YOLO-v8 IN CAPTURING IMPERFECTIONS GENERATED BY CHANGING 3D PRINTER PARAMETERS

Ana-Maria TĂLÎNGĂ[1], Anton HADĂR[2,3,4], Marius-Valentin DRĂGOI[5]*, Ionuț NISIPEANU[6], Haider Abdullah ALI[7], Cosmin Petru SUCIU[8]

This paper presents the design and implementation of PLA objects using different parameters of a 3D printer machine, and the detection of imperfections of components surface using a custom YOLO-v8 algorithm. The different parameters used by the 3D printer and environment changes can produce errors on a material surface, which can be classified and revealed by computer vision detection, to avoid flaws in the fabrication process. The implemented Windows application has been trained with thousands of pictures and intensely tested to recognise with a big precision, imperfections on PLA 3D printed objects.

Keywords: 3D printer; Computer vision; Artificial Intelligence (AI); YOLO-v8

1. Introduction

With the promise of a more rapid and efficient production process, 3D printing has existed since 1945 in theory and, in somewhat more limited form, since 1971. Beginning with the introduction of the first stereolithographic (SLA) systems in 1986 [1], the 3D printing processes have expanded throughout the last several decades. According to ASTM (Additive Manufacturing Technology standards) International, there are seven distinct technological steps involved in 3D printing, and several commercial technologies stand in for each of them.

[1] Engineer, Faculty of Industrial Engineering and Robotics, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: ana_maria.talinga@upb.ro
[2] Professor, Faculty of Industrial Engineering and Robotics, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: anton.hadar@upb.ro
[3] Vice-President, Academy of Romanian Scientists, Bucharest, Romania, e-mail: anton.hadar@upb.ro
[4] Corespondent Member, Technical Sciences Academy of Romania, Bucharest, Romania; anton.hadar@upb.ro
[5] Assistant Professor, Faculty of Engineering in Foreign Languages, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: marius.dragoi@upb.ro, marius.valentin.dragoi@gmail.com
[6] Bachelor Student, Faculty of Engineering in Foreign Languages, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: ionut.nisipeanu@stud.fils.upb.ro
[7] PhD Engineer, Ministry of Youth and Sport, Baghdad, Iraq, e-mail: h_haider26@yahoo.com
[8] Scientific Researcher III, National research and development institute for gas turbines COMOTI, Bucharest, Romania, e-mail: cosmin.suciu@comoti.ro

One method of producing components is Additive Manufacturing (AM), sometimes known as 3D printing. This procedure involves creating an item by adding material in cross-sectional layers [2]. AM can build complicated forms and structures while controlling resources, reducing waste and other drawbacks of traditional production, making it more attractive [3].

Fused Desing Modeling (FDM) is an AM technique used by the 3D printer to fabricate components used to train the implemented Windows software applications described in this paper. The typical feedstock material used in FDM is a polymer filament spool, often possessing a common 1.5mm diameter, ranging from 1.5 to 3 mm [4], [5]. The pliable polymer is applied onto a heated surface using a computer numerical control system, which follows a 3D model program and moves it along a 3D axis. Upon exiting the extrusion nozzle, the polymer undergoes a cooling process and then hardens. The process involves the sequential deposition of polymer filament layers until the desired result is completed. The FDM method works well with Acrylonitrile Butadiene Styrene (ABS) and Polylactic Acid (PLA) [6], both polymers being the most used for rapid prototyping.

Any changes from the environment, machine parameters or machine components (if one or more components get destroyed) will affect the printed object, on its surface appearing imperfections. These issues are hard to resolve after the object is 3D printed, most of the time, objects are broken on corrections. The use of Artificial intelligence (AI) in the custom detection algorithm for picture analysis will facilitate improvements in the 3D printing process, with all the errors being collected before printing 3D all the objects.

The goal of this research is to create a Windows software application to recognise defects on 3D printed objects, which are called imperfections related to a PLA object with no defects on its surface.

2. Literature review

Much research has been conducted to asses the impact of 3D printing machine parameters on the qualities of products, the surface quality being a notable constraint associated with the FDM technique [7].

Galantucci et al. [8] explain the influence of the surface quality or dimensional accuracy of FDM products, is made by two sets of parameters: procedure and 3D printing circumstances:

- procedure parameters include several factors, including raster angle, layer air gap, thickness and ideal build direction;
    - additional techniques to enhance the quality of the product include pre-processing, including the slicing procedure, and postprocessing, which involves treatments on the surface of the object.

- 3D printing parameters raster width, feed rate, flow rate and extrusion temperature.

