Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 577–584
DOI: 10.15439/2023F2125
ISSN 2300-5963 ACSIS, Vol. 35

AI-based Maize and Weeds Detection on the Edge with CornWeed Dataset

Naeem Iqbal*
DFKI
Plan-based Robot Control
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*—Artificial intelligence (AI) is used more heavily in agricultural applications. Yet, the lack of wireless-fidelity (Wi-Fi) connections on agricultural fields makes AI cloud services unavailable. Consequently, 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. Thus, this paper presents the *CornWeed* data set, which has been recorded on farm 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 frames per second (FPS) metrics for the considered networks on multiple edge devices. Moreover, for the FPS analysis, the detection algorithms are converted to open neural network exchange (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 [1]. If the processing of the images is too slow, the farm device has to adjust its speed, which results in a lower time efficiency. Object detection algorithms are already capable to provide object recognition at real-time speed. Especially neural networks are utilized for fast object detection, but the performance of a neural network - inference speed and accuracy - is influenced by its structure and size which determines if the network can run on an edge device.

Often the desired detection is marked by a bounding box which surrounds the identified object. Object detectors that use bounding boxes can be categorized into one-stage and two-stage detectors. Two-stage detectors first identify regions of interest using a heuristic and, then, detect 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 [2], [3]. 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 originates from poor anchor boxes and the class imbalance problem. Recently, one-stage detectors with an anchor-less approach yielded better accuracy for smaller objects [4]. 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. Also, nowadays an new form of object detectors emerged - transformer networks for object detection [5]. Such networks tend to be large, but they yield high accuracies.

Yet, does it make sense to deploy algorithms directly on farm machines? In [6], 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 are reachable. 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 geometric resolution [7]. Even the alternative of using drones prior to field cultivation or the application of herbicides, is not scaling well as presented in [8]. For example, drone imagery is expensive, as it requires additional personnel, and it is most often limited to good weather [9]. Moreover, the collected information can be outdated by a few days or even a week. For weeding applications, these delays can be critical because weeds can be growing fast. Thus, sensor data should be directly processed on the farm machine especially because the capabilities of edge devices are increasing [10].

For example, in [11], the authors present an object detection algorithm for sugar beets that is able to detect the sugar beets and count their leaves based on red, green, and blue (RGB) and

*Both authors contributed equally.

near infra red (NIR) data. The data set is described in [12]. In [13], a robotic platform is presented that utilizes the detector from [11]. 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 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 that 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 (mAP)).
- Each of the detection algorithms is evaluated on farm edge devices 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 [14]. This system comprises a computer, power supplies, and sensors. System and sensors communicate via the robot operating system (ROS)† such that data can be stored into *ROS Bags* (see Fig.1) which is a ROS specific data format for time-dependent data. The benefit of this system is

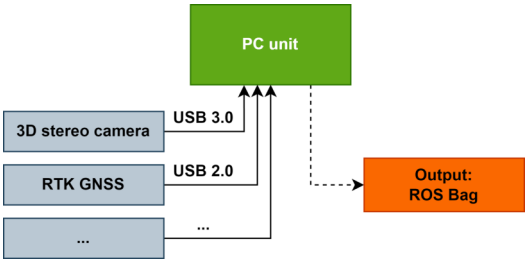

Fig. 1. System perspective of the utilized sensor system.

that it is sensor agnostic, i.e. any sensor can be integrated and connected. Here we used an Intel Realsense D435i (3D stereo camera) and a real time kinematic (RTK) enabled global navigation satellite systems (GNSS) 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 [15], see Fig. 2. In a first step, we integrated

*Dataset Zenodo DOI 10.5281/zenodo.7961764

†Open Source Robotics Foundation https://www.ros.org, accessed on 25th of June, 2023

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.

the sensor system into the 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 a 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 pixel), the resolution of the 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. In both sensor setups, the Intel Realsense D435i camera with a vertical field of view (FOV) of 69 ° was mounted at a height of 0.5 m, looking downwards. Therefore, the geometric size of the each obtained image spans a distance of 0.68 m along the driving direction.

