FEDERAL UNIVERSITY OF PARANÁ

MICHEL DE OLIVEIRA HILGEMBERG

DEEP GENERATIVE MODELS FOR SYNTHETIC IMAGE AUGMENTATION IN COMPUTER VISION-BASED QUALITY INSPECTION

CURITIBA

2023

MICHEL DE OLIVEIRA HILGEMBERG

DEEP GENERATIVE MODELS FOR SYNTHETIC IMAGE AUGMENTATION IN COMPUTER VISION-BASED QUALITY INSPECTION

Bachelor thesis conferred to the Faculty of Mechanical Engineering, from the Technology Sector of the Federal University of Paraná as a partial requirement to obtaining a bachelor's degree in Mechanical Engineering.

Supervisor: Prof. Dr. Fernando Deschamps
Co-supervisor: M.Sc Maximilian Motz

CURITIBA

2023

Approval

MICHEL DE OLIVEIRA HILGEMBERG

DEEP GENERATIVE MODELS FOR SYNTHETIC IMAGE AUGMENTATION IN COMPUTER VISION-BASED QUALITY INSPECTION

Bachelor thesis conferred to the Faculty of Mechanical Engineering, from the Technology Sector of the Federal University of Paraná as a partial requirement to obtaining a bachelor's degree in Mechanical Engineering

---

Prof. Dr. Fernando Deschamps
Supervisor – DEMEC, UFPR

Maximilian Motz
Co-supervisor - Fraunhofer IPT

Pablo Deivid Valle
Internal Guest - DEMEC, UFPR

Renato Fernandes Grottoli
External Guest - Robert Bosch Ltda.

Curitiba, 28th June 2023.

Ao meu pai, Edenilson, exemplo de coragem, trabalho e felicidade.

À minha mãe, Daltiva, pela confiança e incentivos em cada uma das minhas escolhas.

ACKNOWLEDGEMENTS

I express my deepest gratitude to:

Maximilian Motz M.Sc for all the amazing mentorship and support throughout the trials and triumphs during the study's execution. Vielen Dank!

Prof. Dr. Fernando Deschamps for providing valuable insights and feedback over various stages of the study.

Fraunhofer IPT and the 300 department for the resources, opportunities, and incredible projects.

My friends from IPT's fourth floor - Fabian, Fernando, Gustavo, Henrique, Laura, Luiza, Pedro, and Victor - for a great time together and help on various occasions.

The friends I met in Germany for the enjoyable moments and encouragement during my stay abroad.

Em português, agradeço aos meus pais, Edenilson e Daltiva, pelo suporte e apoio em todos os meus objetivos e sonhos.

Aos amigos que a UFPR colocou em minha vida - Lucas, Rafael, Yudi e Yuri - pelo companheirismo e momentos de diversão, mesmo durante rotinas difíceis.

À Sarah, cuja parceria e carinho são indescritíveis.

A Deus.

"What I cannot create, I do not understand"

- Richard P. Feynman

ABSTRACT

The global movement towards *Industry 4.0* is empowering manufacturing to get valuable business insights from production data, especially applying Machine Learning (ML) and Deep Learning (DL) techniques from a data-driven perspective. In this scenario, computer vision quality inspection processes are among the top five uses of DL in industrial manufacturing. These procedures are essential to ensure the final quality of the product and the use of DL offers the potential to increase efficiency and reliability. However, compared with traditional ML, DL techniques require data of better quality and more quantity, representing an often challenge for manufacturing operations where the acquisition of new data is costly. Moreover, conventional augmentation approaches does not consider the true data distribution and usually rely on a process expert to be properly performed and avoid non-representative transformations. Considering this context, the use of deep generative models to create synthetic image samples is an innovative potential solution for small data problems. This technique has been proven positive in other domains, requiring further development in manufacturing applications. In this study, four different deep generative models were applied to perform synthetic image augmentation in a complex manufacturing use case with limited data. Subsequently, the models were compared in terms of image quality metrics and performance in a DL-model-based visual quality inspection task. The comparisons covered a series of possible combinations between original, conventional augmented, and synthetic augmented training data. According to the outcomes, the Deep Convolutional Generative Adversarial Networks (DCGAN) was the best-performing deep generative model in image quality assessment and classification performance. A pre-trained model on DCGAN's synthetic augmentation achieved the best macro F1-Score in the entire analysis. Moreover, when balancing possible information compressions resulting from image resizing in both augmentation process, DCGAN's synthetic data combined with conventional techniques outperformed the conventional augmentation approach by a significant margin. Therefore, among the deep generative models evaluated in this study, the DCGAN architecture demonstrated superior potential for synthetic image augmentation in the context of manufacturing scenarios with limited data.

Key-words: Deep generative models. Synthetic image augmentation. Manufacturing. Visual quality inspection.

RESUMO

O movimento global em direção à *Indústria 4.0* está fazendo com que a manufatura consiga retirar ideias valiosas dos dados coletados em produção, especialmente aplicando técnicas de Aprendizado de Máquina (ML) e Aprendizado Profundo (DL) em uma perspectiva direcionada aos dados. Nesse cenário, processos de inspeção de qualidade com visão computacional figuram entre os cinco usos mais comuns de DL na manufatura. Esses procedimentos são essenciais para garantir a qualidade final do produto e o uso de DL oferece potencial para aumento de eficiência e confiabilidade. Entretanto, comparado com o ML tradicional, as técnicas de DL necessitam de dados de melhor qualidade e em maior quantidade, representando um desafio comum para operações de manufatura onde a aquisição de novos dados possui alto custo associado. Além disso, abordagens de aumento convencional de dados não consideram a real distribuição dos mesmos e geralmente necessitam da validação de um profissional experiente no caso em questão para evitar transformações não representativas. Considerando esse contexto, o uso de modelos generativos profundos para produção de imagens sintéticas é uma inovadora e potencial solução para problemas associados à escassez de dados. Essa técnica têm se mostrado positiva em vários outros domínios, necessitando de um desenvolvimento futuro para aplicações na manufatura. Nesse estudo, quatro diferentes modelos generativos profundos foram aplicados para realizar aumento sintético de imagens em um caso complexo de manufatura com dados limitados. Posteriormente, os modelos foram comparados em relação a métricas de qualidade de imagens e desempenho em uma tarefa de inspeção visual de qualidade baseada em modelo de DL. As comparações abrangem uma série de possíveis combinações entre dados de treino originais, aumentados convencionalmente e aumentados sinteticamente. Segundo os resultados, a arquitetura de Redes Adversárias Generativas Convolucionais Profundas (DCGAN) foi o modelo generativo profundo com melhor desempenho em qualidade de imagem e performance de classificação. Um modelo pré-treinado nos dados sintéticos da arquitetura DCGAN obteve o melhor resultado do estudo em termos de macro F1-Score. Além disso, balanceando possíveis compressões de informação resultantes de redimensionamentos de imagens em ambos os processos, os dados sintéticos da DCGAN combinados com técnicas convencionais superaram a abordagem de aumento convencional por uma margem significativa. Portanto, dentre os modelos generativos profundos avaliados nesse estudo, a arquitetura DCGAN demonstrou um potencial superior para realizar aumento sintético de imagens dentro de um contexto de manufatura com dados limitados.

Palavras-chave: Modelos generativos profundos. Aumento sintéticao de imagens. Manufatura. Inspeção visual de qualidade.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AE	Autoencoder
AI	Artificial Intelligence
Adam	Adaptive Moment Estimation
BCE	Binary Cross-Entropy
CE	Cross-Entropy
CNN	Convolutional Neural Network
CV	Computer Vision
D	Discriminator
DCGAN	Deep Convolutional Generative Adversarial Network
DDPM	Denoising Diffusion Probabilistic Model
DL	Deep Learning
FID	Fréchet Inception Distance
G	Generator
GAN	Generative Adversarial Networks
GD	Gradient Descent
IS	Inception Score
KLD	Kullback–Leibler Divergence
LReLU	Leaky Rectified Linear Unit
ML	Machine Learning
MLP	Multilayer Perceptrons
MSE	Mean Squared Error

MV	Machine Vision
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
SGD	Stochastic Gradient Descent
SSIM	Structural Similarity Index Measure
VAE	Variational Autoencoder

TABLE OF CONTENTS

1 INTRODUCTION

The purpose of this chapter is to establish the motivation for the current study, expose the problem addressed in it, as well as identify the research gap and highlight the potential outcomes of this work. Consequently, the chapter will also define the general and specific objectives of this study.

1.1 MOTIVATION AND PROBLEM STATEMENT

As a global movement, manufacturing companies are increasingly adopting new digital solutions in their own facilities and processes. In the last decade, various manufacturing strategies have been developed to guide this enhancement, for example, the concept of *Industry 4.0* and *Smart manufacturing*. In a broader overview, the general goal of these frameworks is to convert the data acquired across the manufacturing process into business insights, leading to more robust decision-making procedures. (TAO et al., 2018).

One of the main driving forces behind each of these developments is the field of Artificial Intelligence (AI), especially the sub-areas of Machine Learning (ML) and Deep Learning (DL). With a great ability to manage and derive patterns from huge amounts of data, these fields are providing the tools and capabilities to make the manufacturing industry more intelligent (CHEN et al., 2023). These techniques are rapidly increasing and represent a huge economic impact on the production environment. In the 2022 McKinsey's Global Survey on AI, the consulting company reports that the number of AI adoption has more than doubled since the first survey in 2017 (MCKINSEY, 2022). Additionally, in a similar analysis, Deloitte reports that 93% of the surveyed manufacturing companies believe that AI is a driving force for growth and innovation (DELOITTE, 2020).

In an industrial context, ML has been effectively employed in a variety of manufacturing applications, including process optimization, monitoring, and control (WUEST et al., 2016). According to Google's report, quality inspection is among the top five AI areas deployed in day-to-day operations in industrial manufacturing (GOOGLE, 2021). In this scenario, in many manufacturing industries, the visual quality inspection routines

hold significant importance in ensuring the final excellence of the product (SU, 2016). However, this step is a common bottleneck, especially for requiring a significant amount of staff and dealing with the human factor prone to wrong qualitative assessments (KUJAWIŃSKA; VOGT, 2015). Therefore, using computer vision systems based on DL models offers excellent potential to increase efficiency and reliability by automating these procedures (REN et al., 2021).

Compared to traditional ML approaches, DL models could learn and generalize better even hidden patterns of data, being perfect choices for computer vision tasks, usually related to image classification or object detection. Unfortunately, one of the major disadvantages of DL is the necessity for a vast amount of data to achieve optimal performance (WHANG; LEE, 2020). Moreover, in the manufacturing environment, it is often difficult to collect, label, and manage data, with the quality and confidence that DL models demand (LI et al., 2018). Additionally, labeling and acquiring new data is usually a time-consuming and high-cost related activity, requiring a process expert or domain knowledge to be properly performed (PERES et al., 2020). Therefore, frequently only insufficient training data are available or critical classes, such as defect images, are strongly underrepresented.

To address this issue, the application of data augmentation techniques, modifying properties of data instances in order to upsample the dataset, is widely applied. Nevertheless, this technique should be carefully conducted, especially to ensure that it accurately represents the data domain and does not lead to poor model results (GOODFELLOW; BENGIO; COURVILLE, 2016). Similar to labeling new data instances, conventional augmentation often requires expert support.

Considering this circumstance, image data synthetization is a process that applies deep generative models to generate synthetic samples of images, and it has emerged as a solution for small data problems (SHORTEN; KHOSHGOFTAAR, 2019). The use of synthetic data instances has been proven positive in many areas, leading to more data for the model to learn from, together with filling some gaps of missing information in the dataset (SANTOS TANAKA; ARANHA, 2019; SANDFORT et al., 2019; CHATZIAGAPI et al., 2019). From a manufacturing perspective, however, this technique is still in its infancy, requiring further development, especially in terms of validating its efficacy and exploring different approaches in this deep learning domain.

1.2 OBJECTIVES

1.2.1 General objective

The primary goal of this study is to apply and compare different deep generative models for synthetic image augmentation in order to perform a DL-model-based visual quality inspection in a complex manufacturing use case with limited data. The task performance will be assessed through different scenarios, covering a series of possible combinations between original, conventional augmented, and synthetic augmented training data in a manufacturing context with data sparsity.

1.2.2 Specific objectives

The specific objectives of the present work are described in the following topics:

- Generate synthetic data samples by applying different deep generative model architectures.
- Implement a deep learning model focused on a manufacturing-related image classification use case that could be trained either in original or synthetic data.
- Evaluate the classification model across different training data scenarios in order to collect statistical metrics of performance.
- Conduct a comparative analysis to assess the feasibility of the different deep generative models for synthetic image augmentation.

1.3 THEORETICAL AND PRACTICAL IMPACTS OF THE STUDY

From a theoretical perspective, this study has the potential to impact manufacturing related DL applications by synthesizing new data instances with different deep generative models and then identifying the most promising approaches in this context. Furthermore, through a comparison between synthetic data scenarios for training a classification model, the study could derive statements about the performance impact of data synthetization in a manufacturing use case with limited data.

Secondly, from a practical point of view, this work could bring a positive impact on manufacturing visual quality inspection applications that are affected by data sparsity

challenges. In this context, the study could provide important insights into improving the performance of DL models behind these systems, enhancing reliability and leading to a valuable impact on production and business operations.

1.4 THESIS STRUCTURE

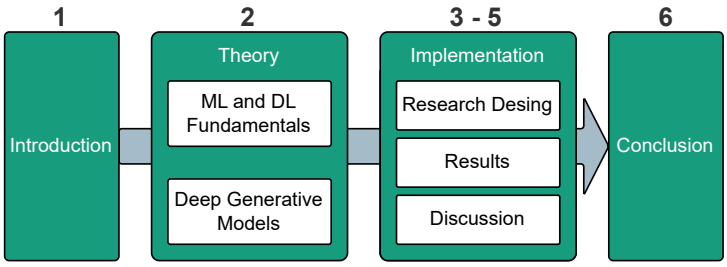
This work consists of six chapters. In chapter 1, the current status and challenges of ML and DL applications in the manufacturing domain are outlined. Subsequently, the motivation for using deep generative models as a solution for small data problems is derived. Based on the motivation and the problem statement, the study's primary and specific objectives are defined. The chapter then details the main theoretical and practical contributions of the study. Finally, this structure section concludes the first chapter of this work.

Chapter 2 addresses the theoretical background of this study. This chapter is composed of two main parts, firstly the fundamentals regarding computer vision, ML, and DL techniques are presented in an abbreviated manner. Secondly, the chapter aborts advanced topics in terms o deep generative models and data synthetization.

Chapter 3 covers the research design of this work. Therefore, according to the primary and specific objectives, an experimental setup is proposed. Additionally, the chapter details the selected use case and implementation of the deep learning models and metrics in this study.

Chapter 4 exhibits the results obtained for each experimental scenario. Afterward, chapter 5 provides a critical discussion and analysis of the outcomes. Finally, chapter 6 presents a conclusion about the findings to finally meet the study's objective, and then an outlook on further works and research in the field of image synthesis in manufacturing. FIGURE 1 provides an overview of the study structure.

FIGURE 1 – STUDY STRUCTURE



REFERENCE: The author (2023).

2 LITERATURE REVIEW

The present chapter aims to provide a literature review and state-of-the-art content for this study. In this context, it will cover the incoming development of the manufacturing domain, the fundamental knowledge of Machine and Deep Learning applications, and the use of deep generative models for image data synthesis.

2.1 INDUSTRIAL ARTIFICAL INTELLIGENCE

2.1.1 From industry 4.0 to industrial artificial intelligence

The term *Industry 4.0* refers to the fourth industrial revolution, where the Internet and digital technologies are leading to a new era of manufacturing (KANG et al., 2016). In this context, one of the most notable concepts is the idea of "Smart Manufacturing", which encompasses a variety of technologies and paradigms that can strategically innovate the current manufacturing industry through the convergence of humans, technology, and information. One of the key principles of this idea is the improvement of the major manufacturing factors, such as productivity, quality, delivery, and flexibility (KANG et al., 2016).

Considering this circumstance, the use of Artificial Intelligence (AI) in manufacturing domains is considered one of the main technologies to achieve disruptive changes in the traditional structure of industrial processes and businesses. In a broader definition, AI is a field of computer science, dedicated to data-related activities, and developing tasks associated with human intelligence (PERES et al., 2020)

Lee, Singh, and Azamfar (2019), defined the concept of *Industrial artificial intelligence* as a systematic discipline to enable engineers to manage the whole life cycle of AI solutions in the industrial domain, focusing on the steps of development, validation, deployment, and maintenance, with sustainable performance. Peres et al. (2020), established five main differences between standard and industrial AI:

1. **Infrastructures:** Industrial AI focuses on real-time processing capabilities, and emphasizes hardware and software reliability, and security.

2. **Data:** In the manufacturing domain, AI applications deal with a huge amount of data, usually associated with high velocity, and from several sources.

3. **Algorithms:** Industrial AI usually requires the integration of physical and digital knowledge, leading to a high-complexity model management scenario.

4. **Decision-making:** The industrial field has a low tolerance for errors, and efficiency in the decision-making process is a key feature for large-scale applications.

5. **Objectives:** Industrial AI addresses concrete values, such as reducing scrap, improving quality, enhancing operator performance, and accelerating ramp-up times.

The use of AI in the manufacturing environment, therefore, has been one of the driving forces of smart manufacturing development. In this particular scenario, closing the gap between new AI implementations and the industrial sector has a key role.

2.1.2 Computer vision in manufacturing

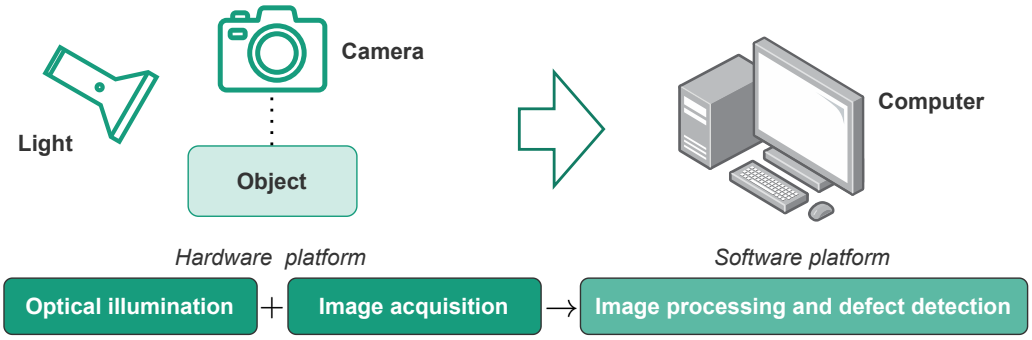
A recent research survey conducted by Google Cloud describes the scenario of AI adoption in manufacturing. The survey identified the top five areas of AI application as quality inspection, supply chain management, risk management, product and/or production line quality checks, and inventory management (GOOGLE, 2021).

Given this particular context, the quality inspection activity is directly related to Computer Vision (CV) and Machine Vision (MV) applications. These two concepts are closely related, but not exactly the same. CV is a crucial subfield of AI, which is associated with Deep Learning techniques focusing on image processing. On the contrary, MV systems, which are a subset of CV, are more specialized and focused on the industrial and manufacturing domain.

Ren et al. (2021) highlight that advanced industrial systems demand superior product performance and higher quality control during production, especially to ensure the levels of product esthetics, user comfort, and overall performance. Therefore, leading to a crucial role in defect detection inside the production chain. However, Beyerer, León, and Frese (2016) report several drawbacks associated with a human visual inspection, such as monotony, subjectivity, limited reproducibility, high cost, and insufficient speed. In this scenario, the primary objective of MV systems is to downgrade the need for

human operators in industrial inspection processes and ensure a more resilient and superior quality control (LIU et al., 2015). The FIGURE 2 represents a typical MV system for defect detection.

FIGURE 2 – INDUSTRIAL VISUAL QUALITY INSPECTION SYSTEM



REFERENCE: Adapted from Ren et al. (2021).

This application of the MV systems is closely related to a series of CV techniques, such as feature detection, recognition, segmentation, and 3-D modeling (ZHOU; ZHANG; KONZ, 2023). However, to successfully apply these techniques, as well as develop new and innovative areas for exploration, a solid understanding of the machine and Deep Learning domain is essential.

2.2 MACHINE LEARNING

2.2.1 Fundamentals of Machine Learning

Before diving into Machine Learning (ML) concepts, it would be helpful to first clarify the definition of the term. Samuel (1959), one of the pioneers in this field, defines Machine Learning as the study that gives computers the ability to learn without being explicitly programmed. In this context, an ML model aims to optimize a performance criterion using the ability to learn from training data and previous experiences. The criterion is related to the given task that the model should solve (ALPAYDIN, 2010).

Based on the task nature and the training procedure, ML can be classified into different methods. The main methods, along with their definitions, are outlined below:

- **Supervised learning:** This method is trained using labeled data, where the correct output is known for each input. A "supervisor" provides explicit instructions to guide the learning system in assigning labels to training examples. Typical

tasks are classification (prediction of discrete variables), and regression (prediction of quantitative variables) (CUNNINGHAM; CORD; DELANY, 2008; HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

• **Unsupervised learning:** On the contrary to supervised techniques, unsupervised learning is used when historical labeling is unavailable in the training data. The algorithm explores the data to find structure within it. Common tasks are clusterization algorithms, anomaly detection, and dimensionality reduction (GERON, 2019).

• **Reinforcement learning:** This method operates in a different manner when compared with the previous approaches. In this case, the learning system, which is called an "agent", has the ability to observe the environment and take action. Then, based on the action's nature, the agent receives rewards or penalties, learning the best policy, or strategy, to achieve the objective once is always seeking maximum rewards over time. This policy will outline the action of the agent in front of a specific scenario. This technique is frequently found in robotics and gaming applications (GERON, 2019).

2.2.2 ML loss functions

In general, ML algorithms can be viewed as the process of solving an optimization problem. Given this particular context, the *loss function* has a key role since the goal of an ML model will be to minimize the total loss incurred (BISHOP, 2006).

For regression tasks, the **Mean Squared Error (MSE)** is a widely used loss function. It measures the average (mean) squared difference between the predicted and actual values for each input. This loss function is defined as:

$$MSE(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 \tag{2.1}$$

Where θ represents the model's parameter vector, containing bias and the values of the weights, $\hat{y}_i$ is the predicted value, and y_i is the actual value.

Regarding classification tasks, the core idea is that the loss function will measure the difference between the predicted output and the actual output for a given input.

For multiclass classification tasks, the **Cross-Entropy (CE)** loss is one of the most commonly applied. It is defined as:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log \hat{p}_k^{(i)} \tag{2.2}$$

In this equation, $J(\theta)$ is the loss value, $y_k^{(i)}$ is the probability that the i^th instance belongs to the specific class k, and m is the number of data instances (GERON, 2019).

For binary classification problems, i.e. $k = 2$, the **Binary Cross-Entropy (BCE)** is a more suitable loss function. In mathematical terms, the BCE equation is equal to the logistic regression functions, and is defined by:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} [y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)] \tag{2.3}$$

Where, $J(\theta)$ is the loss value, y_i represents the actual label, $\hat{p}_i$ the predict probability of the positive class, and m is the number of data instances (GERON, 2019).

2.2.3 Evaluation metrics for ML models

Correctly evaluating an ML model is one of the crucial steps in implementing a successful project, where wrong evaluation techniques can lead to inaccurate results. This section aims to highlight the core evaluation metrics applied for the classifier implementation, more details regarding the deep generative model's metrics are at subsection 2.8.3.

In classification tasks, the confusion matrix is a table that shows the number of correct and incorrect predictions made by the model compared to the actual outcomes. This metric is a valuable visual tool that helps to easily identify where the model may be making errors and could be used for binary or multi-class classification tasks (GERON, 2019). Additionally, in a confusion matrix, each row represents the *actual/true class* and each column the *predicted class*, and the predicted data instance could be then fit in one of the four quarters:

- **True Positive (TP):** Positive samples correct classified as positive.
- **False Positive (FP):** Negative samples incorrect classified as positive

- **True Negative (TN):** Negative samples correct classified as negatives.
- **False Negative (FN):** Positive samples incorrect classified as negatives.

