Contrasting YOLOv5, Transformer, and EfficientDet Detectors for Crop Circle Detection in Desert

Mohamed Lamine Mekhalfi, *Member, IEEE*, Carlo Nicolò, Yakoub Bazi, *Senior Member, IEEE*, Mohamad Mahmoud Al Rahhal, *Member, IEEE*, Norah A. Alsharif, and Eslam Al Maghayreh, *Member, IEEE*

***Abstract*— Ongoing discoveries of water reserves have fostered an increasing adoption of crop circles in the desert in several countries. Automatically quantifying and surveying the layout of crop circles in remote areas can be of great use for stakeholders in managing the expansion of the farming land. This letter compares latest deep learning models for crop circle detection and counting, namely Detection Transformers, EfficientDet and YOLOv5 are evaluated. To this end, we build two datasets, via Google Earth Pro, corresponding to two large crop circle hot spots in Egypt and Saudi Arabia. The images were drawn at an altitude of 20 km above the targets. The models are assessed in within-domain and cross-domain scenarios, and yielded plausible detection potential and inference response.**

***Index Terms*— Aerial imagery, crop circles, detection transformers (DETRs), efficientDet, object detection, precision farming, YOLOv5.**

I. Introduction

PRECISION farming aims at optimizing the output of a farming land with respect to the presented input, thereby reducing cost, time, and human labor without compromising the quality of crops. The literature suggests many contributions. For instance, in [1], an approach for Durum wheat yield assessment via satellite data and normalized difference vegetation index (NDVI) is presented. In [2], a vision system that consists of an optical sensor mounted on tractor, combined with a recognition pipeline, is proposed for yield estimation in kiwifruit orchards. Recurrent neural networks, namely long short-term memory (LSTM) and Bidirectional LSTM (Bi-LSTM), were applied for the assessment of evapotranspiration with the aim of managing water resources. In this work, maximum air temperature and relative humidity were adopted as inputs to the LSTM and Bi-LSTM, and plausible estimation scores were obtained [3]. Disease detection in rice crops was investigated in [4] with convolutional neural networks (CNNs), which has yielded 95.48% on a dataset totaling 500 images of healthy and unhealthy rice crops. CNN was also applied in [5] for Cassava disease detection. Yield estimation in vineyards was addressed in [6], where a prototype that incorporates optical sensors and illumination is mounted on a vehicle that drives along the vines is developed. The work presented in [7] combines CNNs with Watershed algorithm and circular Hough transform for pixel-wise segmentation of apple fruits for further yield estimation. A comprehensive review of recent advances in precision agriculture can be found in [8].

In this context, center pivot is a farming practice that has proven efficient in optimizing water use, where rotating sprinklers pump water around the center of the crop circle. Fossil water has enabled several arid countries to consider crop circles around aquifers in the desert. Thus, locating, monitoring, and planning the distribution of circular crop fields in an automatic fashion constitute an essential first step. However, so far it has not been investigated enough in the relevant literature, which may be due to the lack of accessible datasets. On the other hand, the rapid evolution of deep frameworks for object detection has brought forth several efficient and highly applicable models [9]. However, the remote sensing art remains short of contrastive assessment of recent deep architectures for pinpointing objects in remote sensing images.

Aiming to address these two points, we contrast state-of-the-art object detectors for crop circle detection in remote sensing images, namely the latest version of the You Only Look Once (YOLO) model (i.e., YOLOv5), whose previous editions (from YOLOv1 up to YOLOv4) were a major discussion in previous literature [10]–[14], Detection Transformer (DETR) that was devised recently by Facebook AI [15], and EfficientDet which was put forth last year by Google [16]. To this end, we build two datasets, acquired via Google Earth platform, over two agricultural spots in Egypt and Saudi Arabia (the datasets will be made available upon request). Furthermore, we study the tendency of each of the three models to generalize in a cross-domain case.

Manuscript received March 11, 2021; revised April 18, 2021; accepted May 25, 2021. Date of publication June 14, 2021; date of current version December 17, 2021. This work was supported by the Deanship of Scientific Research at King Saud University under Grant RG-1441-502. *(Corresponding author: Mohamad Mahmoud Al Rahhal.)*

Mohamed Lamine Mekhalfi and Carlo Nicolò are with the Department of Information Engineering and Computer Science, University of Trento, 38123 Trento, Italy (e-mail: mohamed.mekhalfi@alumni.unitn.it; carlo.nicolo@alumni.unitn.it).

