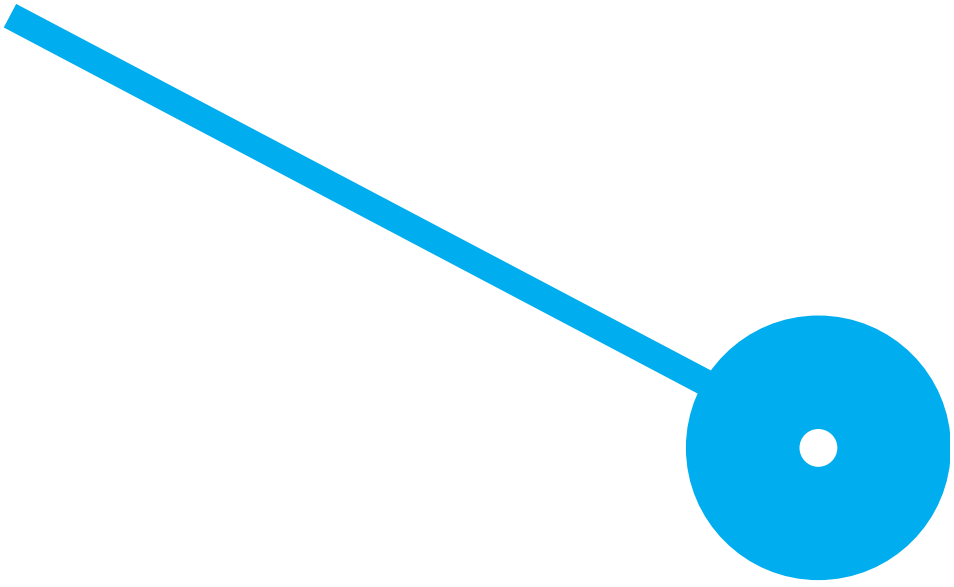
SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

M MASTER
Informatics Engineering

LIME: Optimising the creation of explanations

João Tiago Moreira Pereira

10/2024



SCHOOL OF
MANAGEMENT AND
TECNOLOGY
POLYTECHNIC OF
PORTO

MASTER
Informatics Engineering

LIME: Optimising the creation of explanations

João Tiago Moreira Pereira
8170202

Advisor

Prof. Davide Rua Carneiro

Dissertation submitted in fulfilment of the requirements for the Master's degree in Informatics Engineering in the School of Management and Technology of the Polytechnic of Porto.

10/2024

Declaração de integridade

Eu, **João Tiago Moreira Pereira**, estudante nº **8170202**, do Mestrado **Engenharia Informática** da Escola Superior de Tecnologia e Gestão do Instituto Politécnico do Porto, declaro que não fiz plágio nem auto-plágio, pelo que o trabalho intitulado "**LIME: Otimização da criação de explicações**" é original e da minha autoria, não tendo sido usado previamente para qualquer outro fim. Mais declaro que todas as fontes usadas estão citadas, no texto e na bibliografia final, segundo as regras de referenciação adotadas na instituição.

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao meu orientador do projeto, Professor Davide Carneiro, por todo o apoio, preocupação, paciência e disponibilidade prestados durante todo o desenvolvimento do projeto. Queria também agradecer ao alumni da ESTG, Filipe Oliveira, pelo apoio dado na compreensão inicial dos conceitos deste projeto.

Quero agradecer também à instituição de ensino, Escola Superior de Tecnologia e Gestão que me acolheu durante estes anos desde a licenciatura até ao mestrado e também agradecer a todo o corpo docente por me transmitirem ensinamentos importantes que permitiram complementar ainda mais o meu conhecimento na área de Engenharia Informática.

E por fim, gostaria de agradecer aos meus pais, familiares e amigos, que ao longo de todo o meu percurso académico, nunca deixaram de me apoiar e sempre que precisei de ajuda estiveram sempre presentes.

Abstract

Explainable Artificial Intelligence (XAI) techniques are increasingly necessary for ensuring trust and acceptance of complex machine learning models across various fields. One widely used XAI method, Local Interpretable Model-agnostic Explanations (LIME), is particularly popular for image-based explanations but faces challenges in terms of speed, accuracy, and applicability in different contexts.

An improvement to LIME is proposed to optimize its performance, including faster training times and better prediction accuracy, with a focus on finding an alternative machine learning algorithm that can outperform the current one used by LIME. Additionally, this project defines and explores metrics derived from LIME explanations that can help evaluate the quality of image classification models, even in concept drift scenarios where labeled data may be scarce. These metrics are validated against human feedback, identifying four key metrics that could prove useful for automated systems to assess model outputs.

Furthermore, in domains like manufacturing, LIME explanations must be adapted to context-specific challenges. In the case of defect detection in the textile industry, the permutation generation process used by LIME can mislead the underlying model, generating poor explanations. A methodology is proposed to mitigate this issue, supporting more accurate and contextually relevant explanations that can enhance decision-making and human-centric approaches in industrial scenarios.

Keywords: LIME, Optimization, Machine Learning, Computer Vision, Explainability, Defect Detection, Manufacturing

Resumo

As técnicas de Inteligência Artificial Explicável são cada vez mais necessárias para garantir a confiança e a aceitação de modelos complexos de machine learning em vários domínios. Um método amplamente utilizado, Local Interpretable Model-agnostic Explanations (LIME), é particularmente popular para explicações baseadas em imagens, mas enfrenta desafios em termos de velocidade, precisão e aplicabilidade em diferentes contextos.

Neste trabalho propõe-se uma melhoria do LIME para otimizar o seu desempenho, incluindo redução nos tempos de treino e melhor precisão de previsão, com o objetivo de encontrar um algoritmo alternativo de machine learning que possa superar o atual utilizado pelo LIME. Adicionalmente, este projeto define e explora métricas derivadas das explicações do LIME que podem ajudar a avaliar a qualidade dos modelos de classificação de imagens, mesmo em cenários de concept drift onde os dados etiquetados podem ser escassos. Estas métricas são validadas com base no feedback humano, identificando-se quatro métricas-chave que podem ser úteis para os sistemas automatizados avaliarem os resultados dos modelos.

Além disso, particularmente no domínio industrial, as explicações devem ser adaptadas aos desafios específicos do contexto. No caso da deteção de defeitos na indústria têxtil, o processo de geração de permutações utilizado pelo LIME pode induzir em erro o modelo subjacente, gerando explicações pobres. É proposta uma metodologia para mitigar este problema, apoiando explicações mais exactas e contextualmente relevantes que podem melhorar a tomada de decisões e as abordagens centradas no ser humano em cenários industriais.

Palavras-chaves: LIME, Otimização, Machine Learning, Visão de Computador, Explicabilidade, Deteção de Defeitos, Fabrico

Contents

List of Figures

List of Tables

Abbreviations

AI Artificial Intelligence. v, 1, 6, 7, 19, 21, 22, 26, 32, 45, 46

AP Average Precision. 36, 37

CNN Convolutional Neural Networks. 7–9, 12–15, 18, 26

CV Computer Vision. 6, 7, 11–14, 27, 32, 45, 46

DL Deep Learning. 1, 6–9, 22, 26, 35

DNN Deep Neural Network. 8

DPM Deformable Parts Model. 15

ECG Electrocardiogram. 26

ESAIM European Symposium on Artificial Intelligence in Manufacturing. 3, 46

ESTG Escola Superior de Tecnologia e Gestão. 3

FPN Feature Pyramid Network. 17, 18

FTP Fileira das Tecnologias de Produção. 1

GPU Graphics Processing Unit. 7, 9, 15, 40

IOU Intersection Over Union. 14, 16

LIME Local Interpretable Model-Agnostic Explanations. vii, viii, 1, 3, 4, 23–29, 31–35, 37, 38, 42, 45–47

MAE Mean Absolute Error. viii, 12, 42–44

mAP Mean Average Precision. 13, 18, 35–37

ML Machine Learning. 3, 4, 6, 7, 14, 21, 26, 27, 32, 42, 46

MLOps Machine Learning Operations. 21

PCA Principal Component Analysis. 39

RMSE Root Mean Square Error. viii, 42–44

SC-CNN Spatially Constrained Convolutional Neural Network. 8

XAI Explainable Artificial Intelligence. iii, 19–22, 27, 45

YOLO You Only Look Once. vii, 3, 15–20, 25–27, 31, 36, 46

Chapter 1

Introduction

1.1 Contextualization

Artificial Intelligence (AI) is becoming increasingly significant in industry, namely in quality control. However occasionally, the commonly used Deep Learning (DL) models are opaque, making it hard to assess their quality or comprehend the rationale behind their predictions. There are currently many different explainability methods available, but many of them have drawbacks that make it impossible to utilize them in industrial scenarios on a real-time basis.

LIME is one of the explainability methods that is the focus of this project. Like other explainability methods, LIME has drawbacks when it comes to real-time application in industrial scenarios. Throughout this document, these drawbacks will be examined and appropriate solutions will be provided.

The goal of this project is to optimize the process of explaining the detection of defects in fabrics using a computer vision model to identify them along with the explanations of the detections given by LIME. The industrial context for this project is the textile industry.

This project was developed in the context of PRODUTECH R3 [1]. The goal of the PRODUTECH R3 project is to bring about a fundamental change in Fileira das Tecnologias de Produção (FTP), enabling it to exploit the significant investments that industry will make with the green and digital transition, reducing external technological dependence, increasing the added value generated in the country and contributing to a change in the specialisation of the Portuguese economy.

Based on a partnership that brings together 108 companies and other entities, from FTP, the main sectors of industry and the scientific and technological system, the project envisages the collaborative development of 85 new innovative products and services and their demonstration in more than 52 pilots in companies from the user sectors and the development of complementary actions in the areas of education and training, internationalisation, dissemination and capacity building in FTP, inducing structural change and the creation of a truly dynamic and sustainable innovation eco-system in the area of production technologies.

[1] https://www.produtech.org/projetos/produtech-r3

1.2 Objectives

One of the drawbacks of the LIME method, which was highlighted in the previous section, is that it takes a while to generate explanations for predictions in images. This limits its utility in real-world scenarios, especially for real-time applications like defect or anomaly detection in fabrics. This method also has the drawback of having randomly generating permutations during its operation, which might occasionally lead to inaccurate explanations. And to mitigate this, the following objectives were defined:

1. Improve LIME's computing efficiency to make its use in real-time more realistic.
2. Making LIME more resilient to the challenges of image classification and explainability in industrial environments.
3. Look for an approach that evaluates the quality of the explanations produced by LIME.
4. Make the LIME-generated explanations more precise and clear.

1.3 Research Methodology

In this project it was followed the Design Science Research (DSR) Methodology [1], a form of science knowledge production that involves the development of innovative constructions, intended to solve real world problems. The main outcome of this type of research methodology is an artifact that solves a particular domain problem, also known as solution concept, which must be assessed against criteria of value or utility.

In the context of this project, this means presenting an innovative solution to a problem that covers several domains. This methodology has been pointed out as a suitable research approach when researchers work in close collaboration with organizations (as is the case in this project), as it allows new ideas to be tested in a real-life context. A DSR process commonly includes six steps or activities:

1. Identification of the problem, defining the research problem and justifying the value of a solution.
2. Definition of objectives for a solution.
3. Design and development of artefacts (constructs, models, methods, etc.).
4. Demonstration by using the artefact to solve the problem.
5. Evaluation of the solution, comparing the objectives and the actual observed results from the artefact.
6. Communication of the problem, the artefact, its utility and effectiveness to other researches.

Throughout this document, which represents the sixth and final step in this process, the remaining steps will also be presented: The first and second steps have already been described earlier in this section, while the third, fourth and fifth steps, which essentially describe the implemented solution and its results, will be described in a later section.

1.4 Work Planning

This section will provide a detailed explanation of the steps taken during the project's development, along with a presentation of the Gantt chart, represented in Figure 1.1, and the methodology's phases.

Weekly meetings were used to discuss the results of the tasks that had been set forth in the prior week's meeting and, depending on the conclusions reached during that discussion, to define the tasks for the upcoming week.

Figure 1.1 shows the following phases of the project:

- **Phase 1 - Proposal of a ML Regression Algorithm Superior to Ridge**

The LIME method's concepts were first studied in the first phase. Afterwards, information on the method's execution time was collected to determine where intervention was necessary to maximize its effectiveness. After gathering all of the optimization proposal findings, an article was written for the ESTG Masters event to finish this phase. Chapter 6 contains the pertinent information about it.

- **Phase 2 - Methodology for Defining Optimal LIME Background for Fabric Defect Image Permutations**

It was discovered over the course of this project that the YOLO model, an object detection model whose functioning is described in Section 2.3.2, was incorrectly identifying the intended objects when used in conjunction with LIME. The primary goal at this point was to develop an approach that will be covered in Chapter 4 in order to solve this issue. This problem served as the focal focus of a collaborative effort to write a paper for the ESAIM conference in order to finish up this phase.

- **Phase 3 - Approach to Minimizing Permutation Matrix Size**

During this phase, the main goal was to develop an approach that would minimize the size of the permutation matrix produced by LIME while preserving the variety of each permutation and, as a result, cutting down on execution time, the description of this approach and its results are presented in Chapter 5.

- **Phase 4 - Definition of Metrics for Evaluating LIME Explanation Quality**

Developing metrics to evaluate the quality of the explanations produced by LIME was the focus of the fourth phase. Explanation metrics and their results can be found in Chapter 3. The definition of these metrics served as motivation for collaboration while writing a paper for the IAMOT conference, concluding this phase.

- **Phase 5 - Dissertation writing**

In this last phase, the dissertation was written at the same time as phases 2, 3 and 4.

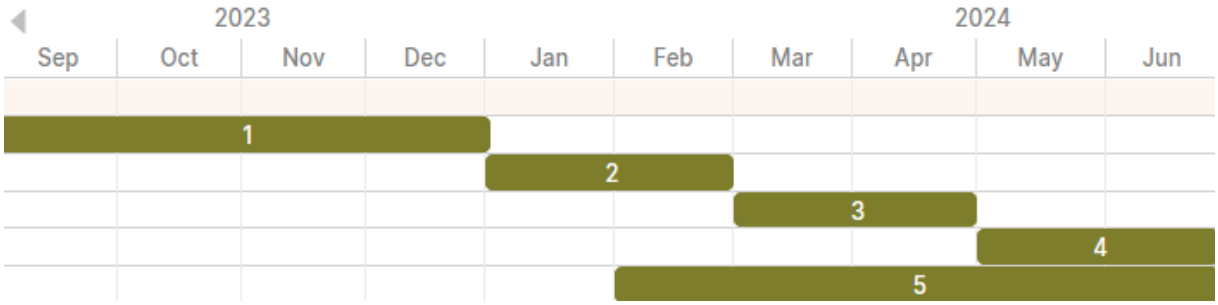


Figure 1.1: Gantt diagram of project timeline.

1.5 Outline of the document

This dissertation has seven chapters. The project's concept is introduced in general and concisely in Chapter 1. The background is covered in Chapter 2, where the different project concepts are provided for a better understanding of them throughout this document.

Presented in Chapter 3 are a set of metrics that assess the LIME-generated explanations, metrics validation via a questionnaire, and the corresponding results from it. In Chapter 4, a methodology for getting around one of LIME's limitations in terms of producing explanations for fabric defects is described.

An approach to enhance LIME's permutation generation process' efficacy and efficiency is provided in Chapter 5. A regression ML algorithm will be proposed to replace the algorithm used by LIME to extract feature importance on superpixels, Ridge, in Chapter 6. At last, Chapter 7 will provide the relevant conclusions, as well as future work and articles that were published throughout this project development.

Chapter 2

Background

2.1 Quality control in the textile industry

The industry has undergone significant changes over the years, with the development of advanced technologies and modern methods for textile production. Despite these advancements, the textile production business still encounters challenges related to the fabric quality. Fabric flaws such as neps, barres, stains, and other defects can significantly affect the quality of the fabric, leading to reduced sales and decreased serviceability [2].

These flaws are often caused by bad yarns, broken equipment, or improper handling of equipment and processes by human operators. The negative impact of fabric flaws on the textile industry is significant, leading to increased wastage, higher expenditures, and decreased profitability [2].

In the context of quality control, the traditional textile industry has long relied on manual human inspection. Skilled workers use their eyes and experience to examine textile products for patterns, defects, and imperfections. This manual inspection ensures that the final products meet the desired quality standards. However, it is a labor-intensive and subjective process prone to human error and fatigue, especially in long production runs. When the patterns are not recognizable, operators will do a destruction test to find the type or original color of the fabric [3].

Every step of the fabric manufacturing process needs quality control because it helps to reduce waste, identify potential issues with a specific machine, and identify the source of defects in the process. Ensuring the quality and efficiency of the whole process is crucial for manufacturers to meet customer demands and maintain competitive advantage. Otherwise, the presence of anomalies or abnormalities can lead to substandard products, increased waste, longer production times and costs, and even potential equipment failures [4].

Nonetheless, and despite the importance of quality inspection, it is becoming increasingly harder to hire and train people for such tedious and repetitive tasks [5]. Quality inspection and defect detection techniques that can be used in practice, desirably in real-time, are thus essential to identify and address such deviations promptly, avoiding increased costs and other negative impacts.

2.1.1 Impact of Computer Vision on Industry Quality Control

Computer vision (CV) technology has emerged as a powerful tool for quality control in various modern industries [3]. By harnessing algorithms and image processing techniques, CV enables the swift and accurate identification of product defects, effectively overcoming the limitations associated with manual inspection [6]. Notably, this technology has been widely embraced for its efficiency and cost-effectiveness across electronics to automotive manufacturing [7, 8].

Furthermore, the pharmaceutical industry has adopted CV for ensuring the quality and safety of drug production [9], and the food and beverage industry employs it to enhance the inspection of food products, guaranteeing adherence to stringent quality standards [6]. Aerospace manufacturing also benefit from CV's ability to detect imperfections in components and ensure the reliability of critical systems [10].

The textile industry has similarly embraced CV by integrating systems equipped with cameras and sensors into production lines. These systems capture detailed product images, facilitating real-time analysis and the effective identification of defects [11].

ML algorithms have been employed to enhance accuracy, enabling the recognition of complex patterns and defects. The data generated through CV inspections contributes to defect identification and can potentially optimize production processes, reduce downtime, and enhance overall operational efficiency [3].

2.2 Artificial Intelligence

AI is commonly defined as "a system's ability to interpret external data correctly, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation" [12] and is also a major factor in many fields these days.

For instance, by analyzing vast amounts of medical data and offering individualized therapy, AI has the potential to help interpret diagnoses and support medical decision-making in the treatment of lung cancer.. As a result, significant strides have been made in the detection of lung cancer [13].