From the definitions above, a series of metrics could be derived. These metrics correspond to a different view of which feature represents a good model.

Accuracy

A common classification metric widely used for assessing ML models is accuracy. This metric corresponds to the sum of diagonal terms in the confusion matrix, divided by the total of terms (FERNÁNDEZ et al., 2018). Mathematically could be defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

While accuracy is commonly used and easy to understand, it has certain limitations when dealing with imbalanced datasets. One of the main drawbacks is that it is relatively easy to achieve high accuracy in highly imbalanced problems. For instance, a trivial classifier that always assigns the majority class to new instances can achieve 99% accuracy in a scenario where the majority class accounts for 99% of the dataset (FERNÁNDEZ et al., 2018).

Precision

The precision metric evaluates the fraction of correctly classified instances when analyzing the positive class. In simple terms, precision could be seen as the accuracy of the positive class (FERNÁNDEZ et al., 2018). Mathematically, the precision could be expressed as:

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

Recall

Recall it is a measure of how many of the actual positive instances in the dataset were correctly identified by the model. In other terms, recall indicates the percentage

of positive instances that were correctly predicted by the model (FERNÁNDEZ et al., 2018). Recall could be defined as:

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

To summarize, precision is focused on the accuracy of positive predictions, while recall is focused on the completeness of positive predictions. Geron (2019) explores the concept of *precision/recall trade-off*, where in ML applications increasing the model precision reduces the recall and vice-versa. Since both metrics are important for a good model, finding the sweet spot between precision and recall values will depend on the specific problem and its requirements.

F1-Score

In this context, F1-Score is a metric for supervised classification problems defined as the harmonic mean of precision and recall and is especially addressed to imbalanced datasets. Mathematically, it is represented as follows:

$$F1 - Score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.7}$$

Whereas the standard means treats all the values equally, the harmonic means assigns more weight to low values. In general, a good F1-Score will represent a model with similar precision and recall values (GERON, 2019).

2.3 DEEP LEARNING

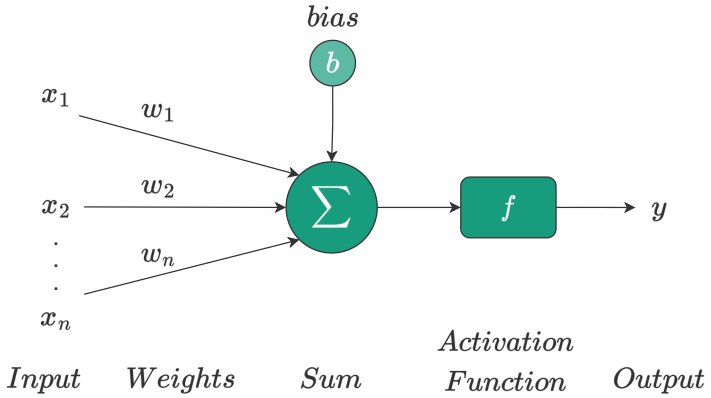
As Goodfellow, Bengio, and Courville (2016) described, while the ML algorithms are effective in addressing numerous significant challenges, they have not succeeded in solving fundamental problems in AI, including speech and object recognition. In this scenario, the development of DL was motivated to overcome these obstacles, especially those related to high-dimensional data.

2.3.1 Convolutional neural networks

The general idea of Deep Learning was inspired by human neuron mechanisms, this concept was born in the 1940s. McCulloch and Pitts (1943) were pioneers in this

field, they defined the basis for the perception, a technique to imitate biological neuron structure to artificial neurons. The first implementation of the perceptron concept in a practical example was conducted by Rosenblatt (1958). The general idea of the perceptron is that the artificial neuron takes some inputs, $x_1, x_2, ..., x_n$, each of these is multiplied by a specific weight, $w_1, w_2, ..., w_n$, sometimes also a bias b get into the equation, and sum together to represent the *logit* of the neuron, $z = \sum_{i=0}^{n} w_i \cdot x_i$. This value is then passed through a function f to generate the final output of the perceptron, $y = f(z)$. When a series of perceptrons are interconnected in layers, a Multilayer Perceptrons (MLP) architecture is defined, also known as a neural network. FIGURE 3 provides a visual representation of a single perceptron and its main components.

FIGURE 3 – PERCEPTRON SQUEMATIC

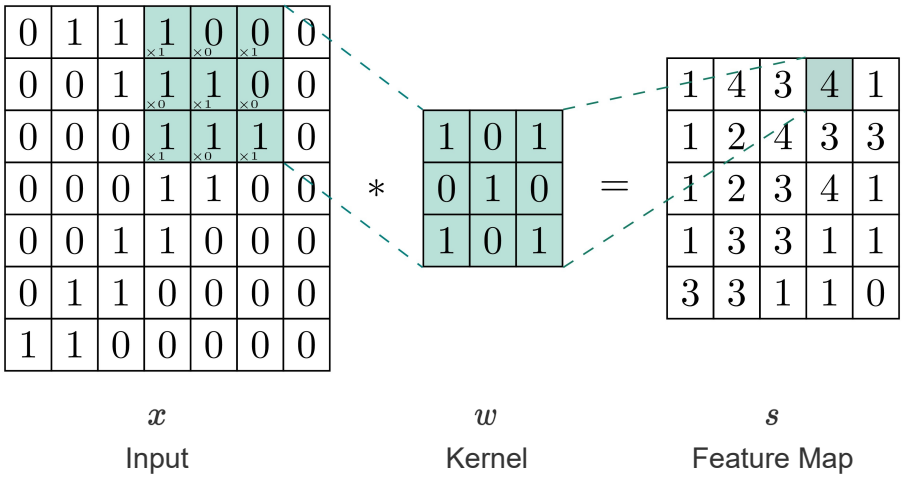


REFERENCE: The author (2023).

In this given context, Convolutional Neural Network (CNN) is a type of neural network that is commonly used for computer vision and natural language processing. The name CNN indicates the implementation of convolution operations.

The convolution is typically denoted with an asterisk, where the first argument x is referred to as the *input*, and the second argument w is called the *kernel*, the output s is referred to as the *feature map* (GOODFELLOW; BENGIO; COURVILLE, 2016). In the image context, the data input could be thought of as a 2-D grid of pixels, as well, the kernel will be defined as a 2-D matrix of weights. In an image application, the convolutional process will be the multiplication of the kernel over the area of the image, i.e. all the pixel values in the 2-D matrix (BUDUMA; LOCASCIO, 2017). Each time the multiplication occurs, it generates a new instance for the feature map output. The FIGURE 4 presents this operation on image data.

FIGURE 4 – CONVOLUTIONAL PROCESS IN IMAGE DATA



REFERENCE: Adapted from Tikz.net[1].

Canziani and LeCun (2020) highlight the strength of CNNs in dealing with natural signals that come in the form of multidimensional arrays, such as images for example. This feature is especially related to three image data characteristics that helps to understand why CNNs are so common in nowadays image-processing tasks:

- **Locality:** This characteristic corresponds to the local correlation between pixel values.

- **Compositionality:** This characteristic refers to the hierarchical composition of features in natural images.

- **Stationarity:** This characteristic pertains to the fact that essential pixels could appear anywhere in the image.

In this regard, CNNs are highly significant in the field of DL due to their ability to automatically learn and extract patterns from raw data. In computer vision, CNNs are essential for image classification, image recognition, object detection, and other related applications. Therefore, they have a positive impact in diverse domains such as product quality inspection, autonomous driving, and medical imaging.

[1] https://tikz.net/

2.3.2 Regularization techniques

One of the biggest challenges in ML and DL applications is ensuring that the algorithm will perform well in previous unsee data instances (GERON, 2019). In this scenario, a series of techniques were designed to reduce the test error, these strategies are called *regularization* (GOODFELLOW; BENGIO; COURVILLE, 2016). In the context of the present study, two regularization approaches have crucial importance.

Transfer Learning

Transfer learning is a technique that improves a model's performance in one domain by leveraging knowledge from a related domain. To better understand this concept, Pan and Yang (2010) suggest a real-life example outside of technical fields, if a specific person already knows how to play the electronic organ, learning how to play piano will be an easier task when compared with someone that has no musical background. In fact, this area of study has been driven by the observation that humans are capable of using their previously acquired knowledge to solve new problems more effectively and efficiently, which motivated researchers to explore the same approach in Machine and Deep Learning models.

There are several DL applications that transfer learning has been successfully tested, such as image classification (WEISS; KHOSHGOFTAAR; WANG, 2016). In this field, the use of transfer learning is usually applied by training models on large-scale datasets, such as ImageNet, and then keeping the weights of the convolutional layers frozen for the desired classification task (SHORTEN; KHOSHGOFTAAR, 2019).

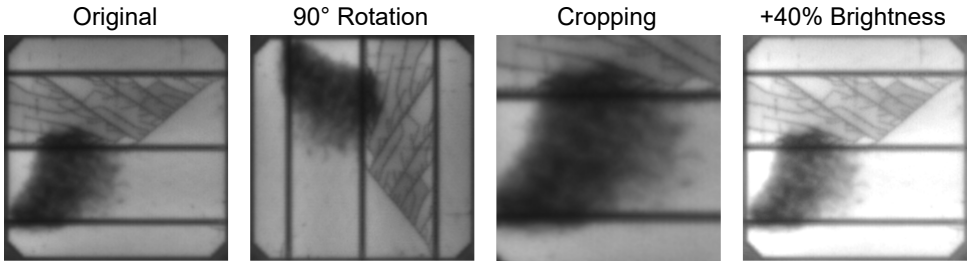
Data Augmentation

Data augmentation is a way to avoid or reduce overfitting issues enabling the model to learn from more data. This is especially relevant given the high cost and effort associated with collecting new data instances (PEREZ; WANG, 2017).

As defined by Goodfellow, Bengio, and Courville (2016), the use of traditional methods for data augmentation has been a particularly effective technique for image classification tasks. Given this context, Shorten and Khoshgoftaar (2019), describes the conventional image augmentation based on basic manipulation techniques. These could be classified as *geometric/spatial transformations*, such as flipping, cropping,

rotation, and translation, as well the *photometric transformations*, leading to a change in the image pixel value, such as the application of random color manipulation (color jitter), and kernel filter techniques (FIGURE 5).

FIGURE 5 – EXAMPLE OF BASIC TRANSFORMATION TECHNIQUES



REFERENCE: Adapted from Buerhop-Lutz et al. (2018).

One of the main problems with basic data augmentation techniques based on image manipulation is that they can introduce unrealistic variations into the data. In this respect, the "safety" of augmentation should be considered. For example, a simple spatial transformation of rotation should not be applied when the classification model is attempting to distinguish between "6" or "9". As well, a color transformation could not be feasible in a context where color is the main feature of a defect in the dataset. The rules to define a "safety" augmentation could vary a lot from one domain to another, this reinforces the challenge of defining general augmentation steps and turns this approach data-specific (SHORTEN; KHOSHGOFTAAR, 2019)

2.4 DEEP GENERATIVE MODELS

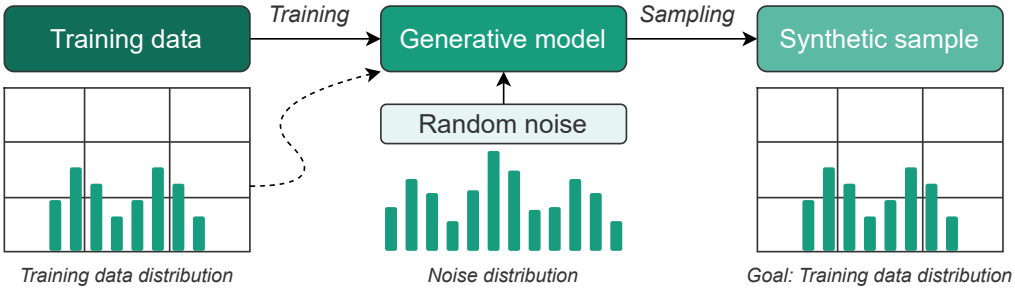
2.4.1 Introduction to deep generative models

Deep generative models have a fundamental focus on the probability distribution of the input data. In this scenario, it can be assumed that some unknown probabilistic distributions explain why some data may appear in the training dataset and why others do not. A generative model aims to learn the underlying patterns and structure of the data, getting to know the probabilistic distributions behind it. With this knowledge, the model will generate previously unseen data points, which are similar to the originals (FOSTER, 2019)

To perform, the deep generative model requires a dataset containing examples of the entity in focus for the specific use case. This dataset will be the *training data*

for the model. Each entity is called an *observation/instance* and each observation is composed of many *features* (FOSTER, 2019). FIGURE 6 summarizes a typical deep generative model process.

FIGURE 6 – DEEP GENERATIVE MODELS OVERVIEW



REFERENCE: Adapted from Foster (2019).

In a common approach, the main part of the Machine Learning problems is likely to be discriminative by nature, e.g. classify inputs with accuracy. One of the largest examples is image and natural language processing. Usually, these applications are in the field of supervised learning, where the training data is expected to have a label, and based on this the model maps the input and output.

As illustrated by Foster (2019), the discriminative nature of common supervised learning models, in a mathematical way, could be described as:

$$p\left(y|x\right) \text{: the probability of } y \text{ given an observation } x \tag{2.8}$$

In contrast, deep generative modeling has a probabilistic nature and addresses tasks such as image and text generation. In this context, this field of Deep Learning could be performed with or without the label of an observation, since it aims to estimate and learn the probability of seeing this specific observation or not.

Foster (2019) describes the statistical relation that supervised (EQUATION 2.9) and unsupervised (EQUATION 2.10) deep generative models are modeling:

$$p\left(x|y\right) \text{: the probability of observing observation } x \text{ given label } y \tag{2.9}$$

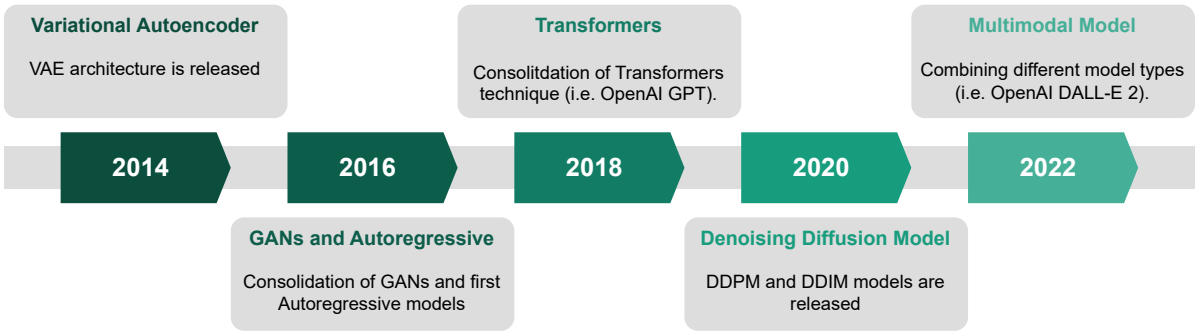
$$p\left(x\right) \text{: the probability of observing observation } x \tag{2.10}$$

It is important to mention the flexibility of the deep generative models since this DL domain could perform in both supervised and unsupervised learning fields.

2.4.2 The era of deep generative models

Deep generative models have their origins in the 1980s, but they have recently emerged as a leading Deep Learning implementation (BOND-TAYLOR et al., 2022). The main reasons are concentrated in the software, architectures, and hardware improvements, nowadays more data is available for training efficient networks in a powerful computational resource. In this scenario, the FIGURE 7 is a timeline representation from 2014 to 2022 of the main relevant deep generative models over this period.

FIGURE 7 – GENERATIVE AI TIMELINE



REFERENCE: Adapted from Foster (2023).

Before the rise of deep generative models, in academic and industrial environments discriminative modeling has been the driving force behind Machine and Deep Learning applications. Exploring the specific industrial and business domain, most of the nowadays solutions required by these fields have a discriminative nature. The goal of building, validating, and monitoring processes are discriminative tasks in almost all scenarios (FOSTER, 2019).

However, the application of deep generative modeling in a series of fields like image generation, speech recognition, speech generation, and robotics, are increasing a lot recently. The potential positive impact of implementing deep generative models extends beyond industrial applications, influencing businesses such as game design, cinematography, and music (OUSSIDI; ELHASSOUNY, 2018). The current status of deep generative models has shown remarkable progress in various fields, especially looking at new implementations such as the OpenAI ChatGPT[2] platform. Therefore, the expression *the era of deep generative models* fits perfectly to this domain.

[2] https://chat.openai.com/

2.5 VARIATIONAL AUTOENCODER

2.5.1 Introduction to Variational Autoencoder

As presented in FIGURE 7, the Variational Autoencoder (VAE) was one of the first deep generative model types to take place. Since then, it has become consolidated as one of the most fundamental architectures for deep generative modeling. The underlying theory of VAE shares similarities with the Autoencoder (AE) definition. As stated by Goodfellow, Bengio, and Courville (2016), an AE architecture consists of two functions, an encoder, defined by $h = f(x)$, and a decoder that produces a reconstruction $\hat{x} = g(h)$. The role of these two functions can also be defined as follows:

- **Encoder:** compress high-dimensional input data, such as an image, to a low-dimensional vector in the latent space.
- **Decoder:** perform the opposite operation, transforming the low-dimensional vector back to the original domain (FOSTER, 2019).

One of the main problems regarding the use of AEs for data generation is the difficulty of the encoder creating a regular latent space in a manner where the decoder could generate new data for the desired domain (BENGIO; COURVILLE; VINCENT, 2013).

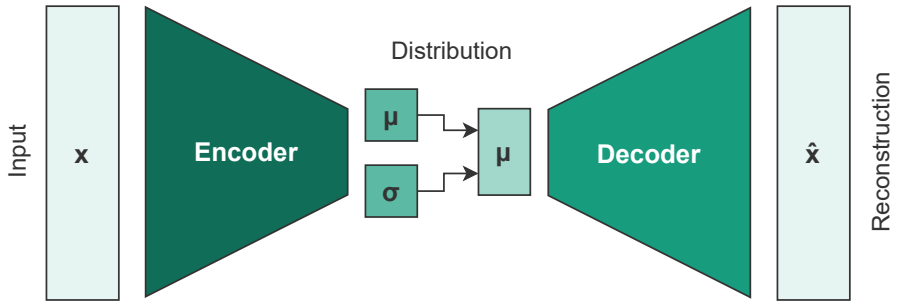
In this matter, the VAE takes place. During the training process, the encoder computes the distribution $q(z|x)$ and the decoder computes $p_{model}(x|z)$ (GOODFELLOW; BENGIO; COURVILLE, 2016). In other terms, the workflow of a VAE architecture could be explained as:

1. The input x is encoded as a distribution over the latent space as $q(z|x)$.
2. From $q(z|x)$, using the mean μ and standard deviation σ, a sampled latent representation z is defined.
3. This sample z pass through the decoder, $p_{model}(x|z)$, in order to generate the reconstruction.

At this point, the main difference between a standard AE and a VAE can be highlighted. In the AE architecture, the encoder compresses the input until a lower-dimensional latent space, such as a vector, for example. In a VAE, however, the input is

encoded as a distribution. In fact, this distribution is forced to follow a Gaussian shape, a procedure that adds extra regularization to the latent space (FIGURE 8).

FIGURE 8 – VARIATIONAL AUTOENCODER



REFERENCE: The author (2023).

Adding more regularization to the latent space, VAEs solve the problem accentuated at the begging of this section, becoming a great architecture for generative modeling. However, one of the prominent limitations of VAEs trained on image datasets is the tendency to generate images with a level of blurry.

2.5.2 Training VAEs

When training a VAE, the main objective is to minimize the loss function. In this model type, the loss is composed of two components:

$$\mathcal{L}(x, \hat{x}) = reconstruction\ loss + regularization\ term \tag{2.11}$$

The $reconstruction\ loss$, measures the differences between the original observations and the new observations reconstructed after the encoding-decoding process (CANZIANI; LECUN, 2020).

There are two common ways to apply the $reconstruction\ loss$, either using BCE, specially for binary or $(0, 1)$ range inputs, as:

$$\mathcal{L}(x, \hat{x}) = -\sum_{i=1}^{n} [x_i \log(\hat{x}_i) + (1 - x_i) \log(1 - \hat{x}_i)] \tag{2.12}$$

And also, for real domain inputs with MSE:

$$\mathcal{L}(x, \hat{x}) = \frac{1}{2} \|x - \hat{x}\|^2 \tag{2.13}$$

The second part of the loss equation, the $regularization\ term$, is based on the Kullback–Leibler Divergence (KLD), this is a way to measure the differences between two probability distributions. In the VAE implementation, this idea is applied to the quantity of how far the normal distribution, based on μ and σ, is from the standard Gaussian distribution (FOSTER, 2019).

Kingma and Welling (2013), in the original VAE paper, explore and define the KLD for this specific use as:

$$\mathrm{D}_{\mathrm{KL}}(q_\phi(z) \,||\, p_\theta(z)) = -\frac{1}{2}\sum_{j=1}^{J}(1 + log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2) \tag{2.14}$$

The KLD term penalizes the network for encoding a latent space distribution that differs significantly from the standard Gaussian distribution, forcing regularization on it, turning the network more feasible to image generation based on a latent input. The goal of VAE training is to minimize the $\mathcal{L}(x, \hat{x})$, either by minimization of the BCE/MSE loss function, or maximization of the KLD term.

2.6 GENERATIVE ADVERSARIAL NETWORKS

2.6.1 Introduction to GANs

Among the whole spectrum of deep generative models type, Generative Adversarial Networks (GAN) have emerged as a highly renowned model. Since Goodfellow et al. (2014) released the first paper of a GAN implementation, this architecture has been showing great capabilities in generating data, especially focusing on the computer vision domain.

GANs are a generative model type using Deep Learning techniques where two neural networks are trained in opposition to each other in a competitive manner. The first network is called Generator (G) and the second network called Discriminator (D).

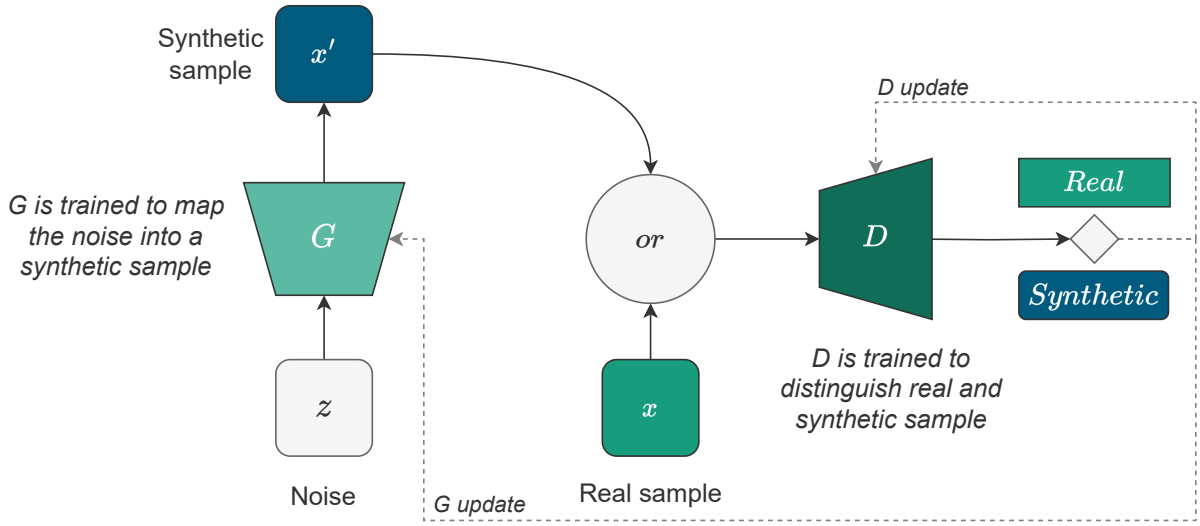
In a GAN architecture, G aims to produce synthetic data close to the real data distribution, while D has the goal of determining whether a sample comes from G, e.g. synthetic data, or from the real data. During the training, G continuously improves its output based on the feedback of D, until the best scenario of indistinguishable samples from real data (GOODFELLOW et al., 2014).

