Journal of Information Technology and Computer Science
Volume 11, Number 2, August 2026, pp. 162-171
Journal Homepage: www.jitecs.ub.ac.id

Implementation of YOLOv8 Instance Segmentation for Automatic Counting and Identification of Multispecies Bacterial Colony

Dimas Firmanda Al Riza*[1]

[1]Universitas Brawijaya, Malang
[1]dimasfirmanda@ub.ac.id
*Corresponding Author

Received 23 April 2024; accepted 04 May 2026

Abstract. This study introduces the implementation of YOLOv8 instance segmentation model, for automating the quantification and identification of multispecies bacterial colonies. In microbiological research, accurate counting and rapid identification of various species within colonies are essential. YOLOv8's real-time capabilities and high accuracy make it particularly well-suited for this task. We demonstrate its efficacy in accurately detecting and segmenting individual bacterial colonies, even when they comprise multispecies like *Bacillus subtilis, Escherichia coli, Pseudomonas aeruginosa, and Staphylococcus aureus*. This innovation streamlines the labor-intensive processes of colony counting and species identification. To improve precision, we incorporate post-processing techniques to handle overlapping colonies, significantly enhancing accuracy compared to manual counting methods. Our proposed method exhibits superior performance and provides a valuable tool for microbiologists and researchers, expediting bacterial colony analysis. In this study, we fine-tuned hyperparameters to achieve the best mean average precision (mAP) for masks and bounding boxes. We used a primary and a secondary dataset for training. The hyperparameter modifications, including using an Adam optimizer with a learning rate of 0.01 and an epoch value of 50, resulted in mAP values of 90%. These findings underscore the importance of optimizing the optimizer type, learning rate, and the number of epochs, revealing their impact on the model's performance in automating bacterial colony quantification and identification.

Keywords: colony counting, instance segmentation, learning rate, optimizer, YOLOv8

1 Introduction

In different areas such as microbiology, biotechnology, and medical research, accurate counting and identification of bacterial colonies play a major role. The traditional manual counting methods are time-consuming and prone to human error. Automated systems that can efficiently perform these tasks therefore need to be developed. Computer vision techniques and machine learning models have been used to develop automated colony counting and identification systems over the last few years [1].

Apart from YOLO, there are several other frameworks or methods of

automatic colony counting including the Convolutional Deep Belief Network (CDBN) and Support Vector Machine (SVM). The CDBN, an unsupervised deep learning method, was initially employed for segmenting images into colonies, agar, plates, and other boundary artifacts [2]. The SVM is utilized to train and distinguish between foreground and background patches. Then, to predict colonies from a pool, the foreground patch will be transmitted to CNN [3]. Existing research results show that deep learning is generally superior to hand-drawn features and traditional reference methods. In a similar study, SVM and CNN have been applied to automated counting of colonies and identification of the different bacteria [4]. In order to obtain image descriptors, CNN is used to encode and classify them by SVM or Random Forest. However, most studies use multi-stage or two-stage object detection architectures, i.e. regions that are extracted and classified to improve object localization, even though deep learning in microbial analysis has proven very successful. In contrast, it is often easier and faster to use YOLO with a single stage object detection architecture to count microbial colonies. The fact that YOLO may also be used for colony segmentation, not only identification and grouping, is another advantage of using this method in comparison with others.

This research focuses on the use of a YOLOv8 instance segmentation model, which offers an innovative solution for simplifying and enhancing bacterial colony analysis. By enabling real-time, high-throughput analysis of bacterial colonies in multispecies samples, YOLOv8 promises to revolutionize this field, with its remarkable speed and accuracy. This approach is expected to substantially reduce labor demands while improving accuracy and consistency. The theoretical basis of YOLOv8 and its adaptation to the unique challenges of bacterial colony analysis will be explored in the following sections of this journal. In addition, we will look at the experimental settings by comparing two hyperparameters of our research like optimizer and learning speed. In addition, the results of our implementation will be discussed, demonstrating the model's ability to automatically count and identify bacterial colonies with remarkable accuracy. This journal is an innovative attempt to bridge the gap between cutting edge computer vision technology and the field of microbiology. We want to leverage the power of YOLOv8 in order to create a valuable tool that can have an important effect on many different industries where bacterial colony analysis is required. In this study, four commonly used bacterial species are employed.

