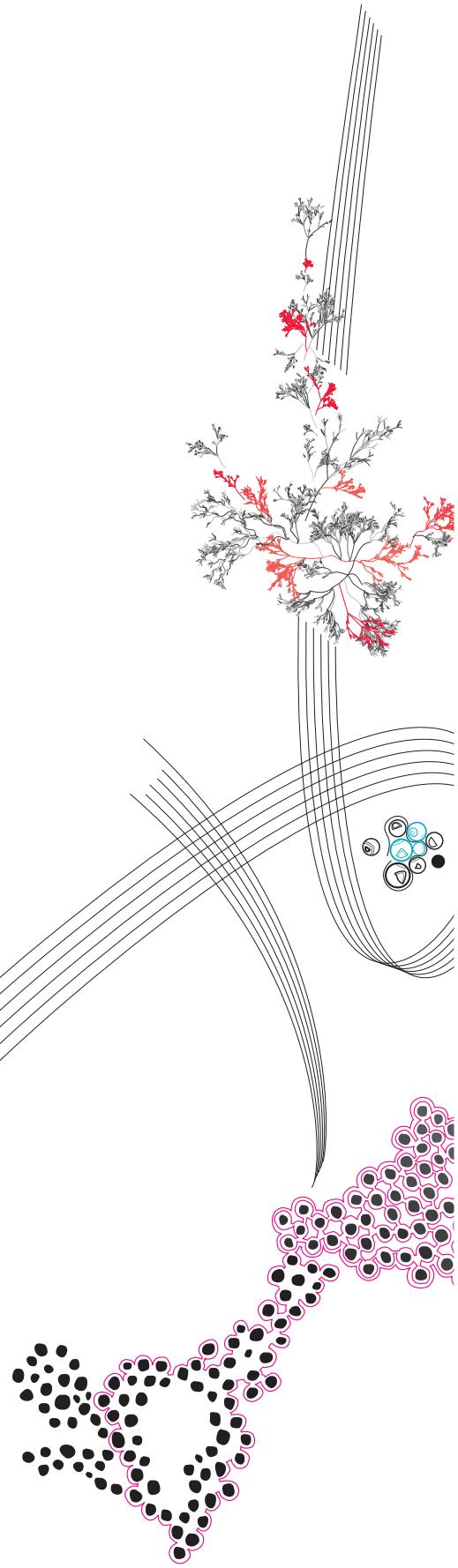


MSc Computer Science
Final Project

# Improved Low-contrast Spaghetti Defect Detection for FDM Printers

Bart Leenheer

Supervisors:
Dr. Ir. Alex Chiumento,
Prof. Dr. Mariëlle I.A. Stoelinga,
Ir. Reinier Stribos (Fraunhofer Innovation Platform for Advanced Manufacturing at the University of Twente)

April, 2025

Department of Computer Science
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

**Abstract**

Spaghetti defects are a common failure in Fused Deposition Modeling (FDM) printing, often caused by object detachment from the print bed, missing support structures, or objects falling over. This often results in extruded filament failing to attach to a previous layer and instead freely curl up, forming balls that look like spaghetti. These defects are mostly unrecoverable, resulting in wasted time and materials. While computer vision-based anomaly detection exists, such as the one built into the Bambu Lab X1 Carbon and Obico, they lack detection performance when the contrast between the filament and the background is minimal, e.g. when using black filament against a dark background.

To combat this problem, this work explores the influence of including a Low-Light Image Enhancement (LLIE) algorithm in the preprocessing steps before applying the anomaly detection algorithm. To do so, this work makes use of a novel approach called CoLIE and combines it with the latest YOLO11 (by Ultralytics) model for spaghetti object detection. This proposed method is evaluated on a Colour dataset, Black dataset, and a third publicly available dataset to show how well it generalizes to unseen circumstances. To establish a baseline, these tests are also performed using an open-source approach, Obico.

Results show that CoLIE does not significantly improve detection performance and, in some case, even reduces it. This is likely due to overenhancement causing the image to lose important features, resulting in reduced precision. However, YOLO11 does show a significant increase in performance over the YOLOv2-based Obico implementation, even on a publicly available dataset, achieving a 17.5% (non-Context-based Low-light Image Enhancement (CoLIE)) and 2.4% (CoLIE) higher F1-score. This work also introduces a Print Failure Stopping Metric (PFSM), to evaluate the theoretical performance of a real-world system. The results show that modern Deep Learning models, specifically YOLO11, are highly effective in the detection of spaghetti defects in FDM printing, without the need for additional preprocessing.

Further research should explore hardware-based solutions, such as depth-sensing cameras or Lidar, to improve detection performance by including contrast-independent depth information. Additionally, alternative Machine Learning approaches, such as autoencoders or differential imaging, could be further investigated to see the impact on low-contrast detection performance.

*Keywords*: 3D Printing, Fused Deposition Modeling (FDM), Spaghetti Defect Detection, YOLO11, Low-Light Image Enhancement (LLIE), CoLIE, Computer Vision, Machine Learning, Deep Learning, Anomaly Detection

# Acknowledgments

I would like to express my sincere gratitude to everyone who supported me throughout the process of writing this thesis.

First off, I want to thank Dr. Ir. Alex Chiumento for his guidance and support as my main supervisor at the University of Twente. His feedback and involvement helped me steer my thesis in the right direction. I would also like to thank Ir. Reinier Stribos for the daily guidance at Fraunhofer. His insights, advice and thorough feedback helped my work take shape from start to finish. Additionally, I would like to thank Prof. Dr. Mariëlle I.A. Stoelinga for being my second UT supervisor and for taking the time to read and evaluate my thesis as part of the committee.

Furthermore, I am very grateful for all the support I've received from my girlfriend Sophie Weidmann. She helped me throughout this whole process and gave me guidance and energy through the more difficult periods. Besides this, her good ideas when sparring have contributed greatly to this work, which resulted in a significant boost in quality of my thesis. Lastly, her extensive feedback helped me dot the i's and cross the t's to form my final version.

A big thank you to my parents for their continuous support and motivation, not just during this thesis, but throughout my entire study.

Finally, thank you to all my friends for being there for me throughout this journey and for helping me take my mind off work when I needed it.

# Contents

# Chapter 1

# Introduction

Additive Manufacturing (AM), commonly known as 3D printing, has increased in popularity for both rapid prototyping and final product manufacturing [1]. Fused Deposition Modeling (FDM) is one of the most common types of AM, widely used due to its cost-effectiveness, ease of use, and variety of supported material types. In FDM, the material is heated and extruded through a nozzle, depositing filament layer-by-layer. Here, each consequent layer adheres to the previous one when the material cools down and solidifies. However, despite the increase in popularity and improvements in FDM technology, print errors still occur, affecting the quality of the finished print, resulting in wasted material or time [2, 3].

One of the most common anomalies is the spaghetti defect [4], named after the ball of curled up filament, resembling a plate of spaghetti. This defect is caused by filament failing to adhere to the previous layer, resulting in filament being printed in the air, causing it to curl up and form spaghetti. This can occur due to a variety of problems, such as detachment from the print bed, missing supporting structures, or parts falling over. In most cases, when spaghetti defects occur, the print is unrecoverable. This means that the print requires a complete restart, resulting in wasted time and materials. In severe cases, the curled up filament attaches to the print head and starts to build up, causing a so-called "blob of death". This can result in a damaged print head, requiring costly repairs.

Multiple detection models exist that are able to detect spaghetti [4, 5, 6]. For example, the printer used in this study – the Bambu Lab X1 Carbon – uses a built-in camera for vision-based spaghetti defect detection [7]. However, this detection system still struggles with dark-coloured filament due to the low contrast between the filament and background. This was confirmed by printing multiple geometries using black filament and observing – for all three detection sensitivities – whether the printer stopped when a spaghetti defect occurred.

Low-Light Image Enhancement (LLIE) is one way to improve contrast in images by enhancing the brightness in darker areas of an image while preserving important details [8]. While research exists in both anomaly detection for FDM printing and LLIE, no research was found that investigates the integration of LLIE into anomaly detection for FDM printing. Besides this, no existing research has covered the challenge of detecting anomalies when using black filament, presenting a research gap that forms the primary motivator for this work.

To address the problem of low-light, or low-contrast, anomaly detection in FDM printing, this work proposes a novel anomaly detection method, incorporating LLIE within a YOLO11 detection pipeline. A recently developed LLIE approach, Context-based Low-light Image Enhancement (CoLIE), was chosen to fulfill this task, as it showed promis-

ing results in domain-independent image enhancement. For the defect detection aspect, YOLO11 is used, as existing literature showed the effectiveness of previous versions in anomaly detection for FDM printers. By combining these two techniques, this work aims to combat the challenge of defect detection when using darker filament.

The effectiveness of the proposed approach is evaluated on three test sets: one with colour Bambu Lab prints, to test the overall effectiveness, one with black Bambu Lab prints, to test the low contrast effectiveness, and a publicly available dataset, which uses a different printer, to test the generalizability. These results are compared to an open-source approach, Obico[1], to form a baseline and show the effectiveness of the proposed method in this work in relation to what is already available. This evaluation is done using a number of metrics based on whether detection is performed accurately, as well as a newly introduced metric, Print Failure Stopping Metric (PFSM), that evaluates how the system would perform in a theoretical real-world implementation where the printer would stop after a set number of consecutive frames containing a predicted anomaly. More details can be found in Section 3.3.1.

This work contains a glossary and an acronym list to provide a short explanation for given terms and to provide an overview of the used acronyms. These can be found on page 69.

## 1.1 Research Questions

The main question that this work tries to answer is:

**RQ1** *How can computer vision techniques be effectively applied to detect spaghetti anomalies in FDM printing, particularly in low-contrast conditions between the filament and background?*

To support this question, the following sub-questions have been devised:

RQ1.1 How can YOLO be optimized for detecting spaghetti anomalies in FDM printing?

RQ1.2 What effect does low-light image enhancement have on spaghetti anomaly detection?

RQ1.3 How does the proposed model compare to Obico in terms of performance on self-collected and publicly available datasets?

RQ1.4 How do different stopping thresholds impact the trade-off between true positives, false positives, and detection delay?

[1] Available: https://github.com/TheSpaghettiDetective/obico-server

# Chapter 2

# Background

This chapter contains background information that is relevant to understand the work that has been done. It contains an explanation of the used 3D printing technique, Machine Learning and several Machine Learning techniques. Next, the object detection model and LLIE algorithm are described, including a description of how they work and why they were chosen for this work. This section will end with an overview of related work in FDM anomaly detection and LLIE.

## 2.1 Additive Manufacturing

Additive Manufacturing (AM) is an all-encompassing term to describe 3D printing techniques, where computer-aided models are turned into actual objects layer-by-layer, without the need for traditional fabrication techniques such as molding, machining, or tooling. Initially, this production method was primarily used for rapid prototyping, due to its manufacturing flexibility and ability to quickly transform a design into a physical product. However, it is now increasingly used for the production of final products [1].

The term AM encompasses a variety of methods that differ based on materials or assembly techniques. For example: robocasting for ceramic materials [9, 10], Powder Bed Fusion (PBF) for metals such as steel [11], or FDM, also known as Fused Filament Fabrication (FFF), which is commonly used for a wide range of polymers [12].

Robocasting is an AM technique where material is extruded in a layer-by-layer fashion onto a build plate. Although it uses a similar approach to FDM – as explained in Section 2.1.1) – the two processes differ in the used material types and the solidification of these materials. In FDM the extruded material solidifies as it is cooling down, whereas in Robocasting the material retains its shape immediately after extrusion due to the structure of the material. As a result, robocasting is generally used for ceramics or other high-density parts [9].

PBF works by spreading a layer of powder on the build plate which is then heated in specific areas by a laser or similar precise heat source. This heat solidifies the powder in the specified areas, which results in a solid object remaining when the rest of the powder is removed. This process is repeated for each layer and, as the layers are thin, the heated parts stick to the underlying layer [13]. This approach can be used to fabricate high-precision metallic parts that can be used in all sorts of applications requiring high durability [6].

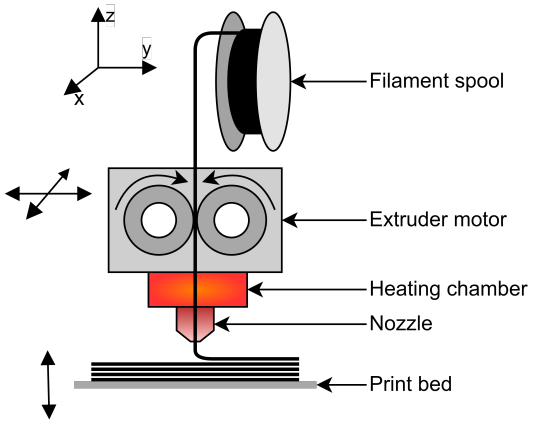



FIGURE 2.1: Schematic of an FDM printer. (Actual configurations may vary)

### 2.1.1 Fused Deposition Modeling

Fused Deposition Modeling (FDM) is an AM approach where filament is melted and extruded to assemble objects layer-by-layer [14]. The material is usually housed on a filament spool, which is fed into a heating chamber, where the filament is melted. The liquid material is then deposited and adheres to the previous layers, after which it quickly solidifies to form a desired shape.

To perform this process, an FDM printer consists of a movable print head containing a driver motor for the filament, a heating chamber, and a nozzle to extrude the material [14]. Figure 2.1 depicts a schematic structure a typical FDM printer. Together with the print bed, the print head is able to move in the x, y, and z axis in a way that the nozzle is able to print in three dimensions. The movement of the components differs between various printers. For example, in the Bambu Lab X1 Carbon printer (see Section 2.1.3), the print head moves in the x and y axis, while the bed moves in the z axis.

As mentioned before, FDM allows for printing a variety of polymers. Commonly used materials are Acrylonitrile Butadiene Styrene (ABS) and Polylactic Acid (PLA) as they both have a good balance between strength and ease of printing [15]. For this project PLA is used, as it is simple to use and cost-effective.

### 2.1.2 FDM Defects

Despite improvements in modern FDM printers, a number of defects can occur, impacting the resulting 3D print quality. Table 2.1 provides an overview of common defect categories, identified by Günaydın and Türkmen [2], and their corresponding visible defects, most of which are listed on the website of simplify3d[1]. The most common defects are spaghetti and stringing [4]. Stringing defects show up as small strings attached to the printed object, as depicted in Figure 2.2, and they occur because the printer does not completely stop extruding while moving from one area to another, which can be caused by incorrect printer settings, insufficient cooling, or an excessively high temperature [16]. Fortunately, stringing defects often do not cause a print to fail, but can be fixed using post-processing. Unfortunately, when a print experiences spaghetti defects, it is most likely unusable and therefore requires a restart, resulting in wasted time and materials, increasing costs.

[1] Available: https://www.simplify3d.com/resources/print-quality-troubleshooting/

| Category | Visible defects |
|---|---|
| Misalignment of the print platform. | • Print not sticking to platform<br>• Warping<br>• Lack of fine details<br>• Spaghetti due to lack of adhesion |
| Misalignment of the nozzle. | • Misaligned layers<br>• Missing layers<br>• Skewed print<br>• Shifted layers |
| Clogging of the nozzle, depletion of printing material, or disrupted material flow. | • Incomplete layers<br>• Incomplete print |
| Lack or loss of adhesion to the print platform. | • Spaghetti caused by detachment<br>• Warping |
| Vibration or shock (from the printer or another source). | • Uneven surface or edges<br>• Blobs |
| Inaccurate adjustments of printer settings. | • Stringing<br>• Spaghetti |

TABLE 2.1: Defect categories and their visible flaws in FDM printing.

### Spaghetti

This work focuses on the spaghetti defects category, due to their frequency and significant impact on the printing process. These defects can occur due to a number of different erroneous settings or failures mid-print, for example: missing support structures, detachment from print bed, and parts falling over, as depicted in Figure 2.3. Consequently, there is no material present at the location of the print head, resulting in the printed filament lacking a surface to adhere to and forming the so-called spaghetti. This can even result in the printed filament spiraling back up to attach to the print head itself, and – as the print head keeps printing – forming a growing "blob of death" on the print head, as depicted in Figure 2.4. In the best case, it is just a time-consuming process to remove this blob. However, in the worst case, the blob cannot be fully removed, requiring costly printer repairs.

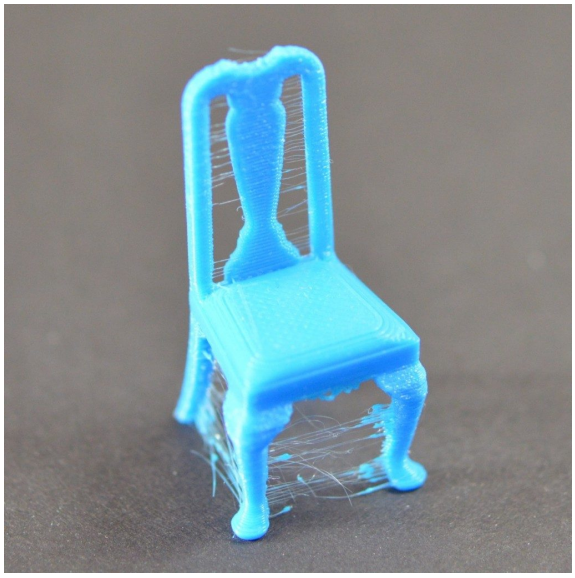


FIGURE 2.2: Example of a stringing defect. Adapted from [17].

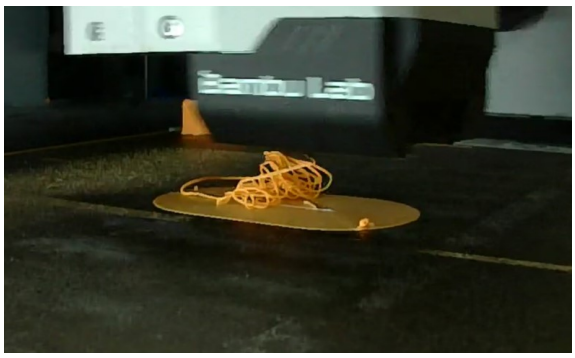


FIGURE 2.3: Example of a spaghetti defect, caused by the object falling over.

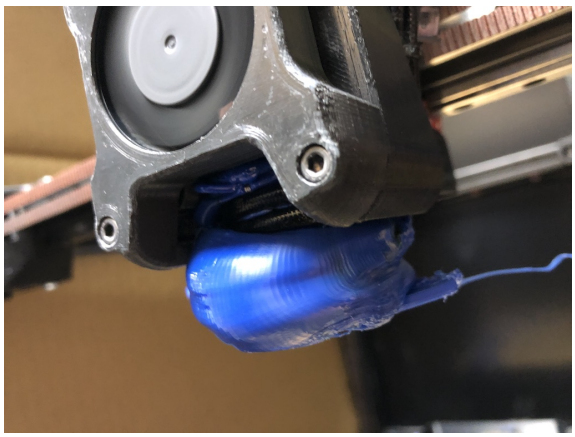


FIGURE 2.4: Example of a "blob of death", caused by filament attaching to the print head, forming a growing blob. Adapted from [18].

FIGURE 2.5: Example of a black filament print with spaghetti defect. Here the printer finished the print while the built-in detection was set to high sensitivity.

### 2.1.3 Bambu Lab X1 Carbon

This research makes use of the Bambu Lab X1 Carbon[2], which is an industry standard, easy-to-use FDM printer with a considerable number of features. These features include simple calibration through auto bed leveling, multi-colour printing, vibration compensation and extrusion compensation for increased smoothness, and a camera with built-in spaghetti defect detection [7] and timelapse function. For this research, the built-in camera is used for data collection, which will further be described in Section 3.2.1.

#### Built-in Spaghetti Defect Detection

The Bambu Lab X1 Carbon has a built-in spaghetti defect detection feature that can be activated directly on the printer. It has three different sensitivity options that correspond to the confidence threshold [7]: low, medium, and high. This built-in spaghetti defect detection feature is a primary motivator for this research, as it works well on colour filaments, but unfortunately lacks robustness for darker filament colours. Some preliminary testing has shown that the built-in spaghetti detection fails to detect issues with black filament, even when set to the highest sensitivity. This was tested observing multiple prints with black filament, including those using artificially induced spaghetti defects, as described in Section 3.2.1. During these prints, the detection sensitivity was varied between low, medium, and high, and the printer's behaviour was observed. In only a few cases, the built-in detection was able to recognize the spaghetti defect, but only after a significant delay, still resulting in a lot of material waste. However, in most cases, the printer failed to stop entirely when spaghetti occurred, requiring manual intervention. A concrete example is depicted in Figure 2.5, where the printer continued to print until it finished the machine code instructions of the last layer, even when set to the highest detection sensitivity. The issue is likely related to the low contrast between the black filament and the background or print bed.

## 2.2 Machine Learning

Machine Learning (ML) describes implementing "learning" in computers, where learning encompasses: the acquisition of new knowledge, the development of skills through instruction or practice, the organization of acquired knowledge into generalizable representations, and the discovery of new facts and theories through observation and experimentation [19].

[2] Available: https://eu.store.bambulab.com/products/x1-carbon

ML consists of a wide range of different methods, each focusing on different problems and using different techniques to solve them. These methods can roughly be grouped into categories such as supervised, unsupervised, semi-supervised, and reinforcement learning, referring to the way that these networks obtain their knowledge [20].

### 2.2.1 Supervised Learning

Supervised ML methods learn to map specific inputs to corresponding outputs, where the desired output can belong to a class (classification), or a continuous value (regression) [21]. To learn this mapping function from the input to the output, labeled training data is required; pairs of inputs and their respective outputs. Ideally, this mapping contains knowledge about the underlying patterns in the data and is able to accurately infer the desired output from unseen data. This category is called "supervised", as it requires external assistance to learn the correct mapping. Examples of supervised algorithms are Decision Trees, Naive Bayes and Support Vector Machine (SVM). An example application is medical diagnoses based on patient records.

### 2.2.2 Unsupervised Learning

Unsupervised ML methods learn relationships within data without access to labeled data [22]. For example, by learning the differentiating features within a dataset, they can divide data into a number of clusters. When providing unseen data points to these algorithms, they should be able to assign them to one of the clusters, based on similarities in their features. This type of ML is primarily used for clustering or dimensionality reduction. Examples of unsupervised algorithms are K-Means Clustering and Principal Component Analysis (PCA). An example application is feature extraction for object detection.

### 2.2.3 Semi-Supervised Learning

Semi-supervised learning falls somewhere in between supervised and unsupervised learning. It benefits from labeled data, but also uses unlabeled data during the training process [23]. The unsupervised part is used to understand underlying patterns in the unlabeled data, whereas the labeled data is used to make predictions. This category of ML methods is mostly used when the access to labeled data is limited or expensive, but the model can benefit from the larger amount of unlabeled data that is readily available. Examples of semi-supervised algorithms are Generative Adversarial Networks and Variational Autoencoders. An example application is image generation.

### 2.2.4 Reinforcement Learning

Reinforcement learning methods are used to determine how (virtual) agents should take actions in a virtual or physical environment based on their observations [24]. In this approach, an agent takes actions, receiving positive rewards for desired states, and negative rewards for undesired ones. The total reward of the simulation is calculated to evaluate the actions of the agent, and the decision making model is updated accordingly. This type of learning is useful for decision-making problems where outcomes depend on sequences of actions, rather than isolated inputs. Examples of reinforcement learning algorithms are Q-Learning and SARSA. An example application is a complex game, such as chess.

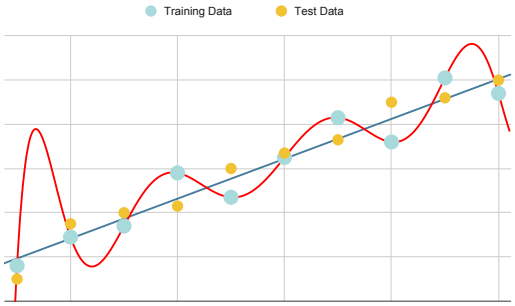



FIGURE 2.6: Illustration of overfitting. The red line represents an overfit model failing to generalize, while the blue line depicts a well-generalized model.

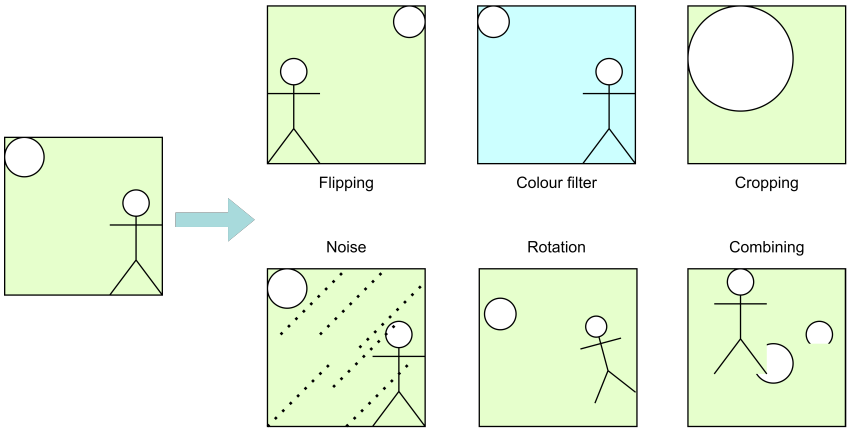



FIGURE 2.7: Examples of data augmentation techniques for image data.

### 2.2.5 Overfitting

Overfitting occurs when a model learns to represent the training data too closely, failing to generalize the underlying patterns [25]. Figure 2.6 illustrates this principle. The red line shows an overfit model that perfectly encapsulates the training dataset, but fails to generalize and therefore performs significantly worse on the test dataset. The blue line depicts a well-generalized model that captured the underlying patters of the training data. It obtains a similar accuracy on both the training and test datasets, showing its ability to perform well on unseen data.