2.6.2 Training GANs

In the training procedure of GANs, the Generator network has the main function to map data distribution from some representation space, called a latent space, to the space of the data. Then, in a mathematical way: $G : G(z) \rightarrow R^{|x|}$. Where z represents a feature from the latent space, and x represents a feature from the desired domain, usually images, with $x \epsilon R^{|x|}$ (CRESWELL et al., 2018).

In contrast, the Discriminator network in a basic GAN can be described as a function that takes image data as input and produces a probability score, in a deterministic approach, indicating whether the input image is from the real data distribution or the Generator distribution, i.e. fake data. In mathematical terms: $D : D(x) \rightarrow (0, 1)$. In the optimal scenario for G, where it can map perfectly the data distribution of the dataset, D will become indifferent predicting 0.5 for all new image data inputs (CRESWELL et al., 2018). In this context, FIGURE 9 illustrates the process of training a GAN model.

FIGURE 9 – GAN TRAINING WORKFLOW



REFERENCE: Adapted fromCreswell et al. (2018).

As described by Goodfellow et al. (2014) the whole process of GAN training could be described as the following minimax equation:

$$\min_G \max_D V(D,G) = E_{x \sim p_{data}(x))}[logD(x)] + E_{z \sim p_z(z))}[log(1 - D(G(z)))] \quad (2.15)$$

The equation above, as known as the cost function of the GAN architecture, is composed of two parts. On the right side, the G cost tries to minimize the term

$log(1 - D(G(z)))$, leading to images more similar to the original dataset. On the left side, the D cost aims to maximize the probability of correctly distinguishing between real and fake data.

One of the biggest challenges referring to GAN applications is that they are broadly recognized as difficult to train. A common problem is an oscillating loss when both G and D loss starts to oscillate wildly. Besides this, the whole training procedure is prone to collapse, this occurs when G finds an easy way to trick D, and, even if the newly generated observations do not look similar to the original data, G will always return the same sample. This problem is also related to another one, the uninformative loss, once there is no direct relationship between the generated image quality and the Generator loss, tracking and applying optimization techniques becomes very difficult (FOSTER, 2019).

2.6.3 DCGAN architecture

As described in the subsection 2.3.1 CNNs are highly suitable for dealing with image data problems. The combination of CNNs and GANs figures is a natural step in terms of deep generative modeling development. In this context, one of the most successful implementations is the Deep Convolutional Generative Adversarial Network (DCGAN) (RADFORD; METZ; CHINTALA, 2016).

This architecture performs the training step in two deep convolutional networks, following the same minimax logic of the standard GAN. Besides this, the G architecture applies techniques to stabilize and speed up the training process, and the D network to better comprehend discriminative features from the input images. In general, the DCGAN applies a series of techniques in order to improve the training process, turning it more stable and less prone to nonsensical outputs.

The DCGAN architecture is an improvement of the standard GAN, implementing new valuable techniques. Therefore, it remains one of the most fundamental approaches for implementing this specific deep generative model type.

2.7 DENOISING DIFFUSION PROBABILISTIC MODELS

2.7.1 Introduction to diffusion probabilistic models

A diffusion process describes the spread or dispersion of some quantity over time due to a random motion. At the macroscopic level, it could be defined as a spontaneous "spreading" of particles from one concentrated region to another, due to the non-equilibrium status of the system (IBE, 2013).

The main idea of diffusion probabilistic models is to use a forward diffusion process to systematically and slowly destroy the structure of the data distribution, and after learning how to perform the reverse operation bring the distribution to the structure form again, also using diffusion processes.(SOHL-DICKSTEIN et al., 2015).

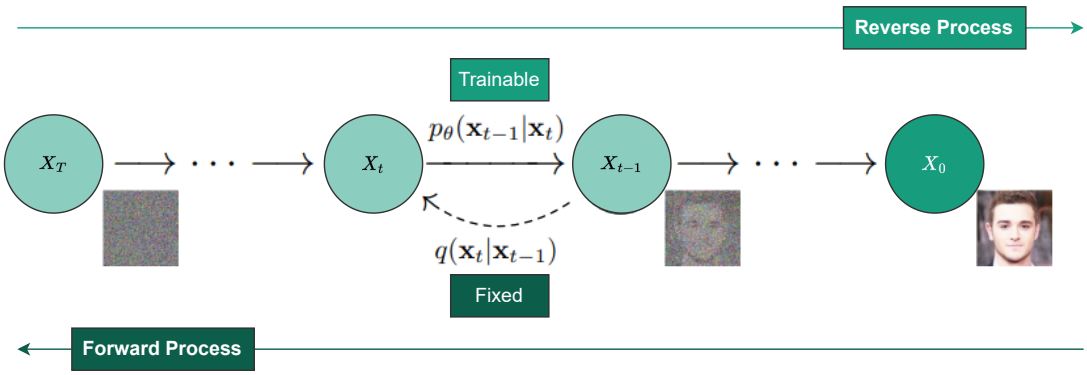
Based on the foundations of diffusion probabilistic models, several novel techniques have been developed in this area of deep generative models. One of the most famous is the Denoising Diffusion Probabilistic Model (DDPM) (HO; JAIN; ABBEEL, 2020). The improvements of this new technique are mainly based on novel connections between diffusion probabilistic models and denoising score matching with Langevin dynamics, which is a technique used to estimate the parameters of a probability distribution by solving a partial differential equation. As a result, the DDPM has become the new standard implementation of diffusion architectures, especially due to the ability to produce high-quality image samples. In this scenario, even considered as a "vanilla" when compared with other diffusion models, the performance of the DDPM architecture could be matched as the state-of-the-art in other model types, such as GANs and VAEs.

2.7.2 Training Denoising Diffusion Probabilistic Models

The DDPM architecture, in general, follow the basis of the diffusion probabilistic models training. The training procedure involves estimating small perturbations during the process, leading to a tractable and flexible result. The operation related to destroying and reconstructing the data distribution described in the subsection 2.7.1 above, is directly related to the training of a DDPM and is called *forward process* and *reverse process* respectively (SOHL-DICKSTEIN et al., 2015).

FIGURE 10 represents both forward, the process to add noise to the image, and the reserve step, responsible for bringing the image back from the noise distribution.

FIGURE 10 – DDPM FORWARD AND REVERSE PROCESS



REFERENCE: Adapted from Ho, Jain, and Abbeel (2020).

Forward process

The forward diffusion process is responsible to add a small amount of Gaussian noise in a data point sampled from the real data distribution $x_0 \sim q(x_0)$. The noise is added in T steps, generating a series of noisy samples similar to $x_1, x_2, ..., x_T$. The level of added noise follows a variance schedule $\beta_1, \ldots, \beta_T$. Mathematically, this process could be described by:

$$q(x_{1:T} \mid x_0) := \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \quad q(x_t \mid x_{t-1}) := \mathcal{N}(x_t; (\sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{2.16}$$

Where the data sample x_0 will gradually become similar to an isotropic Gaussian distribution as far as the value of step t increases, being satisfied in the limit $T \to \infty$. A great property in the forward process is that this implementation allows sampling x_t at an arbitrary timestamp t, this feature, obtained after a reparameterization trick, is very helpful during the training (HO; JAIN; ABBEEL, 2020).

Reverse process

At the end of the forward process, the model will end up basically with a Gaussian noise distribution, $N(x_T; 0, I)$ valid for small values of β_t. Now, the goal is to reverse the previous process to recreate the true sample from the noise, the model will perform this by removing the noise at each timestep to obtain back the original sample. This process could be defined as:

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \quad p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2.17}$$

Ideally, after the reverse process, the implementation will end up with the original sample again (HO; JAIN; ABBEEL, 2020).

The training of DDPM models is focused on the reverse process since the forward is defined by fixed rules based on β_t. The objective function for this model type is similar to the VAE implementation described in subsection 2.5.2. However, Ho, Jain, and Abbeel (2020), proposed a simplified version of the loss function for DDPM models, the mathematical representation is defined as follows:

$$L_{simple}(\theta) := E_{t,x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta\left(x_t, t\right)\|^2\right] \tag{2.18}$$

As described by the authors, optimizing this simplified function (defined as reweighted), leads to better results in sample quality. This conclusion was derived after empirical testing and experimental runs. After the training procedure, the DDPM model will be capable of mapping a Gaussian noise input to the distribution of the original image dataset. This results in the generation of new synthetic data instances where the pixel value distribution imitates the original dataset and is based on the noise input.

2.7.3 DDPM architecture

Besides all the implementation of the training logic described above in the subsection 2.7.2, the main definitions in terms of architecture design for DDPM are based on the reverse process, specific in the model definition to perform this task.

Ho, Jain, and Abbeel (2020), follows the backbone of a PixelCNN+ model (SALIMANS et al., 2017), which is a U-Net architecture. This architecture was introduced by Ronneberger, Fischer, and Brox (2015), and consists of a contracting path and an expansive path, very similar to an AE implementation. The U-Net architecture allows information to be transmitted from high-resolution, early layers to later layers. These "shortcuts" can concatenate features from the original image, simultaneously capturing more advanced semantic information from the deeper layers.

Modern implementations of the U-Net architecture usually apply self-attention and transformers techniques, as well as residual blocks, with the aim of better results. Ho, Jain, and Abbeel (2020) described in detail all the modern methods selected for the DDPM implementation, in the following topics the main characteristics are highlighted:

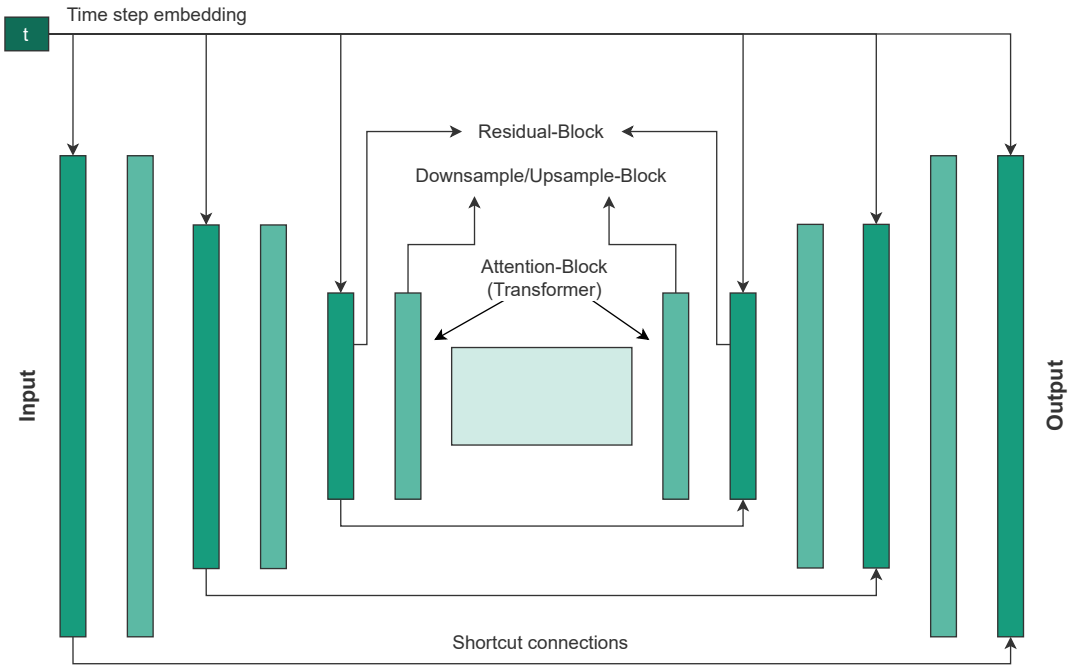
• **Residual-Block:** The concept of residual blocks was proposed by He et al. (2015) when defined the Residual Neural Network (ResNet), the core idea is to ease the training by using residual representations and shortcut connections.

• **Attention-Block (Transformer):** Vaswani et al. (2017) proposed the main idea regarding the Transformer network based on self-attention mechanisms, this architecture allows the network to select the most relevant parts of the input sequence.

• **Timestep embedding:** In the Transformer mechanism, a sinusoidal timestep embedding is used to incorporate the diffusion time t into each Residual-Block of the U-Net architecture.

With a specific focus on comprehension objectives, the general overview of the model is presented in FIGURE 11.

FIGURE 11 – DDPM U-NET ARCHITECTURE



REFERENCE: The author (2023).

In addition to the conventional DDPM, the Conditional DDPM model was introduced by Nichol and Dhariwal (2021). To make it class-conditional, the authors incorporate class information into the training by utilizing the same pathway as the timestep t. Furthermore, recent applications of Conditional DDPM often incorporate

techniques such as *classifier-free guidance* and *exponential moving average*, establishing this architecture as one of the state-of-the-art approaches for generating synthetic images (HO; SALIMANS, 2022; HAYNES; CORNS; VENAYAGAMOORTHY, 2012).

2.8 IMAGE DATA SYNTHETIZATION

2.8.1 Deep generative models and image data synthesis

The era of the deep generative models described in the subsection 2.4.2, holds significant importance to the advance of image data synthesis techniques. In this scenario, these models are employed in the image data domain to generate new observations unseen until the moment. Deep generative models are very effective in synthesizing new image data due to their ability to model complex and high-dimensional data distributions.

As described by Tyagi and Yadav (2021), nowadays the image synthesis field is crucial for a series of image processing tasks, such as image classification and object detection. In this regard, the application of image synthesis has become more and more common in different areas of knowledge. As will be detailed in subsection 2.8.2, one of the typical applications is in synthetic augmentation. Another notable example is OpenAI's DALL-E application (FIGURE 12), a 12-billion parameter version of OpenAI's GPT-3 model trained to generate images from text input (RAMESH; GOYAL; DOVRAT, 2021).

FIGURE 12 – DALL-E IMAGE EXAMPLE

REFERENCE: DALL-E[3].

The image above serves as an example of DALL-E's capabilities and was synthetically generated in response to the author's request for DALL-E to create *"a high-definition image of a machine gear with a defective tooth"*.

2.8.2 Synthetic images for data augmentation

Implementing ML and DL models could be a tricky task, especially due to the number of possible limitations and constraints. Insufficient or non-representative training data is a common challenge, even for simple models thousands of data instances are commonly required. Additionally, it is important to ensure that the training sample encompasses new cases that the model must generalize to. If the training data does not attend to these necessities, the model will be unable to make predictions for unseen data, overfitting the training set (GERON, 2019).

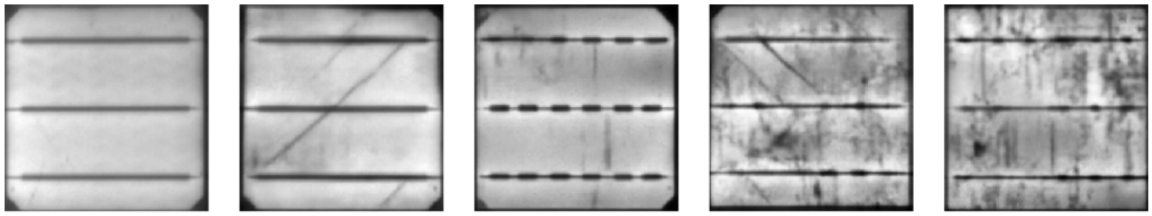
Recently, as described by Shorten and Khoshgoftaar (2019) in their paper *A survey on Image Data Augmentation for Deep Learning*, the use of deep generative models for performing data augmentation techniques is increasing rapidly, representing an advantageous solution for small data problems. Synthetic augmentation is similar to conventional approaches, but uses synthetic data produced by deep generative models to increase the training dataset size, enabling a more accurate and robust model to be trained even in situations with limited data availability. In theory, one significant advantage compared to conventional methods is that synthetic augmentation represents the true distribution of the original data.

As reported by Peres et al. (2020), in a major part of real manufacturing operations the conditions that generate data for an AI implementation are undesired, since this data is often related to defects or anomalies in the production line. Therefore, this characteristic leads to an unfeasible process of getting more data, both from an economic and operational point of view.

Considering this circumstance, Tang et al. (2020) described the use of GAN-based models in a defect identification of a photovoltaic module using electroluminescence images use case (FIGURE 13). The authors reported an increase in classification accuracy by applying generative data augmentation, front traditional techniques. Combining both approaches led to an even greater improvement.

[3] https://labs.openai.com/

FIGURE 13 – SYNTHETIC IMAGES IN A PHOTOVOLTAIC DATASET



REFERENCE: Tang et al. (2020).

In addition, Jain et al. (2020) described the application of three GAN-based models in a manufacturing environment use case of the metal surface defect (NEU dataset (SONG; YAN, 2019)). The three architectures are the DCGAN, the Auxiliary Classifier GAN, and the Information-theoretic GAN, leading to better classification metrics in most of the collected results. Furthermore, Yan, Yeh, and Sergin (2019) combined GAN and VAE to build a deep generative model for metal surface defects, and Lian et al. (2020) applied a defect exaggeration GAN, defining an architecture that highlights the defects in newly generated images.

These are just some examples of deep generative models applied in the manufacturing domain. One characteristic of these applications, as also described by Shorten and Khoshgoftaar (2019) in their survey, is the fact that GAN-based models are more common than any other model type. Therefore, there is a research gap in the evaluation of deep generative models for manufacturing applications, particularly in the exploration and comparison of various model types.

2.8.3 Evaluation metrics

Even with the recent development of deep generative models, it is still very complicated to assess which architecture performs better. The primary challenge with evaluation lies in the probabilistic nature of these model types (LUCIC et al., 2018). As a consequence, traditional evaluation metrics generally cannot be applied or could lead to inadequate results (THEIS; OORD; BETHGE, 2016).

To overcome this limitation, researchers have turned to qualitative comparisons, such as assessing the visual quality of generated samples. However, even valid on specific occasions, these approaches are subjective and may lead to incorrect conclusions (GERHARD; WICHMANN; BETHGE, 2013). In this regard, the lack of quantitative met-

rics focus on evaluating deep generative architecture is one of the biggest impediments to the future development of this area (SAJJADI et al., 2018).

In this context, it is worth noting that different deep generative model types could have different evaluation metrics that fit better in the architecture definition. (BORJI, 2018; NICHOL; DHARIWAL, 2021). In general, benchmarks between deep generative models usually performed by cross-references in papers or via ML hubs in specialized websites, follow the use of Fréchet Inception Distance (FID) and Inception Score (IS) for comparison.

Salimans et al. (2016), introduced the IS metric for assessing deep generative models. This metric usually applies an Inception Network pre-trained model in the ImageNet dataset for feature extraction (SZEGEDY et al., 2016; DENG et al., 2009). IS focuses on the evaluation process of generated samples, determining how well these new samples can be classified and how diverse they are to represent the original data. The IS uses the KLD between the conditional label distribution $p(y|x)$ of samples, and the marginal distribution $p(y)$, which is obtained from all samples. In general, the metric promotes low entropy in $p(y|x)$, indicating the easy classification and better sample quality, and high entropy in $p(y)$, representing high diversity when all classes are equally represented. The IS could be mathematically defined as:

$$IS(G) = \exp\left(E_{x\sim p_g} D_{KL}(p(y|x)||p(y))\right) \tag{2.19}$$

Where, $x \sim p_g$ indicates that x is an image that had been sampled from p_g, the term $D_{KL}(p(y|x)||p(y))$ is the KLD between the conditional class distribution, $p(y|x)$ is the conditional class distribution, and $p(y) = \int_{\mathcal{X}} p(y|x)p_g(x)dx$ is the marginal class distribution (BARRATT; SHARMA, 2018).

The inception score shows a suitable correlation with the generated image quality. However, it has some limitations regarding the use beyond the ImageNet dataset, since the base model was trained on this data any other applications could give misleading results (BARRATT; SHARMA, 2018). Besides, IS metric is easily affected by the resolution of the samples that will evaluate (BORJI, 2018).

The other common metric for the evaluation of deep generative models, FID, was proposed by Heusel et al. (2018). This metric measures the similarity between the distribution of generated images and the distribution of real images. FID uses feature

representations obtained from a pre-trained deep neural network, usually an Inception Network as well. The mean and covariance of both the generated and real data are estimated by treating the embedding layer as a continuous multivariate Gaussian. Finally, the metric applies the concept of Fréchet distance (a.k.a Wasserstein-2 distance), which is a measure of distance between two multivariate Gaussian distributions to quantify the quality of the synthetic data (FR'ECHET, 1957)

The mathematical form of FID could be found below:

$$FID(r,g) = ||\mu_r - \mu_g||_2^2 + Tr\left(\Sigma_r + \Sigma_g - 2\Sigma_r\Sigma_g\right)^{1/2}) \tag{2.20}$$

Where, (μ_r, Σ_r) and (μ_g, Σ_g), are the mean and covariance of the real and synthetic data, respectively. The lower the value of FID, the better the model's capability to generate similar images to the original data. The main constraint regarding FID implementations is the assumption that features are of Gaussian distributions, which is not always true in practice, especially in limited data analysis (BORJI, 2018).

These two metrics are essential when evaluating the performance of deep generative models, leading to a better understanding of the performance in a quantitative manner. In addition to these techniques, some researchers have suggested employing metrics derived from the field of image quality assessment, such as the Structural Similarity Index Measure (SSIM). The SSIM metric was proposed by Wang et al. (2004) and is focused on analyzing how similar an image is compared with a ground truth image reference. This procedure involves the comparison of corresponding pixels and their neighborhoods in both images and could be defined as follows:

$$SSIM\left(x,y\right) = I\left(x,y\right)^{\alpha} C\left(x,y\right)^{\beta} S\left(x,y\right)^{\gamma} \tag{2.21}$$

Where x and y represents pixel values, I is luminance, C contrast, and S structure (BORJI, 2018). Even in data synthetization applications, the SSIM can provide valuable insights and support both IS and FID metrics in a more robust analysis.

3 RESEARCH DESIGN

The research design chapter outlines the methods used to conduct the study. The primary objective of this work is to employ and compare distinct deep generative models for synthetic image augmentation in order to perform a DL-model-based visual quality inspection in a challenging manufacturing use case with limited data.

The chapter starts with the selected use case for the study along with the definition of the experimental setup. Additionally, reports the deep generative models employed and outlines the assessment techniques used to evaluate them.

3.1 MANUFACTURING ENVIRONMENT USE CASE

3.1.1 Use case requirements

One of the primary objectives of this work is to study the application of deep generative model techniques in a manufacturing environment domain. Accordingly, a series of selection criteria have been established to facilitate the process of identifying a suitable use case for the experiment.

The criteria are split into four main areas, and are as follows:

1. Application Domain:
   - The use case must be related to the manufacturing domain.
   - The use case must have a focus on product quality inspection.
2. AI Task Domain:
   - The use case dataset must be in the domain of supervised learning.
   - The dataset annotation must allow the implementation of classification models, even with the necessity of minor modifications.
3. Data Type Domain:
   - The data type of the use case dataset must be an image.
   - The dataset images must allow an easy qualitative data quality assessment of future synthetic data points.