2 Method

The research employs deep learning with YOLOv8, conducting a comparison of different optimizers and learning rate settings to achieve the highest Mean Average Precision (mAP). Deep learning will be applied to detect bacterial species based on colony morphology on agar plates.

2.1 Data Acquisition

Images of bacterial species *Bacillus subtilis*, *Escherichia coli*, *Pseudomonas aeruginosa*, and *Staphylococcus aureus* are included in this dataset. This study uses both primary and secondary data. For the primary data, samples were collected using the Total Plate Count (TPC) method in the Laboratory of Microbiology and the Laboratory of Biotechnology, Department of Food Science and Biotechnology, Universitas Brawijaya. A total of 36 primary images comprising colonies of *E. coli* and *B. subtilis* were obtained.

The Annotated Germs for Automated Recognition (AGAR) dataset was used as the secondary data source, obtained from Neurosys Research [5]. A total of 214

lower-resolution images from the AGAR dataset were utilized, resulting in 250 images overall. Primary data acquisition was conducted in the Bio-AI Laboratory, Department of Agricultural Technology, Universitas Brawijaya, using a Canon EOS 700D camera with JPG format and a resolution of 3456 × 5184 pixels. Images from the AGAR dataset are also in JPG format with a resolution of 2048 × 2048 pixels. The dataset was then divided into training, validation, and testing sets.

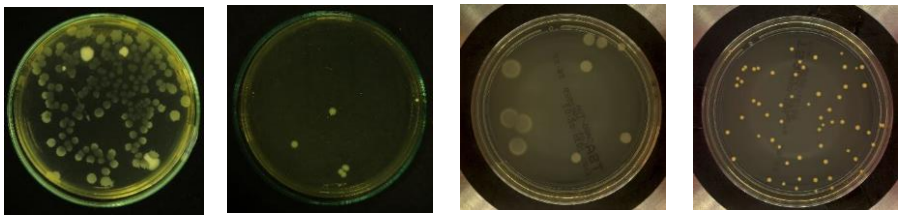

Figure 1. Bacteria colonies dataset. (a) *Bacillus subtilis*, (b) *Escherichia coli*, (c) *Pseudomonas aeruginosa* (d) *Staphylococcus aureus*

2.2 Data Preprocessing

The retrieved dataset needs to be preprocessed before use. This technique is required to normalize the data and remove duplicates. The previously acquired data are still in large pixel dimensions, making the training process computationally heavy [6]. Therefore, adjustments to the size of the data should be made. This preprocessing stage involves data resizing for all image to 640×640 pixels. Roboflow software has been used for image annotation. The data was processed and labeled so that it can be effectively learned by the model. We use polygon tools for resulting in instance segmentation models. The annotated objects are the identified species of bacteria on agar plates.

2.3 Data Augmentation

In image recognition tasks within deep learning, data augmentation is extensively utilized. Its purpose is to expand the size and diversity of training images by applying various transformations and modifications to existing image data to generate new samples. Data enhancement is aimed at enhancing the generalization ability of deep learning models and mitigating overfitting [7].

In this study, we applied augmentation techniques as shown in Table 1. From the augmentation results, a dataset consisting of 450 training samples, 43 validation samples, and 21 test samples was produced.

Table 1. Data Augmentation

Technique	Applied
Flip	Horizontal, Vertical
90º Rotate	Clockwise, Counter-Clockwise, Upside Down
Crop	7% Minimum Zoom, 55% Maximum Zoom
Rotation	Between -32º and +32º

2.4 YOLOv8-seg

The YOLO (You Only Look Once) family is a deep learning framework widely used for object detection and segmentation. YOLOv8, as an evolution of earlier versions such as YOLOv5, maintains a similar overall design while

introducing architectural improvements that enhance performance and efficiency. It is available in five scalable variants (n, s, m, l, x), which differ in depth and width to balance accuracy and computational cost [8].

