Development of an automated AI system for real-time error detection in 3D printing

Kenan Abdrabouh[1], *Urban* Frank[1*], *Tim* Richard[1], *Jannis* Sinnemann[1] and *Arockia Selvakumar* Arockia Doss[2]

[1]BO Smart Factory, University of Applied Sciences Bochum, 44801 Bochum, Germany.
[2]School of Mechanical Engineering, Vellore Institute of Technology, Chennai, Tamil Nadu, India

Abstract. As an additive manufacturing process, 3D printing offers a very wide range of applications in industrial use. From prototype construction to small series production, components that are needed at short notice can be made available quickly without long set-up times. The industrial use of AI methods for quality assurance in Industry 4.0 processes increases the efficiency of manufacturing processes. For automated additive manufacturing, the implementation of AI systems specifically for error monitoring is essential. The selection and training of a suitable AI language model enables reliable and early detection of 3D printing errors, thus facilitating automation and integration into existing manufacturing processes.

1 Introduction

In recent years, 3D printing has evolved from an experimental manufacturing technology to a central component of modern production processes. Its flexibility and the ability to produce complex geometries without costly tools give it considerable advantages in industry and research. It is particularly relevant for the manufacture of diverse products in small quantities, where classic mass production reaches its limits. In the public perception, it is also compared to a potential paradigm shift in production, similar to the effect of letterpress printing on the dissemination of knowledge [1]. Nevertheless, 3D printing is not without its challenges: various errors can occur during the manufacturing process. For example, layer shifts, material interruptions or surface defects can lead to significant quality losses or the complete rejection of the component. In many cases, these errors are currently detected manually, which is time-consuming and resource-intensive. At the same time, artificial intelligence (AI), particularly in the field of deep learning, has made considerable progress in recent years. In image recognition, neural networks and detection algorithms based on them, such as YOLO (You Only Look Once), have proven to be powerful. Such models can identify and classify objects in real time identify and classify objects in real time and are already being used in areas such as autonomous driving and quality control in industrial processes [2][3]. The transfer of this

* Corresponding author: urban.frank@hs-bochum.de

technology to 3D printing opens up new possibilities: automated error detection during the printing process could not only reduce waste, but also enable direct control of the printer. Against this background, the question arises as to how reliably an AI-based system for detecting typical 3D printing errors works and to what extent it can automatically monitor and control the printing process. This thesis therefore develops an approach in which YOLO models are trained and optimised for error detection. In addition to model development, integration into the hardware of a university environment and connection to an automated control system are also investigated.

2 State of technology

2.1 Basics of 3D printing

The term 3D printing refers to a group of additive manufacturing processes in which materials are built up layer by layer to create a three-dimensional object. This technology differs fundamentally from subtractive processes such as milling or turning, in which material is removed by mechanical processing. The variety of 3D printing processes leads to different applications, technical properties and specific advantages and disadvantages. An informed choice of a suitable process therefore requires knowledge of the respective technological conditions, materials and limitations [4]. This paper deals exclusively with the Fused Layer Modelling (FLM) process. The FLM process, also known as Fused Deposition Modelling (FDM) or Fused Filament Fabrication (FFF), is based on the processing of thermoplastics, which are melted by heating, extruded through a nozzle and applied in layers. The FDM printing process can be used to print a wide variety of materials. The most commonly used material for manufacturing objects using the FDM process is plastic, in its many different types and formulations. However, FDM can also be used to print metals, composites and ceramics [5]. An FDM 3D printer usually consists of the following core components: print head & hot end, nozzle, extruder, print bed, linear movements and the control system [4]. In the print head, the filament is heated until it melts and then extruded through a nozzle. The molten material is applied to the print bed in layers. The extruder, a motor that can be installed in the print head or externally, ensures the controlled feed of the filament. The object is created layer by layer on the heated print bed, which improves the adhesion of the first layers and prevents warping. The linear movements of the printing process take place along three axes: X, Y and Z, each of which is controlled by stepper motors. In most printers, the print bed moves along the Y-axis, while the print head moves along the X and Z axes (see FIGURE 2). The control system regulates the axis movements, the extrusion of the filament, the temperatures to be set and the cooling [4]. In FDM printers, the material is fed into the print head or extruder in the form of a filament wound onto a spool. There is a wide variety of colours There is a wide variety of colours and diameters of filaments. The filament and the quality of the material have a significant impact on the print. Materials differ in terms of stability, elasticity and heat resistance, among other things. At the same time, the effort required for processing also varies depending on the material. Common plastic filament materials include PLA, PETG, ABS and PVA [4].

2.2 3D Printing Errors

3D printing errors are very common in FDM 3D printing technology and represent the greatest challenges in the manufacture and automation of 3D printing processes. Such

errors include, for example: stringing, spaghetti, under-extrusion, over-extrusion, blobs, warping, layer shifts, cracks and others.

Stringing: Stringing is a 3D printing error in which fine threads are created by filament trailing during nozzle movements. This defect is usually caused by excessive temperatures, insufficient retraction settings, or excessively moist filament and has a negative effect on surface quality [6], [7].

Spaghetti: The spaghetti defect refers to a production error in which the nozzle continues to extrude material into empty space without building the object correctly. The result is a disordered tangle of filament resembling spaghetti. Depending on the severity, the print can either be salvaged or must be completely aborted [7], [8].

Under-extrusion: The under-extrusion error describes the error that occurs when too little material is extruded. This leads to visible gaps, dots or holes and impairs the mechanical stability and surface quality of the object [9].

Overextrusion: Overextrusion occurs when too much material is extruded. The object is basically similar to the model, but the surface has an uneven or fused structure [7].

Blobs: This refers to excess material that accumulates on the surface of the printed object. This can be caused by a partially clogged nozzle or an excessively high extrusion rate [10].

Warping: Warping describes a distortion of the printed object, which is usually caused by temperature differences during the printing process. Uneven temperature distributions on the print bed, such as higher temperatures in the centre compared to the edges, can lead to material distortion and cause warping [4].