Not only do printing conditions influence FDM products, but also the structure of the material used represents a factor as specified by Kaveh et al. [9].

One artefact that might appear on the surface of 3D FDM printed items is the deposition striae tool mark as Pavlovich has specified [10]. These extra features are added to the impressed and striated ones that are often said to be transmitted to an item during machining using contact and force from the tool's working surface.

Using methods like the Fast Fourier Transform (FFT), Shim et al. [11] found that it was feasible to conduct image analyses of 3D printed items with digital imaging and microscopy to identify surface irregularities and other characteristics that could emerge during the 3D printing process.

AI has become a very useful tool in everyday life for many of the activities we have to perform. Scientific papers from special literature used AI to simulate, investigate or resolve a concrete problem. The specific using of AI to determine imperfections on 3D printed objects represents a new way of using computer vision to reduce industry fabrication fails. Chen and Gu [12] implemented a framework for the next generation of materials with extraordinary characteristics, a framework which is expected to transform methods for designing and optimizing composites.

3. Materials and methods

This section presents the design, 3D printing and analysis of a set of PLA objects. The implementation of a custom YOLO-v8 computer vision algorithm is also presented to recognise imperfections on the printed objects' surface.

3.1 Designing and 3D printing of a set of PLA objects

PLA objects have been designed in IDEA Maker software. The resulting STL files have been imported into the 3D printer Creality Ender-3 S1 Pro to print objects using PLA material. The changing of the printing parameters is done in IDEA Maker software.

Temperature is an important factor in 3D printing to avoid nozzle blockage and serious defects [13].

Fig.1 shows PLA printed objects with no defects on the surface, which parameters from Table 1 have been used.

Fig.1. PLA object 3D printed, with no imperfections on its surface

Table 1

The 3D printer parameters used for the PLA objects with no imperfections on the surface

Infill	Infill type	3D printing head temperature	Print speed	Layer thickness
100%	straight	205°C	50mm/s	0.4mm

3D printer parameters have been changed to obtain PLA objects with imperfections on the surface. The objects with imperfections have had the same shape, infill and infill type as the objects with no imperfections, only peak temperature and print speed being different, as can be seen in Table 2.

Table 2

The 3D printer parameters used for the PLA objects with defects on the surface

PLA object with defects	Infill	Infill type	3D printing peak temperature	Print speed	Layer thickness
1	100%	straight	220°C	70mm/s	0.4mm
2	100%	straight	220°C	40mm/s	0.4mm
3	100%	straight	190°C	50mm/s	0.4mm
4	100%	straight	variable	variable	0.4mm

Fig.2 shows the fault induced by high temperature and speed. The peak printing temperature and speed are 15°C and 20mm/s higher than for the PLA object with no defects. The layers overlapped, as seen in Fig.A2's highlight box 1. Due to extreme heating and rapid speed, the filament fails to solidify until the next layer is placed.

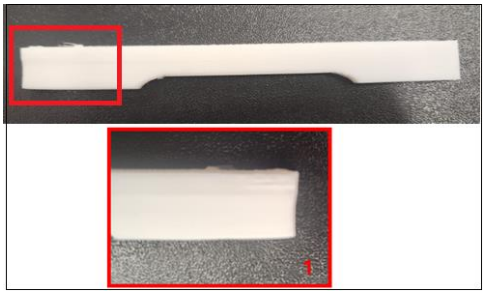

Fig.2. First PLA object with defects on the surface

Fig.3 shows the fault generated by a high temperature and delayed layer deposition. Like before, the printhead temperature is 15°C higher, however, the print speed is 10mm/s lower than required. Due to the slow print speed, the print head continually stayed in one location, scorching the preceding layer. The package marked 2 has this fault.

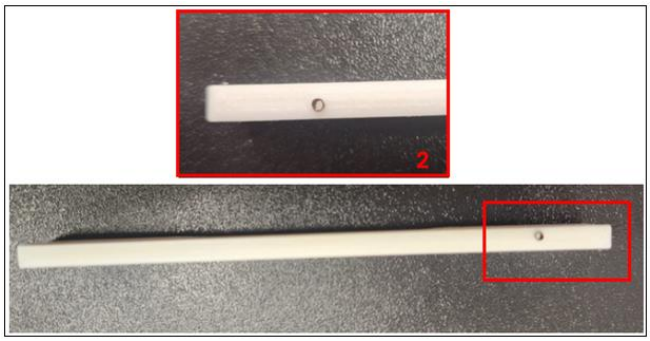

Fig.3. Second PLA object with defects on the surface

In Fig.4, imperfections from a low temperature are seen. The print head temperature is 15°C lower. This caused layer peeling or gaps. The filament's low temperature causes layers to separate or leave gaps. In the index 3 border, the layers have not attached, resulting in uneven filament strands. These threads were attached to the print head and collected as shown. The filament has not achieved the melting temperature needed to make a homogeneous layer, causing gaps in the index 4 border.

