

Herkansing BB8



Onderzoek Onkruid detectie

Kenan de Vries

Studentnummer: 1811470

9-8-2023

<u>1. Beredenering dataset.....</u>	3
<u>2. Data preprocessing.....</u>	4
<u>3. Methodische aanpak.....</u>	5
<u>4. Yolo model.....</u>	5
<u>5. RoboFlow.....</u>	9
<u>6. Conclusie en Advies.....</u>	10
<u>7. literatuurlijst:.....</u>	11

1. Beredenering dataset

Voor de opdracht is gekozen om een alternatieve dataset te gebruiken in plaats van de Asset Insight data. Een van de vereiste tijdens deze opdracht is dat de code in Github wordt ingeleverd, terwijl Asset Insight ons explicet heeft gevraagd geen data te uploaden of te mailen. Gezien in de implementatie ook data wordt gebruikt en weergegeven, zou dit het gebruik van online platforms als Github en Google Collab niet mogelijk maken. Tegelijkertijd opent dit ook de weg naar het verkennen van meer opties en tools om een model te maken, zoals RoboFlow gebruiken om een model te implementeren.

Voor het individuele leerproces levert het ook voordelen op. Er is al onderzoek gedaan over de dataset van Asset Insight en met meerdere modellen en parameters geëxperimenteerd. Hierdoor kan al van tevoren worden geconcludeerd dat Yolo V7 het beste werkt voor de dataset, het model een accuraatheid van een bepaalde MAP moet opleveren en wat de alternatieve mogelijkheden zijn. Door zelfstandig vanaf het begin onderzoek te doen met nog onbekende resultaten is er geen bias en wordt de kern van de opdracht omarmd. De voorkeur voor de alternatieve dataset ging naar die van weed25 omdat deze vanuit school werd aangeraden, maar dit was bijna onmogelijk te downloaden. Alle internetbronnen kwamen op dezelfde downloadpagina uit en deze vereist het downloaden van een (dubieuze) speciale cliënt en een chinees telefoonnummer. Uiteindelijk is de keuze gevallen op een gecombineerde dataset van RoboFlow Universe[\[1\]](#). De data bevat 1299 afbeeldingen van vegetatie en is vooraf gelabeld.



Voorbeelden uit de dataset.

2. Data preprocessing

De grootte van de dataset die momenteel wordt gebruikt (1100) is voor het doeleinde van het project niet optimaal [2]. Hierdoor is de keuze gemaakt om een tweede dataset te zoeken op RoboFlow universe en deze te combineren met de eerste set, waardoor we uitkomen op een totale dataset van 1299 [1]. De nieuwe set zorgt voor wat extra variatie maar bevat ook een aantal afbeeldingen van mindere kwaliteit.



Dataset 1 (links) bevat afbeeldingen van betere resolutie, belichting en focus ten opzichte van dataset 2 (rechts)

De data van mindere kwaliteit die de tweede dataset bevat kan echter ook in het voordeel werken: *“It is common knowledge that the more data an ML algorithm has access to, the more effective it can be. Even when the data is of lower quality, algorithms can actually perform better, as long as useful data can be extracted by the model from the original data set.”* (Perez & Wang, 2017, p. 1) [3].

Dit is ook de reden dat data-augmentation een enorm effectieve techniek is. Door het gebruik van data-augmentatie heeft het model meer data en ook data met meer variatie, wat leidt tot een beter presterend model [3,4,5,6] en overfitting tegengaat [3,5]. Er is uiteraard dan ook besloten om data augmentation te gebruiken om de resultaten van de modellen te verbeteren. De data-augmentation technieken van beide modellen bevatten blurring, greyscale, flippen en de hoeken van de bounding boxes veranderen.



Data waar data-augmentation op is toegepast.

