Available online at www.sciencedirect.com

Manufacturing Letters

Manufacturing Letters 35 (2023) 1052-1059

51st SME North American Manufacturing Research Conference (NAMRC 51, 2023)

Real-Time Tool Detection in Smart Manufacturing Using You-Only-Look-Once (YOLO)v5

Niloofar Zendehdel[a]*, Haodong Chen[a], Ming C. Leu[a]

[a]*Department of Mechanical and Aerospace Engineering, Missouri Universitry of Science and Technology, Rolla MO 65401, USA*

* Corresponding author. Tel.: +1-573-466-2017. *E-mail address:* nzx9d@mst.edu

Abstract

Computer vision plays an essential role in Industry 4.0 by enabling machinery to perceive, analyze, and control production processes. Object detection, a computer vision technique that accurately classifies and localizes objects within images, has gained significant interest. This technique can be applied in various domains, including manufacturing, to assist in the detection of different tools. In this paper, You-Only-Look-Once (YOLO)v5 real-time object detection technique has been developed and optimized, to detect different tool types and their locations in a manufacturing setting. To train the neural network, a dataset of 3,286 tool images from the internet has been collected and annotated. To enhance the model's ability in generalization, three augmented variants of each image have been created to improve rotation invariance. The model's training scheme has been further optimized with stochastic gradient descent after configuring different hyperparameters such as learning rate and momentum. The fine-tuned model achieved a mean average accuracy of 98.3%, demonstrating the high precision of the model in detecting different tool types and their locations in real-time.

Peer-review under responsibility of the Scientific Committee of the NAMRI/SME.
Keywords: Object Detection, Yolov5, Smart Manufacturing

1. Introduction

Industry 4.0 fosters standard manufacturing companies toward smart manufacturing through the Industrial Internet of Things (IIOT), big data analytics, Cyber-Physical Systems (CPS), Augmented Reality (AR), and robotics [1]. Notably, Computer vision lies at the core of many of these technologies, enabling machines and gadgets to perceive and comprehend their environments. As a technique of computer vision, object detection is a problem that deals with not only the class of objects (object classification) but also the location of each object within an image or video streams (object localization). This technique could be used for detecting and locating different types of tools in a manufacturing setting, helping robots with detecting required tools for a process, inventory management and quality control of assembly lines by monitoring the correctness of tools being used at each step.

The majority of current mainstream object detection algorithms are built on deep learning models and could be classified into two groups: single-stage detectors and two-stage detectors. As the name indicates, single-stage detectors try to classify and localize objects at one time (one-shot) using dense sampling. However, in two-stage detectors, one module attempts to identify an arbitrary number of region proposals using selective search Region Proposal Networks (RPN), and a separate module is responsible for the classification and localization adjustment based on the region proposals [2-3] (Fig. 1). Region-Based Convolutional Neural Network (R-CNN) [4], Fast R-CNN [5], and Faster R-CNN [6] are the most well-known ones among two-stage detectors.

Peer-review under responsibility of the Scientific Committee of the NAMRI/SME.

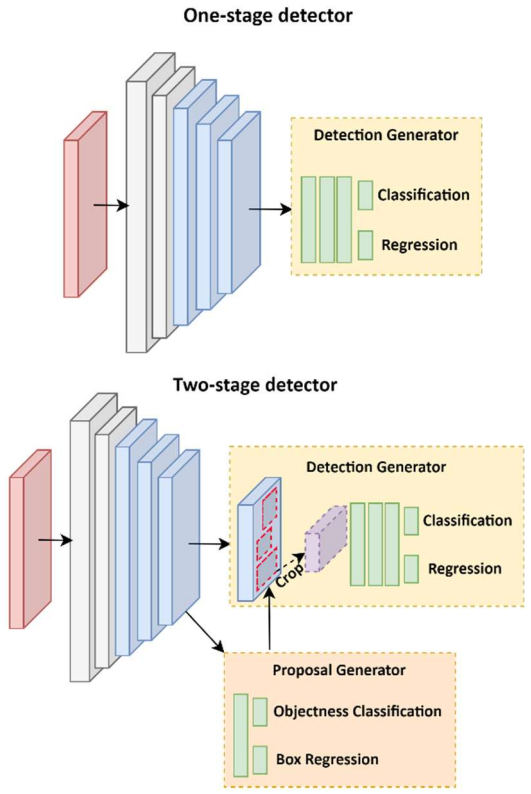

Fig. 1. Architectures of one-stage and two-stage detectors [7]