### 2.2.6 Data Augmentation

A common way to combat overfitting is by increasing variation in the dataset [25]. While this can be done by collecting more data in various settings, this process is often time-consuming and costly. A different approach is data augmentation, which involves applying transformations to existing data to generate more variations. This process increases the model's robustness to handle small variations that can occur in unseen data.

In image processing, data augmentation methods include flipping, applying colour filters, cropping, introducing noise, rotating, and combining images [26], as depicted in Figure 2.7.

### 2.2.7 Evaluation Metrics

To quantify the performance of ML algorithms, certain evaluation metrics can be applied. These metrics provide a way to show the effectiveness of the evaluated model and can be easily compared to the metrics of other algorithms. Different metrics touch on different strengths or weaknesses of the algorithm, allowing for a proper evaluation of possible downsides of the model. Besides that, metrics can be tailored to the ML task at hand.

#### Classification

The basis of classification metrics is divided into four groups: True Positives (TPs), True Negatives (TNs), False Positives (FPs), and False Negatives (FNs) [27], where:

- TP: Number of instances where the model predicts the positive class correctly, i.e., $y_{\text{gt}} = 1 \wedge y_{\text{pred}} = 1$.
- TN: Number of instances where the model predicts the negative class correctly, i.e., $y_{\text{gt}} = 0 \wedge y_{\text{pred}} = 0$.
- FP: Number of instances where the model predicts the positive class incorrectly, i.e., $y_{\text{gt}} = 0 \wedge y_{\text{pred}} = 1$.
- FN: Number of instances where the model predicts the negative class incorrectly, i.e., $y_{\text{gt}} = 1 \wedge y_{\text{pred}} = 0$.

These values are often represented in a confusion matrix, which is a square matrix where rows represent true classes and columns represent predicted classes. This allows for a clear overview depicting the TPs (top left), TNs (bottom right), FPs (bottom left), and FNs (top right). An example confusion matrix with 100 positive and 100 negative samples is depicted in Figure 2.8. For classifiers that have more classes, a confusion matrix can depict whether there is a tendency for a specific class to be classified as another specific class, e.g. dogs being classified as cats. This might be caused by overlapping features, indicating the need for more delicate feature extraction to learn the differences between those classes.

These values in a confusion matrix can be used to formulate more general metrics that highlight different aspects and allow for easier human understandability and comparison to other models. Such metrics are accuracy, precision, recall, False Positive Rate (FPR), and F1-score [27], defined as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.3}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.4}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.5}$$

Here, accuracy is used to describe what percentage of classifications is correct. Similarly, precision describes what percentage of positive predictions is correct, while recall describes what percentage of actual positive instances are correctly classified. The FPR describes

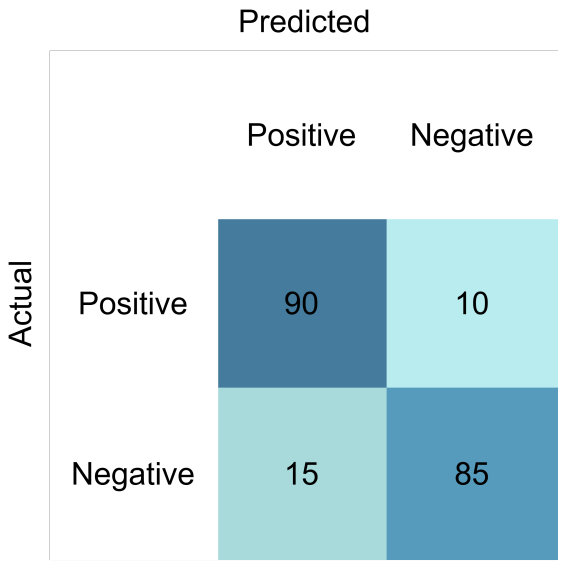



FIGURE 2.8: Example of a confusion matrix with 100 positive and 100 negative samples.

what percentage of actual negative instances is incorrectly classified. Finally, the F1-score provides a single score indicating the balance between precision and recall.

Besides this, certain plots – and the derived measurements from these plots – can be used to show the trade-off between metrics or limitations of the system, e.g. the Receiver Operating Characteristic (ROC) curve – allowing for the measurement of the Area Under Curve (AUC) [27] – and the Precision-Recall (PR) curve [28].

The ROC-curve plot compares the recall against the FPR at different thresholds, several examples of ROC-curves are depicted in Figure 2.9, showing the performance of a bad classifier, random classifier, okay classifier, and better classifier. A higher recall often goes paired with a higher FPR, as a higher recall is achieved by having fewer FNs, which can be achieved by lowering the threshold for positive classification, which in turn increases the rate of FPs. This effect can also be seen in the ROC plot, where higher FPRs show a higher recall. This indicates that an optimal balance between the two should be chosen based on the requirements for the classification application. For example, for medical diagnoses, a high recall is of critical importance, whereas false positives can be mitigated by further medical tests. Better classifiers show a steeper curve towards the top-left corner of the ROC-curve, which means that the recall is higher at lower FPR values. As the area under the ROC-curve is higher when the curve is steeper, the performance of a classifier can be measured by quantifying the AUC, as depicted in Figure 2.10, where a higher AUC indicates a more robust classifier.

The PR curve shows another trade-off in classifiers, this time between the precision and recall metric. As mentioned before, recall can be increased by lowering the threshold for positive classification, however, that also increases FPs, decreasing the precision. This effect can be seen in Figure 2.11, where a drop in precision can be observed for higher values for recall. For this metric, the optimal classifier achieves a curve that is as close to the top-right corner as possible, showing it can achieve both a high precision as well as a high recall. This curve can be used to calculate the Average Precision and Mean Average Precision, which are important metrics for tasks such as object detection.

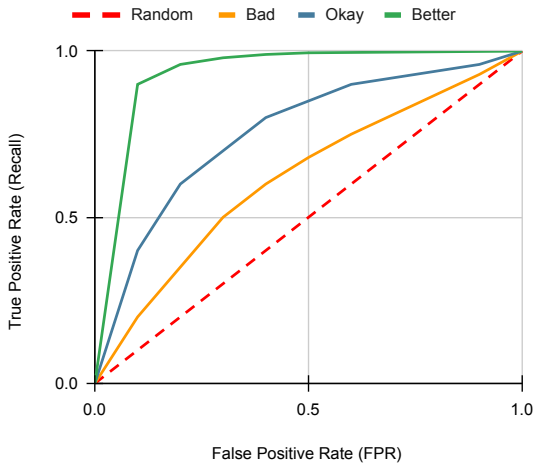



FIGURE 2.9: Example of several ROC-curves showing the trade-off between recall and FPR. The dotted line represents the ROC-curve when a truly random classifier is applied.

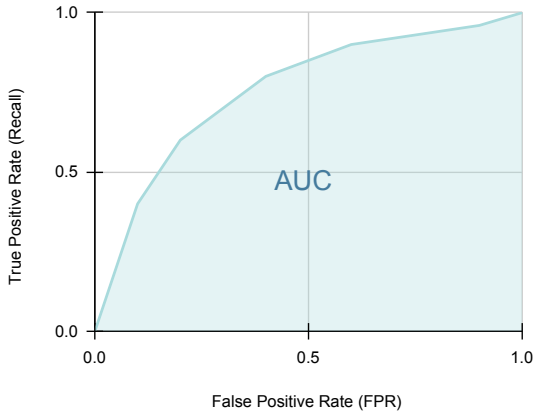



FIGURE 2.10: Example of the AUC, which represents the performance of a classifier across multiple thresholds. A larger area indicates a more robust classifier.

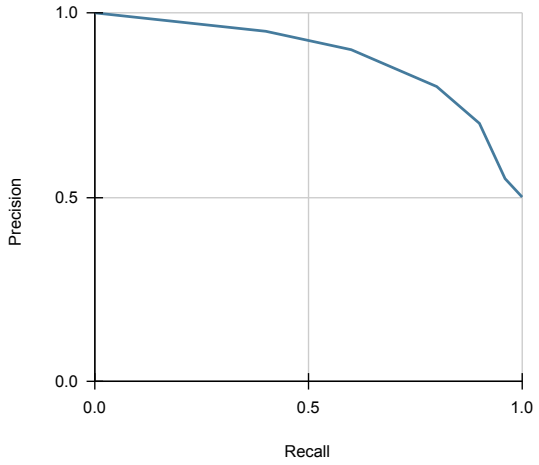



FIGURE 2.11: Example of a PR curve showing the trade-off between precision and recall.

### Object Detection

For object detection problems, the performance of the system is often evaluated using the Average Precision (AP), Mean Average Precision (mAP), and Intersection over Union (IoU) metrics [28]. The Average Precision (AP) for a single class is taken by obtaining a PR curve and calculating the area under the curve. The Mean Average Precision (mAP) is calculated by first obtaining the AP for each class and then taking the average. The IoU is a metric showing the overlap between the predicted bounding box and the ground truth bounding box and the mean IoU can be used as an indication of overall bounding box performance. The IoU is also used in the mAP@50 and mAP@50-95 metrics, where bounding boxes with an IoU lower than the threshold are filtered out and the mAP is computed over the remaining boxes. Figure 2.12 illustrates an example of the intersection and union between two bounding boxes. These metrics are calculated as follows:

$$\text{AP} = \int_0^1 \text{Precision}(r) d\text{Recall} \tag{2.6}$$

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^{C} \text{AP}_c \tag{2.7}$$

$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}} \tag{2.8}$$

Where:

- Precision($r$): The precision value corresponding to a given recall value.
- $C$: Total number of classes.

## 2.2.8 Neural Networks

A Neural Network (NN) is an ML model inspired by the way that a human brain operates [21, 29]. It consists of a system of neurons (or nodes) that – like the previously described methods – aim to learn underlying patterns in data and produce a corresponding output.

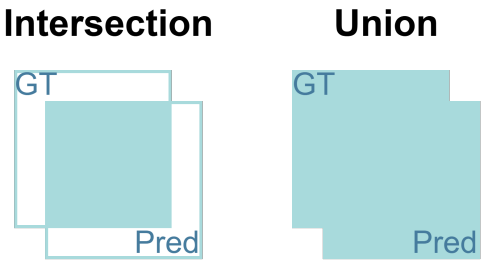



FIGURE 2.12: Depiction of the intersection and the union between a ground truth bounding box and its corresponding predicted (pred) bounding box.

Neural Networks (NNs) are highly flexible and can be designed to combat various problems, including anomaly detection, image recognition, object detection, natural language processing, and speech recognition [29]. Therefore, they can be found in many different areas, such as manufacturing, transportation, computer security, banking, insurance, properties management, marketing, energy, and other areas where conventional mathematical methods fall short. NNs can be implemented in supervised, unsupervised, semi-supervised and reinforcement learning approaches.

A Multi-Layer Perceptron (MLP) is a typical NN. It is a type of NN, where each layer contains a number of neurons that are connected to all the neurons in the next layer, as depicted in Figure 2.13. Each of these connections has its own weight, which determines how much the value of the neuron from the previous layer influences the value of the neuron in the next layer. The workings of a single neuron are depicted in Figure 2.14, and the value of a neuron is then calculated as follows:

$$y = f(b + \sum_{i=1}^{n} x_i \cdot w_i) \tag{2.9}$$

Where:

- $y$: Output of the neuron.
- $f$: Activation function.
- $b$: Bias.
- $n$: Number of neurons in the previous layer.
- $x_i$: Value of the $i$-th neuron in the previous layer.
- $w_i$: Weight of the connection from the $i$-th neuron in the previous layer.

Deep Learning (DL) refers to a type of NN with more neurons and complex layer connections, typically requiring more resources to train than smaller NNs [29]. However, they are able to learn intricate connections within data and often extract features without requiring (a lot of) preprocessing. Examples of Deep Learning (DL) architectures include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Autoencoders.

Training a supervised NN requires an input-output pair to be provided. The input is passed through the NN to produce a prediction, which is then compared to the expected

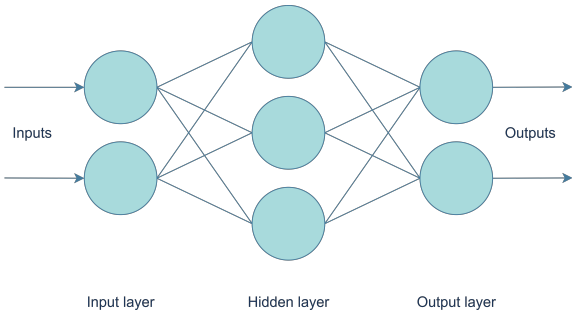



FIGURE 2.13: Example architecture of an MLP.

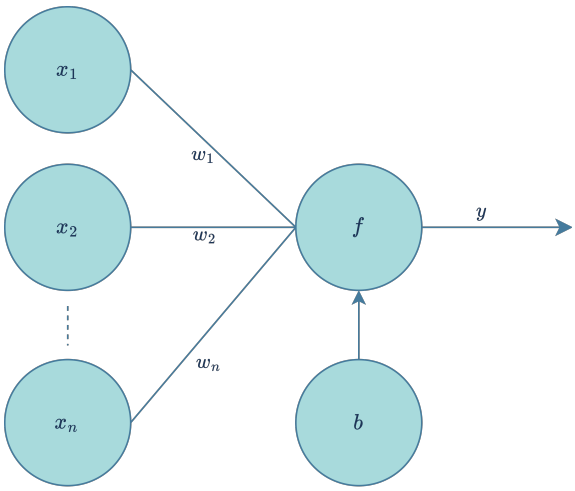



FIGURE 2.14: Depiction of a single neuron.

output using a loss function. The loss function provides a way to evaluate the error, and this value is used during optimization to update the weights.

In the optimization step, backpropagation is used to calculate the gradient of the loss function with respect to the weights, indicating the direction for the weight updates to minimize the loss. This gradient is then used in combination with the learning rate and chosen optimization algorithm to update the weights. The equation used to update the weights is typically expressed as:

$$w^{(t+1)} = w^{(t)} - \eta \cdot \frac{\partial L}{\partial w} \tag{2.10}$$

Where:

- $w^{(t)}$: Weights before the update.
- $w^{(t+1)}$: Weights after the update.
- $\eta$: Learning rate.
- $\frac{\partial L}{\partial w}$: Gradient of the loss function $L$ with respect to the weights $w$, calculated via backpropagation.

However, different optimization algorithms - such as Adam, or AdaGrad - modify the weight update expression to enhance performance and convergence speed.

### Loss function

The loss function provides a way to evaluate the difference between the expected output and the actual output of an ML algorithm [30], similar to evaluation metrics (see Section 2.2.7). The key difference between evaluation metrics and loss functions is their purpose; evaluation metrics provide a way to evaluate the total performance of a model and allow it to be compared to other models, whereas loss functions represent the error in the predictions of a model and the value is used to optimize the model. Not each loss function is equally applicable for each ML task, applying the right loss function results in better quantification of the error, improving training speed and efficiency. Some evaluation metrics can also directly be utilized as loss functions, such as the Mean Average Error (MAE), Mean Square Error (MSE), and Root Mean Square Error (RMSE) metrics.

### Activation function

Activation functions are almost always applied to the output of all neurons in NNs, except for the input layer [21]. The activation functions are nonlinear so that the network can learn more complex non-linear relationships within the data. Some activation functions contain a form of thresholding, such that the output of the neuron is set to 0 when a threshold has not been met. Common activation functions are the Sigmoid, Tanh, and Rectified Linear Unit (ReLU) function, they are calculated as follows:

$$\text{Sigmoid function:} \quad f(x) = \frac{1}{1 + e^{-x}} \tag{2.11}$$

$$\text{Tanh function:} \quad f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.12}$$

$$\text{ReLU function:} \quad f(x) = \max(0, x) \tag{2.13}$$

Where $x$ is the output of a node, i.e. $x = b + \sum_{i=1}^{n} x_i \cdot w_i$.

### 2.2.9 Convolutional Neural Networks

CNNs are a type of DL architecture designed for handling grid-structured data, such as images, making them suitable for tasks such as image recognition and object detection [31, 32]. CNNs typically consist of convolutional layers, pooling layers, and fully connected layers. This architecture significantly reduces the complexity of NNs when working with images. For example, in a fully connected NN, an RGB image of $64 \times 64$ pixels would require $64 \times 64 \times 3 = 12288$ input neurons, resulting in 12288 trainable weights and a trainable bias for each neuron in the next layer. In contrast, a $5 \times 5$ convolutional kernel operating on each colour channel would reduce this to $5 \times 5 \times 3 = 75$, creating just 75 trainable weights and a trainable bias for each neuron in the next layer. This noteworthy reduction in trainable parameters not only improves computational efficiency, but also reduces the risk of overfitting, impacting its generalizability.

Convolutional layers work by sliding a kernel (or filter) over the input data to extract features. These kernels can vary in size and shape, but are commonly a square with odd dimensions, such as $3 \times 3$ or $5 \times 5$. Figure 2.15 illustrates an example of a convolution. At each step, the kernel performs a calculation over a small patch of the input, and by going over the image step by step, the output feature map is generated.

Convolutional layers have a number of hyperparameters that can be changed to alter their behaviour, including the kernel size, stride, depth, and padding:

- Kernel size: Dimensions of the sliding window.
- Stride: Step size of the sliding window. Lager strides provide less window overlap and reduce output size.
- Depth (or output channels): Amount of kernels convolving over the input.
- Padding: Determines if and how extra values should be added to the edges of the input, increasing the size of the output. A common padding option is zero-padding, which adds zeroes to the edges of the input.

These parameters allow CNNs to be tuned to extract features efficiently and effectively.

Convolutional layers are often paired with pooling layers to further process the extracted features [33]. Pooling layers reduce the dimensions of the feature maps, making the model more computationally efficient and robust against overfitting, while losing some lower-level information. Pooling can be performed in various ways, such as max pooling or average pooling. Like convolutional layers, pooling layers slide over the input with a defined window size and stride. However, instead of applying a kernel calculation, they reduce the information in a patch using a specific function. In max pooling, the maximum value in each patch is taken as the output, whereas in average pooling, the average of the values in each patch is used as the output.

## 2.3 You Only Look Once (YOLO)

You Only Look Once (YOLO) was initially presented by Redmon et al. (2016) as an approach to object detection that treats object detection as a regression problem [34]. The method consists of a single end-to-end NN that predicts bounding boxes and class probabilities in one evaluation, using full images as inputs. This structure eliminates the need for more complex pipelines and therefore enables high-speed processing up to 155

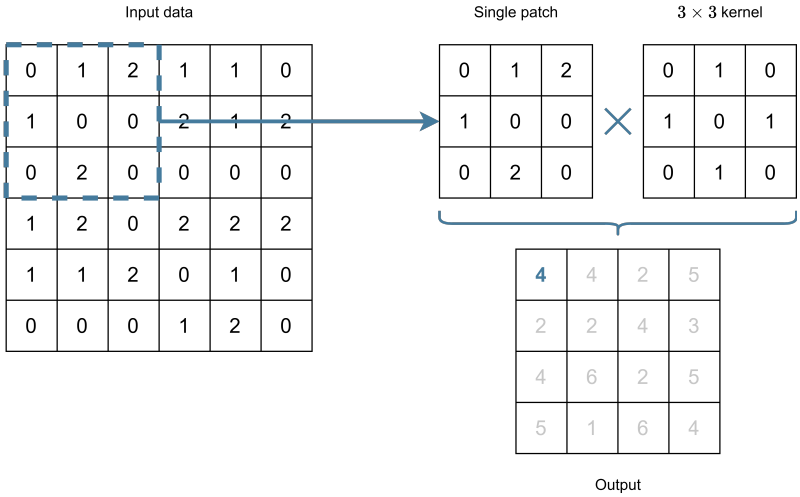



FIGURE 2.15: Example of a convolution using a $3 \times 3$ kernel, stride of 1, and no padding.

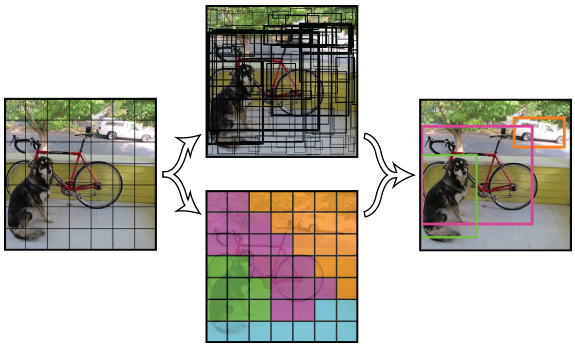


FIGURE 2.16: The YOLO model workings. Adapted from [34].

frames per second. Furthermore, in comparison to sliding window and region proposal-based algorithms, this method looks at the entire input while making the predictions, allowing it to take in the full context of the image.

The rectangles that surround the detected objects are called bounding boxes. The classes are determined per bounding box, so the output of YOLO inference is the image with bounding boxes and their corresponding predicted classes. Figure 2.16 shows an example of these bounding boxes being determined.

YOLO works by dividing the input into a grid, where each grid cell predicts class probabilities and bounding boxes. For the boxes, five properties are predicted: the x and y coordinates of the center, the width and height, and the confidence scores that the box contains an object. The grid size is defined by $S$, the amount of bounding boxes possible per cell $B$, and the amount of classes by $C$, resulting in an output size of $S \times S \times (B \cdot 5 + C)$. Figure 2.16 illustrates this process.

### 2.3.1 Training

To train the bounding box prediction, the model uses a weighted sum-squared error loss. Here, the bounding box coordinates that contain an object outweigh the loss of boxes without objects. Besides this, the model predicts the square root of the height and width

instead of the actual height and width to minimize differences between small and big bounding boxes, thus somewhat equaling out the significance of small deviations in the predictions. Furthermore, YOLO predicts multiple bounding boxes per grid cell, so to determine which box is responsible for which object, the box with the highest IoU with the ground truth is selected and used for loss calculation. During inference, duplicate bounding boxes are removed using Non-Maximum Suppression (NMS), where overlapping predicted bounding boxes with a lower confidence score are removed. Here, the amount of overlap is determined by the IoU between predicted boxes.

During training, the following loss function is optimized [34]:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left( C_i - \hat{C}_i \right)^2 \\
&+\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left( C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
\tag{2.14}
$$

Where:

- $\mathbb{1}_i^{\text{obj}}$: 1 if object is present in grid cell $i$ in the ground truth, else 0.
- $\mathbb{1}_i^{\text{noobj}}$: 1 if object is *not* present in cell $i$ in the ground truth, else 0.
- $\mathbb{1}_{i,j}^{\text{obj}}$: 1 if the $j$-th bounding box in cell $i$ is "responsible" for that prediction (else 0).
- $\lambda_{\text{coord}}$: Constant scalar for the coordinate loss, set to 5.
- $\lambda_{\text{noobj}}$: Constant scalar for the loss due to cells without objects, set to 0.5.
- $S$: Grid size.
- $B$: Number of predicted bounding boxes per cell.
- $x_i, y_i, w_i, h_i$: Ground truth $x$ and $y$ coordinates of the center, and the width and height of the bounding box.
- $\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i$: Predicted $x$ and $y$ coordinates of the center, and the width and height of the bounding box.
- $C_i$: Ground truth confidence score for a bounding box and whether it contains an object.
- $\hat{C}_i$: Predicted confidence score for a bounding box and whether it contains an object.
- $p_i(c)$: Ground truth conditional class probability for class $c$.

| Scale | Depth | Width | Max channels |
|---|---|---|---|
| n (nano) | 0.50 | 0.25 | 1024 |
| s (small) | 0.50 | 0.50 | 1024 |
| m (medium) | 0.50 | 1.00 | 512 |
| l (large) | 1.00 | 1.00 | 512 |
| x (extra-large) | 1.00 | 1.50 | 512 |

TABLE 2.2: An overview of the different YOLO scales and their modifiers.

- $\hat{p}_i(c)$: Predicted conditional class probability for class $c$.

The purpose of each line in the loss function is as follows:

1. Calculates the error in the predicted x and y coordinates of the bounding box center compared to the ground truth.
2. Calculates the error in the width and height. The square root is used to equalize the impact of deviations in both small and larger bounding boxes.
3. Calculates the error in the confidence score for each bounding box, where the confidence score is comprised of whether the grid cell contains an object and how well the bounding box matches the ground truth by calculating the IoU.
4. Calculates the error in the confidence score for grid cells that do not contain an object. A lower weight is then applied to this loss to reduce the impact of incorrectly predicted background cells.
5. Calculates the error in predicted class probabilities, which is only counted for cells that actually contain an object.

### 2.3.2 Ultralytics YOLO11

YOLO11 is the newest YOLO version from Ultralytics [35] and is chosen to be used in this work. In the ten versions since YOLOv1, the algorithms has seen some improvements, which will be summarized in this section.

#### Architecture

YOLO comes in 5 different scales, these determine the amount of parameters in the model by modifying the depth, width and maximum channels of certain layers, but the architecture stays the same. The scales and their corresponding alterations are described in Table 2.2. A schematic of the architecture is depicted in Figure 2.17.

**Backbone** The YOLO backbone can be seen as the core of YOLO. Here, the image is progressively transformed into more refined feature maps that allow the head of the algorithm to perform proper object detection. Where the first YOLO version used a feature extraction backbone inspired by GoogLeNet, utilizing a simple CNN architecture [34], YOLO11[3] upgraded to using a mix of convolutional layers, and C3k2, SPPF and C2PSA blocks [36, 37].