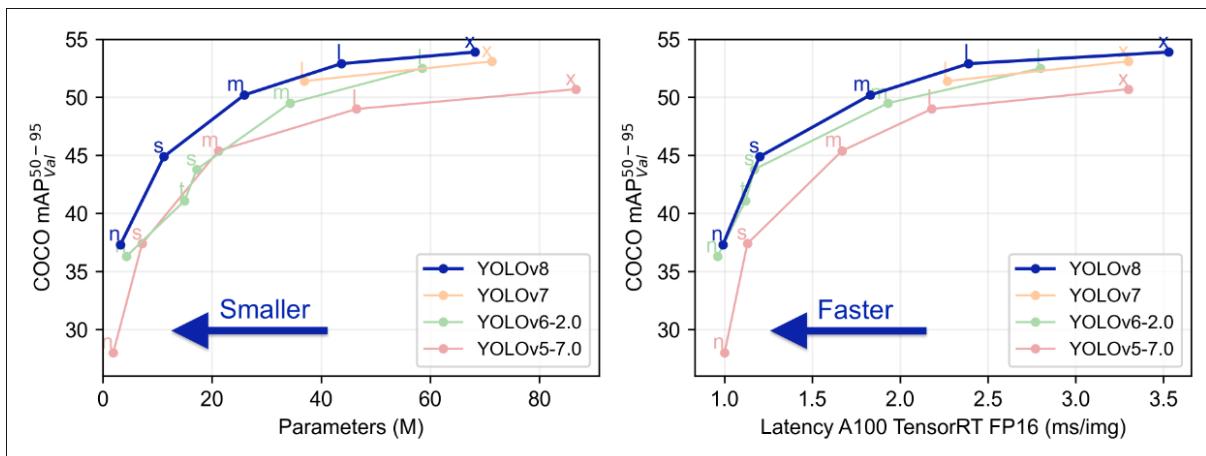
3. Methodische aanpak

Dit project wordt zelfstandig uitgevoerd waardoor de beschikbare middelen beperkt zijn. De hardware waar dit project op uitgevoerd moet worden bevat enkel een Low-medium end laptop wat mogelijkheden beperkt. Tijdens het BB8 project werd dit ook al ondervonden, maar intensieve object detection modellen als Faster R-CNN vallen hierom bij voorbaat al af: *'These models have many layers and parameters, which require high-end hardware and resources to train and run.'* (Deep Learning, 2023) [7]. Voor het onderzoek is er daarom gericht gezocht naar object detection modellen die niet intensief zijn voor de hardware, bekend als een lightweight model. Ook is er opgelet dat de gevonden bronnen overeenkomen met de dataset waar dit project mee gewerkt is.

Yolo vertoont veelbelovende resultaten bij vergelijkbare datasets [8,9] en kan zelfs op hardware als een Raspberry Pi 4 draaien [6]. Door de goede resultaten in vergelijkbare onderzoeken en de ruime hoeveelheid documentatie is er besloten met een yolo model te experimenteren. Yolo is een SSD (Single Shot Detector) algoritme, wat betekent dat het de voorkeur geeft aan snelheid en niet aan accuraatheid zoals bij two shot detection algorithms. Hierdoor ontstond interesse om een two shot detection model te onderzoeken om te kijken of dit daadwerkelijk accurater is. Van de gevonden modellen die hier gebruik van maken vertoont R-CNN en varianten hierop zoals Fast R-CNN het beste op onkruid datasets [10,11]. Idealiter zouden deze modellen zelf getraind en getest worden, maar door de hardware limitaties is besloten om dit via RoboFlow te doen. RoboFlow maakt het door middel van RoboFlow train mogelijk om een aantal populaire machine learning modellen als Faster R-CNN te trainen via een extern online platform.

4. Yolo model

Er zijn enorm veel verschillende yolo modellen, die ieder weer variatie hebben in de versies van de modellen. De onderstaande afbeeldingen van de Ultralytics github [12] laat de prestaties van de verschillende modellen zien op de welbekende COCO dataset.



