Lightweight and Efficient Attention-based CNN Models for In-field Strawberry Instance Segmentation

Ziang Zhao[1,*], Yulia Hicks[1], Xianfang Sun[2], Benjamin J. McGuinness[3], Hin S. Lim[3]

Abstract— **Accurately segmenting strawberries within real-world production settings not only helps the development of automated harvesting robots but also allows for precise calculation of the number and size of strawberries, providing accurate yield information for agricultural planning and resource optimization. This paper proposes lightweight and efficient CNN models specifically designed for strawberry instance segmentation, consisting of an efficient self-attention-based backbone, a feature pyramid network (FPN), and a decoder with an instance branch and a mask branch. The proposed models surpass the original and simplified Mask R-CNN with significant 21.46 and 22.97 AP gains respectively, with the Base backbone achieving the highest AP of 70.22. Additionally, our models demonstrate efficiency by requiring much fewer parameters (17.42M) and floating-point operations (78.3G) compared to Mask R-CNN (35.08M / 877.4G), making them suitable for deployment on devices with limited computational resources.**

I. INTRODUCTION

The rapid progress of artificial intelligence has made digitization, precision, and intelligent agriculture crucial elements in advancing the modernization of the agricultural sector. In recent years, various methods for harvesting robots have been developed for fruits and crops, such as apple [1], green citrus [2], radiata pine [3] and asparagus spear [4]. Strawberries are a type of high-value fruit, involving labour-intensive cultivation and harvesting processes with a considerable dependency on manual operations. Therefore, the introduction of automated harvesting robots not only helps alleviate the challenges posed by labour shortages but also significantly enhances production efficiency and sustainability. In this development, strawberry instance segmentation plays a pivotal role.

Instance segmentation is a computer vision task that aims to accurately identify and segment individual instances of objects in an image. Unlike semantic segmentation, which identifies object categories, instance segmentation provides unique identifiers for each instance of an object [5]. As instance segmentation models provide independent identification for each strawberry, robots can accurately locate each strawberry instance, execute precise harvesting actions, and avoid damage to plants and ensure the picking of only fruits.

For the past few years, convolutional neural network (CNN) models have been widely used in fruit detection and segmentation. MangoNet was proposed to segment and count mangoes in open-field RGB images, showing promise for precise yield estimation [6]. FoveaMask was created for green fruit segmentation, incorporating a position attention module in the embedding mask branch to gather essential information from pixels [7]. DaSNet-v2 introduces a one-stage detection network that combines both instance and semantic segmentation, which allows the network to conduct fruit instance and branches semantic segmentation simultaneously [1]. Mask R-CNN [8] was applied to showcase strong performance in the detection, segmentation, and tracking of grape clusters [9]. Mask R-CNN was improved by using ResNet and DenseNet as the feature extraction backbone, improve the accuracy of apple recognition in challenging scenarios with overlaps, agglomerations, and occlusions [10]. Concerning strawberry instance segmentation, there are some existing works ([11], [12], [13]), most of which are based on Mask R-CNN. Mask R-CNN is an effective segmentation model, it demonstrates good performance in generating accurate segmentation masks for each object.

However, training and inference with Mask R-CNN demand substantial computational resources, making it challenging to deploy in resource-constrained environments. To solve this problem, we propose lightweight, effective and efficient CNN models. The contributions of this paper are:

- Our models outperform state-of-the-art Mask R-CNN with higher AP and fewer parameters and floating-point operations.
- Efficient Self-Attention is applied in the backbone to extract semantic features.
- The CoordConv module and Instance Activation Maps are incorporated to add position and instance-aware weighted maps to the decoder.

II. DATASET

The StrawDI_Db1 database [12] comprises 3100 images captured in strawberry plantations at various times throughout a complete picking campaign. These images are organized into training, validation, and testing subsets, encompassing 2800, 100, and 200 images, respectively. A statistical overview is presented in Table I. It is worth noting that the StrawDI_Db1 dataset offers instance-level annotations for all sets. Example annotations are illustrated in Fig. 1.

*This work was supported by Cardiff University/University of Waikato Collaboration Seed Fund.