These detectors provide higher localization and object recognition accuracy at the cost of adding complexity and time required for generating region proposals. YOLO is a one-stage object detector that enables the classification and localization of objects in a single step. This regression-based object detector, which was first proposed in 2016, only contained convolutional layers and achieved the detection speed of 45 frames per second, making it an ideal candidate for real-time purposes [8]. Anchor boxes, batch normalization, k-means clustering, and high-resolution detector were introduced in YOLOv2 to improve the accuracy of previous model [9]. YOLOv3 was the first model among YOLO series that used residual networks in its architecture [10]. In YOLOv4, Cross Stage Partial Networks (CSP), Spatial Pyramid Pooling (SPP) and Path Aggregation Network (PANet) were introduced [11]. On the basis of YOLOv3 and YOLOv4, YOLOv5 incorporated a special Focus and CSP module to further enhance and integrate image features and achieve a high level of speed and accuracy [12].

In this paper, the YOLOv5 object detection algorithm has been optimized by adjusting hyperparameters and taking advantage of stochastic gradient descent optimizer (SGD) to be used for detecting various lightweight industrial tools. Real-time tool detection could be further applied to Human-Robot Collaboration (HRC), CPS, and AR within the Industry 4.0 framework. The application, implementation and fine-tuning of the algorithm, data acquisition and annotation have been the primary focuses of this work. The remainder of this paper is organized as follows: Section 2 explains the optimized YOLOv5 algorithm, and the dataset is presented in the Section 3. The results are discussed in Section 4, followed by a conclusion in Section 5.

2. The Proposed Method

In this study, we fine-tuned and optimized YOLOv5 parameters to detect different tools in real-time. As a unified object detection model, YOLOv5 generates the bounding box coordinates and matching class probability for objects within an image. It works by dividing the input image into a grid; if the center of an object falls within a grid cell, that grid cell is in charge of detecting the object by predicting multiple bounding boxes, each of which consisting of the center coordinates (x, y) and dimensions (width and height) of the bounding box, as well as the confidence score (Fig. 2).

The architecture of YOLOv5 is primarily composed of three modules: backbone, neck, and head. The backbone module, which consists of CNN layers, extracts the main features from the input image. This is done using focus structure, CSP, and Spatial Pyramid Pooling (SPP). The focus block consists of four parallel slice layers to interlace an input image of size 3*640*640 into four 3*320*320 images. Using the concatenate module, images are spliced from depth to form 320*320*12 to be further processed in CBL module consisting of convolutional layers, batch normalization and leaky ReLu activation function (Fig.3). By reconstructing a low-resolution image from a given high-resolution input, the focus module allows the spatial information of the input image to be sent to the channel dimension without any of the details lost in the process. The main objective of this module is to speed up model execution by reducing parameters, the number of calculations, and the memory space required by the GPU [13].

CSP module is a neural network architecture that employs a cross-stage feature fusion technique. In this technique, input features are split into two groups: one group goes through a CBL module followed by a convolutional layer, while the other group is only processed by a single convolutional layer. These two groups are then combined for subsequent processing. This approach enables CSP to leverage both high-level and low-level features, resulting in richer combination of features and improved performance in terms of detection speed and accuracy [14-15].

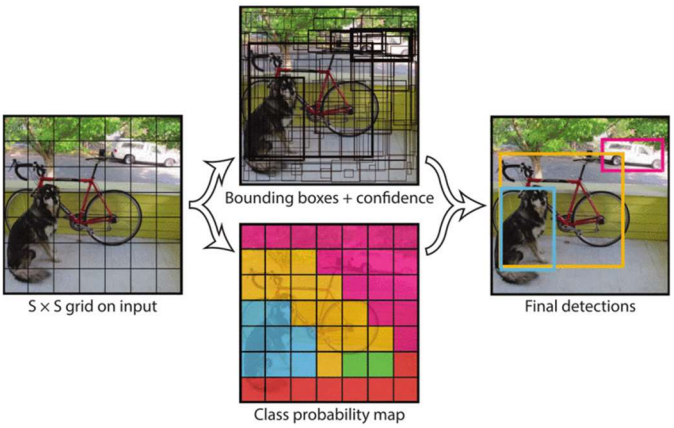

Fig. 2. Basic concept of YOLO [8]

Deep residual networks (ResNet) used in CSP1-X module, includes two CBL units and a skip connection which is a shortcut that allows the gradient to flow directly to earlier layers in the network, helping to prevent the vanishing gradient problem and enabling the training of deeper neural networks with improved accuracy. The SPP block mainly consists of max pool layers for multi-scale feature fusion.