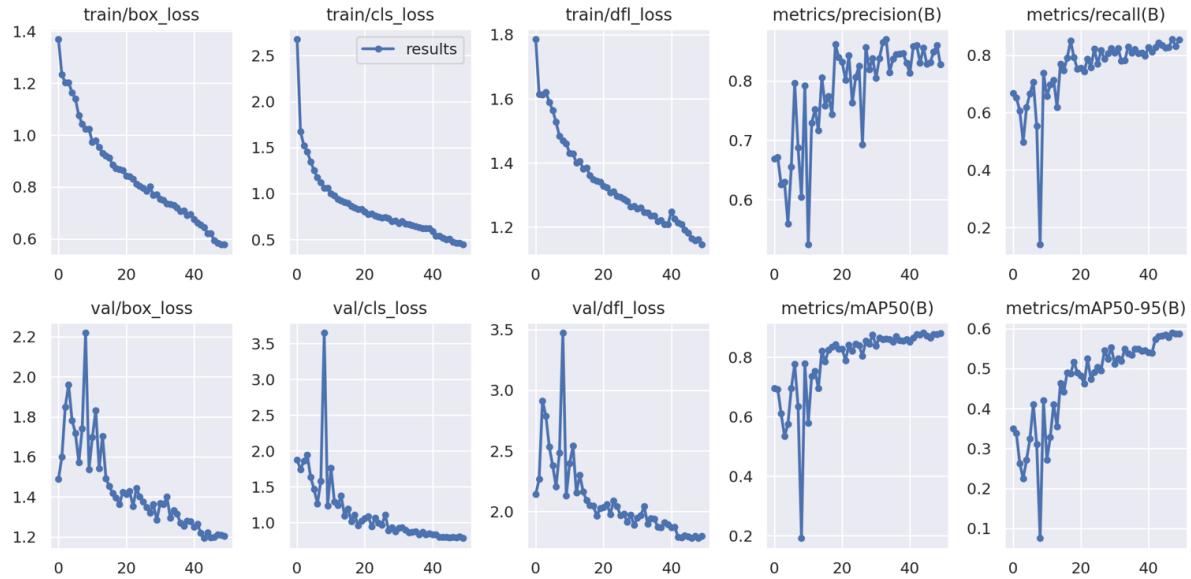
Vergelijking Yolo modellen op de COCO dataset [12]

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Vergelijking verschillende Yolo V8 modellen op de coco dataset [12]

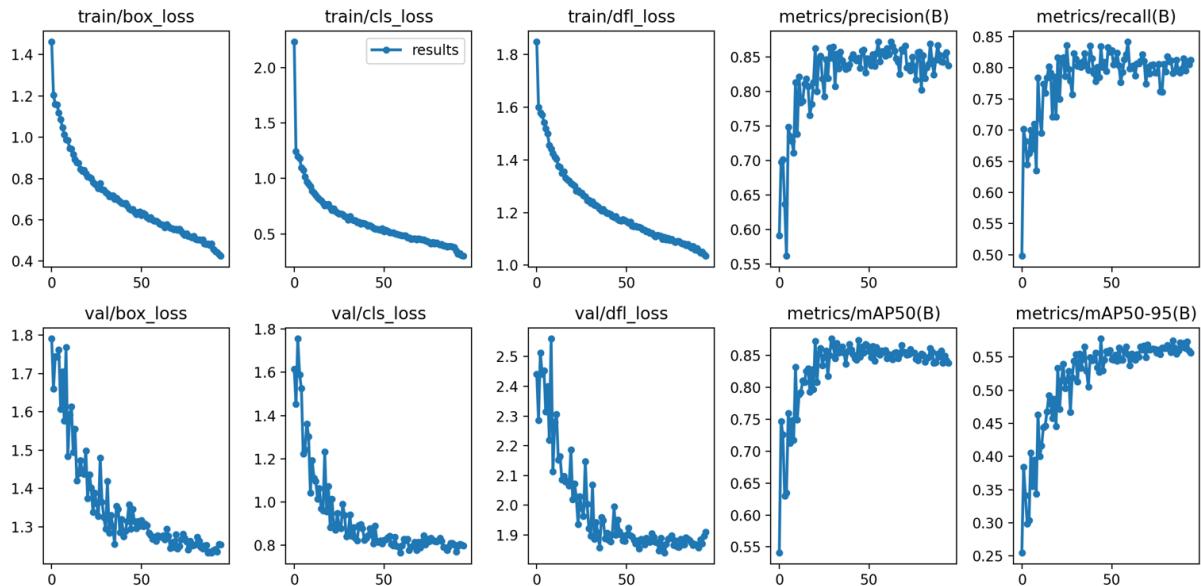
Aan de hand van deze resultaten lijkt Yolo V8, in het specifiek Yolo V8x, het best te presteren onder de modellen op het gebied van accuraatheid. De detectie van yolo modellen bij onkruid gerelateerde problemen laat ook zien dat een heavyweight versie van Yolo V8 het beste gaat presteren ten opzichte van de voorgangers [\[8\]](#).