Another example is in the area of Industry 4.0, where AI is increasingly seen as the main component of industrial transformation, enabling intelligent machines to perform tasks autonomously, such as self-monitoring, interpretation, diagnosis and analysis, especially using ML and DL models that support manufacturers and industries in predicting their maintenance needs and reducing downtime [14].

2.2.1 Deep Learning

Machine learning is a subfield of the AI domain that procures mechanisms to acquire self-learning skills or knowledge from the experience and focuses on the development of computational systems accessing and using data to find hidden patterns to make predictions or decisions [15–17].

However, conventional ML techniques are limited in their ability to process natural data in their raw form. Building a ML system has required substantial domain expertise skills

that can take raw data to design a representation form and classify them into their relevant target for three decades [18].

Instead of strictly following static program instructions, ML algorithms can be used by constructing a model to make data-based decisions and predictions. In other words, a ML system uses a single layer and tries to extract significant results using representation form (features). However, conventional ML techniques are not enough to represent all significant features hidden in an input. The shortcomings in these techniques contributed to researchers developing new solutions, mainly based on DL. [19].

The term 'deep learning' was first introduced by Aizenberg et al. in the context of an artificial neural network in 2000 [20]. Hinton demonstrated how a multilayer feed-forward neural network can effectively train a layer at each iteration in an article published in 2006 (each layer was trained with an unsupervised restricted Boltzmann machine), and then fine-tuned with a supervised back-propagation method [21]. Hinton et al. designed similar architectures in 2012. In their GPU-supported studies, the normalization method called 'dropout' was used to reduce overfitting [22]. The effectiveness of the dropout method has been proved when it gave remarkable results in the ILSVRC-2012 ImageNet competition [23].

DL allows computational models that are composed of multiple processing layers to learn a representation of data with multiple levels of abstraction. Until Hinton introduced the DL model, DL algorithms were not preferred because they increased the number of hidden layers in conventional artificial networks and depth in a network causes computational complexity [24, 25].

The emergence of DL is an important progression for the AI field. Many practical problems and challenges can be simplified and solved in a more rectified form. The accessibility and extendibility of DL methods are correct for users to apply, not only for data practitioners but also for ordinary developers. Researchers have built several DL architectures for various tasks [19]. Hereby, six common network architectures are illustrated in Figure 2.1 and Convolutional Neural Networks (CNN) are the most important architecture in CV and will be described in subsection 2.2.4.

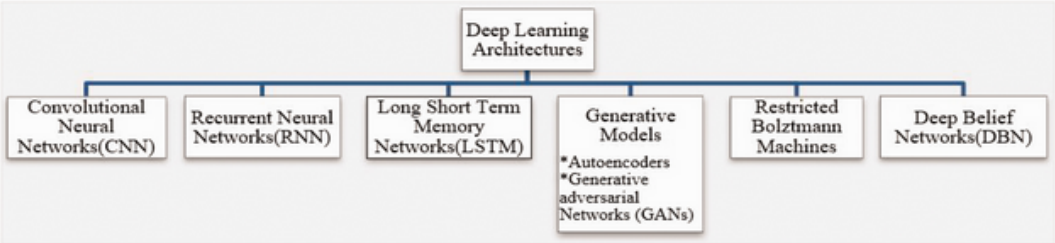


Figure 2.1: Common Deep Learning architectures [19].

2.2.2 Deep Learning Applications

Natural Language Processing

In natural language, DL is applied in many areas such as voice translation, machine translation, computer semantic understanding, and so on. However, the success of DL is

more notable in two fields, i.e., image processing and the natural language processing [26].

In 2012, Schwenk et al. [27] proposed a phrase-based statistical machine translation system based on Deep Neural Network (DNN). It was able to learn meaningful translation probabilities for unseen phrases which were not presented in the training set. In 2014, Dong et al. [28] proposed a novel Adaptive Multi-Compositionality (AdaMC) layer in the recursive neural network. This model introduced more than one composition function, which was adaptively selected based on the input features. In 2014, Tang et al. [29] presented a DNN on Twitter data for sentiment analysis.

In 2015, Google introduced Word Lens recognition technology based on DL, which used word lenses in real-time call translation and video translation. This technology not only could read the words in real-time, but also those words could be translated into the desired target language. Also, the translation work could be done through the phone without networking. The current technology could be applied in more than a visual translation of 20 languages. In addition, Google proposed an automatic mail reply function in Gmail, which used a DL model for extracting the e-mail content and analyzing it semantically. Finally, a reply is generated based on the semantic analysis. This technique is fundamentally different from the traditional e-mail auto-respond functionality [26].

Speech recognition

In order to realize the Human–Computer Interaction, researchers made great efforts. In 1952, Bell Institute's Davis and others successfully developed the world's first experimental system that could identify 10 English digital pronunciations. The research on speech recognition technology has few decades of history, and voice recognition was the dictator used in certain areas as it was mentioned by the United States press as one of the top ten events in computer development [26].

In the last two decades, speech recognition technology has made significant progress. With the continuous improvement of DL models, a large number of speech recognition devices or applications have begun to move from the laboratory to the market. In 2014, Baidu launched Deep Speech, a voice recognition system with DL technology, which can achieve 8% accuracy in noisy. Maas et al. [30] analyzed different architectures and parameters of the DNN for training a very large speech data. They found simple architecture and simple optimization methods that gave stronger performance than the other more complex models.

Medical applications

The forecast function of DL and its automatic feature identification makes it a popular technique in disease diagnosis also [26].

In 2014, Li et al. [31] proposed customized CNN to classify lung image patches. This model used the dropout method and single-volume structure to avoid overfitting. In 2015, Li et al. [32] proposed a DNN-based framework to differentiate the identity stages of Alzheimer's Disease (AD) from the Magnetic Resonance Imaging (MRI) and Positron Emission Tomography (PET) scan data. In 2016, Srinukunwattana et al. [33] proposed a Spatially Constrained Convolutional Neural Network (SC-CNN) to analyze the histopathology images and identify the nucleus of the cancerous cells. Their SC-CNN method had better performance than the classical feature classification method.

In 2016, Google developed a vision system for identifying early-stage ocular diseases. They worked with the Moorfields Eye Hospital, such as diabetic retinopathy and age-related macular degeneration to provide early prevention methods. A month later, Google used DL techniques to design a head and neck cancer radio-therapy method which had an effective control of the patient's radiotherapy time and it could minimize the radiotherapy of the patient's injury. With the continuous development of DL technology in the field of precision medical care, it lead to more prominent contributions [26].

2.2.3 Overview of a Convolutional Neural Network

CNN [34] was first proposed in the 1980s. It is inspired by the cat's cortex. [35] The LeNet-5 system was a classical model of a convolutional neural network. Its error rate was only 0.9% on the MNIST dataset, a dataset of handwritten digits. It had been widely used to identify handwritten checks on banks, but it did not recognize large images [26].

With the development of Graphics Processing Unit (GPU) technology, Krizhevsky et al. [36] used an efficient GPU supported program to solve the ImageNet problem in 2012, which made the convolutional neural network again popular. In fact, one of the bottlenecks of deep neural nets was that it took a long time for training because of the many hidden nodes in its network. But as the GPUs became faster in parallel computing, this bottleneck was overcome [26].

At present, the convolutional neural network is a hot topic in the field of voice data analysis and image recognition. The convolutional neural network has a network structure with shared permission, which makes it closer to the biological neural network. This network structure in the convolutional neural network can effectively reduce the complexity of the network model and also reduce the number of the weights [26].

Mainly, it is more efficient to deal with high-dimensional images, which can directly consider the image as the input of the entire network and effectively avoid the complex feature extraction and reconstruction of the traditional algorithm. In the process of image recognition, the convolutional neural network has a high degree of invariance in scaling, tilting, translating, and other forms of image deformation [26].

The use of CNNs in fabric detection is proven. In fabric classification, recent studies have highlighted the effectiveness of CNNs in capturing intricate patterns and textures [3].

2.2.4 Structure of a Convolutional Neural Network

As a multilayer neural network, each layer in the convolutional neural network structure is composed of a number of two-dimensional planes and each plane has independent neurons. The sparse connections are used between the layers [26].

This means that the neuron in each feature map only connects the neurons in a small area in the upper feature map, rather than the traditional neural network. A typical model of the convolutional neural network is shown in Figure 2.2. The convolutional neural network structure mainly depends on the shared weight, the local experience field, and the sub-collector to ensure the invariance of the input data [26].

These factors can be explained as follows: Local experience field: The first hidden layer contains six feature maps (see Fig. 2.2). Each feature map corresponding to a small box

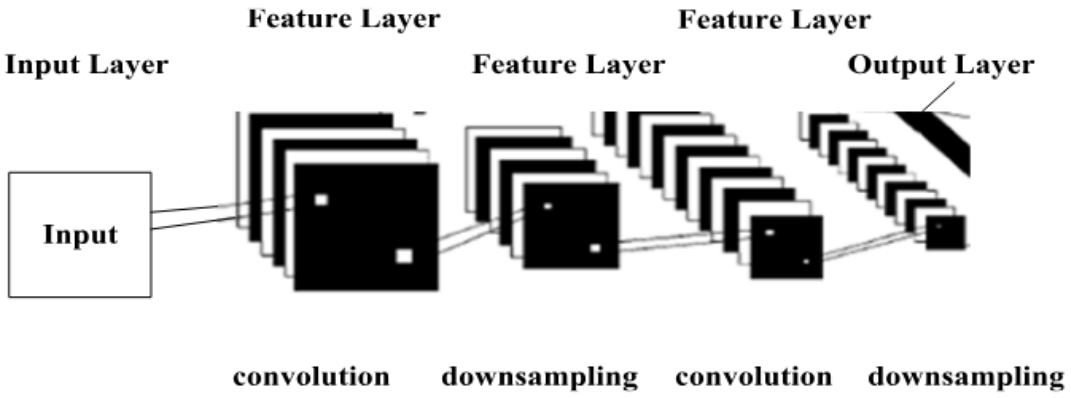


Figure 2.2: Convolutional Neural Network model [26].

in the input layer is a local experience field, or it is called the sliding window [26].

Convolution: The activation value a_j^l of the j^{th} feature map in the convolution layer l, which is expressed in the following equation:

$$a_j^l = f(b_j^l + \sum_{i \in M_j l} a_i^{l-1} * k_{ij}^l) \tag{2.1}$$

where f is a non-linear function, $*$ is a two-dimensional convolution operator, b_j^l is the jth offset in the lth layer, and k_{ij}^l is the weight, which is a cumulative input in the feature map i of l - 1th layer. The index vector of i is $M_j l$ in the feature map of l - 1th layer. The feature map needs to be accumulated in the lth layer. Weight share: Each convolution layer is usually composed of several feature maps and the weight of the same features in Figure 2.2 is the same, which can reduce the number of its own parameters [26].

Sub-sampling: The translation of the convolution layer will also translate its output at the same time. However, its characteristics remain unchanged, and as long as a feature is detected, its exact position will no longer be considered, as only the relative position of the other features can be preserved. So, each convolution layer has a sub-sampled layer that performs local averaging to reduce the sensitivity associated with the deformation and the translation of the output. The feature mapping of the subsampling layer is denoted by Equation 2.2 as follows: [26]

$$a_j^l = down(a_i^{l-1}, N^l) \tag{2.2}$$

where N^l is the boundary size of the subsurface required for the l sub-sampling layer and *down* is the down sampling function in the factor. The above formula 2.2 is the mean operation of the localized non-overlapping parts of size. If the neuron output layer is C-dimensional, then the C class can be identified, and the output layer which is the output characterization for front connection feature mapping is expressed in Equation 2.3 as follows [26]:

$$output = f(b^0, W^0 f_v) \tag{2.3}$$

where b^0 is the partial value vector, W^0 is the weight matrix, f_v is the eigenvector, and k_{ij}^l, b_l^j, b^0, W^0 are the model parameters. As the convolutional neural network structure is mainly alternately composed of convoluted layer and sub-sampling layer. With the reduction of the spatial resolution, the number of the feature maps is also increasing. The training process of the convolutional neural network starts with a forward training phase and it consists of the following three steps: [26]

- Select the samples according to the given sample set randomly.
- Put the samples as initial data into the network.
- Calculate the corresponding output data.

The second stage is the backward propagation phase and it consists of the following two steps [26]:

- Calculate the difference between the ideal data information and the output data information.
- Adjust the weight matrix according to the minimization of the error method for the reverse transmission.

2.3 Computer Vision

Computer vision (CV) is an application of artificial intelligence and its techniques have played an important role in promoting the informatization, digitalization, and intelligence of industrial manufacturing systems [37]. Computer vision is an interdisciplinary field that deals with how computers can gain high-level understanding from digital images or videos. It can use computers and cameras to replace the human eye for the target object recognition, tracking, measurement, and for other visual problems. And then deal with the graphics so that the computer can achieve image processing capabilities even beyond the eye [26].

CV techniques have been applied to different manufacturing industries since the early 1970s, such as food, pharmaceutical, automotive, aerospace, railway, semiconductor, electronic component, plastic, rubber, paper, and forestry-related fields [38].

In the early days, only a few CV methods were practically used in commercial manufacturing because of limited computing capabilities, until the 1990s [39]. Opportunities for CV in manufacturing can be grouped into three broad categories: 1) image-based metrology; 2) manufacturing process interpretability; and 3) material structure analysis [40]. From the perspective of application scenarios, industrial applications of CV technologies can be classified into: visual inspection, part identification, process control, and robotic guidance mechanism [41].

2.3.1 Computer Vision Techniques

The purpose of a CV system is to generate a symbolic description of what is being portrayed in a scene [42]. This description includes understanding of the scene and then be applied to guide the next operations of the robot system. There are multiple kinds of tasks and algorithms in the CV field, such as detection, recognition, segmentation, and

3-D reconstruction [37]. This subsection presents the state of the art of several important CV techniques.

Feature Detection

Visual feature detection (e.g., point, edge, and line detection) is the basis of many other CV algorithms. In some cases, we are more interested in a specific region of the image, such as human eyes, license plates, and corner shapes, which are called keypoint features.

In some other cases, we care more about the edge features of objects in an image. A few CV algorithms can be applied to identify and match both keypoint features and edge features among different images [37]:

- **Keypoint Detection:** There are mainly two types of keypoint detection methods: 1) local search detection and 2) global search detection. Common local search detection methods include correlation methods, least squares methods, and learning-based methods [43]. Local search detection methods are more suitable for scenes where images are taken continuously in a high frequency, such as video sequences. Different from local search methods, global search detection methods search the entire image and then match features based on feature appearance. Therefore, global search methods are more suitable for scenes where there are a lot of movements or appearance changes [44].
- **Edge Detection:** Edges in images often appear at boundaries between different objects, resulting in sudden changes in color and intensity. One method of edge detection is based on the gradient through the image [45], even though the gradient is easily affected by noise. Hence, a low-pass filter (e.g., Gaussian filter) is needed to filter the image before gradient calculation. Horizontal and vertical convolution operations can be used for edge recognition. The zero-crossing method can also be used for edge detection in which zero points of a second-order derivative expression are searched to find edges [46]. Recently, deep-learning methods, especially CNNs, have been widely applied to edge detection [47, 48].

Commonly used evaluation metrics for feature detection include Mean Absolute Error (MAE), Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM) [37].

Throughout the development history of CV, the rise of CNNs has been one of the most important breakthroughs for the recent rapid development of CV technologies [37]. In recent years, numerous CNNs have been proposed and applied in classification problems, and their models are getting deeper and deeper. Sorted from earlier to recent works, popular CNNs include LeNet-5 [49], AlexNet [36], VGG-16 [50], R-CNN [51], Fast R-CNN [52], inception networks [53], ResNet-50 [54], Xception [55], and ResNeXt-50 [56]. Before ResNet-50, most of the proposed CNNs only innovated by adding an increased number of layers in the network, alongside any other engineering changes needed for good performance [37].

The "bottleneck" of this trend was that the model's accuracy gets saturated and then rapidly decreases as the network becomes deeper and deeper [37]. The seminal residual network first inserted shortcut connections in its deep model to deal with this problem [54]. The object detection neural network Fast R-CNN trained the deep VGG16 network 9×

faster than R-CNN and 213× faster at test time. Ren et al. proposed the Faster R-CNN [57] in which a region proposal network was used to share full-image convolutional features with the detection network. For the very deep VGG-16 model, Faster R-CNN had a frame rate of 5fps with 0.732 mAP accuracy on PASCAL VOC 2007 and 0.704 mAP accuracy on PASCAL VOC 2012 [37].

Recognition

Recognition is another important task for CV techniques. From the perspective of target objects, recognition problems can be grouped into three categories: 1) instance recognition; 2) class recognition; and 3) general category recognition, as follows:

- **Instance Recognition:** In instance recognition, the goal is to identify a specific known object. Feature matching strategies can be used for this recognition problem [58]. Other instance recognition methods include viewpoint-invariant feature-based strategies [59], and sparse feature matching [60]. A popular application of instance recognition is face recognition [61]. Some learning-based approaches have also been widely applied to this problem, such as Support Vector Machines (SVMs) [62], boosting [63], and neural networks [64].
- **Class Recognition:** Different from instance recognition, class recognition does not have a specific object as the target. In class recognition problems, the goal is usually to recognize the presence of an instance of a specific category of objects, such as cars or pedestrians. Class recognition problems can be considered as a specific classification problem in which the input is an image and the output is the classification of that image. [37]
- **General Category Recognition:** General category recognition is the most challenging recognition problem because we need to identify not only the locations of different objects in the image but also what category each object belongs to. In this task, all different kinds of objects in the image need to be recognized. Common approaches of general category recognition include bag-of-words models and part-based models. In bag-of-words models, the distribution of visual words in the target image is compared to the training data [65]. In the part-based models, different parts of an image are considered separately so as to determine whether and where an object of interest exists [66]. Recent advances in this task have taken advantage of deep residual learning [54] and deep neural networks [67].
- **Action Recognition:** The current challenging problem for the recognition task is action recognition. It is not difficult for humans to recognize actions in videos, but it is challenging for machines. Accurately recognizing the actions and behaviors in videos is of great significance for different scenarios, such as pedestrian motion monitoring in autonomous driving [68], and elderly fall detection [69]. In the manufacturing industry, identifying the behaviors of workers in the workshop has also been shown to be a very valuable approach to ensure production safety [70].

Segmentation

As one of the most classic tasks in the CV field, image segmentation aims to label pixels into different groups according to the objects that each pixel belongs to. Image segmentation can essentially be considered as a clustering problem. Early segmentation

techniques usually used region division and merging methods [70]. Later segmentation algorithms applied indicators of consistency, such as intraregional consistency and interregional dissimilarity [71]. Other segmentation approaches include mean shift [72], graph-based merging [73], graph cut-based Markov [74], and level sets [75].

