2025 European Conference on Computing in Construction

CIB W78 Conference on IT in Construction

Porto, Portugal
July 14-17, 2025

AUTOMATED DEFECT DETECTION IN FUSED FILAMENT FABRICATION COUPLING DEEP LEARNING AND COMPUTER VISION

Thamer Al-Zuriqat, Mahmoud Noufal, Patricia Peralta, Kosmas Dragos, and Kay Smarsly
Hamburg University of Technology, Hamburg, Germany

Abstract

Fused filament fabrication (FFF) is an additive manufacturing technique, popular due to its versatility and cost-effectiveness. However, FFF machines, such as 3D printers, are prone to runtime errors, wasting time and material, while requiring constant human supervision. This paper presents a defect detection approach for FFF processes based on artificial intelligence, combining convolutional neural networks and computer vision. The defect detection approach is validated using 3D prints designed to mimic common FFF defects. The results demonstrate the capability of the proposed approach to automatically detect defects in FFF processes, thereby reducing time and material waste as well as the need for human supervision.

Introduction

The increasing demand for efficient and cost-effective products has led to an increase in product design complexity (ElMaraghy et al., 2012). As design requirements become increasingly intricate, a transformative approach to manufacturing, known as "additive manufacturing" (AM), has emerged, which has unburdened the "creative" production process from traditional machining limitations (Lindwall et al., 2021). Furthermore, AM offers great potential in mass production with high complexity (Sharon, 2014).

Fused filament fabrication (FFF), commonly known as 3D printing, represents one of the most widely adopted AM processes, with increasing applications in the architecture, engineering, and construction (AEC) industry. In AEC industry, FFF – for example, using concrete or clay 3D printers – have shown promise in fabricating customized building components, formwork, and structural elements, offering advantages such as material efficiency, design flexibility, and reduced construction waste. FFF machines (hereinafter termed "3D printers") use stepper motors and extruders to layer polymer-based materials ("filament") into of predefined shapes forming products ("printed products"). During FFF processes, 3D printers operate by executing numerical control algorithms, referred to as G-code, and require human supervision for monitoring the FFF process (Shahrubudin et al., 2019). During the FFF process, multiple variables such as extrusion speed, nozzle temperatures, bed temperatures, as well as the filament type and moisture level critically influence the printed product quality (Prabhakar et al., 2021).

3D printers are prone to runtime errors, which often manifest in printed products as defects (hereinafter termed "FFF defects"). FFF defects consist of irregularities, imperfections, and structural failures which may degrade the quality of printed products. FFF defects may result in time and material being wasted, since an FFF process is often restarted from the beginning in the event of a defect. Consequently, continuous human supervision of the FFF process is essential. FFF defects may appear on the external surfaces of printed products or may occur internally and remain unnoticed.

Efforts towards automated defect detection approaches for FFF processes have been reported in a body of research. Zhao et al. (2023) have proposed an offline vision-based defect detection approach which is conducted after printing is finished. Zhao et al. (2023) have used a 3D scanner to generate a point cloud of the printed product surfaces to be compared with another point cloud generated from a computer-aided design (CAD) model of the same product. Khan et al. (2021) have introduced a defect detection approach based on a custom binary-classification convolutional neural network (CNN) model for real-time defect detection of FFF processes with 84% accuracy. However, the aforementioned approach is limited to infill patterns due to the top-view setup; therefore, the CNN model is unable to detect other defect classes. Paraskevoudis et al. (2020) have presented a single-shot detector to detect stringing defects of FFF processes. Xie et al. (2022) detected warping defects by comparing key points between the printed product and an image thereof. Xu et al. (2023) have introduced a defect detection approach using a modified light-weight version of the fourth-generation "You Only Look Once" (YOLOv4) CNN architecture. Nonetheless, Xu et al. (2023) have used a homogeneous dataset to train the YOLOv4 model, rendering the model capable of detecting only one type of defects. In summary, despite the importance of defect detection in FFF processes, a general and automated defect detection

approach in FFF processes, regardless of the shape of the printed product and the FFF defect class, has yet to be investigated.

In this paper, an approach towards automated defect detection (ADD) in FFF processes is proposed, which may serve as a foundation for implementing defect detection in the AEC industry, specifically for concrete and clay 3D printers. The ADD approach couples deep learning models, specifically CNN classification models, with computer vision to detect FFF defects. Furthermore, the ADD approach takes into consideration six external FFF defect classes; namely, stringing, cracking, blobs, spaghetti, under-extrusion, and layer gaps. The results demonstrate the capabilities of the ADD approach to detect defects in FFF processes, thus reducing time and waste of materials, as well as the need for human supervision.

The remainder of this paper is structured as follows: First, FFF defect classes included in this research are illuminated. Then, the design of the ADD approach is introduced, implemented, and validated in a real-world test, in which images of printed products are intentionally contaminated with flaws, designed to mimic FFF defects. Next, the results of the validation test are presented, and further mitigation strategies to improve the performance of the classification models are discussed. Finally, the summary and conclusions as well as an outlook on potential future research endeavors are provided.