Voor de eerste implementatie van het Yolo V8 model werd gekozen voor de nano versie. Deze versie trained aanzienlijk sneller (3x) dan het Yolo V8 large model, en dient als goede performance baseline. Er werd een poging gedaan om het model te trainen met de instellingen en parameters voor het maken van een performance baseline als aangeraden door Ultralytics [\[2\]](#). We trainen dus met een split van 70-20-10, een image size van 640, learning rate 0.001 en er wordt data augmentation toegepast. Echter crashte het model al bij het bereiken van de 62 epochs omdat CUDA een gebrek aan GPU memory had. Bij het testen van een Yolo V8 medium model crashte het model al bij de 14 epochs en duurde het trainen meerdere uren. Het aanpassen van de batch-size verhielp het probleem ook niet. Omdat de resultaten van een standaard Yolo V8 Nano model bij 50 epochs al veelbelovend waren (figuur 1) was er besloten om een Google Colab lidmaatschap aan te schaffen. Dit maakt het mogelijk om de modellen te trainen op het online platform van Google en een betere GPU te benutten. Het nadeel hiervan is dat er maar een beperkte hoeveelheid getraind kan worden, omdat bij Google Colab betaald wordt aan de hand van resources die worden gebruikt. Er kan dus niet oneindig getest en getraind worden.



(1) Resultaten Yolo V8 Nano bij 50 epochs.

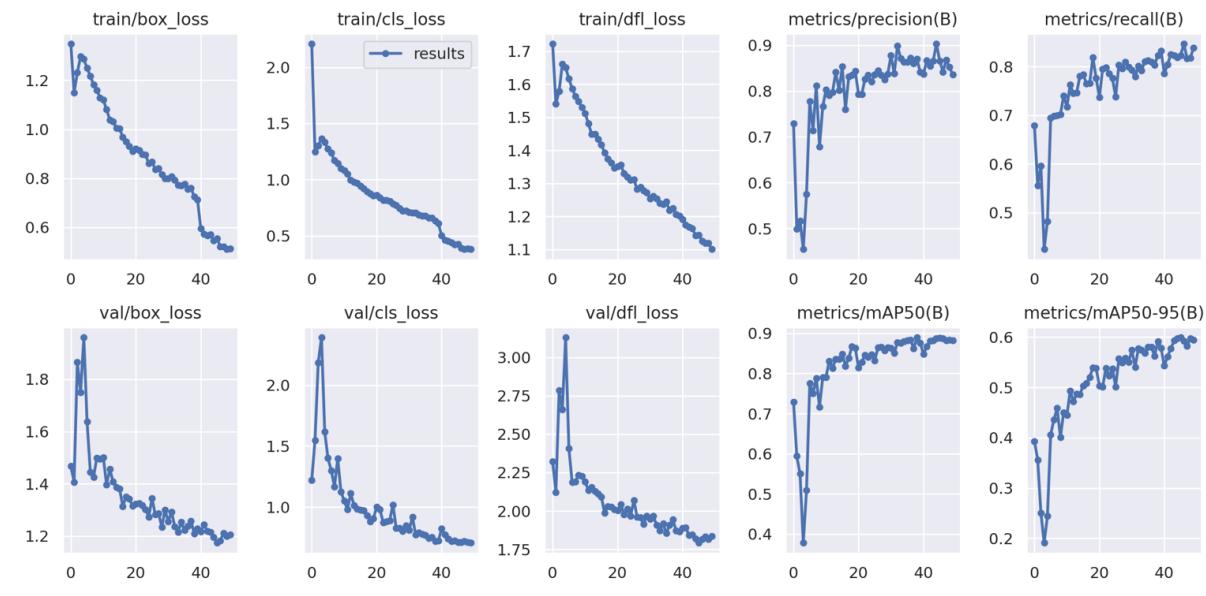
Hetzelfde model bleek bij 100 epochs niet veel beter te presteren (2).