In the learning-based segmentation algorithms, one commonly used loss function is dice loss which is based on the dice coefficient. A dice coefficient can be defined as twice the true positive divided by the sum of twice the true positive, false positive, and false negative. Another commonly used loss function for deep-learning segmentation algorithms is the Intersection Over Union (IOU) [37].

The latest segmentation algorithms based on ML in recent years include Mask RCNN [76] and dual attention network [77]. Recently, U-Net has shown good performance in the segmentation tasks of medical images [78]. There are some variants of U-Net, such as Attention U Net [79], U-Net++ [80], ResUNet++ [81], and TransUNet [82]. The performance of these algorithms not only depends on the algorithm design but also depends on the datasets.

Current challenges and trends in segmentation include 3-D segmentation and 4-D segmentation problems. The goal of 3-D segmentation is to segment 3-D images in three spatial directions, while 4-D segmentation is to segment 4-D data which also includes the time dimension in addition to spatial dimensions. The 3-D U-Net has been applied to 3-D segmentation problems in medical imaging [83] and additive manufacturing defects in X-ray Computed Tomography (CT) images [84].

3-D Modeling

3-D modeling in CV can be categorized into two problems: 1) stereo correspondence and 2) 3-D reconstruction. Stereo correspondence is the process of generating a 3-D model of an object from two or more images of the same object or scene, while the 3-D reconstruction is to generate a 3-D model of an object from only one image [85]. It is a challenge to design a good loss function to evaluate the predicted 3-D point cloud and ground truth. One option is evaluating how well the projections of predicted 3-D point clouds cover the ground-truth object's silhouette [86].

A common stereo correspondence method is to find matching pixels in multiple images and map their position in the 2-D images to 3-D positions in the 3-D model. Popular methods for stereo correspondence include epipolar geometry [87], sparse correspondence [88], and dense correspondence [89].

The earliest approach for 3-D reconstruction is to predict object shape from visual shading, which was first proposed by Horn in 1970 [90]. Later, other "shape from X" methods were proposed, such as shape from texture [91] and shape from focus [92]. Other 3-D reconstruction methods include active rangefinding [93], and model-based reconstruction which has been widely applied to architectural 3-D modeling [94]. Recently, deep-learning-based algorithms promoted significant improvement in system performance of 3-D reconstruction [95, 96].

2.3.2 YOLO

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought [97]. You Only Look Once (YOLO) is a Deep Learning model that aims to emulate this behavior.

Before YOLO was created, detection systems repurposed classifiers to perform detection. To detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image. Models like Deformable Parts Model (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image [66], and approaches like R-CNN use region proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [98].

These complex pipelines are slow and hard to optimize because each individual component must be trained separately. To overcome this issue, Redmon et al [97] reframed object detection as a single regression problem straight from image pixels to bounding box coordinates and class probabilities, by creating an object detection system called You Only Look Once (YOLO) where given an image it predicts what objets are present and where they are.

In Figure 2.3, a single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. Then, YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection [97]:

1. YOLO is extremely fast, since detection is framed as a regression problem, there is no need for a complex pipeline and and is simply executed a neural network on a new image at test time to predict detections. The base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means that a streaming video can be processed in real-time with less than 25 milliseconds of latency. In addition, YOLO achieves more than twice the mean average precision of other real-time systems

2. YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Fast R-CNN, a top detection method [52], mistakes background patches in an image for objects because it can't see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN.

3. YOLO learns generalizable representations of objects. When trained on natural images and tested on artwork, YOLO outperforms top detection methods like DPM and R-CNN by a wide margin. Since YOLO is highly generalizable it is less likely to break down when applied to new domains or unexpected inputs.

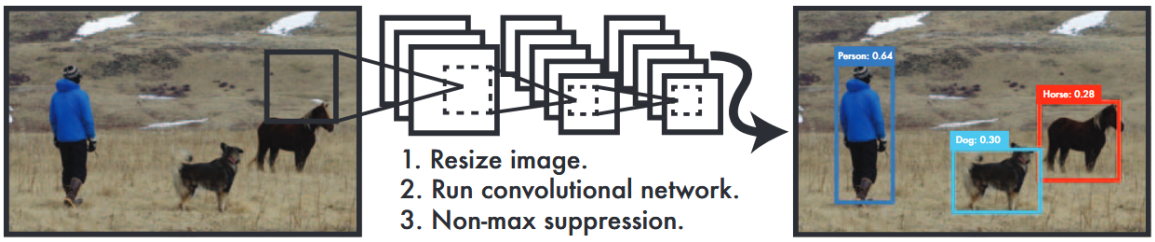


Figure 2.3: The YOLO Detection System. Processing images with YOLO is simple and straightforward. YOLO system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence [97].

YOLO Components

YOLO network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means YOLO network reasons globally about the full image and all the objects in the image [97].

The YOLO design enables end-to-end training and real-time speeds while maintaining high average precision. YOLO divides the input image into an $S \times S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object [97].

Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally, confidence is defined as $Pr(Object) * IOU_{pred}^{truth}$. If no object exists in that cell, the confidence scores should be zero. Otherwise confidence must be scored to equal the IOU between the predicted box and the ground truth [97].

Each bounding box consists of 5 predictions: x, y, w, h and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box [97].

Each grid cell also predicts C conditional class probabilities, $Pr(Class_i|Object)$. These probabilities are conditioned on the grid cell containing an object. Its only predicted one set of class probabilities per grid cell, regardless of the number of boxes B.

Conditional class probabilities are multiplied and the individual box confidence predictions are given by:

$$Pr(Class_i|Object) * Pr(Object) * IOU_{pred}^{truth} = Pr(Class_i) * IOU_{pred}^{truth} \tag{2.4}$$

which gives a class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

YOLO Network Design

YOLO model, represented in Figure 2.4, was implemented as a convolutional neural network and it was evaluated on the PASCAL VOC detection dataset [99]. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates. YOLO network architecture is inspired by the GoogLeNet model for image classification [100] and it has 24 convolutional layers followed by 2 fully connected layers [97].

Instead of the inception modules used by GoogLeNet, its simply used 1×1 reduction layers followed by 3×3 convolutional layers, similar to Lin et al [101]. The YOLO original architecture is shown in Figure 2.5. The final output of YOLO network is the $7 \times 7 \times 30$ tensor of predictions.

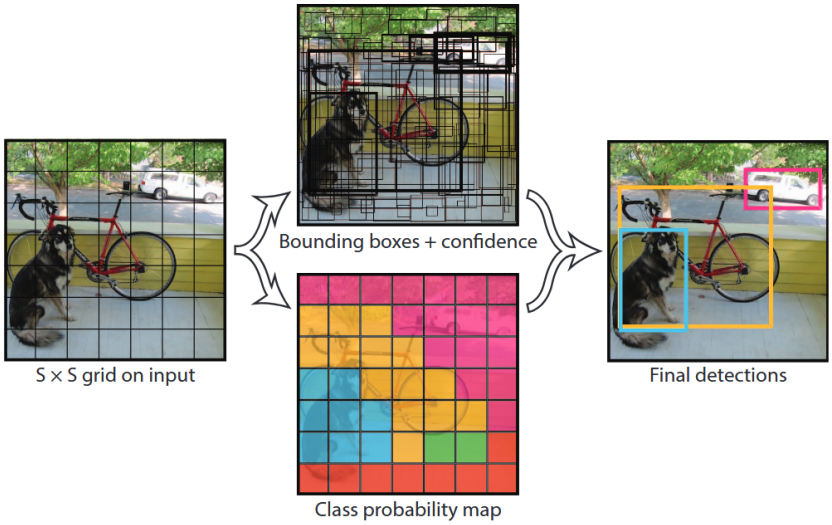


Figure 2.4: The Model. YOLO models detection as a regression problem. It divides the image into an S $\times$ S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor [97].

YOLOv3

YOLO-v3 proposes a hybrid architecture factoring in aspects of YOLO-v2, Darknet-53, and the ResNet concept of residual networks. This enable the preservation of fine-grained features by allowing for the gradient flow from shallow layers to deeper layers [102].

On top of the existing 53 layers of Darknet-53 for feature extraction, a stack of 53 additional layers was added for the detection head, totaling 106 convolutional layers for the YOLO-v3. Additionally, YOLO-v3 facilitated multi scale detection, namely, the architecture made predictions at three different scales of granularity for outputting better performance, increasing the probability of small object detection [102].

YOLOv3 uses multi-label classification, binary cross-entropy loss for each label instead of using mean square error in calculating the classification loss. YOLOv3 predicts objects in three different scales (similar to Feature Pyramid Network (FPN) 24) as shown in Figure 2.6 and the score for each bounding box using logistic regression. DarkNet-53 (YOLOv3 backbone) is used to replace the DarkNet-19 as a new feature extractor. The

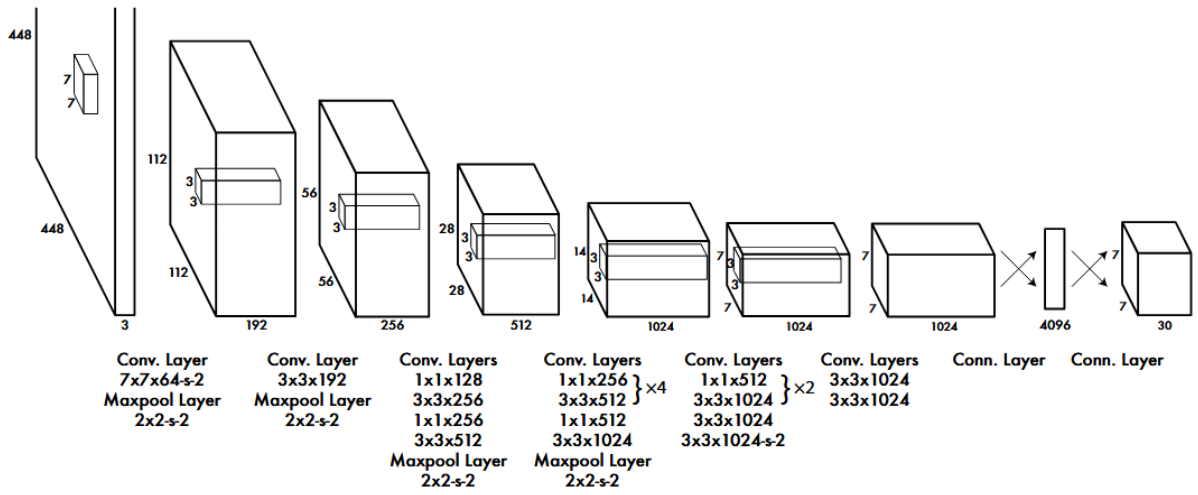


Figure 2.5: The Architecture. YOLO detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. Convolutional layers are pretrained on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection [97].

whole DarkNet-53 network is a chain of multiple blocks with some strides 2 convolution layers in between to reduce dimension [103].

Each block contains a bottleneck structure of 1 × 1, followed by 3 × 3 filters with skip connections similar to ResNet. DarkNet-53 possesses less billion floating point operations (Billion of Floating Point Operations (BFLOP)) compared to ResNet-152, but is two times faster with the same classification accuracy. YOLOv3 shows significant improvement for small objects detection and performs very well in real-time inference [103].

YOLOv7

YOLOv7's architecture, represented on Figure 2.7, has three main high-level components, described by the authors as the backbone, the neck and the head. The **backbone** is a convolutional neural network that aggregates picture pixels to generate characteristics at various levels of detail. The main framework is generally pre-trained on a classification dataset, such as ImageNet [1].

In the backbone of YOLOv7, which was based on YOLOv3, the Darknet53 backbone was substituted with the Resnet50-vd-dcn ConvNet. Resnet is a widely favored architecture, with more frameworks fine tuned for its implementation, and it possesses fewer parameters compared to Darknet53. Using a better pretrained model has shown to improve transfer learning to other domains, resulting in better performance, which led to an enhancement in the mAP, a commonly used performance metric in the field of object detection and information retrieval [104] used in evaluating architectures such as Faster R-CNN, MobileNet SSD, and YOLO [5].

The **neck** consists of a FPN, which combines and mixes the ConvNet layer representations, and then passes them to the prediction head. Another improvement in YOLOv7 was to

[1] https://www.image-net.org/

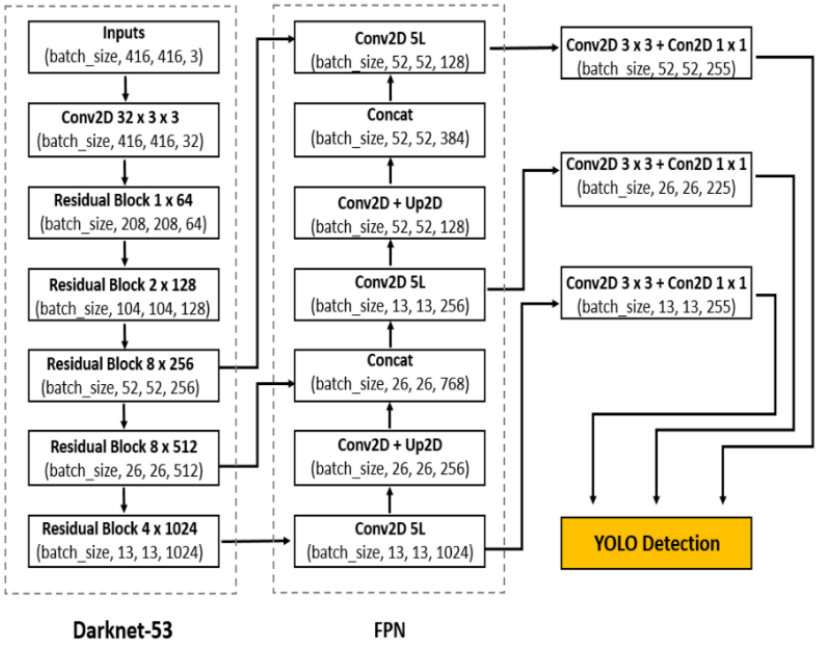


Figure 2.6: YOLO-v3 architecture [103].

move regularization from the backbone to the neck. Moreover, DropBlock regularization is used, which removes entire random blocks of features rather than individual ones, thereby improving generalization [105].

Finally, the **head** is responsible for defining the bounding box and providing a class prediction for each object found in the image. Additional information on the architecture of the model can be found in [106].

2.4 Explainable AI

Although AI is revolutionizing almost every field of work, complex models often act like black boxes, making their decisions difficult to understand and trust. This lack of transparency can be a hurdle to implement AI in real case scenarios. The manufacturing industry is no exception. To address this, Explainable AI (XAI) has emerged as a key area of research, aiming to make AI models more interpretable and trustworthy [108].

Doshi-Velez and Kim [109] define interpretability as the capacity to provide a human-understandable explanation for a result. In the context of medicine, an automated system needs to be highly explainable due to two main reasons: 1) the decisions made by the system would most likely affect users directly; and 2) the task itself could not be simple and could be subject to human judgement. Furthermore, the explanation techniques employed should aim to address multiple user-related issues. For instance, in the medical field, they have to meet the standards set by doctors and earn their trust, improve system transparency, evaluate the quality of the results, and enable the discussion of ethical issues, accountability, and fairness of the outcomes [109].

In some cases, it is possible to build a transparent and interpretable model for a decision-making process. Algorithms like decision trees have been applied successful in many domains, e.g. in predicting and explaining absenteeism risk in hospital patients [110],

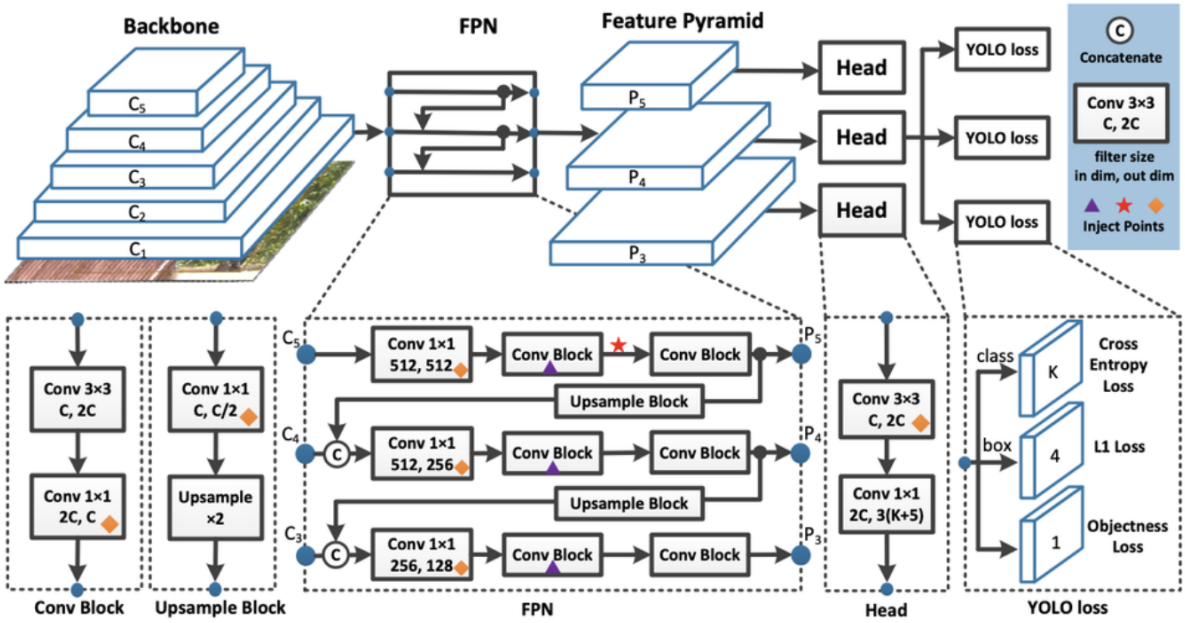


Figure 2.7: YOLO-v7 architecture [107].

explaining Smart Human Mobility [111], to name a few.

However, as the complexity grows, which is the case of Natural Language and Object Detection, there is a need to build a more robust model, sacrificing the interpretable nature of the inner workings of the model. Thus, this is the case where XAI assumes its greatest importance to give insights on how the predictions are made and why, hence giving more trust in the decision-making process.