The primary function of the neck is to aggregate features and build feature pyramids using a Path Aggregation Network (PANet), which improves the ability of the model to detect the same objects with different scales and sizes. The head module performs the final detection part and generates bounding boxes that indicate the category, coordinates, and confidence rates based on the multi-scale feature maps from the neck module [16-19] (Fig. 3).

YOLOv5 is available in four different sizes (small: YOLOv5s, medium: YOLOv5m, large: YOLOv5l, and x-large: YOLOv5x), all of which share the same basic structure other than the difference in depth and width, which determine the depth and the number of convolution cores in the backbone module of the model. Intuitively, a larger network with more tuning options should perform better, but, on the flip side, it increases both training and inference times.

In order to achieve high accuracy and efficiency with deep learning models like YOLOv5, it is crucial to tune the hyperparameters such as learning rate, momentum and weight decay correctly. Setting the learning rate that determines the step size of the optimization algorithm to a high value can cause the loss to oscillate or diverge, while a low learning rate can cause the training to be slow or get stuck in local minima. The linear learning rate scheduler with the formula provided in equation (1) is used to adjust the learning rate over time. Thus, a high learning rate in the initial steps of training increases the convergence speed and by gradually reducing the learning rate to a lower value a better accuracy and performance will be achieved.

$$l_r = (1 - \frac{current_{epoch}}{Max_{epoch}}\ (1 + l_f) \qquad (1)$$

Momentum parameter has also been employed to escape shallow local minima and increase convergence speed. This process works by adding a fraction of the previous steps to the current update, which helps to reduce the amount of oscillation and smooth out the convergence. In essence, it allows the optimization algorithm to remember the direction it was moving in the previous iterations and continue in that direction with a certain amount of force. This parameter should be chosen carefully as a high momentum cause the algorithm to overshoot the minimum and oscillate around it. Additionally, weight decay as a regularization technique that helps preventing overfitting and enhancing generalization is used. This is done by adding a penalty term to the loss function which is proportional to L2 norm of the model's weights multiplied by a hyperparameter.

Finally, SGD optimizer has been used to update the model parameters based on a subset of the training data, rather than the full dataset. This way, instead of computing the gradient of the entire training dataset at once, small batch is used to update the model parameters. This optimizer allows for faster and more efficient training especially on large datasets. The procedure of this work is shown in Fig. 4.

