

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Comparative Evaluation of Generative Models for Image Synthesis in Robotics: From GANs to Fine-Tuned Diffusion Models

Tesi di Laurea Magistrale in
Automation and Control Engineering - Ingegneria dell'automazione

Author: **Francesco Rubini**

Student ID: 247198
Advisor: Prof. Andrea Maria Zanchettin
Co-advisors: Eleonora Pantaleoni
Academic Year: 2024-25

Abstract

This thesis has been developed within the context of robotic assembly applications, where automated systems require reference images in order to recognize and manipulate industrial components. The main objective is to significantly reduce the time associated with manual photographic dataset collection by replacing the traditional physical acquisition process with automatic image generation through generative artificial intelligence models.

As a case study to validate the effectiveness of this approach, the EPART device is employed. EPART is an industrial control panel characterized by multiple configurations of colored buttons mounted on rectangular cases. The EPART domain represents a realistic benchmark to evaluate the ability of generative models to produce controllable and coherent images in an industrial setting subject to typical constraints, such as limited dataset size, the need for geometric precision, and the requirement to generalize to unseen product variants.

Three fundamental generative paradigms are analyzed and compared: conditional Generative Adversarial Networks (GANs) based on discrete attribute vectors, diffusion models trained from scratch with textual conditioning, and parameter-efficient fine-tuning of pre-trained foundation models through Low-Rank Adaptation (LoRA) techniques. Each approach is evaluated in terms of visual quality, semantic fidelity, computational efficiency, and generalization capability, with particular attention to the trade-offs between perceptual realism, controllability, and resource requirements.

The adoption of text-to-image models, especially diffusion-based architectures, enables intuitive interaction through natural language prompts, eliminating the need for advanced technical expertise by operators. Furthermore, the fine-tuning strategy on domain-specific datasets ensures high flexibility: the system can be rapidly adapted to new product categories through partial retraining on targeted datasets, making the approach scalable and applicable to multiple production lines and robotic assembly scenarios.

The results demonstrate that properly configured generative models are capable of producing high-quality synthetic images even when trained on extremely limited datasets, thereby validating the feasibility of this approach for real industrial applications. These

include data augmentation for robotic computer vision systems, virtual prototyping of product variants, and the reduction of data preparation time for automated assembly tasks.

Keywords: Generative artificial intelligence, diffusion models, LoRA fine-tuning, robotic assembly, text-to-image, data augmentation, conditional GANs, transfer learning

Abstract in lingua italiana

Il presente lavoro di tesi nasce nel contesto di applicazioni di assemblaggio robotico, dove sistemi automatizzati necessitano di immagini di riferimento per riconoscere e manipolare componenti industriali. L'obiettivo è quello di ridurre significativamente i tempi associati alla raccolta manuale di dataset fotografici, sostituendo il processo tradizionale di acquisizione fisica con la generazione automatica di immagini tramite modelli di intelligenza artificiale generativa.

Come caso di studio per validare l'efficacia di questo approccio, viene utilizzato il dispositivo EPART, un pannello di controllo industriale caratterizzato da diverse configurazioni di pulsanti colorati montati su case rettangolari. Il dominio EPART rappresenta un benchmark realistico per testare la capacità dei modelli generativi di produrre immagini controllabili e coerenti in un contesto industriale con vincoli tipici: dataset di dimensioni limitate, necessità di precisione geometrica, e requisiti di generalizzazione a varianti non viste durante l'addestramento.

Vengono analizzati e confrontati tre paradigmi generativi fondamentali: reti generative avversariali (GAN) condizionate su vettori di attributi discreti, modelli di diffusione addestrati da zero con conditioning testuale, e fine-tuning parameter-efficient di modelli foundation pre-addestrati tramite tecniche Low-Rank Adaptation (LoRA). Ciascun approccio viene valutato in termini di qualità visiva, fedeltà semantica, efficienza computazionale e capacità di generalizzazione, con particolare attenzione ai trade-off tra realismo percettivo, controllabilità e requisiti di risorse.

L'adozione di modelli text-to-image, in particolare quelli basati su diffusione, consente un'interazione intuitiva attraverso prompt in linguaggio naturale, eliminando la necessità di competenze tecniche avanzate da parte degli operatori. La strategia di fine-tuning su dataset domain-specific garantisce inoltre elevata flessibilità: il sistema può essere rapidamente adattato a nuove tipologie di prodotti mediante riaddestramento parziale su dataset mirati, rendendo l'approccio scalabile e applicabile a diverse linee produttive e scenari di assemblaggio robotico.

Il lavoro dimostra che modelli generativi opportunamente configurati possono produrre

immagini sintetiche di qualità elevata anche con dataset estremamente ridotti, validando la fattibilità dell'approccio per applicazioni industriali reali quali data augmentation per sistemi di visione artificiale robotica, prototipazione virtuale di varianti di prodotto, e riduzione dei tempi di preparazione dati per task di assemblaggio automatizzato.

Parole chiave: Intelligenza artificiale generativa, modelli di diffusione, LoRA fine-tuning, assemblaggio robotico, text-to-image, data augmentation, GAN condizionate, transfer learning

Contents

Abstract **i**

Abstract in lingua italiana **iii**

Contents **v**

1 Introduction **1**
1.1 State of art . . . 1
1.2 Motivation and context . . . 3
1.3 Problem statement . . . 4
1.4 Thesis objective . . . 4
1.5 Structure of the Thesis . . . 5

2 Theoretical Background **7**
2.1 Introduction to Machine Learning . . . 7
2.2 Artificial Neural Network . . . 8
2.2.1 Convolutional Neural Network . . . 10
2.2.2 Encoder-Decoder Architecture . . . 13
2.3 Implicit Generative Models . . . 15
2.3.1 Generative Adversarial Models (GAN) . . . 17
2.4 Denoising Diffusion Probabilistic Models (DDPM) . . . 20
2.4.1 Forward Process . . . 20
2.4.2 Reverse Process . . . 21
2.4.3 Classifier-Free Guidance (CFG) . . . 22
2.4.4 Training Diffusion Models . . . 24
2.5 Fine Tuning of Diffusion Models . . . 26
2.5.1 Low-Rank Adaptation (LoRA) . . . 30

3 Methodology **35**

3.1 Dataset . . . 35
3.2 Conditional GAN . . . 37
3.2.1 Conditioning Strategy . . . 37
3.2.2 Generator . . . 39
3.2.3 Discriminator . . . 40
3.2.4 Loss Function . . . 41
3.2.5 Training strategy and configuration . . . 42
3.3 Text Conditioned Diffusion model from scratch . . . 45
3.3.1 Architecture of the Diffusion Model . . . 46
3.3.2 Conditioning strategy . . . 47
3.3.3 Training strategy and configuration . . . 49
3.4 Fine Tuning Low-Rank Adaption (LoRA) . . . 52
3.4.1 Fine-tuning of a Pre-trained Diffusion Model . . . 52
3.4.2 Dataset and Captions . . . 53
3.4.3 Stable Diffusion Architecture and LoRA Integration . . . 57
3.4.4 LoRA Adaptation Strategy . . . 58

4 Results and Evaluations 63
4.1 Evaluation Metrics . . . 64
4.1.1 Fréchet Inception Distance (FID) . . . 64
4.1.2 Structural Similarity Index (SSIM) . . . 65
4.1.3 CLIPScore . . . 67
4.1.4 Final remarks on combined use . . . 68
4.2 GAN Model Results . . . 69
4.2.1 Quantitative results . . . 71
4.2.2 Qualitative analysis . . . 75
4.3 Diffusion Model trained from scratch results . . . 78
4.3.1 Quantitative results . . . 79
4.3.2 Qualitative results . . . 85
4.4 LoRA Fine-Tuning Results . . . 88
4.4.1 Quantitative Results . . . 89
4.4.2 Qualitative Results . . . 94
4.5 Comparison between Models . . . 100
4.5.1 Overall quantitative comparison . . . 100
4.5.2 Comparative Summary . . . 103

5 Conclusions and Future Developments 105
5.1 Extension to Robotic Applications and Automated Assembly . . . 107

5.2 Limitations of the Study 108
5.3 Future Work 108
5.4 General Conclusion 110

Bibliography **111**

List of Figures **115**

List of Tables **119**

Acknowledgements **121**

1 | Introduction

1.1. State of art

The current landscape of robotics is undergoing a paradigm shift driven by the advent of Generative Artificial Intelligence (Gen-AI), which is transforming robotic systems from rigid task executors into partners capable of co-creation and contextual adaptation [1, 2]. The state of the art can be structured into three main hierarchical levels: the foundational level (data generation), the intermediate level (semantic and visual translation), and the policy level (physical execution) [3].

Foundational level: data generation. For what concerns dataset generation, one of the historical limitations of machine-learning-based robotics is the requirement of extremely large datasets for training [1, 3]. In industrial scenarios, creating and annotating training datasets for machine learning models is time-consuming and resource-intensive, as it requires collecting numerous images across multiple viewpoints, lighting conditions, and object configurations. Recent research has introduced the use of Conditional Generative Adversarial Networks (cGANs) to generate large-scale dataset of synthetic images based on textual input or reference images [1]. This approach enables the creation of realistic object variations, which can then be automatically annotated by Large Multimodal Models (LMMs), drastically reducing human intervention and the level of technical expertise required for creating and annotating training dataset for machine learning models. [1].

Intermediate level: semantic and visual translation. Artificial intelligence can support the translation of abstract commands to executable physical actions, improving the effectiveness of human– robot collaboration [3]. The current state of the art leverages diffusion models for the visual generation of sub-goals: instead of receiving a list of coordinates, the robot interprets a generated image of the final object as a desired target state to be achieved [3]. This process, commonly referred to as visual foresight, enables the system to predict the physical consequences of its actions through images

that represent future results. [3]. In addition, systems such as GroundingDINO are used for zero-shot object detection, allowing robots to recognize previously unseen components during assembly processes [1].

Policy level: physical execution. With respect to physical execution, research has shifted from deterministic models to generative models capable of handling uncertainty [3]. Deterministic approaches typically output a single solution (e.g., a fixed trajectory or grasp pose) given the current state, whereas generative approaches can represent multiple plausible actions and select among them based on constraints and feedback. Variational Autoencoders (VAEs) and diffusion models are currently used to generate stable grasp configurations for objects with irregular geometries [3]. Diffusion Policies enable the generation of smooth trajectories that explicitly account for workspace constraints, overcoming the limitations of traditional models in dealing with dynamic interactions and task multimodality [3].

The integration of Gen-AI also has profound implications for the acceptance of robotic technologies in industrial environments [2]. The use of interfaces based on natural language and vision makes interaction with robotic arms more intuitive and anthropomorphic, increasing operator trustworthiness in the system [2]. This human-centric approach is a key element of Industry 5.0,in which robots collaborate with humans rather than replace them, supporting operators across different tasks [2].

Despite the recent advancements in image generation, the challenge of transferring models trained on synthetic data to real-world physical systems, known as the Sim-to-Real gap, remains open. [1, 3]. Many current generative models are able to produce visually plausible images that are physically impossible to construct. Consequently, the integration of physical constraints through physics-augmented training within the image generation process represents a major research frontier, aimed at ensuring system robustness and reliability in real industrial applications. In this context, *physics-augmented training* refers to incorporating physical feasibility constraints into the learning pipeline so that generated outputs are not only visually plausible but also consistent with basic physical and assembly rules (e.g., avoiding interpenetrations, unstable configurations, or placements that cannot be realized in the workspace). This can be achieved by generating training data in physics-based simulation, by adding constraint-aware regularization terms to the loss, or by validating generated samples through a physics-based checker[3].

1.2. Motivation and context

In recent years, the field of industrial automation and collaborative robotics has undergone rapid development, driven by the growing demand for flexible and adaptive systems capable of operating in highly customized production environments. In particular, the introduction of collaborative robots (cobots) has highlighted the need for new methodologies that enable more efficient human–machine interaction, reducing system configuration times and increasing adaptability to frequent product variations. One of the most critical aspects of human–robot interaction lies in the robot's visual perception of the working environment. Recent works aiming to simplify collaborative robot programming have shown that assembly tasks can be specified through images of *goal situations* provided by an operator, rather than through explicit low-level commands. In this setting, the robot uses the goal image as a visual target and plans actions by comparing the current scene with the desired final configuration [4]. A key limitation of this paradigm is that the set of achievable configurations is inherently tied to the availability of goal images: whenever a new target configuration is required, a corresponding goal image must be captured and provided. This limitation motivates the central question of this thesis, namely generating goal images directly from textual descriptions while emphasizing the capability to generalize the learned concepts beyond the specific examples observed during training.

Traditional industrial robots often rely on carefully acquired real image datasets to recognize and manipulate objects within their workspace. However, this image acquisition process is time-consuming and resource-intensive, as it requires collecting numerous images from multiple viewpoints, under varying lighting conditions, and across different configurations of the final object. In parallel, recent years have witnessed significant progress in the field of generative artificial intelligence, particularly in image generation, where diffusion models have become one of the most widely adopted approaches. These models have demonstrated a remarkable ability to generate realistic images from textual descriptions, opening new perspectives for the integration of generative systems to simplify robotic programming.

The core idea underlying this work emerges precisely from the convergence of collaborative robotics and generative AI. Specifically, this thesis proposes the use of generative models to produce realistic synthetic images of industrial objects directly from textual descriptions, with the goal of reducing, or potentially eliminating, the need to acquire large quantities of real images.

1.3. Problem statement

The main problem addressed in this thesis concerns the effective integration of generative artificial intelligence into collaborative robot programming, with the aim of simplifying the programming process.

According to the current state of the art, as previously discussed in Section 1.1, enabling a robot to assemble a final object from its separate components requires the availability of images of the goal situation, for example a fully assembled object with all its components. This image acquisition process is particularly demanding in terms of time and represents a significant bottleneck during the programming phase of robotic systems. Furthermore, this approach severely limits the flexibility of the system: the robot is constrained to assemble only those object configurations for which a dedicated image dataset has been manually acquired by an operator. Any variation in object configuration or transition to a new product family requires a new and complete image acquisition process, making the system poorly suited to dynamic and frequently changing production environments.

The introduction of generative artificial intelligence models represents a potential solution to this limitation, as it enables the automatic generation of synthetic images directly from textual description of the goal situation (for example the description of a fully assembled object). However, the application of such models in robotic contexts introduces a critical challenge: The generated images must be realistic enough to allow accurate recognition of components in the workspace and to support comparison with the initial state, allowing the robot to infer and execute the sequence of actions required to reach the goal configuration. Significant discrepancies between generated synthetic images and the real images of the components or the initial situation can compromise part recognition, and may ultimately make it impossible for the robot to infer and execute the required actions. The extent to which these requirements can be fully met in practical settings remains an open issue and is discussed among the limitations and future developments of this work.

1.4. Thesis objective

The main goal of this thesis is the implementation of a generative model that can synthesize realistic images from text inputs and that keeps strong generalization capabilities with respect to the concepts learned during training.

To ground the study, a simple industrial case study is adopted: the assembly of a modular remote controller with interchangeable buttons. This case study is used as a means to test

and compare the generative models presented in this thesis, and it does not represent the final goal of the study. Through the definition and use of a controlled dataset, the model is trained to learn fundamental concepts such as the presence of a button, the structure of the remote controller body, and the different color variations. One of the central goals of this work is to enable, from an application-oriented perspective, a human operator to request, through a simple textual prompt, any desired combination of buttons and colors, including configurations that are not explicitly present in the dataset used during training.

To achieve this objective, the thesis proposes and analyzes several models for generating images from text, evaluating their performance, limitations, and generalization capabilities. In particular, the following models are investigated:

- Generative Adversarial Networks (GAN);
- text conditioned denoising diffusion probabilistic models trained from scratch, specifically designed for the problem under consideration;
- pre-trained stable diffusion models, subsequently adapted to the application domain through fine-tuning techniques.

The comparison among these approaches makes it possible to determine an appropriate trade off between visual realism, generalization capability, and training efficiency, with the ultimate goal of identifying a generative model suitable for integration into collaborative robot programming.

1.5. Structure of the Thesis

Chapter 1 introduces the general context of the thesis, outlining the motivation behind the work, the main objectives of the project, and the problem addressed. The chapter also provides an overview of the current state of the art, with particular emphasis on the integration of generative artificial intelligence and collaborative robotics.

Chapter 2 presents the theoretical background required to understand the subsequent chapters. In particular, the chapter begins by introducing the basic principles of deep neural networks used in most image-generation models. It then presents the fundamental concepts of generative modeling, with a focus on Generative Adversarial Networks (GANs) and diffusion models.

Chapter 3 describes in detail the methodology adopted in this work. It presents the implementation of the different generative models considered, analyzing their main characteristics as well as the training strategies and experimental setup.

Chapter 4 is dedicated to the presentation and analysis of the experimental results. A systematic comparison among the different approaches is provided, highlighting the reasons that led to the selection of one model over the others.

Finally, Chapter 5 concludes the thesis by summarizing the results obtained and discussing limitations and future developments.

2 | Theoretical Background

This chapter introduces the main theoretical concepts underlying the methodologies developed throughout the thesis. Particular attention is devoted to the fundamentals of Machine Learning and Deep Learning applied to image processing, with a focus on the neural network architectures that form the core of modern generative models. The purpose of this section is not to provide an exhaustive treatment of the entire field, but rather to present in an organized and thorough manner the conceptual tools necessary to understand the design choices adopted in subsequent chapters.

2.1. Introduction to Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence that focuses on the development of systems capable of learning automatically from data [5, 6]. A Machine Learning system is not explicitly programmed to perform a specific task, but progressively improves its performance through experience, namely the analysis of past examples [5, 6]. Formally, it is stated that a program learns if its performance on a specific task improves as the acquired experience increases [5, 6].