The XAI techniques can be comprised into three key axes:

i Data explainability: This axis emphasizes the importance of understanding the data employed to train the model. It delves into aspects like potential biases within the data, feature selection processes, and overall data quality.

ii Model explainability: This axis concentrates on making the inner workings of the models interpretable. Techniques under this category aim to shed light on the model's decision-making process, fostering trust and transparency.

iii Post-hoc explanations: This axis encompasses techniques that provide explanations for specific predictions generated by the model. These techniques are particularly valuable for gaining a deeper understanding of individual model outputs and can be crucial for debugging or identifying potential issues.

In the literature, (ii) Model explainability is also refered as global explanations, in the sense that the explanations explain the model as a whole, while (iii) Post-hoc explanations are also mentioned as local explanations, since they explain a single prediction [112].

On the other hand, it is also necessary to perform an assessment of explanations. This is, evaluating the quality and effectiveness of the explanations produced by the XAI techniques. This step ensures that the explanations are not only comprehensible to users but also reliable and informative. By evaluating explanations, researchers can determine

their usefulness in building trust in AI systems [113].

The lack of explainability hinders trust and limits the potential benefits of AI in manufacturing settings. The good news is that the XAI techniques can also be applied in manufacturing environments [14].

Machine Learning Operations (MLOps) aims to automate the lifecycle of ML models, including deployment and monitoring. XAI methods can be integrated into MLOps pipelines to automatically assess the quality and explainability of models as they are deployed. This can help to ensure that robust and trustworthy AI systems are used throughout the industrial process.

By applying explainable AI techniques specific to manufacturing needs, we can unlock the full potential of AI for manufacturing, leading to increased efficiency, safety, and trust.

2.4.1 The need for explainable AI

The need for XAI can be discussed from various perspectives as shown in Fig. 2.8. The perspective groups below are to some extent based on the work in [114]:

- **Regulatory perspective**: Black-box AI systems are being utilized in many areas of our daily lives, which could be resulting in unacceptable decisions, especially those that may lead to legal effects. Thus, it poses a new challenge for the legislation. The European Union's General Data Protection Regulation (GDPR)[2] is an example of why XAI is needed from a regulatory perspective [115]. These regulations create what is called the "right to explanation", by which a user is entitled to request an explanation about the decision made by the algorithm that considerably influences them [116]. For example, if an AI system rejects one's application for a loan, the applicant is entitled to request justifications behind that decision to guarantee it is in agreement with other laws and regulations [117]. However, the implementation of such regulations is not straightforward, challenging, and without an enabling technology that can provide explanations, the "right to explanation" is nothing more than a "dead letter" [117–119].

- **Scientific perspective**: When building black-box AI models, we aim to develop an approximate function to address the given problem. Therefore, after creating the black-box AI model, the created model represents the basis of knowledge, rather than the data [120]. Based on that, XAI can be helpful to reveal the scientific knowledge extracted by the black-box AI models, which could lead to discovering novel concepts in various branches of science [115].

- **Industrial perspective**: Regulations and user distrust in black-box AI systems represent challenges to the industry in applying complex and accurate black-box AI systems [121]. Less accurate models that are more interpretable may be preferred in the industry because of regulation reasons [121]. A major advantage of XAI is that it can help in mitigating the common trade-off between model interpretability and performance [122], thus meeting these common challenges. However, it can increase development and deployment costs [115].

[2]https://www.privacy-regulation.eu/en/r71.htm

- **Model's developmental perspective**: Several reasons could contribute to inappropriate results for black-box AI systems, such as limited training data, biased training data, outliers, adversarial data, and model overfitting. Therefore, what black-box AI systems have learned and why they make decisions need to be understood, primarily when they affect humans' lives [115]. For that, the aim will be to use XAI to understand, debug, and improve the black-box AI system to enhance its robustness, increase safety and user trust, and minimize or prevent faulty behavior, bias, unfairness, and discrimination [123]. Furthermore, when comparing models with similar performance, XAI can help in the selection by revealing the features that the models used to produce their decisions [124, 125]. In addition, XAI can serve as a proxy function for the ultimate goal because the algorithm may be optimized for an incomplete objective [109]. For instance, optimizing an AI system for cholesterol control while ignoring the likelihood of adherence [109].

- **End-user and social perspectives**: In the literature of deep learning [119, 126, 127], it has been shown that altering an image such that humans cannot observe the change can lead the model in producing a wrong class label. On the contrary, completely unrecognizable images to humans can be recognizable with high confidence using DL models [127]. Such findings could raise doubts about trusting such black-box AI models [119]. The possibility to produce unfair decisions is another concern about black-box AI systems. This could happen in case black-box AI systems are developed using data that may exhibit human biases and prejudices [119]. Therefore, producing explanations and enhancing the interpretability of the black-box AI systems will help in increasing trust because it will be possible to understand the rationale behind the model's decisions, and we can know if the system serves what it is designed for instead of what it was trained for [119, 128]. Furthermore, the demand for the fairness of black-box AI systems' decisions, which cannot be ensured by error measures, often leads to the need for interpretable models [129].

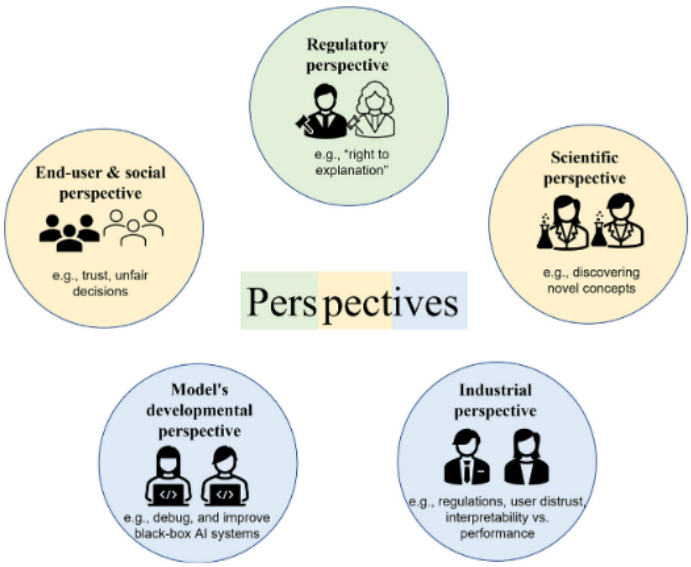


Figure 2.8: The five main perspectives for the need for XAI [115].

2.4.2 LIME

There are various methods for generating explanations for image classification models, which fall into 2 categories: global and local. Global explanations refer to explanations of how the model works, while local explanations refer to a single sample/prediction [130] and LIME [131] uses local explanations. Within this category, there are different types of approaches, such as feature importance, concept-based methods, example-based methods and uncertainty-based [130].

Two central concepts in LIME are those of superpixel and permutation matrix. In LIME, an image is first segmented, using a segmentation algorithm that will be explained on the subsection "Quickshift Segmentation Algorithm" of this section, into superpixels with each one representing a set of pixels that share similar color and location [131]. Their purpose is to simplify the problem by avoiding considering each pixel individually. The permutation matrix has one column for each superpixel. Each row of this matrix represents a variation of the original image, in which some super-pixels have been randomly "turned off" (filled black).

For each row of the matrix, LIME will generate the corresponding image and request the model being explained to classify it. In some cases, the model will detect the object being explained (usually when it is mostly visible in the image), while in others it will not, or it will detect other objects in the image. For each prediction, LIME records the class detected by the model and adds this information to the table.

In the end, a regression model is trained based on this table, whose purpose is to understand the importance (feature relevance) of each super-pixel for classifying the object to be detected. And it is based on this importance that heatmaps are ultimately created showing the importance of each super-pixel in the image.

The explanation of the prediction of an image classification model, in the case of LIME, is thus a heatmap superimposed on the image, which shows the importance of each region of the image for the model that classifies it. In this heatmap, the greener a super-pixel is, the more relevant it is for the predicted class. Red superpixels, on the other hand, highlight regions of the image that were relevant for detecting an object of a different class.

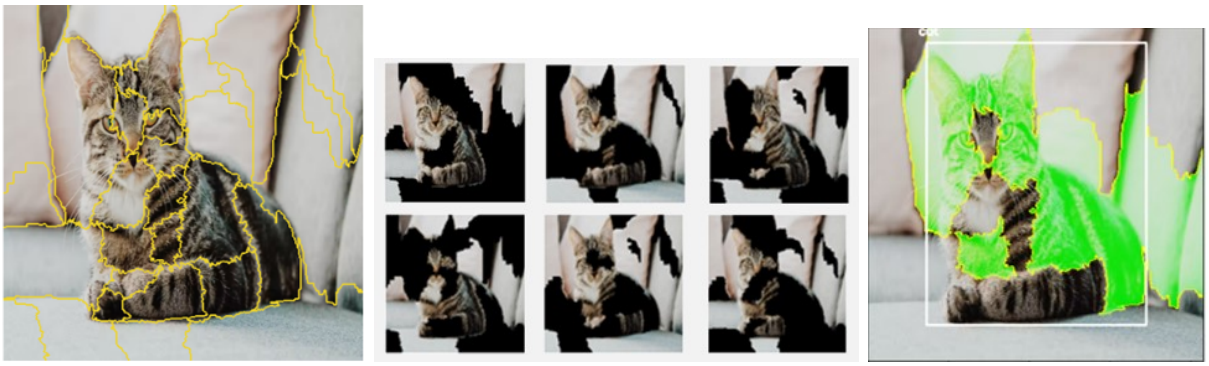
Figure 2.9 illustrates some of the pertinent LIME concepts. From left to right, the following are shown: a) the outcome of the super-pixel definition; b) the top left visualization of a permutation matrix including five random versions of the original image; and c) the explanation of the cat in the LIME-generated image.

Quickshift Segmentation Algorithm

Quickshift [132] is one of the most popular superpixel segmentation algorithms. Its principle is based on an iterative mode-seeking that identifies modes in a set of data points.

In detail, Quickshift is a density-based clustering algorithm used for finding patterns in data. It is similar to MeanShift in that it calculates the direction of a point x_i to move next. It moves to the area with large density and forms a cluster with its surrounding points until all the points in the data are read [133].

However, unlike MeanShift, which iteratively moves each point toward the local mean,



(a) (b) (c)

Figure 2.9: Key LIME concepts illustrated visually: super-pixels, permutations, and explanation (heatmap).

QuickShift simply moves point x_i to the nearest sampled point with increasing density $P(x)$, and restricts the steps to moving over the sampled points without iteration, evaluating the density function only $O(n)$ times. This makes Quickshift faster, and its time complexity is dominated by spatial neighborhood queries that can be executed quickly with the help of KD-trees. Figure 2.10 shows the clustering process using the QuickShift algorithm [133].

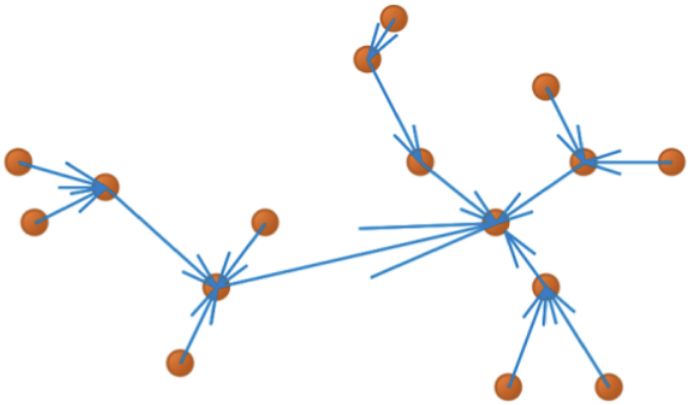


Figure 2.10: QuickShift Clustering Process [133].

To illustrate QuickShift, suppose we are clustering a set of points in 2D space. First, we calculate the point density of all points using the K-Nearest Neighbors (KNN) density estimation method, fix the number k of samples around a sample point x, and then search according to the distance from near to far, until we find k neighboring points. We then take the distance from the K^{th} (farthest) sample y to z as the unit length of the volume V. We calculate the point density using Equation 2.6:

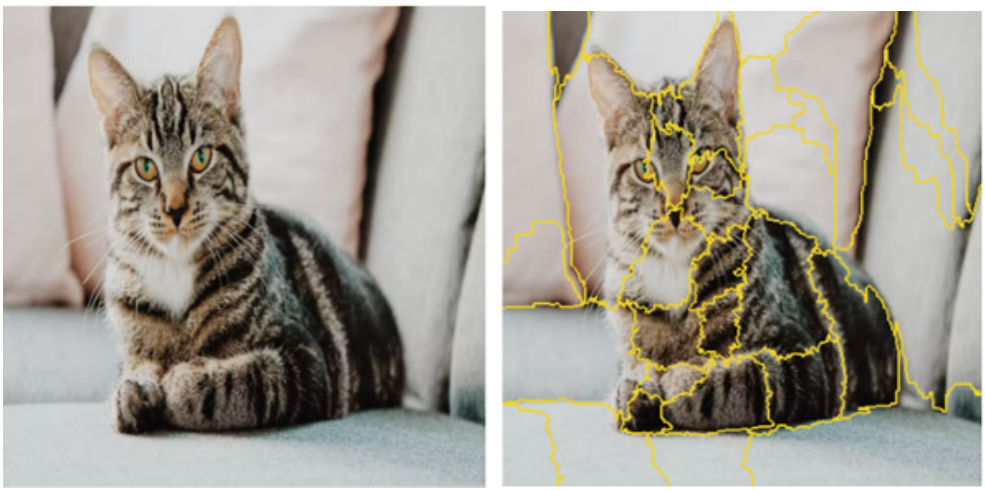
$$P(x) \simeq \frac{k}{NV} = \frac{k}{N \cdot C_D \cdot R_k^D(x)} \tag{2.5}$$

where k is the number of neighbors to look for, c_D is the volume per unit range, N is the total number of data points, $R_k(x)$ is the distance from point x to its k^{th} neighbor, D is the sample dimension, and V is the volume of the region [133].

Once the calculation is done to start moving, we start with a point centered at x_i (randomly chosen) and move the point x_i to a region of higher density by moving the center point x_i to the nearest neighbor whose density function $P(x)$ increases. The core of the algorithm is the rules of how to move the $y(xi)$ point trajectory [133]. The formula is as follows:

$$y_i(1) = \arg \max_{J:d(x_j,x_i)} \frac{P_j - P_i}{D_{ij}} \tag{2.6}$$

Here, D_{ij} is the distance between point i and point j, and the KD-tree search radius for the neighborhood search is also called the neighborhood distance threshold τ. Numerical approximation of the gradients of $P_j - P_i/D_{ij}$ and p along the $x_j - x_i$ direction. When the point stops moving, its density value is locally maximal, and it forms a cluster with its surrounding points. The distance between this cluster and the historical cluster is calculated. If the distance is less than the threshold D, it is merged into the same cluster, and if it is not satisfied, it forms a cluster by itself until all data points are selected [133]. Figure 2.11 shows the final result of the quickshift segmentation.



(a) Raw image of a cat (b) Quickshift superpixels

Figure 2.11: Superpixel segmentations by Quickshift

LIME: Identified Problems

In order to get a better understanding of how long LIME took to execute, the average time, expressed in seconds, was measured for each of the primary steps of LIME method (Table 2.1). It is evident that the process of generating the dataset containing the permutations and the subsequent classification by the model requires more time than the other stages. In other words, the LIME algorithm takes longer to execute when utilizing an image classification or detection model to classify each generated permutation.

Another drawback of the LIME approach in the industrial context is that "turned off" superpixels are mistaken for defects by the YOLO model. This ultimately has a negative impact on the final explanations produced by LIME because it results in a poorly constructed dataset of permutations with false confidence values, identified by YOLO, and the Ridge model gives each superpixel the incorrect weight because of these wrong

Phase	Segmentation	Permutations Dataset Creation and Model Classification	Model Regression Fit	Feature Selection	Total Duration
Mean Duration (seconds)	6.922890s	352.536149s ≈ 6 min	0.001505s	0.001379s	359.461s ≈ 6 min

Table 2.1: Mean duration of each LIME phase in 10 different images with 1000 permutations each

confidence values, which in turn causes LIME to produce weak explanations. In Chapter 4, this issue will be covered in further detail along with respective results and YOLO model detections (Figure 4.1).

Related work

Abdullah et al. [134] presented B-LIME, an enhancement of the LIME, aimed at improving the interpretability of DL models for classifying cardiac arrhythmias from ECG signals. B-LIME addresses the limitations of LIME by incorporating temporal dependencies inherent in ECG data, thereby generating more meaningful and credible explanations.

The authors introduce a novel data generation technique and an explanation method that ensure local fidelity, allowing for accurate insights into model predictions. They apply B-LIME within a hybrid CNN-GRU model, demonstrating its effectiveness in highlighting critical areas of the ECG signal, such as the QRS complex, which are vital for clinical diagnosis. The results indicate that B-LIME provides clearer and more relevant explanations compared to LIME, enhancing the trustworthiness of AI in healthcare applications.

Visani et al. [135] addressed the challenges of interpretability in ML, particularly in the medical field where decision-making is critical. They focus on a LIME's instability due to randomness in its sampling process, which can undermine trust in its explanations. To tackle this LIME's issue, the authors propose a new framework called OptiLIME, which aims to optimize the trade-off between explanation stability and adherence to the original ML model. This framework allows practitioners to select the desired level of stability while ensuring that the explanations remain relevant and mathematically sound.

Particularly, a proposal of a small optimization for this issue of LIME randomness on generating permutation will be addressed on Chapter 5.

Chapter 3

Measuring the quality of explanations in computer vision

3.1 Problem Statement

In ML, models are generally evaluated during their training stage. Moreover, all evaluation metrics (e.g. Mean Absolute Error, Root Mean Square Error) are averages of the model's performance across the test set. So there might be significant variations in performance that are hidden by their average. That is, the model might be very good at predicting in a region of the solution space, and very bad in another. On the other hand, it also often happens that a model that has shown good performance during training, later exhibits a degrading performance. This happens especially in scenarios in which there may be concept drift [136], such as when weather or environmental conditions change, or when a machine wears out with use.

It is also frequent that when models are in production, it is no longer possible to evaluate them on up-to-date labeled data. This happens because labelled data is costly to acquire and this is done only for training purposes. Then, either one keeps evaluating the model during production on outdated labeled data, or one simply trusts the model when it is in production. This is frequent in industrial contexts, in which human resources are scarce and expensive, and data acquisition costly.

In this chapter, we propose to use artifacts from XAI to monitor the quality of a model, mimicking what a human would do. Specifically, we propose and evaluate seven new metrics that might be used to assess the quality of a CV model. These metrics are calculated based on two main elements: 1) the bounding box of the defect detected by the object detection model (a state-of-the-art YOLO v7 model which we specifically trained for this task); and 2) the heatmap produced by LIME.