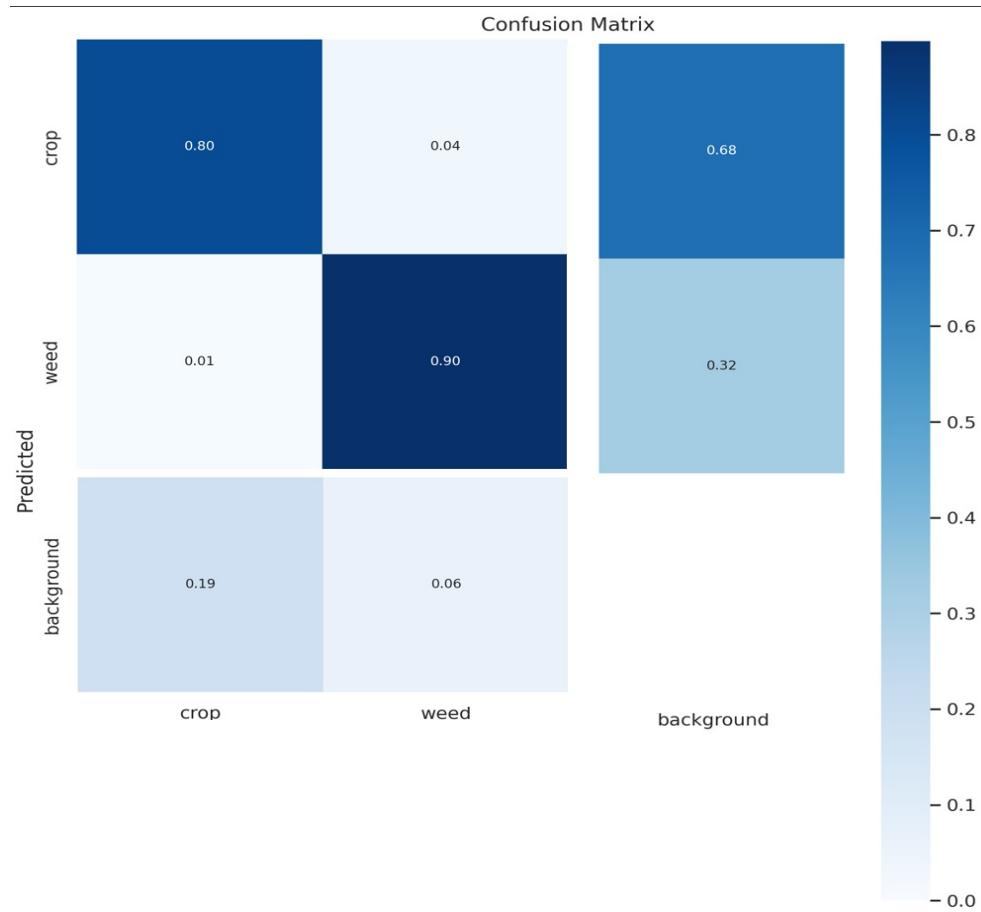
(2) Resultaten Yolo V8 Nano bij 100 epochs.

Wat opvalt bij het Nano model van 100 epochs is dat de accuracy nauwelijks omhoog gaat na de 40 epochs. We kunnen uit figuur 2 wel afleiden dat bij het model nog geen sprake is van overfitting [13]. Op dit punt is er door 42 van de 100 Google Collab resources gegaan.

Omdat het trainen van het Yolo V8 X model veel zwaarder is om te trainen en een minimale verbetering geeft [12] is er besloten om een Yolo V8 Large model te trainen voor 50 epochs (figuur 3, 4 & 5).



(3) Resultaten Yolo V8 Large 50 epochs.



(4) Confusion Matrix Yolo V8 Large 50 epochs.