FFF defect classes

In Figure 1, the external FFF defect classes considered in this work are presented. A stringing occurs due to oozing filament from the nozzle of FFF machines while moving without extrusion, and typically caused by low filament retraction or high nozzle temperatures. Cracking is structural failure that occurs between layers of a printed product due to filament warping, as well as different cooling rates between layers. Blobs are small and visible spherical extrusions of the filament, and commonly occur as a result of extruder pauses between successive horizontal movements. Spaghetti is a wire-like cluster of filaments when filament is extruded over empty regions, often resulting from large overhangs or detachments from the printing bed due to design or G-code issues. Under-extrusion is a result of insufficient filament extruded from the nozzle of FFF ma-chines, causing gaps between printed lines, visible layer lines, which may reduce the structural integrity of printed products. Finally, layer gaps are an extreme case of under-extrusion, in which printed products contain visible gaps, uneven surfaces, or missing sections. The aforementioned FFF defects serve as classes for the ADD approach, which is presented in the following section.

Design of the ADD approach

In this section, the design of the ADD approach is introduced, comprising two phases, (i) data preparation and (ii) classification model development, with each phase comprising several steps. A flowchart describing the workflow of the ADD approach is shown in Figure 2. In what follows, the two phases of the ADD approach are briefly discussed.

Phase 1: Data preparation

1. In the first step, a dataset D_n comprising n images of printed products, is collected from publicly available and open-access sources. The dataset D_n contains images of both printed products with FFF defects, i.e. stringing, cracking, blobs, spaghetti, under extrusion, and layer gaps, as well as defect-free printed products.

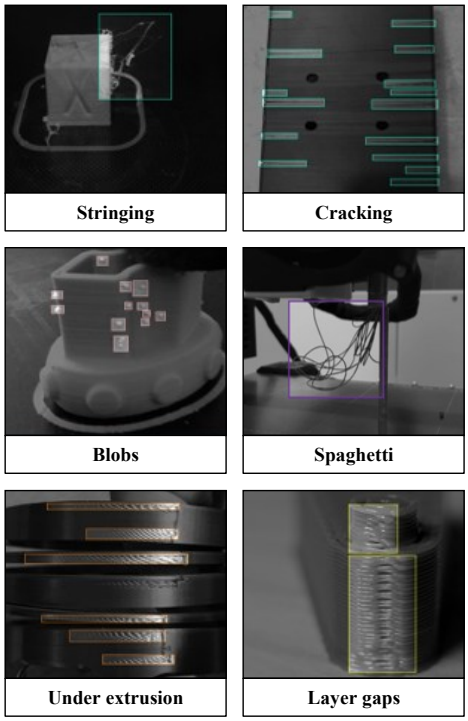

Figure 1: External FFF defect classes

2. Since the images in the dataset D_n are collected from different sources, the images may vary in size, color, and viewing angle. To enhance training outcomes in subsequent steps, the images are preprocessed by, first, cropping and resizing the images to a predefined size. Next, the images are transformed into gray-scale to prevent inadvertently correlating FFF filament colors with specific FFF defect classes. In a gray-scale image, each pixel represents a specific light intensity, ranging from black to white, with no color information. Furthermore, tilted images are adjusted to correct the orientation, ensuring a consistent, horizontal view. Finally, the images are labeled with the corresponding FFF defect class.
3. To enhance model training results and minimize the risk of under-fitting or over-fitting in the next phase, data augmentation techniques, such as mirroring, noise, light exposure (“brightness”), rotating, and random image cutouts, are applied to increase the size of the dataset D_n. Specifically, mirroring and rotating are used to improve robustness to random camera

angles, while noise, cutouts, and variable exposure increase resilience to varying lighting conditions. Then, the images in dataset D_n are split into a training dataset $D_{training}$, a validation dataset $D_{validation}$, and a testing dataset $D_{testing}$.

