocation of sophisticated data processing to the edge of networks. At the same time, powerful object detection approaches based on deep neural networks have been developed in recent years. In this paper, an intelligent camera based on deep learning algorithms and consisting of low-cost hardware with limited computational and storage capacity is presented. The developed object detection solution enables real-time monitoring of the inventory of filled and empty small load carriers in a buffer zone.

[Keywords: Computer Vision, Object Detection, Deep Learning, YOLO, Raspberry Pi, NVIDIA Jetson Nano, Single-Board Computer]

Kurzbeschreibung: Der technologische Fortschritt und zunehmende Datenströme im IoT-Umfeld führen dazu, dass anspruchsvolle Datenverarbeitungsprozesse an den Rand von Netzwerken verlagert werden. Gleichzeitig wurden in den letzten Jahren leistungsfähige Objekterkennungsansätze entwickelt, die auf tiefen neuronalen Netzen basieren. Im Rahmen dieser Arbeit wird eine intelligente Kamera vorgestellt, welche auf Deep-Learning-Algorithmen basiert und aus kostengünstiger Hardware mit beschränkter Rechen- und Speicherkapazität besteht. Die entwickelte Objekterkennungslösung ermöglicht die Überwachung des Bestands von gefüllten und leeren Kleinladungsträgern in einer Pufferzone in Echtzeit.

[Schlüsselwörter: Computer Vision, Objekterkennung, Deep Learning, YOLO, Raspberry Pi, NVIDIA Jetson Nano, Einplatinencomputer]

1 Introduction

Among the five senses that enable us to perceive our environment, the sense of sight is considered as the dominant sense for us humans. Thanks to the ability of visual perception, we can process visual stimuli to obtain information about our surrounding and react accordingly to these observations. The desire to create intelligent machines has led to the emergence of the interdisciplinary field of computer vision, which deals with imitating the human sense of sight and with transferring this ability to artificial systems. One of the many subareas of computer vision is object detection, which focuses on the identification and localization of objects in different scenes [Vin14]. In recent years, significant progress has been made in the field of object detection, which is a fundamental component for many applications such as autonomous driving or robot vision. For a long time, object detection architectures were based on traditional algorithms that allowed the selection of specific regions in images, feature extraction and classification of objects. However, the growing availability of data and technological advancements have led to the increased use of deep learning algorithms that outperform traditional object detection methods. Current state-of-the-art object detection architectures, which are based on deep convolutional neural networks, are characterized by high accuracy and speed [Xia20]. Various convolutional neural networks form the backbone for numerous object detectors which can be divided into two categories: two- and one-stage detectors. Two-stage detectors separate the process of object detection into two successive stages. Firstly, suggestions for image regions that might contain objects are generated. Then, the trained classifier assigns the suggested regions to object categories. Two-stage object detectors are characterized by high accuracy, but they are relatively slow compared to one-stage detectors [Xia20]. The most significant two-stage detectors include R-CNN [Gir13], SPP-net [He15], Fast R-CNN [Gir15], Faster R-CNN [Ren15] and R-FCN [Dai16]. One-stage detectors combine object location and object classification without a prior region proposal process. This increases the speed of the detectors while maintaining a lower but still satisfactory level of accuracy [Pan19]. Some of the best known one-stage detectors are OverFeat [Ser13], YOLO [Red16], SSD [Liu16] and RetinaNet [Lin17].

DOI: 10.2195/lj_Proc_klos_en_202112_01
URN: urn:nbn:de:0009-14-54221

The topic of object detection has gained increased attention in the field of academic research. Increasingly powerful object detectors enable a wide-ranging design of object detection applications in different fields [Jia19]. A major application area for object detection are autonomous vehicles, which require the monitoring of the surrounding with accurate detection of interfering objects, obstacles and the driving path [Che17], [Ban18], [Di19]. In medicine, object detection is mainly applied to image analysis for pattern and object recognition [He19], [She19], [Yoo19], [Jae20]. Another field of application is surveillance and security, where object detection is used for the detection of people, animals or dangerous objects [Hu15], [Sul18], [Zou20]. In manufacturing industries, object detection can be used for quality management. By using cameras and deep learning methods, manufacturing defects [Fer18], [Yan20] or material errors [Nap18] can be detected. In addition, assembly processes can be optimized using assistance systems [For19] often in combination with augmented reality solutions [Su19].

Alongside the increased development of object detection applications, we can observe a continuing networking and digitization of physical objects – a phenomenon known as the Internet of Things (IoT). Cloud computing has been considered a fundamental element for the deployment of IoT systems for a long time, as it enables the central processing and storing of data. However, the increase of data traffic and associated problems, such as long response times and a reduced quality of services in IoT networks, has led to the emergence of edge computing as an alternative to cloud computing. Edge computing refers to the increasing shift of data processing to the edge of networks in order to minimize data transfer and reduce transmission and storage costs [Fra20]. In the industrial sector, edge computing can be used to develop reliable and real-time capable systems for process automation [Cha19]. Moving complex computations and deep learning applications, such as object detection, to the edge of IoT networks requires edge devices that can perform sophisticated computations. However, edge devices are often limited in their computational and storage capabilities. For this reason, selecting the proper hardware and deep learning models is an essential part of implementing deep learning-based applications on edge devices [Mul21].