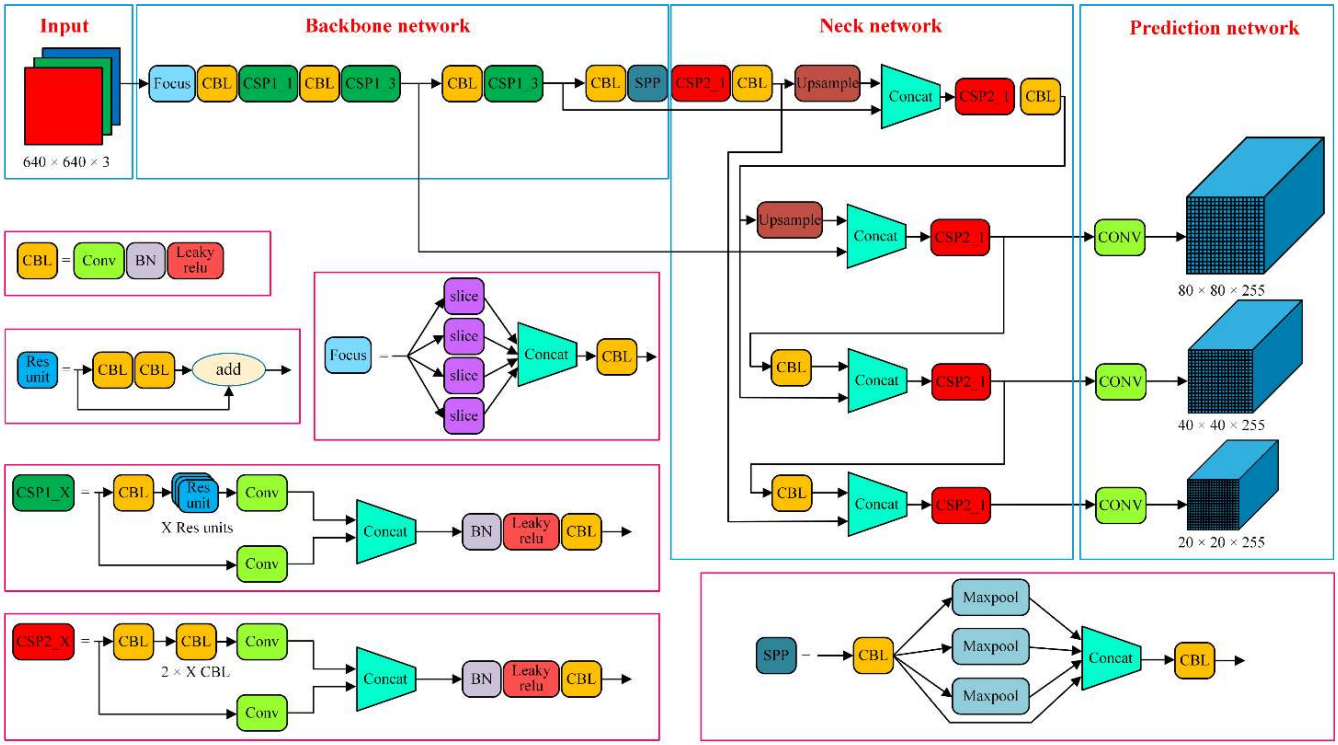

Fig. 3. Architecture of YOLOv5 [20]

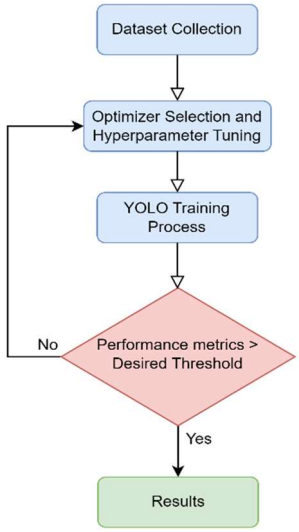

Fig. 4. Procedure of the proposed method

4. Data Collection and Labeling

Object detection requires a large number of annotated images for training and evaluation. For the purpose of this research, 3,286 RGB images consisting of 17 classes of industrial tools have been collected from the internet. The collected dataset consists of adjustable wrench, double open-end wrench, single open-end wrench, double box-end wrench, combination wrench, pipe wrench, pliers wrench, Phillips screwdriver, flat-head screwdriver, pliers, tape measure, level, hammer, mallet, screw, washer, and nuts (Fig. 5).

CNN algorithm is known to be translation invariant as it is endowed with pooling layers. However, it does not have inherent invariance against changes in size or orientation. Therefore, this issue needs to be resolved throughout the training phase. For this reason, numerous variants of tools have been included in the data collection process. This allows the model to be exposed to and learn from a wider range of examples. We considered objects with different settings and backgrounds, different positions, orientations, and scales. Moreover, using augmentation techniques (random horizontal and vertical flips and 90-degree clockwise and counterclockwise rotations) each image obtained a maximum of three augmented variants to become rotation invariant.

Data annotation, also known as labeling, is an essential part of object detection, and it directly impacts the model's ability to learn and improve performance. The Roboflow platform [21] was utilized to label each image in the dataset by drawing a bounding box and assigning a corresponding label. Labels have been generated individually for each image in the text format, consisting of five numbers for each object in the image, with the first number being the class of object, followed by the x and y position of the center of the bounding box, width, and height of the bounding box. On average, 1.9 annotations have been made for each image, and the number of objects per image varied between 0-30, with the single-object image being the most frequent. The total number of objects per class, before applying augmentation, is shown in Fig. 6.