Yakoub Bazi and Norah A. Alsharif are with the Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia (e-mail: ybazi@ksu.edu.sa; 441202939@student.ksu.edu.sa).

Mohamad Mahmoud Al Rahhal is with the Department of Applied Computer Science, College of Applied Computer Science, King Saud University, Riyadh 11543, Saudi Arabia (e-mail: mmalrahhal@ksu.edu.sa).

Eslam Al Maghayreh is with the Department of Applied Computer Science, College of Applied Computer Science, King Saud University, Riyadh 11543, Saudi Arabia, and also with the Department of Computer Science, Yarmouk University, Irbid 21163, Jordan (e-mail: eslam@yu.edu.jo).

Digital Object Identifier 10.1109/LGRS.2021.3085139

The novelties of this letter can be summed up as follows.

1) We address the problem of crop circle detection in desert, and we build two evaluation datasets.
2) We contrast three recent object detection models, namely YOLOv5, Transformer, and EfficientDet, which we believe is essential for object detection in remote sensing imagery.

II. Methodology

In the following subsections, we briefly describe each model.

A. Detection Transformers

DETR regards object detection as a direct set prediction problem. It operates upon two passes, namely: 1) set prediction loss and 2) architecture that predicts objects in parallel and models their relation [15]. As per the former, the number of potential object predictions is fixed; let us denote it as N, which is set to be significantly larger than the number of objects that can be encountered in a typical image. The set of objects is inferred all-together via a decoder.

The loss consists in a bipartite matching between the predictions and the ground truth, followed by a bounding box loss optimization for each individual object. Let y be the set of ground-truth objects and $\hat{y} = \{\hat{y}_i\}_{i=1}^{N}$ the prediction set. To seek a matching between the predictions and the ground truth, we search for a permutation of N elements $\sigma \epsilon \partial_N$ that produce the lowest cost

$$\hat{\sigma} = \underset{\sigma \epsilon \partial_N}{\operatorname{argmin}} \sum_{i}^{N} L_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) \quad (1)$$

where $L_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ represents the cost of the prediction $\hat{y}$ of index $\sigma(i)$ with respect to the ground truth y_i, which can be calculated with the Hungarian algorithm [17].

Both the class prediction and the similarity of bounding boxes corresponding to the predicted and ground-truth objects are taken into account. Therefore, a generic element from the ground truth can be expressed as $y_i = (c_i, b_i)$ where c_i is the object class label and $b_i \epsilon [0, 1]$ is a vector containing the coordinates of the ground-truth box center, its height, and width with respect to the image size. Similarly, for the prediction $\hat{y}$ of index $\sigma(i)$, a probability of class c_i and a predicted box are defined as $\hat{p}_{\sigma(i)}(c_i)$ and $\hat{b}_{\sigma(i)}$, respectively. Thus, the loss can be expressed as

$$L_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) = -1_{\{c_i \neq \emptyset\}} \hat{p}_{\sigma(i)}(c_i) + 1_{\{c_i \neq \emptyset\}} L_{\text{box}}(b_i, \hat{b}_{\sigma(i)}) \quad (2)$$

where L_{box} is the cost of the predicted bounding box with respect to its ground-truth counterpart. It is given as

$$L_{\text{box}}(b_i, \hat{b}_{\sigma(i)}) = \lambda_1 L_{\text{io}}(b_i, \hat{b}_{\sigma(i)}) + \lambda_2 \left\| b_i - \hat{b}_{\sigma(i)} \right\|_1 \quad (3)$$

where $\lambda_1, \lambda_2 \epsilon \mathbb{R}$ are hyperparameters and $L_{\text{io}}(.)$ is the generalized intersection over union loss given in [18].