In this paper we present a real-time object detection system which is based on deep learning algorithms and consists of cost-effective hardware. In chapter 2, we review use cases for object detection in logistics and successful implementation examples of object detection algorithms on hardware with limited performance. Then in chapter 3, the design of the solution is presented with the used hardware components and object detection model. In addition, the process of data generation and annotation as well as the training of the object detection model is discussed in more detail. In chapter 4, the performance of the engineered system is evaluated, based on specific parameters. The conducted examination includes on the one hand the review of operating data of the edge devices and on the other hand the verification of the accuracy and the general applicability of the object detection system, also under deviating boundary conditions. At the end of this work there is a short summary and an outlook, in which improvement potentials are discussed.

2 Related Works

In addition to the numerous application areas, object detection can also be used to automate and optimize processes in the field of logistics. Potential use cases include the development of driverless industrial trucks, picking and collaborative robots and assistance systems for manual commissioning [Thi18]. One possible application includes the development of intelligent inventory monitoring systems [Ver16]. Pallet identification and localization are among the functions necessary for autonomous transport systems [Li19]. [Pos20] developed a solution for palletizing and depalletizing of small load carriers by industrial robots for autonomous material handling. Even the simple manual task of scanning barcodes to identify packages or other objects can be replaced by automatic reading [Han17].

Looking at the scientific publications of the last few years, increased attempts are being made to implement deep learning algorithms for computer vision solutions on devices with limited computing power. These publications are mainly limited to use cases in the areas of autonomous driving and surveillance. In autonomous driving, the main focus is to recognize people [Ort20], vehicles [Ais18] or traffic signs [Zak20] and to monitor road conditions [Pen20]. In the area of surveillance, objects [Cam19], people [Dan20] or animals [Say21] that pose a potential danger are to be detected. Other publications cover the areas of smart homes [Ash19], agriculture [Maz20], production [Žid19] and medicine [Li21]. In most of the presented cases, one-stage detectors like YOLO and SSD are used as object detection model. The implementation of these deep learning models on edge devices requires suitable hardware architectures and appropriate software [Mul21]. There are efficient CPU- or GPU-based edge devices, such as single-board computers (SBC), available with or without GPU capability [Had19]. In addition, several hardware options are available to increase the speed of machine learning algorithms and improve energy efficiency. These options include application specific instruction processors (ASIP), application specific integrated circuits (ASIC) or field-programmable gate arrays (FPGA) [Sam19].

DOI: 10.2195/lj_Proc_klos_en_202112_01
URN: urn:nbn:de:0009-14-54221

3 METHODOLOGY

3.1 APPLICATION SCENARIO AND SYSTEM DESIGN

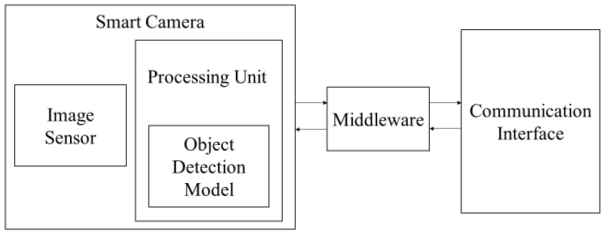
The considered use case describes an intralogistics material supply process. The task is to monitor a buffer zone in which both full and empty small load carriers are stored. The smart camera should be able to detect changes in the inventory of small load carriers so that automatic orders for new material and the collection of empty small load carriers can be initiated. The considered buffer area had a maximum dimension of 5 x 5 meters and the mounting height of the camera was between 3 and 5 meters. The objects to be detected were small load carriers with nominal dimensions between 300 x 200 x 120 mm and 600 x 400 x 220 mm. The smart camera should have some features that allow it to be used in an industrial environment. Based on these features, objectives were formulated that had to be considered for the design of the system:

1. Cost-effectiveness: The object detection solution can be realized with hardware costs under 100€ and free software.
2. Real-time capability: The system should be able to analyze videos in real time with negligible latency.
3. Accuracy: The model for the classification of small load carriers should have a mAP (mean average precision) above 0.95.
4. Transferability: The model should work with acceptable accuracy even with other types of small load carriers or containers, different contents and different backgrounds.

The selection of the necessary hardware components and detection model was carried out considering these objectives. The fulfilment of the objectives was assessed in a final analysis, the results of which are presented in chapter 4.

3.2 SYSTEM OVERVIEW

The proposed intelligent camera is composed of three main components: the image sensor, the processing unit and the object detection model. The camera is controlled via a graphical user interface that can be accessed with mobile devices. A middleware enables data exchange and the control of several connected smart cameras with the communication interface. Figure 1 shows the structure of the entire system.



The processing unit forms the core of the object detection solution. Two different single board computers were chosen as processing unit – the Raspberry Pi 4 4GB and the NVIDIA Jetson Nano 2GB. The Raspberry Pi 4 is a widely used single board computer with a 1.5 GHz ARM Cortex-A72 quad-core CPU and 4 GB RAM. The NVIDIA Jetson Nano, which was especially developed for AI applications, features an ARM Cortex-A57 CPU, a Maxwell GPU and 2 GB RAM. Both devices were selected due to their speed and performance in comparison with relatively low procurement prices. The chosen camera module is the Raspberry Pi camera module 2. The module has an IMX219 sensor from Sony and is suitable for recording high-definition videos. In addition to the camera and the processor, other components are necessary to support the operation of the smart camera. To avoid overheating of the processing unit during inference, fans were selected to cool the systems during use. Additionally, a case was designed to shield the hardware from external influences.