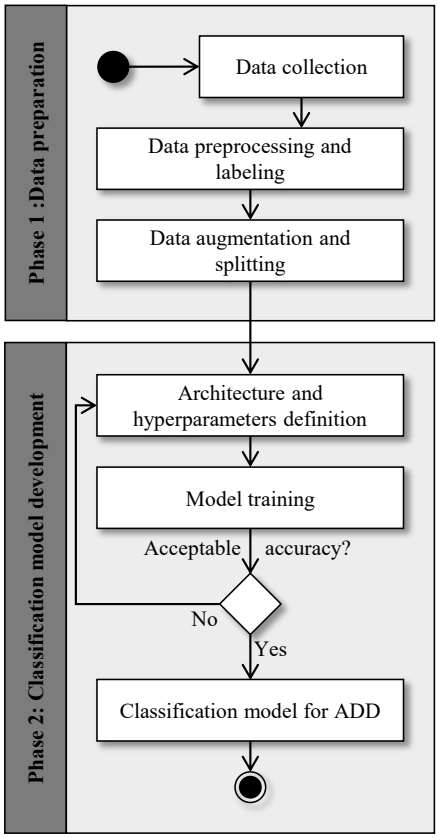

Figure 2: Flowchart of the ADD approach

Phase 2: Classification model development

1. In general, a CNN classification model consists of an input layer, convolutional layers, pooling layers, fully connected layers, and an output layer. The general architecture of a CNN classification model is illustrated in Figure 3. The input layer processes and feeds images to the convolutional layers (also known as kernels) which apply filters to detect low-level features in the images, such as edges or patterns, organized in feature maps. Pooling layers reduce the dimensionality of feature maps, lowering the computational load and making features robust to minor variations. Fully connected layers combine and interpret high-level features extracted by the convolutional and pooling layers to produce classification scores. Finally, the output layer uses the classification scores for predicting the classification category for each input, i.e. the FFF defect class.
2. Upon defining the CNN architecture, the training dataset $D_{training}$ is used to train the CNN classification model. Images in $D_{training}$ are fed into the models via the input layer as a 2D matrix **I**, where each I_{ij} represents the gray-scale pixel intensity at position ij ($0 < i$, $0 < j$) in the image, as shown in Equation 1. Each pixel has a gray-scale pixel intensity value ranging from 0 to 255 (0 is black, 255 is white, and values in between represent shades of gray). During training, a model "learns" from existing relationships between known inputs (images) and known outputs (FFF defect classes, e.g. "spaghetti"). Furthermore, loss metrics are used to guide the optimization of the CNN classification models, and evaluate how far the predictions of the models lie from actual target outcomes. In general, a decreasing loss value trending toward zero indicates that the model is effectively "learning" from the training dataset. Meanwhile, and to prevent overfitting during training process, the validation dataset $D_{validation}$ is used to fine-tune and evaluate the performance of the models during training.

$$\mathbf{I} = \begin{bmatrix} I_{1,1} & \cdots & I_{1,j} \\ \vdots & \ddots & \vdots \\ I_{i,1} & \cdots & I_{i,j} \end{bmatrix}, \qquad I_{i,j} \in R \,|\, 0 \le I \le 255 \quad (1)$$

3. Upon completing training, the testing dataset $D_{testing}$ is used to assess the performance of the classification models. In this step, evaluation metrics, such as precision (*pr*), recall (*rc*), and the mean average precision (*mAP*50) at a single intersection-over-union (*IoU*) threshold of 50% are used. On the one hand, *pr* and *rc* indicate capabilities of the models to avoid false positive (*FP*) classifications and false negative (*FN*) classifications, respectively. On the other hand, the *mAP*50 metric expresses the prediction accuracy of a

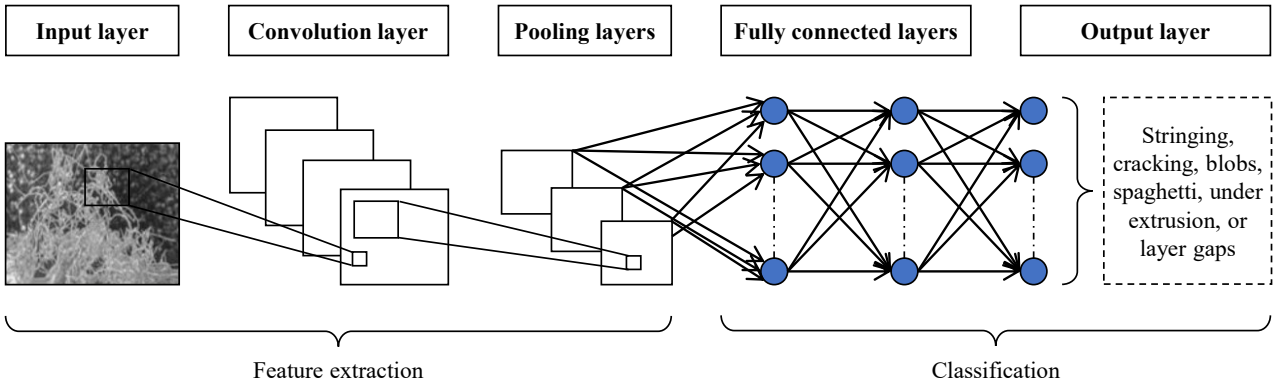

Figure 3: General architecture of a classification CNN

CNN classification model by averaging precision across all classes. The *IoU* threshold of 50%, set for computing the *mAP*50, ensures that a prediction is considered correct if it overlaps with the correct class by at least 50%. Figure 4 shows a distinction between actual and predicted classes in the so-called "confusion matrix", and Equations 2, 3, and 4 present the mathematical formulations of *pr*, *rc*, and *IoU*, respectively. CNN classification models with *mAP*50 value exceeding a predefined threshold are saved for deployment. Otherwise, steps 1 and 2 are repeated upon modifying the CNN architecture.

$$pr = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2}$$

$$rc = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3}$$

$$IoU = \frac{\Omega_o}{\Omega_u} \tag{4}$$

In Equation 4, Ω_o is the overlapping area between the predicted region and the ground truth, and Ω_u is the total area covered by both the predicted region and the ground truth.