Unlike traditional deterministic systems, based on rigid rules defined a priori, Machine Learning enables the modeling of complex and nonlinear relationships, proving particularly effective in domains characterized by high variability and uncertainty, such as computer vision, natural language processing, and robotics [5, 6].

The development process of a Machine Learning model generally follows a well-defined workflow [6]. In the initial phase, data are collected and subjected to preprocessing operations, such as normalization, noise removal, or manual annotation [6]. Subsequently, the model is trained on a subset of the available data, minimizing a cost function that measures the discrepancy between the model's predictions and the expected values [6]. Finally, performance is evaluated on data not seen during training, in order to estimate the system's generalization capability [6]. Figure 2.1 shows the schematic workflow of a machine learning model, illustrating the two main phases that compose the model:

training and testing [6].

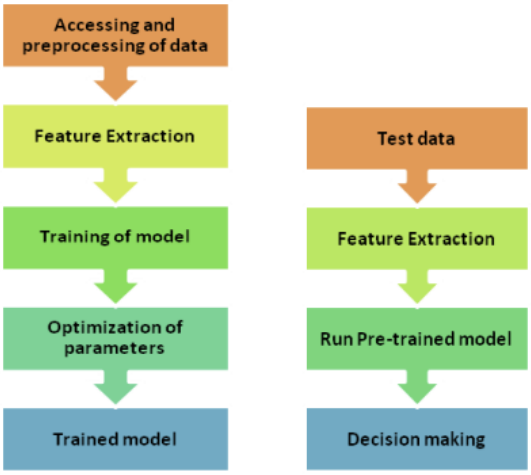


Figure 2.1: Training and testing of a machine learning model [6]

Based on the nature of the training data, Machine Learning methods can be divided into three main categories: supervised learning, unsupervised learning, and reinforcement learning [6].

In supervised learning, the model is trained using labeled data, that is, known input–output pairs [6]. This paradigm is widely used in classification and regression problems, such as object recognition or the prediction of continuous quantities [6].

Unsupervised learning, by contrast, operates on unlabeled data and aims to identify latent structures or recurring patterns within the data, as in the case of clustering or dimensionality reduction [6].

Reinforcement learning introduces an agent–environment dynamic, in which an agent learns an action policy by maximizing a reward function through a trial-and-error process. Although this paradigm is of great interest in robotics, it does not represent the main focus of the present thesis [6].

2.2. Artificial Neural Network

Artificial Neural Networks (ANNs) represent the most general neural model in the field of Machine Learning and constitute the theoretical foundation of Deep Learning [5, 6]. They are inspired by the functioning of biological neurons and are designed to approximate complex functions through the combination of linear and nonlinear operations [5, 6].

An artificial neuron receives a set of input values, each associated with a weight that

determines its influence. The weighted inputs are linearly combined and subsequently transformed through a nonlinear activation function [5, 6]. The presence of such nonlinearities is fundamental, as it enables the network to model relationships that could not be represented by purely linear models [5, 6].

As can be observed in Figure 2.2, an ANN is organized into layers: an input layer, one or more hidden layers, and an output layer [6]. Learning occurs through a weights optimization process, in which a loss function measures the error made by the network and the backpropagation algorithm allows parameters to be updated via gradient descent methods [5, 6].

As the number of hidden layers increases, ANNs become deep neural networks, capable of learning increasingly complex representations of the data [5, 6]. The simplest and most common ANN architecture is the fully connected (dense) network, in which each neuron in a layer is connected to all neurons in the subsequent layer. This design makes it a natural starting point for learning generic mappings from input data to output predictions. However, fully connected networks present significant limitations when applied to high-dimensional data, such as the large number of parameters to be learned and the inability to exploit internal structures within the data itself [5, 6].

These limitations have motivated the development of more specialized neural architectures, capable of incorporating structural assumptions about the input data. In particular, in the case of images, such considerations naturally lead to the introduction of Convolutional Neural Networks, which will be discussed in the next section.

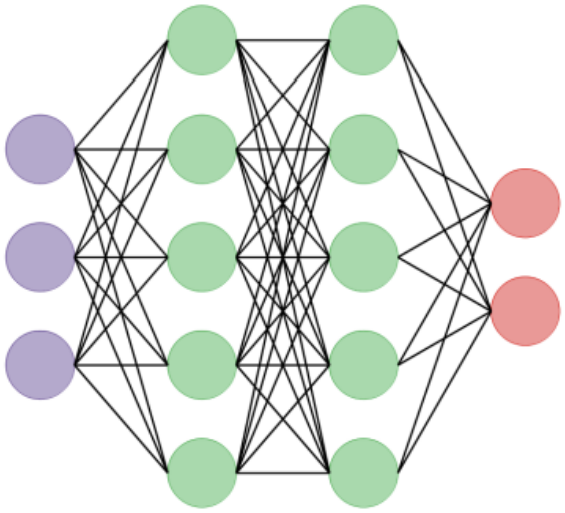


Figure 2.2: Schematic of a fully connected artificial neural network

2.2.1. Convolutional Neural Network

In the context of image processing, Convolutional Neural Networks (CNNs) represent the reference architecture and form the foundation of most modern Deep Learning models. Their widespread adoption is due to their ability to process data organized on regular grids, such as digital images. Convolutional filters exploit local spatial correlations, allowing CNNs to preserve spatial structure while drastically reducing the number of learnable parameters through local connectivity and weight sharing. These properties make CNNs far more scalable than fully connected architectures when applied to high-dimensional data, such as images.

A digital image is a grid of pixels and can be represented as as a matrix of numerical values, where each element represents the light intensity of a pixel. In the case of color images, such as RGB images, this representation takes the form of a three-dimensional tensor, characterized by width, height, and depth, the latter corresponding to the number of color channels. CNNs are designed to directly process this type of data, exploiting the spatial correlation between adjacent pixels.

Unlike fully connected neural networks, where each neuron is connected to all neurons in the previous layer, CNNs introduce the concept of local connectivity. Each neuron in a convolutional layer is connected only to a limited region of the previous layer, called the receptive field. This architectural choice allows for a drastic reduction in the number of parameters to be learned and makes the model more robust to local variations in the image.

The central mechanism of a Convolutional Neural Network is the convolution operation, which consists of applying a small set of weights, the kernel or filter, to the input image. The goal of this operation is the automatic extraction of local visual features, preserving the spatial structure of the image.

The kernel systematically slides over the entire image according to a defined step called stride and, at each position, an element-wise product is performed between the kernel weights and the corresponding local receptive field in the previous layer. The resulting values are subsequently summed to produce a single output value. By repeating this operation across all spatial positions of the previous layer, each convolutional layer produces its own feature map. As the network goes deeper, these feature maps progressively highlight increasingly abstract visual characteristics, ranging from simple edges and corners to more complex patterns.

Figure 2.3, shows the process of applying the kernel to an input tensor in a convolution

operation. For each position of the kernel, an element-wise product is performed followed by a summation operation, generating a corresponding value in the output feature map. This representation highlights how local information is progressively translated into a compact description of the image's relevant features.

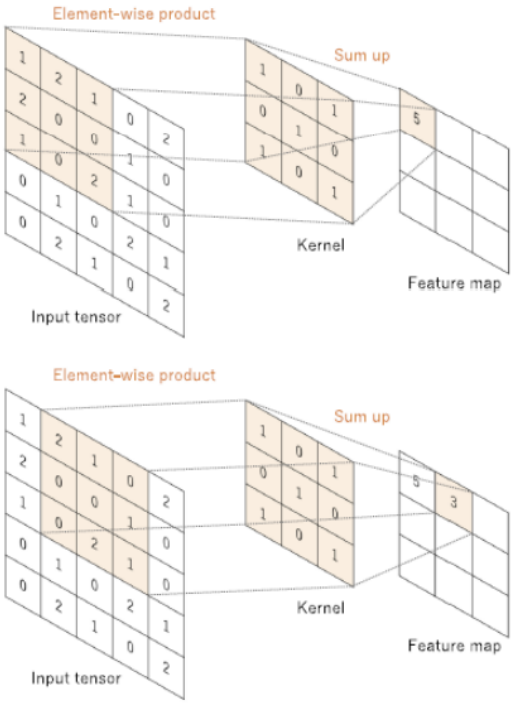


Figure 2.3: Schematic of the convolution operation in a Convolutional Neural Network. A kernel is applied locally to an input tensor: an element-wise product, followed by a summation generate the values of the output feature map.

The simultaneous application of multiple kernels enables the extraction of different types of features from the input image. Each kernel can be interpreted as a specialized feature extractor, capable of responding to specific visual patterns. The weight values that compose the kernels are not defined a priori, but are learned during the model training, making the approach completely data-driven.

A fundamental aspect of CNNs is parameter sharing: the same kernel is applied to all all spatial locations of the input image or of the previous layer's feature maps. This property drastically reduces the number of parameters to be learned and introduces translation invariance, the model's ability to recognize a specific feature regardless of its spatial position. This property is particularly important in computer vision applications and, as will be discussed in subsequent chapters, also plays a key role in image generation.

The convolution operation is intrinsically linear; therefore, to enable the network to model

nonlinear relationships, the generated feature maps are subjected to nonlinear activation functions. Among these, the most commonly used is the Rectified Linear Unit (ReLU), defined as:

$$ReLU(x) = \max(0, x) \tag{2.1}$$

The use of ReLU contributes to improving training stability and mitigating the vanishing gradient problem in deep networks.

Another key component of CNNs is the management of the spatial dimensionality of feature maps. The convolution operation, without specific measures, tends to reduce the image dimensions. To control this effect, padding is introduced, by adding zero borders around the input image. This technique allows maintaining unchanged spatial dimensions of feature maps, proving fundamental in architectures where spatial information must be preserved.

Alongside convolution operations, CNNs frequently employ pooling layers, whose purpose is to reduce the spatial dimension of feature maps. Pooling introduces a form of spatial abstraction, making the model more robust to small local variations in the input image, while progressively compressing the information contained in the image and in the intermediate feature maps into a more compact representation, preserving the most relevant visual patterns for the task. Among various pooling techniques, max pooling is the most widespread: it selects the maximum value within a local region of the feature map, preserving the most salient features.

Training a CNN consists of identifying the kernels that minimize a loss function, whose definition depends on the specific task. In the context of image generation, loss functions such as Mean Squared Error (MSE) are the most commonly used. Parameter optimization occurs through gradient descent algorithms, often in mini-batch configuration, using adaptive optimizers such as Adam.

When working with deep networks and high-dimensional data, such as images, the problem of overfitting frequently emerges, the model tend to memorize training data rather than learning generalizable representations. In CNNs, this phenomenon is counteracted through several strategies, including the use of sufficiently large and diverse datasets, data augmentation techniques, and the introduction of regularization mechanisms. Among these, dropout and batch normalization play a crucial role in improving training stability and the model's generalization capability.

In the context of the present thesis, Convolutional Neural Networks do not merely repre-

sent a tool for feature extraction, but constitute the fundamental architectural building block of the generative models considered. In particular, CNNs form the basis of encoder–decoder structures and U-Net architectures used in diffusion models, where the ability to capture local and global information is essential for generating realistic and coherent images.

2.2.2. Encoder-Decoder Architecture

Encoder–decoder architectures represent one of the fundamental structures of modern Deep Learning and form the core of numerous models employed in data compression, reconstruction, and generation. These architectures are designed to learn a compact representation of the information contained in the input data and, subsequently, to reconstruct an output from such representation. In the context of image processing, encoders and decoders are typically implemented through Convolutional Neural Networks, exploiting the properties of locality and parameter sharing introduced previously.

An encoder–decoder architecture consists of two main modules. The encoder receives the original data as input and progressively compresses it into a lower-dimentional representation, known as latent space. During this process, the encoder captures the most salient features of the input, eliminating redundancies and less relevant information. The decoder, performs the inverse process: starting from the latent representation, it reconstructs the desired output, which may coincide with the original input or represent a transformation of it.

The general operating principle of an encoder–decoder architecture is illustrated in Figure 2.4. The encoder compresses the input information into a compact latent representation, while the decoder uses such representation to reconstruct the output. This structure enables the model to learn an efficient encoding of the data, retaining relevant information while discarding redundant elements.

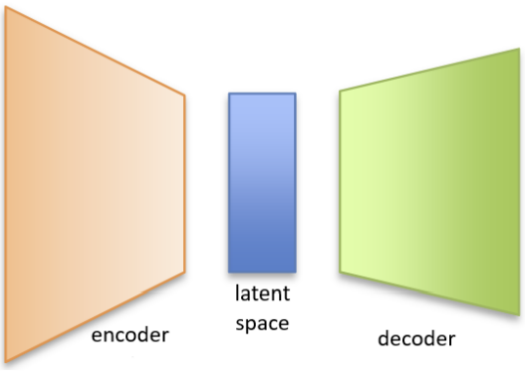


Figure 2.4: Conceptual diagram of an encoder–decoder architecture. The encoder compresses the input into a lower-dimentional representation (the latent space), while the decoder reconstructs the output from the latent space.

In the case of images, the encoder generally consists of a sequence of downsampling operations, performed by a series of pooling and convolutional layers with stride greater than one. This allows for a progressive reduction of the image's spatial resolution, while simultaneously the number of channels increases and thus the level of abstraction of the learned features. The decoder, symmetrically, employs upsampling operations, such as transposed convolutions or interpolations followed by convolution, to reconstruct an image with the same dimention of the input one.

One of the most well-known examples of encoder–decoder architecture is the autoencoder, whose objective is to reconstruct the input as faithfully as possible. In this case, the encoder learns a compressed representation of the image, while the decoder attempts to reproduce the original image from the latent space. Although autoencoders are often associated with unsupervised learning, they can be more accurately interpreted as a case of self-supervised learning, since the training target coincides with the input itself.

The training of an encoder–decoder architecture is based on minimizing a loss function that measures the discrepancy between the reconstructed output and the desired target. In the context of images, loss functions such as Mean Squared Error (MSE) or Binary Cross-Entropy (BCE) are commonly used, depending on the nature of the data and the application. The choice of loss function directly influences the characteristics of the reconstructed image: for example, the use of MSE tends to favor globally coherent reconstructions that are sometimes excessively smoothed, meaning that fine-grained spatial details and sharp local structures may be lost.

A crucial aspect of encoder–decoder architectures is the nature and dimensionality of the latent space. An excessively compact latent representation may lead to excessive infor-

mation loss, compromising reconstruction quality; conversely, an overly large latent space may reduce compression effectiveness and increase the risk of overfitting. Consequently, the design of the latent space dimensionality requires a good balance between reconstruction fidelity and the degree of information compression. This trade-off also plays a central role in the advanced generative models analyzed in subsequent chapters.

Also in image generation, encoder–decoder architectures assume a relevant role. They do not merely reconstruct existing images, but provide a natural structure for modeling complex transformations between different representation spaces. This characteristic is exploited in advanced generative models, such as Variational Autoencoders (VAEs) and, in U-Net architectures employed in diffusion models.

An intrinsic limitation of traditional encoder–decoder architectures lies in the loss of high-resolution spatial information during the compression process. To address this problem, architectures featuring skip connections between encoder and decoder have been introduced, allowing direct transfer of information at different resolution levels. This approach forms the basis of U-Nets, in which feature maps produced by the encoder are concatenated with those of the decoder, significantly improving the quality of reconstructed images.

As will be discussed later, in diffusion models, encoder–decoder architectures and U-Nets in particular, play a central role in the progressive denoising process. In these models, the decoder does not merely reconstruct a static image, but learns to iteratively remove noise from a degraded signal, using the multiscale representations provided by the encoder. The ability to integrate local and global information, typical of encoder–decoder structures based on CNNs, is therefore essential for obtaining realistic synthetic images.

Finally, it is important to emphasize that encoder–decoder architectures are strongly data-dependent. Their generalization capability is closely linked to the quality, variety, and representativeness of the training dataset. In real-world applications, such as those addressed in this thesis, dataset design and data augmentation strategies play a determining role in the model's success.

2.3. Implicit Generative Models

A significant portion of the probabilistic models used in Machine Learning belongs to the category of prescribed models, namely models that explicitly define a probability density

$$q_\theta(x) \tag{2.2}$$

and, consequently, a log-likelihood

$$\log(q_\theta(x)) \tag{2.3}$$

that can be directly computed and optimized during training. This approach forms the basis of numerous classical statistical and probabilistic models. In many real-world applications, however, the explicit definition of a likelihood proves impractical: the system of interest can be described more naturally through a simulator or a complex generative procedure, for which the associated probability density is not accessible in closed form or is computationally intractable.

In these cases, socalled implicit generative models are employed. These models are not defined through an explicit density function, but rather through a stochastic data generation process, while the likelihood remains not directly evaluable [7]. This shift in perspective allows for modeling extremely complex distributions, at the cost of having to adopt alternative learning strategies, called likelihood-free.

A typical formulation of implicit generative models introduces a latent variable

$$z \sim q(z) \tag{2.4}$$

and a parametric deterministic map

$$G_\theta \tag{2.5}$$

generally implemented as a deep neural network, which transforms the latent variable into a sample in the data space:

$$x = G_\theta(z), \quad z \sim q(z) \tag{2.6}$$

This definition is extremely flexible and allows for generating high-dimensional samples with complex structures, such as images. However, precisely because the induced density $q_\theta(x)$ is not available in explicit form, learning the parameters θ cannot occur through direct likelihood maximization, but requires approaches based on comparison between real samples and generated samples [7].

A particularly useful unifying perspective interprets learning in implicit models as a two-sample testing problem. In this scenario, one has two sets of samples: one drawn from

the real data distribution and one generated by the model. The objective then becomes defining a criterion capable of measuring how much the two distributions differ and updating the generator's parameters to reduce such discrepancy [7]. This interpretation naturally leads to the concept of density-ratio estimation, that is, the estimation of the ratio between densities, and clarifies the deep connection between statistical divergences, classification problems, and adversarial objective functions [8].

