AI-based Maize and Weeds detection on the edge with CornWeed Dataset

Naeem Iqbal*
DFKI
Plan-based robot control group
Osnabrueck, Germany.
naeem.iqbal@dfki.de

Christoph Manss*
DFKI
Marine Perception
Oldenburg, Germany.
christoph.manss@dfki.de

Christian Scholz†, Daniel König‡, Matthias Igelbrink§, Arno Ruckelshausen¶
Faculty of Engineering and Computer Science
University of Applied Sciences Osnabrueck
Osnabrueck, Germany.
†c.scholz@hs-osnabrueck.de, ‡philipp-daniel.koenig@hs-osnabrueck.de,
§matthias.igelbrink@hs-osnabrueck.de, ¶a.ruckelshausen@hs-osnabrueck.de

***Abstract*—Agricultural applications with AI methods are used more heavily and the lack of wifi connections on the fields make cloud services unavailable. Consequently, the AI models have to be processed directly on the edge. In this paper, we evaluate state-of-the-art detection algorithms for their use in agriculture, in particular plant detection. The current paper also presents the *CornWeed* data set, which has been recorded on land machines, showing labelled maize crops and weeds for plant detection. The paper provides accuracies for the state-of-the-art detection algorithms on the CornWeed data set, as well as FPS metrics for these networks on multiple edge devices. Moreover, for the FPS analysis, the detection algorithms are converted to ONNX and TensoRT engine files as they could be used as future standards for model exchange.**

***Index Terms*—plant detection, deep learning, agriculture, maize data, data acquisition, vision transformer**

I. Introduction

When it comes to smart agriculture on farm devices, the evaluation speed of obtained images plays a crucial role [2]. If the processing of the images is too slow, the farm device has to adjust its speed, which results in a lower efficiency. Object detection algorithms are already capable to provide object recognition at real-time speed. Especially neuronal networks are utilized for fast object detection, but the performance of a neuronal network - inference speed and accuracy - is influenced by its' structure and size, and this determines if the network can run on an edge device.

Object detectors that utilize bounding boxes can be categorized into one-stage and two-stage detectors. Two-stage detectors use first a heuristic to identify regions of interest and identify the object in this region. One-stage detectors do both tasks in a single network. One-stage detectors are therefore easier to train and are considered to be computationally faster than two-stage detectors [13], [17]. Two-stage detectors generally have a higher accuracy on the location information of the object and they identify smaller objects much better. For one-stage detectors this lower accuracy often origins from poor anchor boxes and the class imbalance problem. Recently, one-stage detectors with an anchor-less approach yielded higher accuracy for smaller objects [25]. This is useful for agricultural applications as plants need to be detected in early growth stages and as farm machines might have limited computational power. Nowadays also an new form of object detectors emerged - transformer networks for object detection [6]. Such networks tend to be large, but they yield high accuracies.

Yet, does it make sense to deploy algorithms directly on the farm machines? In [31], the authors discuss the importance of deploying algorithms directly on the farm machines for better responsiveness and reducing the load on cloud computing. On larger farmlands the network connection might be unreliable such that no cloud services might be reached. It might also be possible to use alternative sensors that are already available such as satellite images and drone imagery. These could be preprocessed before the field work. However, satellite imagery can only give guidance for larger patches of land and can not provide insightful information on individual plants due to limited geometrical resolution. Even the alternative of using drones prior to field cultivation or the application of herbicides, is not scaling well as presented in [4]. For example, drone imagery is expensive as it requires additional personal and it is most often limited to good weather. Thus, sensor data should be directly processed on the farm machine also because the capabilities of edge devices are increasing [32].

For example, in [26], the authors present an object detection algorithm for sugar beets that is able to detect the sugar beets and count their leaves based on RGB and NIR data. The data set is described in [8]. In [28], a robotic platform is presented that utilizes the detector from [26]. This system is able to distinguish weeds from crops such that it can destroy the weeds with a mechanical stamp. As this robot relies on the aforementioned object detector, the system requires RGB and

*Both authors contributed equally.