Confusion matrix		Predicted class	
		Positive	Negative
Actual class	Positive	True positives (TP)	False negatives (FN)
	Negative	False positives (FP)	True negatives (TN)

Figure 4: Distinction between actual and predicted classes

The implementation and validation of the ADD approach using images of printed products designed to mimic common FFF defects is presented in the following section.

Implementation and validation of the ADD approach

In this section, the implementation and validation of the ADD approach is presented. First, the implementation is shown in accordance with the phases of the ADD approach previously described, i.e. "data preparation" and "classification model development". The programming language Python is utilized in both phases of the implementation to detect defects in FFF processes. An additional phase for the implementation involves the deployment and management of the CNN classification model within FFF processes. Then, the validation of the ADD approach is conducted using printed products specifically designed to mimic FFF defects.

Phase 1: Data preparation

The dataset D_n is collected form publicly available and open-access sources, found on the web, such as Roboflow Universe (Roboflow, n.d.), r/FixMyPrint (Reddit, n.d.-a) and r/3Dprinting (Reddit, n.d.-b) on Reddit, and YouTube. Next, to standardize the images size, color, and viewing angle, as well as to label images with the corresponding FFF defect class, a software application equipped with a graphical user interface (GUI) for image preprocessing is developed ("ADD preprocessor"). Since the data preparation phase requires a significant amount of manual input, the purpose of the GUI is to allow faster preprocessing using the ADD preprocessor. Figure 5 presents a view of the GUI. When importing an image into the preprocessing ADD preprocessor the printed products are manually marked using the GUI. Then, the image is resized to a standardized dimension, centered around the area previously marked as printed products to ensure that the region of interest remains the focal point of the image. Next, the viewing angle of the printed product is adjusted to ensure horizontal orientation, and the image is converted to gray-scale.

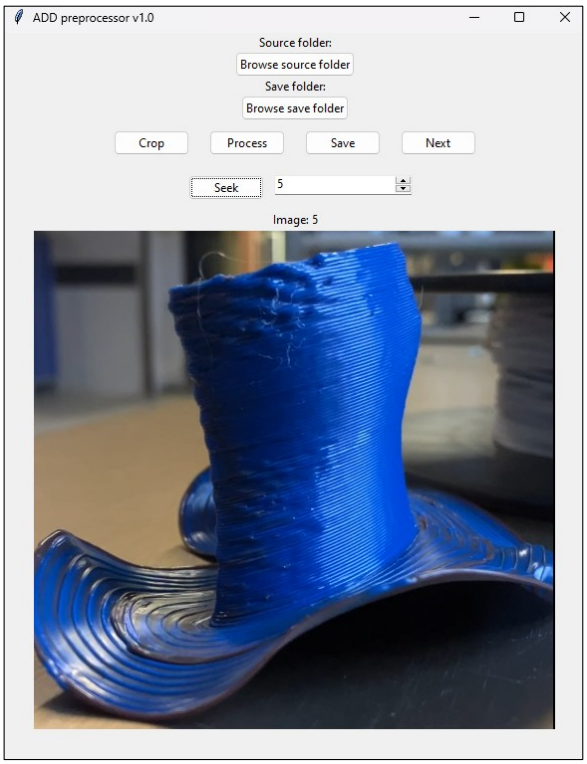

Figure 5: GUI for images preprocessing

Finally, the data augmentation is also handled by the ADD preprocessor, which applies mirroring, adding noise, adjusting brightness, and rotating to the gray-scale images.

Phase 2: Classification model development

In this phase, the CNN architecture is defined based on the "You Only Look Once" (YOLO) paradigm. YOLO is a state-of-the-art CNN architecture by Ultralytics (Ultralytics, n.d.-a), which is used for object detection and tracking, instance segmentation, image classification, and pose estimation. Moreover, the YOLO is designed to prioritize speed, accuracy, and user-friendliness while

preserving computational efficiency. The YOLO CNN architecture has shown faster and more accurate predictions when compared with other alternatives for object detection and classification, such as region-based convolutional neural networks or single-shot detectors (Neural Magic, 2023).

As a result, using the python Ultralytics library (Ultralytics, n.d.-b), several models based on different versions of the YOLO architectures are defined. Upon defining the architectures, the CNN classification models are trained using D_n. Finally, the classification model with the best performance is selected for defect detection in FFF processes.

Classification model deployment and management

A dedicated software application ("ADD manager"), equipped with a GUI, is developed to give users control over the CNN classification model during FFF processes, built around Octoprint (Häußge, n.d.), a web-based interface for managing 3D printers. Figure 6 presents the GUI of the ADD manager.

On the left side of the GUI, a live stream of the FFF process received from Octoprint, is shown, which is fed to the CNN classification model. On the right side of the GUI, a control panel allows users to control the FFF process, choose the processor on which the GUI is running, i.e. CPU or GPU, control the confidence threshold of the CNN classification model (e.g. low confidence threshold increases the sensitivity of the CNN classification model), and display current confidence levels on predictions for each of the six FFF defect classes.