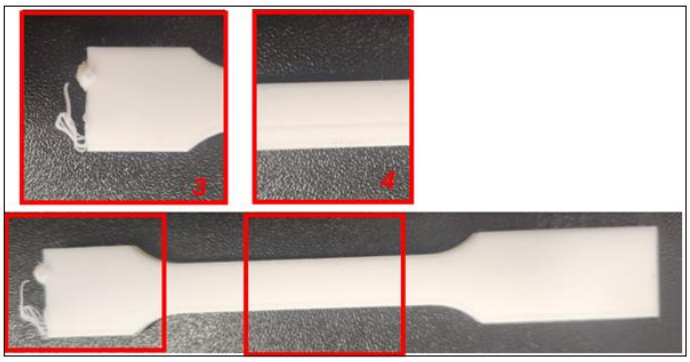

Fig.4. Third PLA object with defects on the surface

The fault in all sections is shown in Fig.5, which correspond to the forth record in Table 2 (with variable 3D printing peak temperature and print speed). Detail-oriented component design is impossible due to the layer's coarse thickness. The layer thickness must be as thin as possible to get the details on the pieces, but this approach might increase printing time. This error on the object surface can be seen for all four defects mentioned before.

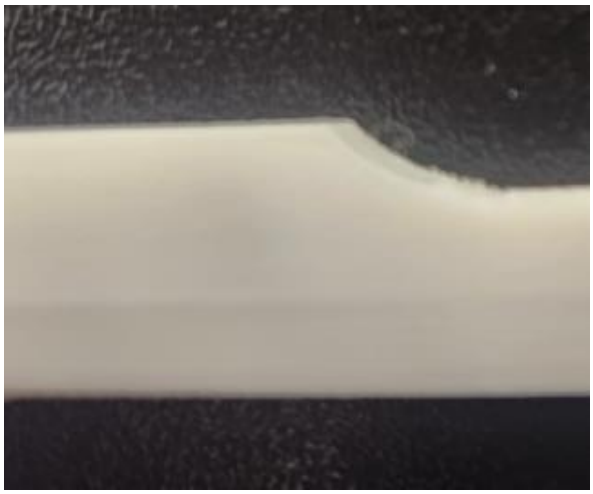

Fig.5. Fourth PLA object with defects on the surface

3.2 YOLO evolution

Object Detection (OD) is a computer vision problem that involves the identification and localisation of items belonging to certain pre-defined classes in input pictures [14].

Computer vision comprises subfields that include:

- image classification - the process by which a machine learning algorithm is trained to identify and classify objects or entities in digital images [15];
- object detection - the process by which a system can identify and locate objects in an image or video section [16];
- object segmentation - an advanced process that involves dividing an image into multiple segments or regions to identify and isolate different objects in an image [17].

Convolutional Neural Networks (CNNs) consist of using a combination of these three domains [18]. CNNs are widely recognised as the standard approach for handling image data, and, in contrast to traditional image processing and artificial detection approaches, use numerous convolutional layers together with pooling structures to uncover profound semantic elements that are concealed within the image pixels [19].

Without numerous phases or area recommendations, Single-Shot object Detection (SSD) may identify objects in an image or video in one pass. The class labels and bounding boxes of objects in an image or video are directly predicted by a single CNN in one-shot object detectors like YOLO (You Only Look Once) and SSD. These models are trained end-to-end utilising a huge dataset of annotated pictures and object-bounding boxes. Various techniques and models, including R-

CNN, Faster R-CNN, YOLO, and SSD, have been created specifically for object identification. These algorithms and models are used in many applications, including object monitoring, security systems and autonomous vehicles [20].

At the beginning of 2023, Ultralytics confirmed the latest member of the YOLO family, YOLO-v8 [21]. Despite an upcoming print release and ongoing development to add more capabilities to the YOLO-v8 repository, first comparisons show that the new YOLO is superior to its predecessors and represents cutting-edge technology in YOLO algorithm development (see Fig.6).