2.3.1. Generative Adversarial Models (GAN)

Within the landscape of implicit generative models, Generative Adversarial Networks (GANs) represent one of the most influential and innovative frameworks [9]. A GAN introduces two neural networks with opposing roles, trained simultaneously through a competitive mechanism.

The generator G_θ receives as input a latent variable

$$z \sim p_z(z) \tag{2.7}$$

and produces synthetic samples $G_\theta(z)$ in the data space. The discriminator $D_\phi(x)$, conversely, is a neural network that receives a sample x as input and estimates the probability that it comes from the real data distribution rather than from the generator.

The training of a GAN is formulated as a minimax game, in which the discriminator seeks to maximize its ability to distinguish between real and generated samples, while the generator attempts to "fool" the discriminator by producing increasingly realistic samples. This dynamic is described by the following objective function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1 - D(G(z)))\right] \tag{2.8}$$

This formulation admits a clear probabilistic interpretation. The discriminator's objective is equivalent to maximizing the log-likelihood of a binary classification problem (real samples versus generated samples), while the generator update induces a progressive modification of the distribution p_g toward the real data distribution p_{data} [9]. Under ideal conditions, the game equilibrium is reached when the generator exactly reproduces the data distribution and the discriminator is no longer able to distinguish between real and synthetic data, assigning probability equal to 0.5 to both classes.

Figure 2.5 illustrates this adversarial learning mechanism, highlighting the information

flow between generator and discriminator and the role of the latent variable in constructing synthetic samples.

From an architectural perspective, the generator can be interpreted as a particular instance of a neural decoder, which maps a latent representation to a high-dimensional space, analogously to what occurs in the encoder–decoder structures discussed previously. However, unlike autoencoders, in GANs there is no explicit encoder: the structure of the latent space is learned indirectly through the feedback provided by the discriminator. This aspect substantially distinguishes GANs from models based on direct reconstruction and contributes to explaining their ability to produce realistic images.

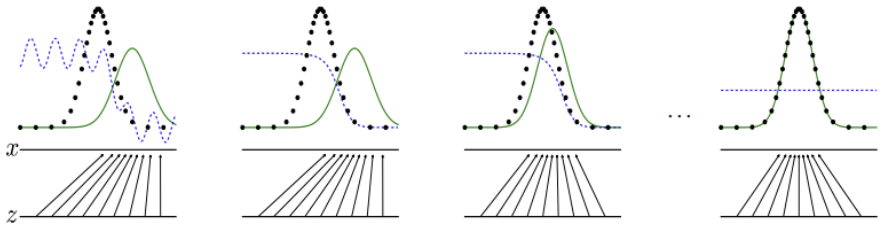


Figure 2.5: Conceptual diagram of a Generative Adversarial Network. The black dotted curve represents the true data distribution p_{data}, while the green curve denotes the distribution of samples generated by the generator p_g. The blue dashed curve corresponds to the discriminator output, i.e., an estimate of the probability that a given sample originates from the real data distribution rather than from the generator. The generator maps a latent variable z to a synthetic sample, while the discriminator evaluates the probability that a sample comes from the real data distribution. The training process occurs through an adversarial mechanism that guides the alignment between p_g and p_{data} [9].

Structural Limitations and Critical Issues of GAN Models

Despite the strong impact that Generative Adversarial Networks have had in the field of image generation, their practical use is accompanied by a series of criticalities that make training complex and often unstable. Such limitations are not attributable to specific architectural choices, but derive directly from the adversarial nature of the framework and the absence of an explicit likelihood function.

A first difficulty concerns the stability of the training process. Since generator and discriminator are optimized simultaneously according to opposing objectives, training assumes the form of a two-player optimization problem, for which there are no general convergence guarantees through gradient descent methods. In practice, the system can manifest prolonged oscillations, divergences, or chaotic behaviors, making it difficult to establish

whether the model is actually improving over time [7, 9]. This instability is particularly accentuated when the generator and discriminator do not have comparable learning capacity: if one network overpowers the other, the adversarial game breaks down.

A second central problem is the phenomenon known as mode collapse. In this case, the generator tends to produce a limited set of samples, neglecting large portions of the real data distribution. From a probabilistic perspective, this implies that the generated distribution p_g approximates only some modes of p_{data}, compromising the diversity of synthetic samples. Mode collapse represents a particularly relevant limitation in applications where high variability of generated data is required, since visually realistic but poorly diversified images are of limited utility [9].

A further critical issue is related to the vanishing gradient problem. During training, the discriminator can rapidly become too effective at distinguishing between real and generated samples. Under such conditions, the gradient provided to the generator tends to vanish, drastically reducing the model's ability to update its parameters significantly. This effect can slow down or even block the generator's learning process, making careful control of the dynamics between the two models necessary [9].

To these difficulties is added the lack of an intrinsic evaluation criterion. Since GANs do not optimize an explicit likelihood, it is not possible to directly monitor an objective function that reliably reflects the quality of generated samples. Consequently, evaluation of the model's performance typically occurs through empirical metrics based on auxiliary networks or indirect statistical comparisons, which do not always correlate univocally with visual realism or with the completeness of the learned distribution [7]. This aspect makes both the selection phase of the best model and the comparison between different architectures complex.

Taken together, these limitations make GANs powerful but difficult to control, especially in applications where reliability, reproducibility, and stability of the training process are required. Such criticalities have motivated the development of alternative approaches to implicit generation, capable of maintaining high visual quality while reducing problems of instability and mode collapse. In particular, diffusion models, which will be analyzed in the following section, introduce a different learning paradigm, based on a progressive generative process and a more stable objective formulation, proving particularly suitable for complex scenarios.

2.4. Denoising Diffusion Probabilistic Models (DDPM)

In recent years, diffusion models have established themselves as one of the most powerful generative models, achieving unprecedented results in terms of quality and diversity in image synthesis and surpassing, in many applications, Generative Adversarial Networks [10, 11]. The success of these models is linked to an elegant and stable probabilistic formulation that avoids adversarial training and is based on learning a progressive denoising process.

Diffusion models learn to invert a data degradation process inspired by non-equilibrium thermodynamics. Specifically, they define a chain of latent variables describing a progressive destruction of data structure through the addition of Gaussian noise, followed by learning an inverse process capable of reconstructing the original data from pure noise [10].

This approach distinctly separates diffusion models from other generative approaches: while Variational Autoencoders rely on a single global latent variable and GANs implicitly learn a distribution through adversarial competition, diffusion models utilize a temporal hierarchy of latent states, each representing the data at a different level of corruption.

2.4.1. Forward Process

The forward process is a fixed Markov chain that progressively transforms a clean sample $x_0 \sim p_{\text{data}}(x)$ into isotropic Gaussian noise through T successive steps. At each iteration, a controlled amount of noise is added to the input data, progressively degrading its structure until only pure noise remains at the final time step.

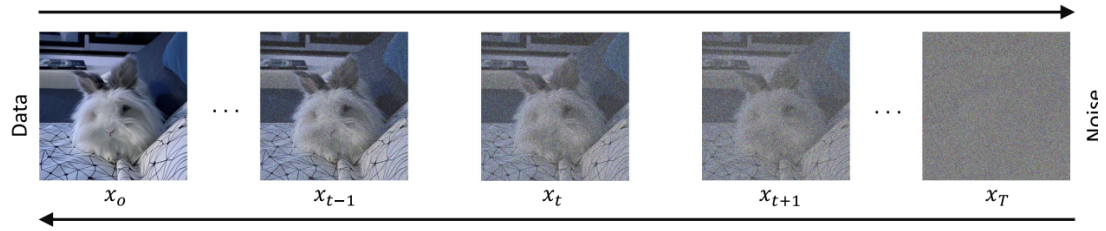


Figure 2.6: Example of the forward diffusion process. Starting from the clean data x_0, Gaussian noise is progressively added until, after T iterations, a distribution close to a standard isotropic Gaussian is obtained. Each intermediate state x_t represents a partially degraded version of the original data. [10]

The transition between two consecutive steps is defined as:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\right), \tag{2.9}$$

where $\beta_t \in (0,1)$ represents a *variance schedule* that controls the amount of noise added at each iteration.

Introducing $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, it is possible to derive a closed form that allows sampling x_t directly from x_0:

$$q(x_t \mid x_0) = \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}\, x_0, (1-\bar{\alpha}_t) I\right). \tag{2.10}$$

This property is crucial from a computational perspective, as it allows generating noisy versions of the data without having to explicitly iterate through all intermediate steps. When T is sufficiently large and the schedule is appropriately chosen, the final distribution x_T converges to a standard Gaussian $\mathcal{N}(0, I)$.

2.4.2. Reverse Process

The heart of diffusion models is the reverse process, which learns to transform pure noise back into data by progressively removing the noise added during the forward diffusion. Ideally, the goal of the reverse process is to model the inverse distribution:

$$q(x_{t-1} \mid x_t), \tag{2.11}$$

which, however, depends on the entire data distribution and is intractable. For this reason, a parametric model p_θ is introduced to approximates this distribution as a Gaussian:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I\right). \tag{2.12}$$

Instead of directly predicting the mean μ_θ, it has been shown more effective to train the network to estimate the noise added during the forward process, namely a function $\varepsilon_\theta(x_t, t)$ [11]. In practice, the parametric model is typically implemented as a neural network (e.g., a U-Net), which takes as input the noisy sample x_t and the corresponding timestep t, and outputs an estimate of the injected noise. By predicting this noise component, the model can subtract it from x_t and progressively denoise the signal, starting from pure noise and iteratively reconstructing the desired output sample. This choice considerably simplifies

the training objective and improves numerical stability. From an architectural perspective, the network used is typically a convolutional U-Net, which combines downsampling and upsampling operations with skip connections. This structure closely resembles the encoder–decoder architectur discussed previously: the forward process can be interpreted as a progressive encoding toward a noisy representation, while the reverse process plays the role of generative decoding [10].

2.4.3. Classifier-Free Guidance (CFG)

Despite the probabilistic formulation of diffusion models enabling stable and high-quality generation, a relevant limitation of unconditioned models lies in the difficulty of precisely controlling the semantic content of generated images. In an unconditioned DDPM, the denoising network learns to model the data distribution without any auxiliary information, generating samples solely from random noise. Conversely, conditioned diffusion models incorporate an additional conditioning signal (e.g., class labels, reference images, or text embeddings) to steer the generation process toward a desired content.

Traditionally, this problem was addressed through classifier guidance, which introduces an external classifier to guide the denoising process. However, this approach presents significant disadvantages, including increased computational complexity and greater numerical instability.

The Classifier-Free Guidance (CFG) method overcomes these limitations by completely eliminating the need for an auxiliary classifier, integrating the guidance mechanism directly into the diffusion model itself [12]. The key idea consists of training a single denoising model in both conditioned and unconditioned modes. During training, the condition (for example, a text embedding) is occasionally removed with a certain probability, inducing the model to simultaneously learn both the conditional distribution $p_\theta(x \mid c)$ (generation given a condition c) and the unconditional distribution $p_\theta(x)$ (generation without any condition).

During the generation process, guidance is applied by combining the noise predictions obtained in the two cases. Specifically, indicating with $\varepsilon_\theta(x_t, c)$ the conditioned noise estimate and with $\varepsilon_\theta(x_t, \emptyset)$ the unconditioned estimate, the guided prediction is defined as:

$$\hat{\varepsilon}_\theta(x_t, c) = (1 + s)\, \varepsilon_\theta(x_t, c) - s\, \varepsilon_\theta(x_t, \emptyset), \tag{2.13}$$

where $s \geq 0$ is the *guidance scale*, a hyperparameter that controls the guidance intensity [12].

From an interpretative perspective, this mechanism amplifies the components of the guidance signal, i.e. the directions of the denoising update (gradients with respect to the sample) that are aligned with the condition, while attenuating those that are not coherent with it. Low values of s favor greater sample diversity, while high values impose greater adherence to the condition, at the cost of a possible reduction in variability.



Figure 2.7: Effect of Classifier-Free Guidance on the generation process. The leftmost image corresponds to the unguided case ($s = 0$), in which the model generates unconditioned samples characterized by high diversity but weak adherence to the conditioning signal (i.e., limited coherence with the target class or text prompt). Moving to the right, increasing the guidance parameter s progressively strengthens the influence of the condition, making the generated samples more consistent with the specified class/text content, at the cost of reduced generative variability [12].

Figure 2.7 intuitively illustrates the role of Classifier-Free Guidance. In the absence of guidance ($s = 0$), the denoising process follows exclusively the model's unconditioned dynamics, producing highly variable samples that are difficult to control from a semantic perspective.

As the guidance parameter s increases, the generation trajectory is progressively deviated toward regions of the latent space coherent with the imposed condition. This mechanism allows continuously regulating the trade-off between diversity and conditional fidelity, without introducing external classifiers or modifying the model architecture.

From an architectural perspective, Classifier-Free Guidance integrates naturally into the denoising process learned by the U-Net. The model, already interpreted as a generative decoder in the reverse diffusion process, is evaluated twice on the same noisy state x_t, differing only in the presence or absence of the condition. The linear combination of the two predictions does not alter the structure of the reverse process but modifies its direction in the gradient field, guiding the generation trajectory toward relevant regions of the data space.

This characteristic makes CFG particularly suitable for applications where fine control of the generated content is required, such as in text-to-image models and, as will be discussed in subsequent chapters, in conditioned generation of images functional to collaborative robots programming.

2.4.4. Training Diffusion Models

Training a diffusion model is based on optimizing a probabilistic objective that aims to learn the reverse denoising process defined in the previous section. Unlike GANs, training is not adversarial and does not require balancing between two competing networks, resulting in a significantly more stable procedure [10, 11].

Formally, the original DDPM objective derives from maximizing a Variational Lower Bound (VLB) on the data log-likelihood. However, it has been shown that a simplified formulation of the objective produces comparable or better results in terms of visual quality, while reducing computational complexity [11].

As previously mentioned, the model is trained to predict the noise ε added during the forward process. Given a sample x_0, a time instant $t \sim \mathcal{U}(\{1, \dots, T\})$ and Gaussian noise $\varepsilon \sim \mathcal{N}(0, I)$, the following is obtained:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon \tag{2.14}$$

The neural network $\varepsilon_\theta(x_t, t)$ is then trained to estimate the noise ϵ used in the forward process, by minimizing the following loss function:

$$\mathcal{L}_{\text{simple}}(\theta) = \mathbb{E}_{t, x_0, \varepsilon}\left[\|\varepsilon - \varepsilon_\theta(x_t, t)\|^2\right] \tag{2.15}$$

This formulation presents several fundamental advantages. First, it transforms generative learning into a well-conditioned regression problem, avoiding numerical instabilities and training collapses. Second, it allows randomly sampling the time step t at each iteration, making training independent of the diffusion chain length [11].

The network ε_θ is typically implemented as a convolutional U-Net, which receives as input the noisy image x_t and a temporal embedding t. Skip connections allow preserving spatial information at different resolution levels, making the architecture particularly suited to the progressive denoising task. This perspective further aligns diffusion models with the encoder–decoder structures discussed previously: training can be interpreted as learning a time-conditioned stochastic decoder [10].

Algorithm 2.1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \text{Uniform}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
    $\nabla_\theta \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2$
until converged

Algorithm 2.2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

Algorithms 2.1 and 2.2 summarize the functioning of DDPM: Algorithm 2.1 describes the training phase, while Algorithm 2.2 presents the sampling (generation) procedure.

The *Training* algorithm (Algorithm 2.1) formalizes the learning objective discussed previously. At each iteration, a clean sample $\mathbf{x}_0$ is corrupted to a random noise level t, generating $\mathbf{x}_t$ through the closed form of the forward process. The network $\boldsymbol{\epsilon}_\theta$ is then trained to predict the added noise $\boldsymbol{\epsilon}$, minimizing a quadratic loss function. The choice of uniformly sampling the timestep t makes training independent of the diffusion chain length and allows simultaneously learning denoising at all noise levels.

The *Sampling* algorithm (Algorithm 2.2) describes the actual generation procedure. Starting from an initial sample of Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the model iteratively applies the learned reverse process, progressively removing noise until obtaining a sample $\mathbf{x}_0$ in the data space. The stochastic term $\mathbf{z}$ guarantees diversity of generated samples, while the explicit dependence on timestep t allows the network to adapt denoising to the current corruption level.

This algorithmic formulation highlights one of the key characteristics of diffusion models: the asymmetry between training and generation. During training, the model directly observes corrupted real samples, while in sampling phase it operates exclusively on synthetic noise, reconstructing data through a sequence of controlled stochastic transformations. This separation contributes to training stability and improves the quality of generated samples. Although training diffusion models is known for its stability, it presents a significantly higher computational cost compared to other generative models. In particular, the need to simulate numerous diffusion steps and evaluate the network for many temporal iterations makes training expensive in terms of time and memory [12].

However, several aspects contribute to making training manageable in practice. The use

of mini-batches, adaptive optimizers such as Adam, and learning rate scheduling strategies are now standard. Moreover, since the forward process is fixed and non-parametric, it does not introduce additional variables to optimize, making the training dynamics easier to analyze and more stable.

A further advantage over GANs is the absence of phenomena such as mode collapse: the model does not learn a deterministic mapping from noise to data space, but a conditioned distribution that is progressively refined, favoring more complete coverage of the data distribution [11]. Despite the theoretical solidity of DDPMs, the standard generation process requires T network evaluations, making sampling relatively slow. To address this limitation, numerous variants and generalizations have been proposed. Among these, Noise-Conditioned Score Networks (NCSN) reinterpret the problem as estimating the score of the data distribution at different noise levels, connecting diffusion models to Langevin dynamics and score-based generative modeling [12]. A further generalization is provided by the continuous formulation through Stochastic Differential Equations (SDEs), which allows describing the diffusion process as a continuous-time system and using advanced numerical solvers to accelerate sampling [12]. Diffusion models today represent one of the most solid and versatile frameworks for image generation. Their probabilistic formulation, combined with stable training and high visual quality, makes them particularly suitable for contexts where reliability, semantic control (i.e., ensuring coherence between the generated content and the conditioning signal, such as a class label or a text prompt), and generalization capability are required.