Regarding the loss function, a linear combination of a negative log-likelihood of class prediction and box loss is adopted, namely a Hungarian loss as follows:

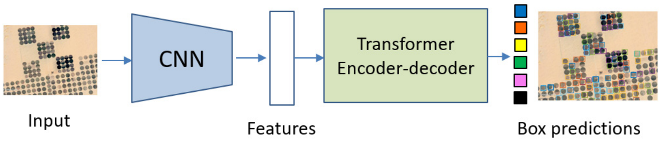

Fig. 1. DETR architecture [15].

$$L_{\text{Hungarian}}(y_i, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + 1_{\{c_i \neq \emptyset\}} L_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \right] \quad (4)$$

where $\hat{\sigma}$ is the optimal assignment drawn from (1).

DETR architecture (Fig. 1) consists of three blocks: 1) feature extraction backbone (ResNet-50); 2) encoder–decoder transformer which is the essence of DETR; and 3) feed forward network (FFN) that draws the final detection.

Given an RGB image of dimension $3 \times H_0 \times W_0$, the backbone produces a feature map of $H \times W$. At the transformer encoder stage, the process proceeds by a 1×1 convolution to reduce the channel dimension to d, creating a feature map $z_0 \in \mathbb{R}^{d \times H \times W}$. Since the encoder receives a sequence at the input, z_0 is collapsed into a feature map of dimension $d \times HW$.

The encoder may consist of several layers, where each one has a multihead self-attention module and an FFN. To mitigate the permutation invariance of the transformer module, positional encodings are presented alongside the CNN backbone output. The encoder outputs N vector embeddings of size d, also termed object queries, corresponding to the typical number of objects present in a given image. The learned embeddings are then decoded (via the decoder module, using multiheaded self- and encoder–decoder attention mechanisms) and passed to a shared FFN in order to infer the detection (i.e., normalized center coordinates, bounding box height and width, and the class label) through a softmax function or a "no object" class. The final prediction is obtained by a three-layer perceptron with ReLU activation and a linear projection layer.

B. EfficientDet

EfficientDet was posed by Google Brain Team; it capitalizes mainly on two blocks, namely a bidirectional feature pyramid network (BiFPN) and the EfficientDet detector [16].

Representing multiscale features has been a long-standing bottleneck in computer vision, FPN is one of the widely adopted mechanisms in this regard. The fact that information flow is unidirectional has instigated follow-up research [19], [20]. However, the features in such approaches are typically resized to the same resolution and summed up, which renders their contributions to the fused output equal, which is inadequate as they exhibit different resolutions. Moreover, such techniques tend to be prohibitively expensive to compute.

BiFPN was introduced in [16], where personalized weights are learned to penalize the features according to their importance while applying bottom-up and top-down multiscale feature fusion. BiFPN is inspired by three intuitions: 1) if a node has only one input edge with no feature fusion, it will likely have a smaller contribution to the feature network; 2) extra

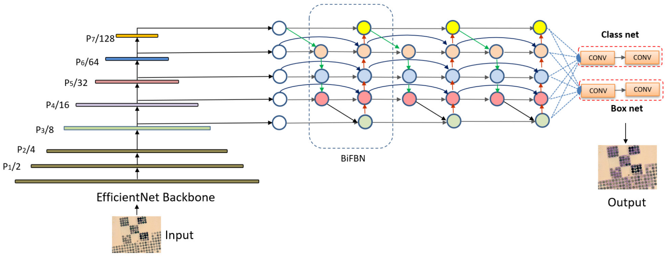

Fig. 2. EfficientDet architecture [16].

edge is adopted to tie the input node to the output node if they are at the same level, which allows fusing more features at a minimum cost; and 3) each bidirectional (top-down and bottom-up) path is regarded as one feature layer, which is repeated several times to enable high-level feature fusion.

The weights that are allocated to the features can be computed in three approaches. The first one is named "unbounded fusion," which is expressed as

$$O = \sum_i w_i \cdot I_i \tag{5}$$

where w_i is a learnable weight which can be a scalar, a vector, or multidimensional tensor per feature, channel, or pixel, respectively. A downside of this technique is that the scalar weight is unbounded, which may cause inconvenience in the training.

The second one consists in a "Softmax fusion" as follows:

$$O = \sum_i \frac{e^{w_i}}{\sum_j e^{w_j}} \cdot I_i \tag{6}$$

while this strategy bounds the weight within [0, 1] and guarantees feature contribution preservation, and its latency cost is relatively higher.