YOLOv8-seg is a one-stage deep learning model designed for object detection and instance segmentation. It introduces improvements such as the C2f module for enhanced feature extraction and a decoupled head for better task-specific learning. These modifications enable efficient and accurate segmentation of bacterial colonies, making the model suitable for real-time microbiological analysis. Head section of YOLOv8 adopting widely utilized decoupled head structure, employs Task Aligner positive sample assignment strategy for loss function calculation and introduces a distribution focal loss [9]. During training, mosaic augmentation is disabled in the final epochs to refine performance. The architecture consists of a backbone and a head, further divided into neck and segmentation components. As demonstrated in [10], enhanced YOLOv8-seg architectures with improved feature fusion can achieve higher segmentation accuracy and real-time performance, particularly in complex scenarios involving occlusion, overlapping objects, and varying surface characteristics.

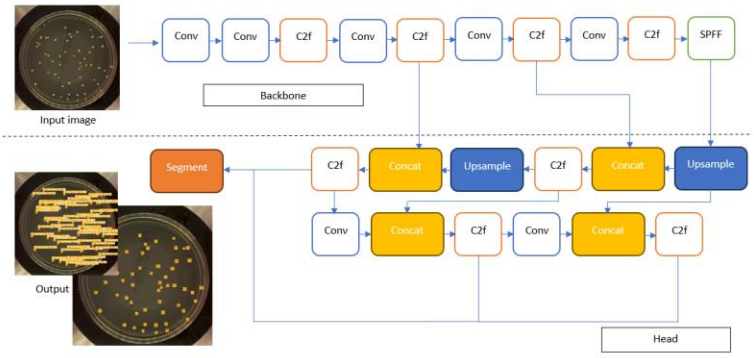

Figure 2. Yolov8-seg Architecture

2.5 Hyperparameter tuning Optimizer and Learning Rate

Stochastic Gradient Descent algorithm (SGD) is an optimization algorithm the model loss function [11]. The SGD recalculates the weights of each training example x_i, y_i instead of computing a gradient across all training examples and adjusting their weights collectively. The algorithm is based on random selection of training sample minibatches, which are periodically updated with the model parameters. The algorithm seeks to find the optimum values that minimize loss function by iterative update of model parameters with SGD. The randomness introduced by the minibatch selection helps SGD escape local minima and explore the optimization landscape more efficiently. The equation for the SGD algorithm can be represented in the following way [12]:

$$w = w - lr.\nabla_w L(x_i, y_i, W) \quad (1)$$

Adaptive Moment algorithm (Adam) is a combination of Momentum and RMSProp [13]. It also computes personalized learning rates for each parameter. For each parameter, it also calculates a personal learning rate. Adam maintains an

exponentially decaying average of previous gradient squares v_t like AdaDelta and RMSProp also preserves an exponentially decaying average of previous gradient m_t similar to Momentum. The Adam algorithm equation is presented in the following way:

$$\mathrm{m} \leftarrow \beta_1 \mathrm{m} - (1-\beta_1)\nabla_\theta J(\theta) \quad (2)$$

$$v_t \longleftarrow \beta_2 v_t + (1-\beta_2 \nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta) \quad (3)$$

$$m \leftarrow m \otimes (1-\beta_1^t) \quad (4)$$

$$v_t \leftarrow \frac{v_t}{(1-\beta_2^t)} \quad (5)$$

$$\theta \leftarrow \theta + \frac{\eta m}{\sqrt{v_t + \mathrm{U}}} \quad (6)$$

Root Mean Square Propagation algorithm (RMSProp) is an adaptive learning rate method that addresses the limitations of traditional stochastic gradient descent (SGD) optimizer [14] RMSProp adjusts the learning rate based on an exponentionally decreasing average of squared gradients. This allows an algorithm to adapt its learning rate for each parameter, depending on the magnitude of the gradient. By doing so, RMSProp helps to accelerate convergence and effectively manage the different scales of the gradients. The RMSProp algorithm equation is presented in the following way:

$$s \leftarrow \beta s + (1-\beta)\nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta) \quad (7)$$

$$\theta \leftarrow \theta - \frac{\eta \nabla_\theta J(\theta)}{\sqrt{s + \mathrm{U}}} \quad (8)$$