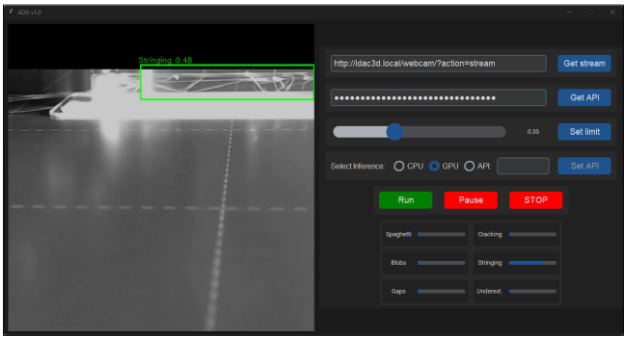

Figure 6: GUI of the ADD manager application

Finally, to pause the printing process when detecting an FFF defect, a pausing algorithm is used. In the pausing algorithm, a dynamic event queue monitors predictions over a time period, updating every 5 seconds. If an FFF defect is detected consistently across a defined window (e.g., 20 s), the printer is automatically paused. Consequently, the pausing algorithm ensures that only persistent defects trigger the pause, avoiding unnecessary interruptions of the printing process.

Description of the validation test

To validate the ADD approach, a real-world test is conducted using printed products designed to mimic FFF defect classes. The main goal of the validation test is to assess the ability of the CNN classification model to accurately detect the FFF defect and, consequently, stop the 3D printing process.

For the validation test, a Prusa Original i3 MK3S 3D printer (Prusa Research, 2023) is employed as the printing hardware. A Raspberry Pi 4 with 8GB of RAM (Raspberry Pi Foundation, 2020) is connected to the 3D printer via a USB cable, and acts as a control unit for managing the 3D printer. A Logitech 1080p Pro Stream Webcam (Logitech, n.d.), capable of recording and streaming high-definition video at 1080p and 30FPS, is mounted on the z-axis (vertical) to monitor the printing area, and is connected to the Raspberry Pi. Figure 7 illustrates the hardware setup of the validation test.

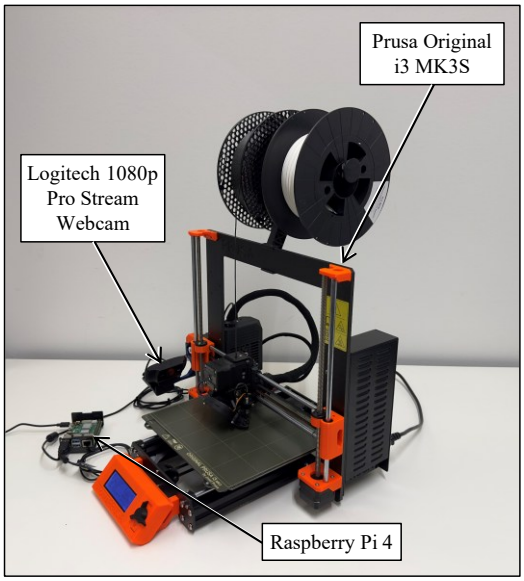

Figure 7: Validation test set-up

For data preparation, a dataset of size n = 4,124 is collected ($D_{n=4,124}$), comprising images of 3D printed products exhibiting FFF defects as well as defect-free images. In $D_{n=4,124}$, a total number of 832 images included 3D printed products with FFF defects, and a total number of 3292 defect-free images. Next, the images in the dataset are sequentially imported into the ADD preprocessor application and resized to a dimension of 640×640 pixels, which is considered a trade-off between efficient batch training within system memory limits and preservation of sufficient image details. The data augmentation, performed using the ADD preprocessor, is performed by generating two copies of each image in $D_{n=4,124}$ are generated, one of which is mirrored. Then, the two copies of each image are subjected to noise in which 4% of the gray-scale pixel intensity value I_{ij} at positions i and j ($0 < i < 640$, $0 < j < 640$) are replaced with a random value between 0 and 255 (see phase 2, step 2). Then, the two copies are exposed to a random change of brightness in value of ±35%, as well as a random rotation of ±20°. It is important to mention that high degrees of rotation may result in an unrealistic appearance of FFF defects. For example, strings and blobs tend to lean downwards due to gravity, and a falling upward filament is not logical. As a

result, of the dataset $D_{n=4,124}$ is increased by three times to n = 12,372 and split into 80% training dataset ($D_{training=9,897}$), 15% validation dataset ($D_{validation=1,855}$), and 5% testing dataset ($D_{testing=620}$).