Layer shifts: Layer shifts occur when individual layers shift during the printing process, resulting in misalignment in the final product [10].

Cracks: Cracks occur in 3D printing mainly due to material properties or thermal stresses, for example when a draught causes the temperatures of the component to change rapidly. Cracks have a negative effect on the strength and quality of the end product [10].

All these errors have a significant impact on the quality of the printed parts and involve a great deal of effort. Printing times are often very long and the errors often only become apparent later on. This very often leads to a waste of materials, time and operating costs. As a rule, these errors are detected and corrected manually, which is both time-consuming and prone to errors.

2.3 Image recognition

According to Nischwitz et al. [11], image recognition can be understood as a subfield of digital image processing that pursues not only the improvement of image material, but above all the extraction of information from digital image data. To this end, images are first segmented and their characteristic features extracted. These Stringing Spaghetti Under-extrusion Over-extrusion Blobs Warping Layershifts Cracks features are recorded in the form of descriptors which, similar to a fingerprint, enable unique identification and allow comparison with structures in other images. On this basis, individual segments are merged into objects and then classified, whereby not only geometric shapes are relevant, but also their functional significance. Image recognition thus encompasses the entire process from capture and feature extraction to classification and response to recognised objects [11]. In addition to segmentation, [12] describes template matching as a classical method of object recognition. This involves comparing the image content with a predefined template in order to make a yes/no decision about the presence of an object. Maximum matching occurs when the template lies exactly over the object being searched for. This method is considered robust against noise interference, but involves a high computational effort [12].

2.4 Artificial intelligence and deep learning

In recent years, the field of machine learning, especially deep learning, has shown great potential for automating defect detection in manufacturing. Current advances in AI-supported additive manufacturing processes emphasise the growing importance of AI-based methods for defect detection and quality assurance [2]. The term artificial intelligence can refer to the field of research concerned with the development of intelligent computer systems on the one hand, and on the other hand to the ability of such systems to perform tasks that normally require human intelligence [13]. The basic structure of artificial intelligence is based on the concept of artificial neurons organised in networks. These neural networks consist of layers that process inputs, transform them and pass them on to subsequent levels until a decision or classification is made [13].

Neural networks are simplified computational models based on the functioning of the human brain. They consist of artificial neurons, the concept of which was first introduced by McCulloch and Pitts in 1943 [13]. Each neuron processes input signals by assigning weights to them and summing them up. An activation function then transforms the sum into an output value that can be passed on to other neurons. The structure of a network can vary in complexity. Simple networks consist of only one layer, while multi-layer networks consist of an input layer, an output layer and intermediate hidden layers [13]. Feedforward networks, in which the signals flow in one direction only, are particularly widespread. A key feature of neural networks is their ability to learn. At the beginning, the weights are initialised randomly. During the training process, these weights are iteratively adjusted so that the network learns to respond correctly to certain patterns [13]. The learning process is based on the principle of backpropagation.

Deep learning is a sub-discipline of machine learning and refers to the use of multi-layered neural networks that are trained with extensive data sets to perform complex information processing tasks. Deep learning is based on deep neural networks consisting of an input layer, several hidden layers and an output layer (see FIGURE 4). In these networks, the neurons in each layer are fully connected to the neurons in the next layer. The learning process takes place by adjusting the weights between the neurons so that the input data is gradually transformed until the desired output is produced [14]. Convolutional neural networks (CNN) (see Figure 5) are widely used in image and object recognition. These NNs follow a clearly structured sequence of processing steps. The starting point is the input of the image in its multidimensional structure, consisting of height, width and colour channels. In the first steps, convolutions are used, which, with the help of trainable filters, extract local features such as edges, textures or simple patterns extract. Immediately after each convolution, a ReLU activation function is applied, which provides the necessary non-linearity and sets negative values to zero. This sequence of convolution and activation is supplemented by pooling layers, which enable a reduction in the amount of data while retaining essential information. By repeatedly applying this combination of convolution, ReLU and pooling (q-fold), increasingly complex features are extracted from the image in a hierarchical manner [14]. In the next step, the extracted features are transferred to a multilayer perceptron (MLP) consisting of several fully connected layers (r-fold). Here, too, ReLU activation functions are used to model non-linear decision boundaries. Finally, a softmax output layer provides the probability distribution across the possible classes, enabling the model to make a clear decision on the recognised object category. The architecture thus illustrates that CNNs represent a combination of automatic feature extraction and subsequent classification, with deep structures enabling successive abstraction of the image information [14].

2.5 The YOLO models

YOLO ('You Only Look Once') is a powerful and widely used object detection model that can be used with many different deep learning frameworks. YOLO models can detect errors at high speed in real time error detection. Unlike other approaches such as R-CNN (Region-based Convolutional Neural Networks), these models can infer the position of classes directly from the image pixels by looking at an image. With R-CNN, ROIs (Regions of Interest) must first be defined, which represent the interesting area of the image, and then a classification must be created [11]. With the YOLO algorithm, the image size is adjusted to the input settings of the neural network. After entering the image size, all data is calculated. If the probability value (confidence value) in the output layer exceeds a certain threshold, a classification is made and a corresponding bounding box (BB) is output together with the recognised class. The probability value indicates the probability with which the model assumes that a particular object has been correctly assigned to a class. Fig. 1 shows how the YOLO model works [11].

After classification, the model outputs a vector with five entries (x, y, w, h and class) that describes the position, width and height of the detected bounding box as well as the associated object class. A class refers to a specific object predefined during model training, for example a human being who is to be identified by the detection process [11]. The lightweight structure of YOLO models also extends their range of applications to edge devices, enabling quality control directly on site with minimal computing power. These advantages reinforce the choice of YOLO for detecting 3D printing errors [2].