The third one is a "fast normalized fusion" given by

$$O = \sum_i \frac{w_i}{\in + \sum_j w_j} \cdot I_i \tag{7}$$

where $w_i \geq 0$ is confirmed with a ReLU function for each w_i, and $\in =$ 0.0001 for numerical stability. This technique is favored as it ensures a bounded weight and a faster response.

As per the architecture of EfficientDet (Fig. 2), it follows a one-stage detection scheme. EfficientNet is adopted as backbone. Next, BiFPN is appended as a feature network, acquiring features from the third to the seventh level of the backbone. The fused output is fed to a class and bounding box prediction network.

C. YOLOv5

YOLO belongs to one stage detectors family and is one of the most famous and cutting-edge object detectors so far. Redmond *et al.* [10] presented the first version (YOLO) in 2016, which was perceived as a breakthrough in real-time object detection/tracking.

Preceding object detectors such as R-CNN and Fast R-CNN rely on region proposals in order to suggest potential object bounding boxes in a given image, followed by a bounding box classification to categorize the objects within, and concluded with bounding box refinement and duplicate removal. Although such frameworks achieved great success, their architecture is rather complex. For instance, each of the previous components needs to be trained separately, while the inference remains slow. To address these bottlenecks, YOLO tackles the problem of object detection from a regression perspective, where the previous components are jointly carried out in a single network. This has cut down sharply on inference overheads as well as false alarms. Moreover, YOLO demonstrated high generalization capability.

YOLO combines a total of 24 convolutional layers, topped by two fully connected layers. The architecture of YOLO is given in [10].

However, YOLO encountered generalization issues when the objects in the image are too small or if the image has different dimensions.

The second version of YOLO has incurred several improvements to remedy the downsides of YOLOv1 [11]. For instance, increased image scale is supported, batch normalization on all convolutional layers was applied, and all fully connected layers are replaced with anchor boxes for bounding box prediction. Moreover, multiscale training was embedded.

Overall, YOLOv2 turned out to be faster and more accurate with respect to YOLOv1. It consists of a 19-layer network (Darknet-19), supplemented with 11 layers for object detection. Therefore, as the layers down-sample the input, fine-grained cues are lost, which hinders the performance on small objects. To mitigate the earlier issue, the third version, YOLOv3, trades accuracy with speed in order to raise the detection rate. In particular, a variant of Darknet-53 which consists of 106 layers is allocated. Moreover, several mechanisms such as skip connections, residual blocks, and upsampling were incorporated. Indeed, the upsampling property preserved the fine-grained features and the performance with small objects is now much higher. However, it comes at an augmented processing cost [12].

YOLOv4 mainly relies on CSPDarknet53 as a backbone. Furthermore, a spatial pyramid pooling is embedded in CSPDarknet53 to improve the receptive field and distinguish contextual features. Another outstanding feature of YOLOv4 is the possibility to execute training on a single GPU [13].

Recently, the last version of YOLO was put forth by ultralytics [14]. Although its denomination stirred some controversy, it is commonly referred to as YOLOv5. CSPNet is adopted as backbone on account of its processing time. In order to deal with object scale changes, PANet is used as a model neck to construct feature pyramids. Similar to previous versions, anchor boxes are applied to infer class probabilities, bounding boxes, and objectiveness scores. In YOLOv5, the number of parameters was dropped. Four different versions are envisioned to train the YOLOv5 network, namely YOLOv5l, YOLOv5m, YOLOv5x, and YOLOv5s. In this letter, we opt for the lightest model, i.e., YOLOv5s. In sum, a key characteristic of YOLOv5 is that it approaches object detection from a regression perspective, while DETR envisions an encoder–decoder architecture, and EfficientDet capitalizes on a feature pyramid network.