2.6 Evaluation

The trained model will be evaluated to determine the effectiveness of the resulting metrics, which are based on mean average precision, recall, and mAP. Precision determines how accurate the model's positive predictions are. It considers the percentage of correctly predicted positive cases among all positively predicted cases. The ratio of true positive predictions to the total of true positive and false positive predictions is used to compute precision. The model's low rate of false positive predictions, indicated by a high precision value, suggests that the objects it classifies as positive are probably accurate [15]. Precision can be written using equation 9 [16].

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (9)$$

Recall is the metric used to evaluate how effectively a model performs in accurately identifying positive cases from a given set of data. Consideration is given to the precentage of real positive cases from all actual positive results in the dataset that the model correctly identified. Since the model correctly detects a significant percentage of the real positive instances, a high recall value suggests that the model is effective in identifying positive instances [17]. The recall may be written in equation 10 [16].

$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (10)$$

A comprehensive model performance metric that integrates precision and recall is called Mean Average Precision (mAP) [18]. It is calculated using the precision-recall curve, which takes into consideration the balance between actual positive rate (recall) and the positive predictive value (precision) [19]. By modifying the detection threshold, which produces the precision-recall curve, the precision and recall at each detection threshold are identified. The average precision (AP) is then calculated using the area under the precision-recall curve. The mean of the AP values over many classes or instances is known as the mAP [6]. Equation 11 can be used to write the mAP equation.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \quad (11)$$

3 Results and Discussion

The results of YOLOv8 model segmentation performance evaluation are explained through performance metrics such as precision, recall and mAP values. The YOLOv8n-seg was developed using Python 3.10.12 and Torch 2.1.0. The input image size is set to 640 × 640 for all image datasets. The model was trained using Google Colaboratory or Google Colab. In this research, the YOLOv8n (nano) segmentation model and its hyperparameter configuration are described in Table 2.

Table 2. Configuration model YOLOv8

Configuration	Value
Model	YOLOv8n-seg
Size	640 × 640
Epoch	50
Batch	32

The ability to generalize well is one of the most important aspects of deep learning precision, which means that the model performs well not only on training data but also on unseen test data. In order to avoid overfitting and improve generalization, regularisation techniques are often applied [20]. Additionally, deep learning models can be fine-tuned and optimized to achieve high performance and accuracy [21]. In this paper, we used three different optimizers and l learning rates, namely SGD, Adam, and RMSProp with learning rates of 0.01, 0.001 and 0.0001, respectively. Model performance results are shown in Table 3.

Table 3. Model performance using different combinations of Optimizer and Learning Rate

Optimizer	Learning Rate	Precision	Recall	mAP@50
SGD	0.01	0.853	0.826	0.869
	0.001	0.805	0.775	0.827
	0.0001	0.522	0.581	0.653
Adam	0.01	0.840	0.866	0.908
	0.001	0.846	0.805	0.867
	0.0001	0.787	0.794	0.851
RMSProp	0.01	0.27	0.02	0
	0.001	0.702	0.749	0.795
	0.0001	0.82	0.756	0.83

The number of colonies is then accurately detected by selecting the segmentation results of the model that yield the highest mAP value. The model's performance outcomes differ when the optimizer and learning rate are combined. Table 3 shows that the RMSProp optimizer with a learning rate of 0.01 yields the lowest mAP value of 0. A combination of the Adam optimizer and a learning rate of 0.01, which is 0.908, results in the highest mAP value. RMSProp shows that for all classes, the value of precision and recall is zero in Figure 3.