In the context of this thesis, diffusion models are of particular interest not only for the quality of generated images, but also for the possibility of integrating conditioning and control mechanisms, such as Classifier-Free Guidance, without having to retrain the model. This aspect makes them particularly compatible with industrial and robotic applications, where flexibility and robustness are fundamental requirements.

2.5. Fine Tuning of Diffusion Models

Fine-tuning is today one of the most relevant paradigms for making pre-trained generative models operational in specific applications. The underlying idea consists of starting from a model trained at large scale (capable of representing a very broad data distribution) and refining its parameters on a restricted but specialized dataset, in order to orient the output toward a domain, style, or object family of interest. In text-to-image diffusion models, this adaptation aims to make the pre-trained model learn stylistic coherence, recurring geometry, or semantic coherence constraints between prompt and output, intervening

mainly on the generative components (typically the U-Net) and, in some cases, also on the text encoder [13].

From an operational perspective, fine-tuning can be interpreted as a cycle in which, given an image- caption pair, the model produces a prediction coherent with the training objective (noise prediction in diffusion models) and the weights are updated to reduce a loss function that measures the distance from the desired behavior. In other words, the knowledge learned in pre-training is not "rewritten from scratch", but readapted: the model retains the general ability to synthesize plausible images, but is pushed to place the probability mass in a more restricted region of the image space, compatible with the target domain. A concrete example is shown in the fine-tuning of Stable Diffusion XL for stylized icon generation, where the training process is described as a weight update aimed at increasing alignment with a reference dataset. A schematic representation of the fine-tuning learning cycle is shown in Figure 2.8.

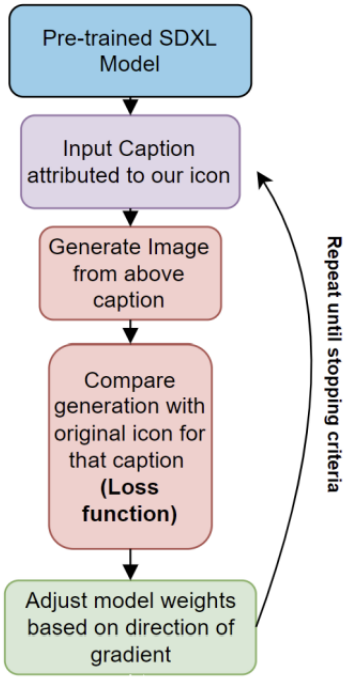


Figure 2.8: High-level schema of the fine-tuning cycle for the text-to-image Stable Diffusion Model

When discussing fine-tuning of a modern diffusion model, it is important to clarify that adaptation is not necessarily an "all or nothing" operation. On the contrary, it is possible to choose which components of the architecture should remain trainable during the fine-tuning phase, leading to very different behaviors in terms of style, fidelity, ability

to follow the prompt, and training stability. In more concrete terms, the main choices typically fall on three blocks: the denoising U-Net, the text encoder, and the attention mechanisms. The most common intervention point is the U-Net (denoiser), as it represents the operational heart of the model: it is the module that, at each time step, receives the noisy image x_t and produces the prediction necessary to progressively remove the noise. Consequently, updating the U-Net weights has a direct impact on the final image rendering. In particular, U-Net fine-tuning tends to effectively modify texture, style, shapes, proportions, and composition, because it intervenes on the actual generative process. For this reason, when the objective is to adapt the model to a specific visual domain (for example, an industrial object with recurring geometries, materials, and coherent details), U-Net adaptation is often the most natural and effective choice. A second component that can be intervened upon is the text encoder, that is, the module that transforms the text prompt into a vector representation then used by the model to condition the generation. In theory, adapting the text encoder as well can increase the system's sensitivity to new concepts or specialized tokens (for example, keywords introduced to describe a component or configuration). However, this choice is more delicate: the text encoder does not only influence "what" is generated, but also how language is interpreted. For this reason, modifying it can alter language–image alignment in a less predictable way and, in some cases, compromise the model's robustness to out-of-distribution prompts. Consequently, in most application pipelines the text encoder is left frozen, or adapted with more conservative strategies compared to the U-Net.

Finally, a very widespread option consists of intervening on attention mechanisms, and in particular on cross-attention, which is the point where textual representations interact with visual features in to the U-Net. This type of adaptation is often considered a good compromise: instead of globally modifying the entire model, the update is applied only to the linear projections on the linear projections that determine how text guides generation. The main advantage is that the model's ability to correctly associate parts of the image with concepts expressed in the prompt is improved (for example, position, color, or presence/absence of a component), without necessarily overturning the general appearance of the generated distribution. For this reason, many practical implementations prefer to selectively update attention projections, especially when the objective is to increase textual control while maintaining good training stability.

In summary, fine-tuning in diffusion models can be seen as a design choice on the degree and type of adaptation desired: the U-Net strongly influences visual rendering, the text encoder acts on language interpretation (but with greater risk), while attention adaptation often allows more targeted conditioning control, with a favorable trade-off between

effectiveness and stability. Despite its effectiveness, full fine-tuning (updating all model parameters) presents structural criticalities, especially when working with large-scale architectures: the computational cost grows rapidly and often becomes incompatible with moderate hardware resources. Furthermore, an additional limitation must be considered: updating the entire model increases the probability that it specializes too much on the target dataset (overfitting problem), reducing flexibility and generalization capabilities. In many scenarios, in fact, a degradation in performance or quality is observed when the model is pushed to replicate too faithfully a small or poorly varied dataset. A related problem, often discussed in literature and practice, is so-called catastrophic forgetting: when a pre-trained model is aggressively adapted on a restricted domain, it can lose part of the general competencies acquired during pre-training. In generative models, this can manifest subtly: images that become more "of the domain" but lose variety, compositional ability, or robustness to out-of-distribution prompts. Precisely for these reasons, many recent research lines prefer parameter-efficient strategies, capable of obtaining competitive adaptation by updating only a small fraction of weights [14]. A further practical aspect, crucial especially in industrial contexts, concerns dataset quality. Unlike large-scale training, where the mass of data compensates for noise and inconsistencies, in fine-tuning a small dataset can amplify systematic defects: overly complex backgrounds, non-centered objects, inconsistent lighting conditions, and ambiguous captions can lead to ineffective or unstable learning. An experimental study on fine-tuning clearly shows that "clean" and focused data improve adaptation effectiveness, while confused data reduce result quality [15].



(a) Base Stable Diffusion



(b) After fine-tuning with LoRA

Figure 2.9: Comparison between generation with base model (a) and fine-tuned model (b). Fine-tuning significantly improves image quality, with more realistic details and a more natural composition. [13]

2.5.1. Low-Rank Adaptation (LoRA)

To overcome the limitations of full fine-tuning, one of the most influential "parameter-efficient" adaptation techniques is LoRA (Low-Rank Adaptation). The key idea is that, when adapting a pre-trained model to a new domain, the necessary weight update need not necessarily be "full rank": in practice it can be well approximated by a low-rank perturbation, which captures the principal directions of the required change without having to re-optimize the entire parameter matrix [14]. This allows preserving the model's general knowledge, drastically reducing both computational cost and the risk of excessive specialization typical of aggressive adaptations.

From a mathematical perspective, LoRA freezes the pre-trained weights and introduces a much smaller trainable parametric correction. If $W_0 \in \mathbb{R}^{d\times k}$ is a weight matrix of the pre-trained model, LoRA rewrites the effective weights as

$$W = W_0 + \Delta W = W_0 + BA, \tag{2.16}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, with $r \ll \min(d, k)$ [14]. In terms of forward pass, this means that the layer output becomes

$$h = W_0 x + BAx. \tag{2.17}$$

The intuition is that BA acts as a "targeted" correction, capable of modifying the model's behavior to the extent necessary for the new dataset, without rewriting the structure learned during pre-training. Typically, B is initialized to 0, or in general the correction is initialized so that the additional contribution is null. In this way, at the beginning of training LoRA does not alter the network in any way: the model starts exactly from the pre-trained behavior, and adaptation emerges as a progressive and controlled deviation. In practice it is common to also introduce a scaling factor to stabilize optimization and make updates comparable as r varies, for example through α/r, obtaining:

$$W = W_0 + \frac{\alpha}{r} BA, \tag{2.18}$$

where α is a hyperparameter [14]. This measure is important because it allows maintaining consistent update "scale", avoiding that a higher rank automatically produces larger updates and therefore less stable training.

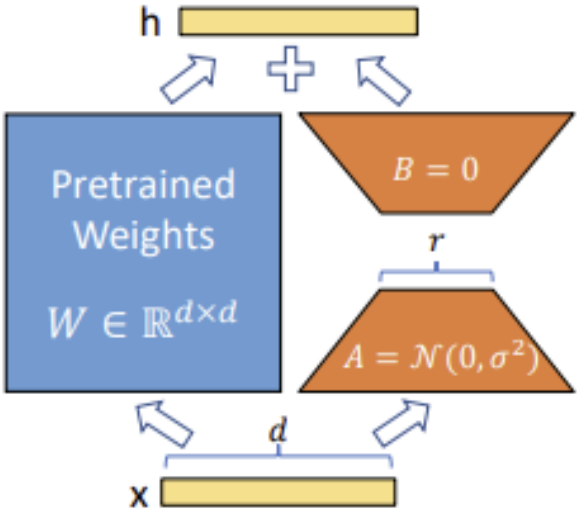


Figure 2.10: LoRA architecture: the pre-trained weights W_0 remain frozen, while adaptation occurs through the low-rank decomposition BA with $r \ll d$. The typical initialization is $B = 0$, $A \sim \mathcal{N}(0, \sigma^2)$. [14]

Figure 2.10 illustrates the reparametrization introduced by LoRA: the frozen weights W_0 are flanked by a trainable low-rank branch that implements the correction $\Delta W = BA$. When moving to diffusion models, LoRA becomes even more interesting because it is not only a way to reduce parameters, but also a way to make adaptation more controllable. In diffusion models the main element that is adapted is often the U-Net denoiser: it is the one that, step after step, determines how noise is removed and therefore directly influences visual rendering, texture, contours, composition, and fidelity to details. However, adaptation is not necessarily uniform: intervening on internal transformations related to attention (particularly in projections governing textual conditioning) tends to improve text-image alignment, making the prompt more incisive in determining configuration and attributes; conversely, intervening more extensively on convolutional blocks can have a more marked impact on perceptual quality, definition, and style. In this sense LoRA is not merely a fine-tuning shortcut: it is also a mechanism through which one can choose where to "let in" adaptation in the model.

This connects naturally to the encoder–decoder structure discussed previously. The U-Net of diffusion models can be read as an encoder–decoder with skip connections that preserve spatial information at multiple scales; LoRA, grafting itself as a residual correction along crucial transformations, allows "bending" the denoising mapping toward the target domain without rewriting the entire network. In other words, the architecture remains unchanged and general knowledge remains available, while adaptation occurs by

updating only a small fraction of weights: this reduces costs, facilitates experimentation, and typically attenuates the risk of overfitting compared to full fine-tuning [14].

LoRA offers relevant advantages not only computationally, but also methodologically, because it makes adaptation of very large models more sustainable and, above all, more controllable. The first benefit is the drastic reduction of effectively trainable parameters: since during training only the A and B matrices of the low-rank update are optimized, the memory requirement and training time are significantly lower compared to full fine-tuning, which instead requires updating the entire set of model weights [14]. This advantage is accompanied by an aspect often decisive in application contexts: at inference LoRA does not necessarily introduce additional latency. Once adaptation is finished, in fact, the update can be "merged" into the original weights through $W = W_0 + BA$, obtaining a functionally equivalent model but identical, in terms of structure and time, to the pre-trained one [14]. The same logic also favors storage efficiency: instead of saving complete checkpoints, it is sufficient to archive only the LoRA modules (that is, the low-rank matrices), making it practical to maintain many adaptation variants, for example different styles, domains, or configurations, with a drastically reduced memory cost [14].

Beyond these "quantitative" aspects, LoRA proves particularly convenient for iterative experimentation. In a real workflow, the possibility of updating only a small portion of parameters allows performing rapid trial cycles: one can vary the rank, modify target layers, update dataset and captions, or compare conditioning strategies without having to retrain or duplicate the entire model. This modularity is one of the main reasons why LoRA is often preferred in application projects, where time and resource constraints make extended full fine-tuning impractical. Supporting these advantages, the original work on LoRA also includes numerical comparisons between different adaptation strategies and number of trainable parameters, highlighting how the low-rank approach allows substantial savings without necessarily compromising quality [14].

A central point in practice is the choice of rank r, which represents a compromise between adaptation capacity and computational cost. In general terms, a higher rank increases the model's ability to absorb complex variations of the target domain, but reduces part of the efficiency benefits; conversely, a rank that is too low can produce insufficient adaptation, with outputs still too "generic" or poorly aligned to the new domain. In most cases, moderate values offer an effective balance, especially when combined with reasoned selection of layers on which to apply LoRA. The literature also discusses various extensions and measures aimed at improving stability and quality while maintaining efficiency, for example by varying initializations, scaling (such as α/r), or strategies for selecting target layers and optimization [16].

In application perspective, it is finally useful to emphasize that the effectiveness of fine-tuning, even when made efficient by LoRA, does not depend solely on the adaptation technique, but on a combination of factors that must remain coherent with each other. In particular, result quality is determined by the interaction between the dataset (quantity, variety, and cleanliness), the adaptation strategy (full fine-tuning or LoRA, and choice of layers to update), and conditioning (coherent and unambiguous captions/prompts). Jointly considering these three elements allows reducing training instability, limiting overfitting on small datasets, and maximizing the model's ability to generalize in a controlled way, while maintaining visual fidelity and robustness with respect to prompt variations.

3 | Methodology

This chapter describes the methodology adopted for the generation of images of EPART industrial control devices using deep learning generative models. The proposed pipeline is motivated by an industrial use case in which controllable and photorealistic images can support data-driven automation, particularly in vision-based robotic assembly. In this context, EPART refers to a modular industrial control unit composed of a rectangular protective case featuring two circular slots that can host different button components (e.g., standard colored buttons and an emergency button). The limited variability of real acquisitions and the cost in terms of time of collecting and annotating large-scale industrial datasets motivate the adoption of generative models to synthesize plausible product configurations under controlled conditions. In particular, three methods are implemented: a Conditional Generative Adversarial Network (cGAN) trained from scratch on a limited and highly structured dataset; a text conditioned diffusion model also trained from scratch and a Stable diffusion model fine-tuned through on Low-Rank Adaptation (LoRA) strategy. These three different generative paradigms will be compared in detail in chapter 4, within the same application.

The objective of the proposed methodology is to achieve a controlled and reproducible text-to-image generation, in which it is possible to clearly specify the main characteristics of the target images (for example, the presence and combination of components in our assembled object). To evaluate the three different generative paradigms within a consistent experimental setting, the assembled EPART control device is adopted as as reference use case. The following sections therefore describe the EPART images dataset and three considered pipelines: a conditional GAN trained from scratch, a diffusion model implemented and trained from scratch, and a fine-tuning procedure based on LoRA.

3.1. Dataset

In this section we present the dataset used for the three generative models. The EPART dataset used in this work consists of real images depicting a modular industrial control device, acquired under controlled conditions and observed from a top-down view. The

device is composed of a black rectangular base (case), on which one or more buttons can be mounted in predefined positions. The considered configurations include three standard buttons, distinguished by color (red, green, and yellow), and an emergency mushroom-type button, characterized by larger dimensions and a distinctive shape.

The images present a uniform light-wood background, stable illumination, and the absence of significant occlusions, in order to reduce irrelevant variability and focus the learning process on the structural differences between configurations. In some instances, the case may be empty (empty configuration), while in others one or two buttons are present, with combinations that include both standard buttons and the emergency button. The dataset includes images of the device in several configurations: an empty case, a case with one or two buttons (including combination of standard and emergency buttons) and buttons without the case.



(a) Case with one button (b) Case with two buttons (c) Empty case (d) Single button

Figure 3.1: Examples of different configurations present in the dataset: case with an emergency button (a), combinations of colored standard buttons (b), empty cases (c), and buttons without a case (d).

Figure 3.1 shows representative examples from the dataset, illustrating different device configurations in terms of the number, type, and arrangement of buttons. As shown, the spatial layout of the components, i.e. their relative positioning within the two circular slots, including alignment, spacing, and left/right placement, remains consistent across image, while the combination of present elements varies. This characteristic makes the dataset particularly suitable for the study of conditional generative models, where the objective is not only the synthesis of realistic images, but also the enforcement of well-defined structural constraints, such as the fixed two-slot geometry of the case, the requirement that buttons appear centered within the circular slots, and the correct left/right occu-

pancy pattern (e.g., one slot filled and one empty, or two buttons).

The original dataset consists of approximately 300 images acquired under controlled conditions. In order to increase the overall dataset size and improve training robustness, data augmentation procedure is applied, generating variants of the original images, for example through small rotations, brightness variations, and mild degradations such as blur, while preserving the semantic structure of the represented configuration (i.e. the arrangement of the buttons).

The final dataset composition is overall well balanced across the main categories, with the number of images per category ranging approximately between 320 and 350 samples. The detailed category distribution is summarized in Table 3.1.