3.3 OBJECT DETECTION MODEL AND FRAMEWORKS

Due to the high accuracy and high speed, as well as the large scope of usable frameworks, YOLOv4 was chosen as the model for the small load carrier detection solution. YOLOv4 extends the previous YOLO network architectures by several features. This includes the selection of PANet for feature aggregation, the addition of an SPP block to increase the receptive field, the adaptation of the loss function and the integration of data augmentation methods. Additionally, YOLOv4 comprises a method for separating overlapping bounding boxes of an object instance. Batch normalization is performed by using a cross mini-batch normalization technique. The use of the so-called DropBlock regularization technique permits the supplementary simulation of occlusions [Boc20]. Besides YOLOv4, there is also YOLOv4-tiny, which is characterized by a simpler network structure and a smaller number of parameters. The lighter version of YOLOv4 has a lower accuracy, but is suitable for the implementation on devices with limited computing capacity due to the reduced computational complexity [Jia20].

DOI: 10.2195/lj_Proc_klos_en_202112_01
URN: urn:nbn:de:0009-14-54221

For the CPU-based Raspberry Pi TensorFlow was specified as the framework for inference. Due to the comparatively lower computing power, both YOLOv4 and YOLOv4-tiny were implemented to allow a comparison between the two deep learning models. TensorRT was selected for the GPU-based NVIDIA Jetson Nano, since it is especially designed for high-performance inference on NVIDIA GPUs.

3.4 DATA SET GENERATION

The first task in the development of deep learning-based object detectors is the collection of image data that is used for training and validation. Over the years, datasets with pre-annotated images have been created, which can be used for many applications. Among the best known are ImageNet [Den19], PASCAL VOC [Eve10] and MS COCO [Lin14]. However, there are no datasets containing images with or without annotations that can be used to train learning algorithms for the considered use case. The most suitable dataset is the LOCO dataset [May20], which contains annotated images from the logistics sector that also include small load carriers and other containers. The images of this dataset show the small load carriers mainly from the side view, while for the present application top view images are needed. The availability of suitable free online images that can be used as training data is also limited.

Since no suitable datasets exist for the presented use case, a custom dataset had to be generated as a basis for the development of the small load carrier detection solution. Figure 2 shows the different small load carriers used and a selection of the objects used for filling the small load carriers.

A high-quality training dataset contains images in which the objects to be detected are represented in the same way as they will be captured by the camera during operation [Lec15]. For the creation of the dataset, it is therefore necessary to capture the objects from different angles and in different arrangements to simulate possible situations in the operation environment [Liu20]. A total of three different locations could be selected for image generation. Figure 3 shows the locations with the three different backgrounds.

Background 1 (black) Background 2 (white) Background 3 (wood)

To create different situations with varying arrangements of the small load carriers, videos were created during which individual small load carriers were added, removed or repositioned. In this way, various arrangements and combinations of small load carriers with changing filling states could be simulated. In a subsequent pre-annotation step, individual frames were extracted from the recorded videos. These frames were then annotated and added to the dataset. In this way, a total of 3,835 images with 30,754 object instances could be generated. Overall, it was possible to create balanced datasets in which full and empty small load carriers are equally present. A detailed overview with a breakdown of the exact composition of the datasets can be found in table 1.

Table 1. Composition of self-generated dataset

	Number of images	Number of annotations	Class balance (empty/full)
Base dataset 1 (black)	1,387	12,367	52.6% / 47.4%
Base dataset 2 (white)	1,577	11,317	40.9% / 59.1%
Base dataset 3 (wood)	871	7,070	47.4% / 52.6%
Total	**3,835**	**30,754**	**47.1% / 52.9%**

3.5 TRAINING PROCESS

Due to the capacity limitations of the selected edge devices, the training of the object detection models had to be performed on a more powerful system. The training of the models was carried out on a server of the Institute of Material Handling and Logistics on a NVIDIA GeForce GTX 2070 graphics card. The custom Darknet framework written by Joseph Redmon [Red13] was used. A total of three different datasets, representing different combinations of the base datasets, were used for training:

- Dataset 1: Base dataset 1 (black)
- Dataset 2: Base dataset 1 (black) and base dataset 2 (white)
- Dataset 3: Base dataset 1 (black), base dataset 2 (white) and base dataset 3 (wood)

These datasets were split into a training set (70%), a validation set (20%) and a testing set (10%). The training

DOI: 10.2195/lj_Proc_klos_en_202112_01
URN: urn:nbn:de:0009-14-54221

and validation set where used for fitting and tuning the weights of the object detection model, whereas the training set was used to assess the performance of the trained model. The selected network resolution was 416 x 416 pixels. Convolutional weights pre-trained on the COCO dataset were used as initial training weights [Boc18]. Depending on the size of the training and validation datasets, the training time varied between half an hour and one hour for YOLOv4-tiny and between five and six hours for YOLOv4.