NIR data. However, often only RGB data is available.

In this paper we empirically evaluate typical object detection networks for their applicability on the edge for the detection of maize and weeds with RGB data. Because such networks require large amounts of data to be trained, we also present a data set which provides box labelled maize and weeds. The networks are then trained from scratch with the presented data set. Our contribution is therefore as follows:

- We present an agricultural dataset named CornWeed dataset where maize and weeds plants have been labelled for box object detection [*].
- We evaluate object detection algorithms with various neural network architectures based on their detection accuracy (mean average precision).
- Each of the detection algorithms are evaluated on a farm edge device based on a Nvidia Jetson Xavier NX and Jetson AGX Orin regarding their real-time capabilities (frames per second).

II. Data Set

A. Hardware Setup and Data Acquisition

For data acquisition, we utilized a previously designed sensor system [15]. This system comprises a computer, power supplies, and sensors. System and sensors communicate via ROS[†] such that data can be stored into ROSBags, see Fig.1. The benefits of this system is that it is sensor agnostic, i.e. any

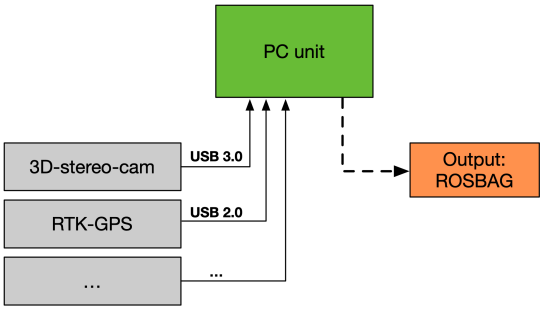

Fig. 1. System perspective of the utilized sensor system.

sensor can be integrated and connected. Here we utilized an Intel Realsense D435i (3D stereo camera) and an RTK-GPS receiver, as presented in Fig.1. For a robust and consistent data base, data collection was conducted using two different agricultural machines, an implement on a tractor and on a remotely steered research platform BoniRob [1], see Fig.2. In a first step, we integrated the sensor-system into BoniRob platform (Sensorbox 2) to evaluate optimal camera angles, heights, resolution, light conditions etc. on a small scale. In a second step, the sensor system (Sensorbox 1) was mounted on a conventional hoe with shifting frame and pulled through the field trials with a tractor. For this setup, based on the first data acquisition with Sensorbox 2 (640 x 480), the resolution of the

[*]Dataset Zenodo DOI 10.5281/zenodo.7961764

[†]Open Source Robotics Foundation. Robotic Operating System.https://www.ros.org

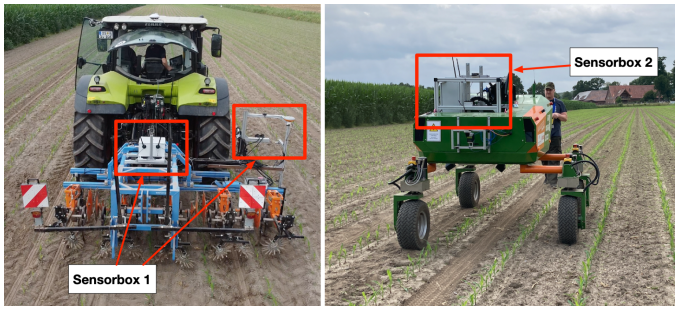

Fig. 2. Platforms for data acquisition. On the left an implementation on the tractor on a conventional hoe with shifting frame (Sensorbox 1). On the right the BoniRob with Sensorbox 2.

RGB camera on Sensorbox 1 was increased to 1280x720 pixel for a higher quality of the image data. Yet, both resolutions are kept in the data set for variability.