4. Challenges Domain:

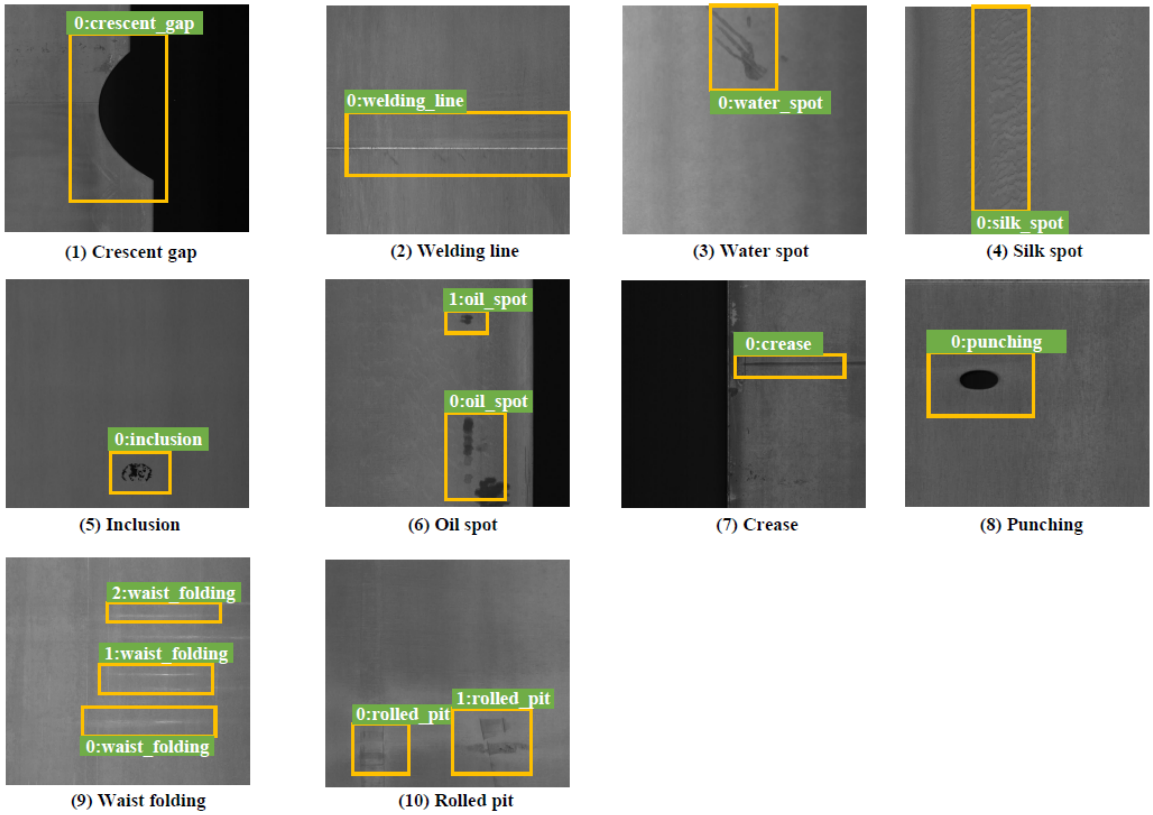
   - The dataset must exhibit a degree of imbalance among the classes.
   - The dataset must present a scenario of limited data.

3.1.2 GC10-DET dataset

Based on the requirements defined above, open-source datasets were pre-selected. Among them, the GC10-DET dataset was the final selection for the development of this work (LV et al., 2020).

The GC10-DET is a large-scale metallic surface defect detection dataset composed of 2312 grayscale real images classify into ten different defect classes, punching (Pu), weld line (Wl), crescent gap (Cg), water spot (Ws), oil spot (Os), silk spot (Ss), inclusion (In), rolled pit (Rp), crease (Cr), and waist folding (Wf) (LV et al., 2020). The FIGURE 14 provides an overview of the dataset classes.

FIGURE 14 – DEFECT CLASSES IN THE GC10-DET DATASET



REFERENCE: Lv et al. (2020)

As reported by Lv et al. (2020) in the released paper of the dataset, the description of each defect class could be found as follows:

- **Punching:** Mechanical failure producing unwanted punching in the steel strip.
- **Welding Line:** Weld line produced to join two could of the steel strip, needs to be removed in further process.
- **Crescent gap:** Mechanical failure produced by cutting procedures.
- **Water spot:** Stain caused by water drying during the strip production. The criteria for this defect may differ among the products but could be easily misclassified as an oil spot.
- **Oil spot:** Visual defect produced by oil contamination, usually from the lubrification of mechanical parts in the production line.
- **Silk spot:** A wave-like plaque defect generated by variation in the process parameters, in general temperature and pressure.
- **Inclusion:** A typical metal surface defect, generally showing small spots or strips, the inclusion defect aggregates material to the steel strip.
- **Rolled pit:** Periodic protuberance defect distributed along the steel strip length, mainly caused by damage in production line rolls.
- **Crease:** A mechanical defect producing vertical transverse fold shape defect across the steel strip, produced in general by failure in the uncoiling process.
- **Waist folding:** Large local deformation defect generating folds in the steel strip, usually related to the low carbon chemical composition of the steel alloy.

The TABLE 1 describes the number of instances per class in the GC10-DET for the train, test, and complete dataset. Detailed information on the ratio between the train and test sets can be found in subsection 3.2.1. As observed, the dataset is imbalanced and with a significant difference between the classes silk spot (majority class) and rolled pit (minority class). This imbalance adds an important aspect for the deep generative modeling final evaluation, especially regarding the capability of the models to synthesize new instances based on varying amounts of available train data. For the development of this work, the dataset was downloaded from Kaggle[1] website.

[1] https://www.kaggle.com/datasets/alex000kim/gc10det

TABLE 1 – INSTANCES PER CLASS IN THE GC10-DET

Discrete class	Class	Train set	Test set	Total instances
0	Punching	140	79	219
1	Welding Line	174	99	273
2	Crescent gap	144	82	226
3	Water spot	184	105	289
4	Oil spot	130	74	204
5	Silk spot	416	235	651
6	Inclusion	138	78	216
7	Rolled pit	19	12	31
8	Crease	33	20	53
9	Waist folding	96	54	150
TOTAL		1474	838	2312

REFERENCE: The author (2023).

The GC10-DET dataset is labeled for defect detection based on object localization applications. In this study, an adapted version of the labels was applied, potentially leading to the issue of assigning images with multiple defects to a single class. This characteristic could impact the final performance of the classification.

In summary, is important to mention that this dataset presents some common real-world challenges, such as the imbalance issue among classes, and the labeling problems. Given this context, the GC10-DET dataset is a relevant use case for the manufacturing environment, offers the possibility to assess the impact of different train dataset sizes on the synthetic data quality, and provides an opportunity to evaluate realistic scenarios and challenges.

3.2 EXPERIMENTAL SETUP

3.2.1 Initial experimental setup

The initial experimental setup represents an essential component of the study, serving as a guide for the developmental phase of the research. The complete sequence of steps and experiments of the workflow is detailed as follows:

1. The raw GC10-DET is initially split between the train and test datasets in a 64/36% ratio, respectively. The test dataset is saved in another folder directory remaining intact until the final classification step.

2. The training dataset will generate four scenarios of training data. The first one is the raw training data, which is the original data from the GC10-DET resized

and normalized to meet the requirements of the classification model (see subsection 3.5.2).

3. The second scenario is composed of the raw plus augmented training data, produced by conventional augmentation techniques applied to the raw training dataset, such as spatial and color transformations (section 3.3). Each class of the raw training data will be augmented until achieve 1000 instances per class.

4. The third scenario represents the raw and synthetic training data, produced by the deep generative models defined in the section 3.4. There will be a different final training dataset for each deep generative model selected. Similarly to scenario 2, after adding the synthetic images to the raw training dataset all classes will have 1000 instances. At this point, the different deep generative models are also evaluated by FID, IS, and SSIM metrics.

5. The fourth scenario combines conventional and synthetic augmentation techniques. This particular scenario is created by randomly selecting samples from both conventional augmented and synthetic data. In the end, the original images are kept and the pipeline adds the necessary amount of instances to oversample the dataset until 1000 instances per class.

6. The classification model is trained on each of the outcome scenarios and evaluated in the test dataset. The training and evaluation steps are performed in multiple runs. Finally, the collected metrics are statistically analyzed to assess the performance of each scenario (see subsection 3.5.3).

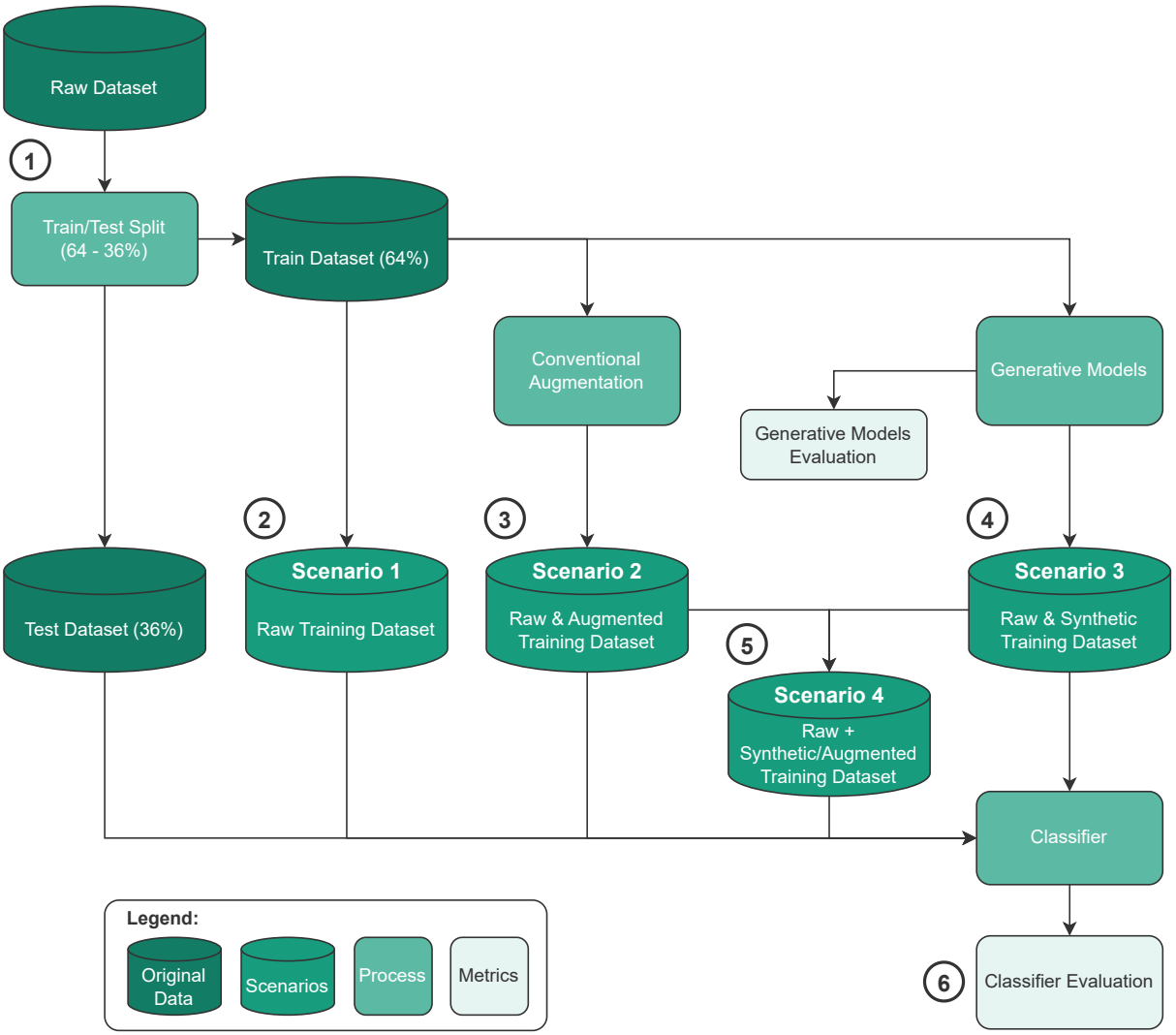
In this context, the TABLE 2 helps to visualize the differences in the four initial scenarios in terms of training data combination, and the FIGURE 15 displays a visual workflow of the experimental setup matching the topics above.

TABLE 2 – TRAINING DATA COMBINATION OVER INITIAL SCENARIOS

Scenario	Raw	Conventional augmented	Synthetic augmented
1	X		
2	X	X	
3	X		X
4	X	X	X

REFERENCE: The author (2023).

FIGURE 15 – INITIAL EXPERIMENTAL SETUP



REFERENCE: The author (2023).

3.2.2 Extra scenarios on successful synthetic data

After the analysis of the results, the most successful synthetic data based on image quality and classification performance metrics will be assessed through more two different scenarios:

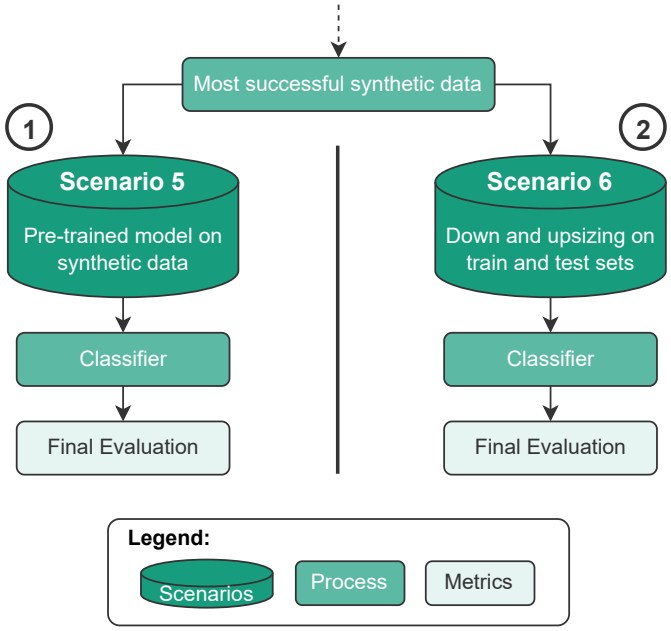
1. The fifth scenario will pre-train the classification model on the selected synthetic data. The final purpose is to evaluate the performance impact on scenario 2 (conventional augmentation) by using the pre-trained model.

2. The sixth scenario, where the classification model is evaluated after downsizing and subsequent resizing the raw training and test sets. This procedure aims to

ensure the same level of information in the original and synthetic data, evaluating the impact on the final performance.

These new approaches are defined as Scenario 5 and 6, respectively. In this context, FIGURE 16 displays in a visual manner the implementation. The numbers in the figure are aligned with the description above.

FIGURE 16 – EXTRA SCENARIOS ON SUCCESSFUL SYNTHETIC DATA



REFERENCE: The author (2023).

It is important to emphasize the motivation behind the Scenario 6 definition. The deep generative models generate 64x64 images, especially due to the computational complexity. The images are then resized until 224x224 pixels to match the classification model requirements. On the contrary, the original data used in the raw training dataset and in the test set are downsized from 2048x1000 to 224x224 pixels, leading to higher information density per pixel compared to the synthetic instances.

Given this context, Scenario 6 was proposed to address this issue. In the pipeline, the original images are downsized to 64x64 and resized again to 224x224 pixels. Balancing possible information compressions in both augmentation methods.

3.3 CONVENTIONAL AUGMENTATION

The conventional augmentation technique implemented to oversample the raw training dataset (section 3.2), was composed by the transformations in TABLE 3.

TABLE 3 – CONVENTIONAL AUGMENTATION IMPLEMENTATION

Transformation	Domain	Value	Probability
Rotation	Spatial	180°	0.5
Horizontal Flip	Spatial	True	0.5
Vertical Flip	Spatial	True	0.5
Brightness	Color	0.75 - 1.25	0.75
Contrast	Color	0.75 - 1.25	0.75
Saturation	Color	0.75 - 1.25	0.75
Gaussian Blur	Color	0.1 - 2.0	0.25

REFERENCE: The author (2023).

As defined by Shorten and Khoshgoftaar (2019), these transformation techniques are some of the basic approaches to perform conventional augmentation. In this use case, rotation angles other than 180° were excluded to maintain the similarity to the original dataset. Additionally, the values of the color transformations were qualitatively validated in terms of brightness, contrast, and saturation appearance.

3.4 SYNTHETIC AUGMENTATION

3.4.1 Deep generative models selection

In an effort to define the deep generative models for this study, some criteria were established to guide the selection:

- **Criterion 1 - Popularity and open source code:** The first criterion refers to defining the model types and is based on popularity. Three types will be chosen based on their usage and acceptance within the deep learning community. Popularity will be measured using the submission count in the Papers With Code[2] website on the CIFAR10 dataset, which is a benchmark for image generation.

- **Criterion 2 - *Vanilla* architectures:** The second criterion refers to selecting model architectures of each defined model type. The selected architecture should be a commonly used and tested solution applied in several domains. Both works of GM et al. (2020) and Bond-Taylor et al. (2022) were consulted to define the *vanilla* architectures.

- **Criterion 3 - State-of-the-art architecture:** At the end of the process, one of the selected *vanilla* architectures will be chosen as the basis for a *state-of-the-*

[2] https://paperswithcode.com/

art model implementation. This will ensure a representative coverage of model complexities while covering the main architecture types.

After applying the criteria in the current scenario of the deep generative model community, the outcome is highlighted in TABLE 4.

TABLE 4 – SELECTION OF DEEP GENERATIVE MODELS

Model type	**Model architecture**	**Complexity**
Generative Adversarial Network	DCGAN	Vanilla
Variational Autoencoder	Convolutional VAE	Vanilla
Diffusion Probabilistic Models	Unconditional DDPM	Vanilla
Diffusion Probabilistic Models	Conditional DDPM	State-of-the-art

REFERENCE: The author (2023).

These are the selected deep generative model types and architectures for the development of the present work. In this context, the implementation of each of them will be detailed in the coming sections.

3.4.2 Convolutional VAE implementation

The Convolutional VAE implementation follows the definition of the base paper released by Kingma and Welling (2013). In the paper, the authors employed an MLP architecture to build the encoder-decoder structure. However, especially on account of the image processing field, a neural network with four convolutional layers was chosen instead. The architecture was inspired by PyTorch-VAE [3], a well-known open-source GitHub repository for VAE implementations in PyTorch.

Regarding the architecture, the encoder consists of four convolutional and three linear layers that take 1x64x64 images as input and convert them to a distribution. On the other hand, the decoder follows almost the opposite architecture of the encoder and takes the input distribution to synthesize an image with the same format as the input. The model was trained for 800 epochs on individual subsets that exclusively contain instances of a specific class from the entire training dataset.

In this scenario, the TABLE 5 presents the hyperparameters for the Convolutional VAE implementation. Where *weight decay* is a type of regularization technique for deep learning models applied directly in the optimizer definition.

[3] https://github.com/AntixK/PyTorch-VAE

TABLE 5 – CONVOLUTIONAL VAE HYPERPARAMETERS

Hyperparameter	**Value**
Seed	42
Batch size	4
Input size	1 x 64 x 64
Latent dimension	4
Optimizer	Adam
Learning rate	0.0001
Weight decay	0.00001
Epochs	800
Loss function	BCE

REFERENCE: The author (2023).

3.4.3 DCGAN implementation

The DCGAN architecture implemented in this work closely follows the definition presented in the Radford, Metz, and Chintala (2016) paper. To maintain consistency with the original approach the same hyperparameters were used, with only minor modifications such as the number of epochs. The official PyTorch DCGAN[4] implementation was also consulted for a practical overview and adapted to fit the paper's definitions.

The TABLE 6 presents the hyperparameters for the DCGAN implementation. Where *Adam beta 1* and *2* are parameters for the Adam optimizer as described in the EQUATION 1.1.

TABLE 6 – DCGAN HYPERPARAMETERS

Hyperparameter	**Value**
Seed	999
Batch size	8
Input size	3 x 64 x 64
Latent vector size	100
Optimizer	Adam
Learning rate	0.0002
Adam beta 1	0.5
Adam beta 2	0.999
Epochs	800
Loss function	BCE

REFERENCE: The author (2023).

In the architecture, the Generator (G) receives a latent space input of a 100-dimensional noise vector after performing successive conversions using transpose convolutional layers until gets a 3x64x64 image as output. In total, G is composed of five transpose convolutional layers, applying batch normalization and a ReLU activation

[4] https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

function between each of them. The Discriminator receives synthetic or original images as input, then classifies the input as real or synthetic, providing feedback to increase the G performance. The DCGAN model was also trained for 800 epochs on subsets containing a unique class of the training dataset.

3.4.4 Unconditional DDPM implementation

The DDPM model was implemented based on the original paper by Ho, Jain, and Abbeel (2020), which follows an unconditional approach. However, some adjustments were made to the hyperparameters to optimize the model training for the specific use case and available computational resources.

The TABLE 7 presents the hyperparameters for the unconditional DDPM implementation. Where *noise steps* represents the number of steps that the noise is added to the original images, following the noise schedule parameterized by the values of *beta start* and *beta end*.

TABLE 7 – UNCONDITIONAL DDPM HYPERPARAMETERS

Hyperparameter	**Value**
Seed	42
Batch size	8
Input size	3 x 64 x 64
Noise steps	300
Optimizer	Adam
Learning rate	0.00001
Noise scheduler beta start	0.0001
Noise scheduler beta end	0.02
Epochs	400
Loss function	MSE

REFERENCE: The author (2023).

The forward process, responsible to add noise to the original images based on the noise scheduler, is constructed with basic Python code where the key feature is the noising processing algorithm. The reverse process, responsible to remove the noise from the images, is composed of a U-NET architecture as the backbone, with residual blocks and transformers applied. The DDPM model was trained for 400 epochs on each subset of a specific class of the training dataset.

3.4.5 Conditional DDPM implementation

The structure of the conditional DDPM model is based on the same backbone as the previously defined DDPM. However, the key difference is that the conditional DDPM is trained on the entire training dataset, rather than class-specific subsets. As a conditional model, the network takes all the training data as input, along with the instance label to differentiate between the different classes of data.

This architecture applies state-of-art techniques, such as classifier-free guidance and exponential moving average, in order to generate better quality synthetic data and smooth the training process. As a consequence of having differences in the architecture, new hyperparameters are defined for the conditional DDPM implementation, as shown in the TABLE 8.

TABLE 8 – CONDITIONAL DDPM HYPERPARAMETERS

Hyperparameter	**Value**
Seed	42
Batch size	4
Input size	3 x 64 x 64
Noise steps	350
Optimizer	Adam
Learning rate	0.00001
Noise scheduler beta start	0.0001
Noise scheduler beta end	0.02
Epochs	180
Loss function	MSE

REFERENCE: The author (2023).

3.5 IMAGE QUALITY ASSESSMENT

3.5.1 Image quality metrics

Fréchet Inception Distance (FID)

The FID metric was implemented based on the FID score for PyTorch[5] a well-known open-source GitHub repository for the use of this metric in PyTorch framework. The FID was applied in the original/raw training dataset and the synthetic training datasets. The last evaluation was conducted for each deep generative model described in the section 3.4.

[5] https://github.com/mseitzer/pytorch-fid

The analysis was conducted in a class-wise definition, based on a randomly selected subset of images. The value of FID for each class in the dataset was evaluated, and then a weighted average based on the number of instances per class was applied to define the general FID value for the dataset.

The FID metric exhibits certain limitations in the present use case, especially related to the number of instances. Besides, the random selection of the subsets may introduce a bias in the analysis. Consequently, it is crucial to consider the assessment of other relevant metrics to support the FID results.

Inception Score (IS)

The IS was also based on a widely-known GitHub repository[6] for this metric, with minor modifications to ensure the use of the most recent PyTorch conventions related to pre-trained models.

The implementation follows the same FID approach regarding scenarios, keeping the class imbalance among the subsets. However, the IS was evaluated for the entire dataset in 10 random subsets, since the main characteristic of the metric does not allow a class-wise approach.

Structural Similarity Index Measure (SSIM)

The SSIM was applied using a method inside the *Scikit-image*[7] library for image processing in Python, enabling faster and standard evaluation.

For this purpose, two images were randomly selected from the original and synthetic data in the same class. Subsequently, the value of SSIM was obtained for this image pair. This procedure was repeated 350 times, in a five runs analysis, mainly to mitigate the impact of any potential bias introduced by the random selection. Finally, the average SSIM per class and the weighted average for the whole dataset were obtained based on the number of instances.

3.5.2 Classification model - DenseNet201

The goal of using a classification model to evaluate the image quality is to bring the assessment close to a realistic structure, where the data is used for a typical DL

[6] https://github.com/sbarratt/inception-score-pytorch
[7] https://scikit-image.org

task, in this case, image classification. Given this particular context, each of the four initial scenarios defined in the experimental setup (subsection 3.2.1) was used to train the desired classification model, and the results were compared at the end.

Three approaches were employed for the purpose of model definition. The first involved selecting the model using a qualitative metric of popularity, which led to the selection of the VGG16 architecture (SIMONYAN; ZISSERMAN, 2015). The second is based on the model's performance ranking of main ML and DL frameworks, such as *PyTorch*[8] and *Keras*[9], leading to EfficentNet-V2 M (TAN; LE, 2021). The last approach was based on the benchmark reported by Bianco et al. (2018), resulting in the selection of DenseNet201 for its optimal balance between performance and complexity (HUANG et al., 2018).

For each model, a basic experiment was conducted. The model was trained in a version of the original training dataset for 30 epochs and after used to predict the test dataset. The experiments applied the same transfer learning technique for all three models. The results are shown in TABLE 9 in terms of the f1-score macro metric.