4 Experimental Results

4.1 Performance measurement

At the beginning of the design of the automatic detection solution for small load carriers, the following objectives were defined, which should be accomplished during the development of the system: cost-effectiveness, real-time capability, accuracy and transferability (see chapter 3.1). By selecting suitable hardware, the costs for the AI-based cameras could be reduced to a minimum. Both systems, which feature the Raspberry Pi or the NVIDIA Jetson Nano as the computing unit, can be constructed with hardware costs under €100, including the camera module and the other additional components. The subsequent analysis was intended to determine the degree of fulfillment of the other defined objectives for the small load carrier detection solution.

To assess the performance of the object detector different metrices were used. The speed of the detection was measured in processed frames per second. The accuracy of the object detection was evaluated using the following criteria: precision, recall and mean average precision (mAP). A detection algorithm returns the predicted detections $\{(b_k, c_k, p_k)\}_k$ of an object k, where b_k corresponds to the bounding box, c_k to the predicted category and p_k to the confidence score. The correspondence between the detected objects and the ground truth boxes is checked for each image I_j to determine the accuracy of the predictions. For a predicted detection to be considered a true positive, the overlap between the predicted bounding box and the ground truth box, also known as Intersection over Union (IoU), must meet or exceed a specified threshold. Additionally, the detected category needs to match the ground truth label and the confidence score must exceed a specified threshold β for the detection to be accepted [Liu20]. Precision describes the number of correctly detected objects in relation to the total number of detected objects by the algorithm at a given threshold. Recall is defined as the ratio between the number of detected objects and the total number of ground truth boxes. Precision and recall for an object category C_i in an image or an image set can be calculated using the number of true positives (TP), false positives (FP) and false negatives (FN).

$$P_{C_{ij}} = Precision_{C_{ij}} = \frac{TP_{C_{ij}}}{TP_{C_{ij}} + FP_{C_{ij}}}$$

$$R_{C_{ij}} = Recall_{C_{ij}} = \frac{TP_{C_{ij}}}{TP_{C_{ij}} + FN_{C_{ij}}}$$

Based on the precision and the recall of an object category, the average precision (AP) can be calculated. By varying the threshold value β, different recall and precision values are obtained, from which a precision-recall curve can be derived. The AP summarizes the area under the curve and is calculated as the mean of the precision values of specified recall levels r [Eve10]. The AP of all object categories can be used to determine the mean average precision, which is a measure of the detection performance of the model across all categories [Xia20].

$$AP_{C_i} = \int_0^1 P_{C_i}(r)dr$$

$$mAP = \frac{1}{n}\sum_{i=1}^{n} AP_{C_i}$$

The mAP is often used to compare the performance of different algorithms. However, the way of determining the mAP differs depending on the number of considered IoU thresholds [Kuz20].

4.2 Device Utilization and Latency

A real-time capable detector requires acceptable object detection latency and uninterrupted execution of all applications running on the device that are responsible for small load carrier detection or data exchange. For this purpose, the performance data of the Raspberry Pi and the Jetson Nano were collected and analyzed during operation. On the Raspberry Pi, object detection could be performed with a frame rate of about 0.2 frames per second using YOLOv4. The average object detection for a single image takes about 4.5 seconds. Further image processing steps require another 0.34 seconds, which brings the total time for analyzing a frame to 4.84 seconds. If YOLOv4-tiny is used as object detection model on the Raspberry Pi a frame rate of about 1.3 frames per second is achieved. With this frame rate, changes in the number of boxes can be detected without major delays. The inference time is about the same as the processing time for the images, reducing the total detection time to about 0.79 seconds. On the Jetson Nano, only YOLOv4 was implemented as the object detection model due to its higher performance compared to YOLOv4-tiny. Due to its graphic card and the TensorRT framework, the Jetson Nano achieves a frame rate of 4.5 frames per second. The average total time for detection is 0.22 seconds. The inference time accounts for the largest part with 0.21 seconds, the average image processing time is only 0.01 seconds. Figure 4 shows the different times for inference and image processing on the different devices.

DOI: 10.2195/lj_Proc_klos_en_202112_01
URN: urn:nbn:de:0009-14-54221

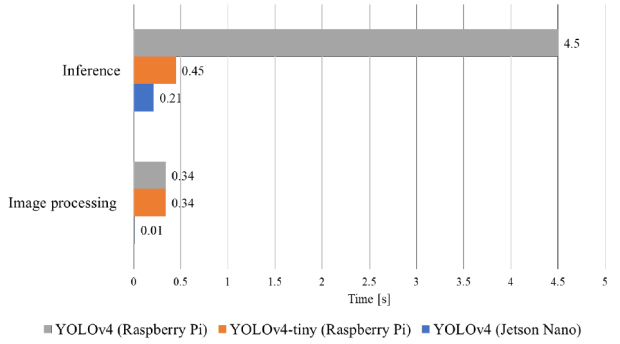
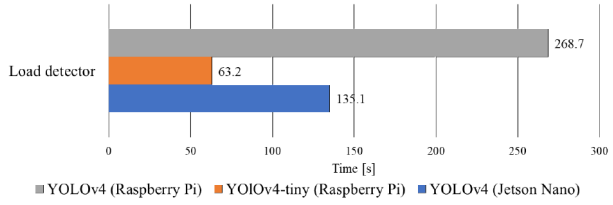


Figure 4. Comparison of average inference and processing times

The loading time of the models is also a critical factor. Before the inference can start, the model must be loaded once at the beginning. The different loading times can be found in figure 5. Since it takes several minutes to load the models, depending on the model and device, the object detection application must run continuously to update the data in real time without delay.