Configuration	**Number of images**	**Percentage (%)**
Empty case	350	6.9
Red button	350	6.9
Green button	350	6.9
Emergency button	350	6.9
Yellow button	348	6.8
Case with green button	336	6.6
Case with red button	336	6.6
Case with red and green button	336	6.6
Case with red and yellow button	336	6.6
Case with red and emergency button	336	6.6
Case with yellow button	336	6.6
Case with green and emergency button	336	6.6
Case with green and yellow button	336	6.6
Case with yellow and emergency button	324	6.4
Case with emergency button	324	6.4

Table 3.1: Distribution of images in the dataset after augmentation

3.2. Conditional GAN

3.2.1. Conditioning Strategy

For the generation of images representing different configurations of the EPART device, producing visually realistic samples alone is not sufficient: it is necessary to explicitly

control the structural configuration of the generated object, in terms of the presence and combination of its components. For this reason, the GAN adopted in this work is formulated in a *conditional* setting, where conditioning information describing the desired configuration is provided to the model during both training and generation. In classical conditional generation, conditioning is often implemented by providing the model with a coarse control signal, such as a one-hot category label or a continuous embedding, which may implicitly correlate with multiple visual properties. In this implementation, instead, the conditioning is expressed through a compact binary attribute vector, where each dimension encodes a specific and interpretable property of the EPART device (e.g., slot occupancy and component presence). This design yields a deterministic, non-ambiguous specification of the desired configuration and enables straightforward post-hoc verification of compliance with the conditioning signal.

To enable controlled and unambiguous generation, each image in the dataset is associated with a vector of binary labels describing the configuration of the control device in terms of the components present. In particular, a 6-dimensional multi-label representation is adopted, defined as

$$\mathbf{y} = [y_{\text{red}}, y_{\text{green}}, y_{\text{yellow}}, y_{\text{emg}}, y_{\text{empty}}, y_{\text{case}}] \in \{0,1\}^6,$$

where the first four elements indicate the presence of the buttons (red, green, yellow, emergency), the fifth element identifies the configuration without buttons (empty), and the last element encodes the presence of the case. Table 3.2 summarizes the semantic meaning of each component of the vector. This choice makes it possible to explicitly separate the structural properties of the object from purely perceptual variations (illumination, background texture, small pose changes), ensuring deterministic conditioning during generation. The configurations used experimentally correspond to a finite set of combinations consistent with the considered assembly process. Compared to text-based conditioning, the multi-label encoding avoids linguistic ambiguities and enables a rigorous definition of the space of admissible configurations, while also facilitating extension to new components through the addition of further dimensions to the vector. However, precisely because it relies on multi-label conditioning, this approach exhibits significant limitations in terms of generalizability.

Index	Component	Meaning
0	red	Presence of the red button
1	green	Presence of the green button
2	yellow	Presence of the yellow button
3	emergency	Presence of the emergency button
4	empty	Absence of buttons (empty case)
5	has_case	Presence of the case

Table 3.2: Semantic mapping of the multi-label vector

3.2.2. Generator

The generator of the Conditional GAN is designed to map a random latent representation, conditioned on the multi-label semantic representation of the desired configuration, into an RGB image of the EPART control device at a resolution of 128×128. The architecture follows a *DCGAN-inspired* paradigm based on progressive upsampling, in which the spatial resolution is gradually increased starting from a compact high-dimensional representation.

The input to the generator consists of the concatenation of a latent variable $\mathbf{z} \in \mathbb{R}^{128}$, sampled from a standard normal noise distribution, and the multi-label vector $\mathbf{y} \in \mathbb{R}^{6}$. The resulting vector, with dimensionality 134, is projected through a fully connected layer into a three-dimensional space of size $8 \times 8 \times 512$.

Starting from this representation, the generator applies a sequence of upsampling blocks, each of which doubles the spatial resolution while progressively reducing the number of channels. For the 128×128 configuration, the generator is organized into four upsampling blocks, each responsible for doubling the spatial resolution. Beginning from an 8×8 feature map, the resolution is increased in stages to 16×16, 32×32, 64×64, and ultimately 128×128.

Each block consists of: a stride-2 transposed convolution, which implements learnable upsampling, Batch Normalization, introduced to improve training stability and reduce sensitivity to initialization, and a LeakyReLU activation with slope $\alpha = 0.2$, which helps maintain gradient flow also for negative activations. By repeating this block design, the generator reconstructs images in a coarse-to-fine manner, progressively increasing spatial detail while preserving global consistency.

The final layer of the generator is a standard convolution that produces a three-channel

(RGB) image. A kernel size of 7×7 is employed to promote greater spatial continuity in the final output. A final sigmoid activation constrains pixel intensities to the range ($[0, 1]$), matching the preprocessing pipeline in which both real and generated images are normalized to the same interval before being passed to the discriminator.

The main design choices of the generator architecture are motivated by training stability and visual quality. The use of Batch Normalization contributes to more stable convergence during training, while the adoption of LeakyReLU reduces the risk of inactive units in the early stages of optimization, i.e., neurons whose outputs remain constantly equal to (or very close to) zero for most inputs. This phenomenon can occur with standard ReLU when a unit receives predominantly negative pre-activations, causing the gradient to be zero and preventing the corresponding weights from being updated (the so-called "dying ReLU" effect). By allowing a small, non-zero slope for negative inputs, LeakyReLU preserves gradient flow and keeps these units trainable during the initial phase of learning. In line with DCGAN design guidelines, dropout is not applied within the generator, as it may introduce visual artifacts and reduce the realism of the generated images.

Layer	**Output shape**	**Kernel / Units**	**Stride**	**Activation**
Input $(\mathbf{z}, \mathbf{y})$	(134)	–	–	–
Dense + Reshape	$(8, 8, 512)$	32768	–	LeakyReLU
Conv2DTranspose	$(16, 16, 256)$	4×4	2	LeakyReLU
Conv2DTranspose	$(32, 32, 128)$	4×4	2	LeakyReLU
Conv2DTranspose	$(64, 64, 64)$	4×4	2	LeakyReLU
Conv2DTranspose	$(128, 128, 32)$	4×4	2	LeakyReLU
Conv2D	$(128, 128, 3)$	7×7	1	Sigmoid

Table 3.3: Generator architecture for the synthesis of 128×128 images

Table 3.3 summarizes the generator architecture adopted for the synthesis of images at 128×128 resolution.

3.2.3. Discriminator

The discriminator evaluates whether an image is both real and consistent with the provided configuration label. Unlike the generator, which receives the conditioning as a vector concatenated to noise, the discriminator integrates label information directly into the spatial domain. The 6-dimensional binary label vector is spatially replicated to match the image dimensions and concatenated as additional channels: for a $128 \times 128 \times 3$ RGB

image, the label becomes a $128 \times 128 \times 6$ tensor, resulting in a $128 \times 128 \times 9$ input. This spatial conditioning enables the discriminator to perform local label-image consistency checks at every pixel location, rejecting images that contradict the specified configuration (e.g., a green button when the label indicates red). By penalizing such inconsistencies, the discriminator forces the generator to produce outputs that are both realistic and semantically aligned with the conditioning input.

The convolutional pathway of the discriminator consists of a sequence of four convolutional blocks with stride 2, which progressively reduce the spatial resolution to obtain a compact representation of size $8 \times 8 \times 512$. The number of channels increases gradually, allowing the model to learn increasingly abstract and discriminative features. Each convolutional layer is followed by Batch Normalization, a LeakyReLU activation function with coefficient $\alpha = 0.2$, and dropout, used as a regularization mechanism to mitigate overfitting.

The final representation is flattened and projected through a fully connected layer to a single scalar value, representing the logit associated with the probability that the input image is real and consistent with the provided conditioning. No activation function is applied to the final output, as the loss function used during training operates directly on logits.

Layer	**Output shape**	**Kernel**	**Stride**	**Activation**
Input (image + labels)	$(128, 128, 9)$	–	–	–
Conv2D	$(64, 64, 128)$	4×4	2	LeakyReLU
Conv2D	$(32, 32, 256)$	4×4	2	LeakyReLU
Conv2D	$(16, 16, 512)$	4×4	2	LeakyReLU
Conv2D	$(8, 8, 512)$	4×4	2	LeakyReLU
Flatten	(32768)	–	–	–
Dense	(1)	–	–	–

Table 3.4: Discriminator architecture for the synthesis of 128×128 images

Table 3.4 summarizes the discriminator architecture, highlighting the progressive reduction of spatial resolution and the increase in the number of channels along the convolutional pathway. The spatial integration of the labels allows the model to jointly evaluate visual realism and structural correctness of the requested configuration.

3.2.4. Loss Function

Training of the Conditional GAN is based on an adversarial mechanism in which the generator and the discriminator are optimized through complementary loss functions.

In this work, Binary Cross-Entropy (BCE) operating directly on the logits produced by the discriminator is adopted, a choice that improves numerical stability during training and avoids issues related to activation function saturation, that occurs when the input to an activation function falls into regions where the gradient becomes extremely small or zero. For example, the sigmoid function outputs values close to 0 for very negative inputs and close to 1 for very positive inputs; in both cases, the derivative approaches zero, causing the vanishing gradient problem that prevents effective weight updates during backpropagation.

The discriminator is trained to distinguish between real and generated images, both conditioned on the same semantic configuration. A labeling convention is used in which real images are assigned label 0, while generated images are assigned label 1. This choice does not alter the behavior of the model, provided that consistency is maintained between the discriminator loss function and the generator objective.

The generator is optimized with the opposite objective, namely to produce images that the discriminator classifies as real with respect to the provided conditioning. During generator updates, the discriminator parameters are frozen and used only as a fixed evaluator. The generator is trained by passing its synthetic images through the discriminator and computing the loss against the real target label (i.e., the label that the discriminator would assign to authentic images). In this way, the generator is encouraged to produce outputs that the discriminator cannot distinguish from real samples, effectively learning to fool it without modifying the discriminator itself. In this way, the generator is encouraged to produce samples that are increasingly realistic and coherent with the requested configuration.

In order to improve training stability, a slight perturbation of the target labels is applied during discriminator updates by adding a small uniform noise to the binary values used in the loss. This technique prevents the discriminator from becoming overly confident in its predictions and contributes to maintaining a stable balance between the two adversarial networks, reducing the risk of training instability.

3.2.5. Training strategy and configuration

The training of the Conditional GAN is designed to ensure stability and to maintain a proper balance between the generator and the discriminator, which are particularly critical aspects in adversarial architectures. The optimization procedure and the training hyperparameters are chosen by considering both the guidelines reported in the literature and the specific characteristics of the EPART dataset and the adopted architecture.

Both networks are optimized using the Adam algorithm, chosen for its effectiveness in handling noisy gradients and non-stationary learning dynamics. For both the generator and the discriminator, the Adam optimizer is used with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ (instead of the default $\beta_1 = 0.9$). Here, β_1 denotes the exponential decay rate for the first-moment estimate (i.e., the momentum term) in Adam. Setting β_1 to a lower value reduces the momentum effect and promotes a fast response to rapid gradient variations during adversarial training, as commonly suggested in DCGAN implementations.

The learning rates of the two networks are initially set to different values, with a learning rate of 2×10^{-4} for the generator and 1×10^{-4} for the discriminator, resulting in a 2:1 ratio that ensures sufficient discriminator learning capacity in the early stages. This asymmetry was introduced to prevent the discriminator from learning too quickly during the early stages of training, a condition that could hinder the generator's learning process. Starting from epoch 40, to mitigate dynamics in which the discriminator became excessively strong with respect to the generator, the discriminator learning rate is further reduced to 1×10^{-5}, adopting a 20:1 ratio with respect to the generator learning rate, while keeping the generator learning rate unchanged. This choice aimed to weaken the discriminator learning speed and preserve a more balanced adversarial dynamic, promoting more stable generation in the later stages of training.

Training is performed using a batch size of 16 images, set due to memory constraints, and a total of 300 epochs, with the best-performing checkpoint selected at epoch 180. This configuration allows the model to undergo an initial stabilization phase followed by a progressive refinement phase of the learned visual structures, while maintaining an adequate trade-off between optimization stability and computational cost. Images were used at a resolution of 128×128 pixels, selected as a compromise between computational cost and the model's ability to capture the global structure of the device. Overall, training required approximately two hours on a NVIDIA Tesla P100 GPU.

The training process is implemented through a custom loop, which enables explicit control over the update order of the networks and the handling of conditioning. In particular, for each batch of real data, the discriminator is updated using both real images and generated images conditioned on the same labels, followed by an update of the generator while keeping the discriminator frozen. The complete configuration of the training hyperparameters is summarized in Table 3.5.

Parameter	Value
Optimizer	Adam
β_1	0.5
β_2	0.999
Generator learning rate	2×10^{-4}
Discriminator learning rate (phase 1, epochs 1–40)	1×10^{-4}
Discriminator learning rate (phase 2, epoch 40+)	1×10^{-5}
Batch size	16
Number of epochs	300
Best checkpoint	Epoch 180
Latent distribution	$\mathcal{N}(0, I)$
Latent vector dimension	128
Resolution	128×128
Random seed	fixed
Hardware	NVIDIA Tesla P100 GPU
Training time	$\sim$2 hours

Table 3.5: Summary of training hyperparameters for the Conditional GAN

The architectural and training choices described in this section define a fully supervised baseline for the controlled generation of EPART images. In the following chapters, this approach will be compared with diffusion models and parameter-efficient fine-tuning techniques, highlighting conceptual and operational differences.

Once training is completed, the Conditional GAN is used to generate synthetic images from binary conditioning vectors. The inference protocol is designed to systematically evaluate the model's ability to produce images corresponding to the requested configurations.

Operationally, the trained generator takes as input a latent noise vector sampled from $\mathcal{N}(0, I)$ concatenated with a 6-dimensional binary label vector encoding the desired configuration. To ensure reproducibility, the random seed is fixed for all experiments, isolating the effect of conditioning from stochastic variability.

The discrete nature of the conditioning mechanism enables exhaustive evaluation: all 15 configurations can be explicitly requested by specifying the appropriate binary vector, allowing precise verification of attribute adherence. Multiple samples per configuration are generated to assess intra-category consistency.

A key limitation is the lack of flexibility: generating configurations not explicitly encoded in the training set (e.g., unseen button colors) is structurally impossible without retraining the model with an extended label schema, representing a fundamental trade-off compared to text-conditioned models.

The results obtained through this protocol are presented and analyzed in the following chapter, alongside those of the diffusion-based models.

3.3. Text Conditioned Diffusion model from scratch

In the previous section, the use of Generative Adversarial Networks (GANs) for the generation of synthetic images of the EPART device was investigated. While the GAN-based approach enabled the model to capture the visual distribution of the target objects, it also revealed several limitations with respect to the objectives of this work. In particular, adversarial training proved to be sensitive to instability issues, making the learning process difficult to control and reproduce. Moreover, the model was label-conditioned, which constrained its ability to interpret complex textual descriptions and limited its capacity to generalize to configurations not explicitly observed during training.

In response to these limitations, this section introduces an alternative generative approach based on Denoising Diffusion Probabilistic Models (DDPMs). Diffusion models operate according to an iterative and non-adversarial generative paradigm, which generally leads to more stable optimization dynamics and greater flexibility in the conditioning mechanism. In particular, the native integration of textual conditioning makes diffusion models especially suitable for image generation from linguistic descriptions, a central requirement in the application scenario considered in this thesis.

The proposed diffusion model is designed to generate images of the EPART device from textual prompts describing the presence, type, and color of the components. The objective is not only to reproduce configurations already present in the dataset, but also to promote the model's ability to generalize, enabling the generation of novel combinations of components consistent with the provided textual descriptions. This capability is particularly relevant in collaborative robotics applications, where flexibility in defining final configurations represents a fundamental requirement.

The remainder of this section presents the implementation of the adopted diffusion model, detailing its architecture, the textual conditioning mechanism, and the training and generation procedures.

3.3.1. Architecture of the Diffusion Model

The diffusion model adopted in this work is implemented as a denoising network based on a conditional *U-Net* architecture, designed to operate on RGB images with a resolution of 64×64. The architecture receives as input a noisy image x_t, obtained from forward diffusion process at timestep t (see chapter 2, section 2.4.1), and outputs an estimate of the noise $\hat{\epsilon}$ added to the original image. This formulation reframes the generative task as a well-conditioned regression problem, thereby promoting training stability.

As previously discussed in chapter 2, the U-Net follows a symmetric encoder–decoder structure, in which the spatial resolution of the image is progressively reduced through a series of downsampling blocks and subsequently reconstructed through upsampling blocks. Skip connections between corresponding encoder and decoder blocks preserve spatial information across multiple scales, improving reconstruction quality and facilitating gradient flow during training.

From an implementation perspective, the model is based on the `UNet2DConditionModel` class from the *Diffusers* library (Hugging Face), which represents a widely adopted standard for text-conditioned diffusion models. Given the limited size of the EPART dataset (approximately 5,000 images), the architecture is appropriately scaled down compared to configurations typically employed in large-scale models. In particular, the number of channels in the convolutional layers is reduced in order to limit the overall parameter count and mitigate the risk of overfitting.

The structure of the model is summarized in Table 3.6.

Stage / Block	**Output shape**	**Block type**	**Conditioning**	**Notes**
Input (x_t, t)	$(64, 64, 3)$	–	Time embedding	Noisy RGB image at timestep t
Down Block 1	$(32, 32, 64)$	DownBlock2D	t	Convolution + downsampling
Down Block 2	$(16, 16, 128)$	DownBlock2D	t	Convolution + downsampling
Down Block 3	$(8, 8, 256)$	CrossAttnDownBlock2D	t + text	Cross-attention with text embeddings
Down Block 4	$(4, 4, 256)$	CrossAttnDownBlock2D	t + text	Conditioning at low resolution
Bottleneck	$(4, 4, 256)$	MidBlock2D (+ attn)	t + text	Highest abstraction level
Up Block 1	$(8, 8, 256)$	CrossAttnUpBlock2D	t + text	Upsampling + skip connections
Up Block 2	$(16, 16, 256)$	CrossAttnUpBlock2D	t + text	Upsampling + skip connections
Up Block 3	$(32, 32, 128)$	UpBlock2D	t	Upsampling + skip connections
Up Block 4	$(64, 64, 64)$	UpBlock2D	t	Upsampling + skip connections
Output $(\hat{\epsilon})$	$(64, 64, 3)$	Conv2D	–	Predicted Gaussian noise

Table 3.6: Conditional diffusion U-Net architecture adopted for EPART images generation (64×64)

As shown in the table, cross-attention layers are introduced in the lower-resolution blocks, where spatial representations are more abstract. This design choice allows the model to integrate textual information more effectively, influencing the global structure of the generated image rather than low-level details. Conversely, higher-resolution blocks rely exclusively on convolutional operations, focusing on spatial coherence and fine-grained detail reconstruction.