B. Data Variability

To represent different stages of growth and weed pressures, we conducted the field trials on multiple days. Therefore, the data samples were recorded over a period of three weeks to ensure different growth stages. Since the primary focus of the research was to root out the weeds early enough to ensure maximum crop growth, only the early growth stages were targeted for detection application since only at that time, weeds compete with the crop for resources (water, sunlight, etc.) and the crops eventually take over. Hence, later growth stages are less relevant for weeding applications. The data set only contains samples in the daylight with cloudy and sunny weather conditions, however, evening and early morning samples in future could be added to extend the domain knowledge for deep neural networks. The field trials always took place on the same field such that the same soil conditions and the same type of weeds occur in the data set.

C. Data Labelling

The number of detected weeds instances plays a crucial role for later selective weeding. To keep track of the number of detected objects, bounding boxes were chosen as the medium of annotation. The data set contains 3574 outdoor field images of *maize and weeds*, which are also the annotated classes in the data set. An example image of the data set for Sensorbox 1 with labels is displayed in Fig. 3. The annotations were generated by human annotators and reviewed by a different human reviewer. We used the open-source CVAT labelling tool [24] provided by Intel Corporation. The trained model on the data set can be further incorporated into this tool to further reduce the average labelling time. Thus, to speed up the process of annotation, intermediate object detectors have been trained during the annotation process with the interim data to provide proposal annotations. The annotator then fine tuned the proposed annotations by adding not detected weeds and maize, changing the class labels of false positives, and changing the sizes of the boxes. Such an interim detector can

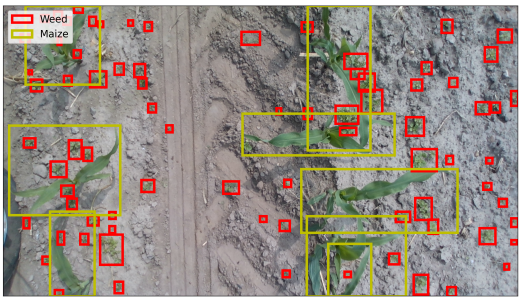

Fig. 3. An example image of the data set taken with the setup on the conventional hoe (Sensorbox 1). The images have resolutions of 640×480 with Sensorbox 2 and 1280×720 with Sensorbox 1.

also be provided by models trained on a synthetic training data as done by Naeem et. al. [12] in a similar use-case.

III. Detection algorithms

For a real-time detection scenario the accuracy is as important as the achievable detection rate. In the considered use-case of selective weeding, the movement speed of the farm device constraints the minimum frames per second (FPS). In our data acquisition setup, we used the Intel Realsense D435i camera with a vertical field of view (FOV) of 69 °. The camera was mounted at a height of 0.5 m, looking downwards. The geometric size of the obtained image covers a length of 0.68 m. To cover the whole ground with an average velocity of 8 km/h (2.2 m/s) we require at least 3-4 FPS. Higher frame-rates are of course desirable and would make the system more reliable. Given the low frame-rate requirement, two-stage detectors such as Faster-RCNN [23] can also be used since they have higher accuracy than single stage detectors as shown by Garcia and Mateo et. al. [7]. The authors show that while one-stage detectors are generally faster in inference speeds at lower image resolution, two-stage detectors outperform in terms of accuracy and detecting small objects in the image. This is specially relevant for the considered use-case here, since most of the weeds should be rooted out in the early growth stages before they start competing with the actual crop for resources.

This leads to the accuracy aspect because many of the weeds are small, which might lead to poor object detection performance. For example, anchor-based approaches [16], [22], [23] have difficulties to find very small objects in the image if the anchor boxes are not small enough. There are however object detectors that use an anchor-free approach [10], [25] and these are supposed to have a substantially better performance on small objects. More recently, object detectors based on transformer networks yield high accuracies in multiple applications [6]. Accordingly, for the considered use-case, we chose networks of the aforementioned categories. The networks are introduced in the following.

A. Faster R-CNN

R-CNN is a two-stage detector where the first stage produces region proposals that are then fed into the second stage where the object detection takes place. First versions of R-CNN have been published in 2014 [11] and the following versions have improved to be more computational effective and more accurate. In this paper, the considered version is the Faster RCNN [23]. This version uses a convolutional neural network (CNN) as backbone to identify feature maps, which are then sent to a region proposal network and a detection network.