The Raspberry Pi never reaches its limits when it comes to memory usage. If only the object detection application is running, an average of about 2.3 GB RAM is needed. Running all applications at the same time increases the memory usage to 2.6 GB. The difference in memory usage between YOLOv4 and YOLOv4-tiny is not significant. During operation without a fan, the Raspberry Pi reached temperatures around 80 °C. With additional cooling the temperature could be kept at a constant level of 50 °C. Due to its smaller RAM, there were occasional delays and crashes when operating the Jetson Nano with graphical user interface. Operating the Jetson Nano without graphical output resulted in a maximum memory usage of 1.8 GB. Without additional cooling, the Jetson Nano heated up to 95 °C at full load. By using the fans, a constant operating temperature of around 40 °C could be maintained. It is important to note that these analyses were performed for object detection with a live stream at 480p video quality. Increasing the video quality results in increased processing times and memory utilization.

4.3 Accuracy of Object Detection

The results show that the mean average precision of the YOLOv4 and YOLOv4-tiny models is always above 99%. Overall, YOLOv4 delivers better results, but the difference to YOLOv4-tiny is marginal. The reliability of full and empty small load carrier detection varies depending on the image data used. Overall, the number of false negatives almost consistently exceeds the number of false positives. This means that predominantly small load carriers are not detected as such, rather than objects being falsely identified as full or empty small load carriers.

In general, both YOLOv4 and YOLOv4-tiny provide a good reliability in the detection of small load carriers. Both free-standing and collocated small load carriers are detected with high reliability. The average IoU also appears to be sufficient for the use case. Figure 6 shows two examples of error-free small load carrier detection.

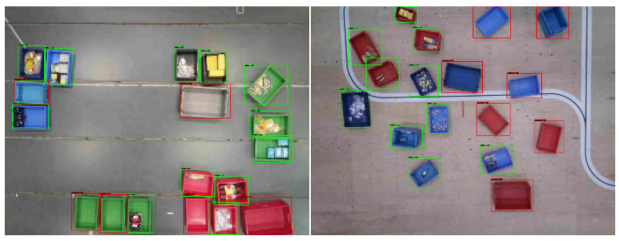


Figure 6. Visualization of exemplary detection results

Looking at the different types of errors, we can see that a large part of false detections are false negatives representing small load carriers that were not recognized. False positives, which make up the smaller portion of incorrect detections, can be interfering objects that are detected as boxes or boxes that are detected, but incorrectly classified as full or empty. These interfering objects include, for example, people captured by the camera. Reasons for misidentified empty boxes include dirt, reflections or shadows that are sometimes mistakenly recognized as objects in the box. Furthermore, it may occur that the items in the box do not distinguish themselves enough from the box due to their size or color or that they are partially covered by the edge of the box, and thus the box is incorrectly identified as empty instead of full. However, the erroneous detections represent only a very small proportion of the total detections.

DOI: 10.2195/lj_Proc_klos_en_202112_01
URN: urn:nbn:de:0009-14-54221

Table 2. Detection results of YOLOv4 and YOLOv4-tiny detectors

	YOLOv4			YOLOv4-tiny		
	DS 1	**DS 2**	**DS 3**	**DS 1**	**DS 2**	**DS 3**
Mean average precision @0.50	99.75%	99.95%	99.92%	99.80%	99.78%	99.85%
Average precision empty box	99.84%	99.90%	99.91%	99.83%	99.71%	99.84%
Average precision full box	99.66%	99.99%	99.92%	99.78%	99.85%	99.85%
Precision	1.00	1.00	1.00	1.00	1.00	1.00
Recall	1.00	0.99	1.00	1.00	0.99	0.99
F1-Score	1.00	1.00	1.00	1.00	0.99	0.99
True positives	1,144	2,277	2,845	1,143	2,268	2,835
False positives	2	7	5	2	6	8
False negatives	2	12	14	3	21	24
Average IoU	91.82%	92.58%	92.47%	91.73%	93.21%	90.65%

4.4 Transferability of Object Detection

4.4.1 Deviating Containers and Contents

Another desired feature of the developed object detector is its applicability in different scenarios. The object detection solution should be able to be used for the detection and classification of different types of small load carriers without much additional re-training effort. Small load carriers and other containers not used for training were deployed to test the generalizability of the detector. Besides the new boxes, different types of fill items were selected that do not appear in the training dataset. In addition to normal sized items, five smaller objects were selected to determine the minimum size at which an item in the box would reliably be detected. The smallest objects were a cube with an edge length of 2.5 cm and a cylindrical metal part about 1.5 cm long. Figure 7 displays all the items used.

Figure 7. Overview of new containers and filling objects

Table 3 shows the results of inference on the new test data. As in previous evaluations, a confidence threshold of 0.75 was chosen for inference. The dataset consisted of 231 images with a total of 680 objects to be detected. About 85% of full and empty small load carriers were correctly detected, using both YOLOv4 and YOLOv4-tiny. In contrast to the previous accuracy analyses, there was no significant difference between the two different models.