[1]Ziang Zhao and Yulia Hicks are with School of Engineering, Cardiff University, United Kingdom {zhaoz60, hicksya}@cardiff.ac.uk

[2]Xianfang Sun is with School of Computer Science and Informatics, Cardiff University, United Kingdom {sunx2}@cardiff.ac.uk

[3]Benjamin J. McGuinness and Hin S. Lim are with School of Engineering, University of Waikato, New Zealand {ben.mcguinness, hin.lim}@waikato.ac.nz

TABLE I
STATISTICS OF THE STRAWDI_DB1 DATASET.

Metric	Category	Train	Val	Test
Ratio of size	Small (area $\leq 32^2$)	0.21	0.22	0.22
	Medium ($32^2 <$ area $\leq 96^2$)	0.48	0.44	0.48
	Big(area $> 96^2$)	0.31	0.34	0.30
Mean/standard deviation	Number of strawberry instances	5.8/2.9	5.7/2.7	5.7/2.8
	Ratio of strawberry pixels per image (%)	5.6/2.7	5.7/2.4	5.4/2.5

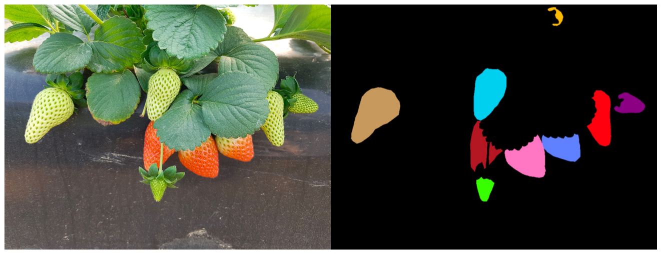

Fig. 1. Example annotations of the StrawDI_Db1 dataset.

III. PROPOSED MODEL

A. Meta architecture

Our proposed model is designed to be lightweight and efficient in performing strawberry instance segmentation, of which the architecture is shown in Fig. 2. It is constructed by two main parts: an encoder and a decoder. The encoder consists of a backbone and a Feature Pyramid Network (FPN) [14], which extracts contextual information from images and builds multi-scale features for later prediction. The decoder is anchor-free and does not require a Region Proposal Network (RPN) to generate anchors, it mainly contains two branches and predicts class and masks directly based on features extracted by the encoder.

B. Encoder

1) Backbone: Modelling in computer vision has been dominated by CNNs for a long time. On the other hand, the tremendous success of Transformer [15] in the language domain inspired the emergence of Vision Transformer (ViT [16]) [17]. Compared with CNNs, ViT offers a powerful approach to capturing global dependencies and contextual understanding in images. The attention mechanism plays a crucial role in capturing relationships between different parts of an image in ViT, enabling ViT to attend to and aggregate information from all image patches simultaneously. Therefore, instead of directly using CNNs like ResNet [18] as the backbone in previous similar work [19], we propose an efficient attention-based backbone to extract the feature from input images.

The vanilla self-attention is calculated by Eq. (1). Firstly, the input embedding, including positional encoding, is linearly transformed into three sets of vectors: query Q, key K and value V. Then, the attention scores are computed using the scaled dot-product attention mechanism. For each token, its attention to other tokens is determined by the dot product of its query vector with the key vectors of other tokens. Next, The result is scaled by the square root of the dimension of key $\sqrt{d_k}$. The attention scores are normalized using the Softmax function to obtain attention weights. Finally, the value V are multiplied by the attention weights, and the resulting weighted vectors are summed to produce the attention output.

$$Attention = Softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

The vanilla self-attention in ViT undergoes a sequence of steps that contribute to its progressive and effective modelling of relationships between different parts of an input image [16]. However, the vanilla self-attention has a quadratic time and space complexity concerning the sequence length, which makes them computationally expensive.