B. RetinaNet

The RetinaNet [16] is a one-stage object detector that is based on the single-shot detector (SSD) [18] object detector. The main idea of SSD is that the detection requires information at different scales. Therefore this network pools directly from multiple convolutional layers, which are referred to as *convolutional predictors for detection*. This network utilizes default boxes and aspect ratios, which have to be determined beforehand. Each default box is then used for prediction on a grid on the image. As the number of predicted boxes can become large, hard negative mining is applied. Due to this only few candidate boxes are considered during the training of an SSD network, which is also known as the foreground-background class imbalance problem [20]. To address this problem, RetinaNet introduces the focal loss to put more emphasis on the hard training examples instead of easy ones. The authors showed that the focal loss substantially improves the performance of one-stage detectors.

C. FCOS

Another one-stage detector that does not use anchor boxes is the fully convolutional one-stage (FCOS) detector [25]. This detector does a pixel-wise detection and computes then a *center-ness* of each pixel according to the ground-truth boxes. The benefits of this are that the intersection over union (IoU), which is computationally expensive, does not need to be computed and that no anchor boxes are required. A downside of this approach is that in the detection ambiguities can occur as one pixel might be the center of multiple boxes. In such cases the larger box is ignored such that the detector has a better accuracy for smaller objects. For the use-case at hand, this is actually good as there are many small weeds.

D. YOLO

The you only look once detector, initially published in [21], has become very popular and has been extended in various aspects. The you only look once (YOLO)V5 is an efficient implementation [14] in Pytorch, which uses basically the same network as introduced in [3]. In this detector, the authors make excessive use of the so called *bag of freebies* – methods that only change the training strategy or the training cost – and *bag of specials* – methods of plugins that have a good performance to inference cost ratio. The bag of freebies are for example data augmentation methods that increase the robustness of the detector. The bag of specials on the other hand are spatial pyramid pooling, a spatial attention module, or other activation functions.

A successor of the YOLOV5 is the YOLOX [10], where the YOLO detector is re-designed into an anchor-free detector. Furthermore, YOLOX uses other advanced detection techniques, which would fall into the bag of specials, such as no default anchors or decoupled heads. However, we do use this version of the YOLO.

E. DINO Transformer

Most of the above object detection models require a prior knowledge of the task in the form of anchors (single stage) or proposals (two stage). The prior knowledge makes the model specialized to a specific task but loses performance when moved to a different detection task making transfer learning difficult. Carion et. al. [5] propose DEtection TRansformer (DETR) that is an end-to-end object detection transformer. With this architecture, there is no need to post process the bounding boxes or risk counting the same object twice due to its bipartite matching loss function. The authors combine the power of DETR with the following improvements: 1) Adding a noisy version of the ground truth labels during training to speed up the training process. 2) Mixed query selection 3) Box update based on the current layer and the next layer during back propogation. In Zhang et. al. [30], the authors argue that even though with prior knowledge, anchor based detectors still outperform the DETR.

IV. Experimental Setting

This section goes through the lifecycle of the neural networks:

1) Training a neural network and selecting the best variant from the *training pipeline*.
2) *Deployment pipeline* which explains how the network is optimized for a particular edge device to maximize performance throughput.
3) Edge devices used to evaluate inference speed of neural networks on an Agricultural use-case.

A. Training pipeline

For all the models mentioned above, very sophisticated Github repositories already exist. Thus, for the Faster RCNN, the Retinanet, and the FCOS, we used the Detectron2 repository from Meta [29]. For each of these models, we set the batch size to 32, the learning rate to 0.01, and the optimizer was stochastic gradient decent. For YOLOV5, we utilized the implementation of Ultralytics [14]. There, we set the batch size to 32 and 16 for YOLOV5m and YOLOV5l, respectively. Also, we specified the image size to be 800 pixel and to be rectangular to have comparable results with the other networks. During training, we also used the *multi-scale* option, where the image size is varied during training. For the DINO transformer, we used the Detrex research platform [9], which is based on the Detectron2 repository. Due to the size of the DINO transformer, we had to set the batch size to 4. The learning rate was set to 0.0001. All other parameters of the algorithms were left to the default values.