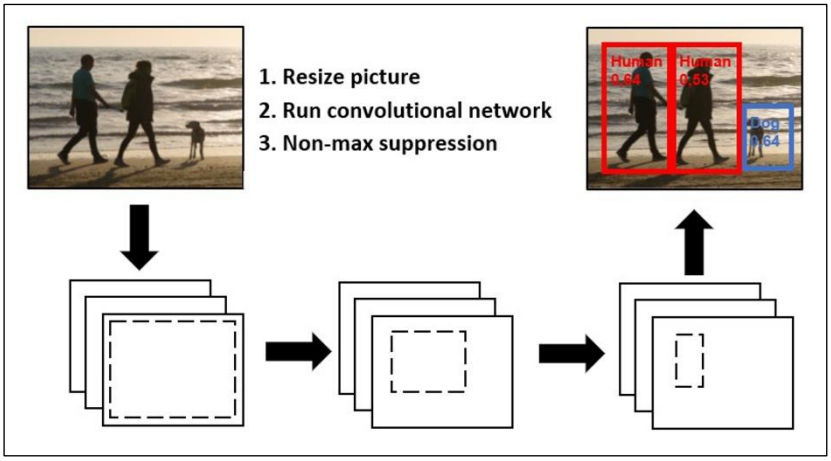

Fig. 1. The YOLO model

3 Development of the AI model

3.1 Preliminary tests on an exemplary computer setup

The preliminary tests on an exemplary PC without an interface to a 3D printer are carried out by first selecting the data set and narrowing down the Yolo models. The AI models are then trained with the selected data set, executed on the PC using Python code and tested.

The data set plays a crucial role in optical fault detection using deep learning. The size and quality of the data set is of central importance for training the AI model. Manually creating your own dataset consisting of self-made images is very time-consuming, especially if large datasets are to be used for training in order to ensure the generalisation capability of error detection. For this reason, the data sets used for the experiments were mainly taken from Roboflow [12]. This is an open-source platform known for high-quality data sets in the field of object recognition. Roboflow generally serves as a tool for managing and labelling image data sets. It enables both the cropping and pre-processing of images as well as manual labelling with class labels and coordinates, making the data usable for training a neural network [9]. The datasets from Roboflow Universe are not only already labelled and classified, but also include automatic division into training, validation and test data, as well as various methods of data augmentation, such as reflections, rotations, brightness and colour changes. For these reasons, selecting a suitable data set from Roboflow is an efficient method for quickly training a simple, high-quality YOLO model. The data set for the preliminary test consists of 323 images and a single error class, the spaghetti error [17]. In this data set, the training, validation and test data are distributed as follows: 73% training data, 13% validation data and 14% test data. This distribution deviates slightly from the usual data distribution (see Section 2.4.2). The images from this dataset are already augmented. In this case, two transformed instances were generated for each original training image by subjecting the image to a mirror transformation (see Fig. 2):

- Horizontal mirroring: transformation along the vertical axis.
- Vertical mirroring: transformation along the horizontal axis.

These transformations result in equivalent but varied representations of the original data.

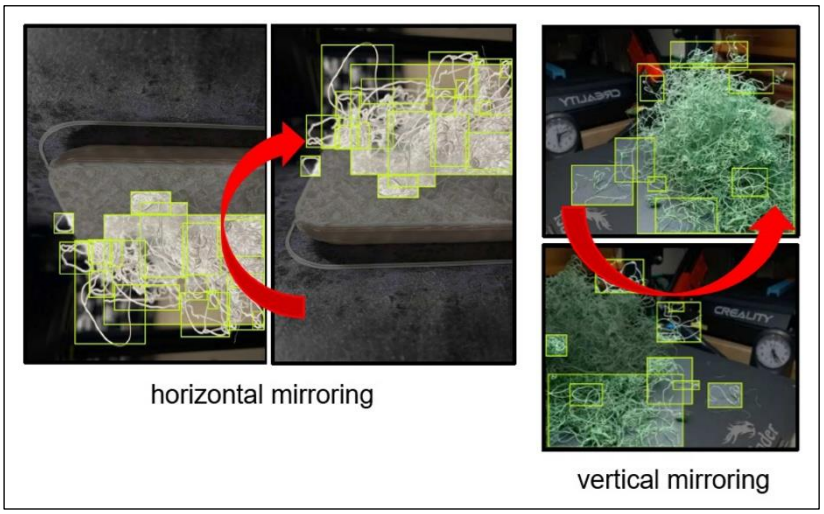

Fig. 2. Sample images from the dataset with augmentation (Source: Own presentation, image[16])

The YOLO models used for this work are trained on the basis of the open-source Ultralytics platform [18]. Since YOLOv5, Ultralytics has been offering a consistent and user-friendly framework that has become widely used in research and applications. A key advantage of the Ultralytics models (e.g. YOLOv5, YOLOv8, YOLOv11) is the combination of high accuracy, speed and user-friendliness. The framework offers a uniform API that enables training, evaluation and deployment with just a few Python commands. It also supports various tasks such as object detection, instance segmentation and pose estimation within the

same system [19]. Regular updates ensure that the models keep pace with the latest developments in deep learning. This makes Ultralytics a simple, practical and sustainable solution compared to other implementations. For these reasons, it was decided to use Ultralytics to train the YOLO models used in this work.

In the research work by Sani et al. in [2], various YOLO models were tested for error detection in 3D printing. Analysis of the YOLO models revealed significant differences in terms of convergence speed and performance. YOLOv9c and YOLOv10x achieved optimal results after just a few epochs, while YOLOv11m showed a good balance between training time and accuracy. YOLOv11s achieved the best test results with a mAP@0.5 of 0.8308 (see Chapter 2.4) and proved to be particularly suitable for real-time quality control in 3D printing. YOLOv11m also impressed with its high precision (0.9128) and recall (0.8990). In terms of computational efficiency, YOLOv11n, YOLOv10n and YOLOv9t proved to be particularly resource-efficient with low FLOPs, while YOLOv8x incurred significantly higher computational costs [2]. When implementing the AI model on a single-board computer (e.g. Raspberry Pi 5), care must be taken to ensure that the computing power does not overload the single-board computer. The performance of a computer can be assessed using the GFLOPs (giga floating-point operations per second) metric.