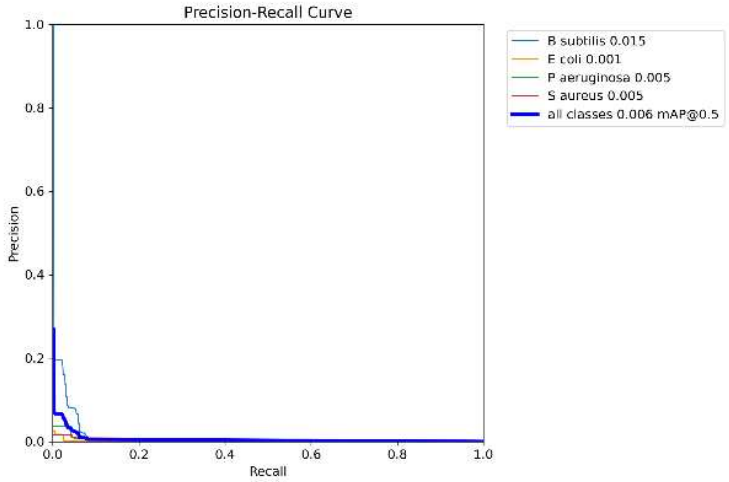

Figure 3. Precision–recall curves for all bacterial classes using the RMSProp optimizer with a learning rate of 0.01, showing near-zero detection performance across classes.

Figure 3 shows the performance of YOLOv8 using the RMSProp optimizer and learning rate 0.01 comprising curves for all bacterial classes in the PR curve. All classes of bacteria were calculated from the average of all mAP classes. Based on the curve, The majority of mAP values for all bacteria are zero; therefore, the overall mAP value becomes zero.

Figure 4 shows different result compared to Fig. 3. Each class of bacteria has different mAP values. *B. subtilis* has the largest mAP that is 0.985 followed by *E. coli* 0.95 mAP, *P. aeruginosa* 0.869 mAP and the smallest mAP is *S. aureus* which has 0.814 mAP.

Figure 5 shows the final validation using the test dataset. The figure shows that combinations capable of segmenting and identifying the type of bacterial colonies precisely not only in combinations that produce the highest mAP values but also in other combinations can show accurate results. In the SGD optimizer, the model can detect correctly at learning rates of 0.01 and 0.001 with mAP values of 0.869 and 0.827. In the meantime, SGD optimizers is not able to detect correctly the number and type of colonies in each plate when learning at a rate of 0.0001 with an mAP value of 0.653. In the use of the Adam optimizer, all three variational learning rates showed accurate and precision detection results. In the use of the RMSProp optimizer, there is a mismatch in the detection results at the learning rates of 0.01 and 0.001. However, at a learning rate of 0.0001 the RMSProp can show appropriate and precise results.

The dataset used in this study consists of 250 images collected from both primary and secondary sources, with unequal representation across bacterial species. This class imbalance may influence model learning by biasing predictions toward

more dominant classes. As a result, performance variations across species (e.g., lower mAP for *Staphylococcus aureus*) may partially reflect this imbalance. Although data augmentation was applied to increase dataset diversity, it may not fully compensate for the unequal class distribution.

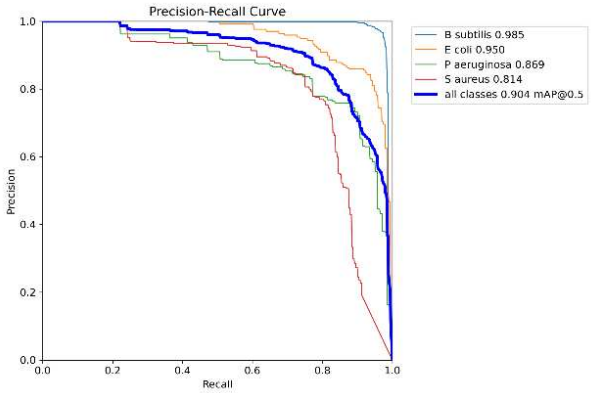

Figure 4. The precision recall curve of Adam optimizer with learning rate 0.01

Although the results demonstrate consistent performance trends across optimizer configurations, this study reports single-run evaluations due to dataset limitations. Therefore, the reported differences in mAP values should be interpreted with caution. Future work should incorporate repeated experiments, cross-validation, or confidence interval estimation to provide statistical significance and robustness in performance comparison.