The resulting model exhibits a computational complexity compatible with *from scratch* training on small-scale datasets and consumer-grade hardware, while preserving the fundamental architectural components of modern text-conditioned diffusion models. Consequently, the proposed architecture represents an effective compromise between expressive capacity and overfitting control, making it well suited to the methodological investigation conducted in this thesis.

3.3.2. Conditioning strategy

Unlike the GAN-based approach discussed in the previous section, in which conditioning was implemented through a vector of discrete labels (multi-label conditioned), the diffusion model adopted in this work integrates a text-conditioning mechanism, where the textual prompt represents the conditioning input.

The adoption of text-conditioning does not necessarily imply a significant improvement in the model's generalization capabilities, which remain strongly influenced by the limited size of the dataset and by the *from scratch* training regime. Rather, the main advantage of this approach lies in the possibility of achieving a more direct and stable semantic control over the generative process, allowing the model to interpret articulated textual descriptions instead of relying on a restricted set of predefined labels.

Figure 3.2 provides a schematic representation of the adopted text-conditioning mechanism, highlighting the role of textual embeddings and cross-attention layers within the U-Net during the iterative denoising process. The figure illustrates how the textual prompt is transformed into a numerical representation (embedding) and injected into the denoising network, enabling the textual information to guide the generation process across the different steps of the reverse diffusion procedure.

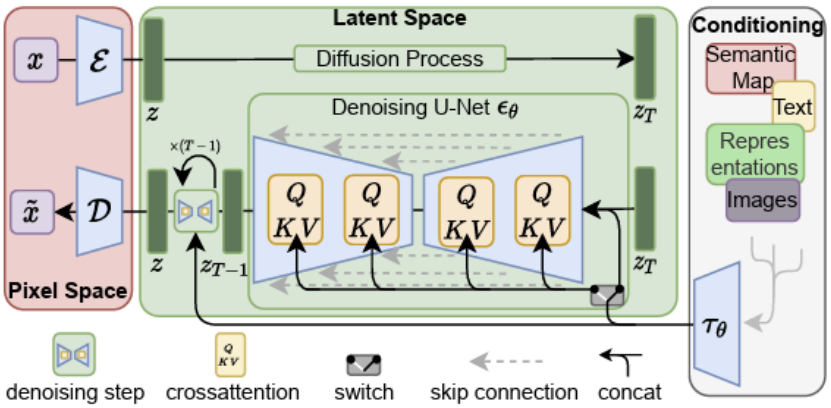


Figure 3.2: Text-conditioning mechanism in diffusion models. The prompt is encoded into semantic embeddings and injected into the U-Net denoising network through cross-attention layers (Q, K, V). At each denoising step, the textual representation conditions the reconstruction of the image, guiding the generation process toward the desired semantic content. [17]

Operationally, the textual prompt is processed by a pre-trained text encoder based on CLIP (*Contrastive Language–Image Pre-training*). The text is first tokenized, meaning that the input sentence is split into smaller units called tokens (typically words or sub-words) and converted into numerical identifiers that can be processed by the neural network; and subsequently transformed into a sequence of dense embeddings that encode the semantic information contained in the description. The text encoder is kept frozen during training of the text conditioned DDPM, treating linguistic information as a stable external constraint, while the optimization process involves exclusively the U-Net-based denoising network.

The integration between text and image is achieved through cross-attention layers implemented within the U-Net architecture. In each cross-attention layer, the internal visual feature maps of the model are modulated as a function of the textual embeddings, allowing the denoising process to be dynamically guided by the content of the prompt. In this way, the model is able to associate textual concepts such as button type, color, or the presence of the case with specific regions of the image, improving the alignment of the final output with the provided description.

The main components involved in the text-conditioned generation mechanism and their role within the pipeline are summarized in Table 3.7. The table highlights how textual conditioning is introduced in a structured and modular manner, without altering the general formulation of the diffusion model, but acting directly on the internal representations

of the U-Net during the iterative denoising process.

Component	Configuration	Role in the pipeline
Prompt	Structured EPART caption	Specifies desired components and case
Tokenizer	CLIP tokenizer	Converts text into token IDs
Text encoder	CLIP text encoder (frozen)	Produces semantic text embeddings
Embedding dimension	512	Conditioning dimension for cross-attention
Conditioning injection	Cross-attention layers in U-Net	Modulates image features during denoising
U-Net input	Noisy image x_t and timestep t	State to be denoised at diffusion step t
U-Net output	Predicted noise $\hat{\epsilon}$	Used to compute the denoising update

Table 3.7: Components of the text-conditioned Diffusion model

Within the EPART image-generation framework, the use of a pre-trained text encoder makes it possible to avoid the definition of a narrow dataset-specific vocabulary, which would otherwise limit the expressive capacity of the model. Even when using simple and highly structured textual descriptions, the cross-attention mechanism enables the exploitation of a richer semantic representation compared to multi-label based conditioning (section 3.2.1, making it more suitable for scenarios in which textual descriptions represent the primary interface between the user and the generative system.

3.3.3. Training strategy and configuration

Once the model architecture and the text-conditioning mechanism have been defined, the diffusion model is trained by learning to predict the Gaussian noise added to real images during the forward diffusion process. The training follows the standard paradigm of Denoising Diffusion Probabilistic Models, in which the model learns to progressively invert the image degradation process through a sequence of text-conditioned denoising steps.

During training, at each iteration a timestep t is uniformly sampled from $\{1, \ldots, T\}$ and a corresponding noisy version of the clean image is generated by applying the forward diffusion process. Concretely, this amounts to perturbing the original sample with Gaussian noise according to the predefined noise schedule (as introduced in chapter 2): early timesteps correspond to mild corruption, whereas later timesteps approach an almost pure-noise regime. The model is then trained to predict (and remove) the injected noise at that timestep, learning to invert the diffusion process across the entire range of noise levels. The U-Net receives as input the noisy image, the timestep, and the textual embeddings associated with the prompt, and produces an estimate of the added noise. The

objective of the training process is to minimize the discrepancy between the predicted noise and the noise actually used in the forward diffusion process, enabling the model to learn a coherent denoising function across the entire temporal trajectory.

The adopted loss function is the Mean Squared Error (MSE) between the true noise and the noise predicted by the model. This formulation reduces the training objective to a stable and well-conditioned regression problem, avoiding the typical issues associated with adversarial models (section 3.2.5). As previously anticipated, the optimization process involves exclusively the parameters of the U-Net, while the CLIP text encoder is kept frozen, treating the linguistic signal as a fixed external constraint.

Training is performed for a limited number of epochs, selected according to the reduced size of the dataset. To mitigate overfitting phenomena and to avoid an unstable selection of the final checkpoint, an early stopping strategy based on a smoothed training loss is adopted. Here, instability in model selection refers to the fact that, when the validation set is small, the validation loss can exhibit high variance and fluctuate markedly across epochs/steps. As a consequence, selecting the "best" model by the minimum validation loss may become unreliable, since the chosen checkpoint can be driven by stochastic oscillations (e.g., batch sampling noise) rather than by a consistent improvement of the generative model. By relying on a smoothed training-loss signal, the selected checkpoint corresponds to a more robust model state that better balances learning progress and stability, without depending on a noisy validation loss.

Overall, the training procedure of the diffusion model proves to be more stable than the GAN-based approach, due to the absence of adversarial dynamics and the formulation of the task as a noise regression problem. This property makes the training process more predictable and reproducible, while maintaining a computational complexity compatible with *from scratch* training on small-scale datasets.

To complete the description of the training procedure, Table 3.8 summarizes the main hyperparameters adopted for training the diffusion model. The configuration was selected by prioritizing optimization stability and the prevention of overfitting, in accordance with the size of the EPART dataset and the *from scratch* training regime.

Hyperparameter	Value	Motivation
Image resolution	64×64	Sufficient for EPART objects, reduced memory footprint
Batch size	32	Stable training with limited dataset size
Number of epochs	150	Allows convergence when training from scratch
Optimizer	Adam	Standard choice for diffusion models
Learning rate	1×10^{-4}	Conservative value for stable optimization
LR scheduler	Cosine annealing	Gradual decay of learning rate
Diffusion timesteps (training)	1000	Standard DDPM configuration
Diffusion scheduler (training)	DDPM	Stable noise scheduling during training
Diffusion scheduler (inference)	DDIM	Faster sampling with comparable quality
Inference steps	50	Trade-off between speed and image quality
Gradient clipping	1.0	Prevents exploding gradients
Text encoder	CLIP (frozen)	Stable semantic conditioning

Table 3.8: Training hyperparameters adopted for the diffusion model

Once the training phase is completed, the diffusion model can be used to generate new images from textual descriptions provided by the user. Unlike training, where the model observes real images progressively degraded by noise, the inference phase starts from pure Gaussian noise and iteratively applies the reverse diffusion process to obtain an image consistent with the prompt.

The textual prompt is encoded through the CLIP text encoder, producing the embeddings that act as conditioning signals for the entire denoising procedure. At each iteration, the U-Net receives as input the current state of the image, the corresponding timestep, and the textual conditioning, and predicts the noise to be removed. The inference scheduler is the numerical procedure that controls how the latent sample is updated at each denoising iteration: given the model prediction (e.g., the estimated noise or velocity), it applies the chosen update rule to compute the next latent state and to decrease the noise level according to a predefined schedule. In other words, it defines how large each denoising step is and how the sample transitions from a noisy latent to a clean image representation.

For image generation, a DDIM-based scheduler is adopted. DDIM (Denoising Diffusion Implicit Models) replaces the original stochastic sampling procedure with a more efficient, partially or fully deterministic update rule that follows an implicit reverse process. In practice, DDIM enables the use of substantially fewer inference steps than the $T = 1000$ steps used during training, because it allows the model to traverse the denoising trajectory with larger, more informative jumps while preserving the global semantics imposed by the prompt. This results in a significant reduction in sampling time, typically with only a limited degradation in perceptual quality, making DDIM well suited for evaluation

and iterative experimentation. This choice enables generation times compatible with interactive use of the model, while preserving consistency with the diffusion process learned during training.

The inference process is fully deterministic once the textual prompt, the random seed, and the number of denoising steps are fixed. This property allows the reproduction of generations and the controlled analysis of the effect of textual conditioning on the produced images. Finally, the generated image is mapped back to the original pixel space, meaning that the model output, which is not directly a standard RGB image, is converted into a conventional image representation. Concretely, in latent diffusion pipelines the denoising U-Net produces a latent representation; this latent is then passed through the VAE decoder to obtain an RGB image. The resulting tensor is subsequently de-normalized, i.e., rescaled from the normalized range used during training/inference to valid pixel intensities (typically $[0, 1]$, and then optionally to $[0, 255]$ for saving).

3.4. Fine Tuning Low-Rank Adaption (LoRA)

3.4.1. Fine-tuning of a Pre-trained Diffusion Model

The fine-tuning of a pre-trained diffusion model represents a methodological strategy that is alternative to the training of a diffusion model *from scratch* described in the previous section. While training a model entirely from scratch provides full control over the architecture and learning process, it is heavily constrained by the size and diversity of the available dataset. In this work, the EPART image dataset, although structured and semantically coherent, is relatively limited in size, which restricts the model's ability to learn complex and generalizable visual representations.

Large-scale diffusion models, such as Stable Diffusion XL (SDXL), are trained on massive collections of image–text pairs and therefore incorporate much richer visual and semantic knowledge. The core idea behind fine-tuning is to leverage this pre-existing knowledge and adapt it to the specific domain of EPART objects through a controlled specialization process. In this way, the model benefits both from the generative capabilities of a *foundation model* and from the semantic consistency required by the application domain.

Another critical aspect concerns the computational sustainability of the training process. Training a model such as SDXL from scratch would be impractical, both in terms of hardware resources and computational time. Moreover, a full fine-tuning of all model parameters would entail a high risk of overfitting, with the model potentially memorizing the EPART dataset rather than learning its structural characteristics.

For these reasons, this work adopts an efficient parameter adaptation strategy based on *Low-Rank Adaptation* (LoRA). This approach modifies the behavior of the model by introducing a limited number of additional parameters while keeping the original weights of the pre-trained model frozen. As a result, the model can be specialized to the EPART domain without compromising the general generative capabilities acquired during pre-training.

From a methodological perspective, LoRA-based fine-tuning makes it possible to explore a different trade-off between expressive power, i.e., the capacity of the model to represent and synthesize a wide range of visually plausible configurations with fine-grained details and complex compositions, generalization and semantic control compared to the approaches discussed previously. Whereas the GAN model and the diffusion model trained *from scratch* are strongly constrained by the dataset distribution, the LoRA fine-tuned model occupies an intermediate position between a fully generic model and a fully specialized one. This property makes the approach particularly interesting for evaluating the impact of pre-training on the quality and flexibility of the generative process within the EPART image generation domain.

In summary, the introduction of LoRA fine-tuning for a pre-trained diffusion model addresses three main methodological objectives: improving the visual quality of generated images, preserving the model's generalization capability, and reducing the computational cost of training. The following section describes in detail the construction of the dataset and the design of the textual captions used for the model adaptation process.

3.4.2. Dataset and Captions

In the LoRA fine-tuning of Stable Diffusion XL, the construction of the dataset and the design of the captions represent a particularly relevant methodological step. Unlike the previous approaches, which relied on larger datasets with limited resolution, this phase is conducted on a smaller set of images, approximately 150 high resolution samples of the EPART device. The images are processed at 512×512 pixels, a choice consistent with the native resolution of SDXL and motivated by the need to preserve informative visual details (such as the geometry of the case, the shape of the slots, the proportions of the buttons, and the characteristics of the surface finish).

The limited size of the dataset makes the role of the captions even more critical. In text-to-image models, textual input represents the primary channel of semantic conditioning and directly influences the quality with which the model learns to associate linguistic concepts with visual patterns. In this context, captions cannot be regarded as simple

descriptive metadata; instead, they play a structural role in guiding the adaptation of the pre-trained model to the EPART domain.

For this reason, the captions are designed according to a coherent and repeatable schema, with the goal of making the discriminative elements of the device explicit and ensuring clear and unambiguous textual description. Compared to the captions used for the diffusion model trained *from scratch*, the descriptions in this phase are more specific and richer in detail. This choice is motivated by the need to provide sufficiently informative conditioning to specialize a large-scale foundation model using a limited number of images. At the same time, a relatively regular structure is maintained in order to avoid excessive linguistic variability, which would make learning more difficult and could introduce spurious correlations between text and images.

A distinctive element of the adopted strategy is the introduction of a dedicated domain token: "EPART". In practice, the domain token is a specific word deliberately and consistently included in the training prompts (e.g., *"a photo of EPART product case"*) so that the model can associate it with the target visual concept. Rather than describing an attribute (such as color or shape), this token acts as an identifier that disambiguates the EPART device from generic "cases" present in the pre-training distribution, effectively defining a new concept within the model's text-to-image mapping.

By repeatedly pairing the token with images of the EPART device during fine-tuning, the model learns to interpret "EPART" as a semantic handle that activates the specific geometry and appearance of the target product. This anchoring mechanism stabilizes the association between textual descriptions and the object, and helps confine the adaptation to the intended concept, thereby reducing undesired interference with the general knowledge acquired during large-scale pre-training.

The captions explicitly describe the device configurations in terms of the presence of the case, the number of slots, and the type of buttons. Particular attention is devoted to the handling of the special case of the emergency button, which is consistently distinguished from standard buttons, since this component exhibits specific visual characteristics. Overall, the design of the captions aims to represent in text the attributes that must be controllable during generation, making it clearer to the model which aspects should be preserved and which may vary.

Table 3.9 presents representative examples of the captions used during fine-tuning. As highlighted in the table, the segments in bold correspond to the attributes that convey semantic control over the EPART configuration (e.g., presence or absence of the case, number of slots, type and color of buttons), whereas descriptors related to the acquisition

style (e.g., product photography context and controlled lighting) are kept constant in order to reduce undesired variability in the textual conditioning.

Configuration (EPART)	Caption (with semantic-control segments highlighted)
Empty case (two slots)	a photo of **EPART** product case, **two circular button slots**, **both slots empty**, **black rectangular plastic case**, industrial control panel, product photography, studio lighting, neutral background
Emergency + empty slot	a photo of **EPART** product case, **two circular button slots**, **one large red emergency button**, **one empty slot**, **black rectangular plastic case**, industrial control panel, product photography, studio lighting, neutral background
Emergency + one push button	a photo of **EPART** product case, **two circular button slots**, **one large red emergency button and one green button**, **black rectangular plastic case**, industrial control panel, product photography, studio lighting, neutral background
Two push buttons	a photo of **EPART** product case, **two circular button slots**, **one red button and one yellow button**, **black rectangular plastic case**, industrial control panel, product photography, studio lighting, neutral background
Single button (no case)	a photo of **EPART** product case, **single green button without case**, industrial control panel, product photography, studio lighting, neutral background

Table 3.9: Examples of captions used for SDXL LoRA fine-tuning, with semantic-control segments highlighted in bold.