The CNN architecture is selected upon conducting preliminary tests with four YOLO architectures, namely the YOLOv8l, YOLOv9c, YOLOv10l, and YOLOv10b. The training is performed on a computer equipped with an Intel Core i9-14900K processor with 16 cores paired with 64GB of DDR5-6000 RAM, NVIDIA GeForce RTX 4090 GPU with 24 GB of memory, as well as 2TB Samsung 990 Pro hard disk drive. The GPU is used to accelerate the training process enabling efficient handling of the computationally intensive training of the CNN classification models. The number of training epochs is set to 250, and the loss is computed at every epoch. Loss in YOLO is based on multiple components, including: (i) Box loss for bounding box localization, (ii) binary cross-entropy loss for classification (class loss), and (iii) dual focal loss for refining bounding ensuring accurate and robust object detection (Ultralytics, n.d.-a). Periodically (upon completing a number of epochs), the validation dataset $D_{validation=1,855}$ is propagated through the CNN to evaluate its loss ("validation loss") and ensure that the validation loss follows the same trend as the training loss, as a means of avoiding overfitting. The training is performed for all candidate CNN architectures; Figure 8 and Figure 9 exemplarily depict the loss metrics for $D_{training=9,897}$ and $D_{validation=1,855}$ of the YOLOv8l and YOLOv9c, respectively.

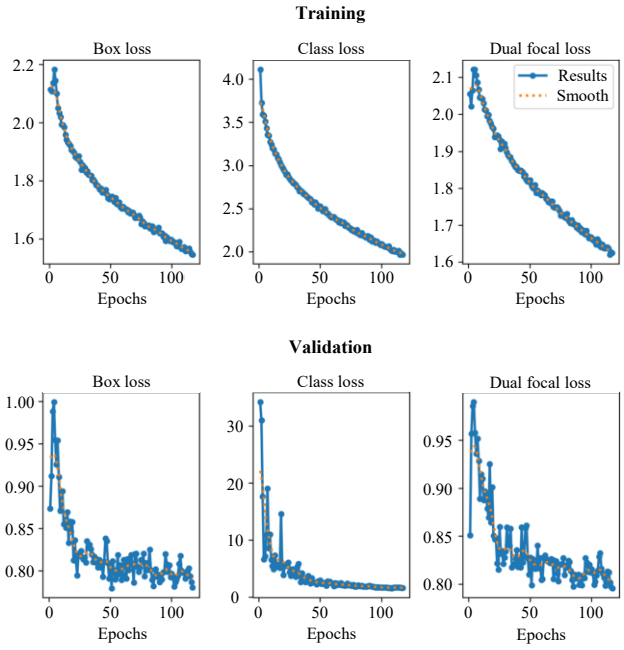

Figure 8: Loss metrics for $D_{training=9,897}$ (top) and $D_{validation=1,855}$ (bottom) against training epochs for the classification model based YOLOv8l architecture

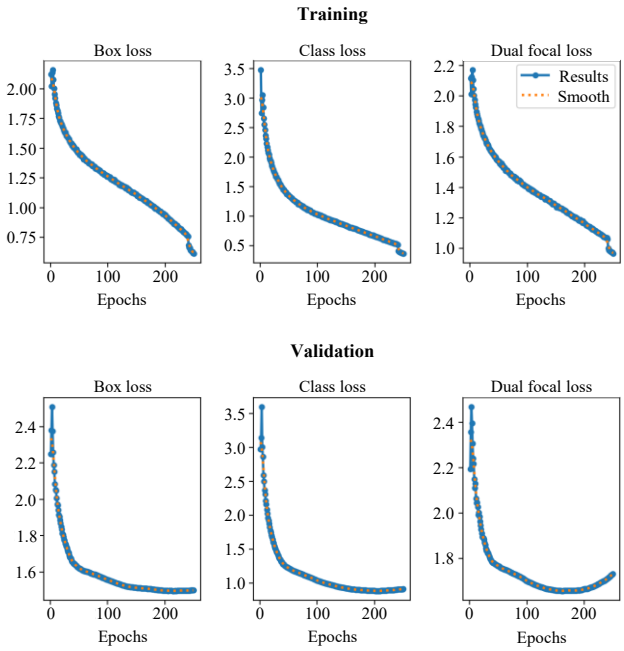

Figure 9: Loss metrics $D_{training=9,897}$ (top) and $D_{validation=1,855}$ (bottom) against training epochs for the classification model based YOLOv9c architecture

The performance of the candidate CNN architectures is evaluated based on the dataset $D_{testing=620}$ (not introduced during training). Table 1 summarizes the performance of all candidate architectures, in terms of loss metrics and mAP50 values. As a result, the architecture YOLOv9c is selected for the CNN classification model in this study, as exhibiting the highest *mAP*50 value of 82.2%.

Table 1: Performance of all candidate architectures

Model architecture	Box loss	Class loss	mAP50
YOLOv8l	0.824	2.101	37.6%
YOLOv9c	1.497	0.890	82.2%
YOLOv10l	3.118	1.978	81.7%
YOLOv10b	3.107	2.058	81.7%

Next, a CAD model, featuring two vertical columns fixed on a base plate, is designed. Then, the CAD-model is processed using the open-source slicing software developed by Prusa Research, PrusaSlicer 2.8.1, which is used to prepare models for 3D printing by generating the corresponding G-code (Prusa Research, 2024). Polylactic acid (PLA) filament is used, for which the nozzle temperature typically stays within the range of 180-220°C. To intentionally induce a FFF defect, specifically stringing, the nozzle temperature is set to 260°C.