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. Here, the primary focus of the application was to root out the weeds early enough to ensure maximum crop growth. Thus, only the early growth stages of maize crops were considered for the detection application because only at that time crops compete with weeds for resources (water, sunlight, etc.) and otherwise the crops outgrow the weeds. Hence, later growth stages of maize 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 types of weeds persist throughout the data set.

C. Data Labelling

The number of detected weeds instances plays a crucial role for 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

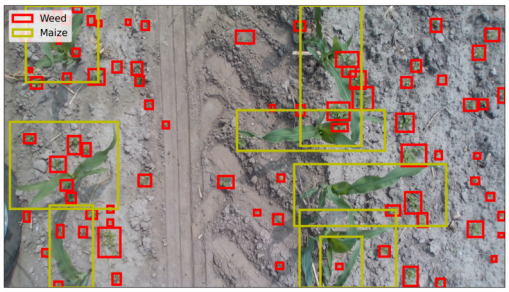

Fig. 3. An example image of the data set taken with the setup on the conventional hoe (Sensorbox 1). The images have resolutions of 640×480 pixel with Sensorbox 2 and 1280×720 pixel with Sensorbox 1. Here, the labelled instances of Maize are shown in yellow and the weed instances are shown in red.

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 computer vision annotation tool (CVAT) labelling tool [16] provided by CVAT.ai corporation. The model trained on the data set can be subsequently 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, adjusting the class labels of false positives, and changing the sizes of the boxes. Such an interim detector can also be provided by models trained on a synthetic training data as done by Naeem et. al. [17] for 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). 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 region-based convolutional neural network (R-CNN) [18] can be used also as they have higher accuracy than single stage detectors as shown by Garcia and Mateo et. al. [19]. 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 especially relevant for the considered use-case here, since most of the weeds should be rooted out in the early growth stages, when they are small, before they start competing with the actual crop for resources.

This leads to an accuracy aspect: while weeds are small, detectors might have poor object detection performance. For example, anchor-based approaches [18], [20], [21] 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 [4], [22] 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 [5]. Accordingly, for the considered use-case, we chose networks of the aforementioned categories. The networks are introduced in the following subsections.

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 [23] and the following versions have improved to be more computationally effective and more accurate. In this paper, the considered version is the Faster R-CNN [18]. 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 [21] is a one-stage object detector that is based on the single-shot detector (SSD) [24]. 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 [25]. 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 [4]. 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 (YOLO) detector, initially published in [26], has become very popular and has been extended in various aspects. It is a single-stage detector that outputs class probabilities and bounding box coordinates in a single step that are filtered with non-maximum suppression. The YOLOv5 is an efficient implementation [27] in PyTorch, which uses basically the same network as introduced in [28]. 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.

YOLOv5 comes with many variants with different model layers and backbones. For the scope of this paper, we only use YOLOv5 medium and large variants of the YOLO architecture.

E. DINO Transformer

Most of the above object detection models require a prior knowledge of the task in the form of anchors (one-stage) or region 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. [29] 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. Following the DETR architecture, Zhang et. al. [30] came up with an improved variant of DETR called DETR with improved denoising anchor boxes (DINO) transformer. Zhang et. al. proposed the following improvements to the DETR architecture: 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. Despite the recent trends in the transformer-based detector architectures, Carion et. al. pointed out the reduced performance of DETR in detecting small objects in the image, while Zhang et. al. argued that the anchor-based approaches still show superior accuracy compared to the transformers. For our experiments, we used the DINO transformer to represent the transformer-based family of object detectors.

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, official Github repositories already exist [31] [32] [27]. Thus, for the Faster R-CNN, the RetinaNet, and the FCOS, we used the Detectron2 repository from Meta [31]. 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 [27]. There, we set the batch size to 32 and 16 for YOLOv5 medium and YOLOv5 large variants, 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 [32], 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. Also, we did not tune the augmentations if any are used by the model.