Fig. 5. Sample images in the dataset

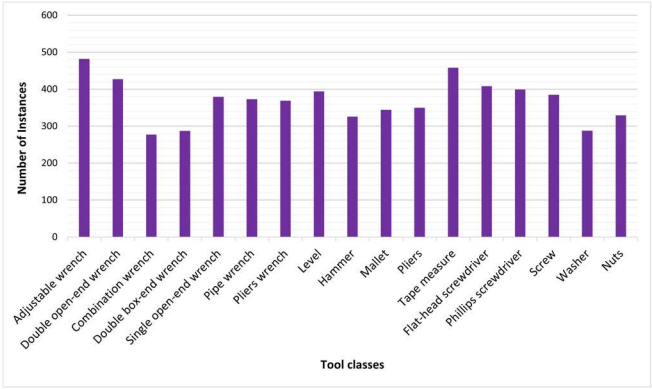

Fig. 6. Class-specific annotations

5. Results and Discussion

The experiment has been performed on a computer equipped with an Intel® Xeon® processor and an NVIDIA RTX A5000 graphics processor (24 GB memory). Network learning rate fine-tuned and adjusted to 0.001 and the SGD method has been adopted to optimize the training process. The weight decay and momentum parameter were optimized and set to 0.0006 and 0.9, respectively. The dataset has been randomly divided into training, test, and validation sets with the ratio of 7:1.5:1.5, and the maximum number of iterations

has been chosen to be 200. The performance of the model is mainly evaluated, using the precision (P) and recall (R) metrics for each class provided in equation (2-3):

$$P = \frac{TP}{TP + FP} \tag{2}$$

$$R = \frac{TP}{TP + FN} \tag{3}$$

where TP (true positive) is the number of tools that are successfully detected, FP (false positive) denotes the number of tools that are incorrectly detected as other categories of tools, and FN (false negative) specifies tools that are not detected. For instance, considering precision and recall for the class of tape measure, TP is the actual tape measures that the model correctly detected, and FP denotes other objects, such as level, wrench, etc., that were mistakenly detected as tape measures and FN denotes tape measures that the model was not able to detect as tape measures. In brief, precision indicates how well a model can make predictions within a given classification, and recall refers to the number of times a model successfully detected a specific class [22].

Precision and recall cannot individually evaluate the entire performance of the model. Hence, average precision (AP), the area under the P-R curve, is employed to summarize the PR Curve to one scalar value. Average precision calculated by equation (4) has a high value when both recall and precision values are high, and when either precision or recall drops in value, AP becomes low.

$$AP = \int_0^1 P(R)dR \tag{4}$$

Average precision is calculated on a class-by-class basis and considers each class individually. In order to evaluate the performance of the model for all classes using one unified metric, mean average precision (mAP) is utilized and can be calculated using the equation (5):

$$mAP = \frac{1}{|n_c|}\sum_{i=1}^{n_c} AP(i) \tag{5}$$

where n_c is the number of classes. Fig. 7 depicts mAP for each epoch with the final mean average precision of 98.308%.

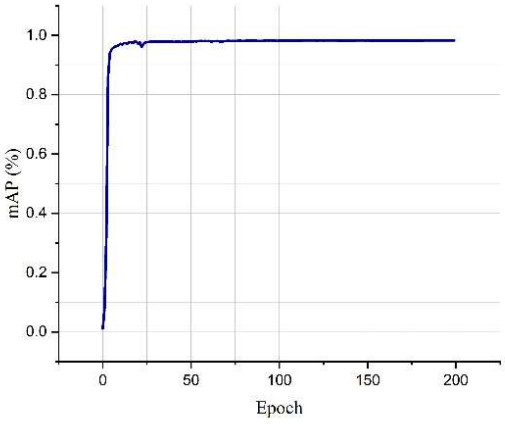

Fig. 7. mAP results for each epoch

Precision and recall metrics are shown in Fig. 8 and Fig. 9, respectively. The maximum recall value is 96.44%, and the maximum precision value is 98.07%. As both of these values are high enough, it can be inferred that most tools are accurately detected by the model. The precision-recall curve shown in Fig. 10 indicates that the model maintains high precision while achieving high recall for all classes.