[3]As defined in the official Ultralytics repository: https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/models/11/yolo11.yaml

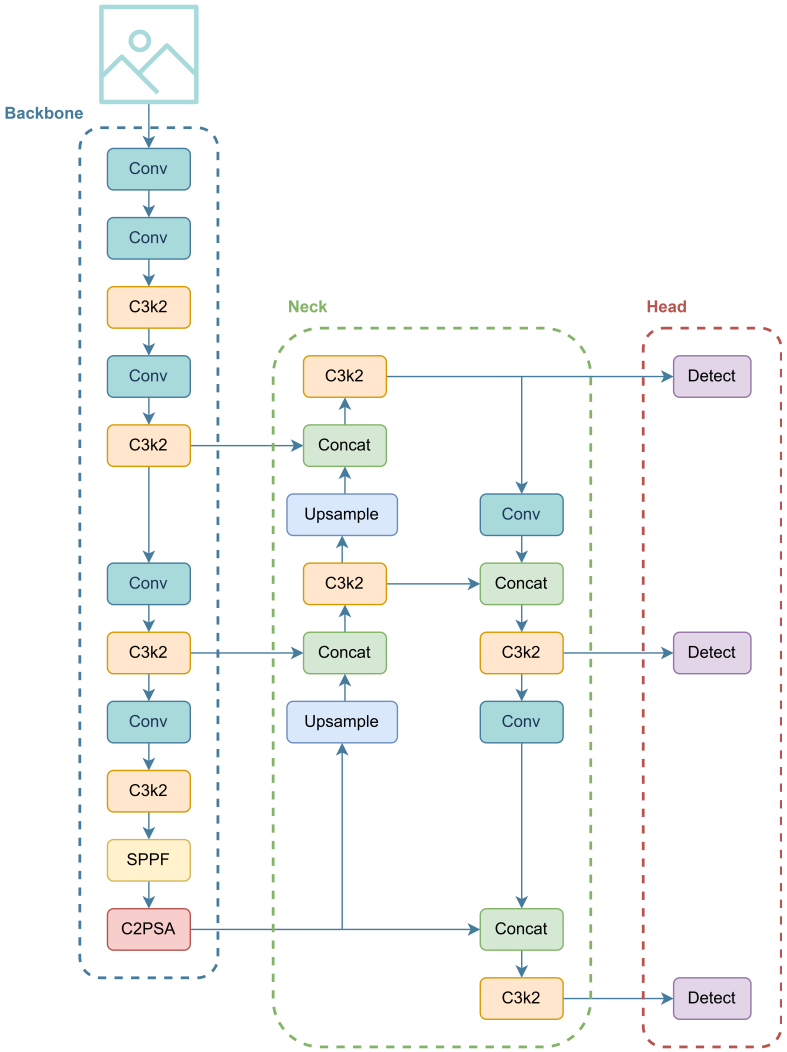



FIGURE 2.17: A schematic overview of the YOLO11 architecture.

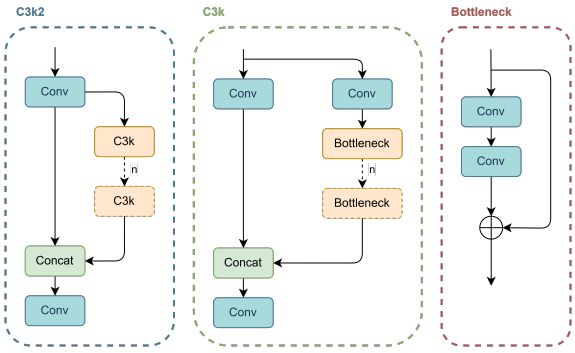



FIGURE 2.18: A schematic overview of the C3k2, C3k, and bottleneck blocks in YOLO.

The backbone starts with convolutional layers that downsample the input image, while increasing the channels, allowing the network to extract more in-depth features. These are followed by C3k2 blocks – an efficient implementation of the Cross Stage Partial (CSP) Bottleneck – which contain one or multiple C3k blocks, depending on the depth parameter of the chosen YOLO scale. As depicted in Figure 2.18, these C3k blocks consist of bottleneck layers, which are layers with fewer neurons, forcing the network to learn a more general representation of the input, reducing dimensionality and increasing generalizability. These C3k2 blocks retain knowledge about higher-level features through the residual (skip) connections, but learn a lower-dimensional feature set in the bottleneck layers.

Near the end of the backbone, the model includes a Spatial Pyramid Pooling - Fast (SPPF) layer, which is a faster version of SPP, reducing computational complexity. This layer uses max pooling at multiple scales to extract more general and more specific features in a single representation, allowing the rest of the network to use a diverse set of information. This layer enhances the network's ability to detect objects of different sizes and positions, making it more robust.

The backbone is closed with a C2PSA block, with one or multiple Position-Sensitive Attention (PSA) blocks, depending on the depth parameter of the chosen YOLO scale. As depicted in Figure 2.19, the input is divided into two parts: one passes through PSA layers, which prioritizes focus on important regions of the feature map, while the other bypasses them.

The backbone passes outputs at different resolutions via the neck to the head, this happens at $P3/8$, $P4/16$ and $P5/32$. Here the $P$ stands for the processing stage in the backbone, and the number after the slash stands for the downsampling rate. Meaning that $P3/8$ passes the highest resolution feature map for finer details (smaller objects), whereas $P5/32$ passes the lowest resolution feature map for more contextual details (bigger objects).

**Neck** The YOLO neck uses a combination of upsampling, concatenation and C3k2 layers to bring the different scaled features together and send them to the YOLO head. Combining the features at different scales improves robustness in multi-scale object detection through the combination of finer details and broader contextual information. Here, the following happens:

1. The features from $P5/32$ are upsampled and concatenated with $P4/16$, after which a C3k2 layer is applied.

2. This is then upsampled and concatenated with $P3/8$, after which a C3k2 layer is applied.

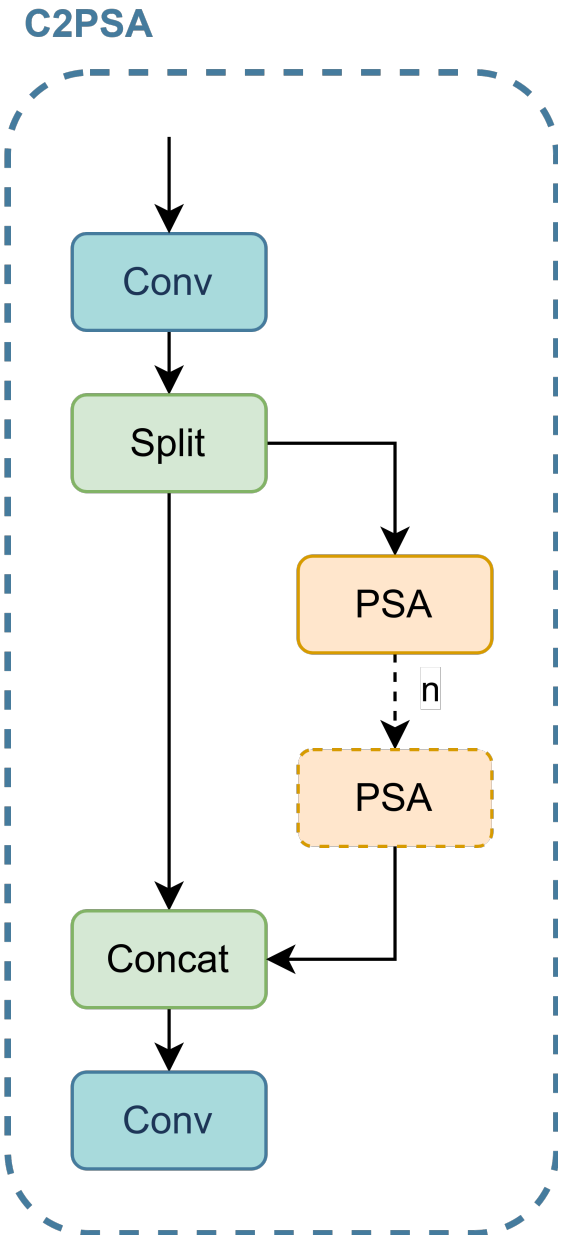



FIGURE 2.19: A schematic overview of the C2PSA block.

3. This is then downsampled and concatenated with the output of **1.**, after which a C3k2 layer is applied.

4. This is then downsampled and concatenated with the output of the backbone, $P5/32$, after which a C3k2 layer is applied.

5. The output of **2.**, **3.** and **4.** are passed to the Detect layer, which makes up the head of the network.

**Head** The YOLO head consists of the detection module and is responsible for predicting the bounding box coordinates and class probabilities. Unlike earlier versions, YOLO11 uses an anchor-free design, eliminating the use of pre-defined anchor-boxes. The anchor-free design is a simpler approach, reducing the computational complexity. For each of the different feature maps passed by the neck ($P3$, $P4$ and $P5$), each grid cell predicts a bounding box ($x$-offset, $y$-offset, $w$, $h$) and class probability for each class resulting in a tensor shape of $(B, N, \text{no})$, where:

- $B$ : Batch size.
- $N$ : Number of anchors (equal to the number of grid cells).
- $\text{no} = \text{number of classes} + \text{reg\_max} \cdot 4$ : Outputs per anchor.

Bounding box regression makes use of discretization, representing each coordinate as a probability distribution. Here reg_max references the fixed number of bins representing the bounding box coordinates, by default set to 16. A weighted sum of the bins is used to obtain the final coordinate value. This distribution is trained using Distribution Focal Loss (DFL), which learns to focus on the two bins closest to a ground-truth target value. E.g. if the target value for $x$ is 0.6, then the distribution is altered so that bins 0.5625 and 0.625 (when $\text{reg\_max} = 16$) will obtain a higher probability. By using this discretized approach, the model allows for uncertainty to be represented in the coordinates, improving the overall accuracy.

Besides DFL loss, YOLO makes use of a bounding box loss based on IoU, called Complete Intersection over Union (CIoU). This incorporates the distance between box centers and the similarity in aspect ratio to improve over standard IoU. The third loss function that is used for object detection is a Binary Cross-Entropy (BCE) loss to determine the classification loss. This loss function evaluates how accurate the class predictions are in comparison to the labeled classes. In YOLO11, it is possible to assign a weight to these three different losses when the model fails to properly optimize using the standard loss weights.

## 2.4 Context-based Low-light Image Enhancement (CoLIE)

Context-based Low-light Image Enhancement (CoLIE)[4] is a novel method for LLIE, proposed by Chobola et al. [38]. Their approach is based on the Retinex theory (see Section 2.5.2) and therefore incorporates splitting the image into an illuminance component and a reflectance component. It does this by converting the image from RGB into the Hue, Saturation, and Value (HSV) colour space, where the Value (V) component is taken as the illuminance component. Using this colour space reduces the problem into enhancing a single component: the Value.

[4] Available: https://github.com/ctom2/colie

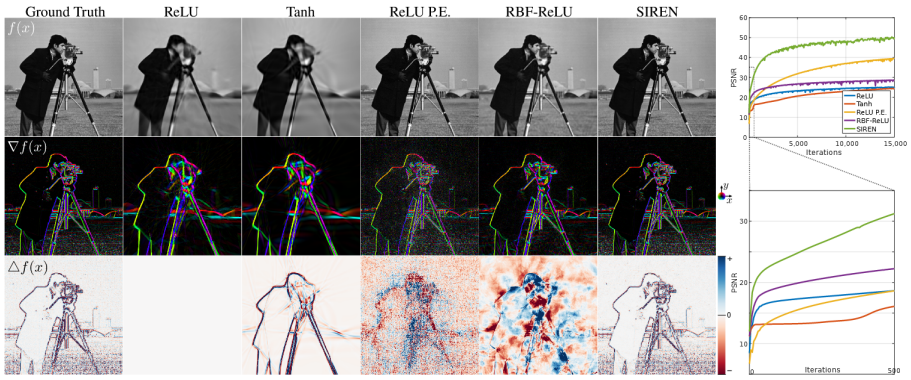



FIGURE 2.20: A visualization of NIRs using different activation functions, including their first and second order derivatives. Adapted from [39].

For this research, the CoLIE method was chosen because of the domain-independence, the high-scoring metrics in LLIE, and the proven effectiveness in optimizing object detection efficacy in low-light images [38].

### 2.4.1 Neural Implicit Representations

CoLIE uses a concept called Neural Implicit Representations (NIRs), particularly Sinusoidal Representation Networks (SIRENs) [39]. This can be used to represent data as a mapping function from input coordinates to a corresponding value by encoding data in the parameters of a fully-connected NN. For example, for an image, a NIR learns to adjust its weights to obtain a mapping from every $(x, y)$ coordinate to the corresponding pixel intensity, effectively storing the contents of the image implicitly in the weights of the NN. This way, the represented data is continuous, allowing for smooth interpolation or obtaining an output image of any resolution.

SIRENs specifically make use of a sinusoidal activation function, which allow the network to capture significantly finer details than NIRs using ReLU activation functions. This can be attributed to the partially linear nature of the ReLU activation functions, giving it a second order derivative of zero. This prevents ReLU-based NIRs from representing finer details that are contained in higher-order derivatives of the signals, as shown in Figure 2.20.

For CoLIE, a NIR is used to map 2D coordinates to the value component of the HSV colour space. Here the NIR is trained to predict the original value component, based on the pixel coordinates and the original value components, a $W \times W$ window centered around the pixel. This addition of providing a context window deviates from conventional NIRs, but it allows the network to understand more intricate connections within the image, preserving finer details. The network starts with two branches, the first processing the context window and the second processing the coordinates, each containing a hidden layer of 256 neurons. Each branch is then passed to a hidden layer containing 128 neurons, after which the two branches are concatenated and passed to a final hidden layer containing 256 neurons. This final layer is connected to a single output value activated using a sigmoid function which represents the intensity of the value component for the provided input. An overview of the NIR used by CoLIE is depicted in Figure 2.21.

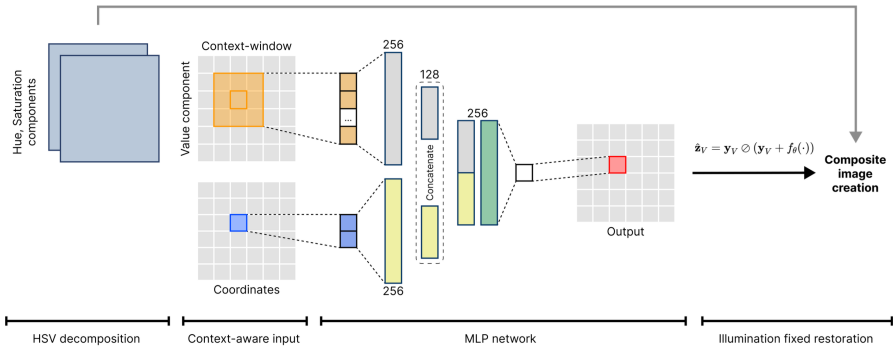



FIGURE 2.21: An overview of the CoLIE NIR algorithm. Adapted from [38].

### 2.4.2 Zero-shot

CoLIE uses a zero-shot approach, allowing the enhancement of any image with any degree of under-exposure, regardless of image domain. Zero-shot refers to a concept within ML where a model is able to handle unseen data without explicitly being trained on similar data. This approach significantly increases the generalizability of an ML model, and it mitigates the need for labeled data to train for the specific task it is being applied to. In the case of CoLIE, the NIR is adapted for each individual image, while optimizing a set of loss functions (see Section 2.4.3), without requiring *any* prior training. The domain-independence allows CoLIE to achieve great performance in many different image domains, from low-light street photography to fluorescence microscopy. This domain-independence does come with a performance trade-off; each image requires re-fitting the NIR, which takes a couple of seconds.

### 2.4.3 Loss function

The loss functions used in CoLIE are defined as

$$\mathcal{L}_{total} = \alpha\mathcal{L}_f + \beta\mathcal{L}_s + \gamma\mathcal{L}_{exp} + \delta\mathcal{L}_{spa} \tag{2.15}$$

$$\mathcal{L}_f = \frac{1}{M}\sum_{i=1}^{M}((\hat{\mathbf{x}}_V)_i - (\mathbf{y}_V)_i)^2 \ () \tag{2.16}$$

$$\mathcal{L}_s = (\|\nabla_i\hat{\mathbf{x}}_V\|_2 + \|\nabla_j\hat{\mathbf{x}}_V\|_2)^2 \ \text{(TV)} \tag{2.17}$$

$$\mathcal{L}_{exp} = \frac{1}{N}\sum_{k=1}^{N}\|\sqrt{\mathcal{T}_k} - L\|_2 \tag{2.18}$$

$$\mathcal{L}_{spa} = \frac{1}{M}\sum_{l=1}^{M}|(\hat{\mathbf{z}}_V)_l| \tag{2.19}$$

Where:

- $\mathcal{L}_f$: Fidelity loss, relating to the pixel-level similarity between the original and the enhanced illumination values, measured using MSE. This is used to obtain a representation similar to the original Value space.
- $\mathcal{L}_s$: Smoothness loss minimizes sudden transitions in the illumination field, improving the consistency of illumination in the enhanced image. It is based on the Total Variation (TV) loss.

- $\mathcal{L}_{exp}$: Exposure loss ensures that the average intensity of local regions in the illumination field matches the desired intensity, which is set using parameter $L$. The usage of local regions ensures that all parts of an image achieve a similar intensity in the illumination field.
- $\mathcal{L}_{spa}$: Sparsity loss penalizes excessively high value components (of the HSV space) in the enhanced image, preventing overenhancement.
- $M$: Total number of pixels in the image.
- $(\hat{\mathbf{x}}_V)_i$: Enhanced illumination value at index $i$.
- $(\mathbf{y}_V)_i$: Ground truth value component at index $i$.
- $\nabla_i \hat{\mathbf{x}}_V$: Vertical gradient of the enhanced illumination component.
- $\nabla_j \hat{\mathbf{x}}_V$: Horizontal gradient of the enhanced illumination component.
- $N$: number of non-overlapping local regions. Set to 16 in the CoLIE algorithm.
- $\mathcal{T}_k$: Average intensity value of the illumination field at index $k$.
- $\hat{\mathbf{z}}_V$: Enhanced value component (of the HSV space) produced by the enhancement algorithm.

And $\alpha$, $\beta$, $\gamma$, and $\delta$ define the weight of each loss.

### 2.4.4 Guided filtering

To speed up performance, CoLIE optimizes a downscaled version of the input image, mitigating any performance impact caused by the size of high-resolution images. To obtain the finalized enhanced image, the low-resolution enhanced value component is scaled up using a guided filtering approach and used to replace the original image's value component. Guided filtering uses a combination of the downscaled original value component, the low-resolution enhanced value component, and the original (high-resolution) value component. Here, it uses the original component as a guide when upscaling the enhanced value component, to ensure the preservation of smaller details and edges present in the high-resolution image.

## 2.5 Related Work

The following section discusses the related work that was used for this thesis. The summary of the methods used for error detection in 3D-printing setups can be found in Table 2.3 and the summary of the Low-light computer vision methods can be found in Table 2.4.

| Method | Description | Performance | Ref. |
|---|---|---|---|
| Acoustic Emission + Clustering | Acoustic emissions classified into machine states. | 90.2% accuracy | [40] |
| Acoustic Emission + SVM | Acoustic emissions classified into machine states. | 92% accuracy | [41] |
| Accelerometer Vibration Monitoring | Nozzle clogging detection by monitoring vibrations. | - | [42] |
| Motor Current Monitoring | Nozzle clogging detection by monitoring motor currents. | - | [43] |
| Edge Detection for ROI-extraction + Differential Image Analysis | Analyzing the print to detect various print failures. | 60–80% detection rate, but 60–80% FPR | [3] |
| Edge Detection + Multi-printer Monitoring | Edge detection for detecting layer shifts and nozzle clogging. | Detected at 60–79% of the print progress | [44] |
| 3D Model Projection Comparison | Comparing 3D model projections with camera input. | 100% accuracy (catastrophic errors) | [45] |
| Stereo Camera + 3D Model Point Cloud Comparison | Comparing real-world point cloud to 3D model point cloud. | - | [46] |
| Structured Light + 3D Model Point Cloud Comparison | Comparing real-world point cloud to 3D model point cloud. | High accuracy for simple shapes | [47] |
| Simple CNN Classifier | Classify images as "success" or "failure". | 70% accuracy | [48] |
| AlexNet + SVM | "Spaghetti" and "stringing" detection. | 87.10% accuracy | [4] |
| Traditional CV methods + CNN (MobileNet-v2) | Traditional methods for missing and shifted layers, CNN for over and underfill. | 87.2% & 75.0% F1-score missing/shifted layers, 86.41% accuracy over/underfill | [49] |
| ResNet-18 CNN | "Stringing" and "under-extrusion" detection. | 83–84% accuracy | [50] |
| YOLOv8 | "Stringing", "obstacle", and "unstick" detection. | 90.0% mAP@50 | [51] |
| YOLOv5 | "Curling", "break", "foreign", "gap", "uneven", and "good" detection. | 92.8% mAP@50 | [52] |
| YOLOv2 (Obico) | Open-source spaghetti detection software. | 60% mAP | [5] |

TABLE 2.3: Overview of related work on error detection in 3D-printing setups.

### 2.5.1 Error Detection in 3D-Printing Setups

The topic of "error detection algorithms" in 3D printing scenarios is not new. Existing work often focuses on one or more specific print defects (see Section 2.1.2), where their proposed methods allow for the detection of one or multiple anomalies. None of the encountered works focus specifically on anomaly detection in low-light scenarios, showing a clear research gap. In the following section, an overview is provided of what is out there.

#### Non-Computer Vision-based Methods

This research makes use of the YOLO algorithm (see Section 2.3), which is a CV based approach. However, some literature proposed different solutions. For example, Liu et al. [40] and Wu et al. [41] made use of acoustic emission sensors to determine the status of the machine. Liu et al. employed a clustering approach to classify data into five machine states; normal, semi-blocked, blocked, material loading, and run-out-of-material. Their approach obtained an average classification accuracy of 90.2%. Instead of a clustering approach, Wu et al. used an SVM to differentiate between these machine states, obtaining a classification accuracy of 92%. Using acoustic emissions shows potential, as it enables accurate machine state monitoring, without much overhead. However, it is not applicable to this work since spaghetti defects occur while the machine is in a normal state, making them undetectable using this approach.

A similar approach by Tlegenov et al. [42] uses an accelerometer to monitor vibration signals to detect nozzle clogging. They modeled the theoretical forces in the machine when nozzle clogging occurs and then compared those to real-world measurements, showing that their approach is a feasible way of monitoring nozzle clogging. In another work by Tlegenov et al. [43], they monitored the changes in motor currents when nozzle clogging occurred and compared this to a similar theoretical model. Their findings show that the measured currents were similar to the theoretical model, making this approach another feasible way to monitor nozzle clogging in 3D printers. As they focused on nozzle clogging, this method is also not applicable to this research, as spaghetti is not always paired with nozzle clogging.

#### Computer Vision-based Methods

Besides non-CV-based approaches, numerous works make use of CV-based approaches, by observing the 3D printer with a camera and registering when an anomaly occurs.

More trivial approaches use simpler CV principles, such as the approach proposed by Baumann and Roller [3]. They used a combination of Houghcircle detection and CannyEdge line detection to calibrate the camera and obtain the Region Of Interest (ROI). Next, the frame is analyzed using a manually selected colour with thresholding, after which the largest connected region matching this colour is extracted and identified as the printed object. Finally, this object is analyzed to observe possible missing material flow anomalies. Another part of their setup uses differential images to determine changes between consecutive frames, which allowed them to monitor if the object detached from the print bed. Their work demonstrated a 60% to 80% detection rate, but also a 60% to 80% False Positive Rate, showing that their solution is not very robust.

Similarly, Becker et al. [44] looked into the monitoring of 3D printers using CV. For the preprocessing step, they made use of Gaussian filtering, grayscale conversion and sobel filtering to find the edges. Their focus was on layer-shift errors and nozzle clogging, and preventing material waste by stopping the print when the errors are detected. The distinctive aspect of this work was the option to monitor multiple 3D printers at once, as the camera is mounted on a movable robot. Their tests introduced errors at around 39%

into the printing process and the average print stopped at 60% for nozzle clogging errors and 79% for layer shift errors, showing a relatively slow detection speed, but still reducing material waste.

Nuchitprasitchai et al. [45] used the STL file of a print and compared the observed image from the camera to the simulated projection of the STL file, raising an error if there is more than a 5% difference. Their approach compared the use of a single and dual-camera setup, concluding that the dual-camera setup is more accurate but takes over 4 times as long; 45-75 seconds compared to 10 seconds for the single-camera setup. Combining the single and dual-camera setup achieved a 100% detection accuracy in their test for catastrophic errors.

Another method comparing the model with the observation was proposed by Holzmond and Li [46]. They used a stereoscopic camera to reconstruct a point cloud of the printed object after each layer. This point cloud was then compared to the point cloud generated by the original Computer Aided Design (CAD) model to observe any deviations, indicating possible defects. Their model showed promising results when observing the generated deviation maps, however they did not provide any concrete metrics to quantify the model's robustness. Their approach also requires optimal lighting conditions and a high-contrast textured filament for the stereoscopic camera to function.

A similar method was proposed by Charalampous et al. [47]. They employed a 3D structural light scanner setup consisting of a stereoscopic camera paired with a projector that emits a light pattern on the surface, mitigating the need for high-contrast textured filaments. They provided MAE and RMSE metrics in their work to quantify the spatial deviations of their measured point cloud data versus the theoretical point cloud data, based on the CAD file. For simple geometries, the system obtained a high accuracy, indicating that this method is suitable for analyzing the observed structure, whereas the error increased for more complex geometries. Charalampous et al. speculated that the increased deviations are caused by lack of precision of the FDM printer for more complex geometries. Nevertheless, their setup proved to be an effective way for analyzing 3D prints for deviations from the original CAD model, while also being adaptable to other forms of AM, such as PBF, due to the reliance on point cloud comparisons instead of specific FDM processes.

While the aforementioned methods contribute to interesting aspects of anomaly detection in 3D printers, they are not directly applicable to this research. The works by Baumann and Roller [3] and Becker et al. [44] are unable to detect spaghetti defects, and the works by Holzmond and Li [46] and Charalampous et al. [47] require extra external hardware and have also not been tested on spaghetti defects.