We trained each model on a NVidia Tesla V100 DGXS 32GB GPU with the CUDA Version 11.1, and used a train-validation split of 80% (2862 images) and 20% (712 images), respectively.

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 also use this intermediate representation directly. In this paper, both the ONNX models and TensorRT models are evaluated to highlight the impact of framework choices. The code repository for testing all the trained models is provided at this link [‡]. 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 [33]. 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.

[‡]Inference speed testing: https://github.com/niqbal996/deployment_testing

[§]We are hopeful that with upcoming TensorRT 8.6, the performance metrics can be updated for remaining networks in Table V-A

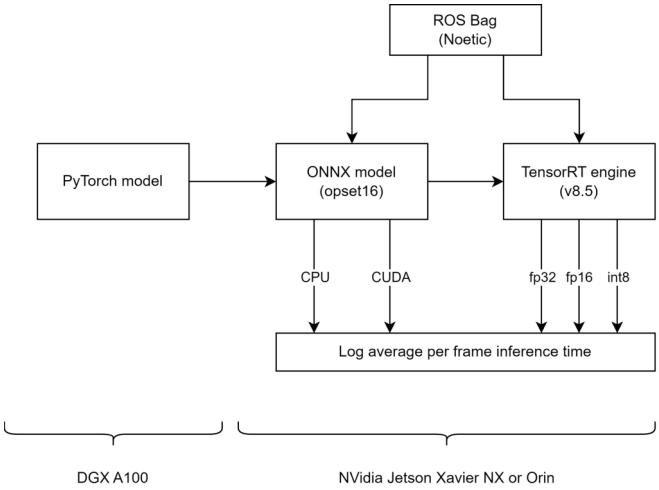

Fig. 4. Deployment workflow showing how the models are trained in PyTorch, converted to open neural network exchange (ONNX) models, and then to TensorRT engine files.

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 a varying number of on-board CPUs and online CPU cores. For the experiments shown in Table V-A (see later in Sec. V-B), the Xavier NX was set to mode ID 6 with all cores online and 1400 MHz CPU frequency. In this mode, the Xavier NX board consumes about 20 W 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*, 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 section presents the accuracy results of the presented networks for the data set and the speed results on the considered edge devices.

A. Training Results

Table I shows the mAP with the mean over both classes (namely *Maize* and *Weeds*) at 50 % confidence value i.e. mAP50 in the first column. The second column shows cumulative average of mAP varied from 50 % to 95 % with step size of 5 % i.e. mAP50:95. In the last two columns, we show the average precision (AP) per class at 50 % confidence to show the accuracy trend on larger sized *maize* objects and smaller sized *weed* objects.

TABLE I
Mean average precision (mAP) of all object detectors at various confidence values and class IDs for Maize and weeds

Model	mAP50	mAP50:95	AP50 Maize	AP50 Weeds
Faster R-CNN FPN	71.6	41.8	88.1	55.1
RetinaNet R50 FPN	68.0	40.2	**89.7**	46.3
FCOS R50 FPN	69.3	39.8	87.0	51.7
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

Across all networks, the YOLOv5 networks have the highest performance of 85.4 and 53.9 for mAP50 and mAP50:95, respectively. The larger variant YOLOv5 large does not proportionally increase the performance while being much larger in size than YOLOv5 medium. This implies that the accuracy has already saturated and increasing network size does not always necessarily lead to increased performance. In terms of comparison, the YOLOv5 repository already offers many variations and augmentations to the input data, see bag freebies in Sec. III-D. This bag of freebies is also optimized during training by the code in the repository. The networks based on Detectron2 do not have such an optimization. Thus, if only the networks based on the Detectron2 repository are compared with each other, the DINO transformer stands out in detecting maize but still gives a lower accuracy of 59.3 for AP50 for weeds compared with YOLOv5. This is due to its reduced capability to detect tiny objects such as weeds which can be in some instances only be a few pixels in size. This behaviour was first brought to light by Carion et. al. [29] for the detection transformers.