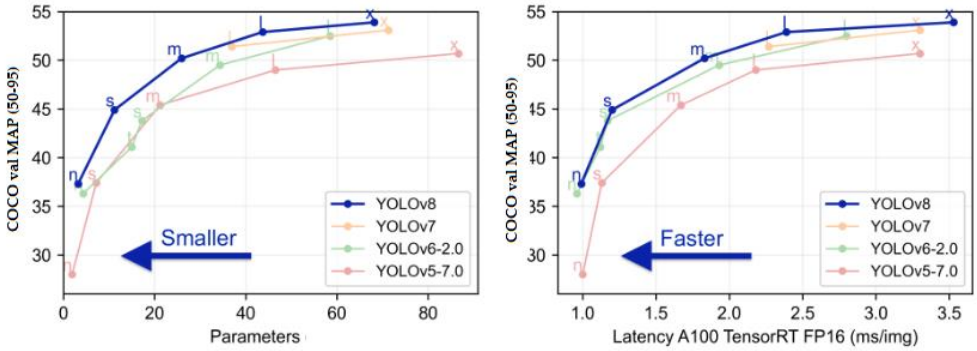

Fig 6. Comparison of YOLO-v8 with its predecessors

Fig.6 demonstrates that all versions of YOLO-v8 have a greater throughput compared to YOLO-v5 and YOLO-v6, even though they have the same number of parameters and are trained on 640 image resolution. This indicates that YOLO-v8 has made hardware-efficient architecture modifications. Ultralytics introduced two versions of their object detection model, YOLO-v8 and YOLO-v5. YOLO-v5 demonstrated remarkable real-time capabilities. According to initial testing findings, YOLO-v8 is anticipated to prioritize deployment on limited edge devices with a high-speed inference capability.

YOLO works in this way [22]:

- Grid division
  - Partition the supplied picture into a grid with size SxS;
  - Each grid cell is tasked with recognising items that have their centres located inside that specific cell.
- Generation of multiple predictions
  - Each grid cell generates B bounding box predictions and corresponding confidence ratings. A bounding box is determined by the coordinates of its centre (x, y) to the grid cell;
  - The dimensions of the box, namely its width and height;
  - The confidence score is the probability that the box includes an item, and it is an indicator of accuracy.
- Classification of objects

- The model generates class scores for all feature classes for each bounding box;
- The class score is added to the confidence score to get the total detected object score.

- Using of the technique Non-Maximum Suppression (NMS)
  - Once all bounding boxes are anticipated, the NMS method is used to eliminate overlaps and retains just the most probable detections;
  - In this phase, the boxes are sorted based on their trust scores;
  - Removal of boxes that exhibit significant overlap - measured by Intersection over Union (IoU), with boxes that possess higher scores.

CNNs used by YOLO, are responsible for extracting distinctive characteristics from the picture. The original YOLO architecture consists of 24 convolutional layers that are used for extracting features. The final predictions are generated by using two completely linked layers.

Overall, YOLO-v8 consistently enhances the effectiveness and productivity of object identification via the integration of new developments in neural networks and detection algorithms.

3.2 Using the Roboflow platform

For this study, the dataset preparation was conducted using the Roboflow platform, which allows image annotation through a browser using simple shapes and polygons (see Fig.7).

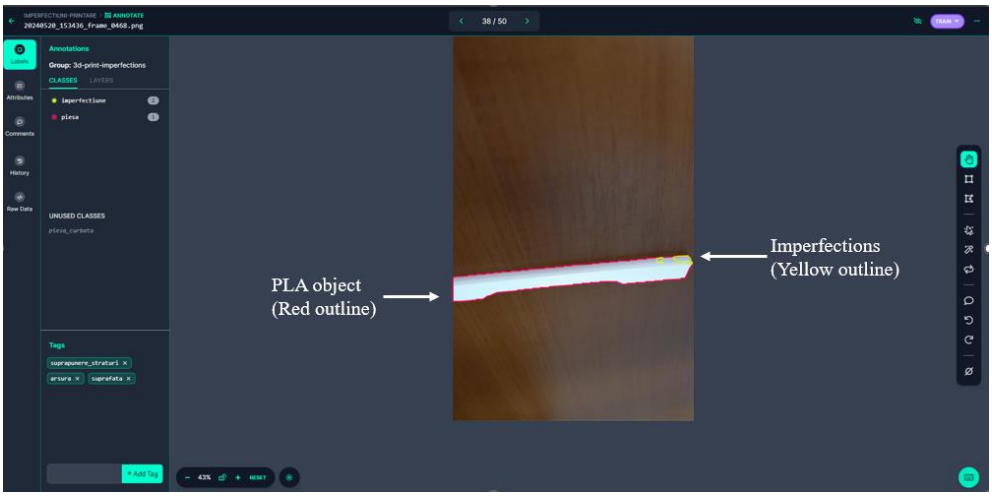

Fig.7. PLA object picture annotated in the Roboflow platform

Initially, more than 900 images were selected and manually annotated. These images were then subjected to two preprocessing variations, specifically the