Generally, the new boxes were recognized, but classified as the wrong type, depending on the filling object. Only in rare cases an object instance was not identified as a full or empty box. The partially changed shapes and colors of the boxes did not have a significant negative influence on the detection. A more detailed examination of the error types revealed that about 80% of the false detections was caused by boxes containing certain small items. The smallest test object was rarely detected as a fill object due to its small size and reflective surface. The other small objects were detected more reliably, although problems occurred when the color difference to the box was not sufficient. Sometimes parts of the object were also hidden by the side panels of the box. Boxes filled with one of the two large test objects were always detected accurately.

	YOLOv4	YOLOv4-tiny
True postitives	580	583
False positives	71	73
False negatives	100	97

4.4.2 Deviating Containers and Contents

Similar to different box types and filling materials, deviating backgrounds should not negatively influence the result of object detection. Due to spatial limitations, this analysis was limited to different combinations of train and test datasets out of the self-generated base datasets. Table 4 shows the results of the detection models trained with varying non-augmented datasets. The results of the models' analysis with test datasets containing only images with backgrounds that did not appear in the training dataset are highlighted in the table.

DOI: 10.2195/lj_Proc_klos_en_202112_01
URN: urn:nbn:de:0009-14-54221

Table 4. Detection results (mAP) testing with deviating backgrounds

	Test data					
	YOLOv4			YOLOv4-tiny		
Training data	**DS 1**	**DS 2**	**DS 3**	**DS 1**	**DS 2**	**DS 3**
DS 1	99.75%	**97.27%**	**99.29%**	99.80%	**96.05%**	**99.22%**
DS 2	99.92%	99.87 %	**99.58%**	99.70%	99.81%	**99.38%**
DS 3	99.90%	99.89%	99.62%	99.84%	99.79%	99.65%

full and empty small load carriers, although some of the datasets also comprise deviating filling objects. The performance difference between YOLOv4 and YOLOv4-tiny is also noticeable in this assessment.

The tests show that it is possible to use the smart camera in different locations without retraining, provided that the same boxes and filling objects must be detected which were also used during creation of the training data. However, this statement is to be considered with reservation, since a more comprehensive analysis with a larger number of different backgrounds was not possible due to the limited space available.

5 Conclusions and Future Work

Within the scope of this work an object detection system was developed, which can be deployed for monitoring the inventory of small load carriers within buffer zones. By implementing fast object detectors with suitable accuracy on hardware with limited computational and storage capacity, it was possible to create an intelligent camera within the context of the use case that detects the specified objects with a high reliability. One prerequisite for high accuracy is that the images used for training the models include the same boxes, filling objects and backgrounds that are encountered in the practical application scenario. The applicability of the smart cameras under changed conditions could be partially confirmed. Regardless of high accuracies in some cases during the tests with unknown locations, boxes and filling objects, an extended analysis is necessary to evaluate the general transferability of the object detection solution.

Despite the good performance, there are some aspects of the solution that could be improved and simplified for operation in a real environment. One possibility is to analyze the effect of increasing the network resolution on the results and the latency of the detection. In addition, other YOLO models, such as Scaled-YOLOv4 or YOLOv5, could be implemented. The use of alternative models, such as EfficientDet, could also be considered. Extending the training dataset with images showing varying scenarios with different types of boxes, backgrounds and filling objects, could facilitate the use of the solution at locations with deviating conditions and increase the overall generalizability.

Literature

[Ais18] Aishwarya, Chaya N., Rajshekhar Mukherjee, and Dharmendra Kumar Mahato. 2018. *Multilayer Vehicle Classification Integrated with Single Frame Optimized Object Detection Framework Using CNN Based Deep Learning Architecture*. In 2018 IEEE International Conference on Electronics, Computing and Communication Technologies, 1–6.

[Ash19] Ashaj, Sudad J., and Ergun Erçelebi. 2019. *Reduce Cost Smart Power Management System by Utilize Single Board Computer Artificial Neural Networks for Smart Systems*. International Journal of Computational Intelligence Systems 12 (2): 1113–20.

[Ban18] Banerjee, Koyel, et al. 2018. *Online Camera LiDAR Fusion and Object Detection on Hybrid Data for Autonomous Driving*. In 2018 IEEE Intelligent Vehicles Symposium (IV), 1632–38.

[Boc18] Bochkovskiy, Alexey. 2018. *Darknet: GitHub.* Accessed March 12, 2021. https://github.com/AlexeyAB/darknet.

[Boc20] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. *YOLOv4: Optimal Speed and Accuracy of Object Detection*.

[Cam19] Cambay, V. Yusuf, Aysegul Ucar, and M. Ali Arserim. 2019. *Object Detection on FPGAs and GPUs by Using Accelerated Deep Learning*. In 2019 International Artificial Intelligence and Data Processing Symposium (IDAP), 1–5.

[Cha19] Chalapathi, G. S. S., Vinay Chamola, Aabhaas Vaish, and Rajkumar Buyya. 2019. *Industrial Internet of Things (IIoT) Applications of Edge and Fog Computing: A Review and Future Directions*.

[Che17] Chen, Xiaozhi, et al. 2017. *Multi-View 3D Object Detection Network for Autonomous*

DOI: 10.2195/lj_Proc_klos_en_202112_01
URN: urn:nbn:de:0009-14-54221

Driving. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6526–34.