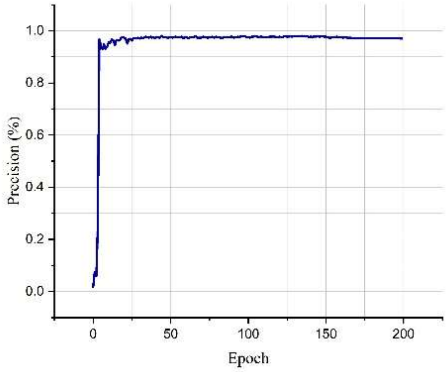

Fig. 8. Precision results for each epoch

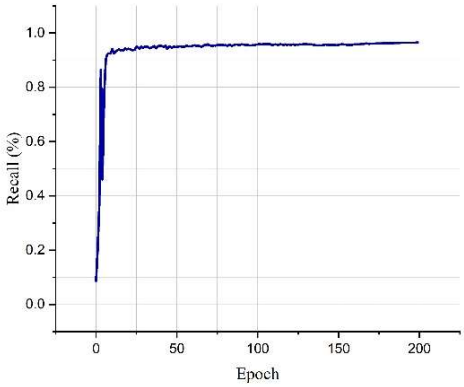

Fig. 9. Recall results for each epoch

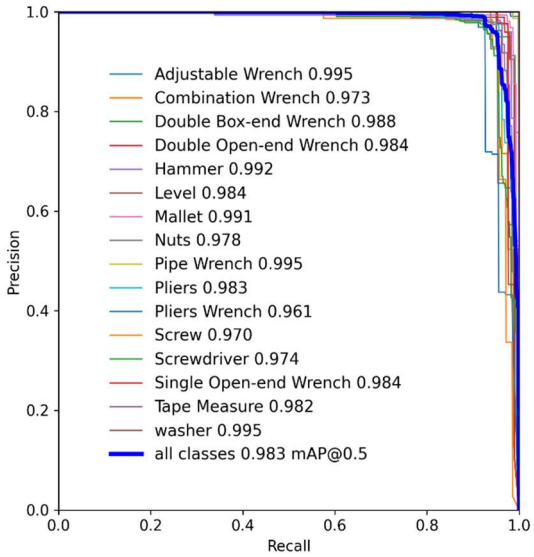

Fig. 10. Precision-recall curve for each class

The confusion matrix is shown in Fig. 11, in which each column indicates the actual category, and each row indicates the predicted class for each object. By analyzing the confusion matrix, it is observed that the model performed well in detecting adjustable wrench, pipe wrench, and washer.

However, it performed the worst in detecting pliers wrench with the accuracy of 90%. About 3% of actual pliers wrench and 2% of combination wrench were misclassified as pliers and double open-end wrench, respectively.

The model localizes each object with a bounding box and classifies it as a tool type with a confidence rate. Fig. 12 illustrates an example of the model's prediction on one sample image, with a confidence rate above 0.9, indicating a high level of confidence in detecting tools. While the model demonstrated proficiency in identifying different tool categories, it is essential to examine the factors that could impact the accuracy of its detecting results, especially in real-world scenarios.

One such factor is the quality and representativeness of the collected dataset. To ensure the model's robustness and accuracy, a high-quality dataset with diverse tool types, backgrounds, scales, and orientations should be used. Moreover, the model's hyperparameters, such as the learning rate, momentum and weight decay can also impact the accuracy. Careful tuning of these hyperparameters can optimize the model's performance and improve its detecting results.

Furthermore, the performance of the trained model in real-world scenarios may be impacted by factors such as tool variations (in terms of size, shape, color, and other visual characteristics) and environmental conditions (lighting and background), which are not fully represented in the training dataset. Hence, despite attempts to create a comprehensive dataset for this study, the model's accuracy could still be influenced by the unpredictability of real-world conditions in which the tools are utilized.