The Raspberry Pi 5 is used as an example in this paper. Benchmarks show that the system achieves approximately 25 GFLOPs in a typical configuration using the High Performance Linpack (HPL) test. When all cores are utilised and under optimal conditions, up to 73 GFLOPs can be achieved [20]. Independent comparative measurements confirm this order of magnitude and, depending on the benchmark method and optimisations, cite average values of 30335 GFLOPs [21]. The average value of 30 GFLOPs is therefore adopted in this paper as the limit for the selection of the YOLO model. To narrow down the model selection, the training data from the study by Sani et al. [2] is used (see Fig. 3).

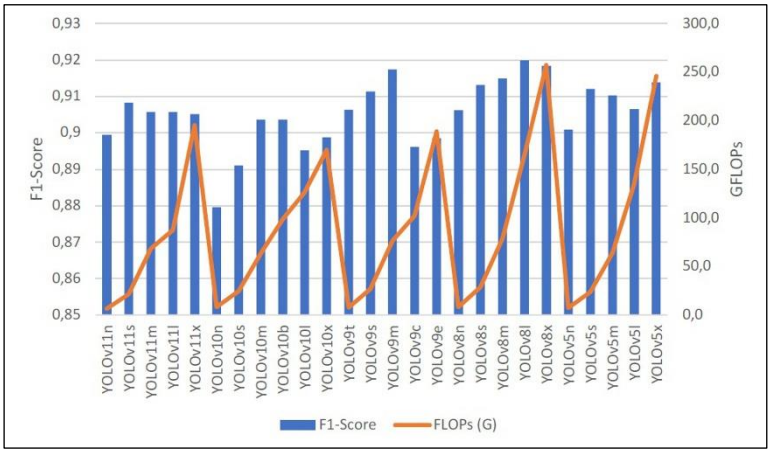

Fig. 3. F1 score and GFLOPs in comparison of different YOLO models - study on 3D printing error detection (Source: Chart based on data from [2])

The figure shows clear differences between the various YOLO models based on FLOPs and the F1 score. It can be seen that models with the suffix n (nano) require low computing power (low FLOPs), which has a positive effect on the efficiency of AI error detection. In contrast, the models with the suffix x require significantly higher computing power. FIGURE 3 also shows that the YOLOv8L model achieved the highest F1 score, while the YOLOv10n model achieved the lowest F1 score. In order to present the comparison from the study more clearly, it is important to consider the optimal combination of low GFLOPs

values (up to a maximum of 30 GFLOPs) and high precision or a good F1 score. Fig. 4 illustrates the YOLO models that are particularly suitable for the experiments in this work.

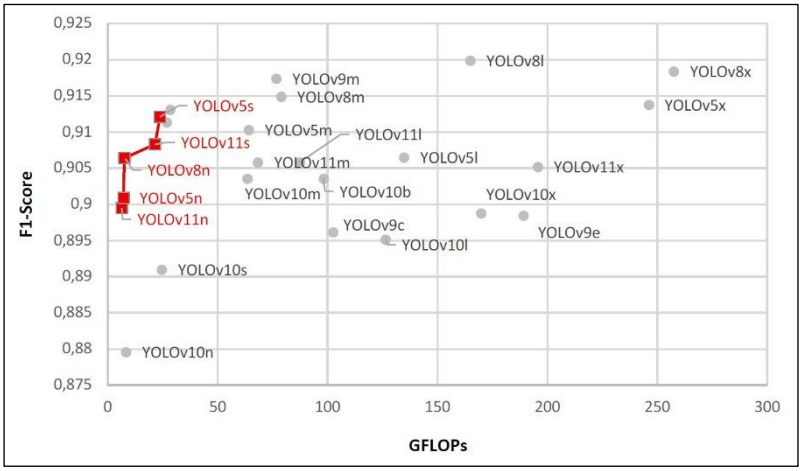

Fig. 4. Trade-off between F1 score and GFLOPs for different YOLO models - study on 3D printing error detection (Source: Chart based on data from [2])

The YOLO models with a particularly good combination of low GFLOPs values and a good F1 score are marked in red in the figure. These are the following five models:

Table 1: YOLO models suitable for this work (Source: Table based on data from [2])

Model	Precision	F1-score	mAP@0.5	mAP@0.5:0.95	FLOPs (G)	convergence (epochs)
YOLOv11n	0.8928	0.8995	0.8166	0.5098	6.4	81
YOLOv11s	0.9021	0.9083	0.8308	0.5361	21.6	44
YOLOv8n	0.9067	0.9063	0.8051	0.4947	8.1	66
YOLOv5n	0.8991	0.9009	0.7925	0.4795	7.1	55
YOLOv5s	0.9093	0.9121	0.8132	0.5101	23.8	55

The results shown in Table 1 compare different YOLO models based on key metrics such as precision, recall, F1 score, mAP@0.5, mAP@0.5:0.95, FLOPs and epoch. A differentiated analysis shows that the YOLOv11n, YOLOv11s and YOLOv8n models in particular have significant advantages over the older YOLOv5 variants. Firstly, it is striking that YOLOv11 consistently achieves very strong results: with an F1 score of 0.9083 and an mAP@0.5-Wert of 0.8308, it outperforms both YOLOv8n and the YOLOv5 models. At the same time, with only 44 epochs, it achieves the fastest convergence in the comparison field, indicating significantly more efficient training dynamics. YOLOv11n, on the other hand, offers a particularly favourable ratio between computational effort and performance. With only 6.4 GFLOPs, it is the most resource-efficient model and still achieves high precision. YOLOv8n proves to be a strong compromise between the two v11 variants. In summary, it can be said that the YOLOv11 models (n and s) show significant progress over the YOLOv5 generation in terms of both efficiency and accuracy. While YOLOv11n combines minimal computational resources with solid accuracy, YOLOv11s impresses with excellent performance and fast convergence. In addition, YOLOv8n is a powerful alternative that stands out in particular for its high precision. For these reasons, only the YOLOv11n,

YOLOv11s and YOLOv8n models will be considered for further investigations and experiments, while the YOLOv5 variants will not be considered further.