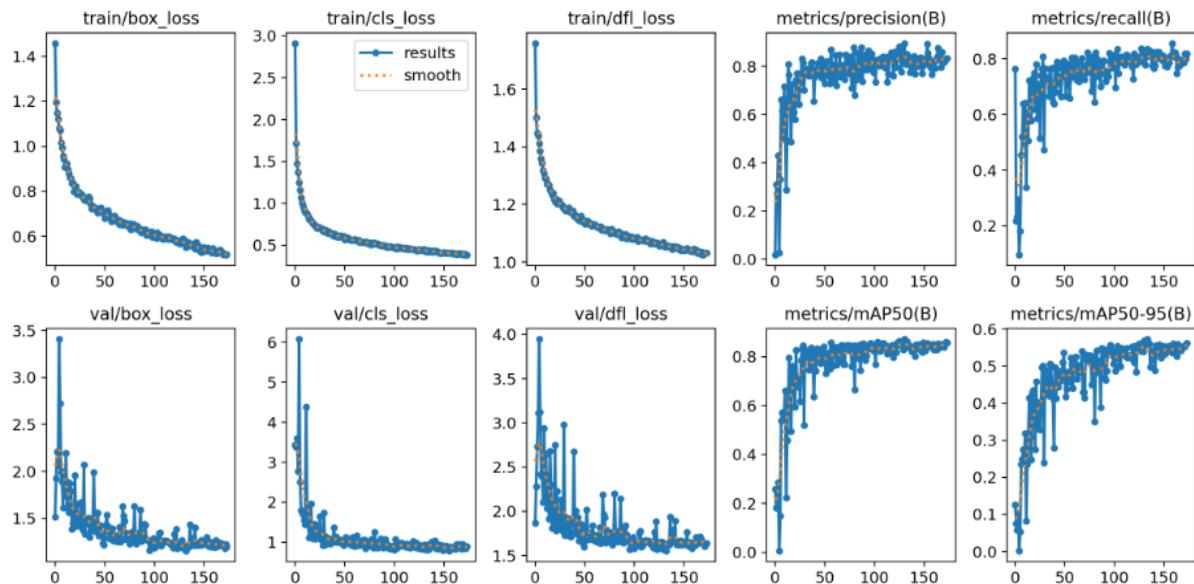
De mAP50, die voor het detectie probleem belangrijk is, komt hier uit op afgerond 89%.



(5) Validatie Yolo V8 Large 50 epochs.

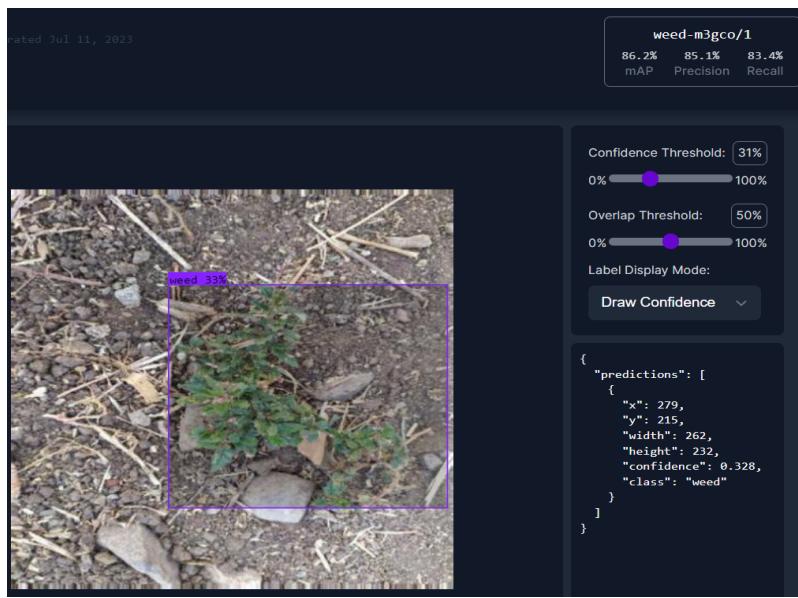
5. RoboFlow

Voor het Roboflow model is gekozen voor een image size van 640, zijn alle mogelijke data augmentation stappen toegepast en is er gekozen voor het Faster R-CNN model. Dezelfde split als bij Yolo is aangehouden. Het model liep voor 200 epochs. Er wordt aangeraden om eerst te kijken hoe het model presteert, dus is er nog geen gebruikgemaakt van pre-trained weights/transfer learning [14]. Omdat het model goed scoorde MAP50 (84%) is er hierna wel transfer learning toegepast met weights van een RoboFlow demo project[15]. De MAP50 ging omhoog naar 86,2% (figuur 6), wat onder de verwachtingen viel.



(6) AutoML Faster R-CNN resultaten van 200 epochs

Wat opvalt bij het RoboFlow model is dat het een lage confidence heeft bij plantachtig onkruid (7).



(7) RoboFlow Confidence op plantachtig onkruid.