If the one-stage networks of the Detectron2 repository are compared with the two-stage detector Faster R-CNN, the RetinaNet reached a better performance of 89.7 on detecting maize but surprisingly a lower AP of 46.3 on weeds. This higher accuracy of 89.3 than 88.1 from Faster R-CNN is due to the usage of Focal loss proposed by [21]. While focal loss helps in outperforming a two-stage detector such as Faster R-CNN, the AP for weeds is still lower. We believe the lower accuracy of 46.3 on weeds to be an outlier due to anchor boxes not modified to the task of weed detection prior to training. Comparing RetinaNet with YOLOv5, the Detectron2 framework does not provide any flexibility to change the anchor boxes to the task of maize or weed detection prior to training. This leads to reduced performance. Ahmed et. al. [34] observed similar behaviour from RetinaNet in detecting tiny objects from aerial images and used anchor optimization to mitigate that.

To achieve the best possible accuracy from an anchor-based detector architecture, the anchor boxes have to be modified to the specific use-case. Task specific modification of anchor

TABLE II
AVERAGE INFERENCE RATE (IN FRAMES PER SECOND (FPS)) 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 PIXEL FOR THE THREE NETWORKS (FCOS, RETINANET, FASTER R-CNN), 800 X 1088 PIXEL FOR BOTH YOLO NETWORKS, AND 512 X 683 PIXEL FOR THE DINO TRANSFORMER. THE VALUES MARKED WITH A * WERE DUE TO A TENSORRT BUG WHERE THE DINO MODEL NOT UTILIZED CUDA.

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**
Faster R-CNN FPN	0.62	1.8	X	X	X	X	X	X
RetinaNet R50 FPN	1.37	4.3	X	X	X	X	X	X
FCOS R50 FPN	1.30	4.0	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*

boxes can be crucial in choosing optimum network architecture for any agricultural use-case where each crop has different size in different growth stages. DINO Transformer, FCOS and other anchor-less implementations reduce the complexity of the task by not having to provide any prior knowledge or anchor optimization before training while giving slightly lower accuracy in general.

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 were logged based on the input image which was fed 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 V-A. For the first three models (Faster R-CNN, RetinaNet, and FCOS), the model contains layers that cannot be converted into a TensorRT engine with version 8.5. With the next release of TensorRT v8.6 accompanied with the new Jetpack release, these models should also be convertible to a TensorRT engine. Comparatively looking at the numbers, in general, the ONNX inference framework gives lower inference rates than TensorRT, even when using *CUDA execution provider*. 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, see Table V-A. For example, YOLOv5 medium yields 33.5 FPS on fp16 precision via TensorRT compared to a mere 6 FPS via ONNX on Jetson Orin. Especially, the 8-bit integer precision yields the highest inference rate for each network, about 6-9 times faster compared with the ONNX models. When comparing different networks, YOLOv5 gives the highest FPS.

Though the model conversion from PyTorch or Tensorflow to ONNX or TensorRT comes with its own challenges. Depending upon the layers that constitute a particular model, not all layer components are easily convertible to an TensorRT engine. While some architectures are easily transferrable to a TensorRT engine, other model architectures while giving more accuracy may be more difficult in transferring to a TensorRT engine. This makes the model deployment to an edge device more difficult and impacts the choice of model architecture.

When comparing the accuracy (see Table I) versus the inference rate (see Table V-A), the YOLOv5 medium version has the same accuracy as the YOLOv5 large variant but has more FPS e.g., 33.5 versus 22 FPS on Jetson Orin (fp16). 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[¶]. While vision transformers usually outperform most networks, they 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 multiple one-stage object detectors (RetinaNet, FCOS, YOLO), a two-stage object detector (Faster R-CNN), and a transformer object detector (DINO) perform on an agricultural use-case on different edge systems. With an accompanying data set, we found out that the YOLOv5 object detector performed well on detecting maize and reasonably well on detecting tiny objects such as small weeds. For the other neural networks, characteristics are highlighted such as a 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

[¶]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.