**Computer Vision and Machine Learning** Other work focused on the combination of computer vision techniques with ML, to extract abstract features from images and detect anomalies. For example, a simple approach was presented by Zhang et al. [48], where a simple CNN setup was used to classify images as "successful" or "failed". Although no special steps were taken, this method already achieved an accuracy of 70%, showing that an ML approach is promising.

Yean and Chew [4] introduced a CNN-based approach for the detection of "spaghetti" and "stringing" defects. Their method employed an AlexNet DL network used for feature extraction and an SVM for classification and obtained an overall accuracy of 87.10%. Their approach did show slightly worse performance in real-world testing, achieving an 86% accuracy. They also mentioned that early-stage defects were harder to detect, as the defect severity was minor when it started to occur.

An approach combining traditional CV methods and ML was proposed by Ern and Jyn [49]. Their method starts with ROI extraction and then applies three defect detection algorithms for "missed layer", "shifted layer", and "overfill and underfill" detection respectively. The missed and shifted layer detection use a combination of histogram backprojection with Otsu's thresholding – an automatic threshold selection approach – for ROI extraction, after which the contours are compared to the expected values to determine deviations. For the shifted layer detection, the contour search space is slightly altered to better handle contour size variations. Finally a CNN-based approach is employed for the overfill and underfill detection, where they compared the performance of three lightweight CNN-based models: Xception, MobileNet-v2, and ShuffleNet-v2. The missed layer detection achieved an average F1-score of 87.2%, the shifted layer detection an average F1-score of 75.0%, and the overfill and underfill detection using the CNN-based model with the best accuracy/speed balance, MobileNet-v2, achieved a test accuracy of 86.41%.

Rettenberger et al. [50] employed a ResNet-18 CNN approach for classification of anomalies. They created a dataset containing four classes: "good", "under-extrusion", "stringing", and "spaghetti", where their approach focuses on the first three classes. They achieved an accuracy of 84% for the "good" and "stringing" class with an 83% accuracy for the "under-extrusion" class. To validate the generalizability of their approach, they tested it on a different silver print bed (instead of black) and from a different angle. When tested under these different conditions, their approach showed a significant reduction in accuracy. For the silver bed test set, the accuracy score reduced to 51%, 87%, and 51%, for "good", "under-extrusion", and "stringing", respectively. For the different angle test set, the accuracy was reduced to 67%, 81%, and 68%. Their work showed that a CNN-based approach for anomaly detection has potential, as the setup is relatively simple hardware-wise and offers real-time detection capabilities. However, the presented method is limited to detection of two anomaly classes and the achieved accuracy in varied environments is lacking, indicating the need for more robust training. Rettenberger et al. published their dataset, which is used in this work to verify the generalizability of the proposed approach, as described in Section 3.2.

These ML methods have shown to be more versatile in their detection capabilities, allowing them to detect more complex anomalies, such as spaghetti. However, ROI extraction can still be tedious and alterations in the setup have a tremendous negative impact on accuracy.

**YOLO-based methods** Another ML algorithm used for anomaly detection is YOLO, as extensively described in Section 2.3. An example of this was presented by Karna et al. [51], where a modified YOLOv8 algorithm was used to detect the following anomalies: "stringing", "obstacle", and "unstick". Additionally, they labeled three different benign shapes: "rectangle", "cube", and "cylinder". Their approach achieved a mAP@50 of 90.0% and a mAP@50-95 of 89.7%, demonstrating accurate localization and class predictions. A major limitation of their approach is the focus on not only a fixed number of anomaly classes, but also benign classes. This makes their approach unusable for other geometries. However, their evaluation metrics demonstrated the effectiveness of using YOLO for anomaly detection in 3D printing.

Yeh et al. [52] showed a similar approach using a YOLOv5 model, where they focused on the following anomalies: "curling", "break", "foreign", "gap", "uneven". They also added a class "good", indicating no defects, but unlike in the work from Karna et al. [51], this class is not limited to a single geometry. Their approach achieved a mAP@50 of 92.8%, demonstrating better results than achieved in the work by Karna et al., while having two

more anomaly classes. This work showed that YOLO can be employed for the detection of many different anomalies.

Obico[5] (formerly known as The Spaghetti Detective) provides an open-source implementation for spaghetti detection in 3D printers, using YOLOv2 as their detection algorithm [5]. They only have one class: "failure", and only achieved a mAP of 60%. However, its open source nature allows it to be used as a baseline model for comparison (see Section 3.3.2).

### 2.5.2 Low-light Computer Vision

In this work, a LLIE algorithm is used to enhance the detection robustness of a YOLO-based anomaly detection framework. This section contains related work in low-light CV, focusing on enhancement algorithms and direct low-light object detection methods.

[5] Available: https://github.com/TheSpaghettiDetective/obico-server

| Method | Description | Performance | Ref. |
|---|---|---|---|
| YOLOv3 | Direct low-light object detection. | 92.5% mAP (train) | [53] |
| NLE-YOLOv5 | YOLOv5 + additional modules for low-light object detection. | 71.3% mAP@50, 43.4% mAP@50-95 | [54] |
| U-Net + YOLOv5 | U-Net feature enhancement combined with YOLOv5. | 62.3% mAP | [55] |
| DiffLight | Denoising and detail enhancement branches combined. | 25.85 dB PSNR, 87.6% SSIM, 0.082 LPIPS | [56] |
| Gamma correction + U-Net | The combination of multiple gamma corrections and a U-Net. | 18.90 dB PSNR, 80.8% SSIM, 0.159 LPIPS | [57] |
| CICGNet (Retinex-based) | Multi-stage network using the Retinex theory. | 22.42 dB PSNR, 89.4% SSIM, 0.073 LPIPS | [58] |
| Retinex with weighting map | Single split Retinex, optimizing weighting map against nonuniform lighting for detail preservation. | - | [59] |
| Retinex + Dark Channel Prior | Robust Retinex combined with haze removal. | 6.94 avg. entropy | [60] |
| CoLIE | NIR-based Retinex with guided filtering. | 17.89 dB PSNR, 62.5% SSIM | [38] |

TABLE 2.4: Overview of related work on low-light computer vision techniques.

### Direct Low-light Object Detection

In contrast to enhancing images before performing object detection, some works have opted to directly use an object detection framework. To achieve low-light object detection, the works by Susa et al. [53], Peng et al. [54], and Ye and Ma [55] all utilized a YOLO algorithm. Susa et al. used a YOLOv3 algorithm trained on the ExDark dataset without making specific changes to the algorithm, achieving a training mAP of 92.5%. However, they did not provide any concrete metrics on a test set. On the other hand, Peng et al. proposed an altered YOLOv5 algorithm, called NLE-YOLO, which uses additional modules aimed at improving low-light image object detection and was also trained on the ExDark dataset. Their approach achieved a mAP@50 of 71.3% and a mAP@50-95 of 43.4%. Ye and Ma made use of a U-Net-based feature enhancement network before the YOLOv5 network in order to improve feature extraction of low-light images. They altered

the YOLOv5 network to incorporate the U-net features and added extra attention layers for improved detection accuracy. They tested their approach on a manually labeled dataset and achieved a mAP of 62.3%.

All three works concluded that their methods demonstrated promising results in low-light object detection. Peng et al. mentioned that their method outperformed baseline YOLO and other modified YOLO versions, while Ye and Ma showed that their method had better performance than baseline YOLO and several other state-of-the-art models.

### Low-light Image Enhancement

LLIE is a topic that is not only relevant for improving visibility for human observation but also significantly impacts CV systems [8]. The latter is relevant for this work, as it investigates the combination of an LLIE algorithm with object detection to improve anomaly detection in low-light conditions. The following section summarizes findings from related work.

LLIE can be done in different ways. Conventional methods, such as grayscale transformation, histogram equalization, adaptive gamma correction, and pixel intensity fuzzification [8, 61], remain present in literature. However, DL methods have shown significant advancements in LLIE.

For example, DiffLight [56] employs two different DL branches to enhance images: the Denoising Enhancement branch, which uses a diffusion model to correct noise and another model to enhance colour and contrast, and the Detail Preservation branch, which employs a U-Net structure containing attention blocks to focus on the recovering smaller details in low-light images. The outputs of these two branches are fused using a weighted fusion method to produce the enhanced image. Their approach was tested on the LOLv1 dataset and achieved a Peak Signal-to-Noise Ratio (PSNR) of 25.85 dB, Structural Similarity Index Measure (SSIM) of 87.6%, and Learned Perceptual Image Patch Similarity (LPIPS) of 0.082.

In addition to DiffLight, Peng et al. proposed another DL-based LLIE method [57]. Their approach first converts the image to the YCbCr colour space and enhances the luminance (Y) channel using multiple gamma corrections. The multiple gamma-corrected luminance channels are then merged using a DL network. The output luminance and chrominance are then passed through a U-Net-shaped deep feature extraction network that includes dense blocks and an attention mechanism. Finally, the image is converted back to the RGB colour space. Their approach was tested on the LOL dataset and achieved a PSNR of 18.90 dB, SSIM of 80.8%, and LPIPS of 0.159.

**Retinex Theory** A common concept referenced in the literature is the Retinex theory [62]. This theory explains the way that humans perceive colour and lightness, even when illumination is not constant, by separating perception into reflectance and illuminance components. The reflectance component contains colour and lightness information, whereas the illuminance component refers to the lighting conditions of the scene. Several LLIE approaches use the Retinex theory as a basis by enhancing the illuminance of an image without affecting the reflection component, thus retaining the colour and lightness information.

Zhao et al. [58] applied this theory in their Content-Illumination Coupling Guided Low-Light Image Enhancement Network (CICGNet), which splits and combines the images in multiple stages using a truss architecture. This iterative process allows the illuminance component to be enhanced while preserving the details in the reflectance component, preventing overenhancements. Their approach was tested on multiple datasets. When tested

on the LOL dataset, their approached achieved a PSNR of 22.42 dB, SSIM of 89.4%, and LPIPS of 0.073.

Alternatively, the method described by Jia et al. [59] uses a single split into reflectance and illuminance components and then iteratively optimizes a weighting map used to enhance detail preservation in the reflectance channel while enhancing the illuminance. This weighting map is used to combat the issue of nonuniform lighting conditions, by altering the enhancement intensity for each region overenhancement is minimized. Jia et al. did not provide any quantifiable metrics in their work.

Thepade and Shirbhate [60] showed a combined approach using a Robust Retinex Model and a Dark Channel Prior-based enhancement – originally designed for haze removal – which enhances low-light images by estimating and removing darker areas. To obtain the final enhanced image, the results of both approaches are merged together using a weighted fusion method, similar to the Difflight approach. They used the ExDark dataset in their work and achieved a 6.9377 average entropy.

The Retinex theory is also employed in the LLIE method adopted in this work: CoLIE [38] (see Section 2.4).

# Chapter 3

# Methodology

To answer the research questions of this work, a system setup was created which has been used to perform several experiments. To assess the performance in these experiments, a number of metrics was selected and implemented. This chapter describes the devised system setup, the established dataset, and the experiments which contributed to answering the research questions.

## 3.1 System Setup

**YOLO11** The system makes use of the YOLO11 object detection algorithm, as described in Section 2.3, to find and classify spaghetti anomalies. The choice for YOLO was inspired by well-performing existing YOLO-based anomaly detection methods proposed in literature, as well as the publicly-available YOLO-based approach, Obico, as described in Section 2.5.1. Besides that, the combination of bounding box detection and classification into a single regression problem provides YOLO with fast detection by elminating the need for preprocessing steps such as ROI-determination.

**CoLIE** To tackle the problem of low-contrast anomaly detection, this work employed the use of CoLIE for Low-Light Image Enhancement. The primary advantage of using CoLIE is its high generalizability, enabled by a zero-shot learning approach. This allows it to be applied to images from any domain without requiring extensive training on similar data. The downside of CoLIE is the added time of fitting the NN for each individual image. This increases the processing time to multiple seconds depending on the hardware. However, as the proposed system does not require continuous monitoring – but can be limited (e.g. one observation per layer) – this added overhead when including CoLIE does not harm the feasibility of the system.

### 3.1.1 Proposed System

While a full implementation is out of the scope of this work, this section describes a conceptual implementation for a system that would make use of the proposed model.

As depicted in Figure 3.1, the system starts with an input image, observed by a camera attached to the printer. Next, the brightness of the image is used to see if it can be classified as low-light based on an average luminance threshold. If it is below this threshold, the image is first enhanced by CoLIE, otherwise the LLIE step is skipped. After this (possible) LLIE step, anomaly detection using YOLO is performed. If an anomaly is detected with confidence score higher than a set threshold, a counter is incremented. If this counter

exceeds another specified threshold, the print is stopped. If not, the system captures a new image and starts the detection cycle immediately, without waiting for the next layer. This is done to reduce detection delay, while still mitigating FPs by requiring multiple consecutive anomaly predictions. If no anomaly is detected, the counter is reset, and the program first waits until the next layer is finished before capturing the next input image.

The usage of a confidence threshold and counter with a threshold in this case combat the FPs that occasionally occur. Here, lower thresholds result in the printer detecting more minimal anomalies, but also possibly stopping prints too early, increasing production time through unnecessary restarts. On the other hand, higher thresholds mitigate the problem of the printer stopping too early, but can result in a delay in anomaly detection, resulting in more material waste. In the worst case, this delay allows the spaghetti to form a "blob of death", possibly resulting in increased costs. By properly tuning these thresholds, the system can be configured to minimize unnecessary interruptions, while still preventing spaghetti defects.

A possible limitation of this approach would be the delay in detection by waiting for new input images being captured until the layer is finished. While this approach saves computational resources, it introduces a detection delay that might be undesirable. To mitigate this problem, continuous monitoring could be used.

## 3.2 Dataset Description

To perform analysis of the models, this work required a dataset containing images from different stages of FDM prints using multiple filament colours. No dataset was found containing prints with dark-coloured filament, likely caused by the lack of research concerning anomaly detection in low-light environments. Therefore, this work included the gathering of sufficient data, labeling said data, and assembling this data into a YOLO-understandable format. This section describes the details of the assembly of the dataset, as well as the contents of the publicly-available dataset containing (high-contrast) spaghetti defects that was used to quantify the generalizability of the models.

### 3.2.1 Data gathering

For this work, a Bambu Lab X1 Carbon printer was used, as described in Section 2.1.3, which includes a built-in camera with timelapse feature. This feature allowed for the collection of images from a good variety of prints, including those containing spaghetti errors. As this printer is used by multiple people at Fraunhofer Innovation Platform for Advanced Manufacturing at the University of Twente (FIP-AM@UT), several natural-occurring anomalies have been captured in real-world prints. Additionally, to increase the amount of spaghetti defect data, two 3D-models were designed to artificially induce spaghetti errors, significantly reducing the overhead of spaghetti defect data collection.

The first model, depicted in Figure 3.2a, was designed to fall over mid-print, removing the surface required for filament adhesion, causing spaghetti. The second model, depicted in Figure 3.2b, has missing support structures, resulting in filament being extruded in the air, causing spaghetti.

The obtained training data was split into two categories, *Colour* and *Black*, to investigate the respective performances. Here, *Black* contains all the images that were captured with black filament, and *Colour* contains the rest. These categories were individually labeled and exported, forming a *Colour*, *Black*, and *Colour & Black (CoBl)* dataset, where the last contains all data of both categories. Another variation of these datasets was added

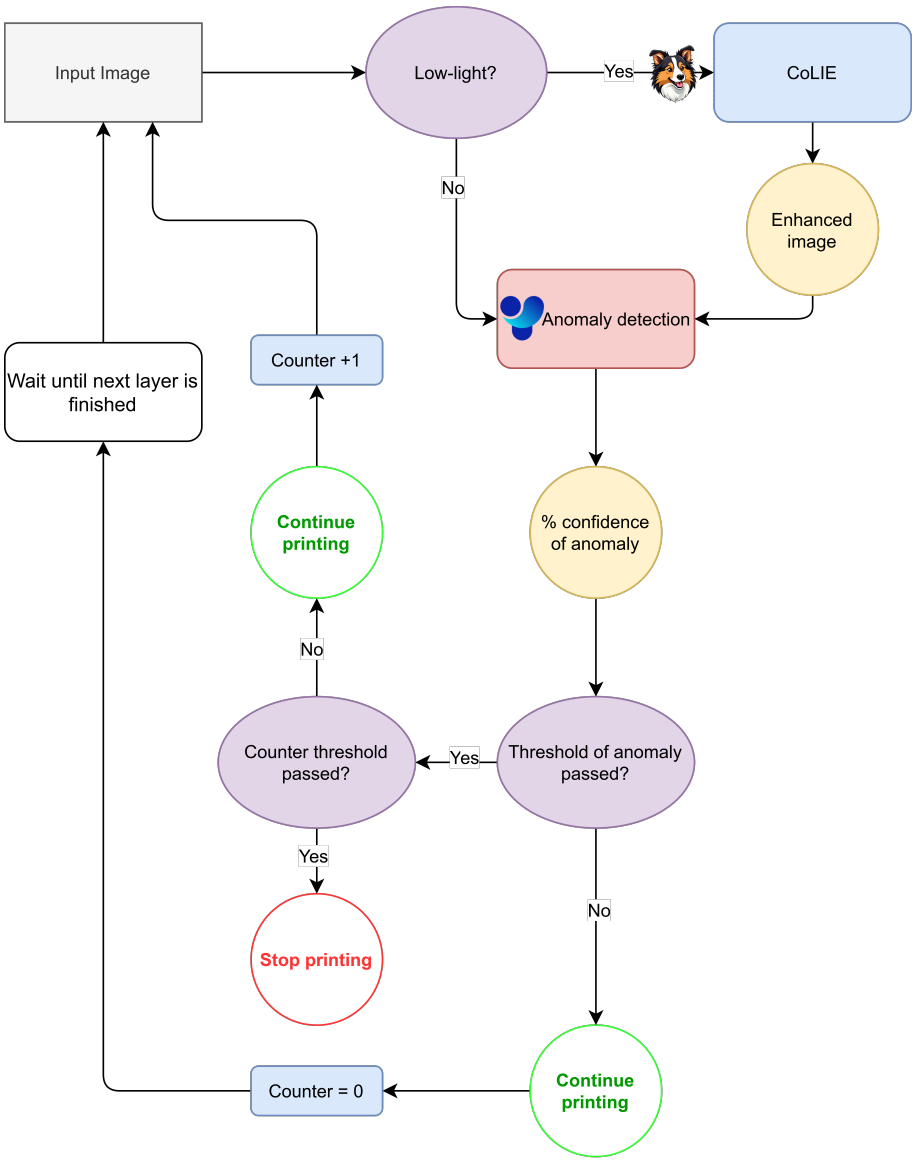
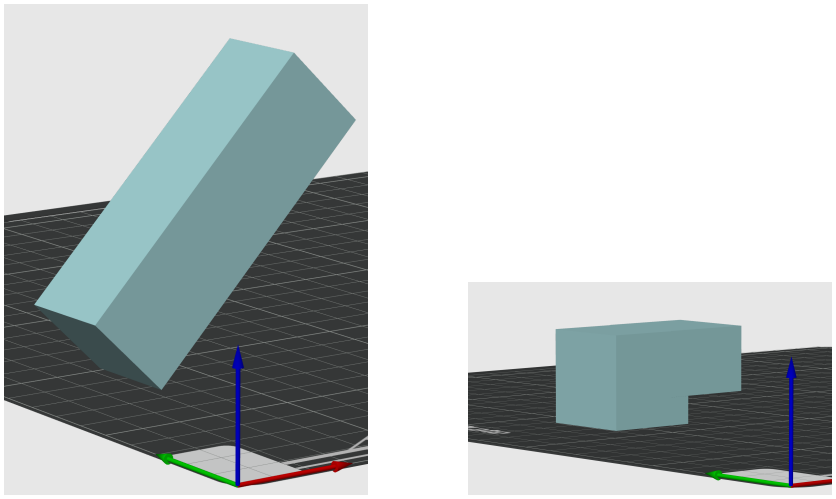



FIGURE 3.1: An overview of a possible system using the low-light anomaly detection algorithm.



(A) (B)

FIGURE 3.2: The two models used for artificially induced spaghetti errors. The object in 3.2a is a rectangle, angled in a way that causes the object to fall over, resulting in spaghetti. The object in 3.2b contains a floating part without supporting structure, when the printer tries to print there, the filament has nothing to adhere to, causing spaghetti.

FIGURE 3.3: An example defect that was left out in the "more tolerant" dataset variation.

where smaller defects were ignored, therefore referred to as the "more tolerant" dataset. This was done to prevent training the model on less important details of spaghetti defects. Figure 3.3 shows an example defect that was left out of the "more tolerant" variation. A final variation of these datasets used a higher sampling rate for background images and is therefore referred to as "more background". The total number of background images and images containing spaghetti, including the "more background" and "more tolerant" variations can be found in Appendix A in Table A.1 for the training datasets.

For the test data, datasets were constructed separate from the training data, also split into two categories: *Colour*, and *Black*. Similar to the "more tolerant" dataset, the minor defects were ignored. The total numbers for the test datasets can be found in Appendix A in Table A.2.

### 3.2.2 Data Labeling & Preparation

To label the data, the open source software Label Studio [63] was used. This tool allows for the labeling of video files where bounding boxes can be drawn that span multiple frames, adjusting in size through interpolation. This interpolation speeds up labeling video files,

especially when the objects move linearly or are static, as not every frame requires manual labeling. These labels were then exported in a JSON format, where for each frame the corresponding bounding boxes were described.

Unfortunately, at the time of writing, Label Studio's export function did not include the actual frame images in the export, but only the frame number. Therefore, the video needed to be split into frames afterwards, which had to then be matched to the labels contained in the JSON file. However, it was found that the frame numbers in the JSON file did not match the frame numbers of the split video, likely through misaligned sampling-rates. To circumvent this issue and properly obtain the corresponding frame images, a script was created that takes the still frames from the Label Studio interface and downloads them. This made sure that the frames in the exported dataset were the exact same as the ones that were labeled, making it significantly easier to match the exported labels to the corresponding frames.

After the export, the downloaded frame images – including Colour and Black – were pre-enhanced by CoLIE and saved in a separate folder. This was done to reduce the computational overhead of CoLIE in the training and testing environment.

Finally, these labeled files were combined and converted into a format that can be processed by YOLO. In this process, the train datasets were split into a "train" and "validation" part, making sure that the split occurs at timelapse-level rather than per frame, which was done to ensure that frames belonging to a single timelapse were not divided over both the train and validation set. The test datasets contain entirely different timelapses than those in the train datasets, and all data in those sets falls under the "test" part, hence no further split was required.

### 3.2.3 External Evaluation Dataset

An external dataset was used to examine the generalizability of the proposed approach. For this aspect, the dataset included in the work by Rettenberger et al. [50] was chosen, as their dataset has been published and contains images of spaghetti defects. However, as their approach did not use YOLO, the bounding boxes had to be added manually using Label Studio.

## 3.3 Experiments

To formulate conclusions for the devised research questions of this work, a set of experiments were conducted. The setup of these experiments is described in the following section.

### 3.3.1 Metrics

To quantify the results of the experiments, a selection of metrics has been used in this work, as described in Section 2.2.7. The base of these metrics is formed by the TPs, TNs, FPs, and FNs. These values were determined by evaluating the predictions made for each frame of the test timelapses as follows:

- TP: The ground truth and prediction both contain at least one spaghetti bounding box.
- TN: The ground truth and prediction both contain no bounding boxes.
- FP: The ground truth contains no bounding boxes, but the prediction contains at least one spaghetti bounding box.

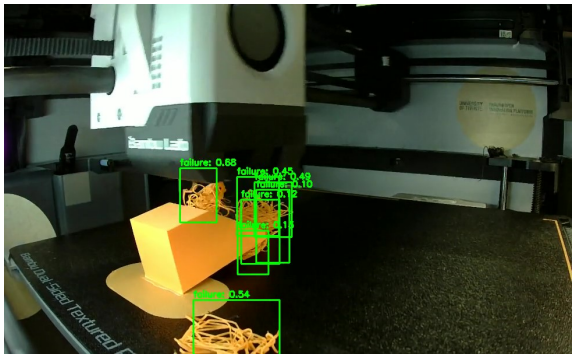



FIGURE 3.4: An example of the bounding boxes generated by Obico, showing multiple overlapping boxes for the same anomaly. This also shows the slight misalignment of some bounding boxes.

- FN: The ground truth contains at least one spaghetti bounding box, but the prediction contains no bounding boxes.

In the testing phase of the models, the number of bounding boxes and the placement of bounding boxes were disregarded. This was done because the goal was to stop the printer in case anomalies were detected, not to accurately indicate where the anomaly occurred or how many individual instances of spaghetti were detected. However, these models did still make use of bounding box position and number of bounding boxes to optimize their predictions in the training phase. This decision was partly made when observing the Obico output while using the test data, as it often showed multiple overlapping bounding boxes for the same piece of spaghetti, as well as slightly misaligned boxes, as shown in Figure 3.4.

The confusion matrix shows the obtained TPs, TNs, FPs, and FNs, and the performance can be expressed in the derived metrics: precision, recall, F1-score, and False Positive Rate. For preliminary testing, primarily the F1-score is used, as it shows the balance between precision and recall, creating an overall image of the performance of the model. To assess the performance of the final model, the F1 and PR curves were calculated to give a better indication of the overall performance of the classifier, as well as to indicate a good value for the confidence threshold.

### Print Failure Stopping Metric

The Print Failure Stopping Metric (PFSM) describes a custom metric devised for this work based on the proposed system in Section 3.1.1. The goal of this metric was to provide insight into how the model would perform when implemented in a real system. It works by virtually "stopping" the print after enough consecutive contain a predicted anomaly, and verifying if the system was supposed to stop. If the system stopped correctly, the delay, expressed in the amount of frames after the anomaly first appeared, is recorded as an additional performance indication. Figure 3.5 shows an example of how this metric works with a threshold of three frames. By obtaining this metric at different thresholds, the trade-off in accuracy versus delay can be shown, which can be important in optimizing the system for each individual use-case.