- **Efficient Self-Attention** To alleviate the problem, we adopt the Efficient Self-Attention (ESMA, [20]) as the basic block of our backbone, the detail is shown in Fig. 3. Firstly, a set of linear layers is adopted on 2D input token $[n, d_m]$ to obtain query Q. Then, the input token is reshaped to 3D one $[n, h, w]$ and performs a depth-wise convolution operation to reduce to $[n, h/s, w/s]$ by a factor s. Next, similar to the vanilla self-attention, the attention scores are computed using the scaled dot-product attention mechanism. Before the Softmax operation, $PWConv$ is used to model the interactions among different heads, which is a 1×1 pixel-wise convolutional layer, as shown in Eq. (2). In the end, the resultant values from each head are concatenated and subjected to a linear projection to create the final output.

$$ESMA = Softmax(PWConv(\frac{QK^T}{\sqrt{d_k}}))V \tag{2}$$

$$PE(x) = x \times \sigma(DWConv(x)) \tag{3}$$

- **Patch Embedding** Attention is originally designed for processing sequences of data, to apply it to images, it is necessary to convert the spatial information of the 3D image into a 2D sequence. Here we use a stack of three 3×3 convolutional layers with stride=3/1/2, padding=1/1/1, as shown in Eq. (3). Batch Normalization and ReLU activation are applied sequentially for the first two layers. The first two convolutional layers downsample and adjust channel dimensions, while the third further reduces spatial dimensions and increases output channels. Positional encoding is applied after the third convolutional layer, making it suitable for integration into our attention-based backbone.

To facilitate different scenarios, three different backbone variants (Tiny, Small and Base) are designed. The pipeline of the backbone is shown in Fig. 3, and the specification of backbone variants is shown in Table II, of which N is the number of blocks, C is the number of embedded dimensions and H is the number of self-attention heads.

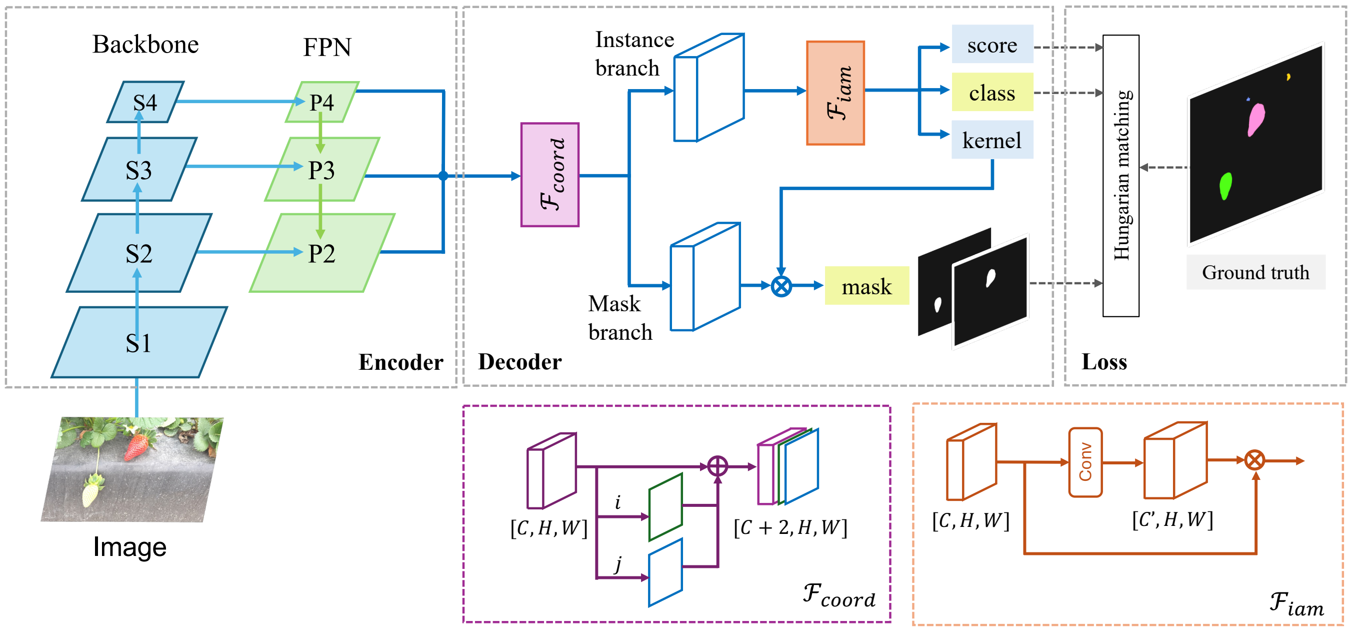

Fig. 2. The architecture of our proposed model.