We trained each model on a NVidia Tesla V100 DGXS 32GB GPU with the CUDA Version 11.1.

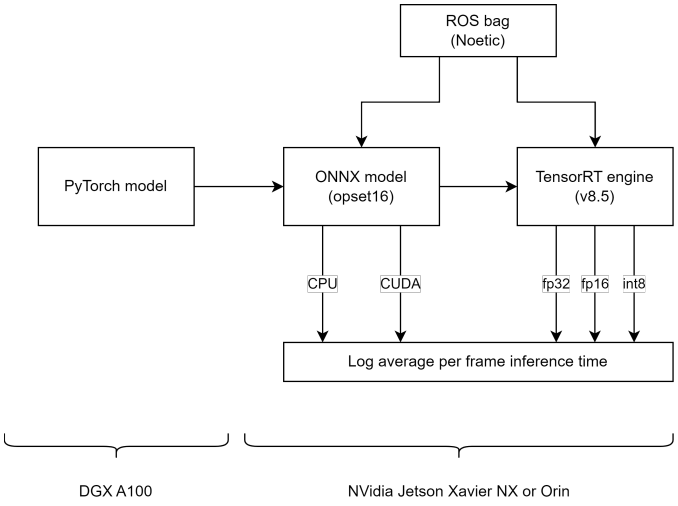

Fig. 4. Deployment workflow showing how the models are trained in PyTorch, converted to ONNX models, and then to TensorRT engine files.

B. Deployment pipeline

After training, the best model file is selected based on the validation data split and then converted to an open neural network exchange (ONNX) model. ONNX is a framework that optimizes and acts as an intermediate representation of neural networks to support conversion to any standard frameworks such as PyTorch, TensorFlow, OpenVINO, TensorRT, etc. However, a user can use this intermediate representation also directly. In this paper, both the ONNX models and TensorRT models are evaluated to highlight the impact of framework choices. The ONNX models of the DINO transformer and YOLO network were then converted to a TensorRT engine file with precision of 16-bit and 32-bit floating points and 8-bit integer. The different precision can make the model more memory efficient and also advanced build-in function can be used for faster computation [19]. The other models consist of the layers that can not be optimized by the TensorRT engine (up until version 8.5 [‡]), hence failing to do the conversion to a TensorRT engine. At the time of this publication, TensorRT version 8.5 was available.

C. Edge Device

For all the experiments in this paper, two NVIDIA Jetson devices were used:

1) the Jetson Xavier NX and
2) the Jetson AGX Orin.

Both edge devices have L4T 35.3.1 with Ubuntu 20.04, ROS Noetic, CUDA 11.4 running on them. Jetson devices come with predefined power modes utilizing varying number of on-board CPUs and number of online CPU cores. For the experiments shown in table II, the Xavier NX was set on mode ID 6 with all cores online and 1400 MHz CPU frequency. In this mode, the Xavier NX board consumes about 20 W

[‡]We are hopeful that with upcoming TensorRT 8.6, the performance metrics can be updated for remaining networks in table II

of power. Similarly, we activated the MAXN power profile on the AGX Orin to utilize all GPU and CPU cores and to remove clock restrictions. In this power profile, the AGX Orin consumes about 60 W. The input to the model is fed in the form of image sequences via a ROS Bag. Each consecutive image sample is read from the ROS Bag, and downscaled and normalized according to the input size expected by the model. The performance metrics are logged from the moment an image is received until the detections are ready to be published into the ROS ecosystem as detection messages.

V. Results

This sections presents the accuracy results of the presented networks for the data set and the speed results on the considered edge devices.