TABLE 9 – CLASSIFICATION MODEL SELECTION

Model	**Training time [h]**	**Test F1-Score (macro)**
VGG16	1.0	0.6771
EfficientNet V2 M	6.0	0.6616
DensetNet-201	1.3	**0.7547**

REFERENCE: The author (2023).

Based on the findings, the final selected classification model was the DenseNet201. The transfer learning technique of feature extraction was employed, keeping the original model weights except for the final layer, which is updated. Since the model was pre-trained on the ImageNet-1K dataset, it was necessary to redefine the classifier layer to match the output format of 10 classes in the GC10-DET. The old classifier was then changed to four linear layers, using dropout and ReLu activation functions. In total, the model ends with 2.1 Million trainable parameters and 18.1 Million non-trainable.

The classification model receives 3x224x224 pixels images as input (three grayscale channels in this use case), before going to the model the images are also normalized using the ImageNet mean and standard deviation, as defined by the PyTorch's

[8] https://pytorch.org/vision/stable/models.html
[9] https://keras.io/api/applications/

official implementation[10] of the pre-trained model. In addition, the hyperparameter configuration of the DensetNet201 is given in the TABLE 10.

TABLE 10 – DENSENET201 HYPERPARAMETERS

Hyperparameter	**Value**
Learning rate	0.0003
Image size	3 x 224 x 224
Batch size	128
Optimizer	Adam
Loss function	CE
Max epochs	75

REFERENCE: The author (2023).

The training of the classification model was implemented based on a predetermined maximum number of epochs, ensuring uniform training budgets across all scenarios. In this particular context, the maximum number of epochs was defined as 75. After each epoch, the model is used to evaluate the test dataset, without performing backpropagation to avoid data leaking. Finally, the selected classification metrics will be analyzed over the whole training budget.

3.5.3 Classification metrics

As a consequence of the multiclass nature of the dataset, this study evaluates two approaches regarding classification metrics: macro and weighted. The macro provides equal assessment for all classes, while the weighted analysis puts more importance on the majority class. In this study, the macro will be preferred over the weighted metric, assuming that each defect has the same level of impact on the product. In both cases, F1-Score, Precision, and Recall were computed.

In order to address the stochastic nature of deep learning, the classification model will be evaluated in five runs, each with a different general seed randomly defined. After collecting the metrics results, the mean and standard deviation of the findings will be computed to evaluate the final performance and robustness of the model. Based on the findings, the impact of each scenario will be assessed.

[10] https://pytorch.org/hub/pytorch_vision_densenet/

4 RESULTS

The results chapter aims to present the outcomes of the research. It is worth mentioning that the primary goal of this study is to apply and compare different deep generative models for synthetic image augmentation in a manufacturing quality inspection with limited data, where this inspection is performed by a DL-based classification model. In this regard, the chapter will begin addressing the synthetic image quality analysis. Secondly, the chapter will present the results regarding the classification model, particularly based on the classification metrics discussed in subsection 3.5.3.

4.1 SYNTHETIC IMAGE QUALITY RESULTS

4.1.1 Quantitative evaluation

The present work uses the FID, IS, and SSIM metrics to quantitatively assess the synthetic images in terms of similarity to the original data and diversity between the different classes. These two characteristics together define the quality of the synthetic dataset.

Fréchet Inception Distance (FID)

The FID score primarily focuses on assessing the similarity between the synthetic and the original datasets. In this study, FID is reported in a class-wise manner, with a general value obtained by a weighted average following the approach defined in the subsection 3.5.1

In this scenario, TABLE 11 displays the results for the FID metric on synthetic data generated by each deep generative model. Lower values indicate a higher similarity between the evaluated dataset and the original data. The table also emphasizes the best results overall and per class regarding synthetic data.

The lower limit for the FID score is the original data, with FID equal to 0. Based on the results, DCGAN was the top-performing synthetic model in 7 of the 10 classes, which leads to also being the best model overall with an FID score of 109.51. The Convolutional VAE achieve the second-best performance with a score of 120.31, followed by Conditional DDPM with 134.12, and Unconditional DDPM with 190.05.

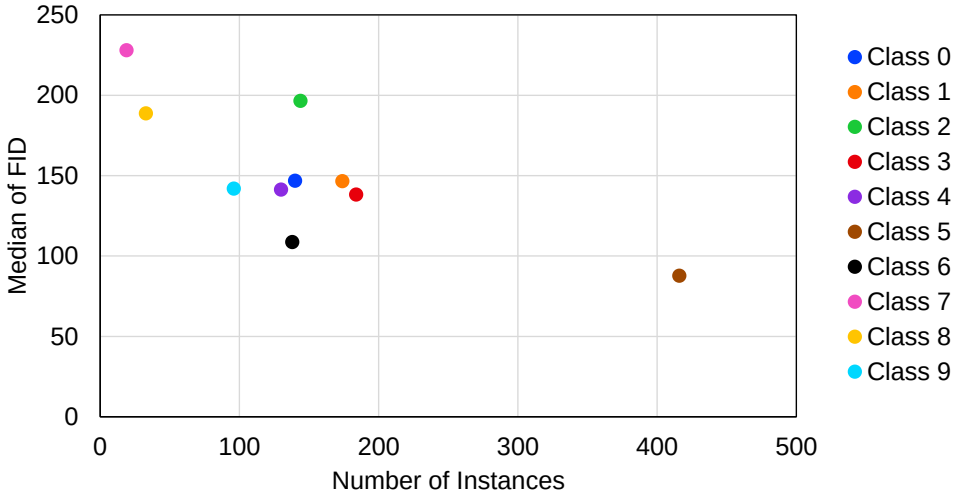
TABLE 11 – FID RESULTS

Classes	Instances	Conv. VAE	DCGAN	Uncon. DDPM	Cond. DDPM
Class 0	140	154.78	**134.84**	259.51	138.75
Class 1	174	144.16	**125.71**	172.63	148.69
Class 2	144	201.57	191.23	291.79	**155.52**
Class 3	184	**83.90**	112.15	164.15	166.46
Class 4	130	136.75	**109.85**	201.04	145.59
Class 5	416	71.60	**57.99**	128.37	103.74
Class 6	138	**94.31**	112.24	158.73	104.96
Class 7	19	243.14	**184.14**	391.64	212.78
Class 8	33	189.04	**156.86**	349.84	188.04
Class 9	96	152.92	**103.43**	220.01	130.79
Weighted mean	-	120.31	**109.51**	190.05	134.12

REFERENCE: The author (2023).

Another important observation from the table is the number of instances per class in the subsets used for FID evaluation, Class 7 appears as the minority class with 19 instances, while class 5 stands out as the majority class with 416 instances. It is worth remarking that these numbers also represent the total training instances per class for the non-conditional models. In this context, FIGURE 17 illustrates the correlation between the median FID value for all models and the number of instances per class.

FIGURE 17 – MEDIAN FID AND NUMBER OF INSTANCES PER CLASS



REFERENCE: The author (2023).

As indicated by the results, generally higher levels of data availability lead to lower FID scores, representing good synthetic data quality. However, an outlier is observed in the results for Class 2, which performs similarly to the minority classes 7 and 8 despite having more instances. This behavior will be discussed later in section 5.2.

Inception Score (IS)

The IS metric evaluates the quality of a synthetic dataset based on both the realism and distinctiveness of the samples, where a higher IS score indicates that the synthetic images have a higher degree of diversity. In the current study, the IS implementation uses the same subsets as the FID metric and follows the definition detailed on subsection 3.5.1. The results obtained for the IS metric are presented in the TABLE 12. The IS is evaluated on 10 splits of the data, and after the mean of the metric is obtained.

TABLE 12 – IS RESULTS

Dataset	**IS mean**
Original	2.07
Convolutional VAE	**1.58**
DCGAN	1.51
Conditional DDPM	1.43
Unconditional DDPM	1.40

REFERENCE: The author (2023).

Considering the IS evaluation in this study, the upper threshold corresponding to the original dataset is the value 2.07. According to the outcomes, the best synthetic model was the Convolutional VAE, with a value of 1.58. The worst performance was obtained by the Unconditional DDPM model, with IS equal to 1.40. The DCGAN and Conditional DDPM models are between these two, with IS of 1.51 and 1.43, respectively.

Structural Similarity Index Measure (SSIM)

The SSIM is a metric that primarily aims to assess the similarity between images, comparing a current instance to a target reference. In this study, the implementation of SSIM follows the definition detailed in subsection 3.5.1. The SSIM values range from 0 to 1, where 0 represents completely opposite images with no similarity, and 1 represents identical images.

The TABLE 13 presents the SSIM results per class and per deep generative model. The overall weighted value is based on the number of instances per class, and all the values of SSIM were acquired after 5 runs.

TABLE 13 – SSIM RESULTS

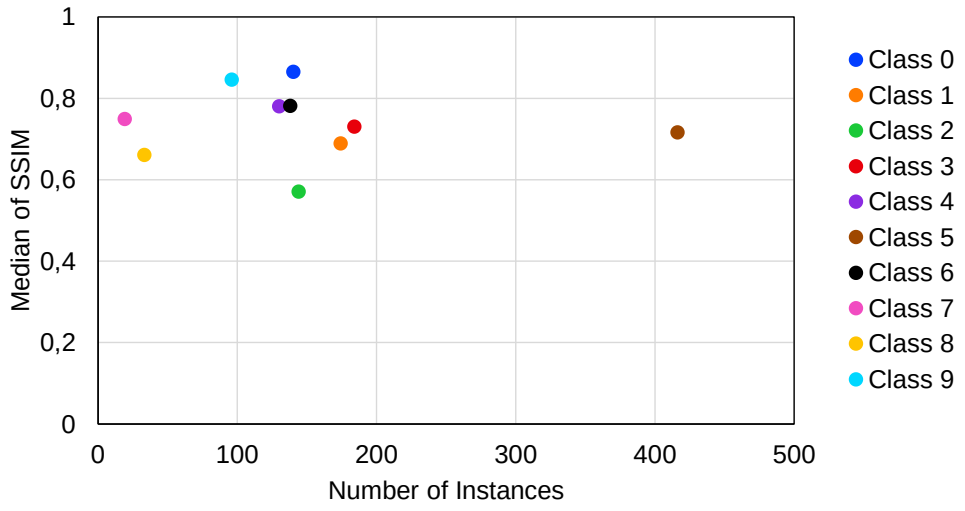
Class	Instances	Conv. VAE	DCGAN	Uncon. DDPM	Cond. DDPM
Class 0	140	**0.8968**	0.8571	0.7596	0.8750
Class 1	174	0.6800	0.6930	0.6864	**0.7198**
Class 2	144	**0.5889**	0.4464	0.5601	0.5823
Class 3	184	0.7360	0.7029	0.7256	**0.7501**
Class 4	130	0.7805	0.7780	0.7817	**0.8068**
Class 5	416	0.7150	0.6919	**0.7391**	0.7183
Class 6	138	0.7879	0.7751	0.7342	**0.8073**
Class 7	19	**0.8314**	0.7442	0.3907	0.7539
Class 8	33	0.6582	0.6646	0.4165	**0.7396**
Class 9	96	**0.8790**	0.8643	0.8035	0.8281
Weighted mean	-	0.7419	0.7128	0.7114	**0.7483**

REFERENCE: The author (2023).

According to the results, the best-performed deep generative model was the Conditional DDPM with a weighted average SSIM of 0.7483, followed closely by the Convolutional VAE with 0.7419. In third place is the DCGAN with 0.7128, and finally the Unconditional DDPM with 0.7114.

All the models performed well in classes 0 and 9, especially the Convolutional VAE with mean SSIM of 0.8968 and 0.8790, respectively. On the contrary, the Unconditional DDPM found difficulty to represents classes 7 and 8, achieving in the first class SSIM of 0.3907 and for the next one 0.4165. As in the FID analysis, the FIGURE 18 shows the correlation between the median SSIM results for all models and the number of instances per class.

FIGURE 18 – MEDIAN SSIM AND NUMBER OF INSTANCES PER CLASS



REFERENCE: The author (2023).

As observed, the SSIM results are not directly related to the number of instances per class. In fact, even the minority classes presented significant outcomes equally or better than classes with more instances. In contrast, the majority class number 5 demonstrated an average performance. As mentioned in the FID analysis, Class 2 represents once again a negative outlier in the results.

Quantitative image quality findings

With the objective of presenting a brief overview of the quantitative evaluation of synthetic image quality metrics, the TABLE 14 provides a summary with the relative and average rank position per model for the three metrics.

TABLE 14 – QUANTITATIVE IMAGE QUALITY RANK

Model	FID rank	IS rank	SSIM rank	Average rank
Convolutional VAE	2	**1**	2	**1.67**
DCGAN	**1**	2	3	2
Conditional DDPM	3	3	**1**	2.33
Unconditional DDPM	4	4	4	4

REFERENCE: The author (2023).

Based on the rank findings, when evaluated only in quantitative terms, the highest quality synthetic data was generated by the Convolutional VAE model. Followed by DCGAN and Conditional DDPM, respectively. The Unconditional DDPM implementation achieved the lowest position in all metrics. The correlation of these findings and the impact on the classification model performance will be analyzed at FIGURE 24.

4.1.2 Qualitative visual analysis

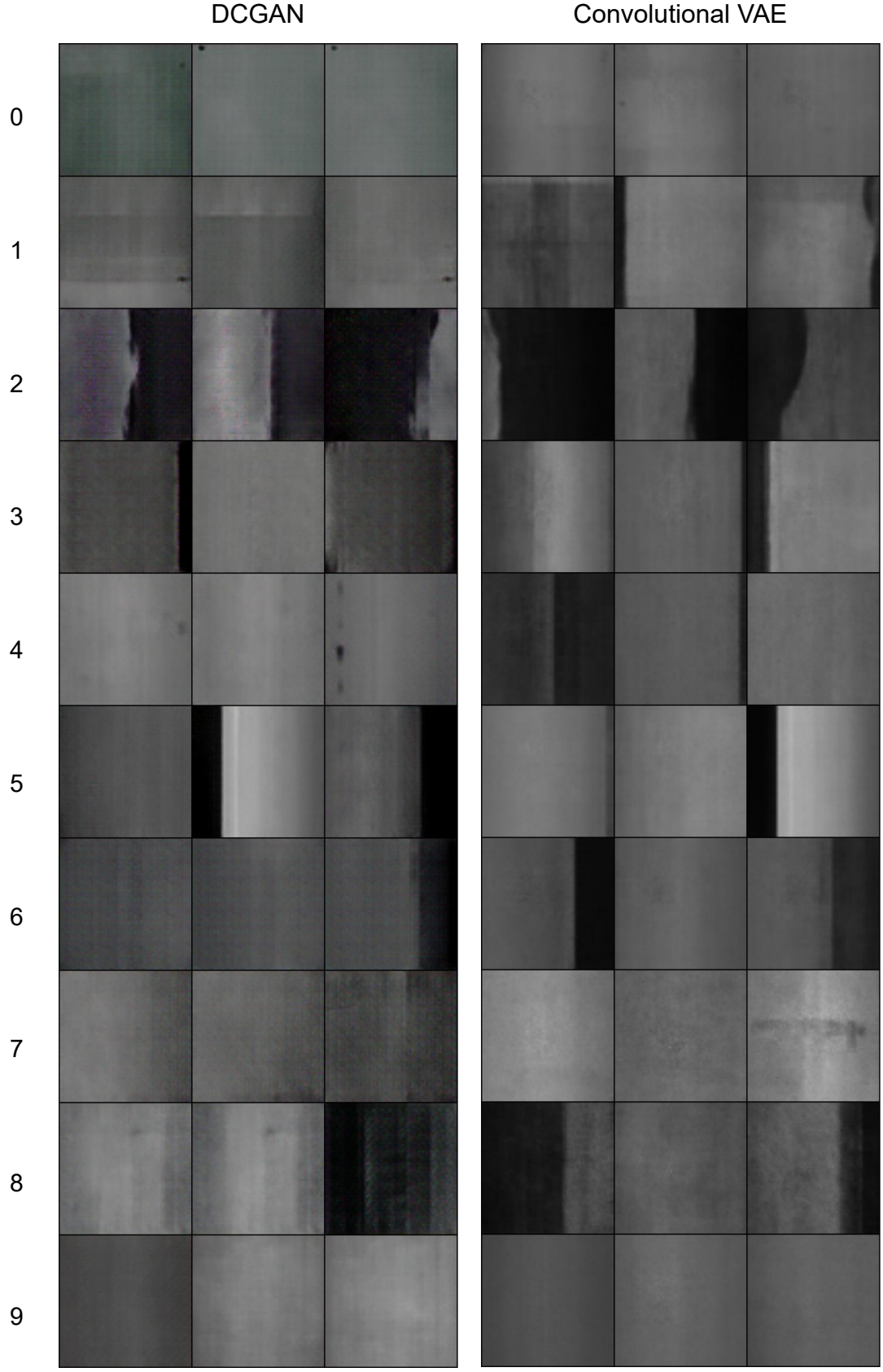
One of the criteria established in the dataset selection process was focused on an easy qualitative assessment of the synthetic data (subsection 3.1.1). This is a widely adopted procedure and could lead to some good insights regarding the model performance.

In this context, the FIGURE 19 and FIGURE 20 display a grid with three synthetic images per class generated by each one of the four deep generative models implemented in this study. The results could be compared to the FIGURE 14 to easily assess how similar the synthetic instances are to the original data from a qualitative analysis.

As indicated by the results, the DCGAN model achieved the best visual aspect, especially by representing image details such as the small features on classes 0 (punching hole) and 4 (oil spot). In contrast, the Unconditional DDPM model produced the lowest visual quality, particularly for classes 7 (rolled pit) and 8 (crease) with a final result close to pure noise. For the other classes, the model generated basic gray images without clear features to distinguish among them.

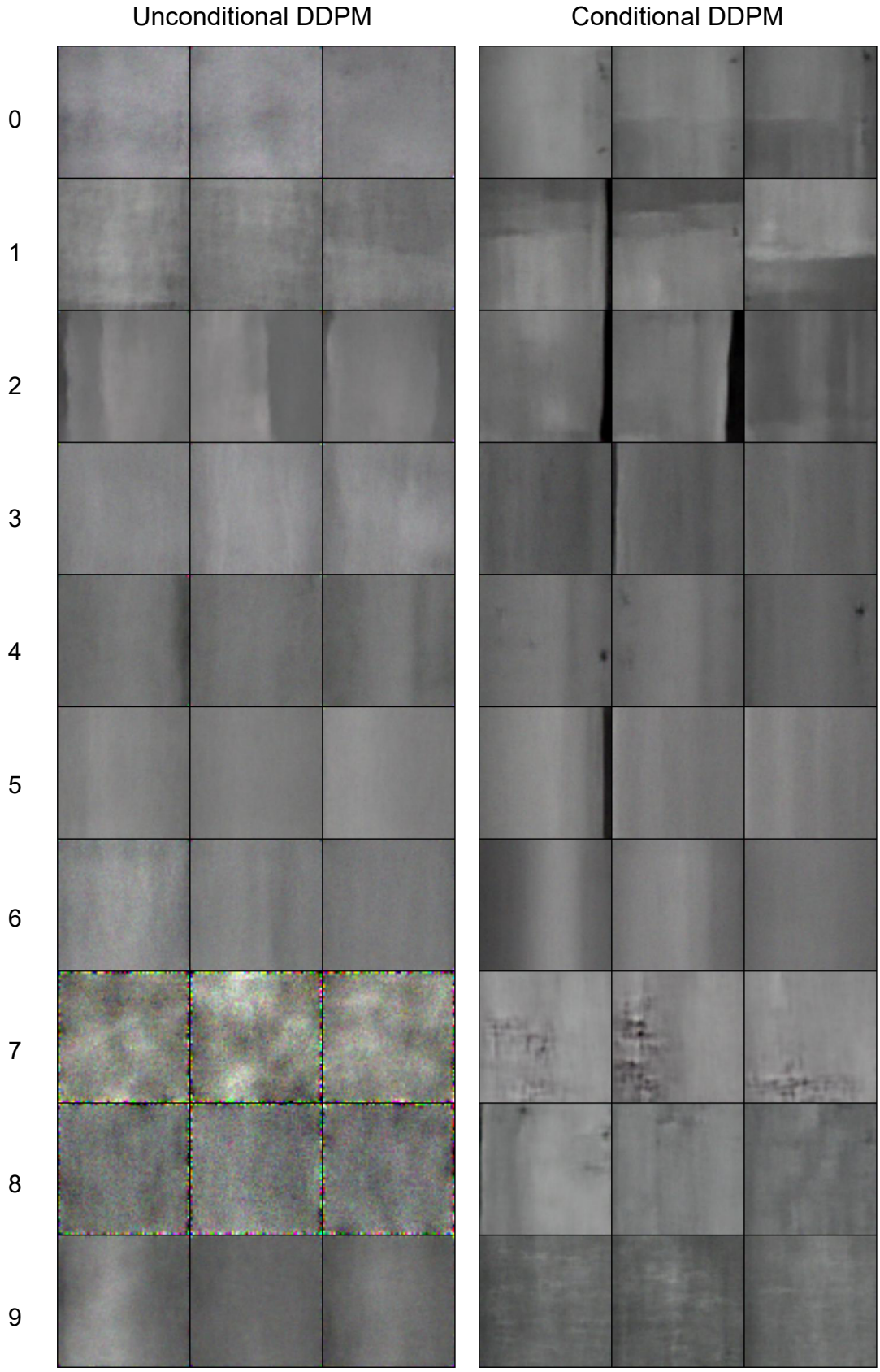
Moreover, Conditional DDPM showed a clear improvement in visual quality compared to the Unconditional DDPM model. However, this new implementation still faced challenges with limited data in classes 7 (rolled pit) and 8 (crease). Finally, Convolutional VAE demonstrated the potential to represent significant image features, such as the black background in some instances. In contrast, this model failed to synthesize details

FIGURE 19 – DCGAN AND CONVOLUTIONAL VAE DATA



REFERENCE: The author (2023).

FIGURE 20 – UNCONDITIONAL AND CONDITIONAL DDPM DATA



REFERENCE: The author (2023).

4.1.3 Synthetic image quality key findings

The key findings of the synthetic image quality evaluation, in both quantitative and qualitative terms, are summarized in the topics below:

- In general, a higher number of instances per class leads to better FID performance. The data availability influence is also noticed in the visual quality inspection. In this context, the deep generative models found difficulties in synthesizing the minority classes.

- Among the image quality metrics, the FID represented the best correlation with the qualitative evaluation of the images. Especially in terms of features and details representation. Additionally, IS and SSIM presented some inconsistencies when assessed in front of the qualitative analysis.

- The DCGAN was the model that better represent the key features of the classes. When evaluating classes 0 and 4, for example, this model was the best to synthesize tiny details. Additionally, this characteristic leads DCGAN to achieve the top-1 FID of the analysis.

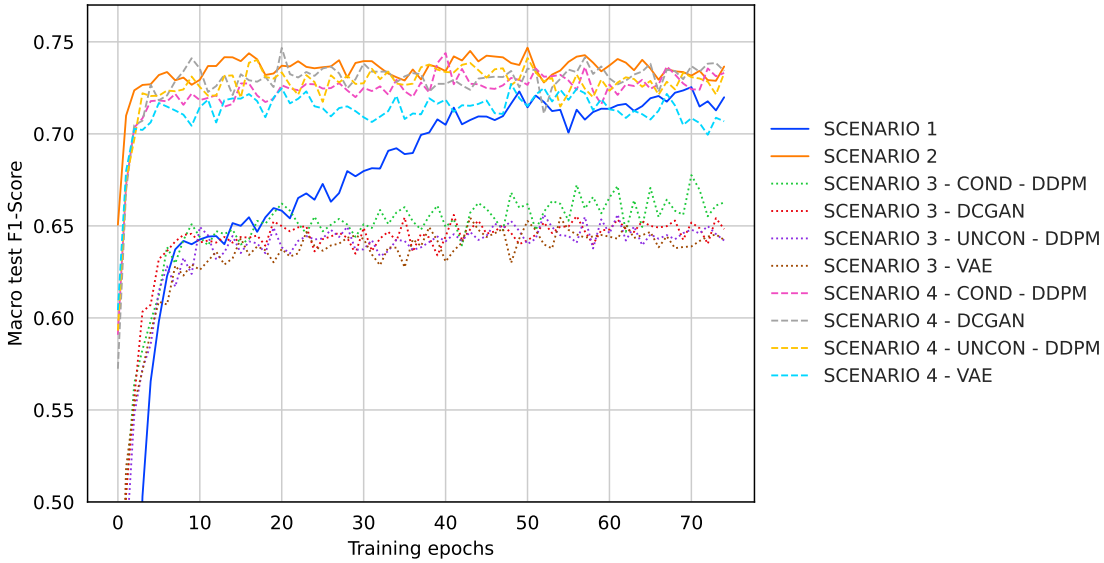
- The Unconditional DDPM model exhibited poor data quality in both quantitative and qualitative aspects. Especially in limited data classes, this model failed to generate meaningful instances.