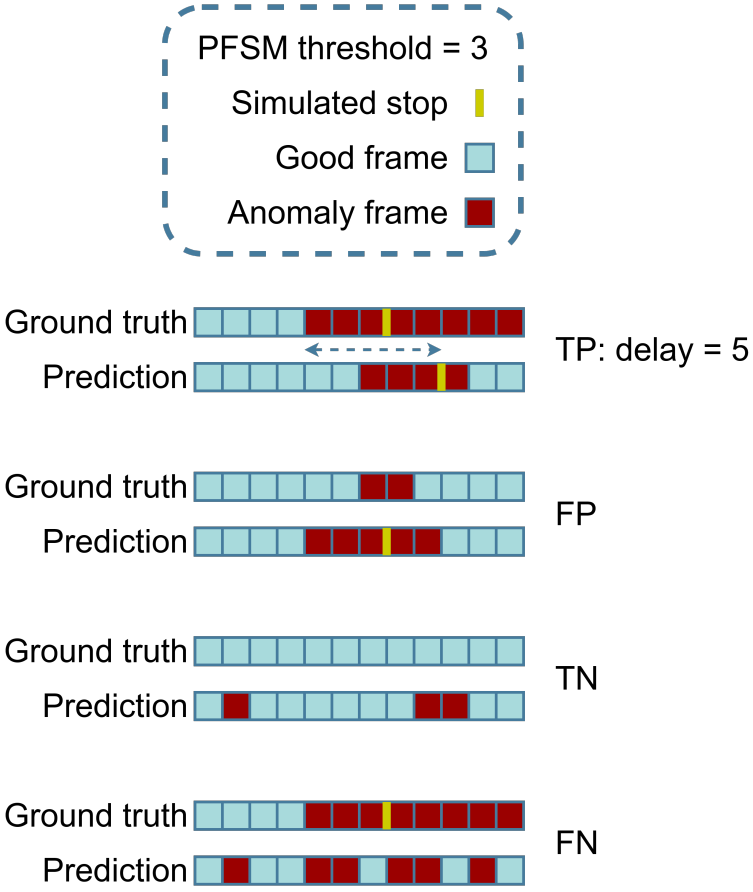



FIGURE 3.5: A depiction of PFSM with a threshold of three frames. This metric shows the theoretical effects of deploying the model in a real system, as well as the delay that is observed within the TPs.

### 3.3.2 Baseline Establishment

The open-source spaghetti detection implementation in Obico (See Section 2.5.1) was used to establish a baseline for spaghetti detection performance. This detection method was applied to the same test sets used for the model described in this work, providing a clear comparison between the two.

### 3.3.3 First Experiment

The first experiment was conducted using either the *Colour*, *Black*, or the combined *CoBl* dataset for training. To influence model performance, the following variations were introduced for each of these datasets:

1. Albumentations (data augmentations). This was done to see what effect these data augmentations have on the performance of the model on the test set.

2. "Background" images. Here, the background images were selected from a number of frames in the timelapses where no anomaly occurs. These were then fed to the YOLO model to learn what should be considered as "background", theoretically reducing the number of FPs. Here, the "more background" variant was also tested to see what effect a higher selection of background images has on the model's performance.

3. CoLIE-enhanced images (20% of images). This was done to see the effects on the model's performance when it has seen enhanced images in the training phase, before applying it to enhanced (and non-enhanced) images in the testing phase.

4. Freezing the YOLO backbone. This was done to see if it would improve the model's performance by reducing the risk of overfitting, as the initial feature extraction layers were kept unmodified during training.

The performance of the models from the first experiment round were then compared by looking at the F1-score, described in Section 3.3.1. Using this comparison, the performance of the models over the Colour, Black, and Rettenberger [50] datasets was shown and the impact of the aforementioned variations can be observed.

### 3.3.4 Second Experiment

After the first round of experimentation, possible preliminary conclusions about the variations could be formed. However, certain variations were not explored. Therefore, a second experimentation round, containing different variations was conducted, with the following setup:

1. Using YOLO11s, YOLO11m, or YOLO11l. For the initial round of experimentation, only YOLO11s was used. In the second round, this was expanded to the medium and large variants, possibly allowing the model to learn more intricate details about spaghetti, allowing for better detection performance.

2. Using 20% or 100% CoLIE-enhanced images. In the first experimentation round, only a variation with 20% CoLIE-enhanced images was investigated, however, the inclusion of 100% CoLIE-enhanced images in the training datasets provides a more complete insight into the possible effects on performance.

3. The "more tolerant" version of the datasets with fewer labels was compared to the original datasets in this round. Section 3.2.1 describes the differences between these datasets.

Similar to the first experimentation round, the models were evaluated by obtaining the metrics as described in Section 3.3.1. After this, their performance was primarily compared by observing the average F1-score over the three test datasets.

### 3.3.5 Additional Experiments

To conclude the experiments, an additional round was performed using the highest-scoring models from the second experimentation round as a baseline. This round included the variations from the first experiment that did not show a clear impact individually. The results of the second round provide good insights into the influence of different YOLO scales and the impact of different amounts of CoLIE-enhanced images in the training dataset. However, the influence of background images, and the influence of using the Colour and Black datasets separately, were left out. The outcome of this additional round provides a full overview of how these training settings can be combined to obtain the optimal model configuration, based on the best overall performance across the three test datasets, focusing on both detection accuracy and generalizability.

# Chapter 4

# Results

This chapter contains the results of the experiments described in Section 3.3, including relevant metrics and figures. Further interpretation of the results are described in Chapter 5.

## 4.1 Baseline Establishment

The first step involved establishing a baseline by running the test sets through an existing open-source spaghetti detection algorithm called Obico, as described in Section 2.5.1. The resulting metrics can be found in Table 4.1 and the F1-curve is shown in Figure 4.1, which shows that the Obico model achieves the highest F1-score of 43% when the confidence threshold is set to 16%, which gives the model the optimal balance between precision and recall. Additional Figures can be found in Appendix B.1.

| Test dataset | Precision | Recall | F1 Score |
|---|---|---|---|
| `Black` | 100.0% | 39.7% | 56.8% |
| `Black (CoLIE)` | 94.9% | 41.5% | 57.8% |
| `Colour` | 63.2% | 17.9% | 27.8% |
| `Colour (CoLIE)` | 37.8% | 18.5% | 24.8% |
| `Rettenberger [50]` | 20.5% | 69.7% | 31.7% |
| `Rettenberger [50] (CoLIE)` | 27.7% | 70.2% | 39.8% |
| **Average** | 51.9% | 38.3% | 35.4% |
| **Average (non-CoLIE)** | 57.7% | 37.9% | 34.8% |
| **Average (CoLIE)** | 46.2% | 38.7% | 35.9% |

TABLE 4.1: Precision, Recall, and F1 Score for the Obico baseline, tested on the different test datasets. The averages are weighted according to the number of frames in each dataset, as shown in Table A.2.

To highlight the effect of CoLIE on the baseline, the metrics achieved on the normal test sets compared to those on the CoLIE-enhanced test sets are shown in Table 4.1 and Appendix B.1, Figures B.4, B.5, B.6, and B.7. Here, Table 4.1 shows that Obico achieves a small 1.0% increase in F1-score when CoLIE is used on the Black dataset, and an 8.1% increase in F1-score when using CoLIE on the Rettenberger [50] dataset.

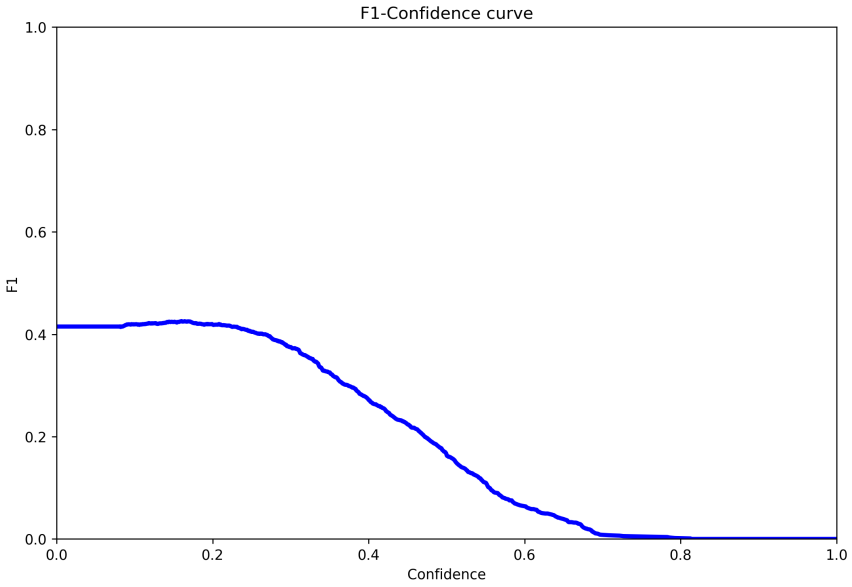



FIGURE 4.1: The F1-curve for the Obico baseline, showing the performance for specified confidence thresholds, indicating that Obico achieves the highest F1-score of 43% at a 16% confidence threshold.

## 4.2 First Experiment

The first experiment was set up as described in Section 3.3.3 and the achieved F1-scores per test dataset can be found in Table 4.2 for the three highest-performing variations and the full overview can be found in Appendix B.2, Figure B.8. This experiment round was primarily used as a preliminary testing round to see the effects of certain parameters or train dataset variations. The results show that most experiments are unable to surpass the Obico baseline; only three experiments achieved a slightly higher average F1-score, with the highest-scoring model in the first round obtaining a 5.2% higher average F1-score. The highest-scoring model in the first round uses the CoBl train dataset, with a low-sample rate inclusion of background images, 20% CoLIE, no albumentations, and a frozen backbone. It shows a decrease in F1-score when applying CoLIE to the test set for Black, Colour, and Rettenberger [50]. However, the second-best model does show a slight increase in F1-score when looking at the Black dataset with CoLIE of 1.1%.

In general, the following trends in training settings can be observed:

- Backbone freezing results in higher F1-scores.
- Using the "more background" dataset overall results in significantly worse F1-scores. However, including background images at a lower sample rate had a mixed impact on performance, achieving F1-scores in both the higher-end, as well as the lower-end of the overall results.
- The albumentations augmentations do not show a clear effect on performance.
- Including CoLIE-enhanced images shows an inconsistent effect on performance.

As the general results did not show significant improvements over the Obico model, a second experimentation round was introduced to explore other variations.

| Train dataset | Background | CoLIE | Albumentations | Frozen Backbone | Black | Black CoLIE | Colour | Colour CoLIE | Rettenberger [50] | Rettenberger [50] CoLIE | Average F1-score |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CoBl | Low-sample | ✓ | - | ✓ | 57.8% | 56.0% | 70.4% | 45.3% | 22.6% | 18.1% | **45.0%** |
| Colour | Low-sample | ✓ | ✓ | ✓ | 58.7% | 59.8% | 43.1% | 44.5% | 22.2% | 18.5% | 41.1% |
| Colour | - | ✓ | - | ✓ | **67.3%** | **67.2%** | 35.4% | 31.9% | 23.2% | 18.5% | 40.6% |
| Obico | - | - | - | - | 56.8% | 57.8% | 27.8% | 24.8% | **31.7%** | **39.8%** | 39.8% |

TABLE 4.2: F1-scores achieved by the three highest-performing train setups tested in the first experiment, including Obico for comparison. The highest F1-scores across all results of the first round are highlighted with bold text. The Obico baseline is marked with red text.

## 4.3 Second Experiment

The second experiment was set up as described in Section 3.3.4, including different YOLO scales and using the "more tolerant" dataset. Additionally, by looking at the findings from the first experiment, the decision was made for the second experiment to continue exploring the inclusion of CoLIE, by testing the influence of a 100% versus 20% CoLIE-enhanced training dataset. Furthermore, the backbone was frozen and the background images were not included for all experiments in the second round. Besides this, to ease interpretation of the results and reduce the number of experiments, only the CoBl dataset has been used for training. The F1-scores for the three highest-performing variations are depicted in Table 4.3 and the full overview can be found in Appendix B.3, Figure B.9.

These results show more setups achieving a higher score than Obico when compared to the first experiment, with the best-performing model scoring a 22.9% higher average F1-score. The best-performing model also showed increased performance in the Rettenberger [50] dataset over the Obico model. Here, the increase is 17.5% and 24% for non-CoLIE and CoLIE respectively.

In general, the following trends in training setups can be observed:

- The "more tolerant" dataset generally achieves higher performance than the normal dataset.
- Using albumentations negatively impacts the performance.
- Using CoLIE-enhanced images in the train dataset does not show a clear impact on performance, for both 20% and 100%.
- The medium and large YOLO model generally show better performance than the smaller scale.

| Train dataset | More tolerant | YOLO | CoLIE | Albumentations | Black | Black CoLIE | Colour | Colour CoLIE | Rettenberger [50] | Rettenberger [50] CoLIE | Average F1-score |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CoBl | ✓ | l | - | - | 76.8% | 72.1% | 69.5% | 66.4% | **49.2%** | **42.2%** | **62.7%** |
| CoBl | ✓ | m | 100 | - | 67.1% | 75.2% | 63.8% | **70.0%** | 43.1% | 39.8% | 59.8% |
| CoBl | ✓ | m | 20 | - | 69.8% | 72.6% | **71.9%** | 63.0% | 39.2% | 33.1% | 58.3% |
| Obico | - | - | - | - | 56.8% | 57.8% | 27.8% | 24.8% | **31.7%** | **39.8%** | 39.8% |

TABLE 4.3: F1-scores achieved by the three highest-performing train setups tested in the second experiment, including Obico for comparison (marked in red). The highest F1-scores across all results of the second round are highlighted with bold text. As the additional experiments round resulted in the same top three, this Figure also depicts the top three achieved in the additional experiments round.

## 4.4 Additional Experiments

While the second experiment showed promising results, an additional experimentation round was set up, as described in Section 3.3.5, to include variations that were inconclusive in the first round but were left out to limit the number of experiments. This additional round aimed to provide a more complete overview of the different settings and their impact on performance. Appendix B.4, Figure B.10 contains the full overview of results of this round, which shows that the setups trained using the individual Colour and Black datasets did not yield better results, and the inclusion of background images further negatively impacted performance, resulting in the same top three as in Section 4.3.

From this final round of experiments, an overall best-performing model was selected, henceforth referred to as **overall best setup**, showing the highest average F1-score over all test datasets – with and without CoLIE. The overall best setup is obtained by taking a "more tolerant" CoBl train dataset, using the large scale YOLO11 model, with backbone freezing, no CoLIE enhancement, no albumentations, and no background images, achieving a 62.7% F1-score, in comparison to Obico achieving a 39.8% F1-score. Table 4.3 further shows the difference in F1-score across the individual test datasets. Here, a performance increase between Obico and the overall best setup can be observed per dataset as follows:

- Black: from 56.8% to 76.8% (20.0% increase)
- Black with CoLIE: from 57.8% to 72.1% (14.3% increase)
- Colour: from 27.8% to 69.5% (41.7% increase)
- Colour with CoLIE: from 24.8% to 66.4% (41.6% increase)
- Rettenberger [50]: from 31.7% to 49.2% (17.5% increase)
- Rettenberger [50] with CoLIE: from 39.8% to 42.2% (2.4% increase)

Table 4.4 presents the additional performance metrics of the overall best setup, including results for the three test datasets with and without CoLIE enhancement. This table shows that when applying CoLIE a slight reduction in each metric can be observed. Furthermore, the optimal confidence threshold can be derived from the F1-curve, depicted in Figure 4.2, achieving the highest F1-score of 66% at a confidence threshold of 27%, which provides the model with the optimal balance between precision and recall.

| **Test dataset** | **Precision** | **Recall** | **F1 Score** |
|---|---|---|---|
| `Black` | 95.9% | 64.0% | 76.8% |
| `Black (CoLIE)` | 93.0% | 58.9% | 72.1% |
| `Colour` | 53.5% | 99.0% | 69.5% |
| `Colour (CoLIE)` | 51.1% | 94.5% | 66.4% |
| `Rettenberger [50]` | 43.8% | 56.1% | 49.2% |
| `Rettenberger [50] (CoLIE)` | 39.6% | 45.2% | 42.2% |
| **Average** | 57.5% | 75.7% | 62.5% |
| **Average (non-CoLIE)** | 59.1% | 79.0% | 64.8% |
| **Average (CoLIE)** | 56.0% | 72.5% | 60.2% |

TABLE 4.4: Precision, Recall, and F1 Score for the overall best setup, tested on the different test datasets. The averages are weighted according to the number of frames in each dataset, as shown in Table A.2.

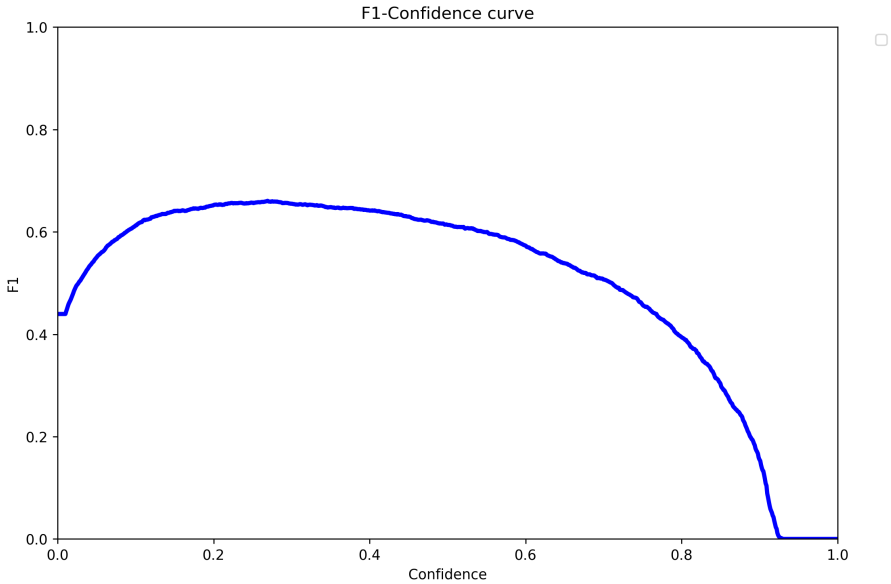


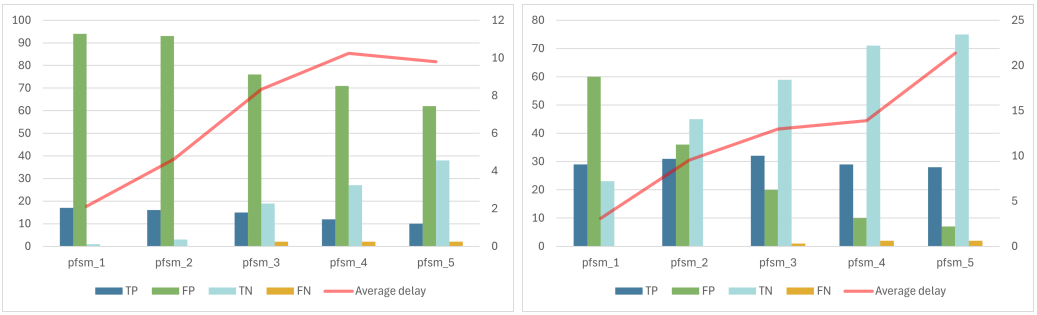

FIGURE 4.2: The F1-curve for the overall best setup, showing the performance for specified confidence thresholds, indicating that the overall best setup achieves the highest F1-score of 66% at a 27% confidence threshold.

The effects of CoLIE on the overall best setup are shown in Table 4.4 and in Appendix B.4, Figures B.12, B.15, B.16, and B.17. Here it is shown that on average CoLIE reduces the F1-score by 4.6%, which results from a 3.1% and 6.5% decrease in precision and recall respectively. Other metrics also show slight decreases when CoLIE is used: AP drops from

0.70 to 0.65, AUC from 0.87 to 0.82, TPs decrease by 67, and TNs decrease by 15.

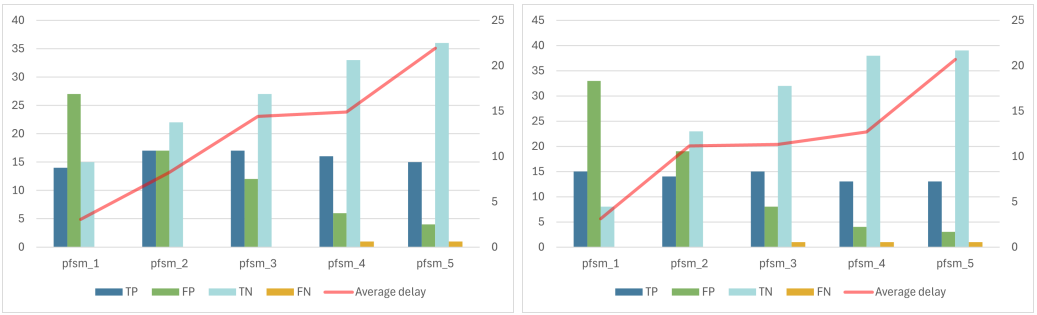
## 4.5 PFSM

To evaluate the theoretical real-world performance, the PFSM was applied to both the baseline Obico model, as well as the best-performing model proposed by this work. The optimal confidence thresholds, as determined in Section 4.1 and 4.4, were used for both models in order to assure that both models performed with an optimal balance between precision and recall. The obtained metrics are depicted in Figures 4.3, 4.4, and 4.5.




(A) Obico (B) Best

FIGURE 4.3: The PFSM for the Obico model versus the overall best setup in this work. Results are taken over all recordings in the test set, with and without CoLIE. The left vertical axis represents the amounts for the bar graph, and the right vertical axis represents the delay in frames for the "Average delay" line graph.

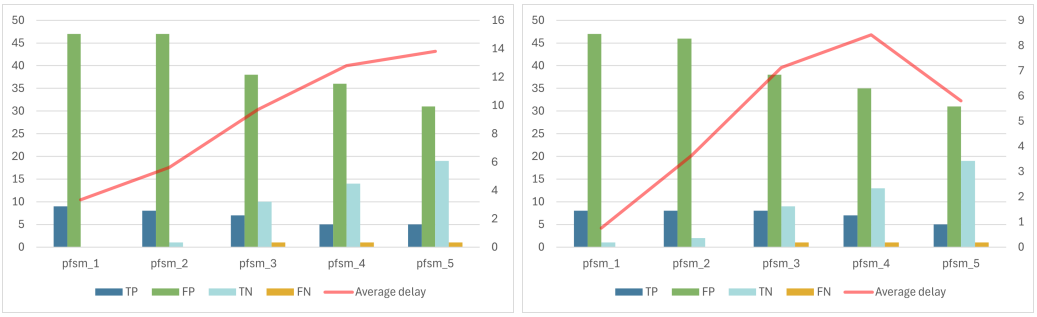



(A) Best Non-CoLIE (B) Best CoLIE

FIGURE 4.4: The PFSM for the overall best setup, split into non-CoLIE and CoLIE. The left vertical axis represents the amounts for the bar graph, and the right vertical axis represents the delay in frames for the "Average delay" line graph.

Figures 4.3a and 4.5 show that Obico obtains an overall high number of FPs, while the TPs remain low and even decreases for increasing thresholds. Furthermore, the TNs start relatively low, but increase with the threshold, a similar trend is observed for the FNs which appear at threshold 3. The delay generally increases with the threshold for the non-CoLIE variant, however, the CoLIE variant scores a lower delay for threshold 5 compared to 4. This effect can also be seen in the overall PFSM depicted in Figure 4.3a, where a slight decrease in delay can be observed between threshold 5 and 4.

The PFSM for the overall best setup is depicted in Figures 4.3b and 4.4. Similar to Obico, a decrease in FPs can be observed while the number of TNs and FNs increases.




(A) Obico Non-CoLIE (B) Obico CoLIE

FIGURE 4.5: The PFSM for the Obico baseline, split into non-CoLIE and CoLIE. The left vertical axis represents the amounts for the bar graph, and the right vertical axis represents the delay in frames for the "Average delay" line graph.

Unlike Obico, the delay increases with the thresholds for all variations, and the TPs slightly increase until threshold 3, after which a decrease can be observed.

Overall, the non-CoLIE PFSM for the overall best setup shows a higher number of TPs, aside from the threshold of 1 frame. It does show a slight increase in FPs for intervals 3, 4, and 5. The average delay remains similar for both CoLIE and non-CoLIE.

# Chapter 5

# Discussion

This chapter serves as an interpretation of the results shown in Chapter 4, which is split into several core findings. Besides this, encountered limitations are described.

## 5.1 Non-effectiveness of CoLIE

Section 4.1 shows that the **Obico** baseline with CoLIE achieves a slight increase in performance for the **Black dataset**. This could contribute to this work's hypothesis of LLIE being able to enhance low-contrast anomaly detection. However, as this difference is very minimal, no clear performance impact can be concluded. Furthermore, the results of the **Colour dataset**, when using CoLIE, even show a decrease in performance. This is not unexpected, as the images already show enough contrast, mitigating the need for CoLIE. The lower performance might suggest that CoLIE enhances certain background features or introduces image artifacts that YOLO incorrectly classifies as anomalies, causing an increase in FPR. Interestingly, however, there is a more substantial performance increase when using CoLIE on the **Rettenberger [50] dataset**, even though those images were not taken in low-contrast conditions. This increase in performance suggests that CoLIE may enhance object detection, not only for low-light environments, but also properly lit ones.

In contrast to the performance influence observed using the Obico baseline, the influence of CoLIE on the **overall best setup**, as defined in Section 4.4, differs. Here, CoLIE negatively impacts the performance on all three test sets (Black, Colour, Rettenberger). These findings contradict the expected outcome of this research, by showing that CoLIE does not improve spaghetti detection in low-contrast conditions. A possible explanation is the already advanced feature extraction capabilities of YOLO11, which are likely able to distinguish the essential features, mitigating the need for CoLIE enhancement. Additionally, adding CoLIE in the preprocessing pipeline might cause over-enhancements, losing certain subtle features that are important for YOLO's object detection. However, further testing would be needed to conclude whether CoLIE is simply redundant when used with YOLO or if it actually removes certain important features.