[Dan20] Danish, Matthew, et al. 2020. *DeepDish: Multi-Object Tracking with an Off-the-Shelf Raspberry Pi*. In Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking, 37– 42.

[Dai16] Dai, Jifeng, et al. 2016. R-FCN: *Object Detection via Region-Based Fully Convolutional Networks*.

[Den19] Deng, Jia, et al. 2019. *ImageNet: A LargeScale Hierarchical Image Database*. In 2019 IEEE Conference on Computer Vision and Pattern Recognition, 248–55.

[Di19] Di, Feng, et al. 2019. *Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges*.

[Eve10] Everingham, Mark, et al. 2010. *The Pascal Visual Object Classes (VOC) Challenge*. International Journal of Computer Vision 88 (2): 303–38.

[Fer18] Ferguson, Max K., el al. 2018. *Detection and Segmentation of Manufacturing Defects with Convolutional Neural Networks and Transfer Learning*. Smart and sustainable manufacturing systems 2 (1).

[For19] Forsman, Mona, et al. 2019. *An AI-Enabled Assembly Support System for Industrial Production*. In 5th Norwegian Big Data Symposium (NOBIDS) 2019.

[Fra20] França, Reinaldo Padilha, et al. 2020. *An Overview of the Edge Computing in the Modern Digital Age*. In Fog/Edge Computing for Security, Privacy, and Applications, edited by Wei Chang and Jie Wu. 1st ed., 33–52. Cham: Springer.

[Gir13] Girshick, Ross, et al. 2013. *Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation*.

[Gir15] Girshick, Ross. 2015. *Fast R-CNN*. In 2015 IEEE International Conference on Computer Vision (ICCV), 1440–48.

[Had19] Hadidi, Ramyad, et al. 2019. *Characterizing the Deployment of Deep Neural Networks on Commercial Edge Devices*. In 2019 IEEE International Symposium on Workload Characterization (IISWC), 35–48.

[Han17] Hansen, Daniel Kold, et al. 2017. *Real-Time Barcode Detection and Classification Using Deep Learning*. In Proceedings of the 9th International Joint Conference on Computational Intelligence, 321– 27.

[He15] He, Kaiming, et al. 2015. *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (9): 1904–16.

[He19] He, Xin, et al. 2019. *Computer-Aided Clinical Skin Disease Diagnosis Using CNN and Object Detection Models*. In 2019 IEEE International Conference on Big Data (Big Data), 4839–44.

[Hu15] Hu, Guosheng, et al. 2015. *When Face Recognition Meets with Deep Learning: An Evaluation of Convolutional Neural Networks for Face Recognition*. In 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), 4321–29.

[Jae20] Jaeger, Paul F., et al. 2020. *Retina U-Net: Embarrassingly Simple Exploitation of Segmentation Supervision for Medical Object Detection*. In Proceedings of the Machine Learning for Health NeurIPS Workshop, 171–83.

[Jia19] Jiao, Licheng, et al. 2019. *A Survey of Deep Learning-Based Object Detection*. IEEE Access 7: 128837–68.

[Jia20] Jiang, Zicong, et al. 2020. *Real-Time Object Detection Method Based on Improved YOLOv4-Tiny*.

[Kuz20] Kuznetsova, Alina, et al. 2020. *The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale*. International Journal of Computer Vision 128 (7): 1956–81.

[Lec15] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. *Deep Learning.* Nature 521 (7553): 436–44.

[Li19] Li, Tianjian, et al. 2019. *Application of Convolution Neural Network Object Detection Algorithm in Logistics Warehouse*. The Journal of Engineering 2019 (23): 9053–58.

[Li21] Li, Hongjia, et al. 2021. *Real-Time Mobile Acceleration of DNNs*. In Proceedings of the 26th Asia and South Pacific Design Automation Conference, 581–86.

[Lin14] Lin, Tsung-Yi, et al. 2014. *Microsoft COCO: Common Objects in Context*. In Computer Vision – ECCV 2014, edited by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Cham: Springer.

[Liu16] Liu, Wei, et al. 2016. *SSD: Single Shot MultiBox Detector*. In Computer Vision ECCV 2016, edited by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, 21–37. Cham: Springer International Publishing.

[Liu20] Liu, Li, et al. 2020. *Deep Learning for Generic Object Detection: A Survey*. International Journal of Computer Vision 128 (2): 261–318.

[Lin17] Lin, Tsung-Yi, et al. 2017. *Focal Loss for Dense Object Detection*. In 2017 IEEE Conference on Computer Vision (ICCV), 2999–3007.

[Maz20] Mazzia, Vittorio, et al. 2020. *Real-Time Apple Detection System Using Embedded Systems with Hardware Accelerators: An Edge AI Application*. IEEE Access 8: 9102–14.

[May20] Mayershofer, Christopher, et al. 2020. *LOCO: Logistics Objects in Context*. In 19th IEEE International Conference on Machine Learning and Applications (ICMLA), 612–17.

[Mul21] Mulimani, Madhura S., and Rashmi R. Rachh. 2021. *Edge Computing in Healthcare Systems*. In Deep Learning and Edge Computing Based Solutions for Smart Healthcare, edited by Suresh A. and Sara Paiva, 63–100. Singapore: Springer.

[Nap18] Napoletano, Paolo, Flavio Piccoli, and Raimondo Schettini. 2018. *Anomaly Detection in Nanofibrous Materials by CNN-Based Self-Similarity*. Sensors 18 (1): 209–24.