4.2 CLASSIFICATION MODEL RESULTS

4.2.1 Initial experimental setup

This subsection presents the performance results of the initial experimental setup (subsection 3.2.1). The FIGURE 21 shows the general overview of the macro F1-Score results over the different scenarios, a zoom was applied in the y-axis for better visualization purposes (see appendix for original). According to the findings, three different regions of results can be highlighted. Among the top-performed configurations are Scenarios 2 and 4, i.e. conventional augmented and synthetic plus conventional augmented data. At the bottom are the Scenario 3 definitions for all the deep generative models. In between, the original training data is represented by Scenario 1.

FIGURE 21 – TEST MACRO F1-SCORE FOR INITIAL SCENARIOS



REFERENCE: The author (2023).

The TABLE 15 summarizes the best macro F1-score value for the test dataset across 75 training epochs for each of the different cases presented in the FIGURE 21. The mean and standard deviation of the macro F1-Score are reported along with the metrics of precision and recall, at the same epoch.

TABLE 15 – BEST TEST MACRO RESULTS FOR INITIAL SCENARIOS

Scenario	F1-Score	Precision	Recall	Epoch	Rank
1	0.7254 ± 0.0108	**0.7890 ± 0.0173**	0.7030 ± 0.0106	70	6
2	**0.7468 ± 0.0043**	0.7612 ± 0.0086	0.7423 ± 0.0064	50	**1**
3 - COND - DDPM	0.6776 ± 0.0150	0.7467 ± 0.0152	0.6681 ± 0.0134	70	7
3 - DCGAN	0.6560 ± 0.0144	0.7218 ± 0.0369	0.6499 ± 0.0139	41	9
3 - UNCON - DDPM	0.6562 ± 0.0103	0.7250 ± 0.0540	0.6482 ± 0.0091	61	8
3 - VAE	0.6545 ± 0.0097	0.6849 ± 0.0231	0.6453 ± 0.0059	43	10
4 - COND - DDPM	0.7439 ± 0.0128	0.7520 ± 0.0166	**0.7447 ± 0.0143**	40	3
4 - DCGAN	0.7467 ± 0.0096	0.7684 ± 0.0073	0.7398 ± 0.0141	**20**	2
4 - UNCON - DDPM	0.7412 ± 0.0153	0.7503 ± 0.0236	0.7418 ± 0.0111	50	4
4 - VAE	0.7271 ± 0.0084	0.7383 ± 0.0139	0.7275 ± 0.0061	48	5

REFERENCE: The author (2023).

According to the outcomes, the best result was obtained in Scenario 2 with a macro F1-Score of 0.7468, followed closely by Scenario 4 - DCGAN with 0.7467. The difference is still inside the standard deviation of both results, the same for precision and recall metrics. Other Scenarios 4 outcomes for macro F1-Score are in the range of 0.7271 to 0.7439. Scenario 3 - VAE has the lowest performance among the cases, with a macro F1-Score of 0.6545. Finally, the original data has a macro F1-Score of 0.7254.

By evaluating the outcomes of the image quality metrics (see subsection 4.1.1) together with the collected results for the classification model performance, the DCGAN model was chosen as the most promising approach to be tested in two further scenarios. It is worth noticing that the key metric for the classification performance was defined in the subsection 3.5.3 as the macro F1-Score, especially due to the equal level of importance among the defect classes. The weighted metrics are in the appendix.

4.2.2 Pre-trained model with synthetic data

Based on the previous findings, one extra technique was applied with the goal of improving the classification results using the most promising synthetic data. This approach is outlined in the subsection 3.2.2 as Scenario 5, and is based on a pre-trained model on Scenario 3 - DCGAN dataset, followed by a fine-tuning approach in Scenario 2 (conventional augmented). The pre-trained model was selected as the best macro F1-Score result for the final epoch among the 5 runs of Scenario 3 - DCGAN. The TABLE 16 presents general information about the model.

TABLE 16 – PRE-TRAINED MODEL ON SCENARIO 3 DCGAN

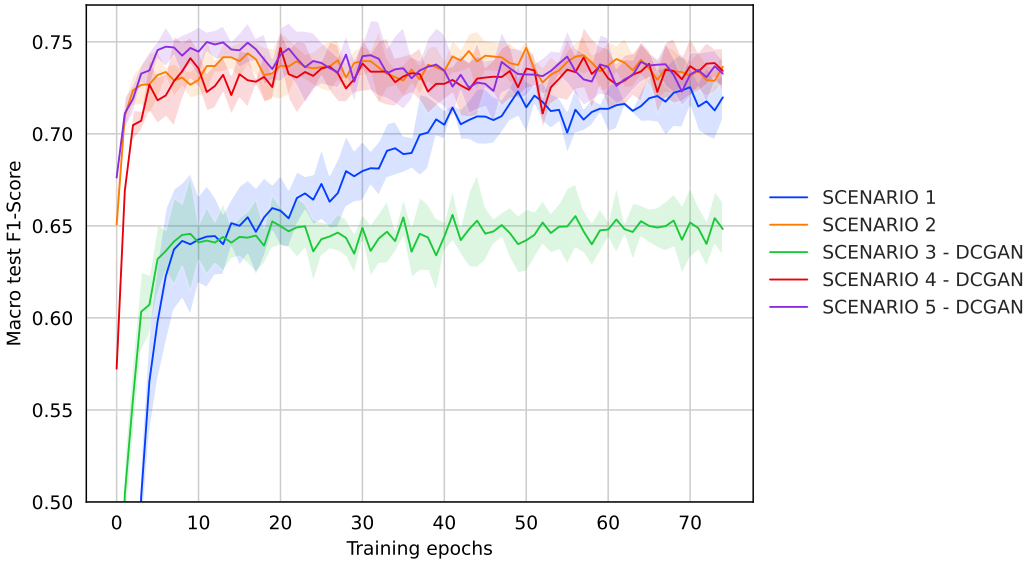
Training Time	**M. F1-Score**	**W. F1-Score**	**Non Trainable P.**	**Trainable P.**
50.9 min	0.6683	0.7703	18.1 M	2.1 M

REFERENCE: The author (2023).

The FIGURE 22 and the TABLE 17 report a comparison, in terms of macro F1-Score, between the traditional techniques and the different approaches regarding synthetic data generated by the DCGAN model. Some of these approaches are already mentioned in the subsection 5.1.2, now they are repeated for comparison purposes. As before, a zoom was applied on the y-axis of all the charts in this subsection for better visualization purposes. The original chart is in the appendix.

From the analysis of the findings, the new Scenario 5 - DCGAN definition demonstrated the highest performance with a macro F1-Score of 0.7499, closely followed by the Scenario 2 data with a value of 0.7468. The worst performance was presented by Scenario 3 - DCGAN, with 0.6559. Additionally, it is important to mention the number of epochs to achieve the best score, where Scenario 5 - DCGAN took 11 epochs against 50 from Scenario 2.

FIGURE 22 – TEST MACRO F1-SCORE IN 5 RUNS



REFERENCE: The author (2023).

TABLE 17 – BEST TEST MACRO RESULTS IN 5 RUNS

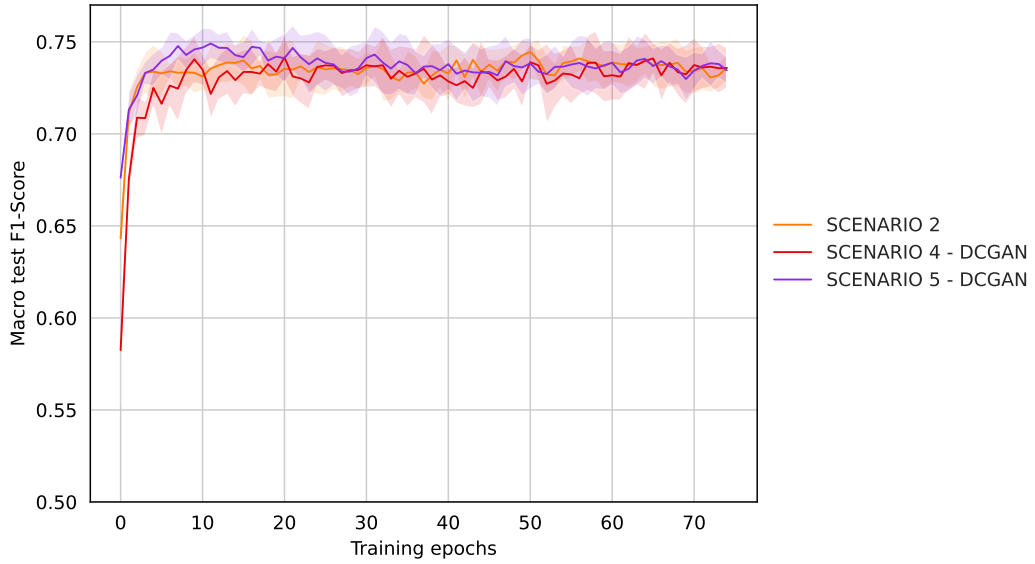
Scenario	F1-Score	Precision	Recall	Epoch	Rank
1	0.7254 ± 0.0108	**0.7880 ± 0.0173**	0.7030 ± 0.0106	70	4
2	0.7468 ± 0.0043	0.7612 ± 0.0086	0.7423 ± 0.0064	50	2
3 - DCGAN	0.6559 ± 0.0144	0.7218 ± 0.0369	0.6499 ± 0.0139	41	5
4 - DCGAN	0.7467 ± 0.0096	0.7684 ± 0.0073	0.7398 ± 0.0141	20	3
5 - DCGAN	**0.7499 ± 0.0056**	0.7616 ± 0.0081	**0.7509 ± 0.0059**	**11**	**1**

REFERENCE: The author (2023).

Despite the superior performance of Scenario 5 - DCGAN, synthetic and conventional techniques have only a slight difference in performance. In this context, 5 additional runs were conducted to explore the robustness of the best three scenarios so far. Based on the previous results, besides Scenario 5 covered in this subsection, the other selected scenarios were Scenario 2 (conventional augmentation) and Scenario 4 (combination of synthetic and conventional augmentation).

The FIGURE 23 shows the results curve for macro F1-Score evaluated along the 10 different runs, the best results over the training epochs were detailed in TABLE 18. By evaluating the outcomes, Scenario 5 - DCGAN figures again as the top-performing approach, with an F1-Score of 0.7491. When comparing the performance between 5 and 10 runs, Scenario 5 - DCGAN presents a delta of -0.0008 in the macro F1-Score, against a delta of -0.0022 for Scenario 2 and -0.0051 for Scenario 4.

FIGURE 23 – TEST MACRO F1-SCORE IN 10 RUNS



REFERENCE: The author (2023).

TABLE 18 – BEST TEST MACRO RESULTS IN 10 RUNS

Scenario	F1-Score	Precision	Recall	Epoch	Rank
2	0.7446 ± 0.0054	0.7602 ± 0.0098	0.7404 ± 0.0082	50	2
4 - DCGAN	0.7416 ± 0.0126	**0.7621 ± 0.0094**	0.7355 ± 0.0153	20	3
5 - DCGAN	**0.7491 ± 0.0084**	0.7589 ± 0.0118	0.7506 ± 0.0074	**11**	**1**

REFERENCE: The author (2023).

4.2.3 Image resizing pipeline

The second proposed technique to evaluate DCGAN synthetic data and conventional approaches is related to the Scenario 6 resizing pipeline. This scenario uses validation and test datasets, applying an early stopping technique based on patience set for 10 epochs, and a small classifier model (MobileNet-V2) due to the data simplicity.

Besides, Scenario 6 is a new evaluation of the already defined five other scenarios. In this context, in order to clarify the analysis and the understanding of the results, the relationship will be indicated using the notation *6.Scenario*. For example, *Scenario 6.1* corresponds to Scenario 1 evaluated in the Scenario 6 pipeline.

The TABLE 19 shows the mean and standard deviation results of macro metrics for the test dataset after 5 runs. From the analysis of the findings, Scenario 4 - DCGAN achieved the best macro F1-Score value with 0.6568, followed by Scenario 5 - DCGAN with 0.6495. The third result was the conventional augmented dataset in Scenario 2,

with a value of 0.6294. Moreover, the lowest performance was exhibited by Scenario 1 with a macro F1-Score of 0.5763.

TABLE 19 – BEST TEST MACRO RESULTS FOR SCENARIO 6

Scenario	F1-Score	Precision	Recall	Rank
6.1	0.5763 ± 0.0165	0.6082 ± 0.0430	0.5725 ± 0.0132	5
6.2	0.6294 ± 0.0231	0.6413 ± 0.0193	0.6353 ± 0.0249	3
6.3 - DCGAN	0.5886 ± 0.0100	0.6180 ± 0.0101	0.5844 ± 0.0131	4
6.4 - DCGAN	**0.6568 ± 0.0162**	**0.6705 ± 0.0226**	0.6641 ± 0.0138	1
6.5 - DCGAN	0.6495 ± 0.0177	0.6510 ± 0.0122	**0.6650 ± 0.0226**	2

REFERENCE: The author (2023).

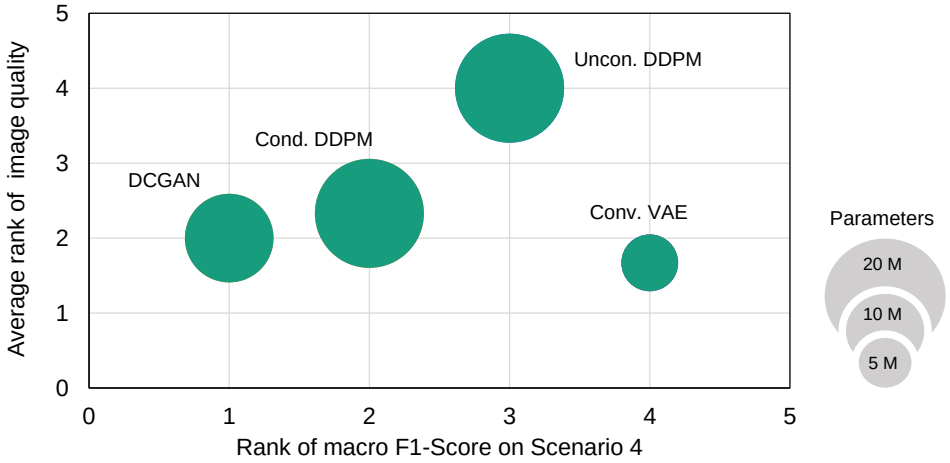
As indicated by the results, the overall performance of the model decreases when compared to the previously defined scenarios one to five. However, the gap between synthetic augmentation and conventional techniques is more evident.

4.2.4 Classification model key findings

The key findings for classification model evaluation are summarized below:

• The FIGURE 24 demonstrates that, except for the Convolutional VAE as an outlier, there is a direct correlation between synthetic image quality quantitative metrics and classification model performance. However, model size and complexity are not necessarily related to better results in the task.

FIGURE 24 – MODEL PERFORMANCE AND RANK OF IMAGE QUALITY



REFERENCE: The author (2023).

• Evaluating the macro F1-Score results for the initial analysis, Scenario 4 - DCGAN was the top-performing deep generative model implementation. In Scenario 5, this architecture presented even better results after applying the proposed pre-trained model. In this context, the combination of synthetic and conventional approaches to upsampling the original dataset demonstrates significant potential to increase the classification performance.

• The best results using synthetic data are slightly better than conventional techniques. However, after balancing the information compressions for both augmentation approaches using the Scenario 6 image resizing pipeline, it was evident that the combination of synthetic augmentation with conventional techniques demonstrated a significant improvement compared to using only synthetic or conventional methods.

5 DISCUSSION

This chapter's goal is to establish connections between the outcomes and the main objective of this study, which is to employ and compare various deep generative models for synthetic image augmentation in a challenging manufacturing use case characterized by limited data. This inspection is conducted by a DL-based classification model. For this purpose, the chapter will begin with a comparison of deep generative models in terms of synthetic image quality metrics and classification model performance. Additionally, it will cover the strengths and weaknesses of the models, as well as the effort to implement each of them.

5.1 COMPARATIVE ANALYSIS OF RESULTS

5.1.1 Synthetic image quality

Taking into account the analysis of the synthetic data, the DCGAN can be considered the most successful deep generative model implemented. Especially because consistently demonstrates good results in terms of synthetic data quality in both quantitative and qualitative approaches. The DCGAN model ranked second in the overall image quality metrics (TABLE 14). However, was able to synthesize the key features in almost all the classes in a qualitative analysis. In this context, see section 5.2 for a more detailed discussion.

The Conditional DDPM model also presented significant findings. Although it ranked third in the average quantitative evaluation (TABLE 14), it exhibited potential in visual quality evaluation. Moreover, it is evident that the implementation of a conditional approach brought advances when compared to the Unconditional DDPM.

On the contrary, the Convolutional VAE ranked in first position in the overall image quality metrics (TABLE 14). However, when performing analysis in comparison with the qualitative assessment this model presents some limitations. The visual evaluation of synthetic data for the Convolutional VAE reveals that the model is basically generating gray images and overlooking some essential features for the most part of the synthetic data. These outcomes highlight the importance of a cross-evaluation analysis with the qualitative approach.

Additionally, the Unconditional DDPM model underperformed all models in all classes and metrics. In this particular model, classes 7 and 8 presented the most significant difficulties. In fact, based on the visual analysis of the outcomes it is evident that this model almost just synthesized noise for these two classes. The unsatisfactory performance can be attributed to the relationship between the complexity of the model and the limited amount of available data.

Except for the Unconditional DDPM model, it can be concluded that all the other architectures successfully perform the synthetic augmentation technique in the GC10-DET dataset. The Convolutional VAE model presented some instability and outliers results. The Conditional DDPM model is a potential approach but had the drawback of poor FID performance. On the other hand, DCGAN reported the most robust performance among the evaluated quantitative and qualitative results.

5.1.2 Initial experimental setup

Initially, it is important to mention that achieving state-of-art performance in the classifier was not the primary goal of this study. However, this work aims to apply and compare different deep generative models for synthetic image augmentation in order to perform a DL-model-based visual quality inspection in a limited data use case related to manufacturing operations. In this context, the following discussion is addressed to this objective.

With respect to the use of synthetic data, the best overall scenario was number 4 (random selection of synthetic and conventional augmented data). Both DCGAN and Conditional DDPM performance reflects the findings related to overall quantitative and qualitative synthetic image quality evaluation, showcasing their superiority also in the impact on the classification results (FIGURE 24). In this context, the highest macro F1-Score was achieved by the DCGAN architecture.

Concerning Scenario 3, which assesses the addition of only synthetic data in the raw training dataset, the results are below the average for all the deep generative models. One possible root cause for these unsatisfying results is the original synthetic data resolution of 64x64 pixels. The classification model learns from a low-density information dataset and is tested in a high-density domain, leading to poor outcomes.

Besides, it is important to cover the relationship between quantitative image

quality metrics and the classification model performance (FIGURE 24). In this case, the findings of the IS and SSIM are not directly related to an improvement in the model metrics. The Convolutional VAE which achieved significant results in these metrics, was the worst-performing model in scenarios 4 and 3, even below Unconditional DDPM. In contrast, the DCGAN architecture presented lower results in the SSIM metric but better outcomes in the classification task.

Summarizing the comparison of the synthetic augmentation approaches in this study, deep generative models that properly represented key features of images led to better outcomes in terms of classification metrics. This statement is particularly supported by the results of DCGAN and Conditional DDPM.

5.1.3 Synthetic and conventional augmentation

This section will explore scenarios 5 and 6 covered in the study execution, especially in terms of comparison between the best-performing deep generative model (DCGAN) and conventional augmentation techniques.

Scenarios 1 and 2 (TABLE 15) using conventional approaches presented a high performance in macro metrics, indicating an effective design of the augmented pipeline. However, with the stability of a pre-trained model, Scenario 5 (TABLE 17) demonstrated that synthetic augmentation techniques could outperform conventional methods in classification metrics and training time, despite a small difference.

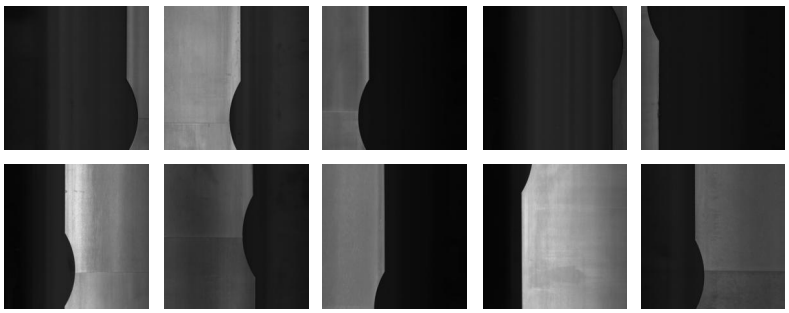
Regarding Scenario 6 analysis (TABLE 19), the overall performance decreased since the model learned from less-resolution images. However, the gap between the synthetic and conventional approaches was more representative. Scenario 6.4 was the best-performing approach outperforming the conventional augmented technique by 4.26% in the macro F1-Score metric. Another positive aspect was Scenario 6.3, which was better than the raw training data scenario in all macro metrics. These support the Scenario 5 discussion, since when balancing information among augmentation approaches the synthetic data expressive outperformed conventional techniques.

5.2 STRENGTHS AND WEAKNESSES

Based on the visual qualitative analysis in addition to the findings of both FIGURE 17 and FIGURE 18, it is possible to derive statements regarding the strengths

and weaknesses of deep generative models for synthetic image quality. In this context, all the models performed below their average in FID and SSIM metrics for Class 2. The FIGURE 25 presents 10 randomly defined images for this class. As observed, there is no well-defined pattern in the data, and this characteristic could introduce a challenge for deep generative models. When evaluating limited data, it becomes apparent that classes 7 and 8 have a significant impact on the quality of synthetic images across different models. In this analysis, the lack of pattern in the class distribution and data sparsity configures it as a weakness for the generative models.

FIGURE 25 – RANDOM SAMPLES OF CLASS 2



REFERENCE: The author (2023).

Regarding Class 2, the best results were achieved by the Convolutional VAE and Conditional DDPM model. These two models presented the ability to generate features that are predominant in the images. The simplicity of Convolutional VAE and the conditional approach in DDPM implementation has a key role in this characteristic.

The better results in FID terms were achieved by Class 5, emphasizing the correlation between the number of instances and FID performance (FIGURE 17). Besides the data availability, all the images in the class follow a well-defined pattern. Concerning the SSIM, the best results were in classes 0 and 9 (FIGURE 18). Nevertheless, both Convolutional VAE and Unconditional DDPM easily fail to synthesize image details, which represents a significant drawback for these architectures (subsection 5.1.2).

Additionally, based on the classification model outcomes it is possible to formulate statements about the strengths and weaknesses of applying synthetic data in a DL-model-based classification task. In this instance, the analysis of the four initial experimental scenarios in addition to Scenario 5 exposes the importance of combining conventional and synthetic approaches. This advantageous aspect of synthetic data implementations proves beneficial to fill the gaps in the data distribution. Moreover, it

confirms the synthetic data potential to improve a classification task performance.

In this study, one limitation to outperforming conventional techniques with synthetic data was the original image size. In this regard, Scenario 6 addresses this weakness of the deep generative model and emphasizes the importance of generating synthetic data at least in the original resolution or in the minimum resolution required for the classification model. This approach avoids issues related to information density in the generated instances.