application of blur and skew effects. Total number of images have been increased to 1900 by using augmentation methods available on the platform Roboflow like: flip, random rotation, random shear, random noise. These images were specifically divided for different stages of the machine learning process: 15% of the images were allocated for testing, 10% for validation, and 75% for training.

3.3 Train YOLO-V8 AI model

The training was conducted on a Windows operating system using Python with the PyTorch and Ultralytics libraries on a computer equipped with an NVIDIA RTX 4070 graphics card, which supports hardware acceleration for AI tasks. The choice of Windows was driven by its widespread compatibility and user-friendly interface, which facilitate the management of various software dependencies and tools. The NVIDIA RTX 4070 graphics card was selected due to its high performance and capability to handle intensive AI computations, providing a significant speedup in training deep learning models through GPU acceleration.

Python and Jupyter Notebooks were used for their versatility and interactive environment. Python is a popular programming language used substantially in the fields of machine learning and data science, offering a rich ecosystem of libraries and tools. Jupyter Notebooks, on the other hand, allow for step-by-step code execution and visualisation of results, aiding in better understanding and debugging during the development process. This setup was crucial for ensuring an efficient and manageable training workflow.

The installation of the PyTorch and Ultralytics libraries was a critical step. PyTorch is a powerful deep-learning framework that offers flexibility and efficiency in building and training neural networks. The Ultralytics library, specifically designed for YOLO models, simplifies the implementation and training of object detection models. These libraries provided the necessary tools and functions to carry out our training process effectively.

Data preparation involved importing the dataset, including images and corresponding annotation files, into the environment. This step ensured that the data was accessible and correctly formatted for the training process. Proper data preparation is essential for the model to learn effectively from the input images, as it directly impacts the quality and performance of the trained model.

The YOLOv8-seg model was selected, and pre-trained for image segmentation detection, due to its suitability for tasks involving the identification of fine details and boundaries within images. This made it ideal for detecting imperfections in printing processes. Using a pre-trained model allowed us to leverage previously learned features, thereby reducing the amount of training time and data required. The training was set up to run for 10 epochs, where each epoch corresponds to a full iteration over the whole training dataset. The iterative method allowed the model to continually modify its weights, hence improving its

performance on the given task. The number of epochs was chosen to balance between adequate training time and computational resources, ensuring the model learned effectively without overfitting.

Executing the training process, which lasted 4 hours, involved the model analysing and learning from the training images. During this period, the model's parameters were continuously updated based on the input data, optimising its ability to detect imperfections. The use of the RTX 4070 GPU significantly reduced the training time by parallelising computations, which is particularly beneficial when dealing with large datasets and complex models.

4. Results

In Fig.8, a sequence of graphs pertaining to the trained model, namely a segmentation model, can be shown. These graphs may be categorized as:

- Loss graphs consist of:
    - Train/Box loss, Train/Seg Loss, Train/Cls Loss, and Train/Dfl loss. These graphs provide insights into the training phase, where decreasing values indicate improved learning over epochs.
    - Validation loss graphs, namely val/box loss, val/seg loss, Val/Cls Loss, and Val/Dfl loss. These graphs represent losses during validation and help understand the model's performance on new data. The model enhances the positive aspect of the declining trend.
- Performance metrics consist of:
    - The metrics Precision(B) and Recall(B) display the precision and recall values for the detection across the training epochs. Precision evaluates the accuracy of positive predictions, while recall reflects the model's ability to recognize relevant instances.
    - The metrics mAP50(B) and mAP50-95(B) represent the average precisions at the 50% IoU threshold and from 50% to 95%, respectively. These metrics provide an indication of the model's accuracy in detecting flaws of different degrees.
    - The metrics accuracy (M) and Recall (M) guarantee the accuracy and recall of the mask, using the same interpretation as bounding box precision and recall.
    - Metrics/mAP50(M) and Metrics/mAP50-95(M) are comparable to bounding box mAP. At a certain threshold, they provide the mean average accuracy for mask detection.