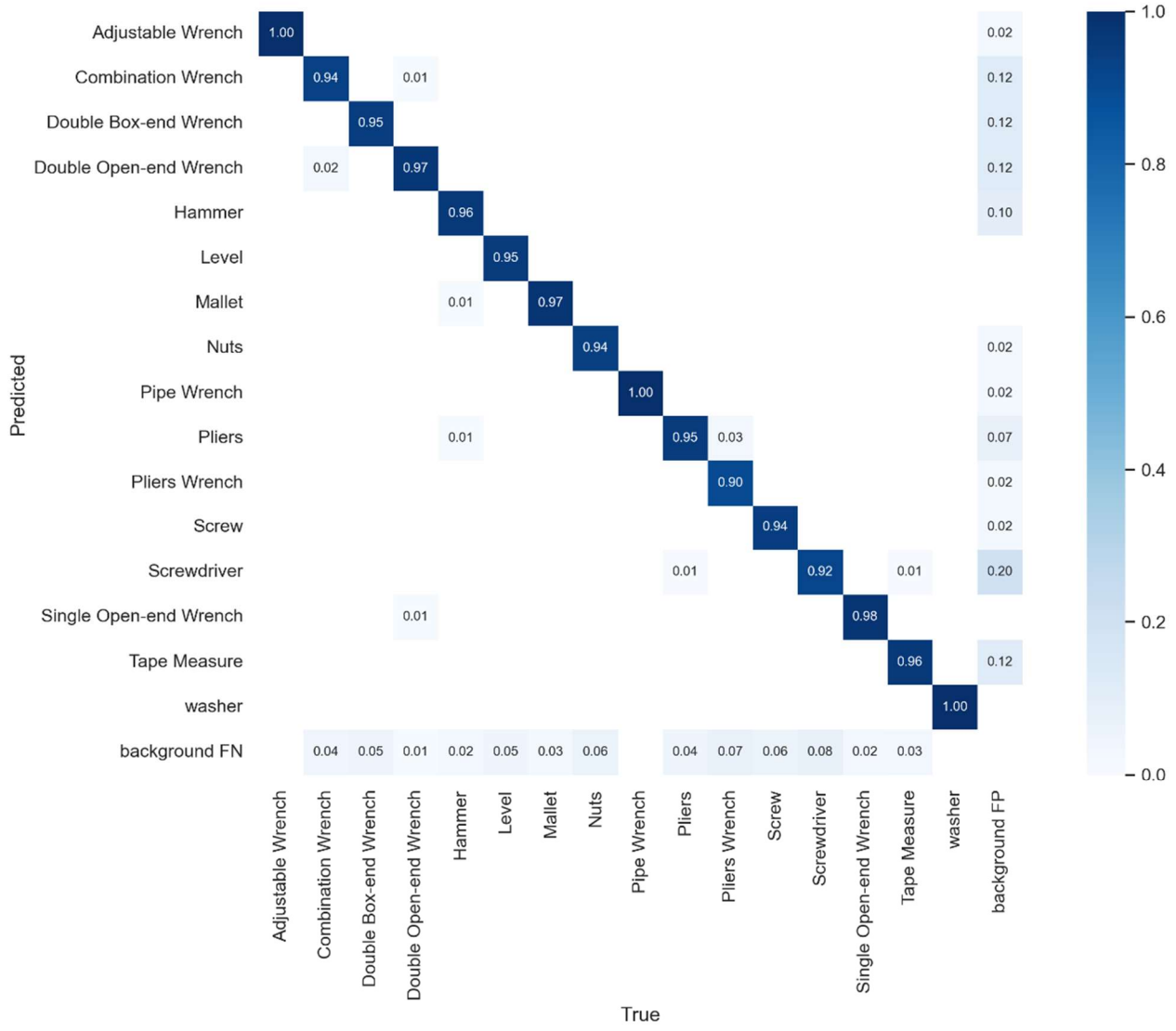

Fig. 11. Confusion matrix

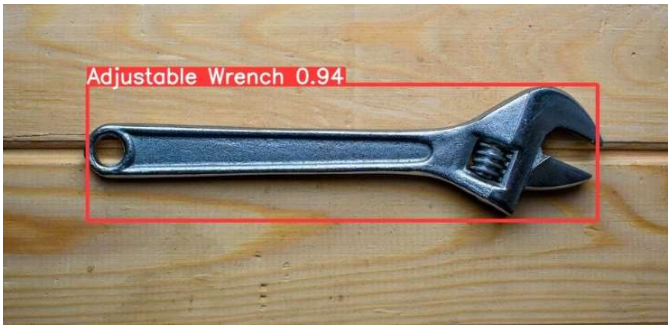

Fig. 12. Tool detection performance

6. Conclusion and Future Work

In this study, we aimed to enhance the performance of the YOLOv5 real-time object detector model to be used as a real-time tool detector for smart manufactories. This has been done by configuring the hyperparameters such as learning rate and momentum and utilizing SGD optimizer. After collecting 3,286 images of 17 classes of lightweight industrial tools with different scales, orientations and backgrounds, each image has been labeled according to the YOLOv5 standard and received up to three augmented variances (horizontal/vertical flips and 90-degrees rotation) to enhance rotation invariance. Finally, the YOLOv5x model has been trained with optimized hyperparameters to detect and localize different tools. The model's effectiveness has been demonstrated by achieving a mean average precision of 98.3%.

In the future, we will try to improve model performance and generalization by employing regularization techniques. Additionally, by integrating our model with a drone we aim to experiment the performance in more real-world scenarios.

Acknowledgements

This work was partially supported by Kummer Innovation and Entrepreneurship Doctoral Fellowship at Missouri University of Science and Technology and by the NSF grant CMMI-1954548. Any opinion, findings and conclusions are those of the authors and do not necessarily represent the views of the National Science Foundation.