The proposed metrics, in which G and R denote the set of pixels marked as green and red, respectively, and B denotes the pixels inside the bounding box predicted by the model, are as follows:

$$M_1 = \frac{|G \cap B|}{|G \cup B|} \times 100 \qquad (3.1) \qquad\qquad M_2 = \frac{|G - B|}{|G|} \times 100 \qquad (3.2)$$

$$M_3 = \frac{|R \cap B|}{|G|} \qquad (3.3) \qquad\qquad M_4 = \frac{|R|}{|G|} \qquad (3.4)$$

$$M_5 = \frac{1}{N} \sum_{i=1}^{N} \sqrt{(x_i - x_{\text{center}})^2 + (y_i - y_{\text{center}})^2} \qquad (3.5)$$

$$M_6 = \frac{|G|}{I_h * I_w} \times 100 \qquad (3.6) \qquad\qquad M_7 = \frac{|G|}{|B|} \qquad (3.7)$$

in which:

- M_1 - Measures the percentage of overlap between the bounding box predicted by the model and the region of green pixels, dividing the intersection of both areas by its union
- M_2 - Measures the percentage of green pixels that are outside the predicted bounding box
- M_3 - Quantifies the amount of red pixels that are inside the predicted bounding box in face of the amount of relevant pixels
- M_4 - Measures the relation between the number of green and red pixels
- M_5 - Measures the average distance between the green pixels and the center of the bounding box predicted by the model, where N is the number of green pixels, (x_i, y_i) are the coordinates of each green pixel, and $(x_{\text{center}}, y_{\text{center}})$ are the coordinates of the center of the bounding box
- M_6 - Measures the percentage of the image marked as green
- M_7 - Measures the relation between the number of green pixels and the size of the bounding box

3.2 Methodology

In order to validate the proposed metrics, a randomly selected group of researchers at INESC TEC and related institutions were invited to provide quantitative feedback. Twenty seven participants took part in the study. The following methodology was implemented. We started by selecting 24 images with random but familiar images (e.g. car, cat, dog, bike, person). We chose to use familiar images instead of images of defects on fabric, so that the specificities of the domain do not constitute an issue for the participants in the study.

Then, we used a YOLOv3 model [97], pre-trained on the ImageNet 1000-class competition dataset [137] to predict on the 24 images. Next, we used LIME to generate explanations for each of these predictions, and overlaid the original images with both the bounding box of the object, so that the participants could know the object being explained, and the heatmap generated by LIME.

These images, three of which are depicted in Fig. 3.1 were then made available through an online questionnaire. The questionnaire started with a brief introduction, in which

the concept of explainability was introduced, and two sample images with explanations. Thus, in this introduction, the participants also learned how to interpret the explanations generated by LIME.

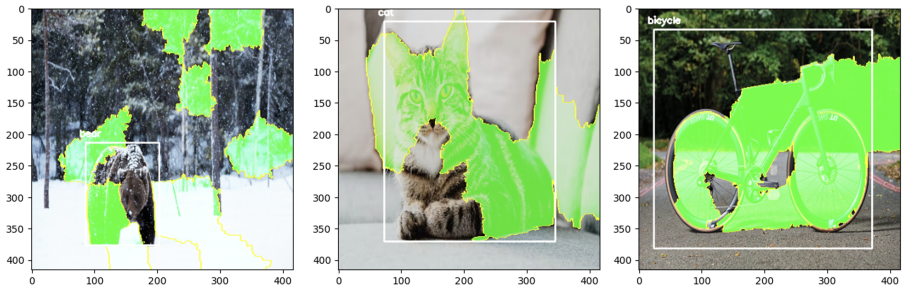


Figure 3.1: Three of the twenty-four images used in the questionnaire, with predictions (bounding box) and explanations for bear, cat and bicycle, respectively.

Participants were presented with the 24 images, and were requested to rate the quality of each prediction/explanation on a 5-point Likert scale in which 1 denotes a very poor prediction/explanation and 5 a very good one. Finally, the proposed metrics were calculated for each of the 24 images, and their value analyzed against the feedback of the participants. Section 3.3 details the results of this analysis.

3.3 Results

Concerning the data collected through the questionnaire, in which participants rated the quality of each prediction/explanation, the results are visually depicted in Fig. 3.2. It can be observed that there are cases in which the participants generally agreed that the prediction/explanation was either good (e.g. toilet, bird) or bad (e.g. horse, bike). However, there are also cases in which the ratings are disperse across the 5-point scale (e.g. elephant, fire hydrant, goat). This shows that: 1) the images used cover generally all the cases (i.e. good, average and bad examples); and 2) the problem is complex and even humans might not fully agree on what a good prediction/explanation is.

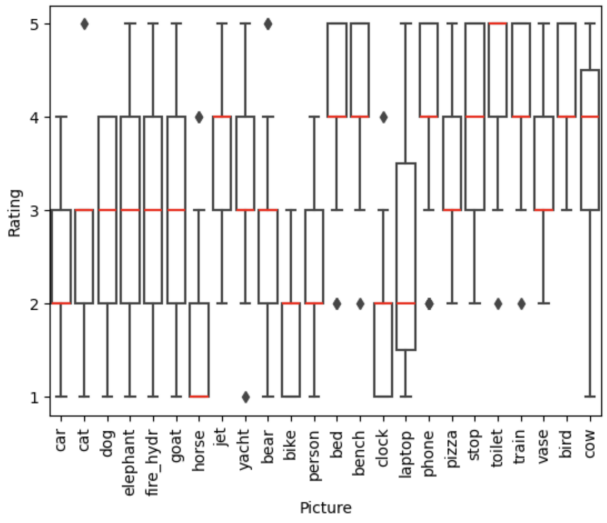
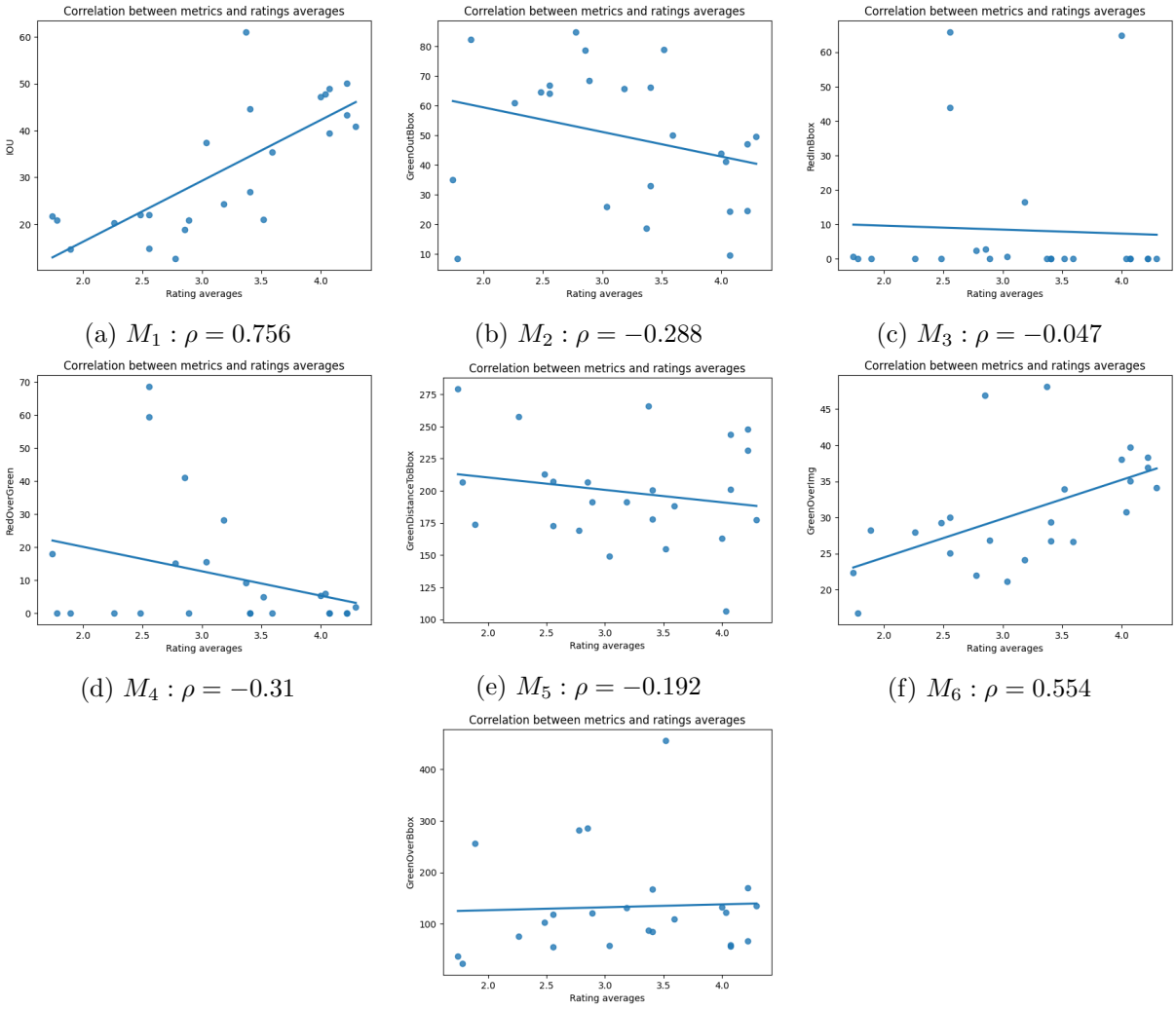


Figure 3.2: Distribution of ratings for the pictures among the 24 participants.

Fig. 3.3 visually depicts the correlation between each of the proposed metrics and the average participants' ratings. Each dot represents one of the 24 pictures, for which the participants' ratings have been averaged. It can be seen that the in two of the metrics (M_1 and M_6), the correlation is particularly strong.



(a) $M_1 : \rho = 0.756$ (b) $M_2 : \rho = -0.288$ (c) $M_3 : \rho = -0.047$

(d) $M_4 : \rho = -0.31$ (e) $M_5 : \rho = -0.192$ (f) $M_6 : \rho = 0.554$

(g) $M_7 : \rho = 0.046$

Figure 3.3: Plot and correlation of the average rating of each picture against each metric.

The values of the Pearson Correlation Coefficient between each metric and the participants' ratings is provided in Table 3.1. Specifically, we grouped the ratings of each prediction/explanation and calculated its mean and median values. Then, we computed the Pearson Correlation Coefficient (PCC) between these and the values of the metrics. It can be seen that significant correlations were found between the average value of the ratings and two metrics: M_1 and M_6 (0.756 and 0.554, respectively). There are also two cases of weaker negative correlations, for metrics M_2 and M_4 (-0.288 and -0.31, respectively).

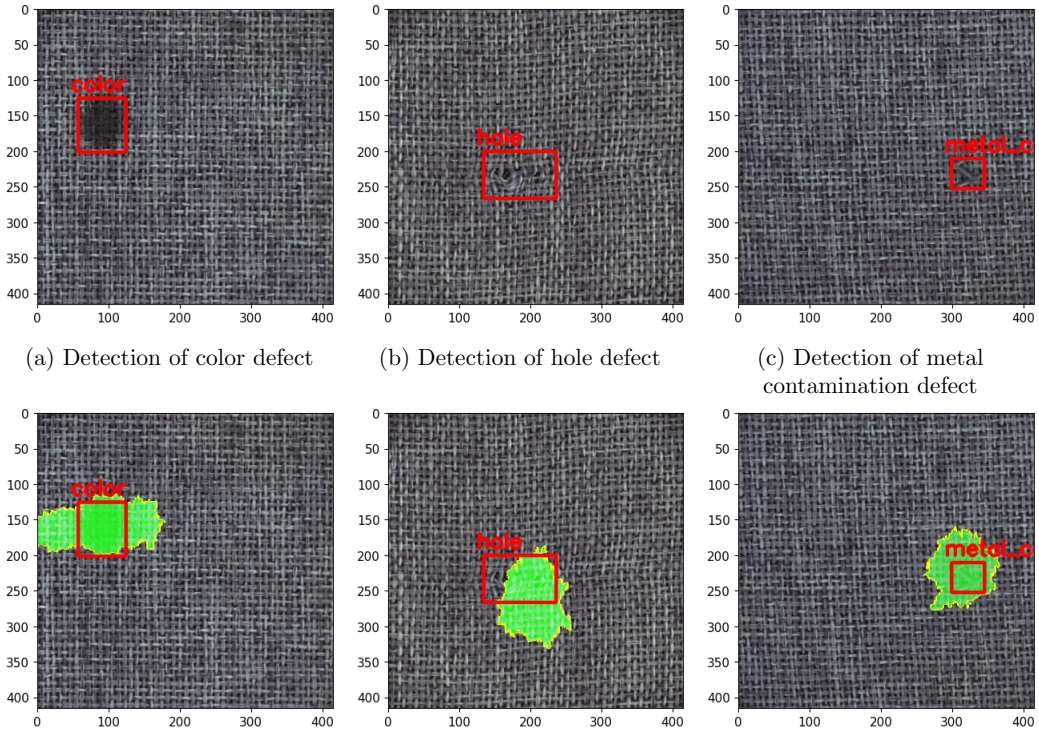
The strong positive correlation observed for M_1 (0.756) suggests that the respondents associate the quality of a prediction/explanation with and increased intersection between the area of the object being explained and the explanation itself. The moderate correlation of M_6 (0.554) indicates that people tend to interpret an explanation as better if larger areas of it are marked as green.

Table 3.1: Pearson Correlation Coefficient between the proposed metrics and the mean and median of the ratings.

Metric	ρ **Mean**	ρ **Median**	**Metric**	ρ **Mean**	ρ **Median**
M_1	0.756	0.588	M_2	-0.288	-0.112
M_3	-0.047	-0.061	M_4	-0.31	-0.239
M_5	-0.192	-0.244	M_6	0.554	0.492
M_7	0.046	0.189			

The moderate negative correlation observed for M_2 (-0.288) points out that the more the explanation spreads outside the bounding box of the object being explained, the worse the prediction/explanation is evaluated. A similar correlation was observed for M_4 (-0.31), which indicates that whenever there are more red pixels, the prediction/explanation is perceived as worse.

A thorough evaluation of image classification techniques such as YOLO is made easier by the MVTEC [1] Anomaly Detection Dataset [138], which comprises 5354 high-resolution color images of various object and texture categories. Figure 3.4 reveals images of fabric with defects and the corresponding detection (top row), carried out by our model. The most pertinent superpixels are emphasized in the bottom row of the image, which shows the explanations produced by LIME on a selection of the Carpet texture category images from the MVTEC dataset.



(a) Detection of color defect (b) Detection of hole defect (c) Detection of metal contamination defect

(d) Explanation for color (e) Explanation for hole (f) Explanation for metal contamination

Figure 3.4: Examples of defect detected in three samples of fabric (top row) and corresponding explanations (bottom row).

[1] https://www.mvtec.com/company/research/datasets/mvtec-ad

Chapter 4

Addressing the Limitations of LIME for Explainable AI in Manufacturing

4.1 Problem Statement

LIME is a technique designed to explain predictions of ML models, including those used in CV tasks. The core idea behind LIME is to provide explanations that are both interpretable and locally faithful to the model's behavior around the specific prediction being explained, which entails several steps.

First, for any given image to be explained, LIME generates several perturbed versions. For object detection tasks, as is the case, this often involves modifying random regions of the image. These regions are called super-pixels, which are obtained using a segmentation algorithm, previously explained in Section 2.4.2. The vanilla version of LIME paints the perturbed superpixels black. This process produces a collection of modified images that differ more or less from the original, depending on the number of superpixels that were perturbed [139].

Next, the modified images are passed through the underlying object detection model being explained, to obtain predictions for each one. This helps understand how different parts of the image influence the model's detection results: if the prediction for two perturbed versions of the image changes significantly when one superpixel was changed, that region is most likely important for the prediction.

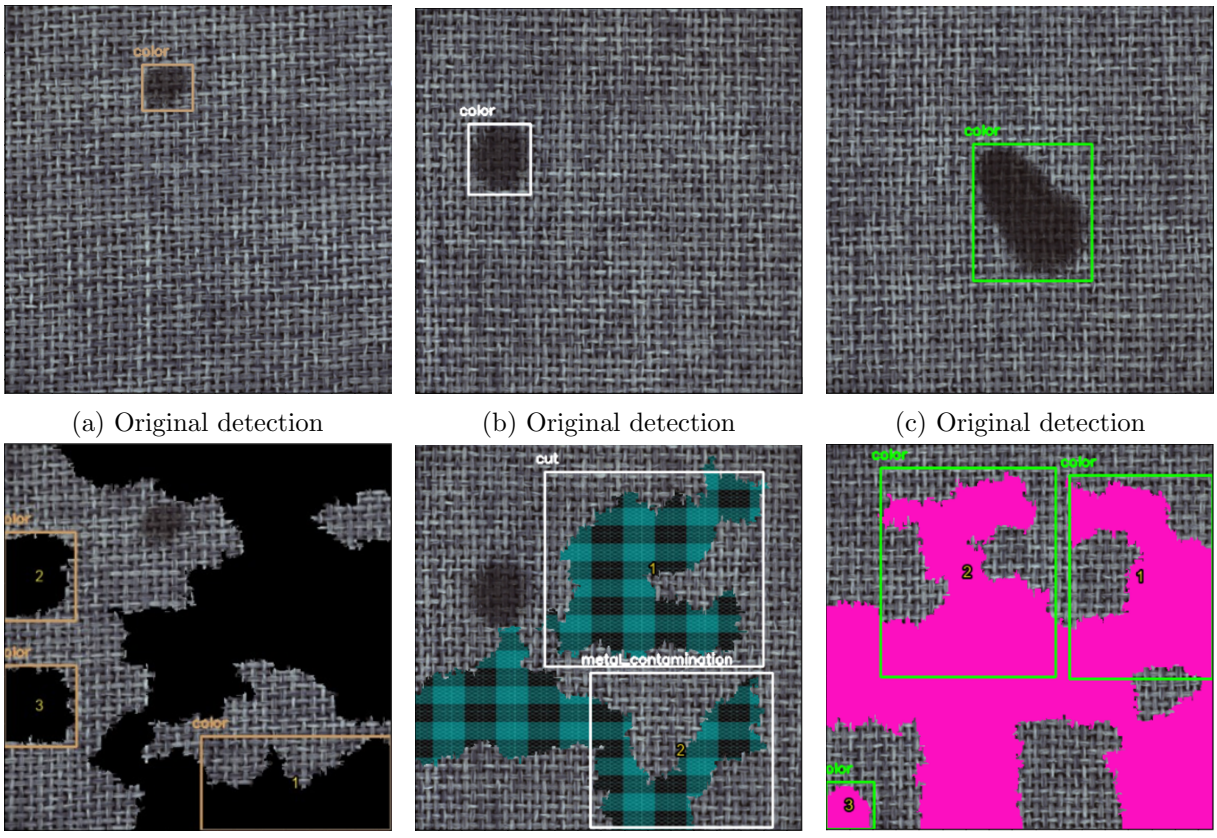
A local surrogate interpretable model is then trained, such as a linear regression, to approximate the behavior of the complex model in the local region defined by the perturbed images. The simple model aims to capture the relationships between features (the perturbed super-pixels) and the prediction output. One can then look at the relevance of each feature in the surrogate model, to identify the super-pixels that are relevant in the original image to classify the object.