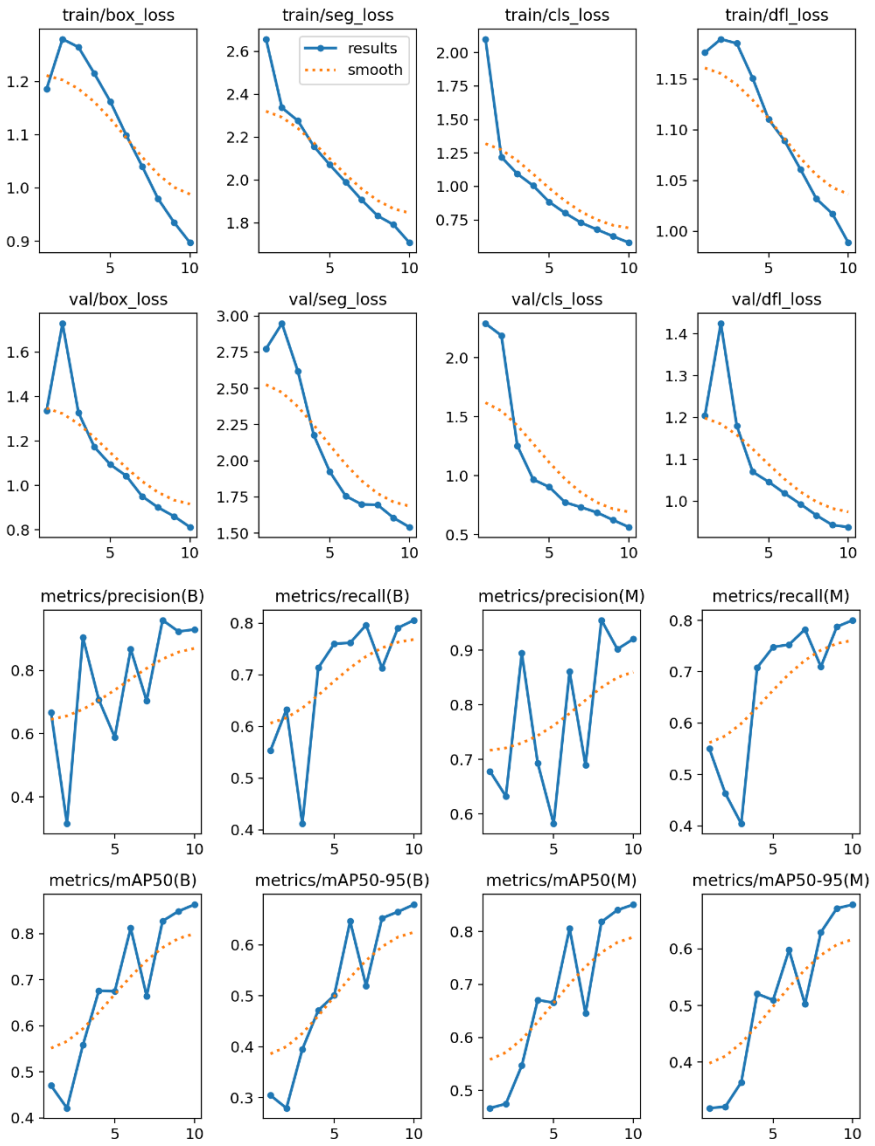

Fig.8. Graphs tracking various training and validation metrics over 10 epochs

Fig.9 presents a confusion matrix, which is a valuable tool for evaluating the classification performance of the model. This matrix compares the actual labels with the predicted ones across different categories: „imperfectiune” (imperfection), „piesa” (piece), „piesa curbata” (curved piece), and background. The diagonal of the matrix has high values, indicating a substantial number of accurate predictions for each class, namely 77 for imperfection and 111 for piece. These results show that the model performs well in these categories.

The off-diagonal values reveal misclassifications, like 78 instances of imperfection being classified as background, highlighting areas where the model needs improvement. The confusion matrix helps identify specific classes where the model excels and others where it struggles. While the model shows strong performance in detecting imperfections and pieces, it indicates potential areas for refinement in distinguishing between curved pieces and background.