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 factors up to 8, if additionally the integer precision is reduced. Thus, using TensorRT yields faster networks, not only in the agricultural domain but also other domains, e.g. [35]. Inference framework-wise, ONNX has a relatively simplified model conversion process that works with most of the models and transferable across edge devices with different GPU architecture. By comparison, TensorRT has a more complicated GPU-specific model conversion process e.g. Jetson Orin needs a separate engine than Jetson Xavier and does not work successfully on all models.

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.

VIII. Acronyms

CNN	convolutional neural network
R-CNN	region-based convolutional neural network
Wi-Fi	wireless-fidelity
SSD	single-shot detector
YOLO	you only look once
FCOS	fully convolutional one-stage
FPS	frames per second
FOV	field of view
IoU	intersection over union
mAP	mean average precision
ONNX	open neural network exchange
AI	artificial intelligence
DETR	detection transformer
DINO	DETR with improved denoising anchor boxes
CVAT	computer vision annotation tool
RTK	real time kinematic
GNSS	global navigation satellite systems
NIR	near infra red
RGB	red, green, and blue
ROS	robot operating system
AP	average precision

References

[1] L. Benos, A. C. Tagarakis, G. Dolias, R. Berruto, D. Kateris, and D. Bochtis, "Machine Learning in Agriculture: A Comprehensive Updated Review," *Sensors*, vol. 21, no. 11, p. 3758, Jan. 2021, DOI:10.3390/s21113758.

[2] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A Survey of Deep Learning-based Object Detection," *IEEE Access*, vol. 7, pp. 128 837–128 868, 2019.

[3] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep Learning for Generic Object Detection: A Survey," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 261–318, Feb. 2020.

[4] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9626–9635.

[5] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[6] X. Zhang, Z. Cao, and W. Dong, "Overview of Edge Computing in the Agricultural Internet of Things: Key Technologies, Applications, Challenges," *IEEE Access*, vol. 8, pp. 141 748–141 761, 2020, DOI:10.1109/ACCESS.2020.3013005.

[7] M. Weiss, F. Jacob, and G. Duveiller, "Remote sensing for agricultural applications: A meta-review," *Remote Sensing of Environment*, vol. 236, p. 111402, 2020, DOI:10.1016/j.rse.2019.111402. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425719304213

[8] E. Cai, S. Baireddy, C. Yang, M. Crawford, and E. J. Delp, "Deep transfer learning for plant center localization," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 277–284, DOI:10.1109/CVPRW50498.2020.00039.

[9] F. López-Granados, "Weed detection for site-specific weed management: mapping and real-time approaches," *Weed Research*, vol. 51, no. 1, pp. 1–11, 2011, DOI:10.1111/j.1365-3180.2010.00829.x.

[10] 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*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019, DOI:10.1109/JPROC.2019.2918951.

[11] J. Weyler, A. Milioto, T. Falck, J. Behley, and C. Stachniss, "Joint Plant Instance Detection and Leaf Count Estimation for In-Field Plant Phenotyping," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3599–3606, Apr. 2021, DOI:10.1109/LRA.2021.3060712.

[12] N. Chebrolu, P. Lottes, A. Schaefer, W. Winterhalter, W. Burgard, and C. Stachniss, "Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1045–1052, Sep. 2017.

[13] X. Wu, S. Aravecchia, P. Lottes, C. Stachniss, and C. Pradalier, "Robotic weed control using automated weed and crop classification," *Journal of Field Robotics*, vol. 37, no. 2, pp. 322–340, 2020, DOI:10.1002/rob.21938.