[Ort20] Ortiz Castelló, Vicent, et al. 2020. *Real-Time on-Board Pedestrian Detection Using Generic Single-Stage Algorithms and on-Road Databases*. International Journal of Advanced Robotic Systems 17 (5).

[Pan19] Pang, Yanwei, and Jiale Cao. 2019. *Deep Learning in Object Detection*. In Deep Learning in Object Detection and Recognition, edited by Xiaoyue Jiang, Abdenour Hadid, and Yanwei Pang, 19–57. Singapore: Springer.

[Pen20] Pena-Caballero, Carlos, et al. 2020. *Real-Time Road Hazard Information System*. Infrastructures 5 (9): 75–93.

[Pos20] Poss, Christian, et al. 2020. *Enabling Robust and Autonomous Materialhandling in Logistics Through Applied Deep Learning Algorithms*. In Deep Learning Applications, edited by Bose, Moamar Sayed-Mouchaweh, Arif M. Wani, and Mehmed Kantardzic, 155–76. Singapore: Springer.

[Red13] Redmon, Joseph. 2013. *Darknet: Open Source Neural Networks in C*. Accessed March 03, 2021. https://pjreddie.com/darknet/.

[Red16] Redmon, Joseph, et al. 2016. *You Only Look Once: Unified, Real-Time Object Detection*. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–88.

[Ren15] Ren, Shaoqing, et al. 2015. *Faster R-CNN: Towards RealTime Object Detection with Region Proposal Networks*.

[Sam19] Samie, Farzad, Lars Bauer, and Jorg Henkel. 2019. *From Cloud down to Things: An Overview of Machine Learning in Internet of Things*. IEEE Internet of Things Journal 6 (3): 4921– 34.

[Say21] Sayagavi, Ashwini V., et al. 2021. *Deep Learning Methods for Animal Recognition and Tracking to Detect Intrusions*. In Information and Communication Technology for Intelligent Systems: Proceedings of ICTIS 2020, 617–26.

[Ser13] Sermanet, Pierre, et al. 2013. *OverFeat: Integrated Recognition, Localization and Detection Using Convolutional Networks*.

[She19] Shen, Li, et al. 2019. *Deep Learning to Improve Breast Cancer Detection on Screening Mammography*. Scientific Reports 9 (1): 12495–507.

[Su19] Su, Yongzhi, et al. 2019. *Deep Multi-Sate Object Pose Estimation for Augmented Reality Assembly*. In 2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct, 222–27.

DOI: 10.2195/lj_Proc_klos_en_202112_01
URN: urn:nbn:de:0009-14-54221

[Sul18] Sultani, Waqas, Chen Chen, and Mubarak Shah. 2018. *Real-World Anomaly Detection in Surveillance Videos*. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, 6479–88.

[Thi18] Thiel, Marko, Johannes Hinckeldeyn, and Jochen Kreutzfeldt. 2018. *Deep-Learning-Verfahren Zur 3D-Objekterkennung in Der Logistik*. Logistics Journal: Proceedings 2018.

[Ver16] Verma, Nishchal K., et al. 2016. *Object Identification for Inventory Management Using Convolutional Neural Network*. In 2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), 1–6.

[Vin14] Vincze, Markus, Sven Wachsmuth, and Gerhard Sagerer. 2014. *Perception and Computer Vision*. In The Cambridge Handbook of Artificial Intelligence, edited by Keith Frankish and William M. Ramsey, 168–90. Cambridge: Cambridge University Press.

[Xia20] Xiao, Youzi, et al. 2020. *A Review of Object Detection Based on Deep Learning*. Multimedia Tools and Applications 79: 23729–91.

[Yan20] Yang, Jing, et al. 2020. *Using Deep Learning to Detect Defects in Manufacturing: A Comprehensive Survey and Current Challenges*. Materials 13 (24): 5755–88.

[Yoo19] Yoo, Sunghwan, et al. 2019. *Prostate Cancer Detection Using Deep Convolutional Neural Networks*. Scientific Reports 9 (1).

[Zak20] Zaki, Pavly Salah, et al. 2020. *Traffic Signs Detection and Recognition System Using Deep Learning*.

[Žid19] Židek, Kamil, et al. 2019. *An Automated Training of Deep Learning Networks by 3D Virtual Models for Object Recognition*. Symmetry 11 (4): 496.

[Zou20] Zou, Lekang, Tanaka Yusuke, and Iba Hitoshi. 2020. *Dangerous Objects Detection of X-Ray Images Using Convolution Neural Network*. In Security with Intelligent Computing and Big-Data Services, edited by L. C. Jain, 714–28. Cham: Springer International Publishing.

M.Sc. Paolo Pagani is working as a Research Assistant at the chair of Robotics and Interactive Systems, Institute for Material Handling and Logistics (IFL), Karlsruhe Institute of Technology (KIT).
E-Mail: paolo.pagani@kit.edu

M.Sc. Matthias Elia Klos studied Industrial Engineering and Management Management at the Karlsruhe Institute of Technology (KIT).

Address: Institute for Material Handling and Logistics (IFL), Karlsruhe Institute of Technology (KIT), Gotthard-Franz-Str. 8, 76131 Karlsruhe, Germany.