III. Experimental Results and Discussion

The experiments were conducted on two crop circle datasets that were built based on images from Google Earth Pro

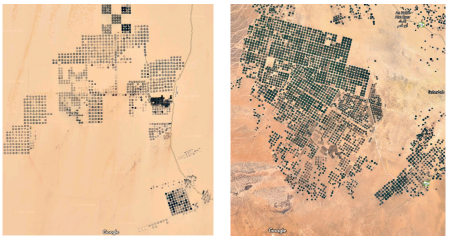

Fig. 3. Overview of the study sites. Left: East Oweinat, Egypt. Right: Wadi As-Sirhan Basin, Saudi Arabia.

TABLE I
DETECTION RESULTS ON EGY AND KSA DATASETS

Method	EGY		KSA	
	Recall	*Precision*	*Recall*	*Precision*
DETR	0.68	0.77	0.69	0.75
EfficientDet	0.85	0.91	0.82	0.80
YOLOv5	0.98	0.85	0.82	0.71

platform, at an altitude of 20 km above the ground. The first dataset corresponds to East Oweinat, in the South Western Desert of Egypt (22°39′00.7″N 28°48′47.7″E), and the second dataset refers to Wadi As-Sirhan Basin in Saudi Arabia (30°18′15.7″N 38°09′51.2″E). An overview of both sites is given in Fig. 3.

The first dataset, named EGY for convenience, totals 690 crop circles for training and 1787 crop circles for testing purposes. The second dataset, termed KSA, amounts to 848 crop circles for training and 1748 for test. Data augmentation (i.e., vertical flips, 45° and 90° rotations) was applied to the training data. Thus, the final training size is 2070 for EGY and 2544 for KSA. Therefore, the training/test split is roughly 54%/46% for EGY and 59%/41% for KSA.

Image size is 1024 × 768, and the average number of objects is 105 and 97 for EGY and KSA, respectively. The EGY dataset features a uniform structure of the targets and large contrast with respect to the background, while KSA presents complex object structure and low contrast between the targets and the background.

The training was carried out on the Google Colab service with a 16-GB Tesla T4 GPU and a 16-GB RAM; the learning rates were set to 10^{-2} for YOLOv5, 10^{-3} for EfficientDet, and 10^{-4} for DETR; and the remaining parameters were set as mentioned in the original letters regarding all models.

Precision and Recall are adopted as evaluation metrics. Precision quantifies the relevance of detected crop circles, and Recall refers to the ratio of detected crop circles with respect to all existing crop circles, as expressed below

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{8}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{9}$$

We report the detection results on Table I for EGY and KSA. Regarding EGY dataset, it can be noted that YOLOv5 is by far the best when considering the detection Recall, owing to its remarkably low "False Negatives," followed by EfficientDet. In terms of Precision, however, EfficientDet outperforms YOLOv5, which can be explained by the tendency of this latter to detect "Positives" which comes at the cost of high "False Positives" rate yet a low precision. DETR remains the least performing among all.

TABLE II
DETECTION RESULTS IN CROSS-DOMAIN SCENARIO

Method	KSA2EGY		EGY2KSA	
	Recall	*Precision*	*Recall*	*Precision*
DETR	0.68	0.77	0.64	0.68
EfficientDet	0.86	0.86	0.90	0.77
YOLOv5	0.87	0.75	0.85	0.72

With regard to KSA dataset, YOLOv5 and EfficientDet are equivalent in terms of Recall, while EfficientDet stands out in terms of Precision. This may be due to the context of the crop circles in the KSA dataset, where crop circles are surrounded by a rough terrain. Moreover, the crop circles are not consistent in color and texture, and they may be too close to each other, forming a cluster. This also explains the relatively higher scores on the EGY dataset, owing to its (mostly) uniform terrain, and consistency of the crop circles in terms of color and texture, which renders their detection relatively within reach. We can observe that YOLOv5 and EfficientDet are far ahead of DETR in terms of both metrics, which may be due to their capability to learn richer features thanks to the feature pyramid that enables dense representations. Furthermore, DETR incorporates feature dimension reduction operations, supplemented with an encoding–decoding step, which may alter the representation of the learned feature cues. Overall, it can be noted that EfficientDet seems more stable (i.e., offers a good balance between Precision and Recall) and more precise.