Results and discussion

In this section, the results of the validation test of the ADD approach are presented, followed by a discussion on

mitigation strategies to improve further the performance of the CNN classification model.

The results of the validation test, shown in Figure 10, confirm the effectiveness of the ADD approach in detecting FFF defects at a proof-of-concept level. The CNN classification model based on the YOLOv9c architecture successfully identified the stringing defect and automatically paused the 3D printer using the pausing algorithm, thereby reducing time and material waste as well as eliminating the need for human supervision. In particular, the stringing defect reached the previously defined persistence time threshold (20 s), and, consequently, paused the FFF process.

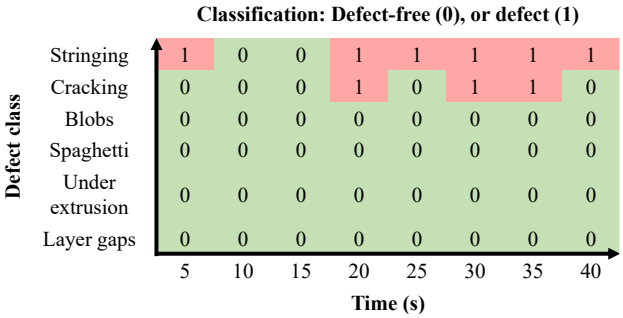

Figure 10: Event queue and defect detection in the validation test

Moreover, as can be seen from Figure 10, misclassifications of the CNN model are reduced to a minimum. Specifically, no identification of blobs, spaghetti, under-extrusion, or layer gaps are observed. However, under certain lighting conditions, the CNN classification model classifies parts of the 3D printer, specifically the extruder, as a cracking defect, as shown in Figure 11. A similar issue is also noticed with the grid lines of printing bed, in which the grid lines are detected as cracking defects.

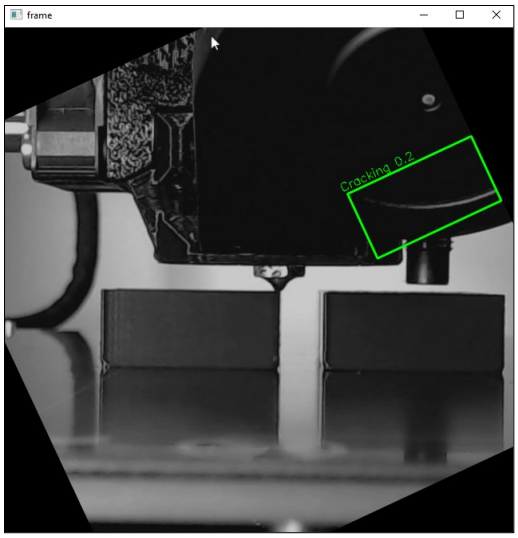

Figure 11: Misclassifying the extruder of the 3D printer as a cracking defect under certain lighting conditions

To overcome misclassifications, mitigation strategies such as masking, and image rotation are used. First, in masking, the live stream is masked with a black rectangle blocking the extruder down to the nozzle. A similar masking region is drawn over the lower part of the live stream to block the grid lines of the printing bed. Secondly, to improve the sensitivity of the CNN classification model to image orientation, the live stream is rotated by 30° clockwise and counterclockwise with respect to the initial orientation for inspection if no defects are detected for a period of 20 s.

Summary and conclusions

Fused filament fabrication represents one of the most widely adopted AM processes, with increasing applications in the AEC industry. In AEC industry, FFF – for example, using concrete or clay 3D printers – have shown promise in fabricating customized building components, formwork, and structural elements, offering advantages such as material efficiency, design flexibility, and reduced construction waste. Despite the importance of defect detection in FFF processes, a general and automated defect detection approach in FFF processes, in which FFF defect classes are detected regardless of the shape of the 3D printed product, has received scarce attention.

This paper has presented an automated defect detection approach in FFF coupling deep learning and computer vision. The ADD approach combines convolutional neural networks as well as computer vision to autonomously detect defect and pause FFF processes in case of defect classes, i.e. stringing, cracking, blobs, spaghetti, under extrusion, and layer gaps, are identified. The ADD approach has been implemented using the programming language Python, including the development of a CNN classification model and two applications, one for data preprocessing and one for managing FFF processes. Validation tests have been conducted, first by defining the CNN architecture, based on preliminary tests with four candidate CNN architectures, the YOLOv8l, YOLOv9c, YOLOv10l, and YOLOv10b. Then, the CNN classification model based on the YOLOv9c architecture has been selected, based on the performance indicated by the highest mAP50 value of 82.2%. Images from printed products have specifically been designed to mimic FFF defect classes. The test results have proven the capability of the ADD approach in detecting defects in FFF processes, at a proof-of-concept level, by correctly identifying a stringing defect and pausing the printing process, thus reducing time and material waste as well as the need for human supervision in FFF processes. Furthermore, sporadic misclassifications, observed in the validation test may be remedied via masking and image rotation.