[14] D. König, M. Igelbrink, C. Scholz, A. Linz, and A. Ruckelshausen, "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*. Bonn: Gesellschaft für Informatik in der Land-, Forst- und Ernährungswirtschaft e.V., 2022, pp. 165–170.

[15] 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*, 2013, pp. 439–446.

[16] B. Sekachev, N. Manovich, M. Zhiltsov, A. Zhavoronkov, D. Kalinin, B. Hoff, TOsmanov, D. Kruchinin, A. Zankevich, DmitriySidnev, M. Markelov, Johannes222, M. Chenuet, a andre, telenachos, A. Melnikov, J. Kim, L. Ilouz, N. Glazov, Priya4607, R. Tehrani, S. Jeong, V. Skubriev, S. Yonekura, vugia truong, zliang7, lizhming, and T. Truong, "opencv/cvat: v1.1.0," Aug. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4009388

[17] N. Iqbal, J. Bracke, A. Elmiger, H. Hameed, and K. von Szadkowski, "Evaluating synthetic vs. real data generation for ai-based selective weeding," in *43. GIL-Jahrestagung, Resiliente Agri-Food-Systeme*, C. Hoffmann, A. Stein, A. Ruckelshausen, H. Müller, T. Steckel, and H. Floto, Eds. Bonn: Gesellschaft für Informatik e.V., 2023, pp. 125–135.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv:1506.01497 [cs]*, Jan. 2016, DOI:10.48550/arXiv.1506.01497.

[19] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, and J. García-Gutiérrez, "On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data," *Remote Sensing*, vol. 13, no. 1, p. 89, Dec. 2020, DOI:10.3390/rs13010089.

[20] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv:1804.02767 [cs]*, Apr. 2018, DOI:10.48550/arXiv.1804.02767.

[21] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020, DOI:10.1109/TPAMI.2018.2858826.

[22] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," Aug. 2021, DOI: 10.48550/arXiv.2107.08430.

[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587, DOI: 10.1109/CVPR.2014.81.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, ser. Lecture Notes in Computer Science, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37, DOI:10.1007/978-3-319-46448-0_2.

[25] K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas, "Imbalance Problems in Object Detection: A Review," *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pp. 1–1, 2020, DOI:10.1109/TPAMI.2020.2981890.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788, DOI:10.1109/CVPR.2016.91.

[27] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, K. Michael, Lorna, A. V, D. Montes, J. Nadar, Laughing, tkianai, yxNONG, P. Skalski, Z. Wang, A. Hogan, C. Fati, L. Mammana, AlexWang1900, D. Patel, D. Yiwei, F. You, J. Hajek, L. Diaconu, and M. T. Minh, "ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference," Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6222936

[28] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv:2004.10934 [cs, eess]*, Apr. 2020.

[29] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. 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*. Springer, 2020, pp. 213–229, DOI:10.48550/arXiv.2005.12872.

[30] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum, "Dino: Detr with improved denoising anchor boxes for end-to-end object detection," *arXiv preprint arXiv:2203.03605*, 2022, DOI:10.48550/arXiv.2203.03605.

[31] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

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

[33] S. Markidis, S. Chien, E. Laure, I. Peng, and J. S. Vetter, "Nvidia tensor core programmability, performance & precision," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2018, pp. 522–531. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/IPDPSW.2018.00091

[34] M. Ahmad, M. Abdullah, and D. Han, "Small object detection in aerial imagery using retinanet with anchor optimization," in *2020 International Conference on Electronics, Information, and Communication (ICEIC)*, 2020, pp. 1–3, DOI:10.1109/ICEIC49074.2020.9051269.

[35] M. Wolf, K. van den Berg, S. P. Garaba, N. Gnann, K. Sattler, F. Stahl, and O. Zielinski, "Machine learning for aquatic plastic litter detection, classification and quantification (APLASTIC-Q)," *Environmental Research Letters*, vol. 15, no. 11, p. 114042, Nov. 2020, DOI:10.1088/1748-9326/abbd01.