References

[1] T. M. Fernández-Caramés, O. Blanco-Novoa, I. Froiz-Míguez, e P. Fraga-Lamas, "Towards an Autonomous Industry 4.0 Warehouse: A UAV and Blockchain-Based System for Inventory and Traceability Applications in Big Data-Driven Supply Chain Management", *Sensors*, vol. 19, nº 10, 2019, doi: 10.3390/s19102394.

[2] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, e B. Lee, "A survey of modern deep learning based object detection models", *Digit Signal Process*, p. 103514, 2022.

[3] L. Jiao *et al.*, "A survey of deep learning-based object detection", *IEEE access*, vol. 7, p. 128837–128868, 2019.

[4] R. Girshick, J. Donahue, T. Darrell, e J. Malik, "Region-based convolutional networks for accurate object detection and segmentation", *IEEE Trans Pattern Anal Mach Intell*, vol. 38, nº 1, p. 142–158, 2015.

[5] R. Girshick, "Fast R-CNN", em *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, p. 1440–1448. doi: 10.1109/ICCV.2015.169.

[6] S. Ren, K. He, R. Girshick, e J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", *Adv Neural Inf Process Syst*, vol. 28, 2015.

[7] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, e J. García-Gutiérrez, "On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data", *Remote Sens (Basel)*, vol. 13, nº 1, 2021, doi: 10.3390/rs13010089.

[8] J. Redmon, S. Divvala, R. Girshick, e A. Farhadi, "You only look once: Unified, real-time object detection", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, p. 779–788.

[9] J. Redmon e A. Farhadi, "YOLO9000: better, faster, stronger", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, p. 7263–7271.

[10] J. Redmon e A. Farhadi, "Yolov3: An incremental improvement", *arXiv preprint arXiv:1804.02767*, 2018.

[11] A. Bochkovskiy, C.-Y. Wang, e H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection", *arXiv preprint arXiv:2004.10934*, 2020.

[12] G. Jocher *et al.*, "ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations", abr. 2021, doi: 10.5281/ZENODO.4679653.

[13] Y. Yu, J. Zhao, Q. Gong, C. Huang, G. Zheng, e J. Ma, "Real-time underwater maritime object detection in side-scan sonar images based on transformer-YOLOv5", *Remote Sens (Basel)*, vol. 13, nº 18, p. 3555, 2021.

[14] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, e I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN", em *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, p. 390–391.

[15] Y. Jing, Y. Ren, Y. Liu, D. Wang, e L. Yu, "Automatic Extraction of Damaged Houses by Earthquake Based on Improved YOLOv5: A Case Study in Yangbi", *Remote Sens (Basel)*, vol. 14, nº 2, 2022, doi: 10.3390/rs14020382.

[16] J. Zhao *et al.*, "A Wheat Spike Detection Method in UAV Images Based on Improved YOLOv5", *Remote Sens (Basel)*, vol. 13, nº 16, 2021, doi: 10.3390/rs13163095.

[17] J. Yao, J. Qi, J. Zhang, H. Shao, J. Yang, e X. Li, "A Real-Time Detection Algorithm for Kiwifruit Defects Based on YOLOv5", *Electronics (Basel)*, vol. 10, nº

14, 2021, doi: 10.3390/electronics10141711.

[18] H.-K. Jung e G.-S. Choi, "Improved YOLOv5: Efficient Object Detection Using Drone Images under Various Conditions", *Applied Sciences*, vol. 12, nº 14, 2022, doi: 10.3390/app12147255.

[19] Y. Fang, X. Guo, K. Chen, Z. Zhou, e Q. Ye, "Accurate and Automated Detection of Surface Knots on Sawn Timbers Using YOLO-V5 Model.", *Bioresources*, vol. 16, nº 3, 2021.

[20] F. Lei, F. Tang, e S. Li, "Underwater Target Detection Algorithm Based on Improved YOLOv5", *J Mar Sci Eng*, vol. 10, nº 3, 2022, doi: 10.3390/jmse10030310.

[21] B. Dwyer, J. Nelson, "Roboflow (Version 1.0) [Software]". https://roboflow.com. https://roboflow.com., 2022. [Online]. Disponível em: https://roboflow.com. https://roboflow.com.

[22] R. Padilla, S. L. Netto, e E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms", em *2020 international conference on systems, signals and image processing (IWSSIP)*, 2020, p. 237–242.