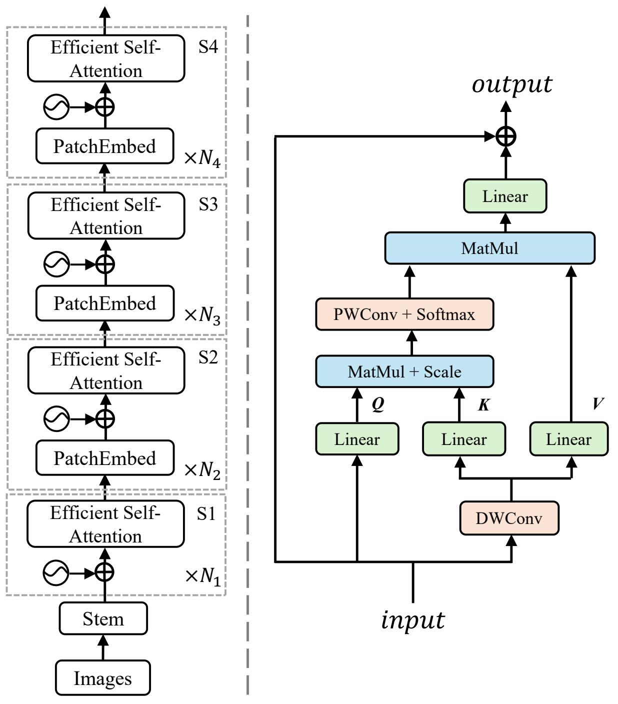

Fig. 3. **Left:** The pipeline of backbone. **Right:** Efficient Self-Attention.

TABLE II
BACKBONE ARCHITECTURE VARIANTS.

Stage	Stride	Backbone-T	Backbone-S	Backbone-B
S0: Stem	1/4	C=64	C=64	C=96
S1: ESAB	1/4	N=2 C=64 H=1	N=2 C=64 H=1	N=2 C=96 H=1
S2: ESAB	1/8	N=2 C=128 H=2	N=2 C=128 H=2	N=2 C=192 H=2
S3: ESAB	1/16	N=2 C=256 H=4	N=6 C=256 H=4	N=4 C=384 H=4
S4: ESAB	1/32	N=2 C=512 H=8	N=2 C=512 H=8	N=2 C=768 H=8

2) Feature Pyramid Network: The FPN introduces a top-down architecture where higher-resolution feature maps from earlier stages of the backbone are combined with lower-resolution feature maps from later stages. This is achieved through lateral connections, which involve upsampling the higher-level features and element-wise addition with the lower-level features. The pyramid typically consists of feature maps at different resolutions, different levels represent features at different scales. These scales correspond to different receptive fields and are crucial for handling objects of various sizes. Here we apply a convolutional layer to aggregate the features of three levels into one at last.

C. Decoder

We adopt the simple decoder of SparseInst [21] to decode the features to predictions, which mainly consists of two branches. Before entering any branch, the feature generated by the Encoder passes the CoordConv Module F_{coord}.

- **The CoordConv Module** F_{coord} is implemented as a simple extension of standard convolution [22]. The details of F_{coord} are shown in Fig. 2. Given an input feature $[C, H, W]$, two coordinate channels i and j with the size of $[1, H, W]$ are created. Specifically, within i, the first row is filled with 0, the second row is filled with 1, the third row is filled with 2, etc. The j channel is similar but with columns filled in with constant values instead of rows. Then, both i and j coordinate values are linearly scaled to fall in the range [–1, 1]. Finally, channels i and j are concatenated with the input feature, resulting in an output $[C + 2, H, W]$. For convolution over two dimensions, two coordinates (i, j) are sufficient to completely specify an input pixel. It is noted that two channels are generated by coordinates thus no extra parameters are introduced, which is friendly to building a lightweight model.