Similarly, Figure 3.3 shows a set of dataset samples paired with their corresponding captions. The figure makes explicit the correspondence between linguistic descriptions and visual content, highlighting how the textual structure is used to encode, in a controlled and consistent manner, the main variables of the EPART domain. The joint inclusion of the table and the figure thus provides an immediate clarification of the form of the supervised signal supplied to the model during the fine-tuning phase.

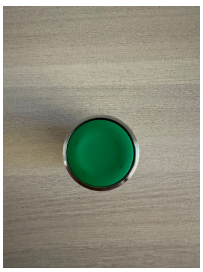
Image	Caption
	a photo of **EPART** product case, **two circular button slots**, **both slots empty**, **black rectangular plastic case**, industrial control panel, product photography, studio lighting, neutral background
	a photo of **EPART** product case, **two circular button slots**, **one large red emergency button**, **one empty slot**, **black rectangular plastic case**, industrial control panel, product photography, studio lighting, neutral background
	a photo of **EPART** product case, **two circular button slots**, **one large red emergency button and one green button**, **black rectangular plastic case**, industrial control panel, product photography, studio lighting, neutral background
	a photo of **EPART** product case, **two circular button slots**, **one red button and one yellow button**, **black rectangular plastic case**, industrial control panel, product photography, studio lighting, neutral background
	a photo of **EPART** product case, **single green button without case**, industrial control panel, product photography, studio lighting, neutral background

Figure 3.3: Examples of EPART training samples paired with their corresponding captions. The structured text explicitly encodes the configuration (slots, button type/color, and case), providing a stable conditioning signal during LoRA fine-tuning.

3.4.3. Stable Diffusion Architecture and LoRA Integration

Stable Diffusion XL (SDXL) is a text-to-image model based on diffusion models that operates in the latent space rather than directly in pixel space. This characteristic is central to its effectiveness: instead of applying the diffusion process to high-resolution images, the input image is first compressed into a latent representation through a Variational Autoencoder (VAE). Generation then takes place in this compact space, and the resulting latent representation is finally decoded back into an RGB image. This formulation significantly reduces computational cost while preserving visual quality.

From an architectural perspective, SDXL consists of three fundamental components: a text encoder, responsible for transforming the prompt into a dense semantic representation; a latent U-Net, which implements the denoising process and represents the core of the generative model; and a VAE, used to encode and decode images in the latent space. During inference, starting from Gaussian noise in the latent space, the U-Net progressively removes noise according to a reverse diffusion schedule (as previously explained in 2.4.2), guided by the textual conditioning produced by the text encoder. Textual information is integrated into the U-Net through cross-attention mechanisms, which allow internal visual features to be dynamically modulated as a function of the prompt content.

The use of SDXL as a base model provides a methodological advantage over models trained from scratch: general knowledge about visual concepts (e.g., colors, materials, lighting, and photographic composition) and linguistic concepts (tokenization and prompt semantics) is already embedded in the pre-trained model. However, the model does not possess a domain-specific representation of EPART device and, without adaptation, is unable to reliably generate the structural characteristics of the device (case, slots, proportions, and distinctive details). For this reason, a specialization mechanism is required.

Adaptation to the EPART domain was achieved through LoRA, which enables selective modification of the model's behavior by introducing a limited number of trainable parameters, without updating the full set of SDXL weights. In practice, the original model is kept frozen, and low-rank matrices are introduced to represent additive updates to the weights of specific layers. This choice is particularly suitable for small datasets, as it limits the model's tendency to memorize training examples and instead promotes controlled adaptation.

In diffusion models, LoRA is typically applied to attention layers, which constitute the primary points at which semantic information and visual representations are combined. In the SDXL U-Net, attention appears both as self-attention (to model dependencies within visual features) and cross-attention (to integrate textual conditioning). Injecting

LoRA into these layers enables effective specialization of the model's generative structure, encouraging a more consistent association between the visual patterns of the EPART device and the textual descriptions provided in the training dataset.

From an interpretative perspective, the effect of LoRA can be understood as a low-dimensional "correction" to the behavior of SDXL: the model retains its ability to generate realistic and semantically coherent images while acquiring greater structural consistency with respect to the EPART domain. This makes it possible to distinguish two contributions: the general knowledge of the base model (e.g., handling photographic styles and general concepts such as colors) and the domain-specific knowledge introduced through fine-tuning (case geometry, slot layout, and button shapes).

3.4.4. LoRA Adaptation Strategy

Once the architecture of the base model and the LoRA adaptation strategy have been defined, the next phase consists in configuring the training process. At this stage, choices regarding hyperparameters, image resolution, batch management, and training duration play a central role, as they determine the balance between the model's learning capacity and the risk of overfitting, which is particularly relevant when working with small datasets.

Unlike the diffusion model trained from scratch, in which the entire model is optimized, LoRA-based fine-tuning restricts parameter updates to a limited subset, enabling the use of lighter and more stable training configurations. Nevertheless, the high complexity of Stable Diffusion XL and the resolution of the images impose computational constraints that directly influence the choice of hyperparameters.

Inthis work, fine-tuning is conducted on images with a resolution of 512×512 pixels, consistent with the native resolution of SDXL. This choice allows full exploitation of the capabilities of the pre-trained model and preserves visually relevant details of the EPART device, but it also entails significant GPU memory consumption. For this reason, the batch size was set to 1, representing a necessary rather than optimal choice.

To compensate for the small batch size, a gradient accumulation strategy is adopted, accumulating gradients over 4 consecutive steps. As a result, the effective batch size is equal to 4, improving optimization stability without increasing memory consumption. This solution is widely used in the fine-tuning of large models on limited hardware and small datasets.

The learning rate is set to 1×10^{-4}, a conservatively chosen value to avoid overly aggressive updates of the LoRA parameters. A higher learning rate could lead to excessively rapid

adaptation, with a consequent loss of general generative capabilities, whereas a lower value would slow down the specialization process on the EPART domain.

Training duration is defined in terms of a maximum number of steps, set to 800. This value is chosen empirically by observing the qualitative evolution of generated images during training and considering the size of the available dataset. The objective is not to achieve full convergence, but rather to identify a balance between learning the structural characteristics of the EPART domain and preserving the model's generalization capability.

During training, the text encoder is kept frozen. This choice is motivated by the desire to preserve the semantic structure of the linguistic space learned during SDXL pre-training. Updating the text encoder in the presence of a limited dataset could introduce semantic drift, altering the relationship between textual tokens and visual concepts and compromising the model's ability to correctly interpret prompts not present in the training dataset.

Training is performed in mixed precision (fp16) to reduce memory consumption and improve computational efficiency. In addition, an 8-bit version of the Adam optimizer is used, further reducing GPU memory usage without compromising optimization stability. Together, these choices define a training pipeline specifically designed for fine-tuning SDXL on a narrow domain under realistic computational constraints.

The configuration of the hyperparameters adopted for LoRA fine-tuning is summarized in Table 3.10. As shown in the table, the selected values reflect a conservative strategy aimed at limiting the complexity of the adapted model and ensuring training stability in the presence of a small dataset.

Category	Parameter	Value
Dataset	Number of images	155
Resolution	Image size	512×512
LoRA	Rank (r)	16
LoRA	Alpha (α)	8
Optimization	Learning rate (U-Net)	1×10^{-4}
Optimization	Learning rate (text encoder)	0 (frozen)
Optimization	Batch size	1
Optimization	Gradient accumulation	4
Training	Max training steps	800
Training	Text encoder	Frozen
Training	Numerical precision	fp16
Inference	Guidance scale	7.5–10

Table 3.10: Hyperparameter configuration for LoRA fine-tuning of Stable Diffusion XL.

The hyperparameters reported in Table 3.10 are specifically designed to maximize the model's adaptation to the EPART image dataset while preserving the general knowledge acquired during SDXL's pre-training. This configuration prioritizes rapid specialization on the target domain: the fine-tuning focuses exclusively on learning the visual characteristics of EPART image products (geometric structure, component layout, material textures) while leveraging the pre-trained model's existing knowledge of colors, lighting, and photographic composition. The result is a model strongly adapted to the industrial domain without requiring the massive datasets and computational resources necessary for training from scratch. In particular, the choice of a moderate rank and a relatively low value of α reflects the intention to limit model adaptation, avoiding excessive specialization on the training images. At the same time, the use of a conservative learning rate and a gradient accumulation strategy contributes to stabilizing the optimization process, compensating for the small batch size.

The decision to keep the text encoder frozen follows the same methodological rationale: preserving the semantic structure of the pre-trained model while focusing adaptation on the visual components of the U-Net. Overall, the hyperparameter configuration highlights an approach oriented toward control and robustness, consistent with the exploratory nature of LoRA fine-tuning in the context of EPART image generation.

Once the fine-tuning stage is completed, the LoRA-adapted model is used to generate

synthetic images from textual prompts. The inference phase is designed as a controlled experimental protocol, with the goal of systematically assessing the model's behavior along three main dimensions: structural adherence to the EPART domain, the ability to interpret textual conditioning, and generalization to configurations not observed during training.

Operationally, the Stable Diffusion XL base model is loaded together with the LoRA weights corresponding to the checkpoint selected as the best-performing during training. To enable a direct comparison between the adapted model and the pre-trained baseline, the pipeline is also configured to generate images with LoRA enabled or disabled, while keeping prompts and inference parameters unchanged. This design makes it possible to isolate the specific contribution of LoRA adaptation to the representation of the EPART domain.

Image generation is carried out by fixing a set of inference parameters, including the number of diffusion steps, the guidance scale, and the pseudo-random generator seed. Keeping these parameters constant reduces stochastic variability and allows the observed differences in generated images to be attributed primarily to the model state (i.e., the learned weights at that specific training checkpoint) and to the content of the textual prompt. As a result, the evaluation becomes more robust and easier to interpret.

To analyze the model's behavior, three groups of textual prompts are defined. The first group includes configurations present in the training dataset and is used to verify the model's ability to faithfully reproduce learned structures. The second group consists of plausible configurations that are under-represented in the dataset, such as rare button combinations or variations in the slot arrangement. The third group contains prompts that introduce attributes not observed during training, specifically button colors absent from the original dataset. This set of prompts enables an evaluation of generalization and helps distinguish between purely memoristic learning and semantic compositionality.

In particular, the test involving unseen colors represents a central element of the experimental protocol. While the training dataset contains only a limited set of button colors, inference introduces novel color descriptions in order to assess whether the model can combine the learned EPART device structure with chromatic attributes that were not present in the fine-tuning dataset. This test helps clarify the role of the pre-trained model in providing general semantic knowledge and the contribution of LoRA fine-tuning in enforcing domain-specific structural specialization.

Beyond prompt variation, the inference phase is also used for a qualitative comparison between the behavior of the adapted model and that of the base model. Specifically,

while the pre-trained model tends to generate semantically plausible images that lack the distinctive structure of the EPART device, the LoRA fine-tuned model reproduces more consistently the case geometry, the presence of slots, and the type of buttons. This comparison provides a methodological basis for the results analysis presented in the following chapters.

Overall, the adopted inference protocol enables evaluation not only in terms of the visual quality of generated images, but also in terms of semantic control, structural coherence, and generalization capability. In this sense, inference is not treated as a mere generation procedure, but rather as an experimental tool to systematically analyze the effectiveness of LoRA fine-tuning in EPART device image synthesis.

The next chapter presents the results obtained with the three models analyzed in the previous sections. The discussion will focus on the main achievements of each approach, the critical issues that emerged, and the improvements introduced, with the goal of providing a systematic comparison across the different generative strategies considered in this work.

4 | Results and Evaluations

This chapter presents the experimental results obtained with the three generative models developed for the EPART device image synthesis: a Conditional GAN trained from scratch as a baseline model, a text-conditioned diffusion model trained from scratch, and a fine-tuning strategy based on LoRA applied to a pre-trained diffusion model. The chapter also provides a comparative analysis of the results obtained with the three developed methods.

The main objective of the evaluation is twofold. On the one hand, it aims to measure the visual quality and fidelity of the generated images with respect to the distribution of real images; on the other hand, it seeks to analyze the effectiveness of the conditioning mechanisms, both in terms of control over product configurations (presence of the case, button combinations, color attributes) and in terms of semantic coherence between the input and the generated output.

Since the three approaches exhibit profoundly different structural and operational characteristics in terms of architecture, required data volume, image resolution, and conditioning modality, the evaluation cannot rely on a single metric. It is therefore necessary to adopt a set of complementary indicators capable of capturing different aspects of generative quality: distributional similarity, structural similarity, and semantic alignment between input and output.

To ensure a coherent comparison among models with different architectures and conditioning strategies, a common evaluation protocol is defined.

The evaluation is conducted along three main dimensions:

- Global quality and diversity of the generated images, understood as the model's ability to reproduce the statistical distribution of real images.
- Structural fidelity with respect to real images, namely the degree of perceptual similarity between real and synthetic images belonging to the same category, i.e. belonging to the identical EPART configuration.
- Effectiveness of conditioning, evaluated as the model's ability to respect the infor-

mation provided as input, in the form of multi-lable vectors (for the GAN) or textual descriptions (for diffusion-based models).

4.1. Evaluation Metrics

To robustly evaluate the three generative models across the different qualitative dimentions introduced above, relying on a single quantitative metric is not sufficient. Generative models may achieve strong performance along one dimension while performing poorly along another: for instance, some models may produce highly realistic images but fail to respect compositional constraints specified in the prompt (e.g., generating three buttons when only one was requested), while others may accurately follow the textual description but exhibit evident visual artifacts such as blurred edges, unnatural textures, or incorrect geometric proportions. For this reason, we adopt a set of complementary metrics, each designed to measure a different aspect of generative quality. Specifically, we use the Fréchet Inception Distance (FID), the Structural Similarity Index (SSIM), and the CLIP Score, which are among the most established and widely used choices for evaluating modern generative models [18–21].

4.1.1. Fréchet Inception Distance (FID)

The Fréchet Inception Distance (FID) was introduced by Heusel in 2017 and has rapidly become one of the most widely adopted metrics for assessing the quality of image generative models [18]. The core idea is straightforward: instead of comparing real and synthetic images at the pixel level—where even minimal variations can lead to large discrepancies—the comparison is performed in a high-level feature space extracted by a pre-trained neural network, typically Inception-v3 trained on ImageNet. In practice, each image is mapped to a vector that captures semantic and structural information (shapes, textures, patterns) and is less sensitive to individual pixels.

Given two sets of images, one of real images and one of generated images, each image is mapped to a feature vector (typically using the activations of the penultimate layer, with dimensionality 2048). The feature distribution of each set is assumed to be well approximated by a multivariate Gaussian, described by a mean $\boldsymbol{\mu}$ and a covariance $\boldsymbol{\Sigma}$. FID then measures the distance between these two Gaussians using a formulation equivalent to the 2-Wasserstein distance for Gaussian distributions:

$$\text{FID} = |\boldsymbol{\mu}_r - \boldsymbol{\mu}_g|_2^2 + \text{Tr}\left(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_g - 2(\boldsymbol{\Sigma}_r\boldsymbol{\Sigma}_g)^{1/2}\right) \tag{4.1}$$

where $\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r$ and $\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g$ denote the mean and covariance of the feature vectors extracted from real and generated images, respectively [18].

An FID value of 0 is obtained (ideally) when the two distributions match perfectly, i.e., when generated images become indistinguishable from real ones in the chosen feature space. In practice, lower FID indicates better performance. Importantly, FID is sensitive to both quality and diversity: a model that produces very realistic images but with limited variability tends to yield covariance statistics that differ from those of the real data, thereby worsening the score.

In the literature, FID is widely regarded as a de facto standard for comparing generative models, particularly GANs and diffusion models, and it is commonly reported in standard benchmarks. In many experimental settings, it also tends to align with human judgments of realism and variety, meaning that improvements in FID often reflect images that are more realistic and whose feature distribution more closely matches that of the real data [18].

Despite its popularity, FID presents some limitations. A well-known issue is that FID estimated from a finite number of samples can be biased; moreover, the bias depends on the evaluated model, so comparing two "raw" FID values can be misleading because part of the difference may stem from statistical estimation error (i.e., inaccurate estimation of distribution parameters from limited samples) rather than from true differences in image quality [22]. A second limitation concerns the Gaussian assumption: collapsing feature distributions into a single Gaussian can be an overly strong simplification when data are complex or highly multimodal. In addition, since the encoder is typically Inception trained on ImageNet, the metric may become less representative when the target domain deviates from ImageNet-like content, or when assessing modern text-to-image systems that generate highly diverse scenes and concepts. For these reasons, recent works have proposed variants of FID or the use of alternative embeddings (e.g., CLIP) coupled with more robust statistical distances [23]. In summary, FID is extremely useful to capture global quality and diversity, but it should not be treated as an absolute measure, especially with limited samples or in domain-specific settings.

4.1.2. Structural Similarity Index (SSIM)

The Structural Similarity Index (SSIM) is a metric introduced by Wang in 2004 in the context of Image Quality Assessment [19]. Unlike FID, which compares distributions, SSIM compares two images at a time and aims to estimate their perceptual similarity. The underlying motivation is that metrics such as MSE/PSNR do not reflect human

perception well: two images may have the same squared error while appearing visually very different. SSIM instead attempts to quantify similarity by focusing on the idea that perceived quality depends primarily on local structures (edges, patterns, textures), rather than on isolated pixel differences [19].

SSIM is typically computed over local sliding windows across the image and, for each window, compares three components: luminance (comparison of means), contrast (comparison of standard deviations), and structure (correlation/covariance between local patterns). In its most common form, these components are combined as follows:

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{4.2}$$

where x, y are the local windows extracted from the two images, μ_x, μ_y are local means, σ_x^2, σ_y^2 local variances, σ_{xy} the covariance, and C_1, C_2 are small positive constants added to prevent division by zero and numerical instability when comparing regions with very low luminance or contrast [19]. The final SSIM score for an image pair is obtained by averaging the window-level values.