A. Training Results

The mean average precision (mAP) of the trained networks for two confidence values is presented in Tab. I. The YOLOV5 networks clearly have the best performance. However, the YOLOV5 repository already offers many variations and augmentations to the input data, see bag freebies in Sec. III-D. These bag of freebies are also optimized during training by the code in the repository. The networks based on Detectron2 on the other hand don't have such an optimization. Thus, if only the networks based on the Detectron2 repository are compared with each other, the DINO transformer stands out. Yet, the DINO transformer still gives lower accuracy on weeds. This is due to its inability to detect tiny objects such as weeds which can be in some instances only be a few pixels. This behaviour was first brought to light by Carion et. al. [5]. If the one-stage networks of the Detectron2 repository are compared with the two-stage detector Faster RCNN, the RetinaNet reached a better performance on detecting maize.

TABLE I
MEAN AVERAGE PRECISION OF ALL OBJECT DETECTORS AT VARIOUS CONFIDENCE VALUES AND CLASS IDS OF MAIZE AND WEEDS

Model	mAP50	mAP50:95	mAP50 Maize	mAP50 Weeds
FCOS R50 FPN	69.3	39.8	87.0	51.7
RetinaNet R50 FPN	68.0	40.2	89.7	46.3
Faster RCNN FPN	71.6	41.8	88.1	55.1
YOLOv5 medium	85.4	53.3	93.0	**77.8**
YOLOv5 large	**85.4**	**53.9**	**93.7**	77.0
DINO Transformer	75.8	45.0	92.3	59.3

B. Speed Results

As described in section IV-B, the trained models are optimized and then deployed for both NVIDIA Jetson Xavier NX and Jetson AGX Orin. The average FPS time was logged based on the input image which was feed from a ROS bag recorded with Intel RealSense D435i camera for testing purposes. For anchor-based implementations (e.g. RetinaNet, YOLO), the non-maximum suppression is also part of the inference time. This creates a fair comparison between anchor-based and anchor-free models. The corresponding FPS metrics are shown in table II. For the first three models (FCOS, RetinaNet and Faster RCNN), the model contains layers that cannot be converted into the TensorRT engine in v8.5. With the next release of TensorRT v8.6 accompanied with the new Jetpack release, these model should also be convertible to a TensorRT engine. Looking at the numbers, in general, the ONNX inference framework gives very low inference rates, even with using *CUDA execution provider*, making it not suitable for application purposes. However, the ONNX model serves as a good intermediate model representation that can be later converted to any other inference framework such as TensorFlow, TensorRT, or PyTorch.

The TensorRT engine files with different floating point and integer precisions yield higher FPS in general. The 8-bit integer precision yields the highest inference rate for each network. When comparing different networks, YOLOV5 gives the highest FPS. Looking at the overall accuracy in table I, the large variant of the YOLOv5 gives higher mAP values for the maize class and in general. When comparing the accuracy vs the inference rate, the YOLOV5 medium version has the same accuracy as the YOLOV5 large variant. This implies that increasing the model size does not necessarily lead to improved performance while the medium variant gives more inference speed on the edge device. On the other hand, the DINO transformer provides high mAP50 on the maize class, but surprisingly lower values on weed class [§]. According to [5], the authors conclude that a transformer architecture has lower performance on tiny object detection which is consistent with our findings. While vision transformers usually outperform most networks, it may not always give the best performance depending on the size of the objects. The GPU memory size and the input image resolution also plays a critical role in deciding for the neural network architecture.

VI. Conclusion

This paper shows how state-of-the-art object detectors perform on an agricultural specific use-case on different edge systems. With an accompanying data set, some neural network characteristics are highlighted such as low accuracy on detecting tiny objects which is a common challenge in agricultural perception tasks. These shortcomings are not immediately visible when training on a different domain data set such as autonomous driving. The model deployment pipeline is also included for readers who embark on a different use-case and want to optimize their model's inference speed. Generally speaking, it is always better to convert a well trained model to an edge device specific engine such as the TensorRT engine. This way the FPS can increase by a factor of four. Thus, using TensorRT yields faster networks, not only in the in the agricultural domain but also other domains, e.g. [27].

[§]The DINO transformer was set to lower resolution of 512 x 683 because otherwise it does not fit onto Jetson Xavier NX with 8GB VRAM. For a fair comparison between Xavier NX and AGX Orin, it was set to that resolution.