One key aspect in this process is on how to fill in the perturbed sections of the images that are generated. The vanilla version of LIME fills them in black. However, while this tends to work generally well in generic tasks, there are cases in which this black artifact that is created in the image might be confused by the underlying model with an actual object. This is exemplified in Figure 4.1. The top row shows three detections of the defect color,

which may be caused by oil or paint stains in the fabric. The bottom row shows three examples of perturbations generated during the use of LIME to explain the predictions, using different backgrounds, and the output of the underlying model for each one.

In d), the actual defect is not detected (false negative), and the model detects three color defects in the super-pixels that were painted black by LIME in this perturbation (false positives). Using other colors or even more complex patterns might have similar results. In e) and f) a complex background and the color pink were respectively used instead of black. In the case of e) the actual defect, while visible, is not detected (as in d)), while two of the superpixels are mistakenly classified by the model as cut and metal contamination, which are another two defects that the model has been trained on. In f) the model wrongfully detects three color defects on super-pixels that were painted pink in the perturbation, when actually no defect should have been detected since the actual defect is hidden by the perturbation.

Thus, when using LIME, it is fundamental that an appropriate color or background is selected for filling in the perturbed superpixels, so that these are not mistaken by actual objects by the underlying model. While this may be laborious to achieve on a case-by-case basis, in Section 4.2 we propose a methodology to do this automatically for any use-case.



(a) Original detection (b) Original detection (c) Original detection

(d) Detection after perturbation (FN + 3xFP) (e) Detection after perturbation (FN + 2xFP) (f) Detection after perturbation (3xFP)

Figure 4.1: Top row shows textile defect detections by the underlying YOLOv7 model. Bottom row shows cases in which the perturbations are mistaken by actual defects (FP: False Positive) and/or in which the actual defect is no longer detected (FN: False Negative).

4.2 Methodology

This section describes the methodology followed to address one of the key limitations of LIME, described in the previous section and made evident in Figure 4.1. We aim to automate the process of selecting a suitable color or pattern for LIME to generate perturbations for each specific use-case, ensuring that the underlying model, whatever it is, does not mistake perturbed super-pixels with the objects/defects being detected. Doing this by hand may be time-consuming and not exhaustive, so automating the whole process is advantageous for the manufacturing domain.

The proposed methodology has two stages. In the first stage we propose an automated methodology to select suitable colors or patterns for LIME perturbations in various use cases. This ensures the underlying model doesn't misinterpret perturbed superpixels as actual objects. Manual selection is time-consuming and potentially incomplete, making automation valuable for manufacturing applications.

The presented methodology begins by taking a set of images showcasing defects along with their respective bounding boxes, which provide the coordinates and class information for each defect. To ensure comprehensive analysis, these images should include representative examples of every defect class being targeted. Additionally, the methodology requires a collection of colors and/or background patterns that the user aims to evaluate as potential options for filling in the perturbed superpixels.

For each input image and each background being tested, we generate a predetermined number of copies with random perturbations. The underlying model being analyzed is then used to make predictions on each of these perturbed images, which are subsequently compared with the model's predictions on the original image. The general intuition is that if the model's predictions for images with visible defects remain consistent with its predictions on the original image, the background color or pattern being tested does not significantly influence the model. Conversely, when the defect is not visible, the model should ideally not detect any defects. If it does, this indicates that the model is likely mistaking the perturbed superpixels for non-existent objects in the image, resulting in false positives.

The model is then tasked with making predictions on each perturbed image, and the performance of each background color or pattern is evaluated for each defect class. Specifically, we consider four scenarios: True Positive (TP) where the model correctly detects a defect in the perturbed image that matches the defect in the original image; False Positive (FP) where the model incorrectly detects a defect in the perturbed image that either does not exist in the original image or belongs to a different class; and False Negative (FN) where the model fails to detect a defect in the perturbed image despite it being visible. True Negatives (TN) are not counted in object detection as they encompass all the bounding boxes that are not predicted.

It's important to note that multiple cases can occur within the same image. For instance, Figures 4.1d and 4.1e illustrate examples of multiple False Positives and a False Negative in a single image, where the original defect is not detected in the perturbed version, but non-existent defects are detected. The results of applying this methodology to the specific use case are detailed in the next subsection, with the performance for each tested background reported by defect type.

In the second stage of the methodology, the focus is on training a defect detection model leveraging a specific dataset. This dataset will be comprised of permutations generated using the background that yielded the most successful results in the previous analysis. By training the model on this data, the goal is to familiarize it with the "background" data.

This approach serves a crucial purpose. When the model encounters a new explanation and needs to predict the generated permutations, it will be less likely to confuse the background with existing defects.

Upon completion of model training, the evaluation phase begins. The mAP metric is used to analyse the model's performance. This metric provides a quantitative measure of the model's effectiveness.

Following mAP evaluation, the results are compared with those obtained from the subsequent analysis stage. This comparison allows for gauging the efficacy of the model training approach. By analyzing the strengths and weaknesses identified in the comparison, a determination can be made regarding the model's success in identifying defects in images with the background without confusing the background with defects and generating explanations.

This second stage had two approaches. First, a defect detection model was trained using only the images of the permutations obtained by LIME, using the background where the best results were obtained for generating them. After analyzing the performance of the model created, it can be concluded that there is a limitation because as the training was only carried out on images with that specific background, the model ends up having greater difficulty in detecting defects when faced with images without the background.

Given the analysis, we decided to create a new model using the dataset created to train the initial model, adding to it a second dataset made up of the images generated in the permutations with the specific background so that the model would not lose quality when faced with images without the background. After analysis, it can be concluded that this second approach works and the model obtains better results.

4.3 Results

This section presents the results of evaluating the proposed methodology. In this use case, in the domain of textiles defect detection, we considered 9 different colors and patterns to create the perturbations in LIME, and use the proposed methodology to find the most suitable background for this specific domain. The colors and backgrounds were defined arbitrarily since, due to the nature of DL models, it is virtually impossible to estimate which will not be confused by the model with the defects being detected.

We evaluate the suitability of each background through the ability of the baseline model to identify defects in the perturbed images, using the metrics of the PASCAL VOC object detection challenge, which are based on the Precision x Recall curve and Average Precision [140]. In the computation of the metrics we used an IoU = 0.5.

In order to compute these metrics, it is also necessary to define the threshold at which a defect is considered visible in the perturbed images. Indeed, when LIME creates the perturbed images, three different cases can occur: the defect is completely visible, the defect is partially hidden by one or more perturbations, or the defect is completely hidden.

Images that are partially hidden must still be labeled as containing the defect or not, so that the model can then be accurately evaluated. For the results reported herein, we assumed that a defect is only considered visible if 10% or less of the area of its bounding box is covered by the perturbation.

Table 4.1 shows the counts of the visible defects in the 2000 perturbations generated for each background (18.000 in total) in the Ground Truth column, and the detections by the underlying model in the Detections column. It is clearly visible that the model tends to mistake certain backgrounds with actual defects, as happens with the color defect in the first two colors and in patterns 1 and 3, or in the cut defect in pattern 4.

Background	**Ground Truth**				**Detections**			
	color	**cut**	**hole**	**metal**	**color**	**cut**	**hole**	**metal**
(0,0,0)	146	52	53	134	9136	127	22	53
(255,15,192)	116	33	71	142	10449	0	9	10
(255,255,255)	123	37	52	139	73	3672	1041	383
(0,177,64)	125	43	55	152	155	29	180	173
Pattern 1	108	39	49	171	8193	50	27	54
Pattern 2	108	44	58	142	232	127	249	227
Pattern 3	139	41	57	143	6387	0	14	20
Pattern 4	101	39	64	148	86	6083	259	0
Pattern 5	118	33	59	147	11	4031	415	5137

Table 4.1: Number of defects of each type visible in the 2000 images with perturbations (ground truth) vs. the number of defects detected by the model (detections), by background.

Table 4.2 summarizes the results for the different backgrounds compared. Specifically, it shows the average precision by class, and the mAP. It allows to compare the different backgrounds in terms of how they confound the model, and also how the same background is more or less confused with each of the defects.

For this specific use case, the best background is Pattern 2, with an mAP = 0.62, followed by color R=0, G=177, B=64, with an mAP = 0.43. All the remaining backgrounds have very poor average results, despite some acceptable results for some specific classes.

The mAP of every baseline was evaluated in both the YOLOv7 model's initial training version and its additional trained versions. The baseline only considered the original images of the defects, whereas the lines of each background used images of the defects with the disabled superpixels filled with the color of each background.

As can be seen in Table 4.2, the original YOLOv7 model, in the baseline line, achieved maximum AP values for the color, cut and hole defects, only achieving a slightly lower value for the metal contamination defect, demonstrating that the original YOLOv7 model does not detect this type of defect with better precision, however this model ended up performing very well as it achieved a mAP close to 1.

The 2nd version of the model, represented by baseline v2, obtained very poor AP values in comparison to the baseline original version of the YOLO model in the following row. It only achieved a minimally acceptable AP value for the color defect and low AP values in the remaining defects, even reaching a minimum AP value for the cut defect. This

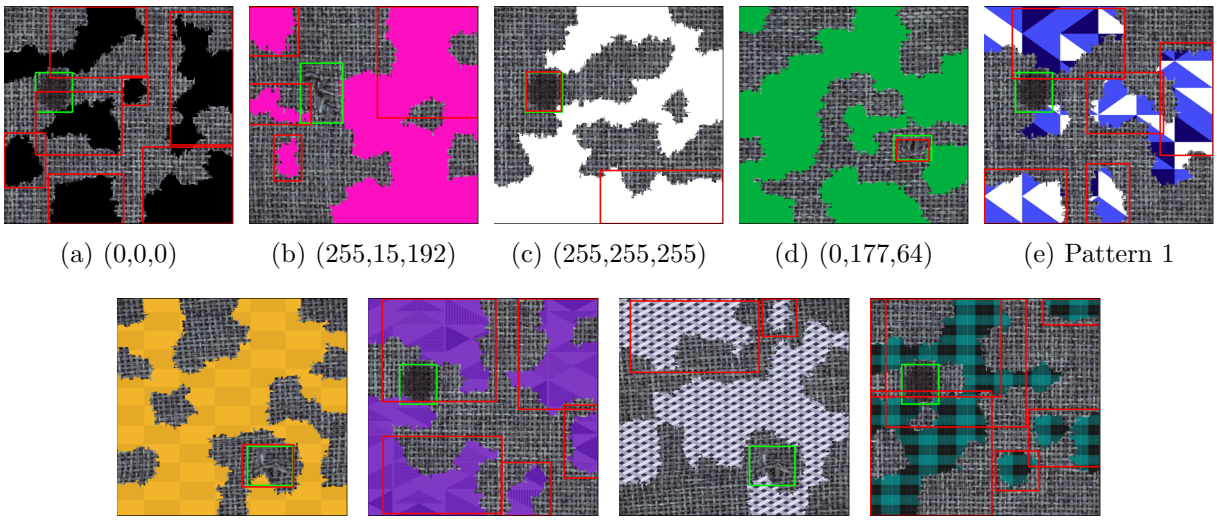
suggests that this version of the model performs poorly in detecting this kind of defect in the original images, as evidenced by the poor mAP value.

Lastly, the third version of the original YOLOv7 model, baseline v3, achieved maximum AP values for all defect types and, as a result, a maximum mAP value. This indicates that the model is capable of effectively detecting all defect types in the original images.

Background	**AP per class**				**mAP**
	color	**cut**	**hole**	**metal**	
(0,0,0)	0.0	0.08	0.15	0.33	0.14
(255,15,192)	0.0	0.0	0.11	0.07	0.05
(255,255,255)	0.44	0.0	0.03	0.14	0.15
(0,177,64)	0.50	0.15	0.41	0.64	0.43
Pattern 1	0.0	0.01	0.10	0.14	0.06
Pattern 2	**0.63**	**0.51**	**0.64**	**0.70**	**0.62**
Pattern 3	0.0	0.0	0.11	0.14	0.06
Pattern 4	0.38	0.0	0.04	0.30	0.18
Pattern 5	0.09	0.0	0.04	0.06	0.05
Baseline	1.0	1.0	1.0	0.8	0.95
Baseline v2	0.60	0.0	0.25	0.20	0.26
Baseline v3	1.0	1.0	1.0	1.0	1.0

Table 4.2: Performance of the model when predicting on images with permutations generated using different colored/patterned backgrounds. The last 3 rows shows the performance of each YOLOv7 model trained versions when predicting on the original images.

Figure 4.2 shows the same image, with a color defect, and one randomly selected perturbation for each background. The ground truth is shown in green (when visible) and the detections of the underlying model are shown in red. Pattern 2, depicted in 4.2f, was the background that achieved the best results for this specific use-case, out of the 9 tested.



(a) (0,0,0) (b) (255,15,192) (c) (255,255,255) (d) (0,177,64) (e) Pattern 1

(f) **Pattern 2** (g) Pattern 3 (h) Pattern 4 (i) Pattern 5

Figure 4.2: Sample (out of 18.000) of one randomly selected perturbation generated by LIME for each background, ground truth (green), and detections by the underlying model (red).

Chapter 5

Improving the efficiency and efficacy of LIME in industrial scenarios

5.1 Improving computational efficiency through data minimization

When it comes to generating explanations, LIME is a slow method that is not practical for use in the industrial sector, specifically when developing explanations that are desirably in real-time, so that the loom can be stoped when the deffect first occur, in order to minimize waste.

The average duration in each LIME phase is shown in Table 2.1, presented in Section 2.4.2. The average duration was calculated from a set of 10 distinct images, with 1000 random permutations produced for each image.

As detailed in the Table 2.1, the "Permutations Dataset Creation and Model Classification" phase is the one that lasts the longest. During this stage, LIME uses the randomly generated permutations from the image dataset to train the YOLOv7 model.

Since more permutations will be produced and the YOLOv7 model will need more time to train, the higher the size of the permutation dataset, the better the explanation supplied, but the longer it will take. Stated differently, if the objective was to accelerate the explanation-generating process, then only decreasing the permutation dataset's size would be enough. However, this would have the unintended consequence of producing worse explanations due to the smaller permutation dataset.

In order to address this problem, the subsequent subsection offers a method that guarantees a higher diversity of permutations in the reduction's final output while decreasing the size of the permutation dataset without affecting the explanation in the end.

To minimize the amount of random permutations produced by LIME, while maximizing variety and covering the majority of possible combinations between them, a dataset was divided into a predetermined number of clusters using the well-known unsupervised marchine learning technique K-Means.

K-means algorithm is the most commonly used simple clustering method. For a large number of high dimensional numerical data, it provides an efficient method for classifying

similar data into the same cluster [141].

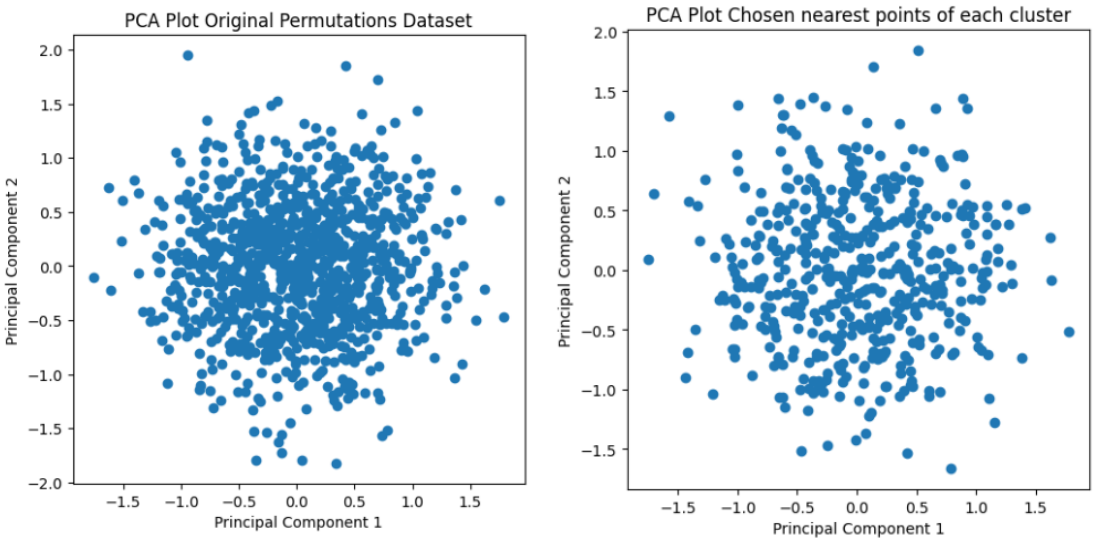
Subsequently, the defined number of permutations is processed by K-Means to decrease it to 50% of its initial size. In this case, 1000 permutations were defined, and after reducing to 50%, a total of 500 permutations were obtained.

The K-Means approach, which would produce 500 clusters, was applied for this reduction process. The Euclidean distance was used to determine the permutation that was closest to each cluster's centroid.

The reduction from 1000 to 500 permutations are resulted in Figure 5.1, that was generated using Principal Component Analysis (PCA). Adbdi et al. [142] define PCA as a multivariate technique that analyzes a data table in which observations are described by several inter correlated quantitative dependent variables. Its goal is to extract the important information from the table, to represent it as a set of new orthogonal variables called principal components, and to display the pattern of similarity of the observations and of the variables as points in maps.

Stated differently, PCA is a method that plots an array of arrays, each with a large size, in two dimensions (x,y). In this particular case, representing each permutation coordinate (with 50 dimensions) in a 2-D plot would be unfeasible due to the approximately 50 superpixels that each permutation comprises. However, PCA plots can be used to achieve this.

As demonstrated, the permutations have become somewhat more dispersed (and not as concentrated as shown in sub-figure 5.1a). This indicates that, following the implementation of the permutation reduction strategy in sub-Figure 5.1b, there is a diversity of permutations and a decrease in permutations that are similar to one another.