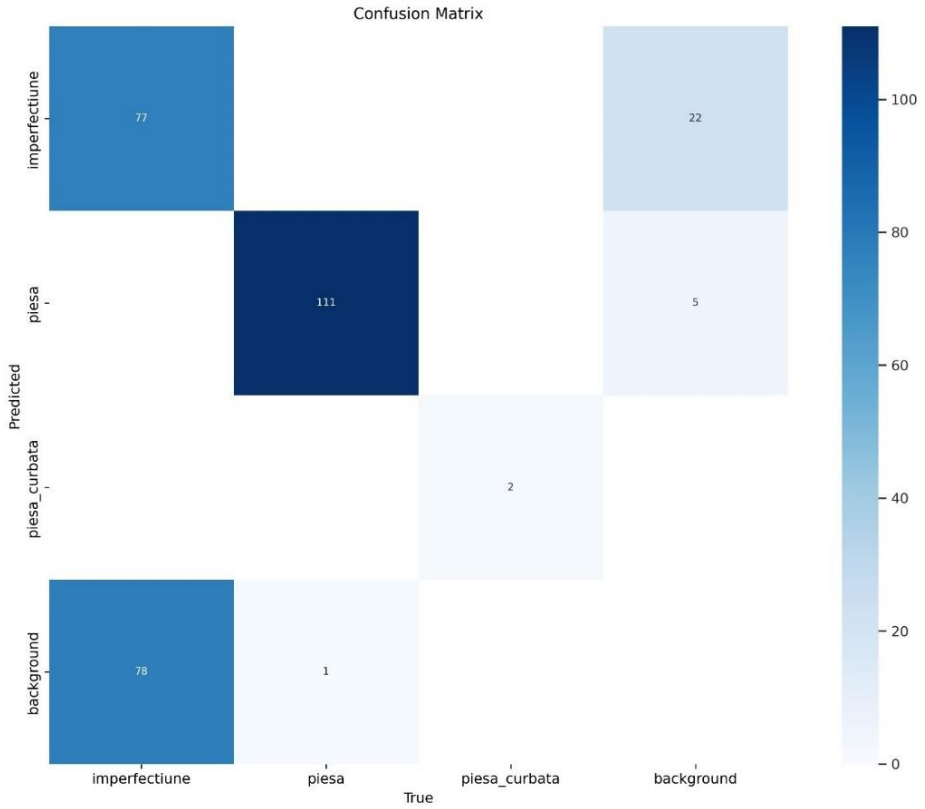

Fig.9. Confusion Matrix

Fig.10 displays an input picture with a highlighted defect, along with a JSON response that provides the imperfection's location and the confidence level of the detection.

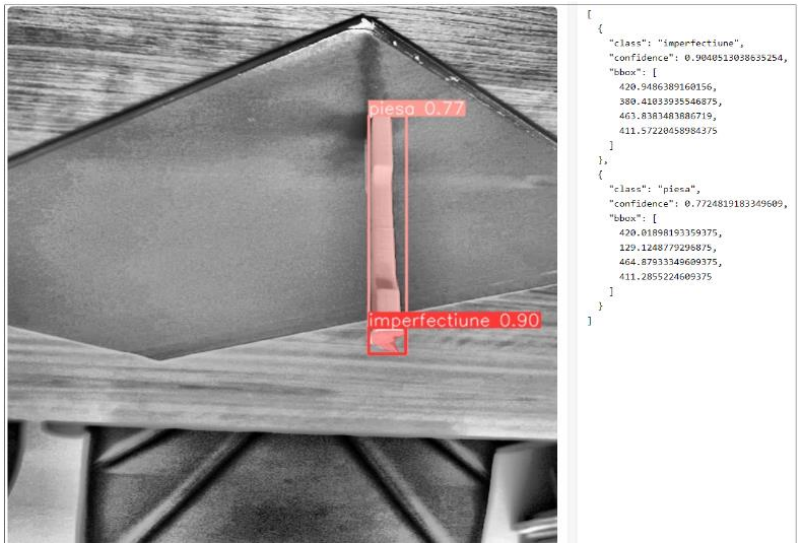

Fig. 10. Result of PLA object image analize by YOLO-v8 AI segmentation model

5. Conclusion

This paper presents a method to detect imperfections on the surface of 3D printed PLA objects. Defects on objects' surfaces appear by changing the parameters of the 3D printer to depart away from the recommended settings of the 3D printer for PLA material. To detect imperfections on the surface of the resulting objects, the YOLO-v8 segmentation AI model has been trained and successfully used to recognise objects' defects. More than 900 images, with annotations (of the piece, imperfections, curved piece and background for each image) have been used in AI model training. The calculated success rate from the resulting graphs is 92.912%.

Because temperature represents an important factor in 3D printing, defects on the surface of the printed objects can appear even if parameters on the 3D printer are not changed manually. If the environment where the 3D printer is located, has a changing temperature, defects can appear on the surface of the objects. The presented implemented application from this paper can recognise all the defects by using a YOLO-v8 AI segmentation model. The AI model must be trained as much as possible to have a success rate of error detection.