TABLE II

AVERAGE INFERENCE RATE (IN FRAMES PER SECOND) FOR ALL THE DETECTORS FROM TABLE I ON EDGE DEVICES WITH ONNX *CUDA Execution Provider* (CEP) AND TENSORRT. THE INPUT IMAGE RESOLUTION WAS 800 X 1067 FOR FIRST THREE NETWORKS (FCOS, RETINANET, FRCNN), 800 X 1088 FOR THE BOTH YOLO NETWORKS AND 512 X 683 FOR DINO TRANSFORMER.THE VALUES MARKED WITH * WERE DUE TO A TENSORRT BUG NOT UTILIZING CUDA WITH DINO MODEL.

Framework	**ONNX with CEP**		**TensorRT**					
Edge device	**Jetson Xavier NX**	**Jetson AGX Orin**	**Jetson Xavier NX**			**Jetson AGX Orin**		
			int8	**fp16**	**fp32**	**int8**	**fp16**	**fp32**
FCOS R50 FPN	1.30	4.0	X	X	X	X	X	X
RetinaNet R50 FPN	1.37	4.3	X	X	X	X	X	X
Faster RCNN FPN	0.62	1.8	X	X	X	X	X	X
YOLOv5 medium	**2.5**	**6.0**	**19.3**	**12**	**3.7**	**51**	**33.5**	**16.8**
YOLOv5 large	1.4	4.0	12.7	6.5	1.65	37	22	10.9
DINO Transformer	0.24	0.55	1.35*	1.47*	0.86*	3.2*	3.1*	3*

VII. ACKNOWLEDGEMENTS

The DFKI Lower Saxony (DFKI NI) is funded by the Lower Saxony Ministry of Science and Culture and the Volkswagen Foundation. The project Agri-GAIA on which this report is based was funded by the German Federal Ministry for Economics and Climate Action under the funding code 01MK21004A: Responsibility for the content of this publication lies with the author. Many thanks to Hof Langsenkamp, Belm, Germany for providing the fields for data acquisition and the used hardware (tractor and implement). We would like to thank Oliver Zielinski for mainly writing on the parts of the proposal that lead to this work. We would also like to thank our labellers: Qalab Abbas, Jule Fröhlich, Novruz Mammadli, Charles Lennart Müller, Dibyashree Nahak, Turgut Nasrullayev, and Simon Zielinski.

REFERENCES

[1] W Bangert, A Kielhorn, F Rahe, A Albert, P Biber, S Grzonka, S Haug, A Michaels, D Mentrup, M Hänsel, et al. Field-robot-based agriculture:"remotefarming. 1" and "bonirob-apps". In *71th conference LAND. TECHNIK-AgEng 2013*, pages 439–446, 2013.

[2] Lefteris Benos, Aristotelis C. Tagarakis, Georgios Dolias, Remigio Berruto, Dimitrios Kateris, and Dionysis Bochtis. Machine Learning in Agriculture: A Comprehensive Updated Review. *Sensors*, 21(11):3758, January 2021.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv:2004.10934 [cs, eess]*, April 2020.

[4] Enyu Cai, Sriram Baireddy, Changye Yang, Melba Crawford, and Edward J. Delp. Deep Transfer Learning for Plant Center Localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 62–63, 2020.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020.

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[7] Manuel Carranza-García, Jesús Torres-Mateo, Pedro Lara-Benítez, and Jorge García-Gutiérrez. On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data. *Remote Sensing*, 13(1):89, December 2020.

[8] Nived Chebrolu, Philipp Lottes, Alexander Schaefer, Wera Winterhalter, Wolfram Burgard, and Cyrill Stachniss. Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields. *The International Journal of Robotics Research*, 36(10):1045–1052, September 2017.

[9] detrex contributors. detrex: An research platform for transformer-based object detection algorithms. https://github.com/IDEA-Research/detrex, 2022.

[10] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: Exceeding YOLO Series in 2021, August 2021.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, Columbus, OH, USA, June 2014. IEEE.