In order to assess further the generalization capability of each model, we considered the potential of each model to generalize on a different domain, where each model is trained on the KSA dataset and applied to EGY dataset (KSA2EGY, for short) and vice versa (i.e., EGY2KSA), and the results are summarized in Table II. The results indicate that the scores have dropped overall, and EfficientDet remains more precise than YOLOv5 notwithstanding their comparable Recall scores. It can be noted also that EfficientDet maintains a balance among Precision and Recall. DETR, on the other hand, shows similar results to the case when it is trained on the same domain, which suggests a better cross-domain robustness with respect to the other two models, although it remains lagging behind in terms of both metrics.

When the models are trained on the EGY dataset and applied on the KSA dataset, relatively a tangible Precision decrease for all models is incurred. For instance, EGY2KSA suggests some increase in the Recall for EfficientDet and YOLOv5, while a decrease in Precision is observed for EfficientDet and DETR. The rationale behind this may be that the EGY dataset features more consistent shape and color in the crop circles, which is the most distinguishing cue of crop circles.

In terms of inference time per image (Table III), on average, EfficientDet is the fastest, followed by YOLOv5 and DETR. Across all models, EfficientDet seems to be the best option, owing to its robustness, stability, and inference speed.

TABLE III

AVERAGE INFERENCE TIME PER IMAGE

	DETR	EfficientDet	YOLOv5
Inference (s)	3.4	0.04	0.07

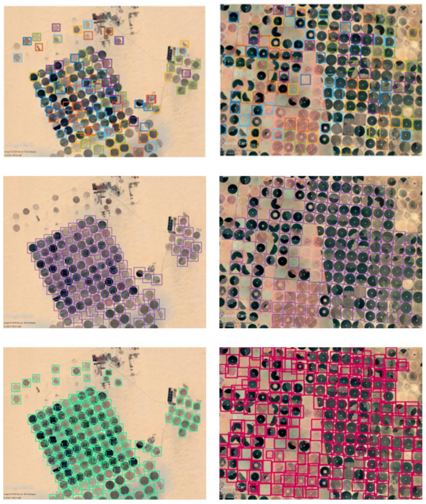

Fig. 4. Detection examples. First column: EGY. Second column: KSA. First row: DETR. Second row: EfficientDet. Last row: YOLOv5.

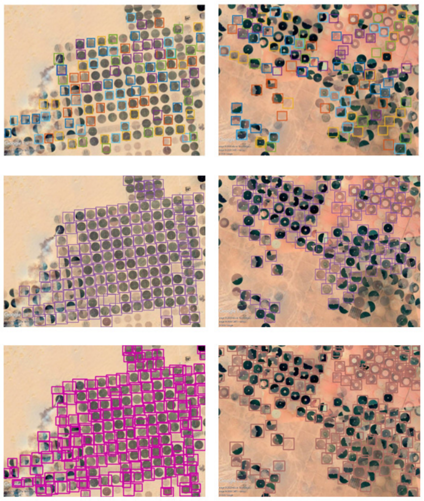

Fig. 5. Cross-domain detection examples. First column: KSA2EGY. Second column: EGY2KSA. First row: DETR. Second row: EfficientDet. Last row: YOLOv5.

Detection examples for each case are displayed in Figs. 4 and 5 for within-domain and cross-domain scenarios, respectively. In particular, YOLOv5 tends to yield more detections with respect to EfficientDet, especially the crop circles that are partially occluded or have a similar color to the background. However, EfficientDet remains more precise. In this respect, we believe that adopting a late fusion based on the detection probability of EfficientDet and YOLOv5 for each crop circle instance would be particularly advantageous in this case. For instance, order-induced metric fusion [21] can be applied to penalize detection priors.

IV. CONCLUSION

This letter provided a comparative study of Transformer, EfficientDet, and YOLOv5 for crop circle detection in remote sensing imagery. Their generalization potential was also assessed when trained and tested on different domains. The models exhibited varying performance in terms of Precision, Recall, stability, and inference time. Overall, DETR remains rather limited; YOLOv5 tends to detect more objects regardless of their precision, while EfficientDet offers a good generalization capability and seems to be more stable.

REFERENCES

[1] P. Toscano, A. Castrignanó, S. F. Di Gennaro, A. V. Vonella, D. Ventrella, and A. Matese, “A precision agriculture approach for durum wheat yield assessment using remote sensing data and yield mapping,” *Agronomy*, vol. 9, no. 8, p. 437, Aug. 2019, doi: 10.3390/agronomy9080437.