R E F E R E N C E S

[1] C. Balletti, M. Ballarin, and F. Guerra, “3D printing: State of the art and future perspectives,” J. Cult. Herit., vol. 26, pp. 172–182, 2017.

[2] W. Gao et al., “The status, challenges, and future of additive manufacturing in engineering,” Comput. Des., vol. 69, pp. 65–89, 2015.

[3] T. D. Ngo, A. Kashani, G. Imbalzano, K. T. Q. Nguyen, and D. Hui, “Additive manufacturing (3D printing): A review of materials, methods, applications and challenges,” Compos. Part

B Eng., vol. 143, pp. 172–196, 2018.
[4] I. Simion and A. F. Arion, "Dimensioning rules for 3D printed parts using additive technologies (FDM)," UPB Sci. Bull. Ser. D Mech. Eng., vol. 78, no. 2, pp. 79–92, 2016.
[5] A. Chouksey, "Study of parametric optimization of fused deposition modelling process using response surface methodology." 2012.
[6] B. N. Turner, R. Strong, and S. A. Gold, "A review of melt extrusion additive manufacturing processes: I. Process design and modeling," Rapid Prototyp. J., vol. 20, no. 3, pp. 192–204, 2014.
[7] D. A. Porter, T. V. T. Hoang, and T. A. Berfield, "Effects of in-situ poling and process parameters on fused filament fabrication printed PVDF sheet mechanical and electrical properties," Addit. Manuf., vol. 13, pp. 81–92, 2017.
[8] L. M. Galantucci, F. Lavecchia, and G. Percoco, "Experimental study aiming to enhance the surface finish of fused deposition modeled parts," CIRP Ann., vol. 58, no. 1, pp. 189–192, 2009.
[9] M. Kaveh, M. Badrossamay, E. Foroozmehr, and A. H. Etefagh, "Optimization of the printing parameters affecting dimensional accuracy and internal cavity for HIPS material used in fused deposition modeling processes," J. Mater. Process. Technol., vol. 226, pp. 280–286, 2015.
[10] S. Pavlovich, "Prototypal forensic intelligence methodologies for the examination of illicit firearms." Murdoch University, 2021.
[11] G. Shim et al., "Elastic Resistance and Shoulder Movement Patterns: An Analysis of Reaching Tasks Based on Proprioception," Bioengineering, vol. 11, no. 1, p. 1, 2023.
[12] C.-T. Chen and G. X. Gu, "Machine learning for composite materials," MRs Commun., vol. 9, no. 2, pp. 556–566, 2019.
[13] N. Lokesh, B. A. Praveena, J. S. Reddy, V. K. Vasu, and S. Vijaykumar, "Evaluation on effect of printing process parameter through Taguchi approach on mechanical properties of 3D printed PLA specimens using FDM at constant printing temperature," Mater. today Proc., vol. 52, pp. 1288–1293, 2022.
[14] R. A. Jarvis, "A perspective on range finding techniques for computer vision," IEEE Trans. Pattern Anal. Mach. Intell., no. 2, pp. 122–139, 1983.
[15] M. Hussain, J. J. Bird, and D. R. Faria, "A study on CNN transfer learning for image classification," in Advances in Computational Intelligence Systems: Contributions Presented at the 18th UK Workshop on Computational Intelligence, September 5-7, 2018, Nottingham, UK, 2019, pp. 191–202.
[16] R. Yang and Y. Yu, "Artificial convolutional neural network in object detection and semantic segmentation for medical imaging analysis," Front. Oncol., vol. 11, p. 638182, 2021.
[17] J. Haupt and R. Nowak, "Compressive sampling vs. conventional imaging," in 2006 International Conference on Image Processing, 2006, pp. 1269–1272.
[18] J. Gu et al., "Recent advances in convolutional neural networks," Pattern Recognit., vol. 77, pp. 354–377, 2018.
[19] H. Perez, J. H. M. Tah, and A. Mosavi, "Deep learning for detecting building defects using convolutional neural networks," Sensors, vol. 19, no. 16, p. 3556, 2019.
[20] P. Kumar and U. Misra, "Deep Learning for Weed Detection: Exploring YOLO V8 Algorithm's Performance in Agricultural Environments," in 2024 2nd International Conference on Disruptive Technologies (ICDT), 2024, pp. 255–258.
[21] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," 2023. https://github.com/ultralytics/ultralytics (accessed May 16, 2024).
[22] M. Hussain, "YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection," Machines, vol. 11, no. 7, p. 677, 2023.