3.2 Training the AI model

Training YOLO models is very time-consuming, depending on the size of the data set, and requires high GPU performance. For these reasons, a notebook on Google Colab is used to train the YOLO models in order to utilise higher GPU performance. For preliminary tests with the small data set, the NVIDIA T4 graphics card with 16 GB of RAM is used for preliminary tests with the small data set. Google Colab also offers a uniform and user-friendly interface for programming with Python 3.11. The following factors influence the training of a YOLO model: batch size, number of epochs, training data (data set), image size (scaling), the choice of optimiser, and the availability of graphics memory. The batch size specifies how many images are processed per iteration. Instead of adjusting the weights of the NN after each individual image, the gradients are first averaged across all images in an iteration and then a joint adjustment is made. This allows training to be more efficient and stable, especially because it can be parallelised more easily [12]. An epoch describes a complete pass through the entire training data set by the neural network [14]. Only after several epochs does the model steadily improve its learning behaviour. An optimiser is a method that adjusts the weights of a neural network (NN) during training in order to minimise the error function. In addition to simple gradient descent, advanced optimisation algorithms such as AdaGrad, RMSProp or Adam are often used, which accelerate and stabilise training through adaptive learning rates and additional parameters [14]. For the preliminary tests with the small data set, the standard batch size of 16 was initially adopted. In addition, the number of epochs was set to 60, and the image size was 640 × 640 pixels. No optimiser was used in this case. A few lines of Python code install the Ultralytics package in the Google Colab notebook so that training can begin. After training the three YOLO models (YOLOv8n, YOLOv11n and YOLOv11s), the Python script is adopted based on open-source code provided on the internet [22] and expanded to include optical output.

3.3 Evaluation of results

The three models were trained using the same data set and examined using training and validation curves in terms of losses and recognition parameters (precision, recall, mAP). The results show clear differences in generalisation ability and recognition accuracy. Although the YOLOv8n model (see FIGURE 5) achieves a steady and clean decrease in training losses (box, Cls and DFL loss), it shows significant fluctuations in validation losses. In particular, the classification and DFL loss in the validation set remain at a high level. This discrepancy indicates overfitting: the model learns the training data very well, but has limited ability to generalise to new data. The recognition performance confirms this finding: Precision and recall stabilise in the range of ~0.18-0.20, while mAP values remain at a very low level (mAP50 j 0.08, mAP50-95 j 0.027). This means that YOLOv8n only achieves limited accuracy in error detection.

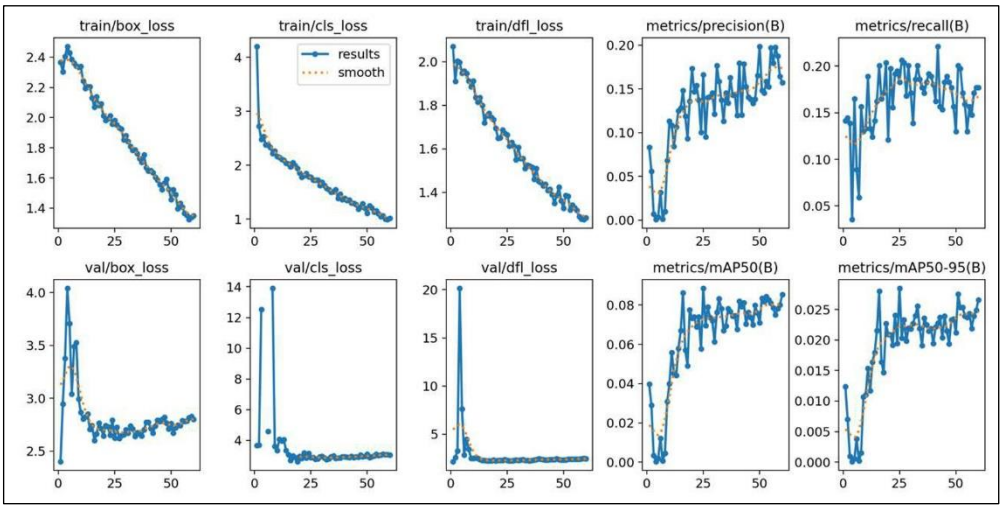

Fig. 5. Results from YOLOv8n for the preliminary test (Source: Own representation, automatically generated by Ultralytics)

The YOLOv11n model (see Fig. 5) shows improved stability in validation losses in comparison. Both the classification and DFLLoss are more consistent and remain at a lower level than with YOLOv8n. After training the YOLO models version 11 (YOLOv11n and YOLOv11s), a minimal increase in accuracy was achieved. To improve the recognition quality, the data set used in the preliminary test was expanded. The additional data base also comes from the open source source Roboflow [23]. The new data set contains a total of 5793 images The new data set contains a total of 5793 images (without augmentation) and is divided into 70% training data (4056 images), 20% validation data (1161 images) and 10% test data (576 images). With this data set, the YOLO model is designed to detect the most common errors in 3D printing. The error classes included are: blobs, cracks, overextrusion, spaghetti errors, stringing and under-extrusion. The distribution is as follows: “spaghetti” (1421 images; approx. 24.5%), “overextrusion” (981 images; approx. 16.9%), “cracks” (978 images; approx. 16.9%), “stringing”(930 images; approx. 16.0%), “underextrusion” (950 images; approx. 16.4%) and “blobs” (554 images; approx. 9.6%). It is noticeable that the total number of classified images (5814) differs slightly from the total number (difference: 21 images). This indicates that a small portion of the data cannot be clearly assigned to an error class or that several error classes have been identified on a few images. The class distribution shows slight imbalances. While spaghetti accounts for the largest share at 24.5%, blobs are the least common at 9.6%. The remaining classes are relatively close together at around 16-17%. As part of the data set preparation, standardised pre-processing steps were first applied. All images were automatically corrected for orientation (Auto-Orient) according to their recording direction as stored in the metadata. They were then scaled to a uniform image size of 640 × 640 pixels, with the original aspect ratios adjusted to the target format by stretching. In addition, systematic data augmentation was performed. Three variants were generated for each original training image, using different transformation operations combined stochastically. These include horizontal and vertical reflections, random image cropping with a scaling of up to ±15%, and rotations in the range from -215° to +15°. In addition, minor image noise was added, affecting up to 1% of the pixels. The augmentations increased the number of training images to 12,168 (4,056 × 3). The new dataset after augmentation thus comprises a total of 13,905 images, which are distributed in a ratio of 88% / 8% / 4% across the training, validation and test sets.