5.3 IMPLEMENTATION EFFORT BEHIND THE MODELS

The effort to implement new industrial AI technologies in manufacturing is a crucial parameter. Especially because this domain often requires a balance between efficiency and state-of-the-art results. In this context, the DCGAN model was the easiest deep generative model to implement. A major drawback of GAN-based architectures is the training instability. However, the selected DCGAN model rarely collapsed during the training step, probably due to staying close to the original whitepaper definition. Additionally, this model type is the most common approach for image synthesis. This aspect is very positive, especially related to the number of open-source applications, diversity of use cases, and free tutorial availability.

Each of the other model types has its advantages and disadvantages. The Convolutional VAE model has the characteristic of being a simple architecture, this makes the model lighter and more stable to train. However, this simplicity also limits its ability to handle more complex datasets. Consequently, the majority of open-source materials for VAE implementations use basic datasets, raising barriers to the use of the available models in most parts of manufacturing applications. In this study, the CNN inside the VAE model was defined based on multiple approaches and validated by comparison of different models, leading to a time-consuming development.

On the contrary, both DDPM architectures are close to the state-of-the-art in terms of image synthetization. This characteristic rise certain challenges, including the small among of open-source materials about the models and the high computational and theoretical complexity. The computational complexity turns the model more sensible to small-data problems and increases the training time, while the theoretical side makes it difficult to perform improvements and troubleshoot.

6 CONCLUSION AND OUTLOOK

The primary goal of this chapter is to present a conclusion for the study. In this regard, it begins by summarizing the essential findings and later will provide an outlook on potential research to further advance the field of data synthetization.

6.1 CONCLUSION

Considering the context of *Industry 4.0* the application of Artificial Intelligence (AI), especially Machine Learning (ML) and Deep Learning (DL) techniques, represent powerful tools for manufacturing companies towards data-driven production. However, to develop a successful DL project it is necessary to have data in both quantity and quality. This requirement usually represents a challenge for manufacturing environments. Additionally, conventional augmentation solutions often require expert supervision to be properly performed, being a costly activity. Considering this circumstance, image data synthetization via deep generative models is a new potential solution for this problem, emerging the necessity for further research and experimental analysis.

Therefore, the main objective of this study was to apply and compare different deep generative models for synthetic image augmentation in order to perform a DL-model-based visual quality inspection in a complex manufacturing use case with limited data. Based on the defined objective, Convolutional VAE, DCGAN, Unconditional DDPM, and Conditional DDPM models were evaluated over different scenarios. In the initial experimental setup, the classification model was trained using original data and different augmentation techniques, like conventional, synthetic, and a combination of both approaches. This study also evaluates two further scenarios for the best-performing model. In the first scenario, a pre-trained framework is employed on a synthetic dataset, while the second approach applies a resizing pipeline to balance possible information compressions in both augmentation techniques, i. e. conventional and synthetic.

According to the outcomes, DCGAN was the best-performed deep generative model in terms of synthetic image quality and positive impact on the classification task. In regards to the classification metrics, the combination of synthetic and conventional techniques outperformed traditional approaches by a minor margin in scenarios of

different image sizes. However, after ensuring that both original and synthetic data have the same resolution, the performance gain for the combination of synthetic and conventional techniques becomes expressive. Thus, deep generative models have been applied to synthesize new data instances for the training phase of an image classification model in a complex manufacturing use case with data sparsity, enabling further comparison across different architectures and techniques. Accordingly, the underlying research objective of this study can be accomplished with positive results.

6.2 OUTLOOK

The ideas and suggestions presented in this outlook section were derived from the discussion of the synthetic image quality and classification task metrics outcomes. Additionally, this section includes aspects of the study's development that were not investigated due to time limitations.

As discussed in subsection 5.1.3, there is a potential to increase the performance of the model trained on synthetic data by generating data in higher resolutions. Therefore, future work should ensure that the synthetic images are generated at least at the minimum resolution required by the classification model. Moreover, future research can implement robust hyperparameter optimization techniques to achieve better results in the synthetic image quality. The study could focus on the most promising model type and deep dive into the implementation of model-based architectures.

Furthermore, the implementation of deep generative models in multiple use cases to evaluate synthetic image quality and classification task performance could lead to valuable insights. In future research, a benchmark could be established leading to a decision-support framework. Besides, the study could address other DL tasks, for example, object detection and image segmentation. These are common implementations in an industrial scenario where deep generative models could also represent improvements.

Finally, there is further potential in directing the synthetic instances to fulfill the gaps in the original data distribution. Future work could focus on a pipeline that ensures this optimal implementation of deep generative models. In this context, addressing the samples could represent an intelligent manner of combining the synthetic data with conventional approaches.

REFERENCES

ALPAYDIN, E. **Introduction to Machine Learning**. 2. ed. London, England: MIT Press, Jan. 2010. (Adaptive Computation and Machine Learning Series). Cit. on p. 22.

BARRATT, S.; SHARMA, R. **A Note on the Inception Score**. [S.l.: s.n.], 2018. arXiv: 1801.01973 [stat.ML]. Cit. on p. 45.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation Learning: A Review and New Perspectives. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 35, n. 8, p. 1798–1828, 2013. Cit. on p. 33.

BEYERER, J.; LEÓN, F. P.; FRESE, C. **Machine Vision**. [S.l.]: Springer Berlin Heidelberg, 2016. Available from: https://doi.org/10.1007/978-3-662-47794-6. Cit. on p. 21.

BIANCO, S. et al. Benchmark Analysis of Representative Deep Neural Network Architectures. **IEEE Access**, v. 6, p. 64270–64277, 2018. Cit. on p. 60.

BISHOP, C. **Pattern Recognition and Machine Learning**. 1. ed. New York, NY: Springer, Aug. 2006. (Information Science and Statistics). Cit. on pp. 23, 97.

BOND-TAYLOR, S. et al. Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Institute of Electrical and Electronics Engineers (IEEE), v. 44, n. 11, p. 7327–7347, Nov. 2022. Available from: https://doi.org/10.1109%2Ftpami.2021.3116668. Cit. on pp. 32, 54.

BORJI, A. **Pros and Cons of GAN Evaluation Measures**. [S.l.: s.n.], 2018. arXiv: 1802.03446 [cs.CV]. Cit. on pp. 45, 46.

BUDUMA, N.; LOCASCIO, N. **Fundamentals of deep learning**. Sebastopol, CA: O'Reilly Media, July 2017. Cit. on p. 27.

BUERHOP-LUTZ, C. et al. **A Benchmark for Visual Identification of Defective Solar Cells in Electroluminescence Imagery**. In: EUROPEAN PV Solar Energy Conference and Exhibition (EU PVSEC). [S.l.: s.n.], 2018. Cit. on p. 30.

CANZIANI, A.; LECUN, Y. **NYU Deep Learning, Spring 2020**. [S.l.: s.n.], 2020. Available from: https://atcold.github.io/pytorch-Deep-Learning/. Cit. on pp. 28, 34.

CHATZIAGAPI, A. et al. **Data Augmentation Using GANs for Speech Emotion Recognition**. In: INTERSPEECH 2019. [S.l.]: ISCA, Sept. 2019. Available from: https://doi.org/10.21437/interspeech.2019-2561. Cit. on p. 17.

CHEN, T. et al. Machine Learning in Manufacturing towards Industry 4.0: From lsquo;For Nowrsquo; to lsquo;Four-Knowrsquo; **Applied Sciences**, v. 13, n. 3, 2023. ISSN 2076-3417. Available from: https://www.mdpi.com/2076-3417/13/3/1903. Cit. on p. 16.

CRESWELL, A. et al. Generative Adversarial Networks: An Overview. **IEEE Signal Processing Magazine**, v. 35, n. 1, p. 53–65, 2018. Cit. on p. 36.

CUNNINGHAM, P.; CORD, M.; DELANY, S. J. **Supervised Learning**. In: MACHINE Learning Techniques for Multimedia. [S.l.]: Springer Berlin Heidelberg, 2008. Available from: https://doi.org/10.1007/978-3-540-75171-7_2. Cit. on p. 23.

DELOITTE. AI Enablement on the Way to Smart Manufacturing. **Deloitte Survey on AI Adoption in Manufacturing**, 2020. Cit. on p. 16.

DENG, J. et al. **ImageNet: A large-scale hierarchical image database**. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2009. P. 248–255. Cit. on p. 45.

FERNÁNDEZ, A. et al. **Learning from Imbalanced Data Sets**. In: CAMBRIDGE International Law Journal. [S.l.: s.n.], 2018. Cit. on pp. 25, 26.

FOSTER, D. **Generative deep learning**. 1st. Sebastopol, CA: O'Reilly Media, July 2019. Cit. on pp. 30–33, 35, 37.

_____._____. 2nd. Sebastopol, CA: O'Reilly Media, May 2023. Cit. on p. 32.

FR'ECHET, M. Sur la distance de deux lois de probabilit'e. **Comptes rendus de l'Acad'emie des Sciences**, v. 244, p. 689–692, 1957. Cit. on p. 46.

GERHARD, H. E.; WICHMANN, F. A.; BETHGE, M. How Sensitive Is the Human Visual System to the Local Statistics of Natural Images? Ed. by Konrad P. Kording. **PLoS Computational Biology**, Public Library of Science (PLoS), v. 9, n. 1, e1002873, Jan. 2013. Available from: https://doi.org/10.1371/journal.pcbi.1002873. Cit. on p. 44.

GERON, A. **Hands-on machine learning with scikit-learn, keras, and TensorFlow**. 2. ed. Sebastopol, CA: O'Reilly Media, Oct. 2019. Cit. on pp. 23, 24, 26, 29, 43, 94.

GM, H. et al. A comprehensive survey and analysis of generative models in machine learning. **Computer Science Review**, v. 38, p. 100285, 2020. ISSN 1574-0137. Available from: https://www.sciencedirect.com/science/article/pii/S1574013720303853. Cit. on p. 54.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. Cit. on pp. 17, 26, 27, 29, 33.

GOODFELLOW, I. J. et al. **Generative Adversarial Networks**. [S.l.]: arXiv, 2014. Available from: https://arxiv.org/abs/1406.2661. Cit. on pp. 35, 36.

GOOGLE. Google Cloud Industries: Artificial Intelligence Acceleration among Manufacturers. **Google Cloud Survey**, 2021. Cit. on pp. 16, 21.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The elements of statistical learning: data mining, inference, and prediction**. 2. ed. [S.l.]: Springer, 2009. Cit. on p. 23.

HAYNES, D.; CORNS, S.; VENAYAGAMOORTHY, G. K. **An Exponential Moving Average algorithm**. In: 2012 IEEE Congress on Evolutionary Computation. [S.l.: s.n.], 2012. P. 1–8. Cit. on p. 42.

HE, K. et al. **Deep Residual Learning for Image Recognition**. [S.l.: s.n.], 2015. arXiv: 1512.03385 [cs.CV]. Cit. on p. 41.

HEUSEL, M. et al. **GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium**. [S.l.: s.n.], 2018. arXiv: 1706.08500 [cs.LG]. Cit. on p. 45.

HO, J.; JAIN, A.; ABBEEL, P. **Denoising Diffusion Probabilistic Models**. [S.l.: s.n.], 2020. arXiv: 2006.11239 [cs.LG]. Cit. on pp. 38–40, 57.

HO, J.; SALIMANS, T. **Classifier-Free Diffusion Guidance**. [S.l.: s.n.], 2022. arXiv: 2207.12598 [cs.LG]. Cit. on p. 42.

HUANG, G. et al. **Densely Connected Convolutional Networks**. [S.l.: s.n.], 2018. arXiv: 1608.06993 [cs.CV]. Cit. on p. 60.

IBE, O. C. **Markov Processes for Stochastic Modeling**. 2nd. Philadelphia, PA: Elsevier Science Publishing, June 2013. (Elsevier Insights). Cit. on p. 38.

JAIN, S. et al. Synthetic data augmentation for surface defect detection and classification using deep learning. **Journal of Intelligent Manufacturing**, Springer Science and Business Media LLC, v. 33, n. 4, p. 1007–1020, Nov. 2020. Available from: https://doi.org/10.1007/s10845-020-01710-x. Cit. on p. 44.

KANG, H. S. et al. Smart manufacturing: Past research, present findings, and future directions. **International Journal of Precision Engineering and Manufacturing-Green Technology**, Springer Science and Business Media LLC, v. 3, n. 1, p. 111–128, Jan. 2016. Available from: https://doi.org/10.1007/s40684-016-0015-5. Cit. on p. 20.

KINGMA, D. P.; WELLING, M. **Auto-Encoding Variational Bayes**. [S.l.]: arXiv, 2013. Available from: https://arxiv.org/abs/1312.6114. Cit. on pp. 35, 55.

KINGMA, D. P.; BA, J. **Adam: A Method for Stochastic Optimization**. [S.l.: s.n.], 2017. arXiv: 1412.6980 [cs.LG]. Cit. on p. 94.

KUJAWIŃSKA, A.; VOGT, K. Human Factors in Visual Quality Control. **Management and Production Engineering Review**, Walter de Gruyter GmbH, v. 6, n. 2, p. 25–31, June 2015. Available from: https://doi.org/10.1515/mper-2015-0013. Cit. on p. 17.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Springer Science and Business Media LLC, v. 521, n. 7553, p. 436–444, May 2015. Available from: https://doi.org/10.1038/nature14539. Cit. on p. 95.

LEE, J.; SINGH, J.; AZAMFAR, M. **Industrial Artificial Intelligence**. [S.l.: s.n.], 2019. arXiv: 1908.02150 [cs.CY]. Cit. on p. 20.

LI, X. et al. Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation. **Journal of Intelligent Manufacturing**, Springer Science and Business Media LLC, v. 31, n. 2, p. 433–452, Nov. 2018. Available from: https://doi.org/10.1007/s10845-018-1456-1. Cit. on p. 17.

LIAN, J. et al. Deep-Learning-Based Small Surface Defect Detection via an Exaggerated Local Variation-Based Generative Adversarial Network. **IEEE Transactions on Industrial Informatics**, v. 16, n. 2, p. 1343–1351, 2020. Cit. on p. 44.

LILLICRAP, T. P. et al. Backpropagation and the brain. **Nature Reviews Neuroscience**, Springer Science and Business Media LLC, v. 21, n. 6, p. 335–346, Apr. 2020. Available from: https://doi.org/10.1038/s41583-020-0277-3. Cit. on pp. 95, 96.

LIU, Z. et al. **Industrial Inspection with Open Eyes: Advance with Machine Vision Technology**. In: INTEGRATED Imaging and Vision Techniques for Industrial Inspection. London, U.K.: Springer, 2015. P. 1–37. Cit. on p. 22.

LUCIC, M. et al. **Are GANs Created Equal? A Large-Scale Study**. [S.l.: s.n.], 2018. arXiv: 1711.10337 [stat.ML]. Cit. on p. 44.

LV, X. et al. Deep Metallic Surface Defect Detection: The New Benchmark and Detection Network. **Sensors**, MDPI AG, v. 20, n. 6, p. 1562, Mar. 2020. Available from: https://doi.org/10.3390/s20061562. Cit. on p. 48.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y., et al. **Rectifier nonlinearities improve neural network acoustic models**. In: ATLANTA, GEORGIA, USA, 1. PROC. icml. [S.l.: s.n.], 2013. v. 30, p. 3. Cit. on p. 97.

MCCULLOCH, W. S.; PITTS, W. A Logical Calculus of the Ideas Immanent in Nervous Activity. **The Bulletin of Mathematical Biophysics**, v. 5, n. 4, p. 115–133, 1943. Cit. on p. 26.

MCKINSEY. The state of AI in 2022 and a half-decade in review. **McKinsey Global Survey on AI**, 2022. Cit. on p. 16.

MHASKAR, H. N.; MICCHELLI, C. A. How to choose an activation function. **Advances in Neural Information Processing Systems**, v. 6, 1993. Cit. on p. 97.

NAIR, V.; HINTON, G. E. **Rectified linear units improve restricted boltzmann machines**. In: PROCEEDINGS of the 27th international conference on machine learning (ICML-10). [S.l.: s.n.], 2010. P. 807–814. Cit. on p. 97.

NICHOL, A.; DHARIWAL, P. **Improved Denoising Diffusion Probabilistic Models**. [S.l.: s.n.], 2021. arXiv: 2102.09672 [cs.LG]. Cit. on pp. 41, 45.

OUSSIDI, A.; ELHASSOUNY, A. **Deep generative models: Survey**. In: 2018 International Conference on Intelligent Systems and Computer Vision (ISCV). [S.l.: s.n.], 2018. P. 1–8. Cit. on p. 32.

PAN, S. J.; YANG, Q. A Survey on Transfer Learning. **IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 10, p. 1345–1359, 2010. Cit. on p. 29.

PERES, R. S. et al. Industrial Artificial Intelligence in Industry 4.0 - Systematic Review, Challenges and Outlook. **IEEE Access**, v. 8, p. 220121–220139, 2020. Cit. on pp. 17, 20, 43.

PEREZ, L.; WANG, J. **The Effectiveness of Data Augmentation in Image Classification using Deep Learning**. [S.l.: s.n.], 2017. arXiv: 1712.04621 [cs.CV]. Cit. on p. 29.

RADFORD, A.; METZ, L.; CHINTALA, S. **Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks**. [S.l.: s.n.], 2016. arXiv: 1511.06434 [cs.LG]. Cit. on pp. 37, 56.

RAMESH, A.; GOYAL, A.; DOVRAT, D. **DALL-E: Creating Images from Text**. [S.l.: s.n.], 2021. https://openai.com/dall-e/. Cit. on p. 42.

REN, Z. et al. State of the Art in Defect Detection Based on Machine Vision. **International Journal of Precision Engineering and Manufacturing-Green Technology**, Springer Science and Business Media LLC, v. 9, n. 2, p. 661–691, May 2021. Available from: https://doi.org/10.1007/s40684-021-00343-6. Cit. on pp. 17, 21, 22.

RONNEBERGER, O.; FISCHER, P.; BROX, T. **U-Net: Convolutional Networks for Biomedical Image Segmentation**. In: LECTURE Notes in Computer Science. [S.l.]: Springer International Publishing, 2015. P. 234–241. Available from: https://doi.org/10.1007/978-3-319-24574-4_28. Cit. on p. 40.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, v. 65 6, p. 386–408, 1958. Cit. on p. 27.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, p. 533–536, 1986. Cit. on p. 95.

SAJJADI, M. S. M. et al. **Assessing Generative Models via Precision and Recall**. [S.l.: s.n.], 2018. arXiv: 1806.00035 [stat.ML]. Cit. on p. 45.

SALIMANS, T. et al. **Improved Techniques for Training GANs**. [S.l.: s.n.], 2016. arXiv: 1606.03498 [cs.LG]. Cit. on p. 45.

SALIMANS, T. et al. **PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications**. [S.l.: s.n.], 2017. arXiv: 1701.05517 [cs.LG]. Cit. on p. 40.

SAMUEL, A. L. Some Studies in Machine Learning Using the Game of Checkers. **IBM Journal of Research and Development**, v. 3, n. 3, p. 210–229, 1959. Cit. on p. 22.

SANDFORT, V. et al. Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks. **Scientific Reports**, Springer Science and Business Media LLC, v. 9, n. 1, Nov. 2019. Available from: https://doi.org/10.1038/s41598-019-52737-x. Cit. on p. 17.

SANTOS TANAKA, F. H. K. dos; ARANHA, C. **Data Augmentation Using GANs**. [S.l.: s.n.], 2019. arXiv: 1904.09135 [cs.LG]. Cit. on p. 17.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on Image Data Augmentation for Deep Learning. **Journal of Big Data**, Springer Science and Business Media LLC, v. 6, n. 1, July 2019. Available from: https://doi.org/10.1186/s40537-019-0197-0. Cit. on pp. 17, 29, 30, 43, 44, 54.

SIMONYAN, K.; ZISSERMAN, A. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. [S.l.: s.n.], 2015. arXiv: 1409.1556 [cs.CV]. Cit. on p. 60.

SOHL-DICKSTEIN, J. et al. **Deep Unsupervised Learning using Nonequilibrium Thermodynamics**. [S.l.: s.n.], 2015. arXiv: 1503.03585 [cs.LG]. Cit. on p. 38.

SONG, K.; YAN, Y. **NEU surface defect database**. [S.l.: s.n.], 2019. http://faculty.neu.edu.cn/yunhyan/NEU_surface_defect_database.html. Cit. on p. 44.

SU, C.-T. **Quality Engineering**. [S.l.]: CRC Press, Apr. 2016. Available from: https://doi.org/10.1201/b13909. Cit. on p. 17.

SUN, S. et al. A Survey of Optimization Methods From a Machine Learning Perspective. **IEEE Transactions on Cybernetics**, v. 50, n. 8, p. 3668–3681, 2020. Cit. on pp. 94, 95.

SZEGEDY, C. et al. **Rethinking the Inception Architecture for Computer Vision**. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2016. P. 2818–2826. Cit. on p. 45.

TAN, M.; LE, Q. V. **EfficientNetV2: Smaller Models and Faster Training**. [S.l.: s.n.], 2021. arXiv: 2104.00298 [cs.CV]. Cit. on p. 60.

TANG, W. et al. Deep learning based automatic defect identification of photovoltaic module using electroluminescence images. **Solar Energy**, Elsevier BV, v. 201, p. 453–460, May 2020. Available from: https://doi.org/10.1016/j.solener.2020.03.049. Cit. on pp. 43, 44.

TAO, F. et al. Data-driven smart manufacturing. **Journal of Manufacturing Systems**, Elsevier BV, v. 48, p. 157–169, July 2018. Available from: https://doi.org/10.1016/j.jmsy.2018.01.006. Cit. on p. 16.

THEIS, L.; OORD, A. van den; BETHGE, M. **A note on the evaluation of generative models**. [S.l.: s.n.], 2016. arXiv: 1511.01844 [stat.ML]. Cit. on p. 44.

TYAGI, S.; YADAV, D. A Comprehensive Review on Image Synthesis with Adversarial Networks: Theory, Literature, and Applications. **Archives of Computational Methods in Engineering**, Springer Science and Business Media LLC, v. 29, n. 5, p. 2685–2705, Oct. 2021. Available from: https://doi.org/10.1007/s11831-021-09672-w. Cit. on p. 42.

VASWANI, A. et al. **Attention is All you Need**. In: ADVANCES in Neural Information Processing Systems. [S.l.]: Curran Associates, Inc., 2017. v. 30. Cit. on p. 41.

WANG, Z. et al. Image quality assessment: from error visibility to structural similarity. **IEEE Transactions on Image Processing**, v. 13, n. 4, p. 600–612, 2004. Cit. on p. 46.

WEISS, K.; KHOSHGOFTAAR, T. M.; WANG, D. A survey of transfer learning. **Journal of Big Data**, Springer Science and Business Media LLC, v. 3, n. 1, May 2016. Available from: https://doi.org/10.1186/s40537-016-0043-6. Cit. on p. 29.

WHANG, S. E.; LEE, J.-G. Data collection and quality challenges for deep learning. **Proceedings of the VLDB Endowment**, Association for Computing Machinery (ACM), v. 13, n. 12, p. 3429–3432, Aug. 2020. Available from: https://doi.org/10.14778/3415478.3415562. Cit. on p. 17.

WUEST, T. et al. Machine learning in manufacturing: advantages, challenges, and applications. **Production &amp Manufacturing Research**, Informa UK Limited, v. 4, n. 1, p. 23–45, Jan. 2016. Available from: https://doi.org/10.1080/21693277.2016.1192517. Cit. on p. 16.

YAN, H.; YEH, H.-M.; SERGIN, N. **Image-based Process Monitoring via Adversarial Autoencoder with Applications to Rolling Defect Detection**. In: 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). [S.l.]: IEEE, Aug. 2019. Available from: https://doi.org/10.1109/coase.2019.8843313. Cit. on p. 44.

ZHOU, L.; ZHANG, L.; KONZ, N. Computer Vision Techniques in Manufacturing. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 53, n. 1, p. 105–117, 2023. Cit. on p. 22.