## 5.2 YOLO11 Performance Depends on Training Setup

Besides the influence of CoLIE on the spaghetti defect detection performance, this work also shows the general performance of YOLO11. By directly comparing YOLO11 to the YOLOv2-based Obico baseline model, the performance influence of the newer YOLO version can be evaluated.

The results of the first experimentation round – as described in Section 4.2 – show that three YOLO configurations are able to surpass Obico when looking at the average F1-score. However, most experiments score worse, which is unexpected when using the newer YOLO11-based approach. As the underlying object detection model is significantly newer, the weaker performance can likely be attributed to the training stage, possibly due to a worse training dataset, or sub-optimal hyperparameters. This is further confirmed by the observed influence of certain training variations, such as the significant negative impact of the high-sample background images, and the positive impact of the frozen backbone. Additionally, Obico performs better on the unseen Rettenberger [50] dataset than any other model in this round, achieving a 31.7% and 39.8% F1-score on non-CoLIE and CoLIE variants respectively. This can likely be attributed to the large variety in the train dataset of the Obico model, compared to the limited variety of data in this work.

To combat this underwhelming performance, the next experimentation rounds focused on optimizing the training setup. This was done using the "more tolerant" dataset, as described in Section 3.2.1, and introducing the medium and large scale YOLO models. These variations showed that YOLO11 is able to outperform the YOLOv2-based Obico baseline model significantly. Specifically, the overall best setup uses the "more tolerant" dataset, the large scale YOLO11 model, backbone freezing, no CoLIE enhancement, no albumentations, and no background images in its training setup. This suggests that YOLO11 is a strong model, being able to achieve high performance in the task of spaghetti defect detection. However, it also shows the high dependence on a proper training setup, otherwise achieving similar or lower scores than the Obico baseline. Furthermore, the overall best setup achieves high performance on the Black test dataset, suggesting that it is also a suitable approach for low-contrast scenarios, without the need for LLIE. Lastly, the introduced alterations also significantly improved the performance on the unseen Rettenberger [50] dataset, even surpassing Obico. This indicates that the overall best setup is able to generalize well on unseen data.

## 5.3 Dataset Variations

As described in Section 3.3, the different experiments made use of a variety of train dataset variations, showing differing effects on detection performance. The second experiment round, as described in Section 4.3, shows that the "more tolerant" dataset generally achieves higher F1-scores than the normal dataset. This is likely caused by the normal dataset causing the model to overfit on unimportant features.

The results of the second experiment round further show that – in contrast to the overall best-scoring model – the second and third-best model show an increase in performance on the Black test dataset when CoLIE is applied, possibly caused by the inclusion of CoLIE-enhanced images in the training step. This inclusion might allow the model to better adapt to CoLIE-enhanced images.

Furthermore, the results of the additional experiments round, as depicted in Figure B.10, show that the variations trained on the Black dataset show a significant increase in performance on the Black test dataset compared to those trained only on the Colour dataset. This highlights the importance of including dark-coloured training data to improve low-contrast detection performance.

Concerning test datasets, interestingly, Table 4.1 and 4.4 show that the performance of Obico and the overall best setup are better on the Black test dataset than the Colour test dataset. The Black test dataset contains more timelapses with artificially induced anomalies – as described in Section 3.2.1 – than the Colour test dataset, which contains

more naturally occurring instances of spaghetti. Artificially creating anomalies might cause clearer spaghetti formation, which would explain the higher performance on the Black test dataset. Additionally, for Obico, the performance on the Rettenberger [50] test dataset is similar as on the Colour test dataset, this is likely because the Rettenberger dataset contains a great variety of objects, similar to the Colour dataset.

## 5.4 Practical Feasibility: PFSM

The introduced PFSM helped to give a clearer view of the performance of Obico and the overall best setup in a real-world system.

Section 4.5 describes the observed PFSM metrics for the Obico baseline. Interestingly, Obico obtains mostly FPs in both the non-CoLIE and CoLIE variations. As the TPs and TNs are significantly outnumbered by the FPs, this approach is not feasible for a real system, as the printer would stop too often without an anomaly occurring. This indicates that the Obico model requires further tweaking or a different system in order to determine whether a print should be stopped. A possible addition could be to look at the number of bounding boxes and their respective confidence scores, as Obico generally predicts multiple bounding boxes with different scores for a blob of spaghetti, as shown in Figure 3.4. A way to further tweak the performance is to change the confidence threshold, as the current threshold was selected using the F1-curve, as depicted in Figure 4.1. This means that the performance of the model was chosen based on a balanced trade-off between precision and recall, however, further tuning of this threshold could further reduce the number of FPs.

For the overall best setup, the PFSM metrics have also been recorded and a comparison between the overall PFSM for the overall best setup versus Obico is depicted in Figure 4.3. This shows that, besides the better performance observed in the other metrics, the overall best setup also shows better performance in a theoretical implementation of the model in a real system. The overall best setup does show slightly higher delays, but those are likely connected to the lower number of FPs, which is probably caused by reduced sensitivity, increasing the detection delay. For the best model, the PFSM metric for thresholds of 3, 4, and 5 frames shows a lower number of FPs than TPs. As this results in a better balance between the two, it indicates a higher feasibility for this theoretical system when using the overall best setup rather than the Obico model, where the FPs are higher than the TPs for each PFSM threshold. However, the number of FPs is still relatively high, which indicates that this approach should still be revised. Similar to the idea for Obico, a possible addition could look into the confidence scores and number of bounding boxes to adjust the total prediction confidence.

## 5.5 Limitations

### 5.5.1 Bambu Lab

Unfortunately, the detection software used on the Bambu Lab FDM printer is closed-source, meaning that it could not be used to obtain objective performance metrics, such as those obtained for Obico and the proposed model. This means that the statement of Bambu Lab's integrated spaghetti detection performing worse on black filament is based on personal experience and subjective observations, rather than concrete numbers.

### 5.5.2 Datasets

The datasets form the basis of multiple limitations in this work. As they were created by using only a single type of printer, the training data does not include data captured in different environments, causing a reduction in generalizability for this model. However, it did show promising performance on the unseen Rettenberger [50] dataset.

Besides this, the dataset was manually labeled, which introduces a subjective aspect to not only the training dataset, but also the test datasets. This limitation is further highlighted by the changed performance of the "more tolerant" dataset versus the normal dataset, which showed that by ignoring very minor defects the performance of the model increased.

### 5.5.3 Low-light Image Enhancement

For this work, only one LLIE algorithm was used in the preprocessing pipeline. It could be possible that other LLIE algorithms might have a different effect on the performance of the object detection model. Besides this, the parameters of CoLIE were fixed and not optimized.

# Chapter 6

# Conclusion

This work investigated the performance of computer vision-based spaghetti defect detection in FDM printing, specifically in low-contrast environments. Furthermore, the effects of including a Low-Light Image Enhancement (LLIE) algorithmm, CoLIE, together with YOLO11 on the anomaly detection performance were shown. Based on the results and discussion, this chapter formulates the answers to the research questions, and later presents the key findings.

## 6.1 Answering Research Questions

**RQ1.1** ***How can YOLO be optimized for detecting spaghetti anomalies in FDM printing?*** To optimize YOLO for detecting spaghetti anomalies in FDM printing, an adequate dataset containing instances of said defect should be provided as training data in several different settings. To reduce overfitting, freezing the backbone layers increases performance on unseen test data. To further increase YOLO's ability to interpret intricate features, the "large" scale model should be used. By analyzing the performance across different test sets while recording the confidence scores, the F1-curve can be used to determine the optimal confidence threshold, in this case 0.27.

**RQ1.2** ***What effect does low-light image enhancement have on spaghetti anomaly detection?*** When used with Obico, CoLIE showed a minimal increase in spaghetti detection performance. However, for the proposed YOLO11-based model, CoLIE negatively impacts the F1-score. This difference is likely caused by the newer YOLO11 in comparison to the older YOLOv2 model used by Obico, where the newer model has strong enough feature extraction capabilities to mitigate the need for additional preprocessing. In this case, the CoLIE preprocessing step might over-enhance certain regions, removing important features, thus decreasing performance.

**RQ1.3** ***How does the proposed model compare to Obico in terms of performance on self-collected and publicly available datasets?*** In the first experimentation round, Obico showed better results on all three datasets – with and without CoLIE enhancement – than most of the variations of the proposed model in this work. This was especially evident on the Rettenberger [50] dataset, likely due to the proposed model only including frames from a single printer setup in its train dataset.

However, by optimizing this model through further experimentation and determining the overall best setup, the proposed model showed improvements in performance, resulting

in significantly better performance than the Obico model, even on the Rettenberger [50] dataset.

This shows that the proposed model is able to outperform current state-of-the-art and that it has a high generalizability to other setups, even without prior training in those settings. This indicates that the proposed model is suitable for a spaghetti defect detection setup, and it could be included in a system such as Obico for improved detection capabilities.

**RQ1.4** ***How do different stopping thresholds impact the trade-off between true positives, false positives, and detection delay?*** When increasing the minimum number of detected frames containing a predicted anomaly before stopping the FDM print, the amount of FPs decreases, and the amount of TNs and FNs increases. The amount of TPs also decreases for the Obico model, however, it shows a peak at PFSM threshold 3 for the overall best setup. Overall, the delay increases when opting for a higher threshold.

This trade-off should be chosen based on the requirements of the system. If the system should not miss any anomalies, the threshold should be set relatively low. However, this might cause an increasing number of FPs. An example case can be if the printer parts are expensive and the costs associated with a "blob of death" would be too high.

If the system should not be unnecessarily interrupted, but it is okay if occasionally an anomaly is missed, then a higher threshold should be chosen. An example case can be if the "blob of death" is no real concern, but the early stopping of defect prints is desired due to reduced material waste.

**Main RQ:** ***How can computer vision techniques be effectively applied to detect spaghetti anomalies in FDM printing, particularly in low-contrast conditions between the filament and background?*** Current state-of-the-art camera-based systems, such the one built into the Bambu Lab X1 Carbon, show suboptimal detection performance for anomalies in low-contrast conditions when using black filament. However, the newest version of YOLO shows a significant improvement in performance, even when the contrast is minimal. Furthermore, for optimal performance, dark-coloured filament prints should be included in the training data.

The tested Low-Light Image Enhancement (LLIE) algorithm in this work, CoLIE, showed no significant impact on the observed performance of the models. On the Obico model, there is a slight increase in F1-score for the Black dataset that is negligible, whereas the proposed model obtains a worse F1-score when CoLIE is applied.

## 6.2 Key findings

This work has shown that Low-Light Image Enhancement (CoLIE) does not improve YOLO11-based anomaly detection for spaghetti defects in FDM printing. In some cases, it even slightly reduced the performance, likely due to the already advanced feature extraction capabilities of the newest YOLO version. This means that modern Deep Learning architectures may already be good enough for anomaly detection in low-light environments, without the need for additional preprocessing.

Additionally, this work has shown that YOLO11 on its own proves to be a promising method to capture such defects, showing a significant improvement in performance when compared to the state-of-the-art. Even when only trained on the data captured for this work, the proposed method is able to generalize relatively well to unseen data, outper-

forming Obico by achieving a 17.5% (non-CoLIE) higher F1-score on the Rettenberger [50] dataset.

These findings contribute to the research in FDM anomaly detection, by showing that modern Deep Learning object detection approaches are able to achieve good performance when used in low-contrast environments. As no prior literature was found specifically addressing FDM anomaly detection for dark filament in low-contrast conditions, this work acts as a first step to explore this challenge. Furthermore, this work presented the Print Failure Stopping Metric (PFSM) metric, providing an easy method to evaluate the theoretical performance of the model in a real system.

# Chapter 7

# Future Work

## 7.1 Extension of this Work

A possible future work could extend on this work and look into the effects of different LLIE algorithms on the anomaly detection performance of YOLO11, as well as the effects of introducing more variety in the training data. Further tweaking of the CoLIE model could also be tested in future work.

## 7.2 Different Anomaly Detection

For further improvements in spaghetti anomaly detection, different anomaly detection algorithms could be applied. Several interesting approaches could be: using differential images to reduce influence from the background, such as in the work of Baumann and Roller [3]; comparison of the printed object to the original 3D model by projecting the 3D model onto the image and comparing it to the camera's observations, such as in the work by Nuchitprasitchai et al. [45]; using autoencoders [23] to learn the representation of proper prints, allowing the model to be trained by only good data. This last one is particularly interesting, as the collected dataset for this work contains a relatively high number of frames without spaghetti, which are currently either partially used or not used at all, depending on whether the variations include "background" images.

## 7.3 Different Hardware Setup

As this work aims to improve anomaly detection in low-contrast environments and software-based enhancement has shown at most minimal improvement, it could be interesting to look into hardware-focused solutions. One of these approaches could include the use of depth-sensing cameras using a structural light scanner setup, as proposed by Charalampous et al. [47], or using ultrasonic or Lidar sensors, optionally combined with the RGB image.

The advantage of this approach would be that depth information does not require a strong contrast in colour between filament and background, while still allowing the printed object to be captured. However, certain depth-sensing approaches still require proper lighting and contrast, such as the stereoscopic camera approach presented by Holzmond and Li [46], indicating that not all depth-sensing hardware is suitable.

# Bibliography


[1] K. Zhou, Ed., *Additive Manufacturing: Materials, Functionalities and Applications*. Cham: Springer International Publishing, 2023. [Online]. Available: https://link.springer.com/10.1007/978-3-031-04721-3

[2] K. Günaydın and H. S. Türkmen, "Common FDM 3D Printing Defects," Apr. 2018. [Online]. Available: https://www.researchgate.net/publication/326146283_Common_FDM_3D_Printing_Defects

[3] F. Baumann and D. Roller, "Vision based error detection for 3D printing processes," *MATEC Web of Conferences*, vol. 59, p. 06003, 2016. [Online]. Available: http://www.matec-conferences.org/10.1051/matecconf/20165906003

[4] F. P. Yean and W. J. Chew, "Detection of Spaghetti and Stringing Failure in 3D Printing," in *2024 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*, Jan. 2024, pp. 293–298. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10475059

[5] K. Jiang, "How To Train a Smart Detective - AKA Behind-The-Scenes Glimpse at Deep Learning in 3D Printing," Aug. 2019. [Online]. Available: https://www.obico.io/blog/2019/08/14/3d-printing-deep-learning-training/

[6] P. Becker, J. Gebert, A. Roennau, F. Finsterwalder, and R. Dillmann, "Online Error Detection in Additive Manufacturing: A Review," in *2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA)*, Apr. 2021, pp. 167–175. [Online]. Available: https://ieeexplore.ieee.org/document/9436729

[7] Bambulab, "Spaghetti Detection." [Online]. Available: https://wiki.bambulab.com/en/knowledge-sharing/Spaghetti_detection

[8] W. Wang, X. Wu, X. Yuan, and Z. Gao, "An Experiment-Based Review of Low-Light Image Enhancement Methods," *IEEE Access*, vol. 8, pp. 87 884–87 917, 2020, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/9088214/?arnumber=9088214

[9] J. Cesarano, "A Review of Robocasting Technology," *MRS Proceedings*, vol. 542, p. 133, 1998. [Online]. Available: http://link.springer.com/10.1557/PROC-542-133

[10] Z. Zhang, Z. Yang, R. D. Sisson, and J. Liang, "Improving ceramic additive manufacturing via machine learning-enabled closed-loop control," *International Journal of Applied Ceramic Technology*, vol. 19, no. 2, pp. 957–967, 2022, _eprint: https://ceramics.onlinelibrary.wiley.com/doi/pdf/10.1111/ijac.13976. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/ijac.13976

[11] B. Stump, "An algorithm for physics informed scan path optimization in additive manufacturing," *Computational Materials Science*, vol. 212, p. 111566, Sep. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927025622003147

[12] P. Sai and S. N. Yeole, *Fused Deposition Modeling - Insights*, Dec. 2014.

[13] L. Ladani and M. Sadeghilaridjani, "Review of Powder Bed Fusion Additive Manufacturing for Metals," *Metals*, vol. 11, no. 9, p. 1391, Sep. 2021, number: 9 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2075-4701/11/9/1391

[14] Y.-a. Jin, H. Li, Y. He, and J.-z. Fu, "Quantitative analysis of surface profile in fused deposition modelling," *Additive Manufacturing*, vol. 8, pp. 142–148, Oct. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214860415000512

[15] A. M. Arefin, N. R. Khatri, N. Kulkarni, and P. F. Egan, "Polymer 3D Printing Review: Materials, Process, and Design Strategies for Medical Applications," *Polymers 2021, Vol. 13, Page 1499*, vol. 13, no. 9, pp. 1499–1499, May 2021, publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2073-4360/13/9/1499/htm

[16] N. Jyeniskhan, A. Keutayeva, G. Kazbek, M. H. Ali, and E. Shehab, "Integrating Machine Learning Model and Digital Twin System for Additive Manufacturing," *IEEE Access*, vol. 11, pp. 71 113–71 126, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10179224/

[17] "Stringing or Oozing | Simplify3D Software." [Online]. Available: https://www.simplify3d.com/resources/print-quality-troubleshooting/stringing-or-oozing/

[18] D. Bell, "Blob of Death," Jan. 2021. [Online]. Available: https://docs.vorondesign.com/community/troubleshooting/120decibell/blob_of_death.html

[19] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, "AN OVERVIEW OF MACHINE LEARNING," in *Machine Learning*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. San Francisco (CA): Morgan Kaufmann, Jan. 1983, pp. 3–23. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080510545500054

[20] B. Mahesh, "Machine Learning Algorithms - A Review," *International Journal of Science and Research (IJSR)*, vol. 9, no. 1, pp. 381–386, Jan. 2020. [Online]. Available: https://www.ijsr.net/archive/v9i1/ART20203995.pdf

[21] M. A. El Mrabet, K. El Makkaoui, and A. Faize, "Supervised Machine Learning: A Survey," in *2021 4th International Conference on Advanced Communication Technologies and Networking (CommNet)*, Dec. 2021, pp. 1–10. [Online]. Available: https://ieeexplore.ieee.org/document/9641998/?arnumber=9641998

[22] M. Khanum, T. Mahboob, W. Imtiaz, H. Abdul Ghafoor, and R. Sehar, "A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance," *International Journal of Computer Applications*, vol. 119, no. 13, pp. 34–39, Jun. 2015. [Online]. Available: http://research.ijcaonline.org/volume119/number13/pxc3904058.pdf

[23] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, Feb. 2020. [Online]. Available: https://doi.org/10.1007/s10994-019-05855-6

[24] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, May 1996, publisher: AI Access Foundation.

[25] X. Ying, "An Overview of Overfitting and its Solutions," *Journal of Physics: Conference Series*, vol. 1168, p. 022022, Feb. 2019. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022

[26] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Jul. 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0

[27] G. Naidu, T. Zuva, and E. M. Sibanda, "A Review of Evaluation Metrics in Machine Learning Algorithms," in *Artificial Intelligence Application in Networks and Systems*, R. Silhavy and P. Silhavy, Eds. Cham: Springer International Publishing, 2023, pp. 15–25.

[28] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, Jul. 2020, pp. 237–242, iSSN: 2157-8702. [Online]. Available: https://ieeexplore.ieee.org/document/9145130/?arnumber=9145130

[29] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, Nov. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405844018332067

[30] L. Ciampiconi, A. Elwood, M. Leonardi, A. Mohamed, and A. Rozza, "A survey and taxonomy of loss functions in machine learning," Nov. 2024, arXiv:2301.05579 [cs]. [Online]. Available: http://arxiv.org/abs/2301.05579

[31] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," Dec. 2015, arXiv:1511.08458. [Online]. Available: http://arxiv.org/abs/1511.08458

[32] J. Koushik, "Understanding Convolutional Neural Networks," May 2016, arXiv:1605.09081 [stat]. [Online]. Available: http://arxiv.org/abs/1605.09081

[33] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, conference Name: IEEE Transactions on Neural Networks and Learning Systems. [Online]. Available: https://ieeexplore.ieee.org/document/9451544/?arnumber=9451544

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," May 2016, arXiv:1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640

[35] G. Jocher and J. Qiu, "Ultralytics YOLO11," 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[36] R. Khanam and M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," Oct. 2024, arXiv:2410.17725. [Online]. Available: http://arxiv.org/abs/2410.17725

[37] A. Ghosh, "YOLO11: Faster Than You Can Imagine!" Oct. 2024. [Online]. Available: https://learnopencv.com/yolo11/

[38] T. Chobola, Y. Liu, H. Zhang, J. A. Schnabel, and T. Peng, "Fast Context-Based Low-Light Image Enhancement via Neural Implicit Representations," Jul. 2024. [Online]. Available: https://arxiv.org/abs/2407.12511v1

[39] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit Neural Representations with Periodic Activation Functions," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 7462–7473. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/53c04118df112c13a8c34b38343b9c10-Abstract.html

[40] J. Liu, Y. Hu, B. Wu, and Y. Wang, "An improved fault diagnosis approach for FDM process with acoustic emission," *Journal of Manufacturing Processes*, vol. 35, pp. 570–579, Oct. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1526612518312714

[41] H. Wu, Y. Wang, and Z. Yu, "In situ monitoring of FDM machine condition via acoustic emission," *The International Journal of Advanced Manufacturing Technology*, vol. 84, no. 5, pp. 1483–1495, May 2016. [Online]. Available: https://doi.org/10.1007/s00170-015-7809-4

[42] Y. Tlegenov, G. S. Hong, and W. F. Lu, "Nozzle condition monitoring in 3D printing," *Robotics and Computer-Integrated Manufacturing*, vol. 54, pp. 45–55, Dec. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S073658451730443X

[43] Y. Tlegenov, W. F. Lu, and G. S. Hong, "A dynamic model for current-based nozzle condition monitoring in fused deposition modelling," *Progress in Additive Manufacturing*, vol. 4, no. 3, pp. 211–223, Sep. 2019. [Online]. Available: https://doi.org/10.1007/s40964-019-00089-3

[44] P. Becker, N. Spielbauer, A. Roennau, and R. Dillmann, "Real-Time In-Situ Process Error Detection in Additive Manufacturing," in *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, Nov. 2020, pp. 426–427. [Online]. Available: https://ieeexplore.ieee.org/document/9287940

[45] S. Nuchitprasitchai, M. Roggemann, and J. M. Pearce, "Factors effecting real-time optical monitoring of fused filament 3D printing," *Progress in Additive Manufacturing*, vol. 2, no. 3, pp. 133–149, Sep. 2017. [Online]. Available: https://doi.org/10.1007/s40964-017-0027-x

[46] O. Holzmond and X. Li, "In situ real time defect detection of 3D printed parts," *Additive Manufacturing*, vol. 17, pp. 135–142, Oct. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214860417301100

[47] P. Charalampous, I. Kostavelis, C. Kopsacheilis, and D. Tzovaras, "Vision-based real-time monitoring of extrusion additive manufacturing processes for automatic manufacturing error detection," *The International Journal of Advanced*

*Manufacturing Technology*, vol. 115, no. 11, pp. 3859–3872, Aug. 2021. [Online]. Available: https://doi.org/10.1007/s00170-021-07419-2

[48] Z. Zhang, I. Fidan, and M. Allen, "Detection of Material Extrusion In-Process Failures via Deep Learning," *Inventions*, vol. 5, no. 3, p. 25, Sep. 2020, number: 3 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2411-5134/5/3/25

[49] J. C. W. Ern and K. V. Jyn, "RECOGNIZING DEFECTS IN FUSED FILAMENT FABRICATED PARTS: A COMPUTER VISION BASED APPROACH," 2021. [Online]. Available: https://www.ftsm.ukm.my/v5/file/research/technicalreport/PS-FTSM-2021-003.pdf

[50] L. Rettenberger, N. Beyer, I. Sieber, and M. Reischl, "Fault Detection in 3D-Printing with Deep Learning," in *2024 IEEE International Conference on Consumer Electronics (ICCE)*, Jan. 2024, pp. 1–4, iSSN: 2158-4001. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10444198

[51] N. B. A. Karna, M. A. P. Putra, S. M. Rachmawati, M. Abisado, and G. A. Sampedro, "Toward Accurate Fused Deposition Modeling 3D Printer Fault Detection Using Improved YOLOv8 With Hyperparameter Optimization," *IEEE Access*, vol. 11, pp. 74 251–74 262, 2023, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10176146

[52] C. R. Yeh, S. Q. Yan, and M. J. Tsai, "YOLOv5 based Intelligent Defect Detection of Dual-color Gradient Object Fabricated by Fused Deposition Modeling Additive Manufacturing System," in *2023 International Automatic Control Conference (CACS)*, Oct. 2023, pp. 1–6, iSSN: 2473-7259. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10326167

[53] J. A. B. Susa, S. V. Militante, A. G. Acoba, R. S. Evangelista, L. L. Lacatan, C. G. Laviña, and B. T. Tanguilig, "A Machine Vision-Based Person Detection Under Low-Illuminance Conditions Using High Dynamic Range Imagery for Visual Surveillance System," in *2022 Third International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*. Bengaluru, India: IEEE, Dec. 2022, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10099673/

[54] D. Peng, W. Ding, and T. Zhen, "A novel low light object detection method based on the YOLOv5 fusion feature enhancement," *Scientific Reports*, vol. 14, no. 1, 2024.