Following the dataset expansion, the YOLO models are now being trained with the new dataset.

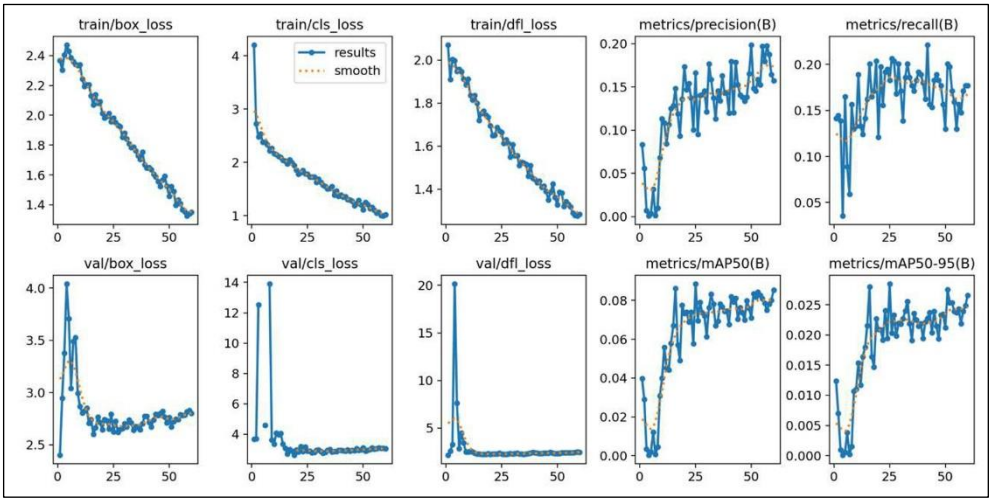

Fig. 6. Results from YOLOv8n with the new dataset (Source: Chart automatically generated by Ultralytics)

The training and validation curves show a consistent and stable learning process of the model. The training losses (box loss, classification loss, distribution focal loss) decrease continuously over the epochs and run almost monotonically decreasing. A comparable behaviour can also be observed in the validation losses, which indicates that the model not only adequately represents the training data, but also achieves good generalisation to the validation data. There are no signs of overfitting in the curves shown. The evaluation of the recognition performance parameters illustrates the effectiveness of the training. Precision increases rapidly and stabilises towards the end of training at values of around 0.9. Recall follows a similar trend and reaches final values in the range of 0.85 to 0.9. While the mAP reaches, the stricter metric mAP@[0.5:0.95] stabilises in the range of 0.7 to 0.8. These results show that the model not only detects 3D printing errors with high confidence but also performs robustly under stricter overlap criteria.

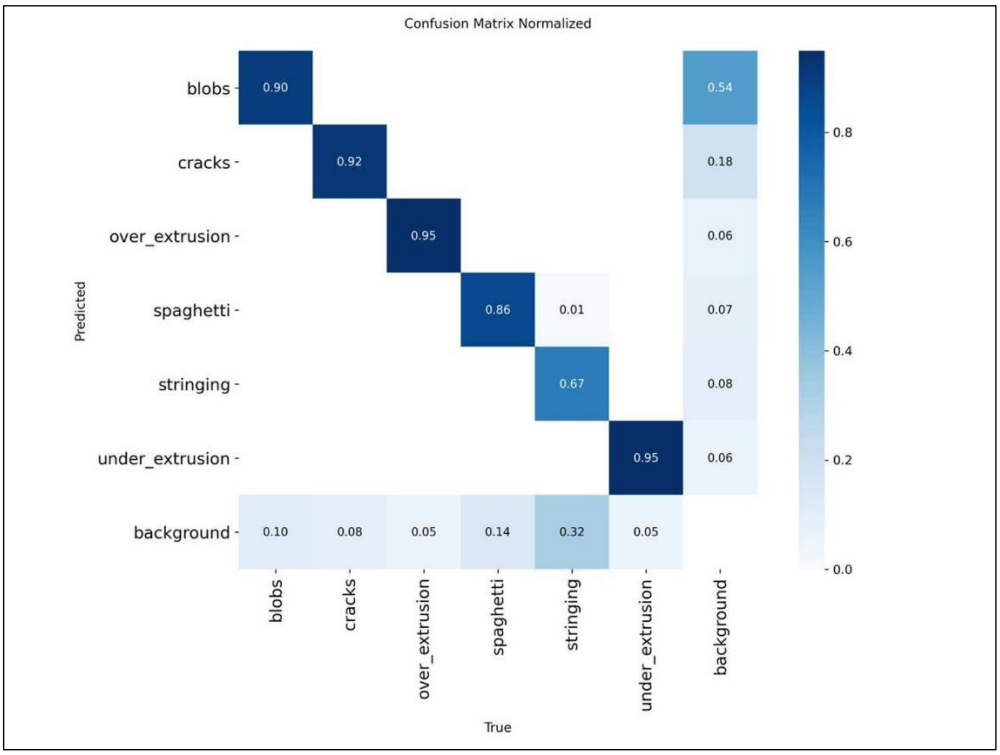

Fig. 7. Normalised confusion matrix for evaluating the classification performance of the trained model (Source: Chart automatically generated by Ultralytics)