6. Conclusie en Advies

Er is onderzoek gedaan naar onkruid detectie met Yolo en Faster R-CNN in combinatie met RoboFlow AutoML. Yolo V8L scoorde het hoogst met een MAP50 van 89% terwijl AutoML Faster R-CNN een MAP50 van 86,2% gaf. Indien accuraatheid de prioriteit is wordt er aangeraden een Yolo V8x model te trainen gezien dit het beter doet dan het V8L model [12]. Doordat het model ook nog niet is begonnen met overfitten wordt ook aangeraden het model voor meer epochs te trainen. Voor een model dat makkelijk inzetbaar is en gebruiksvriendelijk is het advies om RoboFlow te gebruiken. De gebruikte dataset bevat redelijk veel data die niet helemaal correct gelabeld worden waardoor de accuraatheid lager is. Bij het testen op foto's van onkruid op het internet presteren beide modellen foutloos, de verwachting is dan ook dat beide modellen beter werken op een betere dataset.

7. literatuurlijst:

- [1] *Weed Object Detection Dataset (v1, 2023-11-07 1:20pm) by Roboflow.* (n.d.). Roboflow. [3](#), [4](#)
<https://universe.roboflow.com/bb8-6hmqa/weed-m3gco/dataset/1>
- [2] Glenn-jocher, & Sergiuwaxmann. (2022, May 25). *Tips for best training results.* Ultralytics YOLOv8 Docs. [4](#), [6](#)
https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/
- [3] Perez, L.Wang,J. (2017, December 13). *The Effectiveness of Data Augmentation in Image Classification using Deep Learning.* arXiv.org. [4](#)
<https://arxiv.org/abs/1712.04621>
- [4] *AutoAugment: Learning augmentation Strategies from data.* (2019, June 1). IEEE Conference Publication | IEEE Xplore. [4](#)
<https://ieeexplore.ieee.org/document/8953317>
- [5] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1). [4](#)
<https://doi.org/10.1186/s40537-019-0197-0>
- [6] Jabir, B., Falih, N., & Rahmani, K. I. (2021). Accuracy and efficiency comparison of object detection Open-Source models. *International Journal of Online and Biomedical Engineering*, 17(05), 165. [4](#), [5](#)
<https://doi.org/10.3991/ijoe.v17i05.21833>
- [7] Deep Learning. (2023). What are the main challenges and limitations of applying Faster R-CNN and Mask R-CNN to real-world

scenarios? www.linkedin.com. [5](#)

<https://www.linkedin.com/advice/1/what-main-challenges-limitations-applying-faster>

[8] Sportelli, M., Apolo-Apolo, O., Fontanelli, M., Frasconi, C., Raffaelli, M., Peruzzi, A., & Ruiz, M. P. (2023). Evaluation of YOLO object detectors for weed detection in different turfgrass scenarios. *Applied Sciences*, 13(14), 8502. [5](#), [6](#)

<https://doi.org/10.3390/app13148502>

[9] Krishnan, G., Hari, & Rajasenbagam, T. (2021). *A Comprehensive Survey for Weed Classification and Detection in Agriculture Lands*. Government college of technology, Coimbatore, India. [5](#)

[10] Saleem, M. H., Velayudhan, K. K., Potgieter, J., & Arif, K. M. (2022). Weed Identification by Single-Stage and Two-Stage Neural Networks: A study on the impact of image resizers and weights Optimization Algorithms. *Frontiers in Plant Science*, 13. [5](#)

<https://doi.org/10.3389/fpls.2022.850666>

[11] Rahman, A., Lu, Y., & Wang, H. (2023). Performance evaluation of deep learning object detectors for weed detection for cotton. *Smart Agricultural Technology*, 3, 100126. [5](#)

<https://doi.org/10.1016/j.atech.2022.100126>

[12] Ultralytics. (n.d.). *GitHub - ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX > OpenVINO > CoreML > TFLite*. GitHub. [5](#), [7](#), [10](#)
<https://github.com/ultralytics/ultralytics>

[13] Ultralytics. (n.d.-b). *Is that a overfitting problem?* · Issue #1539 · *ultralytics/ultralytics*. GitHub. [7](#)

<https://github.com/ultralytics/ultralytics/issues/1539>

[14] *Train a model in Roboflow - Roboflow Docs.* (n.d.). [9](#)
<https://docs.roboflow.com/train/train>

[15] *Agriculture Object Detection Dataset (v6, 2022-09-30 3:57pm) by Roboflow Demo Projects.* (n.d.). Roboflow. [9](#)
<https://universe.roboflow.com/roboflow-demo-projects/agriculture-328r8/dataset/6>