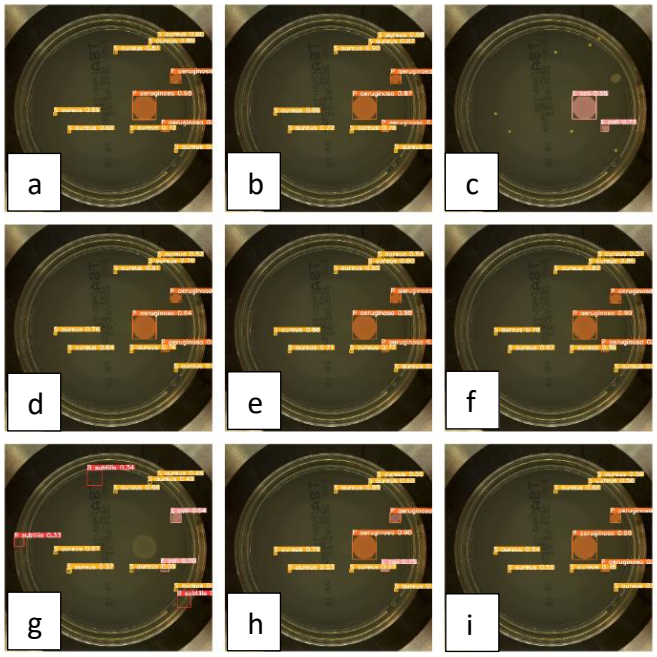

Figure 5. Bacterial Colony Detection Results with Different Optimizer and Learning Rate. (a) SGD; 0.01, (b) SGD; 0.001, (c) SGD; 0.0001, (d) Adam; 0.01, (e)

Adam; 0.001, (f) Adam; 0.0001, (g) RMSProp; 0.01, (h) RMSProp; 0.001, (i) RMSProp; 0.0001.

4 Conclusion

This paper presents the impact of different optimizers and learning rates on performance metrics using the YOLOv8n-seg model. The research focuses on colonies of *B. subtillis, E. coli, P. aeruginosa, and S. aureus*. The aim of this comparison is to identify the best combination that achieves the highest mean average precision (mAP) value. The results show that the best combination is the model trained using the Adam optimizer with a learning rate of less than 0.01. With a precision of 0.84 and a recall of 0.87, this combination achieves an mAP value of 0.908. It is expected that these findings can serve as a reference for future studies on simple, accurate, and fast detection and segmentation of microbial colonies. The results demonstrate that the use of YOLOv8 enables real-time, high-speed analysis of bacterial colonies in multispecies samples while improving the accuracy and consistency of detection results. In the future, further research and application of this technology can be expanded to explore its potential, advance microbiological detection techniques, and extend its use to other fields.

Acknowledgments. This research funded by Faculty of Agricultural Technology, Universitas Brawijaya through Hibah Doktor Non Lektor Kepala (DNLK) Grant 2023 with contract number 3508.2/UN10.F10/PT.01.03/2023.