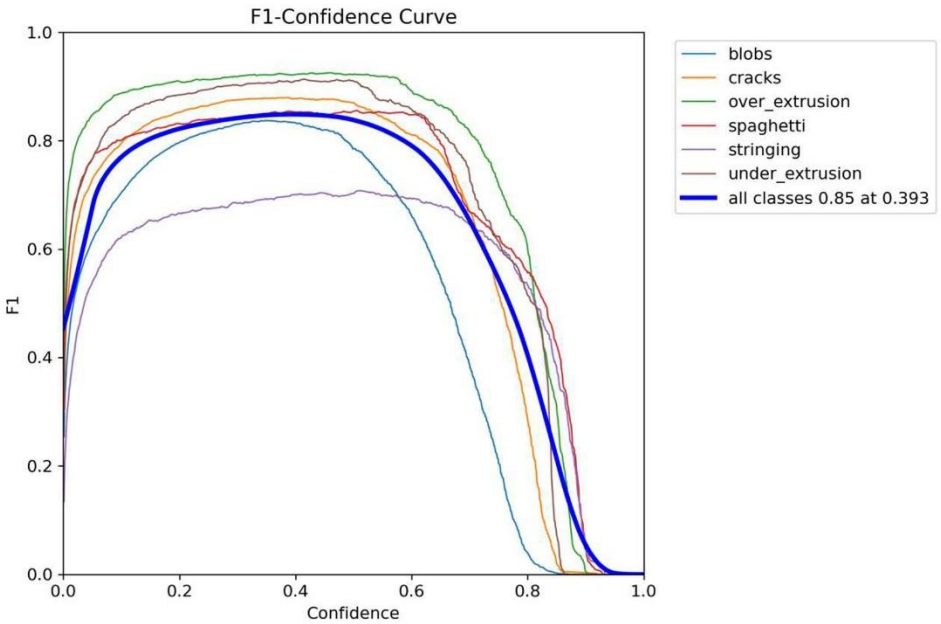

Fig. 8. F1 score of the different classes across different thresholds (Source: Chart automatically generated by Ultralytics)

Fig. 7 and 8 show that the model has high recognition performance, especially for clearly distinguishable defects such as “overextrusion” and “underextrusion”, which are classified very reliably both in the confusion matrix with hit rates of 0.95 and in the F1 curves with values above 0.9. Cracks and blobs also achieve high accuracies (0.92 and 0.90, respectively) and stable F1 values. However, performance is weaker for finer defect structures. “Spaghetti” still achieves a good detection rate (0.86), but shows individual confusions with background and “stringing”. Particularly problematic is the “stringing” class, which is often confused with the background in the confusion matrix background (0.32) and achieves significantly lower values (0.630.7) in the F1 confidence curve. The F1 confidence curve also indicates that the model achieves the best overall performance at a moderate confidence threshold of 0.393 (F1 = 0.85). This illustrates that choosing too high a threshold would greatly reduce the recognition rate.

3.4 Results of the system evaluation

In this experiment, the analysis and evaluation of the results are performed based on the analysis of the Ultralytics training. This allows for a direct comparison between the results during training and the actual results in practice. The system evaluation reflects the performance of three YOLO models (YOLOv8n, YOLOv11n and YOLOv11s) in detecting two typical types of 3D printing errors: spaghetti errors and stringing errors. The subsequent accuracy classification also correlates with the completeness of the detection. It can be concluded that YOLOv8n performs most robustly under different conditions. It delivers stable results in more scenarios and breaks down less frequently than YOLOv11n or YOLOv11s. The latter two models show very good results in some cases, but are more dependent on certain combinations of filament colour and lighting and are therefore less universally applicable. Across all models, it can be observed that spaghetti errors are detected more consistently and better overall than stringing errors. The lighting conditions plays a decisive role. All models show that bright and medium lighting produce the best results, while dark conditions are consistently problematic and sometimes lead to complete failure of detection. The results under “medium brightness” are generally comparable or even better than under bright lighting.

3.5 Discussion of the results

The results show that the tested YOLO models are fundamentally capable of detecting typical 3D printing errors such as spaghetti and stringing errors. Especially under bright and medium lighting conditions in the laboratory and with certain filament colours (especially green and white), the models deliver very high values in precision, recall and F1 score. This means that the existing system can, in principle, be used for practical applications. However, the results also indicate limitations for practical application. Dark lighting conditions in particular lead to significant performance drops across all models, and in some cases even to a complete failure of error detection. There is also a dependency on filament colour: while lighter or high-contrast filaments such as green and white are reliably detected, black filament causes problems in many scenarios. In practice, this means that the system is best used in well-lit printing environments with light-coloured or high-contrast materials. Several approaches for future optimisation can be derived from these findings. Firstly, technical adjustments could be made to the detection systems, for example by using additional or spectrally optimised lighting, infrared cameras or adaptive image processing to enhance contrast even in difficult conditions. In addition, an autofocus lens would be useful. During the tests, it was often noticed that the error could be located in different places on the print bed and that, due to the movement of the print head during the

printing process, the camera focused on the print head, which meant that the actual error could not be optimally detected by the model. Open questions remain, particularly with regard to the generalisability of the results. It is unclear how well the models perform with other printers, filament types or error characteristics that were not included in the current test. It also remains unclear how stable the detection is during longer printing times or in industrial environments with changing lighting, dust or vibrations.