1) Instance Branch: This branch consists of an Instance Activation Map (IAM) and three prediction heads. The Instance Activation Map (IAM) was inspired by CAM [23], which suggests that objects can probably be found in informative regions. The features extracted from the highlighted areas are rich in semantic information and exhibit instance awareness, aiding in the recognition and differentiation of strawberries. The details of F_{iam} is shown in Fig. 2. F_{iam} is a 3×3 convolutional layer with 4 groups to aggregate instance features by concatenating features from a group. Given an input feature $[C, H, W]$, the output computed by F_{iam} is with the shape $[C', H, W]$, in which C' is the pre-set number of instance activation maps. Then the sparse instance features can be calculated by multiplying output (normalize to 1) and transposed input. Finally, the sparse instance-aware features are forwarded to three prediction heads to predict score, class and kernel.

2) Mask Branch: Validated by similar work SOLOv2 [24], it is feasible to use trained parameters as the kernel to perform mask prediction. Given the feature generated by FPN and the instance-aware mask kernels generated by the instance branch, the segmentation mask for each instance can be produced by $m_i = w_i \cdot M$, where m_i is the i-th predicted mask and corresponding kernel w_i, and M is the features. The final segmentation masks adopt bilinear interpolation to upsample to the original resolution.

D. Loss Function

We follow by DETR [25], which treats the label assignment problem as a bipartite matching problem. Firstly, a pair-wise dice-based matching score $C(i, k)$ for i-th prediction and k-th ground-truth object is introduced in Eq. (4), which is determined by classification scores and dice coefficients of segmentation masks.

$$C(i,k) = p_{i,c_k}^{1-\alpha} \cdot Dice(m_i, t_k)^{\alpha} \tag{4}$$

where α is a weight for two predictions segmentation=0.8, classification=0.2, c_k is the category label for the k-th ground-truth target and p_{i,c_k} is the probability for the category c_k of i-th prediction. The Dice is defined in Eq. (5)

$$Dice(m,t) = \frac{2\Sigma_{x,y} m_{xy} \cdot t_{xy}}{\Sigma_{x,y} m_{xy}^2 + \Sigma_{x,y} t_{xy}^2} \tag{5}$$

where m_{xy} and t_{xy} refer to the value of pixel located at x,y in predicted mask m and ground truth t respectively. Then, Hungarian algorithm is adopted to solve the matching matrix and find the best match between K ground-truth targets and N predictions. The training loss is defined in Eq. (6)

$$\mathcal{L} = \lambda_c \cdot \mathcal{L}_{cls} + \lambda_s \cdot \mathcal{L}_s + \mathcal{L}_{mask} \tag{6}$$

where $\mathcal{L}_{cls}$ is focal loss [26] for classification, $\mathcal{L}_{mask}$ is the dice loss for mask, and the $\mathcal{L}_s$ is the binary cross entropy loss for score. λ_c and λ_s are loss weights that are set to 2.0 and 1.0.

IV. Experiments and Results

A. Experiments

1) Implementation Details: We conducted experiments on Detectron2 [27] using Python 3.9.13 and PyTorch 1.13 on a computer with an Intel Xeon Gold 6152 @2.1 GHz CPU, 2 Nvidia Tesla P100 GPUs and 32.0GB Memory. The training and testing set of StrawDI_Db1 is used to train and test our proposed model.

During training, the batch size is set to 16 with 27K iterations in all, we use AdamW optimizer and the initial learning rate is set to 0.005 and divided by 10 at iteration 18K and 24K. No pre-trained weights are used and the parameters of the backbone are initialized by a normal distribution. The training data augmentation strategy contains random horizontal flipping, resizing the input images such that the short edge is one of 416, 448, 480, 512, 544, 576, 608 or 640 pixels while the longest is at most 853.

During inference, the data augmentation strategy is only the resizing input images such that the shortest edge is 640 pixels while the longest is at most 853.

2) Evaluation Metrics: In this study, the average precision (AP) and average recall (AR) are selected to measure the performance of segmentation models. The definitions of precision and recall are shown in Eq. (7) and Eq. (8).