[2] M. L. Mekhalfi *et al.*, “Vision system for automatic on-tree kiwifruit counting and yield estimation,” *Sensors*, vol. 20, no. 15, p. 4214, Jul. 2020, doi: 10.3390/s20154214.

[3] H. Afzaal, A. A. Farooque, F. Abbas, B. Acharya, and T. Esau, “Computation of evapotranspiration with artificial intelligence for precision water resource management,” *Appl. Sci.*, vol. 10, no. 5, p. 1621, Feb. 2020, doi: 10.3390/app10051621.

[4] Y. Lu, S. Yi, N. Zeng, Y. Liu, and Y. Zhang, “Identification of rice diseases using deep convolutional neural networks,” *Neurocomputing*, vol. 267, pp. 378–384, Dec. 2017, doi: 10.1016/j.neucom.2017.06.023.

[5] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, and D. P. Hughes, “Deep learning for image-based cassava disease detection,” *Frontiers Plant Sci.*, vol. 8, p. 1852, Oct. 2017, doi: 10.3389/fpls.2017.01852.

[6] S. Nuske, K. Wilshusen, S. Achar, L. Yoder, S. Narasimhan, and S. Singh, “Automated visual yield estimation in vineyards,” *J. Field Robot.*, vol. 31, no. 5, pp. 837–860, Sep. 2014, doi: 10.1002/rob.21541.

[7] S. Bargoti and J. P. Underwood, “Image segmentation for fruit detection and yield estimation in apple orchards,” *J. Field Robot.*, vol. 34, no. 6, pp. 1039–1060, Sep. 2017, doi: 10.1002/rob.21699.

[8] K. Jha, A. Doshi, P. Patel, and M. Shah, “A comprehensive review on automation in agriculture using artificial intelligence,” *Artif. Intell. Agricult.*, vol. 2, pp. 1–12, Jun. 2019, doi: 10.1016/j.aiia.2019.05.004.

[9] X. X. Zhu *et al.*, “Deep learning in remote sensing: A comprehensive review and list of resources,” *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, Dec. 2017, doi: 10.1109/MGRS.2017.2762307.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proc. CVPR*, 2016, pp. 779–788.

[11] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” in *Proc. CVPR*, 2017, pp. 7263–7271.

[12] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” Apr. 2018, *arXiv:1804.02767*. [Online]. Available: https://arxiv.org/abs/1804.02767

[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal speed and accuracy of object detection,” Apr. 2020, *arXiv:2004.10934*. [Online]. Available: http://arxiv.org/abs/2004.10934

[14] ultralytics/yolov5. (2021). *Ultralytics*. [Online]. Available: https://github.com/ultralytics/yolov5

[15] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *Proc. Comput. Vis. ECCV*, Cham, Switzerland, 2020, pp. 213–229, doi: 10.1007/978-3-030-58452-8_13.

[16] M. Tan, R. Pang, and Q. V. Le, “EfficientDet: Scalable and efficient object detection,” in *Proc. CVPR*, 2020, pp. 10781–10790.

[17] R. Stewart, M. Andriluka, and A. Y. Ng, “End-to-end people detection in crowded scenes,” in *Proc. CVPR*, 2016, pp. 2325–2333.

[18] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized intersection over union: A metric and a loss for bounding box regression,” in *Proc. CVPR*, 2019, pp. 658–666.

[19] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” in *Proc. CVPR*, 2018, pp. 8759–8768. Accessed: Mar. 11, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/8579011

[20] G. Ghiasi, T.-Y. Lin, and Q. V. Le, “NAS-FPN: Learning scalable feature pyramid architecture for object detection,” in *Proc. CVPR*, 2019, pp. 7036–7045. Accessed: Mar. 11, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/8954436

[21] B. Mirmahboub, M. L. Mekhalfi, and V. Murino, “Person re-identification by order-induced metric fusion,” *Neurocomputing*, vol. 275, pp. 667–676, Jan. 2018, doi: 10.1016/j.neucom.2017.09.019.