[12] Naeem Iqbal, Justus Bracke, Anton Elmiger, Hunaid Hameed, and Kai von Szadkowski. Evaluating synthetic vs. real data generation for ai-based selective weeding. In Christa Hoffmann, Anthony Stein, Arno Ruckelshausen, Henning Müller, Thilo Steckel, and Helga Floto, editors, *43. GIL-Jahrestagung, Resiliente Agri-Food-Systeme*, pages 125–135, Bonn, 2023. Gesellschaft für Informatik e.V.

[13] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A Survey of Deep Learning-based Object Detection. *IEEE Access*, 7:128837–128868, 2019.

[14] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael, Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, Adam Hogan, Cristi Fati, Lorenzo Mammana, AlexWang1900, Deep Patel, Ding Yiwei, Felix You, Jan Hajek, Laurentiu Diaconu, and Mai Thanh Minh. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference, February 2022.

[15] Daniel König, Matthias Igelbrink, Christian Scholz, and Andreas Linz. Entwicklung einer flexiblen Sensorapplikation zur Erzeugung von validen Daten für KI-Algorithmen in landwirtschaftlichen Feldversuchen. In *42. GIL-Jahrestagung, Künstliche Intelligenz in der Agrar- und Ernährungswirtschaft*, pages 165–170, Bonn, 2022. Gesellschaft für Informatik e.V.

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[17] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, 128(2):261–318, February 2020.

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 21–37, Cham, 2016. Springer International Publishing.

[19] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S. Vetter. NVIDIA Tensor Core Programmability, Performance & Precision. In *2018 IEEE International Parallel and Distributed*

Processing Symposium Workshops (IPDPSW), pages 522–531, May 2018.

[20] Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Akbas. Imbalance Problems in Object Detection: A Review. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 1–1, 2020.

[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *arXiv:1506.02640 [cs]*, May 2016.

[22] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv:1804.02767 [cs]*, April 2018.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497 [cs]*, January 2016.

[24] Boris Sekachev, Nikita Manovich, Maxim Zhiltsov, Andrey Zhavoronkov, Dmitry Kalinin, Ben Hoff, TOsmanov, Dmitry Kruchinin, Artyom Zankevich, DmitriySidnev, Maksim Markelov, Johannes222, Mathis Chenuet, a andre, telenachos, Aleksandr Melnikov, Jijoong Kim, Liron Ilouz, Nikita Glazov, Priya4607, Rush Tehrani, Seungwon Jeong, Vladimir Skubriev, Sebastian Yonekura, vugia truong, zliang7, lizhming, and Tritin Truong. opencv/cvat: v1.1.0, August 2020.

[25] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully Convolutional One-Stage Object Detection. *arXiv:1904.01355 [cs]*, August 2019.

[26] Jan Weyler, Andres Milioto, Tillmann Falck, Jens Behley, and Cyrill Stachniss. Joint Plant Instance Detection and Leaf Count Estimation for In-Field Plant Phenotyping. *IEEE Robotics and Automation Letters*, 6(2):3599–3606, April 2021.

[27] Mattis Wolf, Katelijn van den Berg, Shungudzemwoyo P. Garaba, Nina Gnann, Klaus Sattler, Frederic Stahl, and Oliver Zielinski. Machine learning for aquatic plastic litter detection, classification and quantification (APLASTIC-Q). *Environmental Research Letters*, 15(11):114042, November 2020.

[28] Xiaolong Wu, Stéphanie Aravecchia, Philipp Lottes, Cyrill Stachniss, and Cédric Pradalier. Robotic weed control using automated weed and crop classification. *Journal of Field Robotics*, 37(2):322–340, 2020.

[29] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[30] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.

[31] X. Zhang, Z. Cao, and W. Dong. Overview of Edge Computing in the Agricultural Internet of Things: Key Technologies, Applications, Challenges. *IEEE Access*, 8:141748–141761, 2020.

[32] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang. Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing. *Proceedings of the IEEE*, 107(8):1738–1762, August 2019.