(a) 1000 Permutations

(b) 500 Permutations

Figure 5.1: PCA Plots with reduction from 1000 to 500 permutations.

5.2 Results

The presented results were obtained using a laptop with CPU AMD Ryzen 5 5600H and GPU NVIDIA GeForce GPU RTX 3060.

For a total of 20 images, 5 of each of the 4 types of defects (color, cut, hole, and metal contamination) were taken, and each of the metrics (described in Section 3.1) were calculated for each image in order to better understand the effects of the permutation reduction strategy that was presented.

The final number of permutations that have been derived from the original 1000 permutations is indicated by each row in Table 5.1 and Table 5.2.

Table 5.1 shows the results of the metrics that obtained interesting results (M1, M2, M3 and M4) in an online questionnaire described in Section 3.2, when reduced from 1000 permutations to each permutation number (100, 250, 500 and 750 permutations).

It immediately becomes clear that the greater the number of permutations, the longer it will take to generate the explanation.

There is a direct proportional relation between the M1 metric and the number of permutations; as the number of permutations reduces, so does the M1 metric's value.

In contrast, the M2 metric exhibits a positive correlation with fewer permutations. This implies that a reduced proportion of green pixels outside the bounding box produced by the YOLOv7 prediction will result in a more precise explanation of the defect's location, although at the expense of increased execution time.

The M3 metric showed extremely low values; its value increased with the number of created permutations, suggesting that the greater the number of permutations, the greater the percentage of red pixels that can occur inside the YOLOv7 bounding box in the explanation.

Finally, the M4 and M2 metrics have the same correlation with the total number of permutations. The explanation will be more solid and assertive if the M4 metric value is higher since fewer red pixels will show up in it.

Number of Permutations	Global Mean M1	Global Standard Deviation M1	Global Mean M2	Global Standard Deviation M2	Global Mean M3	Global Standard Deviation M3	Global Mean M4	Global Standard Deviation M4	Execution Time (mins)
100	22.06	7.68	76.06	10.42	0.26	0.89	27.34	7.52	5.44
250	27.04	6.21	70.84	8.5	0.2	0.58	19.17	12.04	8.49
500	28.72	5.86	69.0	8.49	2.84	7.96	13.69	12.53	15.33
750	29.44	5.13	68.07	7.64	2.74	7.95	4.53	6.95	21.61
1000	30.77	5.7	66.68	8.4	3.2	7.74	3.38	5.48	24.27

Table 5.1: Global mean of metrics M1, M2, M3 and M4 on every permutations reduced number with superpixels that have a weight greater than 0.05

Table 5.2 can be analyzed to determine the percentage differences between each number of permutations shown in Table 5.1 and the overall base value of the 1000 permutations row. Negative values indicate that a particular number of permutations had a metric value that was higher than the 1000 permutations' metric value, and positive values the opposite.

Number of Permutations	Global Mean M1	Global Standard Deviation M1	Global Mean M2	Global Standard Deviation M2	Global Mean M3	Global Standard Deviation M3	Global Mean M4	Global Standard Deviation M4	Reduction Time (%)
100	28.31%	34.74%	14.07%	24.05%	-91.88%	-88.50%	710.65%	37.23%	-77.59
250	12.12%	8.95%	6.24%	1.19%	-93.75%	-92.51%	467.16%	119.71%	-65.02
500	6.66%	2.81%	3.48%	1.07%	-11.25%	2.84%	305.03%	128.65%	-36.84
750	4.32%	-10.00%	2.08%	-9.05%	-14.37%	2.71%	34.02%	26.82%	-10.96
1000	30.77	5.7	66.68	8.4	3.2	7.74	3.38	5.48	0.0

Table 5.2: Percentage difference of metrics M1, M2, M3 and M4 on every permutation number with superpixels that have a weight greater than 0.05

Chapter 6

Proposal of light optimization on LIME explanations creation speed

6.1 Problem Statement and Methodology

Since LIME is a widely used method today, particularly in the context of image classification explanation, it can take an average of 6 minutes to generate an explanation for an image, as shown in Table 2.1 in Section 2.4.2, due to the fact that it often interacts with the model being explained. This means that its applicability in a real context is limited, particularly for real-time applications such as defect or anomaly detection.

The primary goal of this chapter is to suggest a way to optimize the LIME approach to make it faster and more useful in practical settings.

Figure 6.1 illustrates the methodology that was followed. Firstly, the regression ML algorithm that was being used to generate the relevance of the super-pixels in the explanations was identified, which in this case was the Ridge algorithm. Next, in order to better understand the algorithm's performance, LIME was run on 10 different images, with 50, 100, 500 and 1000 permutations of each image, and the execution times were noted.

Three criteria were used to choose the regression algorithms that would be compared to the Ridge method. Because of their simplicity, the first three algorithms (Linear Regression, Lasso Regression, and Elastic Net) from the scikit-learn tool were used. The next step was using two popular Auto ML tools, AutoSklearn from scikit-learn [143] and AutoML from H2O [144–146], to find ML models.

For each, four models with comparatively good performance were identified. Five-fold cross-validation [147] was used to evaluate all of the examined methods, including the original Ridge algorithm. Section 6.2 discusses the results of this approach, which involved selecting an algorithm to replace Ridge based on analysis of predictive performance (RMSE and MAE) and computational performance (training time).

6.2 Results

This section only addresses the steps of training the regression model and features selection, as explained in Section 6.1. The goal is to identify a regression algorithm that

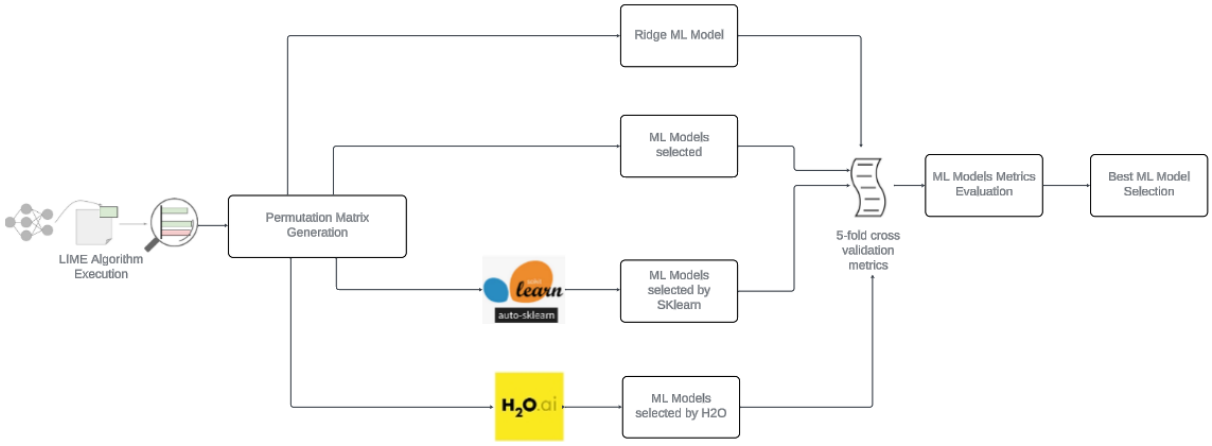


Figure 6.1: Methodology fluxogram.

would outperform the current algorithm, Ridge, in terms of training time and results (with better MAE and RMSE).

The outcomes of this approach are summarized in Table 6.1, where the tested algorithms are compared with the original Ridge algorithm and grouped together. The three manually chosen algorithms—Linear Regression, Lasso Regression and Elastic Net—are in the first group. The algorithms that AutoSklearn selected are in the second group. The algorithms that were chosen using H2O are finally included in the third group.

To produce these results, 10 permutation matrices were used (with 1000 permutations each), each for a different image classification problem. Thus, all the results presented are the average value of the algorithm over 10 different instances. Specifically, the average training time for each algorithm was calculated, as well as the average and standard deviation of the MAE and RMSE metrics.

Model	Mean train duration (in seconds)	Mean MAE	Mean Standard Deviation MAE	Mean RMSE	Mean Standard Deviation RMSE
Ridge	0.0151	0.220115	0.008463	0.276704	0.012521
Linear Regression	0.0247	0.220075	0.008486	0.276763	0.012555
Lasso Regression	0.0217	0.322233	0.008038	0.362009	0.012348
Elastic Net	0.0207	0.322233	0.008038	0.362009	0.012348
Ada Boost Regressor	9.3639	**0.142195**	0.013924	0.269508	0.019667
Extra Tree Regressor	5.8252	0.193794	**0.007768**	0.253452	**0.011877**
Gaussian Process Regressor	288.2741	0.174664	0.008200	**0.247288**	0.012360
Hist Gradient Boosting Regressor	1.2101	0.203012	0.008296	0.260145	0.012309
Gradient Boosting Machine	0.0940	**0.182037**	0.065697	**0.244895**	0.057050
Deep Learning	4.8299	0.197212	0.067869	0.259106	0.058557
Generalized Linear Model	**0.0137**	0.220722	0.068445	0.276757	0.057748
Distributed Random Forest	0.2274	0.2083	0.073352	0.261336	0.061389

Table 6.1: Comparison of the metrics MAE, RMSE and training time between Ridge and the selected algorithms: 1st group scikit-learn, 2nd group AutoSklearn, 3rd group AutoML. A value in bold indicates the algorithm's superior performance compared to Ridge.

Considering the first group of algorithms compared to Ridge, it can be seen that Ridge is the algorithm with the shortest training time and also has the lowest MAE and RMSE values compared to the other regression algorithms, making it the best algorithm in this small sample of regression algorithms.

The Ada Boost Regressor, Extra Tree Regressor, Gaussian Process Regressor, and Hist Gradient Boosting Regressor were the top algorithms identified by AutoSklearn. It is evident by comparing their results to Ridge's that all of the algorithms outperformed Ridge in terms of RMSE and MAE measures, which is encouraging. Nonetheless, their average training times were comparatively greater than Ridge's. The steps "Fit the regression model" and "Select the features" shown in Table 2.1 in Section 2.4.2 are included in this, as previously mentioned. As it happens, the new average times displayed are far longer than the typical Ridge time.

Therefore, even though the Gaussian Process Regressor algorithm produces more accurate predictions than Ridge, it was excluded because to its average duration of roughly 4 minutes, which barely makes a difference to the average training time. In a similar vein, we eliminated the Hist Gradient Boosting Regressor method, which took only a few seconds to complete but produced the worst results among the new regression algorithms in terms of the MAE metric and the third worst overall in terms of the RMSE metric. The Ada Boost Regressor and Extra Tree Regressor algorithms are the last ones. These outperformed Ridge in the MAE and RMSE metrics, although they took 900x and 500x longer, respectively.In this sense, these algorithms are not considered to be in line with the proposal of this chapter and have also been discarded.

The algorithms discovered by H2O are in the final group. As observed, despite a little longer training period, the Gradient Boosting Machine outperformed Ridge in this group in terms of the MAE and RMSE parameters. By closely examining the mean training duration, one algorithm was identified on this set that executed faster than Ridge: the Generalized Linear Model. In terms of average MAE and average RMSE, the algorithm is not as good as Ridge, but both values are rather close.

Because it offers a lower execution time and comparable predictive performance to Ridge, the Generalized Linear Model method was chosen as Ridge's replacement based on the results.

Chapter 7

Conclusion

7.1 Conclusions and Future Work

The work described in Chapter 3 had as main goal to propose and validate a set of metrics derived from the field of XAI, to be used as a complement in estimating the quality of each output of a CV model. This specifically addresses the problem of monitoring model quality once it is in production, in the absence of up-to-date labeled data.

The metrics validated so far resulted in the identification of 2 strong correlations, and 2 weaker ones, a finding which supports the longer-term goal of being able to estimate the quality of each prediction of a CV model automatically, based on its explanation. In essence, we aim to emulate the analysis that a human does when looking at an explanation, and trying to determine whether the model can be trusted or not.

Work depicted in Chapter 4 covers two aspects. On the one hand, we developed objective metrics for evaluating the quality of explanations. This will allow to assess two different aspects separately: the quality of the model and the quality of the generated explanations. On the other hand, we included permuted images together with the original training data when training models, so that they naturally learn to ignore the permuted superpixels.

The methodology proposed in that chapter is used to decide which background to use in each problem prior to the training, minimizing the necessary training data. The approach described is thus one step forward towards the inclusion of more human-centric and transparent approaches in industrial AI applications, in which human oversight for end-users and developers is seen as fundamental.

A strategy to reduce by 50% the number of permutations required for LIME to generate a good explanation was presented in Chapter 5. The permutation reduction effects were assessed using metrics specified in Chapter 3. By using this approach, we can ensure that we obtain a more varied collection of permutations while also slightly speeding up the LIME process of producing permutations.

In respect to Chapter 3 as a future work, we aim to automate and validate this process in a labelled dataset. To do so, we will implement a feature engineering and selection process in an attempt to find meaningful features, possibly through linear combinations of them, to train a model that is able to distinguish between good and bad predictions.

This model will then be used as a layer of validation, that will request human intervention when the output of the CV model is not trusted by the explanation. This keeps the human-in-the-loop, but only when necessary. Moreover, the input of the user will be added to the labelled dataset, allowing for its incorporation in future versions of the model, ensuring that it is kept up-to-date with minimal human effort.

An additional future work will involve the development of a ML model that predicts the quality of the LIME explanation, thereby simplifying the evaluation process and providing more effective support for decision-making. This ML model will be based on the evaluation metrics described in chapter 3.

It was discovered an algorithm with predictive performance comparable to the Ridge method utilized by LIME, but requiring less training time, as described in Chapter 6. As one of the future works, Ridge will be replaced with the suggested Generalized Linear Model as one of the future works since, albeit being negligible, the difference will help LIME perform slightly better.

7.2 Scientific Publications

The following articles were published or submitted during the work on this project:

1. João Pereira, Filipe Oliveira and Davide Carneiro (2024) Addressing the Limitations of LIME for Explainable AI in Manufacturing: A Case Study in Textile Defect Detection. 2nd ESAIM (European Symposium on Artificial Intelligence in Manufacturing) 2024 Conference, Athens, Greece.

2. João Pereira, Filipe Oliveira, Miguel Guimarães, Davide Carneiro, Miguel Ribeiro and Gilberto Loureiro (2024) Using explanations to estimate the quality of computer vision models. 33rd IAMOT (International Association for Management of Technology) 2024 Conference, Porto, Portugal.

3. João Pereira, Filipe Oliveira, Davide Carneiro, Miguel Ribeiro and Gilberto Loureiro Making explainable AI work for the manufacturing domain. International Journal of Computer Vision (submitted).

Article no. 1 has been published for the 2nd edition of the ESAIM Conference [1], and its contents were outlined in Chapter 4. The specific use-case for this article's methodology is defect identification in the textile manufacturing industry. It addresses a limitation in LIME where some superpixels are confused by defects instead of being ignored by the YOLO Model. This article's main contribution is demonstrating that explainability techniques based on picture permutations might not be suitable for the manufacturing sector since the underlying model frequently misinterprets the black superpixels added to the images and detects them as defects.

The scope of the ESAIM'24 symposium is around recent developments of AI in manufacturing so as to develop an understanding of key concepts and technologies, as well as understanding the benefits and barriers when applying such technologies in industrial practice.

[1] https://aim-net.eu esaim2024/

The content of article no. 2 was covered in Chapter 3; it included a questionnaire with 24 participants' ratings on each explanation, along with a proposal of a set of metrics to assess the quality of the LIME-generated explanations. This article was also presented in the 33rd IAMOT Conference, as seen in Figure 7.1.

The IAMOT event took place in Portugal for the first time (hosted in Alfândega do Porto, in Porto), with close to 250 participants from 30 countries. The 33rd edition of the IAMOT [2] (International Association for Management of Technology) Conference set out to discuss the role of technology in implementing sustainable, people-centred practices to address social, environmental, and economic challenges. The event also strengthened international cooperation, ensured advances in technological knowledge and gave the country a global projection.

Figure 7.1: 33rd IAMOT 2024 Conference Presentation in Alfândega do Porto, Portugal.

[2] https://www.iamot2024.com/

Chapter 8

Bibliography

[1] J. vom Brocke, A. Hevner, and A. Maedche, "Introduction to design science research," in *Progress in IS*, pp. 1–13, Cham: Springer International Publishing, 2020.

[2] A. Mehta and R. Jain, "An analysis of fabric defect detection techniques for textile industry quality control," in *2023 World Conference on Communication & Computing (WCONF)*, IEEE, July 2023.

[3] V. S. Yosephine, T. Hanna, M. Setiawati, and A. Setiawan, "Machine learning for quality control in traditional textile manufacturing," *J. Rekayasa Sist. Ind.*, vol. 13, pp. 165–174, Apr. 2024.

[4] H. M. Ferreira, D. R. Carneiro, M. Â. Guimarães, and F. V. Oliveira, "Supervised and unsupervised techniques in textile quality inspections," *Procedia Comput. Sci.*, vol. 232, pp. 426–435, 2024.

[5] F. Oliveira, D. Carneiro, H. Ferreira, and M. Guimarães, "Fabric defect detection and localization," in *Lecture Notes in Mechanical Engineering*, Lecture notes in mechanical engineering, pp. 177–184, Cham: Springer Nature Switzerland, 2024.

[6] D. Wu and D.-W. Sun, "Colour measurements by computer vision for food quality control – a review," *Trends Food Sci. Technol.*, vol. 29, pp. 5–20, Jan. 2013.

[7] F. K. Konstantinidis, S. G. Mouroutsos, and A. Gasteratos, "The role of machine vision in industry 4.0: an automotive manufacturing perspective," in *2021 IEEE International Conference on Imaging Systems and Techniques (IST)*, IEEE, Aug. 2021.

[8] M. Javaid, A. Haleem, R. P. Singh, S. Rab, and R. Suman, "Exploring impact and features of machine vision for progressive industry 4.0 culture," *Sens. Int.*, vol. 3, no. 100132, p. 100132, 2022.

[9] D. L. Galata, L. A. Mészáros, N. Kállai-Szabó, E. Szabó, H. Pataki, G. Marosi, and Z. K. Nagy, "Applications of machine vision in pharmaceutical technology: A review," *Eur. J. Pharm. Sci.*, vol. 159, p. 105717, Apr. 2021.

[10] Y. D. V. Yasuda, F. A. M. Cappabianco, L. E. G. Martins, and J. A. B. Gripp, "Aircraft visual inspection: A systematic literature review," *Comput. Ind.*, vol. 141, p. 103695, Oct. 2022.

[11] J.-O. Kim, M. K. Traore, and C. Warfield, "The textile and apparel industry in developing countries," *Text. Prog.*, vol. 38, pp. 1–64, Jan. 2006.