Future work may involve developing the ADD approach further to detect defects in multi-nozzle FFF machines. Furthermore, the ADD approach may serve as a foundation and be transferred to defect detection in the similar material extrusion processes in the AEC industry, specifically for concrete or clay 3D printers.

Acknowledgments

This paper is the result of a collaborative effort of authors funded by different sources. The authors would like to gratefully acknowledge the support offered by Federal Ministry for Digital and Transport (BMDV) within the mFUND program under grants 01FV2013B and 01FV2059C as well as by the German Research Foundation (DFG) under grants SM 281/20-1, SM 281/22-1, and SM 281/33-1. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of BMDV or DFG.

References

ElMaraghy, W., ElMaraghy, H., Tomiyama, T. and Monostori, L. (2012) Complexity in engineering design and manufacturing. CIRP Annals, 61(2), pp.793-814.

Häußge, G. (n.d.) OctoPrint.org. OctoPrint.org. Available at: https://octoprint.org/ (Accessed: 12 November 2024).

Khan, M.F., Alam, A., Siddiqui, M.A., Alam, M.S., Rafat, Y., Salik, N. and Al-Saidan, I. (2021) Real-time defect detection in 3D printing using machine learning. Materials Today: Proceedings, 42, pp.521-528.

Lindwall, A. and Wikberg Nilsson, Å. (2021) Exploring creativity management of design for additive manufacturing. International Journal of Design Creativity and Innovation, 9(4), pp.217-235.

Logitech (n.d.) C922 Pro Stream Webcam Datasheet. Available at: https://www.logitech.com/assets/64889/c922-pro-stream-webcam.pdf?srsltid=AfmBOoqmSnQgKGzhXEPK5tlEiBUAAqBd-TSp9fQUCAuNO01dMc1ybzkM (Accessed: 7 November 2024).

Neural Magic (2023) YOLOv8 Detection 10x Faster With DeepSparse—Over 500 FPS on a CPU. Neural Magic. Available at: https://neuralmagic.com/blog/yolov8-detection-10x-faster-with-deepsparse-500-fps-on-a-cpu (Accessed: 27 October 2024).

Paraskevoudis, K., Karayannis, P. and Koumoulos, E.P. (2020) Real-time 3D printing remote defect detection (stringing) with computer vision and artificial intelligence. Processes, 8(11), p.1464.

Prabhakar, M.M., Saravanan, A.K., Lenin, A.H., Mayandi, K. and Ramalingam, P.S. (2021) A short review on 3D printing methods, process parameters and materials. Materials Today: Proceedings, 45, pp.6108-6114.

Prusa Research (2023) Prusa3D MK3S Manual. Available at: https://www.prusa3d.com/downloads/manual/prusa3d_manual_mk3s_en_3_11.pdf (Accessed: 7 November 2024).

Prusa Research, (2024). PrusaSlicer, version 2.8.1 [computer software]. Available at: https://www.prusa3d.com/prusaslicer/ (Accessed: 5 December 2024).

Raspberry Pi Foundation (2020) Raspberry Pi 4 Datasheet. Available at: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf (Accessed: 7 November 2024).

Reddit (n.d.-a) r/FixMyPrint: A Community for 3D Printing Troubleshooting. Reddit. Available at: https://www.reddit.com/r/FixMyPrint/ (Accessed: 30 April 2024).

Reddit (n.d.-b) r/3Dprinting: Discussing All Things 3D Printing. Reddit. Available at: https://www.reddit.com/r/3Dprinting/ (Accessed: 12 May 2024).

Roboflow (n.d.) Universe: Explore AI Datasets and Pretrained Models. Roboflow. Available at: https://universe.roboflow.com/ (Accessed: 29 April 2024).

Shahrubudin, N., Lee, T.C. and Ramlan, R.J.P.M. (2019) An overview on 3D printing technology: Technological, materials, and applications. Procedia manufacturing, 35, pp.1286-1296.

Ultralytics (n.d.-a) Ultralytics Models Documentation. Ultralytics. Available at: https://docs.ultralytics.com/models/ (Accessed: 19 October 2024).

Ultralytics (n.d.-b) Ultralytics. Available at: https://pypi.org/project/ultralytics/ (Accessed: 19 October 2024).

Xie, J., Saluja, A., Rahimizadeh, A. and Fayazbakhsh, K. (2022) Development of automated feature extraction and convolutional neural network optimization for real-time warping monitoring in 3D printing. International Journal of Computer Integrated Manufacturing, 35(8), pp.813-830.

Xu, L., Zhang, X., Ma, F., Chang, G., Zhang, C., Li, J., Wang, S. and Huang, Y. (2023) Detecting defects in fused deposition modeling based on improved YOLO v4. Materials Research Express, 10(9), p.095304.

Zhao, X., Li, Q., Xiao, M. and He, Z. (2023) Defect detection of 3D printing surface based on geometric local domain features. The International Journal of Advanced Manufacturing Technology, 125(1), pp.183-194.