4 Conclusion

The aim of this work was to develop an AI-supported, automated error detection system for 3D printing and to systematically evaluate its performance. The developed system represents a first step towards further work and developments in the field of visual KIF error detection and correction in 3D printing at Bochum University of Applied Sciences. Initially, preliminary tests were carried out with selected YOLO models, which were chosen and further developed on the basis of existing studies. The evaluation of the initial results showed that the data set needed to be expanded in order to improve detection performance. Following this adjustment, the AI model was successfully implemented and optimised on the university's hardware. In addition, the start-stop mechanism was automated and a Telegram bot was integrated to ensure a user-friendly and practical application. The investigations have shown that the combination of suitable data set creation, targeted model training and hardware optimisation provides a robust basis for real-time error detection. The practical suitability of the system was demonstrated by its successful implementation on a 3D printer and Raspberry Pi 5 was confirmed. The automation functions in particular highlight the practical benefits for both industrial and private applications. The research questions regarding recognition accuracy, the influencing factors of the data set and the automation possibilities were answered. It became clear that the quality, diversity and size of the data set have a decisive influence on the performance of the AI model. External conditions such as stable lighting or improved camera technology also contribute significantly to recognition performance. In addition, it was found that optimising the hardware and Python code significantly increased the efficiency of the system. Overall, it was demonstrated that the use of deep learning models such as YOLO for 3D printing defect detection enables high reliability while simultaneously conserving resources. Despite the results achieved, there are many opportunities for further development. A key challenge remains the expansion and diversification of the data sets in order to achieve even greater robustness against different types of errors and printing conditions. Similarly, the investigation of alternative or hybrid AI models, such as transformer-based approaches, offers a promising research avenue. On a practical level, the integration of the developed system into industrial manufacturing environments opens up great potential. Linking it to production control systems or IoT platforms could enable a fully automated monitoring, quality assurance and control system. In the long term, it also seems sensible to develop self-learning systems that continuously adapt to new scenarios during operation, thereby gradually increasing their performance.
In summary, the work shows that AI-based fault detection in 3D printing offers great potential for increasing efficiency and quality assurance. The results achieved form a robust basis for future research and underline the relevance of the topic for both science and industrial applications.

References

1. A. Grunwald and R. Hillerbrand, Handbuch Technikethik (J. B. Metzler Verlag, Berlin, 2021)
2. A. R. Sani, A. Zolfagharian, A. Z. Kouzani, Automated defects detection in extrusion 3D printing using YOLO models, Journal of Intelligent Manufacturing, vol. **37**, 351-371 (2024). https://doi.org/10.1007/s10845-024-02543-8
3. J. Wei, A. As'arry, K. Anas Md Rezali, M. Zuhri Mohamed Yusoff, H. Ma, K. Zhang, A Review of YOLO Algorithm and Its Applications in Autonomous Driving Object Detection, IEEE Access, vol. **13**, 93688-93711 (2025). https://doi.org/10.1109/ACCESS.2025.3573376
4. A. Pusch, N. Haverkamp, 3D-Druck für Schule und Hochschule: Konstruktion von naturwissenschaftlichem Experimentiermaterial mit Best-Practice-Beispielen, (Springer Spektrum, Berlin Heidelberg, 2022). https://doi.org/10.1007/978-3-662-64807-0
5. H. K. Dave, J. P. Davim, Fused Deposition Modeling Based 3D Printing. Materials Forming, Machining and Tribology (Springer International Publishing AG, 2021)
6. S. S. Haque, Minimizing Stringing Issues In FDM Printing (2020). https://doi.org/10.13140/RG.2.2.35536.74247
7. E. Cenedese, 3D Printing Anomaly Detection: Implementation of a ML system using YOLOv5 and EfficientNet-Lite, Master Thesis, University of Torino, Torino, Italy, 2022
8. S. Kwon, D. Hwang, Understanding and Resolving 3D Printing Challenges: A Systematic Literature Review, Innovations in Manufacturing Processes and Systems for Sustainable Practices, vol. **13**, 1772, (2025), https://doi.org/10.3390/pr13061772
9. K. Tagirova, A. Vulfin, P. Kachkaeva, Industrial 3D Printing Quality Control System Based on Machine Vision", 2025 International Russian Smart Industry Conference (SmartIndustryCon), IEEE, 990-995, Sochi, Russian Federation, 2025. https://doi.org/10.1109/SmartIndustryCon65166.2025.10986006
10. W. Ahmed, A. El Hassan, E. Zaneldin, The effect of embedded blobs irregularities on the characteristics of 3d printed panels with dissimilar materials, Solid Mechanics Symposium, University of Sherbrooke, Department of Mechanical Engineering, 2023. https://doi.org/10.17118/11143/21143
11. A. Nischwitz, M. Fischer, P. Haberäcker, G. Socher, Bildverarbeitung: Band II des Standardwerks Computergrafik und Bildverarbeitung (Springer Fachmedien, Wiesbaden, 2020). https://doi.org/10.1007/978-3-658-28705-4
12. M. Werner, Digitale Bildverarbeitung: Grundkurs mit neuronalen Netzen und MATLAB®-Praktikum (Springer Fachmedien, Wiesbaden, 2021). https://doi.org/10.1007/978-3-658-22185-0
13. M. Flasinski, Introduction to Artificial Intelligence (SpringerLink Books, Springer, Wiesbaden, 2016). https://doi.org/10.1007/978-3-319-40022-8
14. H. Ernst, J. Schmidt, G. H. Beneken, Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis - eine umfassende, praxisorientierte Einführung, 7th edition (Springer Vieweg, Wiesbaden, 2020). https://doi.org/10.1007/978-3-658-30331-0
15. Fabian Dillenhöfer, Entwicklung einer KI-Objekterkennung für technische Zeichnungen im Maschinenbau, Ph.D. thesis, Dortmund University, Department of Mechanical Engineering (2023)

16. “Roboflow Universe”, source for image detection, 2025. https://universe.roboflow.com/
17. KOKI, 3D Print Save Dataset. Roboflow Universe, Open Source Dataset (2025) https://universe.roboflow.com/koki-fx5bo/3d-print-save-ziw0z
18. Modelltraining mit Ultralytics YOLO, documentation (2025) https://docs.ultralytics.com/de/modes/train/
19. Ultralytics YOLO Dokumentation (2025) https://docs.ultralytics.com/de/
20. Raspberry Pi 5 Benchmarks and Stress Tests, forum thread (2025) https://forums.raspberrypi.com/viewtopic.php?t=363656
21. Raspberry Pi 5 vs. Orange Pi 5 Plus vs. Rock 5 Model B, manufacturer forum thread (2025) https://picockpit.com/raspberry-pi/raspberry-pi-5-vs-orange-pi-5-plus-vs-rock-5-model-b/
22. Evan Juras, Train-and-Deploy-YOLO-Models, Github project (2025) https://github.com/EdjeElectronics/Train-and-Deploy-YOLO-Models
23. syLucauc, 3d-printing-failure-detection Dataset. Roboflow Universe (2025) https://universe.roboflow.com/sylucauc/3d-printing-failure-detection