[12] M. Haenlein and A. Kaplan, "A brief history of artificial intelligence: On the past, present, and future of artificial intelligence," *Calif. Manage. Rev.*, vol. 61, pp. 5–14, Aug. 2019.

[13] Y.-H. Lai, W.-N. Chen, T.-C. Hsu, C. Lin, Y. Tsao, and S. Wu, "Overall survival prediction of non-small cell lung cancer by integrating microarray and clinical data with deep learning," *Sci. Rep.*, vol. 10, p. 4679, Mar. 2020.

[14] I. Ahmed, G. Jeon, and F. Piccialli, "From artificial intelligence to explainable artificial intelligence in industry 4.0: A survey on what, how, and where," *IEEE Trans. Industr. Inform.*, vol. 18, pp. 5031–5042, Aug. 2022.

[15] M. N. Murty and V. S. Devi, *Introduction to pattern recognition and machine learning*, vol. 5. World Scientific, 2015.

[16] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, *Machine learning, neural and statistical classification*. Ellis Horwood, 1995.

[17] Y. Jin, H. Wang, and C. Sun, "Introduction to machine learning," in *Studies in Computational Intelligence*, Studies in computational intelligence, pp. 103–145, Cham: Springer International Publishing, 2021.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[19] Y. Kahraman and A. Durmuşoğlu, "Deep learning-based fabric defect detection: A review," *Text. Res. J.*, p. 004051752211307, Oct. 2022.

[20] I. N. Aizenberg, N. N. Aizenberg, and J. Vandewalle, "Multiple-valued threshold logic and multi-valued neurons," in *Multi-Valued and Universal Binary Neurons*, pp. 25–80, Boston, MA: Springer US, 2000.

[21] G. E. Hinton, "Learning multiple layers of representation," *Trends Cogn. Sci.*, vol. 11, pp. 428–434, Oct. 2007.

[22] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, pp. 84–90, May 2017.

[24] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, July 2006.

[25] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[26] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Comput. Sci. Rev.*, vol. 40, p. 100379, May 2021.

[27] H. Schwenk, "Continuous space translation models for phrase-based statistical machine translation," in *Proceedings of COLING 2012: Posters*, pp. 1071–1080, 2012.

[28] L. Dong, F. Wei, M. Zhou, and K. Xu, "Adaptive Multi-Compositionality for recursive neural models with applications to sentiment analysis," *Proc. Conf. AAAI Artif. Intell.*, vol. 28, June 2014.

[29] D. Tang, F. Wei, B. Qin, T. Liu, and M. Zhou, "Coooolll: A deep learning system for twitter sentiment classification," in *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pp. 208–212, 2014.

[30] A. L. Maas, P. Qi, Z. Xie, A. Y. Hannun, C. T. Lengerich, D. Jurafsky, and A. Y. Ng, "Building DNN acoustic models for large vocabulary speech recognition," *Comput. Speech Lang.*, vol. 41, pp. 195–213, Jan. 2017.

[31] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, "Medical image classification with convolutional neural network," in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, IEEE, Dec. 2014.

[32] F. Li, L. Tran, K.-H. Thung, S. Ji, D. Shen, and J. Li, "A robust deep model for improved classification of AD/MCI patients," *IEEE J. Biomed. Health Inform.*, vol. 19, pp. 1610–1616, Sept. 2015.

[33] K. Sirinukunwattana, S. E. Ahmed Raza, Yee-Wah Tsang, D. R. J. Snead, I. A. Cree, and N. M. Rajpoot, "Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images," *IEEE Trans. Med. Imaging*, vol. 35, pp. 1196–1206, May 2016.

[34] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, May 2018.

[35] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, pp. 106–154, Jan. 1962.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[37] L. Zhou, L. Zhang, and N. Konz, "Computer vision techniques in manufacturing," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 53, pp. 105–117, Jan. 2023.

[38] Z. Liu, H. Ukida, K. Niel, and P. Ramuhalli, "Industrial inspection with open eyes: Advance with machine vision technology," in *Integrated Imaging and Vision Techniques for Industrial Inspection*, Advances in computer vision and pattern recognition, pp. 1–37, London: Springer London, 2015.

[39] A. Soini, "Machine vision technology take-up in industrial applications," in *ISPA 2001. Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis. In conjunction with 23rd International Conference on Information Technology Interfaces (IEEE Cat. No.01EX480)*, Univ. Zagreb, 2002.

[40] J. A. Noble, "From inspection to process understanding and monitoring: a view on computer vision in manufacturing," *Image Vis. Comput.*, vol. 13, pp. 197–214, Apr. 1995.

[41] H. Golnabi and A. Asadpour, "Design and application of industrial machine vision systems," *Robot. Comput. Integr. Manuf.*, vol. 23, pp. 630–637, Dec. 2007.

[42] B. Horn, *Robot vision*. MIT press, 1986.

[43] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 448–461, Mar. 2010.

[44] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Comput. Vis.*, vol. 74, pp. 59–73, Aug. 2007.

[45] S. Jain, "Edge detection using fuzzy gradient information," Aug. 2020. US Patent 10,740,903.

[46] G. Shrivakshan and C. Chandrasekar, "A comparison of various edge detection techniques used in image processing," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 269, 2012.

[47] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," 2016.

[48] C. Jeong, H. S. Yang, and K. Moon, "A novel approach for detecting the horizon using a convolutional neural network and multi-scale edge detection," *Multidimens. Syst. Signal Process.*, vol. 30, pp. 1187–1204, July 2019.

[49] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE Inst. Electr. Electron. Eng.*, vol. 86, no. 11, pp. 2278–2324, 1998.

[50] K. Simonyan, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[51] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[52] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[53] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," *Proc. Conf. AAAI Artif. Intell.*, vol. 31, Feb. 2017.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[55] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[56] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[57] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[58] J. Ponce, *Toward category-level object recognition*, vol. 4170. Springer Science & Business Media, 2006.

[59] J. Sivic and A. Zisserman, "Efficient visual search of videos cast as text retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, pp. 591–606, Apr. 2009.

[60] K. Lai, L. Bo, X. Ren, and D. Fox, "Sparse distance learning for object recognition combining RGB and depth information," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, May 2011.

[61] I. Adjabi, A. Ouahabi, A. Benzaoui, and A. Taleb-Ahmed, "Past, present, and future of face recognition: A review," *Electronics (Basel)*, vol. 9, p. 1188, July 2020.

[62] G. Guo, S. Z. Li, and K. Chan, "Face recognition by support vector machines," in *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, IEEE Comput. Soc, 2002.

[63] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, pp. 137–154, May 2004.

[64] Y. Sun, D. Liang, X. Wang, and X. Tang, "DeepID3: Face recognition with very deep neural networks," 2015.

[65] L. Wu, S. C. H. Hoi, and N. Yu, "Semantics-preserving bag-of-words models and applications," *IEEE Trans. Image Process.*, vol. 19, pp. 1908–1920, July 2010.

[66] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 1627–1645, Sept. 2010.

[67] Z. Wu, C. Shen, and A. van den Hengel, "Wider or deeper: Revisiting the ResNet model for visual recognition," *Pattern Recognit.*, vol. 90, pp. 119–133, June 2019.

[68] L. Chen, N. Ma, P. Wang, J. Li, P. Wang, G. Pang, and X. Shi, "Survey of pedestrian action recognition techniques for autonomous driving," *Tsinghua Sci. Technol.*, vol. 25, pp. 458–470, Aug. 2020.

[69] X. Wang, J. Ellul, and G. Azzopardi, "Elderly fall detection systems: A literature survey," *Front. Robot. AI*, vol. 7, p. 71, June 2020.

[70] C.-L. Yang, S.-C. Hsu, Y.-W. Hsu, and Y.-C. Kang, "Human action recognition on exceptional movement of worker operation," in *Advances in Manufacturing, Production Management and Process Control*, Lecture notes in networks and systems, pp. 376–383, Cham: Springer International Publishing, 2021.

[71] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

[72] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 603–619, May 2002.

[73] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-Based image segmentation," *Int. J. Comput. Vis.*, vol. 59, pp. 167–181, Sept. 2004.

[74] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient N-D image segmentation," *Int. J. Comput. Vis.*, vol. 70, pp. 109–131, Nov. 2006.

[75] D. Cremers, M. Rousson, and R. Deriche, "A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape," *Int. J. Comput. Vis.*, vol. 72, pp. 195–215, Apr. 2007.

[76] K. He and G. Gkioxari, "P. doll ar, and r. girshick,"mask r-cnn,"," in *Proc. IEEE Int. Conf. Comput. Vis*, pp. 2980–2988, 2017.

[77] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3146–3154, 2019.

[78] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Lecture Notes in Computer Science*, Lecture notes in computer science, pp. 234–241, Cham: Springer International Publishing, 2015.

[79] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention U-Net: Learning where to look for the pancreas," Apr. 2018.

[80] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: A nested U-Net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, Lecture notes in computer science, pp. 3–11, Cham: Springer International Publishing, 2018.

[81] D. Jha, P. H. Smedsrud, M. A. Riegler, D. Johansen, T. D. Lange, P. Halvorsen, and H. D. Johansen, "ResUNet++: An advanced architecture for medical image segmentation," in *2019 IEEE International Symposium on Multimedia (ISM)*, IEEE, Dec. 2019.

[82] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, "TransUNet: Transformers make strong encoders for medical image segmentation," 2021.

[83] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D u-net: Learning dense volumetric segmentation from sparse annotation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, Lecture notes in computer science, pp. 424–432, Cham: Springer International Publishing, 2016.

[84] V. Wong, M. Ferguson, K. Law, Y.-T. T. Lee, and P. Witherell, "Segmentation of additive manufacturing defects using u-net," *J. Comput. Inf. Sci. Eng.*, pp. 1–44, Nov. 2021.

[85] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[86] N. Zubić and P. Liò, "An effective loss function for generating 3D models from single 2D image without rendering," in *IFIP Advances in Information and Communication Technology*, IFIP advances in information and communication technology, pp. 309–322, Cham: Springer International Publishing, 2021.

[87] G. Xu and Z. Zhang, *Epipolar geometry in stereo, motion and object recognition: a unified approach*, vol. 6. Springer Science & Business Media, 2013.

[88] J. Maciel and J. P. Costeira, "A global solution to sparse correspondence problems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 187–199, Feb. 2003.

[89] C. Liu, J. Yuen, and A. Torralba, "SIFT flow: dense correspondence across scenes and its applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, pp. 978–994, May 2011.

[90] B. K. Horn, "Shape from shading: A method for obtaining the shape of a smooth opaque object from one view," 1970.

[91] J. Aloimonos, "Shape from texture," *Biol. Cybern.*, vol. 58, no. 5, pp. 345–360, 1988.

[92] S. K. Nayar and Y. Nakagawa, "Shape from focus," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 8, pp. 824–831, 1994.

[93] M. Hebert, "Active and passive range sensing for robotics," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, IEEE, 2002.

[94] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys, "Interactive 3D architectural modeling from unordered photo collections," *ACM Trans. Graph.*, vol. 27, pp. 1–10, Dec. 2008.

[95] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 216–224, 2018.

[96] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4460–4470, 2019.

[97] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, June 2016.

[98] S. Ginosar, D. Haas, T. Brown, and J. Malik, "Detecting people in cubist art," *AI Matters*, vol. 1, pp. 16–18, Mar. 2015.

[99] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, pp. 98–136, Jan. 2015.

[100] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[101] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013.

[102] M. Hussain, "YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection," *Machines*, vol. 11, p. 677, June 2023.

[103] M. O. Lawal, "Tomato detection based on modified YOLOv3 framework," *Sci. Rep.*, vol. 11, p. 1447, Jan. 2021.

[104] C. Li, J. Li, Y. Li, L. He, X. Fu, and J. Chen, "Fabric defect detection in textile manufacturing: A survey of the state of the art," *Secur. Commun. Netw.*, vol. 2021, pp. 1–13, May 2021.

[105] A. Rasheed, B. Zafar, A. Rasheed, N. Ali, M. Sajid, S. H. Dar, U. Habib, T. Shehryar, and M. T. Mahmood, "Fabric defect detection using computer vision techniques: A comprehensive review," *Math. Probl. Eng.*, vol. 2020, pp. 1–24, Nov. 2020.

[106] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Dropblock: A regularization method for convolutional networks," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[107] J. Solawetz, "What is yolov7? a complete guide." `https://blog.roboflow.com/yolov7-breakdown/`, January 4 2024. Roboflow Blog. Accessed on 2024-9-05.

[108] D. Minh, H. X. Wang, Y. F. Li, and T. N. Nguyen, "Explainable artificial intelligence: a comprehensive review," vol. 55, pp. 3503–3568, Springer Science and Business Media LLC, June 2022.

[109] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017.

[110] A. Borges, M. Carvalho, M. Maia, M. Guimarães, and D. Carneiro, "Predicting and explaining absenteeism risk in hospital patients before and during COVID-19," *Socioecon. Plann. Sci.*, vol. 87, p. 101549, June 2023.

[111] L. Rosa, M. Guimarães, D. Carneiro, F. Silva, and C. Analide, "Explainable decision tree on smart human mobility," in *Ambient Intelligence and Smart Environments*, IOS Press, June 2022.

[112] J. Schrouff, S. Baur, S. Hou, D. Mincu, E. Loreaux, R. Blanes, J. Wexler, A. Karthikesalingam, and B. Kim, "Best of both worlds: local and global explanations with human-understandable concepts," 2021.

[113] S. Ali, T. Abuhmed, S. El-Sappagh, K. Muhammad, J. M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. Del Ser, N. Díaz-Rodríguez, and F. Herrera, "Explainable artificial intelligence (XAI): What we know and what is left to attain trustworthy artificial intelligence," *Inf. Fusion*, vol. 99, p. 101805, Nov. 2023.

[114] K. Gade, S. C. Geyik, K. Kenthapadi, V. Mithal, and A. Taly, "Explainable AI in industry," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, (New York, NY, USA), ACM, Jan. 2020.

[115] W. Saeed and C. Omlin, "Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities," *Knowl. Based Syst.*, vol. 263, p. 110273, Mar. 2023.

[116] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision making and a 'right to explanation'," *AI Mag.*, vol. 38, pp. 50–57, Sept. 2017.

[117] W. Samek and K.-R. Müller, "Towards explainable artificial intelligence," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, Lecture notes in computer science, pp. 5–22, Cham: Springer International Publishing, 2019.

[118] S. N. Payrovnaziri, Z. Chen, P. Rengifo-Moreno, T. Miller, J. Bian, J. H. Chen, X. Liu, and Z. He, "Explainable artificial intelligence models using real-world electronic health record data: a systematic scoping review," *J. Am. Med. Inform. Assoc.*, vol. 27, pp. 1173–1185, July 2020.

[119] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, pp. 1–42, Sept. 2019.

[120] C. Molnar, *Interpretable Machine Learning*. 2019.

[121] L. Veiber, K. Allix, Y. Arslan, T. F. Bissyandé, and J. Klein, "Challenges towards {Production-Ready} explainable machine learning," in *2020 USENIX Conference on Operational Machine Learning (OpML 20)*, 2020.

[122] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138–52160, 2018.

[123] R. Confalonieri, L. Coba, B. Wagner, and T. R. Besold, "A historical perspective of explainable artificial intelligence," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 11, Jan. 2021.

[124] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," Aug. 2017.

[125] L. Arras, F. Horn, G. Montavon, K.-R. Müller, and W. Samek, ""what is relevant in a text document?": An interpretable machine learning approach," *PLoS One*, vol. 12, p. e0181142, Aug. 2017.

[126] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," Dec. 2013.

[127] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, June 2015.

[128] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," *J. Artif. Intell. Res.*, vol. 70, pp. 245–317, Jan. 2021.

[129] Z. C. Lipton, "The mythos of model interpretability," *ACM Queue*, vol. 16, pp. 31–57, June 2018.

[130] P. Messina, P. Pino, D. Parra, A. Soto, C. Besa, S. Uribe, M. Andía, C. Tejos, C. Prieto, and D. Capurro, "A survey on deep learning and explainability for automatic

report generation from medical images," *ACM Comput. Surv.*, vol. 54, pp. 1–40, Jan. 2022.

[131] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (New York, NY, USA), ACM, Aug. 2016.

[132] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *Lecture Notes in Computer Science*, Lecture notes in computer science, pp. 705–718, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[133] R. Gao, S. Cui, H. Xu, Q. Kong, Z. Su, and J. Li, "A method for obtaining maize phenotypic parameters based on improved QuickShift algorithm," *Comput. Electron. Agric.*, vol. 214, p. 108341, Nov. 2023.

[134] T. A. A. Abdullah, M. S. M. Zahid, W. Ali, and S. U. Hassan, "B-LIME: An improvement of LIME for interpretable deep learning classification of cardiac arrhythmia from ECG signals," *Processes (Basel)*, vol. 11, p. 595, Feb. 2023.

[135] G. Visani, E. Bagli, and F. Chesani, "OptiLIME: Optimized LIME explanations for diagnostic computer algorithms," 2020.

[136] A. L. Suárez-Cetrulo, D. Quintana, and A. Cervantes, "A survey on machine learning for recurring concept drifting data streams," *Expert Systems with Applications*, vol. 213, p. 118934, 2023.

[137] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[138] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, "The mvtec anomaly detection dataset: a comprehensive real-world dataset for unsupervised anomaly detection," *International Journal of Computer Vision*, vol. 129, no. 4, pp. 1038–1059, 2021.

[139] L. Tan, C. Huang, and X. Yao, "A concept-based local interpretable model-agnostic explanation approach for deep neural networks in image classification," in *International Conference on Intelligent Information Processing*, pp. 119–133, Springer, 2024.

[140] R. Padilla, S. L. Netto, and E. A. Da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 international conference on systems, signals and image processing (IWSSIP)*, pp. 237–242, IEEE, 2020.

[141] S.-S. Yu, S.-W. Chu, C.-M. Wang, Y.-K. Chan, and T.-C. Chang, "Two improved k-means algorithms," *Appl. Soft Comput.*, vol. 68, pp. 747–755, July 2018.

[142] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 2, pp. 433–459, July 2010.

[143] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[144] M. Schmitt, "Automated machine learning: AI-driven decision making in business analytics," vol. 18, p. 200188, Elsevier BV, May 2023.

[145] P. Gijsbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, and J. Vanschoren, "An open source AutoML benchmark," arXiv, 2019.

[146] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss, and R. Farivar, "Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, Nov. 2019.

[147] X. Zhang and C.-A. Liu, "Model averaging prediction by k-fold cross-validation," Elsevier BV, May 2022.