APPENDIX 1 – TRAINING DEEP FEEDFORWARD NETWORKS

The training procedure in ML and DL applications is based on an optimization problem, this configuration sets an optimization technique focused on minimizing the objective function of the model, in this scenario the loss function (subsection 2.2.2). By this procedure, the ML/DL model learns the best parameters to perform the task and achieve the desired results. A series of effective optimization methods were developed over the years, helping to improve the model results (SUN et al., 2020).

Gradient-based algorithms

These algorithms work by computing the gradients (derivates) of the cost function with respect to the model parameters. In this scenario, the Gradient Descent (GD) method is a common application, this method is a versatile optimization algorithm that can discover optimal solutions for a huge range of problems. The underlying concept of Gradient Descent involves continuously adjusting parameters to minimize a cost function in an iterative manner. The Stochastic Gradient Descent (SGD) is a variation of GD. This technique selects one random instance of the training data and computer the gradient based on it, therefore this approach is stochastic, i.e. have a random nature, this approach leads to more unstable results in every single evaluation, however, decreases to an optimum value on average. The key feature, when compared to the traditional approach, is that this implementation is less prone to get stuck in local minimum values (GERON, 2019).

Geron (2019) also highlights that evaluating single images per time, the standard SGD could take several minutes to perform, in this context, the idea of applying mini-batches at every interaction, i.e. computing the gradients in a subset of the dataset, instead of a single instance or the whole dataset, represents a balance between the efficiency and local minima avoidance.

Another famous optimization algorithm, especially in DL applications, is the Adaptive Moment Estimation (Adam), an algorithm used for first-order gradient-based optimization of stochastic objective functions (KINGMA; BA, 2017).

The Adam algorithm could be mathematically defined as:

$$\theta_{t+1} = m_t - \eta \frac{\sqrt{1-\beta_2}}{1-\beta_1} \cdot \frac{m_t}{V_t + \epsilon} \tag{1.1}$$

Where β_1 and β_2 are the exponential decay rates for the first and second moments of the gradient, ϵ is a small constant added for numerical stability, and η is the learning rate of the model (SUN et al., 2020). Additionally, m_t and V_t could be found as follows:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{1.2}$$

$$V_t = \beta_2 V_{t-1} + (1-\beta_2) g_t^2 \tag{1.3}$$

Where g is the gradient of the objective function with respect to θ, and t is the time step (SUN et al., 2020). The DL models applied for this monograph execution, are based on the Adam optimizer.

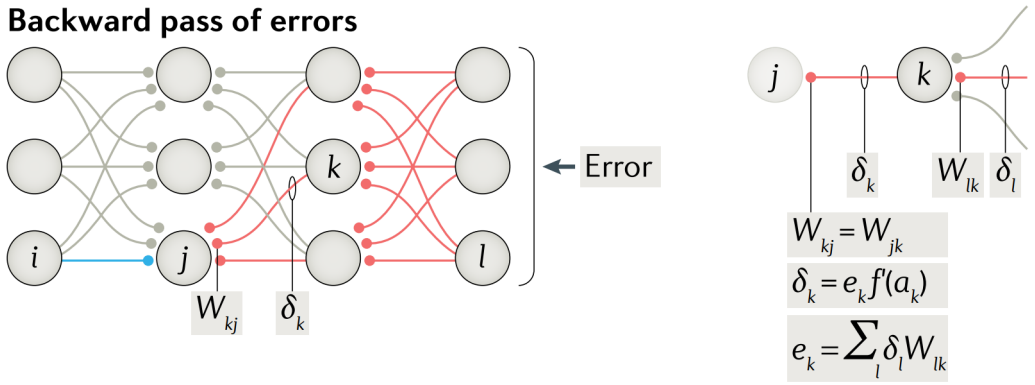
Backpropagation

Backpropagation is a commonly used algorithm in the field of DL to train multi-layered neural networks. Was defined by Rumelhart, Hinton, and Williams (1986), and involves the calculation of gradients or partial derivatives of a cost function concerning the model's weights, which is then used to adjust the weights of the neural network to minimize the cost function.

As explained by LeCun, Bengio, and Hinton (2015), the process of backpropagation is essentially an implementation of the chain rule for derivatives. The crucial understanding is that the derivative (or gradient) of the objective can be calculated by working backward from the gradient concerning its output. The backpropagation equation can be iteratively applied to propagate gradients through all modules, starting from the top to bottom. This gradient is then used to update the weights of the network using an optimization algorithm, such as SGD or mini-batch GD.

Lillicrap et al. (2020) in their paper *Backpropagation and the brain* make use of helpful visualizations to get understand the process of backpropagation, one of them is shown as follows (FIGURE 26)

FIGURE 26 – BACKPROPAGATION IN A NEURAL NETWORK



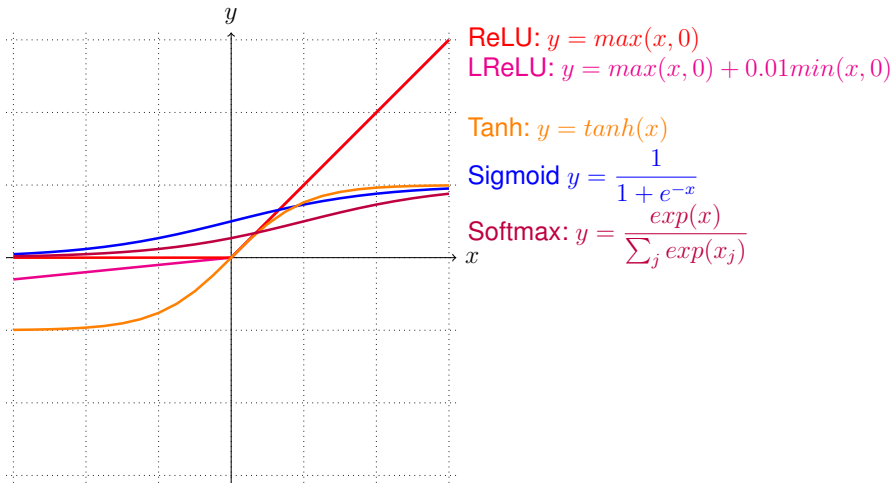
REFERENCE: Lillicrap et al. (2020)

Where W_jk, W_k, δ_k, and δ_l are the weights and error values between the neuron j and k, and between k and the output layer l, respectively. The term a represents the total incoming activity in a neuron, and $f'(a_k)$ represents the backward derivate process (LILLICRAP et al., 2020).

APPENDIX 2 – ACTIVATION FUNCTIONS

The activation functions perform a crucial role in ML and DL implementations. FIGURE 27 presents the most common for DL applications:

FIGURE 27 – COMMON ACTIVATIONS FUNCTIONS FOR DL MODELS



REFERENCE: The author (2023).

- **Rectified Linear Unit (ReLU):** Being x the input of to the activation function, ReLU eliminates the negative part of it (NAIR; HINTON, 2010)

- **Leaky Rectified Linear Unit (LReLU):** A variation of the ReLU function that keeps the negative part of the input (in the chart the positive part is equal to the ReLU function) (MAAS; HANNUN; NG, et al., 2013).

- **Tanh:** This function has an S-shaped curve. It accepts a real number as input and returns a value that falls within the range of -1 to 1 (BISHOP, 2006).

- **Sigmoid:** This function is particularly used in the output layer of neural networks that address binary classification tasks. It transforms real numbers into values ranging from 0 to 1 (MHASKAR; MICCHELLI, 1993).

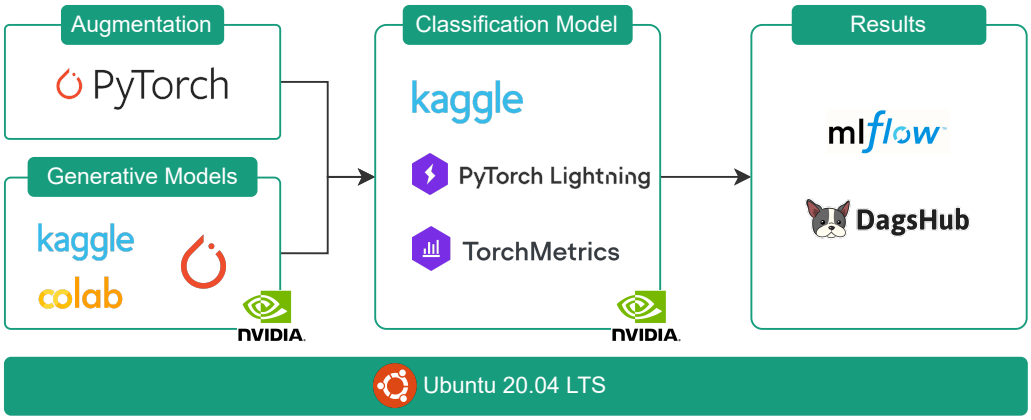
- **Softmax:** The softmax activation function is often applied to the output of fully connected layers to calculate the probability distribution across C classes, being a generalization of the Sigmoid function (BISHOP, 2006).

APPENDIX 3 – HARDWARE AND SOFTWARE DEFINITION

Concerning hardware, the initial development infrastructure was supported by Fraunhofer IPT, and it was made in a virtual machine with 16GB of RAM and an 8-core CPU. For the deep generative models training and final experiments, free cloud solutions were used. The convolutional VAE and DCGAN images were generated using the *Google Colab*[1] platform, which provides a GPU NVIDIA Tesla T4 with 15GB (as availability), CPU Intel(R) Xeon(R) CPU @ 2.30GHz - 2 cores, and 12.7GB RAM. The DDPM and Conditional DDPM models images were generated in the *Kaggle*[2] due to more stability during the runtimes, this platform offers a similar configuration as Google, except uses NVIDIA Tesla P100 as GPU model. Kaggle was also used for the final classification experiments.

Regarding software, all functionalities were implemented in 64-bit *Python* with a version greater than 3.7 and the operating system used was Linux-based (*Ubuntu 20.04 LTS*). The complete architecture for the code development is shown in the FIGURE 28.

FIGURE 28 – FRAMEWORK ARCHITECTURE



REFERENCE: The author (2023).

In order to implement the machine and deep learning features, the framework of choice was *PyTorch*[3] . The main reasons are concentrated on the level of support, documentation, and open-source implementations available from a research perspective.

[1] https://colab.research.google.com/
[2] https://www.kaggle.com/
[3] https://pytorch.org/

APPENDIX 4 – PRE- AND POST-PROCESSING FOR DATA

Regarding the raw training dataset, the original image is 1x224x224 pixels, for the Convolutional VAE input, they are resized to 1x64x64 pixels (one grayscale channel) and normalized between 0 and 1. For the DCGAN, DDPM, and conditional DDPM implementation, the image is resized 3x64x64 pixels (three grayscale channels) and normalized between -1 and 1, in order to match the model specifications.

The synthetic images generated by each deep generative model are saved in a specific directory for further evaluation. Before the saving process, the normalization technique applied by the model is reversed to ensure the correct pixel range for the synthetic data. Additionally, the images are resized again to 224x224 pixels, in the case of DCGAN, DDPM, and conditional DDPM, the three color channels are kept to qualitatively inspect any synthetic color that may differ from the original data. However, for the quantitative evaluation steps, all the images are turned back again to the grayscale condition.

As described in the section 3.2, the number of synthetic images generated is class-related, representing the difference between the number of instances in this class on the training dataset and 1000. The TABLE 20 displays the quantity of synthetic data generated per class by each of the four implemented models.

TABLE 20 – SYNTHETIC DATA GENERATED PER CLASS

Discrete class	Class	Synthetic instances
0	Punching	860
1	Welding Line	826
2	Crescent gap	856
3	Water spot	816
4	Oil spot	870
5	Silk spot	584
6	Inclusion	862
7	Rolled pit	981
8	Crease	967
9	Waist folding	904
TOTAL		8526

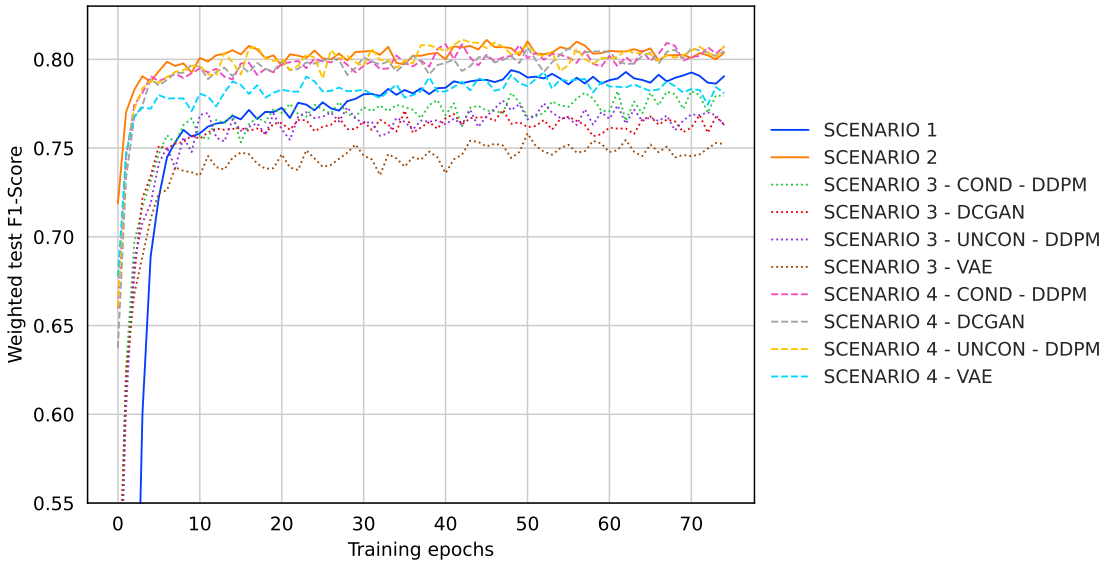
REFERENCE: The author (2023).

APPENDIX 5 – WEIGHTED CLASSIFICATION METRICS

Initial experimental setup

Concerning the initial experimental setup, FIGURE 29 displays the general overview of the weighted F1-Score results. The y-axis was zoomed for better visualization and the original view is in the next appendix. In the weighted metrics, the gap between the different scenarios becomes shorter, the results are more homogenous distributed ranging from 0.75 to 0.81. However, the weighted analysis keeps the sequence from higher to lower results of Scenarios 2 and 4, Scenario 1, and Scenario 3.

FIGURE 29 – TEST WEIGHTED F1-SCORE FOR INITIAL SCENARIOS



REFERENCE: The author (2023).

TABLE 21 shows the best weighted F1-Score for the different scenarios across the training epochs. The weighted metrics of precision and recall are also reported in the table for the same epoch.

Based on the results, the best performance in terms of weighted F1-Score was achieved by Scenario 4 - Unconditional DDPM, with the value of 0.8112, followed by the conventional augmentation in Scenario 2 with 0.8109. The gap between the two metrics is inside the standard deviation of the analysis. For other configurations of Scenario 4, the weighted F1-Score ranged from 0.7928 to 0.8093. Once again, Scenario 3 - VAE achieved the lowest result among the analyzed scenarios, with a value of 0.7583.

TABLE 21 – BEST TEST WEIGHTED RESULTS FOR INITIAL SCENARIOS

Scenario	F1-Score	Precision	Recall	Epoch	Rank
1	0.7939 ± 0.0048	0.8027 ± 0.0056	0.8005 ± 0.0033	48	5
2	0.8109 ± 0.0053	0.8160 ± 0.0049	0.8110 ± 0.0046	45	2
3 - COND - DDPM	0.7821 ± 0.0085	0.7898 ± 0.0046	0.7914 ± 0.0067	70	7
3 - DCGAN	0.7716 ± 0.0014	0.7785 ± 0.0053	0.7823 ± 0.0030	47	9
3 - UNCON - DDPM	0.7768 ± 0.0102	0.7766 ± 0.0097	0.7871 ± 0.0093	47	8
3 - VAE	0.7583 ± 0.0046	0.7596 ± 0.0061	0.7635 ± 0.0031	50	10
4 - COND - DDPM	0.8093 ± 0.0130	0.8162 ± 0.0113	0.8105 ± 0.0137	67	3
4 - DCGAN	0.8068 ± 0.0055	0.8116 ± 0.0050	0.8088 ± 0.0057	50	4
4 - UNCON - DDPM	**0.8112 ± 0.0078**	**0.8182 ± 0.0093**	**0.8117 ± 0.0074**	**42**	**1**
4 - VAE	0.7928 ± 0.0076	0.8012 ± 0.0073	0.7912 ± 0.0080	52	6

REFERENCE: The author (2023).

Pre-trained model with synthetic data

For the use of a pre-trained technique in Scenario 5, the model was also evaluated in weighted metrics, the results are in the TABLE 22 and TABLE 23.

TABLE 22 – BEST TEST WEIGHTED RESULTS IN 5 RUNS

Scenario	F1-Score	Precision	Recall	Epoch	Rank
1	0.7939 ± 0.0048	0.8027 ± 0.0056	0.8005 ± 0.0033	48	4
2	0.8109 ± 0.0053	0.8160 ± 0.0049	0.8110 ± 0.0046	**45**	2
3 - DCGAN	0.7716 ± 0.0014	0.7785 ± 0.0053	0.7823 ± 0.0030	47	5
4 - DCGAN	0.8068 ± 0.0055	0.8116 ± 0.0050	0.8088 ± 0.0057	50	3
5 - DCGAN	**0.8127 ± 0.0089**	**0.8176 ± 0.0072**	0.8134 ± 0.0090	55	**1**

REFERENCE: The author (2023).

TABLE 23 – BEST TEST WEIGHTED RESULTS IN 10 RUNS

Scenario	F1-Score	Precision	Recall	Epoch	Rank
2	0.8091 ± 0.0068	**0.8143 ± 0.0063**	0.8088 ± 0.0073	56	2
4 - DCGAN	0.8066 ± 0.0099	0.8105 ± 0.0094	0.8091 ± 0.0100	58	3
5 - DCGAN	**0.8096 ± 0.0075**	0.8142 ± 0.0069	**0.8104 ± 0.0078**	**55**	1

REFERENCE: The author (2023).

Based on them, Scenario 5 - DCGAN achieved the best performance for the scenarios reported for 5 and 10 runs. However, when compared to global results, the value reported in the 10 runs analysis is lower than previous outcomes, for example, Scenario 4 - Unconditional DDPM.

Image resizing pipeline

The TABLE 24 shows the mean and standard deviation results of weighted metrics for the Scenario 6 test dataset after 5 runs.

TABLE 24 – BEST TEST WEIGHTED RESULTS FOR SCENARIO 6

Scenario	F1-Score	Precision	Recall	Rank
6.1	0.6937 ± 0.0084	0.6917 ± 0.0150	0.7076 ± 0.0072	4
6.2	0.7202 ± 0.0093	0.7436 ± 0.0094	0.7140 ± 0.0089	3
6.3 - DCGAN	0.6911 ± 0.0110	0.6966 ± 0.0086	0.6996 ± 0.0123	5
6.4 - DCGAN	0.7385 ± 0.0143	**0.7595 ± 0.0185**	**0.7369 ± 0.0126**	2
6.5 - DCGAN	**0.7386 ± 0.0142**	0.7569 ± 0.0137	0.7343 ± 0.0140	**1**

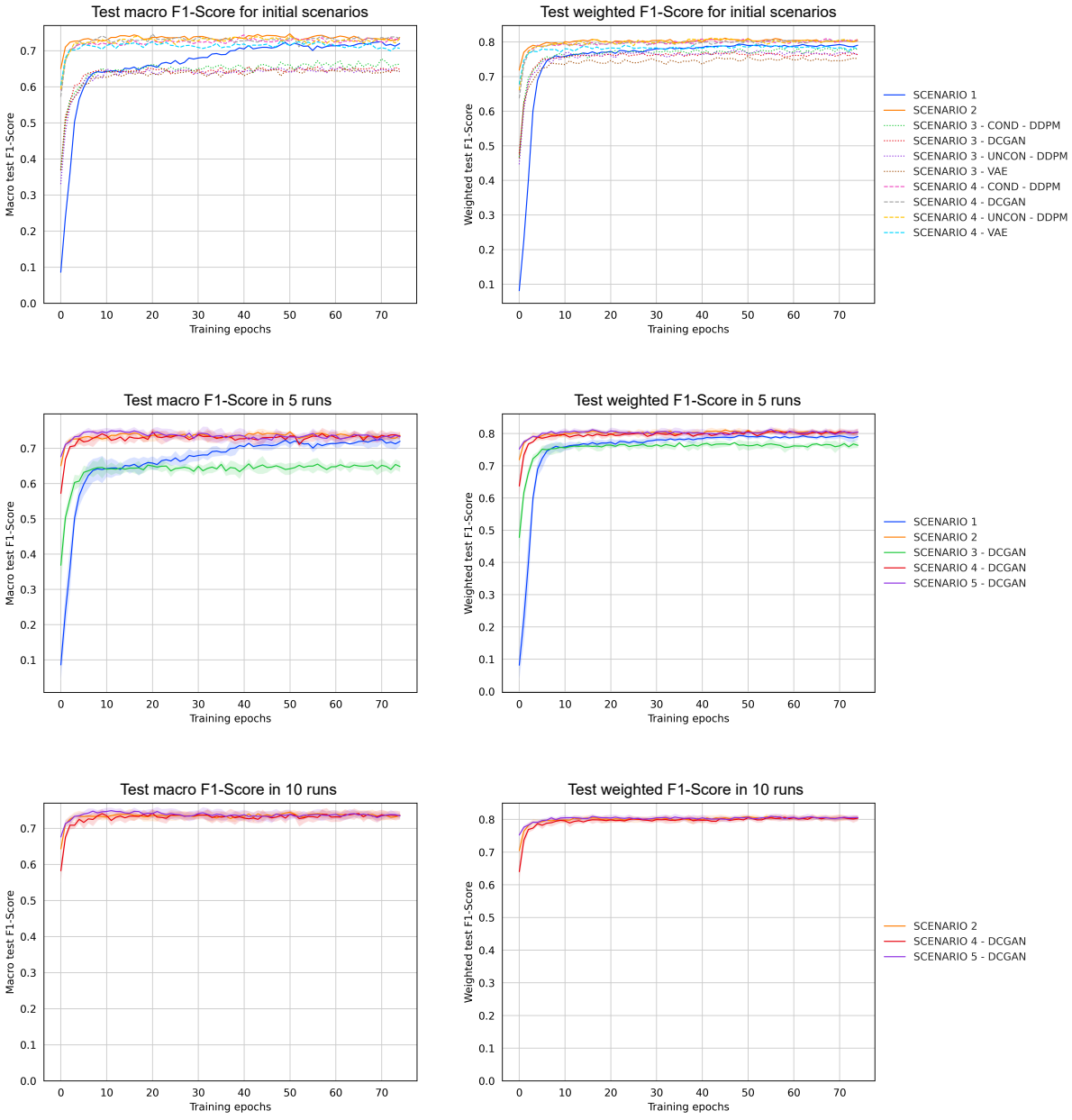
REFERENCE: The author (2023).

Considering the outcomes, both Scenarios 4 and 5 presented basically the same value of weighted F1-Score, with 0.7385 and 0.7386 respectively. The third best approach was Scenario 2, with a result of 0.7202. In the lowest position figures Scenario 3 - DCGAN. In general, the mean results were close to each other ranging from 0.6911 to 0.7386 for this analysis.

APPENDIX 6 – CLASSIFICATION CHARTS FULL Y-AXIS

The FIGURE 30 displays the F1-Score results in both macro and weighted analysis over epochs without the y-axis zoom.

FIGURE 30 – F1-SCORE OVERVIEW IN FULL Y-AXIS



REFERENCE: The author (2023).

APPENDIX 7 – THESIS REPOSITORIES

The TABLE 25 provides the links to the thesis repositories on GitHub, GitLab (Fraunhofer IPT internal), Kagglem, and DagsHub.

TABLE 25 – THESIS REPOSITORIES

Location	**Visibility**	**Content**	**Link**
GitHub	Public	Source code	https://github.com/michelhilg/data-synthesis
GitLab	Internal	Source code	https://gitlab.cc-asp.fraunhofer.de/mic79462/data-synthesis
Kaggle	Public	Datasets	https://www.kaggle.com/michelhilgemberg/datasets
DagsHub	Public	Results	https://dagshub.com/michelhilg/data-synthesis

REFERENCE: The author (2023).