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

where TP is the number of cases in which the target is a strawberry and is correctly detected, FP is the number of cases in which the target is not a strawberry, but is wrongly detected, and FN is the number of cases in the target is a strawberry, but it is not detected. The mean average precision (AP) serves as our primary metric for evaluating model performance. It is computed by averaging across 10 Intersection over Union (IoU) thresholds, ranging from 0.50 to 0.95.

3) Main results: Fig. 4 illustrates the training loss of our proposed model with different backbones. The losses stabilized at the end of the training, indicating that models are fully trained and converged.

Mask R-CNN is a state-of-the-art instance segmentation model that has been applied to strawberry images. An original [11] and a simplified version [12] of Mask R-CNN have been used to perform strawberry instance segmentation. Additionally, a fully convolutional neural network [13] has been proposed to solve the same task more efficiently. Table III summarizes the results of their models [13] and our proposed models on the StrawDI_Db1 dataset testing set. As shown, our proposed model with a Base backbone achieves the highest AP with 70.22.

Firstly, all of our proposed models demonstrate significant improvement over previous work, even our lowest performer, the Tiny backbone, has an AP 21.46, 22.97, and 14.21 higher

TABLE III

INSTANCE SEGMENTATION RESULTS ON STRAWDI_DB1 TESTING SET.

Model	Backbone	AP	AP_{50}	AP_{75}	AP_{small}	AP_{medium}	AP_{large}
Mask R-CNN [11]	Res50	45.36	76.57	47.09	07.35	50.03	78.03
Mask R-CNN' [12]	Res50	43.85	74.24	45.13	07.54	51.77	75.90
FCNN [13]	Res50	52.61	69.24	57.84	16.96	65.26	53.31
Our model	Tiny	**66.82**	85.99	71.78	28.53	70.25	87.67
	Small	**69.39**	87.32	73.96	30.04	71.85	92.15
	Base	**70.22**	87.70	76.05	31.44	73.63	90.29

than the original, simplified Mask R-CNN and FCNN respectively. Secondly, model performance gains are progressively enhanced as backbone complexity and capacity increases. For example, models with Small and Base backbones deliver 2.57 and 3.4 higher AP than the Tiny backbone. We assume that more features can be provided by more layers and bigger embedded dimensions, which is helpful in locating the targets. Thirdly, the AP_{50} of our models are larger than AP_{75}, and of which gaps between AP_{50} and AP_{75} are narrower than the original and simplified Mask R-CNNs, indicating that our models usually output high-accurate results regardless of different IoU criteria. Finally, our models demonstrate better performance when dealing with medium and large strawberries than small strawberries. We suggest that the reasons for this can be that small strawberries have a similar colour to leaves and are normally covered, and they can be lost when resizing the input images to smaller ones.

TABLE IV

PARAMS AND FLOP OF THE MODELS.

Model	Backbone	Param/M	FLOPs/G
Mask R-CNN[27]	Res50	35.08	877.4
Our model	Tiny	**17.42**	**78.3**
	Small	20.58	86.9
	Base	33.54	111.7

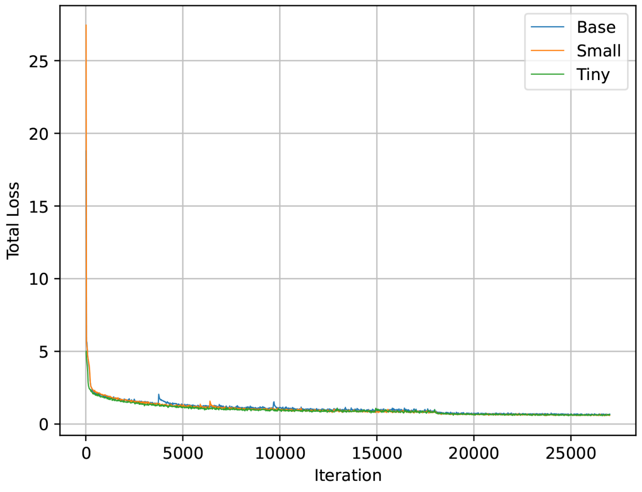

Fig. 4. Training loss of our proposed model.