[55] L. Ye and Z. Ma, "LLOD:A Object Detection Method Under Low-Light Condition by Feature Enhancement and Fusion," in *2023 4th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*. Nanjing, China: IEEE, Jun. 2023, pp. 659–662. [Online]. Available: https://ieeexplore.ieee.org/document/10212748/

[56] Y. Feng, S. Hou, H. Lin, Y. Zhu, P. Wu, W. Dong, J. Sun, Q. Yan, and Y. Zhang, "DiffLight: Integrating Content and Detail for Low-light Image Enhancement," 2024, pp. 6143–6152. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2024W/NTIRE/html/Feng_DiffLight_Integrating_Content_and_Detail_for_Low-light_Image_Enhancement_CVPRW_2024_paper.html

[57] Y. Peng, S. Tu, and J. Qian, "Low-light image enhancement network based on luminance prior and depth feature extraction," in *2024 5th International Conference on Computer Vision, Image and Deep Learning (CVIDL)*. Zhuhai, China: IEEE, Apr. 2024, pp. 1092–1096. [Online]. Available: https://ieeexplore.ieee.org/document/10603826/

[58] R. Zhao, M. Xie, X. Feng, X. Su, H. Zhang, and W. Yang, "Content-illumination coupling guided low-light image enhancement network," *Scientific Reports*, vol. 14, no. 1, p. 8456, Apr. 2024, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-024-58965-0

[59] F. Jia, S. Mao, X.-C. Tai, and T. Zeng, "A Variational Model for Nonuniform Low-Light Image Enhancement," *SIAM Journal on Imaging Sciences*, vol. 17, no. 1, pp. 1–30, Mar. 2024, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/full/10.1137/22M1543161

[60] S. D. Thepade and A. Shirbhate, "Visibility Enhancement in Low Light Images with Weighted Fusion of Robust Retinex Model and Dark Channel Prior," in *2020 IEEE Bombay Section Signature Conference (IBSSC)*. Mumbai, India: IEEE, Dec. 2020, pp. 69–73. [Online]. Available: https://ieeexplore.ieee.org/document/9332217/

[61] M. Alavi and M. Kargari, "A new contrast enhancement method for Color dark and low-light images," in *2022 9th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*. Bam, Iran, Islamic Republic of: IEEE, Mar. 2022, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/9756426/

[62] E. H. Land and J. J. McCann, "Lightness and Retinex Theory," *JOSA*, vol. 61, no. 1, pp. 1–11, Jan. 1971, publisher: Optica Publishing Group. [Online]. Available: https://opg.optica.org/josa/abstract.cfm?uri=josa-61-1-1

[63] M. Tkachenko, M. Malyuk, A. Holmanyuk, and N. Liubimov, "Label Studio: Data labeling software," 2020. [Online]. Available: https://github.com/HumanSignal/label-studio

[64] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and Flexible Image Augmentations," *Information*, vol. 11, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/125

# Glossary

**albumentations** A data augmentation package, created by Buslaev et al. [64], natively supported by YOLO, including colour, brightness, and contrast alterations, blurs, and compression, available: https://github.com/albumentations-team/albumentations. 46, 49, 50, 51, 56

**Learned Perceptual Image Patch Similarity (LPIPS)** A metric commonly used for DL image enhancement models. This metric indicates how well the individual feature maps at each layer perceive an input image in comparison to a ground truth image. 36, 37, 38

**mAP@50** The mAP score for bounding boxes with a minimum IoU score of 50%. 16, 31, 34, 36

**mAP@50-95** The mAP calculated at multiple IoU thresholds, ranging from 50% to 95%. This gives an indication of the model's capabilities across increasing levels of detection difficulty. 16, 34, 36

**Mean Square Error (MSE)** Mean of the squared difference between the predicted and ground thruth value. Similar to MAE, commonly used as a metric or loss function.. 19, 29, 69

**Mean Average Error (MAE)** Mean of the average difference between the predicted and ground truth value. Commonly used as a metric or loss function. 19, 33, 69

**Peak Signal-to-Noise Ratio (PSNR)** A metric used for image enhancement tasks, where the ratio between the maximum possible pixel intensity value and the noise is described in decibel. Here, the noise is described by the MSE. 36, 37, 38

**Root Mean Square Error (RMSE)** The root of the MSE, commonly used as a metric or loss function. 19, 33

**Structural Similarity Index Measure (SSIM)** A metric used for image enhancement tasks, where the structural similarity is measured between a reference image and an enhanced image. 36, 37, 38

# Acronyms

**ABS** Acrylonitrile Butadiene Styrene. 7

**AM** Additive Manufacturing. 4, 6, 7, 33

**AP** Average Precision. 14, 16, 52, 74, 75, 81, 82

**AUC** Area Under Curve. 14, 15, 53, 74, 76, 81, 82

**BCE** Binary Cross-Entropy. 27

**CAD** Computer Aided Design. 33

**CIoU** Complete Intersection over Union. 27

**CNN** Convolutional Neural Network. 17, 20, 31, 33, 34

**CoBl** Colour & Black. 40, 46, 49, 50, 51

**CoLIE** Context-based Low-light Image Enhancement. 1, 4, 27, 28, 29, 30, 36, 38, 39, 43, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 75, 76, 80, 82

**CSP** Cross Stage Partial. 25

**CV** Computer Vision. 1, 31, 32, 34, 35, 37

**DFL** Distribution Focal Loss. 27

**DL** Deep Learning. 1, 17, 20, 33, 37, 60, 61, 69

**FDM** Fused Deposition Modeling. 1, 4, 5, 6, 7, 8, 33, 40, 59, 60, 61

**FFF** Fused Filament Fabrication. 6

**FIP-AM@UT** Fraunhofer Innovation Platform for Advanced Manufacturing at the University of Twente. 1, 40

**FN** False Negative. 13, 14, 43, 44, 53, 60

**FP** False Positive. 13, 14, 40, 43, 44, 46, 53, 54, 57, 60

**FPR** False Positive Rate. 13, 14, 15, 32, 44, 55, 74, 81

**HSV** Hue, Saturation, and Value. 27, 28, 30

**IoU** Intersection over Union. 16, 22, 23, 27, 69

**LLIE** Low-Light Image Enhancement. 1, 4, 6, 27, 28, 35, 37, 38, 39, 55, 56, 58, 59, 60, 62

**mAP** Mean Average Precision. 14, 16, 31, 35, 36, 37, 69

**ML** Machine Learning. 1, 6, 10, 11, 13, 16, 19, 29, 33, 34

**MLP** Multi-Layer Perceptron. 17, 18

**NIR** Neural Implicit Representation. 28, 29, 36

**NMS** Non-Maximum Suppression. 22

**NN** Neural Network. 16, 17, 19, 20, 28, 39

**PBF** Powder Bed Fusion. 6, 33

**PCA** Principal Component Analysis. 11

**PFSM** Print Failure Stopping Metric. 1, 5, 44, 45, 53, 54, 57, 60, 61

**PLA** Polylactic Acid. 7

**PR** Precision-Recall. 14, 16, 44, 74, 75, 81, 82

**PSA** Position-Sensitive Attention. 25

**ReLU** Rectified Linear Unit. 19, 28

**RNN** Recurrent Neural Network. 17

**ROC** Receiver Operating Characteristic. 14, 15, 74, 76, 81, 82

**ROI** Region Of Interest. 31, 32, 34, 39

**SIREN** Sinusoidal Representation Network. 28

**SVM** Support Vector Machine. 11, 31, 32, 33

**TN** True Negative. 13, 43, 44, 53, 57, 60

**TP** True Positive. 13, 43, 44, 45, 53, 54, 57, 60

**TV** Total Variation. 29

**YOLO** You Only Look Once. 1, 4, 5, 20, 21, 22, 23, 24, 25, 27, 31, 32, 34, 35, 36, 37, 39, 40, 43, 46, 47, 50, 51, 55, 56, 59, 60, 62, 69

# Appendix A

# Datasets

| Dataset Name | Anomaly | | Background | | Total | |
|---|---|---|---|---|---|---|
| | # Train | # Val | # Train | # Val | # Train | # Val |
| **Colour more background** | 710 | 148 | 4718 | 667 | 5428 | 815 |
| **Colour** | 710 | 148 | 245 | 44 | 955 | 192 |
| **Colour more tolerant** | 482 | 139 | 255 | 38 | 737 | 177 |
| **Black more background** | 999 | 380 | 1613 | 25 | 2612 | 405 |
| **Black** | 999 | 380 | 806 | 15 | 1805 | 395 |
| **Black more tolerant** | 508 | 216 | 719 | 99 | 1227 | 315 |
| **COBL more background** | 1709 | 528 | 6331 | 692 | 8040 | 1220 |
| **COBL** | 1709 | 528 | 1051 | 59 | 2760 | 587 |
| **COBL more tolerant** | 990 | 355 | 974 | 137 | 1964 | 492 |

Table A.1: Train dataset composition.

| Dataset Name | # Anomaly | # Background | # Total |
|---|---|---|---|
| **Colour** | 308 | 2504 | 2812 |
| **Black** | 542 | 589 | 1131 |
| **Rettenberger [50]** | 228 | 1481 | 1709 |

Table A.2: Test dataset composition.

# Appendix B

# Experiment results

## B.1 Baseline (Obico)

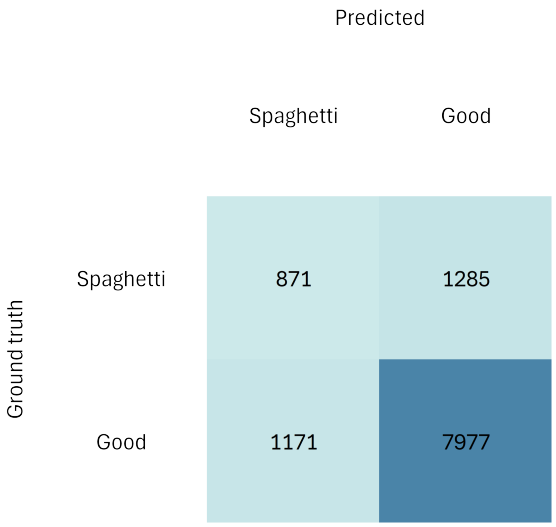



FIGURE B.1: Overall confusion matrix for the Obico baseline.

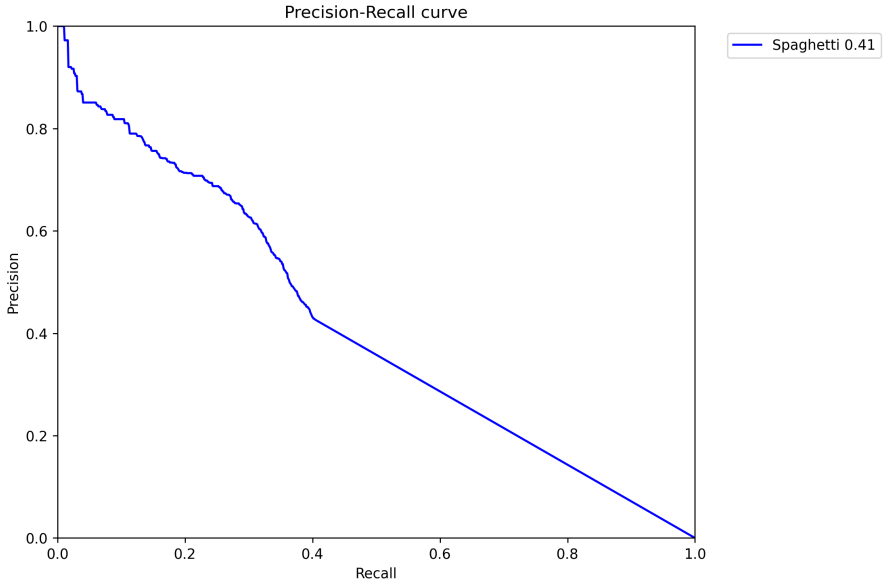



FIGURE B.2: The PR-curve for the Obico baseline shows the trade-off between precision and recall, achieving an AP of 0.41.

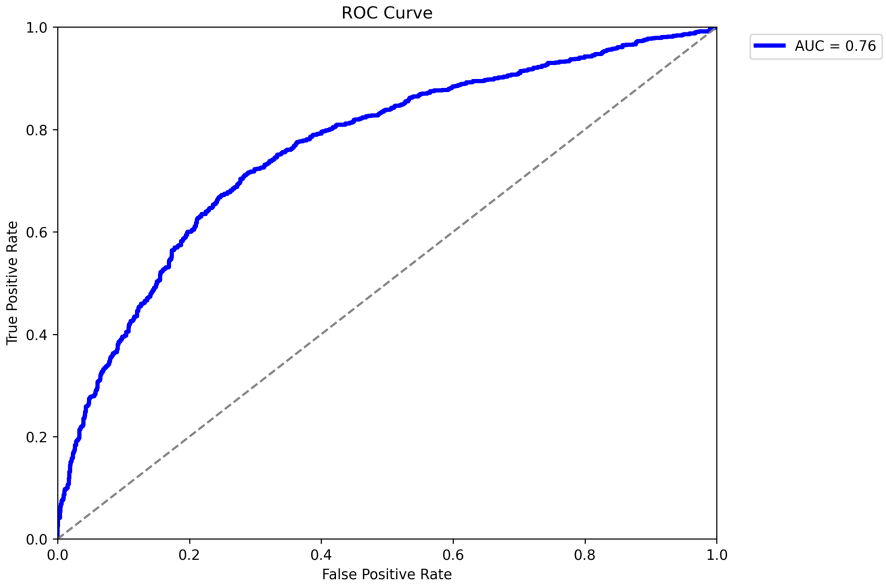
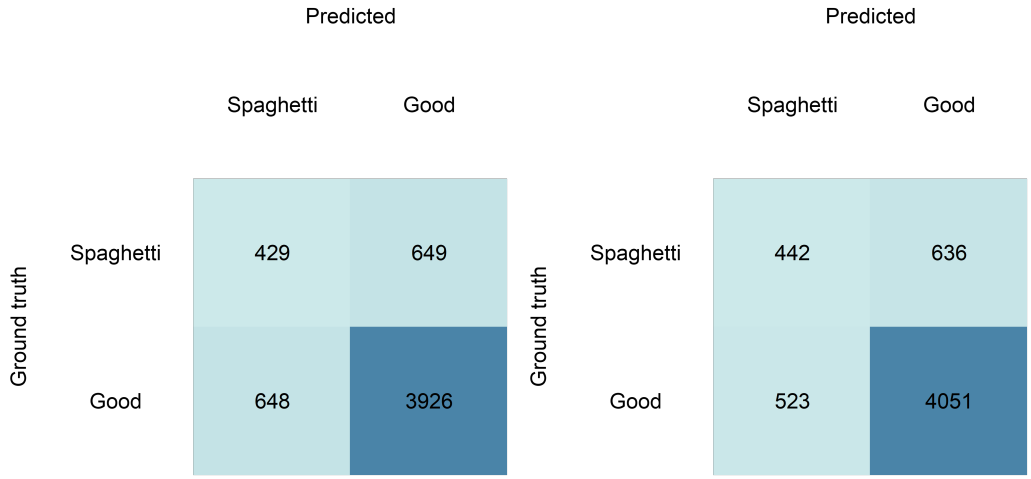
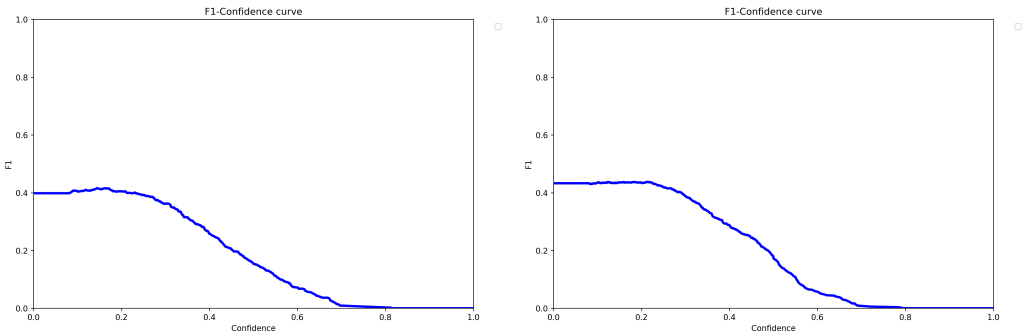



FIGURE B.3: The ROC-curve for the Obico baseline shows the trade-off between false positives and true positives (by plotting the FPR against recall), achieving an AUC of 0.76.



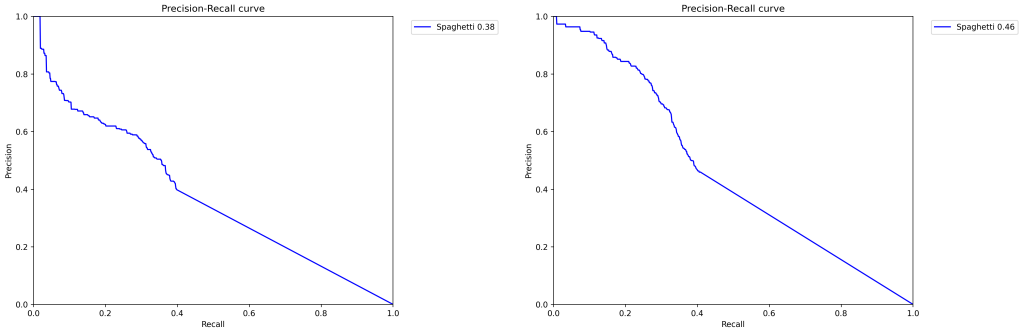

(A) Non-CoLIE (B) CoLIE

FIGURE B.4: Confusion matrices for the Obico baseline, split into non-CoLIE and CoLIE.



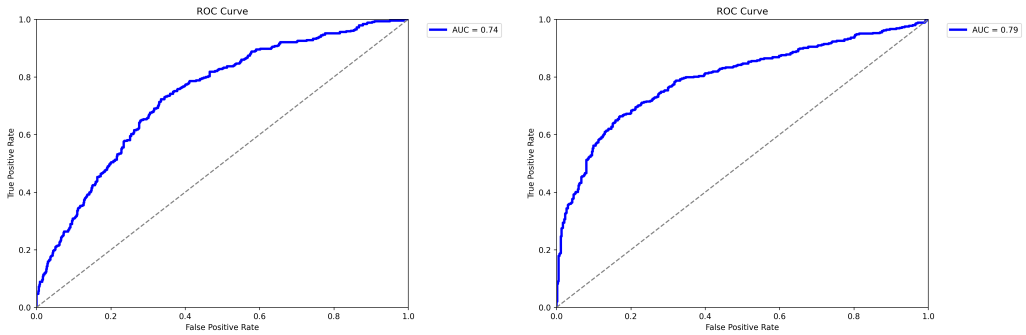

(A) Non-CoLIE (B) CoLIE

FIGURE B.5: The F1-curves for the Obico baseline, split into non-CoLIE and CoLIE. This shows that Obico achieves the highest F1-score of 42% at a 14% confidence threshold without CoLIE, whereas it achieves a 44% F1-score at a 21% confidence threshold.




(A) Non-CoLIE, AP = 0.38. (B) CoLIE, AP = 0.46

FIGURE B.6: The PR-curves for the Obico baseline, split into non-CoLIE and CoLIE.




(A) Non-CoLIE, AUC = 0.74. (B) CoLIE, AUC = 0.79.

FIGURE B.7: The ROC-curves for the Obico baseline, split into non-CoLIE and CoLIE.

## B.2 First experiment

| Train Dataset | Background | CoLIE | Albumentations | Frozen backbone | Test Black F1 Score | Test Black CoLIE F1 Score | Test Colour F1 Score | Test Colour CoLIE F1 Score | Test Rettenberger F1 Score | Test Rettenberger CoLIE F1 Score | Average F1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **CoBl** | **Low-sample** | ☑ | ☐ | ☑ | 57.8% | 56.0% | 70.4% | 45.3% | 22.6% | 18.1% | **45.0%** |
| Colour | Low-sample | ☑ | ☑ | ☑ | 58.7% | 59.8% | 43.1% | 44.5% | 22.2% | 18.5% | 41.1% |
| Colour | - | ☑ | ☐ | ☑ | **67.3%** | **67.2%** | 35.4% | 31.9% | 23.2% | 18.5% | 40.6% |
| Obico | - | ☐ | ☐ | ☐ | 56.8% | 57.8% | 27.8% | 24.8% | **31.7%** | **39.8%** | 39.8% |
| CoBl | Low-sample | ☐ | ☑ | ☑ | 50.7% | 50.8% | **74.4%** | 35.9% | 11.0% | 11.0% | 39.0% |
| Colour | - | ☑ | ☑ | ☑ | 66.0% | 60.1% | 37.7% | 33.4% | 22.0% | 14.4% | 38.9% |
| CoBl | Low-sample | ☑ | ☑ | ☑ | 53.5% | 51.8% | 60.1% | 45.4% | 12.2% | 9.6% | 38.8% |
| CoBl | - | ☑ | ☑ | ☑ | 47.3% | 45.4% | 48.5% | 43.5% | 24.4% | 16.6% | 37.6% |
| CoBl | - | ☑ | ☑ | ☐ | 44.7% | 51.7% | 52.0% | **51.6%** | 8.6% | 14.6% | 37.2% |
| Black | High-sample | ☐ | ☑ | ☑ | 59.3% | 49.4% | 39.6% | 23.6% | 30.8% | 16.7% | 36.6% |
| CoBl | - | ☐ | ☑ | ☑ | 57.6% | 55.0% | 57.9% | 26.9% | 11.4% | 9.5% | 36.4% |
| CoBl | High-sample | ☑ | ☐ | ☑ | 53.8% | 51.2% | 59.1% | 43.1% | 2.6% | 0.8% | 35.1% |
| CoBl | - | ☑ | ☐ | ☑ | 49.4% | 58.0% | 60.6% | 31.5% | 0.8% | 0.0% | 33.4% |
| Colour | - | ☐ | ☐ | ☑ | 55.4% | 46.2% | 25.6% | 37.7% | 11.7% | 11.4% | 31.3% |
| CoBl | - | ☐ | ☑ | ☐ | 47.5% | 46.0% | 37.9% | 30.7% | 1.6% | 6.9% | 28.4% |
| Black | - | ☐ | ☑ | ☑ | 53.9% | 44.0% | 30.2% | 17.6% | 13.2% | 9.8% | 28.1% |
| CoBl | - | ☑ | ☐ | ☐ | 57.1% | 44.6% | 24.6% | 18.9% | 8.3% | 9.4% | 27.2% |
| CoBl | High-sample | ☐ | ☐ | ☑ | 42.0% | 36.7% | 57.1% | 25.9% | 0.0% | 0.0% | 27.0% |
| Black | - | ☑ | ☑ | ☑ | 52.9% | 39.6% | 27.8% | 23.9% | 7.7% | 6.8% | 26.5% |
| CoBl | Low-sample | ☐ | ☐ | ☑ | 39.5% | 38.3% | 43.2% | 27.0% | 8.4% | 0.0% | 26.1% |
| Black | Low-sample | ☑ | ☑ | ☑ | 42.5% | 43.3% | 32.5% | 16.0% | 11.8% | 10.1% | 26.1% |
| CoBl | High-sample | ☐ | ☑ | ☑ | 48.4% | 45.1% | 36.0% | 18.9% | 0.0% | 0.0% | 24.7% |
| Colour | Low-sample | ☐ | ☑ | ☑ | 41.3% | 33.0% | 35.3% | 19.2% | 11.9% | 7.1% | 24.6% |
| Colour | Low-sample | ☐ | ☐ | ☑ | 34.1% | 29.8% | 36.3% | 31.4% | 5.9% | 5.0% | 23.7% |
| CoBl | Low-sample | ☑ | ☐ | ☐ | 30.6% | 33.7% | 40.7% | 30.5% | 2.5% | 0.9% | 23.2% |
| Colour | Low-sample | ☑ | ☐ | ☑ | 30.4% | 27.4% | 31.0% | 23.9% | 21.2% | 3.8% | 22.9% |
| Black | High-sample | ☑ | ☐ | ☑ | 46.2% | 45.4% | 26.1% | 11.6% | 5.7% | 2.5% | 22.9% |
| Black | High-sample | ☑ | ☑ | ☑ | 52.4% | 45.1% | 19.2% | 7.2% | 6.9% | 6.3% | 22.8% |
| CoBl | High-sample | ☑ | ☑ | ☑ | 51.4% | 42.2% | 24.7% | 12.0% | 4.3% | 1.7% | 22.7% |
| CoBl | - | ☐ | ☐ | ☐ | 43.5% | 38.0% | 33.7% | 14.8% | 0.9% | 0.8% | 21.9% |
| CoBl | Low-sample | ☑ | ☑ | ☐ | 28.2% | 36.2% | 32.4% | 32.1% | 2.3% | 0.0% | 21.9% |
| Colour | - | ☐ | ☑ | ☑ | 36.6% | 24.8% | 29.1% | 26.1% | 8.9% | 0.8% | 21.1% |
| Black | Low-sample | ☑ | ☐ | ☑ | 38.1% | 38.5% | 22.7% | 20.7% | 2.2% | 0.9% | 20.5% |
| Black | High-sample | ☐ | ☐ | ☑ | 42.7% | 39.6% | 24.4% | 10.5% | 0.8% | 5.0% | 20.5% |
| CoBl | High-sample | ☑ | ☑ | ☐ | 30.4% | 7.5% | 47.2% | 34.7% | 0.8% | 0.8% | 20.2% |
| CoBl | - | ☐ | ☐ | ☑ | 31.8% | 12.9% | 42.4% | 21.2% | 5.1% | 2.4% | 19.3% |
| Black | Low-sample | ☐ | ☑ | ☐ | 39.4% | 39.1% | 11.0% | 10.6% | 6.9% | 2.6% | 18.2% |
| Black | Low-sample | ☐ | ☑ | ☑ | 47.3% | 39.8% | 13.3% | 1.9% | 2.7% | 2.7% | 17.9% |
| Black | Low-sample | ☐ | ☐ | ☑ | 36.5% | 36.0% | 15.7% | 3.3% | 7.6% | 6.8% | 17.7% |
| Black | - | ☐ | ☐ | ☐ | 48.3% | 36.6% | 9.3% | 6.7% | 0.0% | 4.0% | 17.5% |
| Black | - | ☐ | ☑ | ☐ | 42.0% | 23.9% | 11.3% | 0.6% | 11.4% | 13.7% | 17.2% |
| Colour | - | ☑ | ☐ | ☐ | 10.7% | 6.3% | 36.3% | 31.6% | 10.0% | 7.4% | 17.0% |
| Black | - | ☐ | ☐ | ☑ | 38.2% | 31.7% | 12.0% | 4.5% | 7.0% | 5.3% | 16.4% |
| Colour | Low-sample | ☐ | ☐ | ☐ | 13.7% | 5.0% | 41.3% | 20.6% | 2.6% | 14.3% | 16.3% |
| Black | - | ☑ | ☐ | ☑ | 27.7% | 19.6% | 24.6% | 10.0% | 7.4% | 3.0% | 15.4% |
| CoBl | High-sample | ☐ | ☑ | ☐ | 25.0% | 17.2% | 33.0% | 13.8% | 0.0% | 0.0% | 14.8% |
| CoBl | Low-sample | ☐ | ☑ | ☐ | 32.5% | 13.1% | 29.7% | 6.0% | 2.0% | 0.0% | 13.9% |
| Black | High-sample | ☐ | ☐ | ☐ | 37.9% | 37.7% | 0.0% | 0.0% | 1.8% | 0.6% | 13.0% |
| Colour | High-sample | ☑ | ☑ | ☑ | 30.4% | 15.3% | 13.9% | 5.7% | 4.3% | 0.9% | 11.7% |
| Colour | - | ☐ | ☑ | ☐ | 19.2% | 0.7% | 8.5% | 5.1% | 16.4% | 16.3% | 11.0% |
| Colour | - | ☑ | ☑ | ☐ | 16.0% | 7.0% | 25.4% | 11.2% | 2.2% | 4.4% | 11.0% |
| CoBl | - | ☑ | ☑ | ☐ | 18.2% | 13.1% | 10.9% | 15.6% | 3.3% | 4.9% | 11.0% |
| CoBl | Low-sample | ☐ | ☐ | ☐ | 20.5% | 5.9% | 33.8% | 2.9% | 2.6% | 0.0% | 10.9% |
| Black | Low-sample | ☑ | ☐ | ☐ | 24.6% | 26.0% | 9.4% | 2.7% | 1.7% | 0.0% | 10.7% |
| Colour | - | ☐ | ☐ | ☐ | 31.0% | 5.3% | 12.7% | 8.9% | 2.0% | 4.4% | 10.7% |
| Black | - | ☑ | ☑ | ☐ | 26.5% | 29.7% | 3.7% | 3.4% | 0.8% | 0.0% | 10.7% |
| Colour | High-sample | ☐ | ☐ | ☑ | 14.4% | 13.7% | 17.2% | 13.3% | 0.0% | 0.0% | 9.8% |
| Black | Low-sample | ☐ | ☐ | ☐ | 33.2% | 22.0% | 0.0% | 0.0% | 0.0% | 0.0% | 9.2% |
| Black | High-sample | ☐ | ☑ | ☐ | 18.4% | 31.3% | 0.6% | 0.0% | 2.5% | 1.6% | 9.1% |
| Colour | High-sample | ☐ | ☑ | ☑ | 10.8% | 7.5% | 15.2% | 12.7% | 6.0% | 0.0% | 8.7% |
| Black | - | ☑ | ☐ | ☐ | 24.0% | 11.8% | 10.2% | 2.1% | 1.7% | 0.0% | 8.3% |
| Black | High-sample | ☑ | ☑ | ☐ | 22.3% | 19.9% | 2.5% | 0.0% | 3.2% | 0.9% | 8.1% |
| Black | High-sample | ☑ | ☐ | ☐ | 21.7% | 22.3% | 1.9% | 0.0% | 0.0% | 0.0% | 7.6% |
| Black | Low-sample | ☑ | ☑ | ☐ | 15.0% | 21.7% | 1.3% | 0.0% | 0.0% | 4.1% | 7.0% |
| Colour | Low-sample | ☑ | ☑ | ☐ | 3.3% | 6.4% | 15.2% | 14.9% | 0.0% | 0.0% | 6.6% |
| Colour | Low-sample | ☐ | ☑ | ☐ | 18.4% | 0.7% | 14.0% | 4.8% | 0.0% | 0.0% | 6.3% |
| Colour | High-sample | ☑ | ☐ | ☑ | 13.4% | 3.6% | 13.9% | 5.7% | 0.0% | 0.9% | 6.2% |
| CoBl | High-sample | ☐ | ☐ | ☐ | 11.1% | 7.8% | 5.2% | 1.3% | 0.0% | 0.0% | 4.2% |
| Colour | Low-sample | ☑ | ☐ | ☐ | 4.7% | 1.5% | 15.0% | 1.3% | 0.0% | 0.0% | 3.7% |
| CoBl | High-sample | ☑ | ☐ | ☐ | 8.1% | 5.4% | 8.6% | 0.0% | 0.0% | 0.0% | 3.7% |
| Colour | High-sample | ☐ | ☑ | ☐ | 0.0% | 0.0% | 1.9% | 0.0% | 0.0% | 0.0% | 0.3% |
| Colour | High-sample | ☐ | ☐ | ☐ | 0.7% | 0.0% | 0.0% | 0.0% | 0.8% | 0.0% | 0.3% |
| Colour | High-sample | ☑ | ☑ | ☐ | 0.0% | 0.0% | 0.0% | 0.0% | 1.5% | 0.0% | 0.2% |
| Colour | High-sample | ☑ | ☐ | ☐ | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |

FIGURE B.8: F1-scores achieved by the train setups tested in the first experiment. The highest F1-scores are highlighted with bold text. The Obico baseline is marked with red text.

## B.3 Second experiment

| Train Dataset | More tolerant? | YOLO | CoLIE | Albumentations | Test Black F1 Score | Test Black CoLIE F1 Score | Test Colour F1 Score | Test Colour CoLIE F1 Score | Test Rettenberger F1 Score | Test Rettenberger CoLIE F1 Score | Average F1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **CoBl** | ☑ | **l** | **-** | ☐ | 76.8% | 72.1% | 69.5% | 66.4% | **49.2%** | **42.2%** | **62.7%** |
| CoBl | ☑ | m | 100 | ☐ | 67.1% | 75.2% | 63.8% | **70.0%** | 43.1% | 39.8% | 59.8% |
| CoBl | ☑ | m | 20 | ☐ | 69.8% | 72.6% | **71.9%** | 63.0% | 39.2% | 33.1% | 58.3% |
| CoBl | ☑ | s | 20 | ☐ | 73.5% | 66.9% | 64.0% | 47.8% | 50.0% | 25.5% | 54.6% |
| CoBl | ☐ | l | - | ☐ | **78.2%** | **78.1%** | 65.1% | 59.7% | 30.4% | 10.9% | 53.7% |
| CoBl | ☑ | l | 20 | ☐ | 75.4% | 74.9% | 55.2% | 51.4% | 39.7% | 21.2% | 53.0% |
| CoBl | ☑ | s | 20 | ☑ | 64.2% | 60.1% | 67.9% | 62.7% | 22.0% | 19.1% | 49.3% |
| CoBl | ☑ | l | 100 | ☐ | 63.2% | 69.3% | 43.9% | 52.4% | 32.0% | 21.7% | 47.1% |
| CoBl | ☐ | m | - | ☐ | 71.8% | 72.7% | 67.2% | 42.0% | 19.3% | 8.9% | 47.0% |
| CoBl | ☐ | m | 20 | ☐ | 62.0% | 65.9% | 50.9% | 39.3% | 49.1% | 8.9% | 46.0% |
| CoBl | ☑ | s | 100 | ☐ | 77.9% | 76.7% | 45.4% | 37.0% | 18.3% | 13.7% | 44.8% |
| CoBl | ☐ | l | 20 | ☐ | 49.6% | 45.9% | 54.7% | 45.5% | 38.1% | 29.7% | 43.9% |
| CoBl | ☑ | m | - | ☐ | 66.3% | 64.6% | 49.7% | 39.7% | 25.5% | 12.7% | 43.1% |
| CoBl | ☐ | l | 100 | ☐ | 52.8% | 60.5% | 43.9% | 45.7% | 31.1% | 24.0% | 43.0% |
| CoBl | ☐ | m | 100 | ☐ | 48.8% | 60.9% | 41.9% | 53.6% | 33.3% | 16.3% | 42.4% |
| CoBl | ☑ | s | - | ☐ | 75.4% | 72.6% | 28.2% | 29.0% | 26.8% | 20.5% | 42.1% |
| CoBl | ☑ | s | 100 | ☑ | 64.6% | 64.8% | 45.7% | 38.0% | 25.5% | 13.7% | 42.1% |
| Obico | ☐ | - | - | ☐ | 56.8% | 57.8% | 27.8% | 24.8% | 31.7% | 39.8% | 39.8% |
| CoBl | ☐ | s | - | ☑ | 57.9% | 51.2% | 66.7% | 29.3% | 20.1% | 2.4% | 37.9% |
| CoBl | ☑ | s | - | ☑ | 70.9% | 66.5% | 29.7% | 27.5% | 17.8% | 14.9% | 37.9% |
| CoBl | ☐ | s | 20 | ☑ | 55.0% | 52.6% | 50.4% | 38.5% | 11.0% | 8.2% | 35.9% |
| CoBl | ☑ | l | 100 | ☑ | 61.7% | 62.0% | 27.4% | 27.3% | 20.6% | 15.3% | 35.7% |
| CoBl | ☑ | m | - | ☑ | 45.5% | 67.3% | 22.5% | 24.7% | 22.0% | 30.0% | 35.3% |
| CoBl | ☐ | s | - | ☐ | 56.8% | 49.5% | 60.4% | 39.4% | 2.8% | 2.3% | 35.2% |
| CoBl | ☐ | s | 100 | ☑ | 49.0% | 52.2% | 31.2% | 30.4% | 25.3% | 9.6% | 32.9% |
| CoBl | ☐ | s | 20 | ☐ | 47.5% | 41.2% | 63.4% | 35.3% | 0.0% | 0.0% | 31.2% |
| CoBl | ☑ | l | - | ☑ | 46.0% | 64.7% | 22.1% | 19.2% | 15.1% | 16.1% | 30.5% |
| CoBl | ☐ | s | 100 | ☐ | 40.2% | 44.2% | 36.0% | 33.5% | 19.6% | 3.3% | 29.5% |
| CoBl | ☑ | l | 20 | ☑ | 49.9% | 52.9% | 35.5% | 24.9% | 6.0% | 1.2% | 28.4% |
| CoBl | ☑ | m | 20 | ☑ | 43.7% | 46.5% | 24.4% | 15.7% | 21.0% | 17.6% | 28.2% |
| CoBl | ☐ | l | 20 | ☑ | 42.5% | 40.5% | 40.7% | 38.1% | 2.7% | 0.0% | 27.4% |
| CoBl | ☑ | m | 100 | ☑ | 38.9% | 35.1% | 41.1% | 39.0% | 4.7% | 4.7% | 27.2% |
| CoBl | ☐ | m | 20 | ☑ | 40.9% | 32.5% | 41.8% | 34.9% | 3.2% | 2.2% | 25.9% |
| CoBl | ☐ | l | 100 | ☑ | 40.5% | 36.6% | 28.0% | 27.6% | 7.7% | 11.6% | 25.3% |
| CoBl | ☐ | m | - | ☑ | 42.4% | 38.2% | 29.7% | 21.6% | 0.0% | 0.9% | 22.1% |
| CoBl | ☐ | m | 100 | ☑ | 28.8% | 42.1% | 24.9% | 31.3% | 3.6% | 1.4% | 22.0% |
| CoBl | ☐ | l | - | ☑ | 38.4% | 42.2% | 27.0% | 14.7% | 5.5% | 3.0% | 21.8% |

FIGURE B.9: F1-scores achieved by the train setups tested in the second experiment. The highest F1-scores are highlighted with bold text.

## B.4 Additional experiments

| | | | | | | Test Black | Test Black CoLIE | Test Colour | Test Colour CoLIE | Test Rettenberger | Test Rettenberger CoLIE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Train Dataset | More tolerant | YOLO | CoLIE | Albumentations | Background | F1 Score | F1 Score | F1 Score | F1 Score | F1 Score | F1 Score | Average F1 |
| **CoBl** | ☑ | l | - | ☐ | - | **76.8%** | 72.1% | 69.5% | 66.4% | **49.2%** | **42.2%** | **62.7%** |
| CoBl | ☑ | m | 100 | ☐ | - | 67.1% | 75.2% | 63.8% | **70.0%** | 43.1% | 39.8% | 59.8% |
| Cobl | ☑ | m | 20 | ☐ | - | 69.8% | 72.6% | **71.9%** | 63.0% | 39.2% | 33.1% | 58.3% |
| CoBl | ☑ | m | 20 | ☐ | Low-sample | 58.7% | 59.2% | 45.8% | 39.4% | 45.8% | 34.0% | 47.2% |
| Black | ☑ | m | 100 | ☐ | - | 76.5% | **76.5%** | 36.8% | 36.0% | 26.8% | 23.9% | 46.1% |
| Black | ☑ | m | 20 | ☐ | - | 71.5% | 68.9% | 48.2% | 36.8% | 26.8% | 12.3% | 44.1% |
| Black | ☑ | l | - | ☐ | - | 76.6% | 67.3% | 60.1% | 46.3% | 2.4% | 0.8% | 42.3% |
| Black | ☑ | l | - | ☐ | Low-sample | 64.1% | 56.7% | 85.5% | 23.8% | 8.1% | 11.9% | 41.7% |
| Colour | ☑ | m | 100 | ☐ | - | 45.6% | 49.1% | 33.8% | 42.2% | 46.5% | 26.9% | 40.7% |
| Colour | ☑ | m | 100 | ☐ | Low-sample | 54.9% | 58.1% | 56.7% | 56.8% | 11.5% | 3.8% | 40.3% |
| Obico | ☐ | - | - | ☐ | - | 56.8% | 57.8% | 27.8% | 24.8% | 31.7% | 39.8% | 39.8% |
| CoBl | ☑ | l | - | ☐ | Low-sample | 38.6% | 36.5% | 60.5% | 49.6% | 28.6% | 21.1% | 39.2% |
| Colour | ☑ | m | 20 | ☐ | Low-sample | 37.7% | 46.2% | 43.9% | 44.0% | 43.0% | 19.0% | 39.0% |
| CoBl | ☑ | m | 100 | ☐ | Low-sample | 44.3% | 53.8% | 52.5% | 58.6% | 9.6% | 5.2% | 37.4% |
| Colour | ☑ | m | 20 | ☐ | - | 33.2% | 45.1% | 41.6% | 47.4% | 24.5% | 22.5% | 35.7% |
| Colour | ☑ | l | - | ☐ | Low-sample | 47.1% | 47.9% | 56.9% | 50.6% | 0.0% | 1.6% | 34.0% |
| Colour | ☑ | l | - | ☐ | - | 40.9% | 36.4% | 39.3% | 30.8% | 1.7% | 2.6% | 25.3% |
| Black | ☑ | m | 20 | ☐ | Low-sample | 35.3% | 35.5% | 28.1% | 23.4% | 3.2% | 1.6% | 21.2% |
| Black | ☑ | m | 100 | ☐ | Low-sample | 29.8% | 43.1% | 21.8% | 19.5% | 2.3% | 0.7% | 19.5% |

FIGURE B.10: F1-scores achieved by the additional variations applied to the highest-scoring setups in the second experiment. The highest F1-scores are highlighted with bold text.

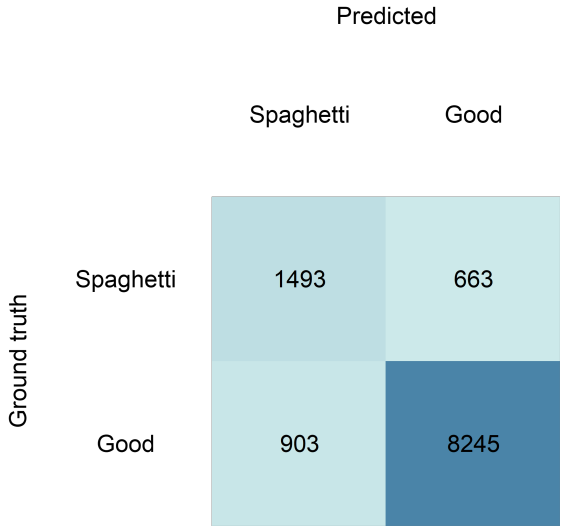
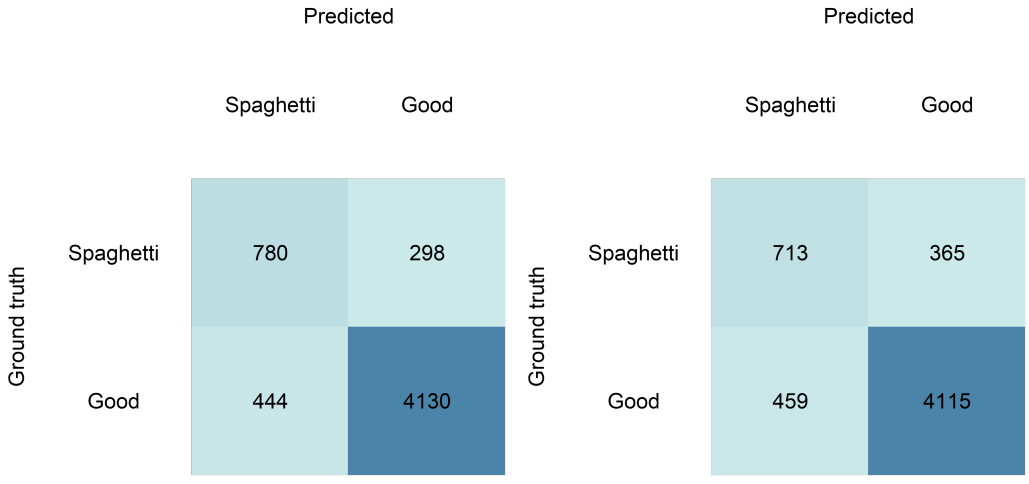



FIGURE B.11: Overall confusion matrix for the best setup.




(A) Non-CoLIE (B) CoLIE

FIGURE B.12: Confusion matrices for the best setup, split into non-CoLIE and CoLIE.

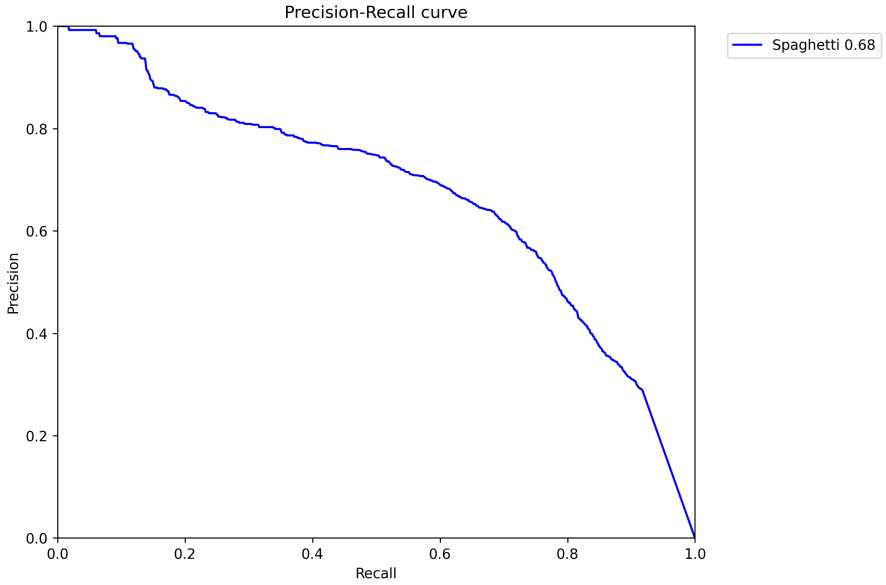



FIGURE B.13: The PR-curve for the best setup shows the trade-off between precision and recall, achieving an AP of 0.68.

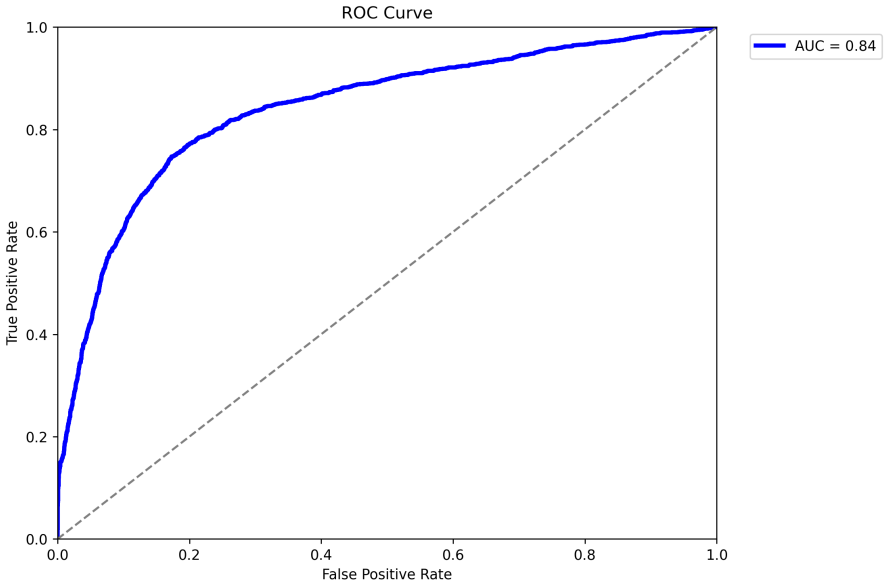
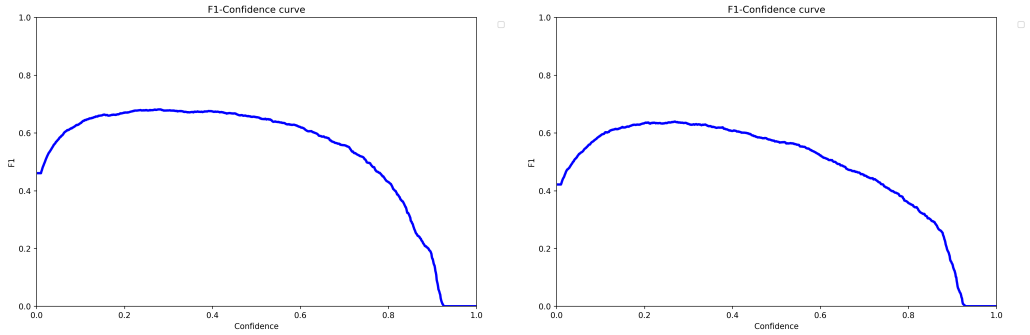
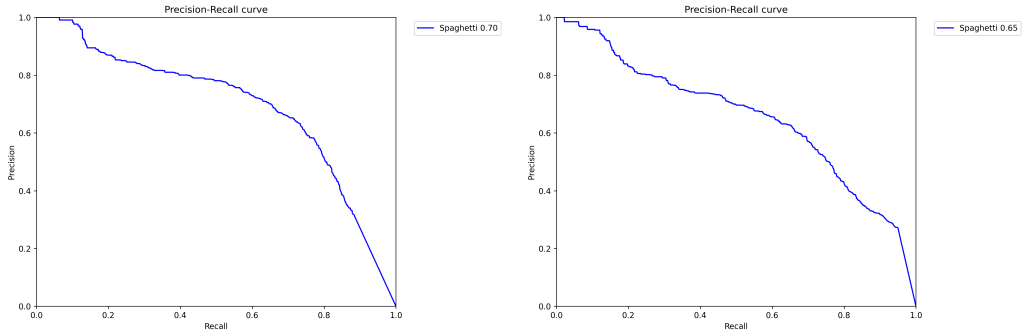



FIGURE B.14: The ROC-curve for the best setup shows the trade-off between false-positives and true-positives (by plotting the FPR against recall), achieving an AUC of 0.84.



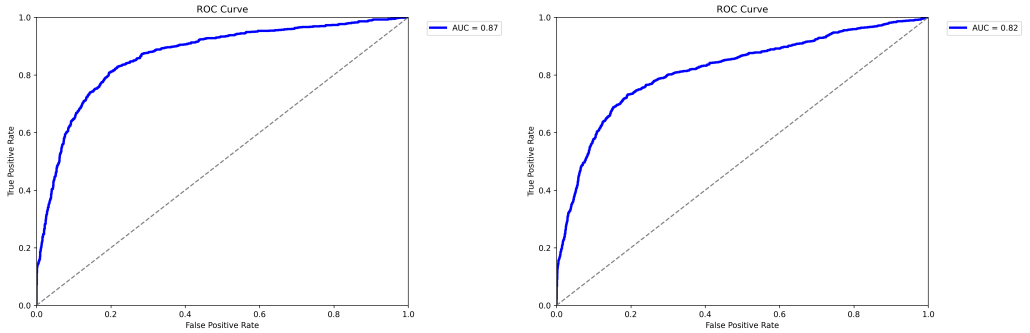

(A) Non-CoLIE (B) CoLIE

FIGURE B.15: The F1-curves for the best setup, split into non-CoLIE and CoLIE. This shows that the best setup achieves the highest F1-score of 68% at a 28% confidence threshold without CoLIE, whereas it achieves a 64% F1-score at a 27% confidence threshold with CoLIE.




(A) Non-CoLIE, AP = 0.70. (B) CoLIE, AP = 0.65

FIGURE B.16: The PR-curves for the best setup, split into non-CoLIE and CoLIE.




(A) Non-CoLIE, AUC = 0.87. (B) CoLIE, AUC = 0.82.

FIGURE B.17: The ROC-curves for the best setup, split into non-CoLIE and CoLIE.