References

1. A. D. Balomenos, P. Tsakanikas, Z. Aspridou, A. P. Tampakaki, K. P. Koutsoumanis, and E. S. Manolakos, "Image analysis driven single-cell analytics for systems microbiology," *BMC Systems Biology*, vol. 11, no. 1, Apr. 2017. doi: 10.1186/s12918-017-0399-z.
2. D. Nie, E. A. Shank, and V. Jojic, "A deep framework for bacterial image segmentation and classification," in *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, Sep. 2015. doi: 10.1145/2808719.2808751.
3. A. Ferrari, S. Lombardi, and A. Signoroni, "Bacterial colony counting with Convolutional Neural Networks in Digital Microbiology Imaging," *Pattern Recognition*, vol. 61., pp. 629–640, Jan. 2017. doi: 10.1016/j.patcog.2016.07.016.
4. B. Zieliński, A. Plichta, K. Misztal, P. Spurek, M. Brzychczy-Włoch, and D. Ochońska, "Deep learning approach to bacterial colony classification," *PLOS ONE*, vol. 12, no. 9, p. e0184554, Sep. 2017. doi: 10.1371/journal.pone.0184554.
5. S. Majchrowska et al., "AGAR a microbial colony dataset for deep learning detection." *arXiv*, 2021. doi: 10.48550/ARXIV.2108.01234.
6. R. S. Passa, S. Nurmaini, and D. P. Rini, "YOLOv8 Based on Data Augmentation for MRI Brain Tumor Detection," *Scientific Journal of Informatics,* vol. 10, no. 3, pp. 363-370, 2023.
7. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition." *arXiv*, 2014. doi: 10.48550/ARXIV.1409.1556.
8. J. Terven and D. Cordova-Esparza, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *arXiv*, 2023, doi: 10.48550/ARXIV.2304.00501.
9. C. Feng, Y. Zhong, Y. Gao, M. R. Scott, and W. Huang, "TOOD: Task-aligned One-stage Object Detection." *arXiv*, 2021. doi: 10.48550/ARXIV.2108.07755.
10. X. Yue, K. Qi, X. Na, Y. Zhang, Y. Liu, and C. Liu, "Improved YOLOv8-Seg Network for Instance Segmentation of Healthy and Diseased Tomato Plants in the Growth Stage,"

Agriculture, vol. 13, no. 8, p. 1643, Aug. 2023. doi: 10.3390/agriculture13081643.
11. L. Bottou, “Stochastic Gradient Descent Tricks,” Lecture Notes in *Computer Science*. Berlin Heidelberg: Springer, pp. 421–436, 2012. doi: 10.1007/978-3-642-35289-8_25.
12. R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, and P. Richtarik, “SGD: General Analysis and Improved Rates,” *arXiv*, 2019, doi: 10.48550/ARXIV.1901.09401.
13. E. Hassan, M. Y. Shams, N. A. Hikal, and S. Elmougy, “The effect of choosing optimizer algorithms to improve computer vision tasks: a comparative study,” *Multimedia Tools and Applications*, vol. 82, no. 11, pp. 16591–16633, Sep. 2022. doi: 10.1007/s11042-022-13820-0.
14. D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” *arXiv*, 2014. doi: 10.48550/ARXIV.1412.6980.
15. B. S. S. Rao and B. S. Rao, "An Effective WBC Segmentation and Classification Using MobilenetV3–ShufflenetV2 Based Deep Learning Framework," *IEEE Access*, vol. 11, pp. 27739-27748, 2023, doi: 10.1109/ACCESS.2023.3259100
16. Z. Ning, X. Wu, J. Yang, and Y. Yang, “MT-YOLOv5: Mobile terminal table detection model based on YOLOv5,” *Journal of Physics: Conference Series*, vol. 1978, no. 1, p. 012010, Jul. 2021. doi: 10.1088/1742-6596/1978/1/012010.
17. K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” *arXiv*, 2015. doi: 10.48550/ARXIV.1512.03385.
18. A. Nikfarjam, J. D. Ransohoff, A. Callahan, V. Polony, and N. H. Shah, “Profiling off-label prescriptions in cancer treatment using social health networks,” *JAMIA Open*, vol. 2, no. 3, pp. 301–305, Jul. 2019. doi: 10.1093/jamiaopen/ooz025.
19. W. Zhao, F. Chen, H. Huang, D. Li, and W. Cheng, “A New Steel Defect Detection Algorithm Based on Deep Learning,” *Computational Intelligence and Neuroscience*, vol. 2021, pp. 1–13, Mar. 2021. doi: 10.1155/2021/5592878.
20. C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning (still) requires rethinking generalization,” *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, Feb. 2021. doi: 10.1145/3446776.
21. M. Cowan, T. Moreau, T. Chen, and L. Ceze, “Automating Generation of Low Precision Deep Learning Operators.” *arXiv*, 2018. doi: 10.48550/ARXIV.1810.11066.