4) Model complexity: To measure the model size and complexity, the number of learnable parameters and the number of floating-point operations during training are computed. Previous work ([11], [12], [13]) did not provide information about their model size and complexity, therefore we compare the complexity between Mask R-CNN with Res50 from Detectron2 and our proposed model, the results are shown in the Table IV.

- **Learnable Parameters (Params)**: Learnable parameters are the variables in a neural network that are learned from the training data, which include weights and biases in the case of fully connected or convolutional layers. All of our models have fewer parameters compared to Mask R-CNN, among which the Tiny backbone has less than half the number of parameters compared to Mask R-CNN. This indicates that our model design is lightweight and efficient, making the models suitable for resource-constrained environments.
- **Floating-Point Operations (FLOPs)**: Floating-point operations are arithmetic operations (additions and multiplications) involving floating-point numbers. FLOPs is a measure of the computational workload during the forward pass. Here we use 100 images from the testing set to compute the FLOPs. Our proposed models generally require significantly fewer FLOPs for each image during inference compared to Mask R-CNN. This suggests that our models are computationally efficient.

In summary, our proposed models not only have fewer parameters and require fewer floating-point operations during inference compared to the Mask R-CNN but also demonstrate a trade-off between model complexity and computational efficiency, which offer options for scenarios with strict resource constraints. The models with Tiny and Small backbones offer lightweight options for scenarios with strict resource constraints, while the Base backbone provides a higher-capacity variant for tasks that demand more accurate strawberry segmentation.

B. Visualization

We visualize the performance of our model in Fig. 5. The model can segment strawberries under various conditions. There are some difficult cases in which strawberries are located at the side of the image or are partially covered, however our proposed models can segment them accurately.

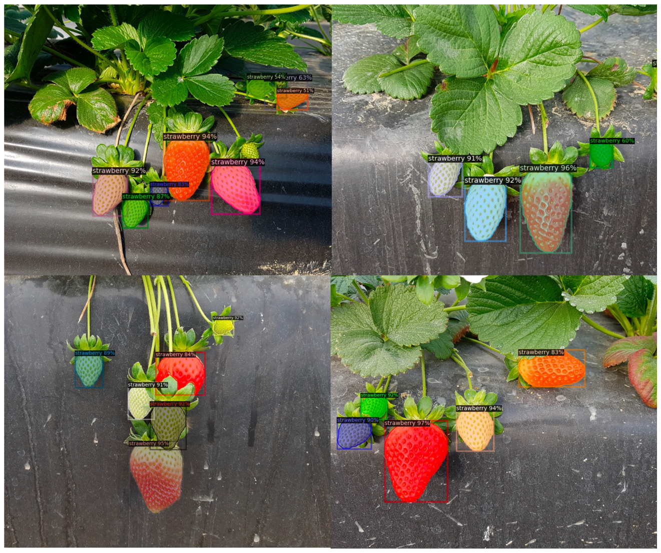

Fig. 5. Visualization of our proposed model segmentation.

V. Conclusions

Accurately detecting and segmenting each strawberry within real-world production environments is pivotal for the development of automatic strawberry harvesting robots. In this paper, we propose lightweight and attention-based CNN models for strawberry instance segmentation. The simple models consist of an encoder (a backbone and a FPN) and a decoder. Our models outperform the original and simplified Mask R-CNN with significant 21.46 and 22.97 AP improvements respectively, among which the one with Base achieves the highest AP of 70.22. Besides, our models require much fewer parameters and FLOPs compared to Mask R-CNN. In summary, this study introduces lightweight, efficient, and effective models designed for strawberry instance segmentation. These models hold promise for deployment on embedded devices with limited computational resources in the future.

References

[1] H. Kang and C. Chen, "Fruit detection, segmentation and 3D visualisation of environments in apple orchards," *Computers and Electronics in Agriculture*, vol. 171, p. 105302, Apr. 2020.

[2] J. Lu, P. Chen, C. Yu, Y. Lan, L. Yu, R. Yang, H. Niu, H. Chang, J. Yuan, and L. Wang, "Lightweight green citrus fruit detection method for practical environmental applications," *Computers and Electronics in Agriculture*, vol. 215, p. 108205, Dec. 2023.

[3] B. J. McGuinness, M. D. Duke, K. C. Au, and H. S. Lim, "Field factory for the automated harvesting of forestry tree stock," *Biosystems Engineering*, vol. 227, pp. 52–67, Mar. 2023.

[4] M. Peebles, S. H. Lim, M. Duke, and B. McGuinness, "Investigation of Optimal Network Architecture for Asparagus Spear Detection in Robotic Harvesting," *IFAC-PapersOnLine*, vol. 52, pp. 283–287, Jan. 2019.

[5] S. Islam, H. Elmekki, A. Elsebai, J. Bentahar, N. Drawel, G. Rjoub, and W. Pedrycz, "A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks," June 2023. arXiv:2306.07303 [cs].

[6] R. Kestur, A. Meduri, and O. Narasipura, "MangoNet: A deep semantic segmentation architecture for a method to detect and count mangoes in an open orchard," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 59–69, Jan. 2019.

[7] W. Jia, Z. Zhang, W. Shao, S. Hou, Z. Ji, G. Liu, and X. Yin, "FoveaMask: A fast and accurate deep learning model for green fruit instance segmentation," *Computers and Electronics in Agriculture*, vol. 191, p. 106488, Dec. 2021.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, Oct. 2017. ISSN: 2380-7504.

[9] T. T. Santos, L. L. de Souza, A. A. dos Santos, and S. Avila, "Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association," *Computers and Electronics in Agriculture*, vol. 170, p. 105247, Mar. 2020.

[10] W. Jia, Y. Tian, R. Luo, Z. Zhang, J. Lian, and Y. Zheng, "Detection and segmentation of overlapped fruits based on optimized mask R-CNN application in apple harvesting robot," *Computers and Electronics in Agriculture*, vol. 172, p. 105380, May 2020.

[11] Y. Yu, K. Zhang, L. Yang, and D. Zhang, "Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN," *Computers and Electronics in Agriculture*, vol. 163, p. 104846, Aug. 2019.

[12] I. Pérez-Borrero, D. Marín-Santos, M. E. Gegúndez-Arias, and E. Cortés-Ancos, "A fast and accurate deep learning method for strawberry instance segmentation," *Computers and Electronics in Agriculture*, vol. 178, p. 105736, Nov. 2020.

[13] I. Perez-Borrero, D. Marin-Santos, M. J. Vasallo-Vazquez, and M. E. Gegundez-Arias, "A new deep-learning strawberry instance segmentation methodology based on a fully convolutional neural network," *Neural Computing and Applications*, vol. 33, pp. 15059–15071, Nov. 2021.

[14] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, July 2017. ISSN: 1063-6919.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *arXiv:1706.03762 [cs]*, Dec. 2017. arXiv: 1706.03762.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," June 2021. arXiv:2010.11929 [cs].

[17] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," Aug. 2021. arXiv:2103.14030 [cs].

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. ISSN: 1063-6919.

[19] Z. Zhao, Y. Hicks, X. Sun, and C. Luo, "Peach ripeness classification based on a new one-stage instance segmentation model," *Computers and Electronics in Agriculture*, vol. 214, p. 108369, Nov. 2023.

[20] Q. Zhang and Y. Yang, "ResT: An Efficient Transformer for Visual Recognition," Oct. 2021. arXiv:2105.13677 [cs].

[21] T. Cheng, X. Wang, S. Chen, W. Zhang, Q. Zhang, C. Huang, Z. Zhang, and W. Liu, "Sparse Instance Activation for Real-Time Instance Segmentation," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (New Orleans, LA, USA), pp. 4423–4432, IEEE, June 2022.

[22] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski, "An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution," Dec. 2018. arXiv:1807.03247 [cs, stat].

[23] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization," Dec. 2015. arXiv:1512.04150 [cs].

[24] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "SOLOv2: Dynamic and Fast Instance Segmentation," Oct. 2020. arXiv:2003.10152 [cs].

[25] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," May 2020. arXiv:2005.12872 [cs].

[26] 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, pp. 318–327, Feb. 2020. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[27] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "facebookresearch/detectron2," 2019.