SSIM equals 1 when two images are identical (or nearly identical) and decreases as edges and local structures change; in extreme cases it can even become negative. In practice, SSIM is useful for assessing how well a reconstructed image preserves the local structural patterns of the original. Accordingly, it is widely used in compression, denoising, super-resolution, and more generally in reconstruction tasks where a reference image is available and considered the correct target [19].

In the context of generative models, SSIM is a delicate metric because it assumes that the two images being compared depict the same content and differ mainly through distortions. If a generative model produces a plausible image that does not correspond pixel-by-pixel to any real image, SSIM may be low without necessarily implying poor quality. Moreover, SSIM is sensitive to small misalignments: minor translations or rotations can substantially lower the score, even if they are nearly irrelevant to a human observer [19]. For this reason, modern literature often complements SSIM with perceptual metrics based on deep features (such as LPIPS), since a model can obtain a high SSIM by producing overly smooth, low-detail outputs, while a more realistic output that is not identical to the reference may be penalized [24].

Within this thesis, SSIM is therefore used as an indicator of structural fidelity in controlled comparisons: for each generated image, a real image belonging to the same EPART configuration is selected as reference (category-level pairing). The 15 categories correspond

to the distinct EPART configurations introduced in Section 3.1: for instance, all images depicting a black rectangular case with one red button in the left slot and one empty slot constitute a single category, regardless of whether they are real photographs or generated. Given its resolution-sensitivity and its tendency to penalize desirable variability, SSIM is always interpreted jointly with FID and the other metrics, particularly when models operate at different resolutions or when the objective is to generate plausible variation rather than replicate existing images.

4.1.3. CLIPScore

CLIPScore is based on the idea of using CLIP as an automatic judge of text–image alignment. CLIP (Contrastive Language–Image Pretraining) is an OpenAI model trained on large-scale image–text pairs with the objective of mapping images and descriptions into a shared embedding space: semantically coherent texts and images are mapped to nearby vectors [21]. Hessel introduced CLIPScore as a reference-free metric to evaluate the quality of captions; the same principle naturally extends to text-to-image models by comparingthe input text with the generated image [20].

Given a prompt T and an image I, the cosine similarity between the CLIP embeddings of the image and the text is computed as:

$$\text{CLIPScore}(I, T) = sim(E_I, E_T) = \frac{E_I \cdot E_T}{|E_I||E_T|} \tag{4.3}$$

where E_I and E_T denote the normalized CLIP embeddings of the image and the text, respectively [20, 21]. A higher score indicates that CLIP judges the image as coherent with the text, meaning that the generated content follows the textual conditioning.

CLIPScore has rapidly become a central metric for text-to-image models because it addresses a limitation of FID and SSIM: neither metric directly indicates whether the model is truly following a prompt. CLIPScore instead provides an automatic, reference-free measure of how well the image matched the prompt's content, and is particularly suitable for comparisons across multiple prompts and for evaluating compositional capabilities [20, 21].

CLIPScore is also not without shortcomings. A key limitation is that CLIP can behave similarly to a "bag-of-words" model: it can recognize objects and attributes well, but may struggle with precise relations, negations, or counting [25]. Moreover, CLIPScore does not measure visual quality: an image may be unrealistic yet obtain a high score if CLIP still recognizes the main concepts. For this reason, it is good practice to complement CLIPScore with quality/distribution metrics such as FID, in order to disentangle prompt

adherence from "perceptual realism" [20, 23].

4.1.4. Final remarks on combined use

In summary, the selected metrics are complementary and capture different dimensions of generative quality: FID effectively reflects global adherence to the real distribution (both quality and diversity), SSIM provides an indication of structural fidelity when controlled comparisons are available, and CLIPScore measures semantic text–image alignment for prompt-conditioned models [18–20].

Since each metric has specific limitations, the interpretation of the results in the following sections relies on a joint reading of the matrics: discrepancies across metrics are often informative and help identify meaningful trade-offs. For example, a model may achieve a low FID (good overall distribution match) but a relatively low SSIM if it generates higher-resolution images with greater variability in fine details; this does not necessarily indicate poor quality, but rather that the model prioritizes diversity and detail over pixel-level correspondence. Similarly, a high CLIP Score indicates strong adherence to the textual prompt, but it does not guarantee visual realism: an image may be semantically coherent while still exhibiting artifacts or unnatural textures, aspects that are more directly captured by FID [22–24].

Table 4.1 summarizes the main characteristics of the three metrics used, highlighting their objective, typical interpretation, strengths, and limitations.

Table 4.1: Summary of evaluation metrics

Metric	Purpose	Interpretation	Strengths and Limitations
FID [18]	Distance between real and generated distributions in Inception-v3 feature space	**Lower** = better *Indicative thresholds:* < 50: excellent 50–100: good > 200: poor quality	+ Captures quality and diversity + Correlates with human judgment – Sensitive to sample size – Dependent on ImageNet features
SSIM [19]	Local structural similarity considering luminance, contrast, and structure	**Higher** = better *Range:* $[-1, 1]$ 1 = identical > 0.7: high 0.5–0.7: moderate < 0.5: low	+ Perceptually motivated + Useful for same-category comparisons – Strongly resolution-dependent – Penalizes realistic variability
CLIP Score [20, 21]	Semantic alignment between text prompt and generated image via CLIP embeddings	**Higher** = better *Range:* $[0, 1]$ > 0.35: excellent 0.28–0.35: good < 0.28: poor	+ Reference-free + Captures semantic compositionality – Does not measure visual realism – Weak on spatial relations and counting

4.2. GAN Model Results

This section presents the evaluation setup and experimental configuration of the GAN model described in section 3.2. As already introduced in section 3.2, the adopted model is a Deep Convolutional GAN (DCGAN) [26], trained from scratch on the EPART device image dataset. Generation is conditioned through a 6-dimensional binary vector designed

to control the presence or absence of specific device components (red, green, and yellow buttons, an emergency button, an empty case, and a standalone component). This design covers the space of the 15 configurations available in the dataset and provides a useful reference point for comparison with more modern approaches discussed in the following sections.

Generation is controlled via a 6-dimensional binary vector encoding all 15 EPART configurations (see subsection 3.2.1). Table 4.2 illustrates the correspondence between conditioning vectors and generated outputs.

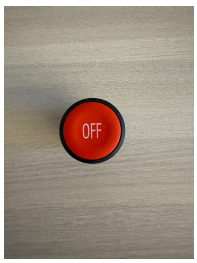
Label Vector	Image	Description
[1,0,0,0,0,0] *Empty case*		Case with both slots empty.
[0,1,1,0,0,0] *Red + Green*		Combination of a red and a green button.
[0,0,0,0,1,0] *Emergency*		Emergency button (large red button).
[0,1,0,0,0,1] *Standalone*		Standalone component without the case (bit 6).

Table 4.2: Examples of images in dataset used by the GAN model under different multi-label conditioning vectors.

4.2.1. Quantitative results

This section presents the quantitative evaluation of the Conditional GAN using the metrics introduced in Section 4.1. The overall results are summarized in Table 4.3.

Metric	Value	Interpretation
FID Score	202.06	Baseline quality
SSIM (mean)	0.5967 ± 0.0569	Moderate-to-high structural similarity
SSIM (min–max)	[0.434, 0.783]	Dependency on configuration complexity
Label Accuracy	91.4%	Effective conditioning
Resolution	128×128	Moderate
Training samples	5,000	Medium-size dataset
Training time	∼2h	Fast

Table 4.3: Quantitative results for the GAN model (DCGAN).

FID. The obtained FID is 202.06. In absolute terms, this is a high value compared to state-of-the-art models (where FID values typically < 50 indicate excellent quality), but it still provides a useful quantitative reference to assess the progress achieved with more advanced architectures. First, the model is trained from scratch and does not benefit from priors learned via pre-training on large-scale datasets: whereas modern diffusion-based models can leverage semantic representations learned from billions of images, the DCGAN must simultaneously learn both the domain structure and the conditioning→image mapping solely from the 5,000 available examples. Second, the dataset size itself is a significant constraint: 5,000 images are sufficient to learn the overall structure of the device (rectangular case, circular slots, main colors), but limited with respect to the volumes typically required to push more sophisticated GANs toward high realism and rich intra-class variability. As a point of comparison, state-of-the-art GANs such as StyleGAN are trained on hundreds of thousands or millions of images. Finally, DCGAN is deliberately a "classical" architecture that does not incorporate improvements introduced in more recent years: for instance, it lacks self-attention mechanisms to capture long-range dependencies, progressive growing techniques to stabilize high-resolution training, and advanced normalizations such as spectral normalization to improve convergence. In this context, a FID of 202.06 indicates that the model learns important regularities of the target distribution,

as reflected by its ability to generate devices with coherent structure, yet a substantial gap remains in terms of overall distributional quality, diversity, and perceptual realism.

SSIM. Alongside FID, structural similarity measured via SSIM provides a more local and complementary indicator of fidelity. The mean SSIM is 0.597 with a standard deviation of 0.057, computed via matched category pairing on 750 image pairs (real and generated images belonging to the same category, where each generated image is paired with a randomly selected real image depicting the identical EPART configuration). This result suggests a moderate-to-high structural similarity between generated images and real images from the same category: in other words, the model can reproduce the main edges, shapes, and spatial organization of the device in a fairly coherent manner, capturing the essence of the structure even if not the fine details. The contained standard deviation (0.057) is particularly meaningful because it indicates relatively stable and predictable quality: generations do not oscillate dramatically between high- and low-quality samples, but tend to preserve similar structural characteristics within each category.

However, the observed range $[0.434, 0.783]$ highlights a dependency on the complexity of the requested configuration. Simpler categories, such as the empty case or single-button configurations with one empty slot, present SSIM values of approximately 0.68 and are easier to reproduce with high structural fidelity: the geometry is simple, the elements are few and well separated, and the model can concentrate its representational capacity on a limited number of components. Conversely, configurations involving multiple elements (e.g., two buttons simultaneously present) tend to be more variable and achieve slightly lower SSIM values (mean ≈ 0.583–0.586, down to 0.434), likely because the number of spatial interactions to be modeled correctly increases, along with the chance of positioning errors or overlaps. Importantly, in a generative setting SSIM should not be interpreted as a measure of "copying" the real images: values around 0.6 indicate that the model preserves the overall structure well, capturing the arrangement of elements, relative proportions, and main contours, while introducing unavoidable local variations (due to stochastic sampling in latent space) and inevitably losing fine-grained details (material textures, micro-reflections, surface imperfections). This trade-off between structural fidelity and local variability is intrinsic to generative models and is, in fact, desirable for data augmentation applications, where the goal is to produce plausible variants rather than exact replicas.

Label Accuracy. Since the Conditional GAN uses discrete binary conditioning rather than textual prompts, the CLIP Score, which measures semantic alignment between a generated image and a natural language description, is not applicable in this context. As

an alternative measure of conditioning effectiveness, Label Accuracy is adopted: a pre-trained classifier is used as a proxy for human judgment to verify whether the requested components are correctly present or absent in the generated images.

Accuracy is computed using the Hamming metric. Given a 6-dimensional binary conditioning vector $\mathbf{v} = [v_1, v_2, v_3, v_4, v_5, v_6]$ representing the requested configuration and a predicted vector $\hat{\mathbf{v}} = [\hat{v}_1, \hat{v}_2, \hat{v}_3, \hat{v}_4, \hat{v}_5, \hat{v}_6]$ produced by the classifier from the generated image, the Hamming Accuracy is defined as:

$$\text{Hamming Accuracy} = \frac{1}{B} \sum_{i=1}^{B} v_i \cdot \mathbb{1}[v_i = \hat{v}_i] \tag{4.4}$$

where $B = 6$ is the number of bits and $\mathbb{1}[\cdot]$ is the indicator function. Unlike exact match accuracy, which considers a prediction correct only if all six bits are simultaneously correct, the Hamming metric evaluates each attribute independently: an image that correctly reproduces five out of six requested components contributes positively to the score rather than being counted as entirely wrong. This property makes the Hamming metric particularly appropriate for multi-attribute conditioning, where partial correctness carries meaningful information about the model's behavior.

For instance, consider a prompt requesting a case with a red button and an empty slot, encoded as $\mathbf{v} = [1, 1, 0, 0, 0, 0]$. If the classifier predicts $\hat{\mathbf{v}} = [1, 0, 0, 0, 0, 0]$ (case correctly generated but red button missing), the Hamming Accuracy for that sample is $5/6 \approx 83.3\%$, correctly reflecting that five out of six attributes were reproduced accurately. Under exact match accuracy, the same prediction would yield 0%, failing to capture the partial correctness of the generation.

The mean Label Accuracy over the evaluation set is 91.4% (Hamming average over 1,500 generated images, 100 per each of the 15 categories). This result indicates that the model reliably adheres to the input attribute vector: on average, fewer than one bit out of ten is inconsistent with the input conditioning vector, demonstrating robust control over the generative process.

This aspect is particularly relevant from an applied perspective: the reliable controllability over the generated content is often more critical than absolute visual realism, as the ability to specify exactly which components must appear in a synthetic image directly determines the practical utility of the model.

The per-component analysis (Table 4.4) highlights both the attributes with high accuracy and the more challenging ones. The most visually distinctive attributes achieve excellent

performance: the red button reaches 94.2%, benefiting from a vivid color and a well-defined circular shape; the empty case reaches 93.1%, as the absence of components in the slots constitutes an easily recognizable structural state; and the green button reaches 92.8%, clearly distinguishable from the other colors. These results confirm that the model has learned to reliably reproduce the basic semantics of the conditioning vector for chromatically and geometrically salient attributes.

Difficulties are instead concentrated in visually more ambiguous cases. The yellow button yields the lowest accuracy (85.3%), which can be attributed to two concurrent factors: first, the yellow hue is chromatically close to the wooden background present in all dataset images, reducing the chromatic contrast between the component and its surroundings; second, under slight saturation variations induced by the generator, yellow tones may shift toward the red-orange region of the spectrum, making the button visually similar to the emergency button. The emergency button itself exhibits below-average accuracy (89.7%), suggesting that the generator does not consistently emphasize its defining geometric property, a noticeably larger size compared to standard buttons, producing samples in which the size difference is insufficiently pronounced. Since the emergency button shares the same red-orange hue as the standard red button, its primary discriminative feature is size rather than color; when this size difference is not clearly reproduced, the generated sample becomes ambiguous. Conversely, the standard red button achieves high accuracy (94.2%) precisely because, unlike the emergency button, it does not rely on a relative size cue: its vivid, saturated red tone alone is sufficient for unambiguous identification, and this chromatic property is consistently reproduced by the generator.

This confusion is plausible and understandable given the specific domain: these are industrial components characterized by "warm" colors (red, yellow, orange) that can indeed appear similar under certain conditions, such as artificial lighting, camera angle, or material wear. Moreover, with a dataset of 5,000 images distributed across 15 categories, some less frequent configurations (e.g., specific yellow–emergency combinations) may have limited training examples, making it harder for the model to learn fine distinctions. This observation does not change the overall positive reading: conditioning works well in general, and a 91.4% accuracy is more than sufficient for many practical applications. However, it also suggests that improving the separation between visually adjacent classes would likely require targeted interventions: additional training data covering a wider range of chromatic variations for the yellow button, targeted augmentation explicitly emphasizing size differences for the emergency button (e.g., by including more examples with clearly oversized emergency buttons relative to standard ones), or a more expressive architecture with higher representational capacity capable of consistently encoding subtle geometric

and chromatic distinctions.

Table 4.4: Per-component Label Accuracy (GAN).

Component	Accuracy
Red button	94.2%
Green button	92.8%
Empty case	93.1%
Yellow button	85.3%
Emergency button	89.7%
Standalone component	91.5%
Mean (Hamming)	**91.4%**

4.2.2. Qualitative analysis

The qualitative analysis (Figure 4.1) confirms the trends highlighted by the quantitative metrics and provides direct visual validation of the model's generative capability. Across all 15 configurations, the generated images exhibit a generally coherent structure: the case is reproduced with a dark rectangular shape and two well-positioned circular slots (visible in the first three rows of the grid), while the buttons appear centered within the slots and their colors arevisually consistent with the conditioning. The model correctly differentiates the yellow button from the emergency button: the latter appears noticeably larger and with a intense red hue, indicating that the model has learned the distinctive dimensional and chromatic semantics associated with the emergency button. The model also reproduces the distinction between components mounted on the case and standalone components: in the latter, buttons are generated in isolation without the rectangular case structure, suggesting that the sixth bit of the conditioning vector (absence of the case) has been learned and respected.



Figure 4.1: Samples generated by the GAN model across the 15 configurations. Resolution: 128×128 pixels.

At the same time, typical limitations of a DCGAN approach under a limited-data regime are apparent. The 128×128 resolution inevitably reduces the presence of fine details (plastic material texture, micro-reflections, screws) and often makes surfaces appear overly uniform or "flat". Several samples show background artifacts and noise, for example in the first two rows of fig 4.1, the case contour fades irregularly, which reduce overall realism and would make the images easily distinguishable from real ones. In addition, intra-class variability appears limited: comparing samples with similar configurations shows that, although not identical, images generated under similar conditioning tend to be very similar in terms of button positioning, relative size, and viewpoint. This suggests that the model can reproduce the main structural template, but struggles to generate richer variation in illumination, pose, or micro-detail that would naturally arise in a dataset collected under diverse real-world acquisition conditions. Table 4.5 summarizes the main strengths and limitations observed.

Strengths	Limitations
Consistent case structure (rectangular shape, well-positioned circular slots)	128×128 resolution limits fine details (plastic texture, micro-reflections, screws)
Distinctive and recognizable colors (red, green, and yellow are clearly differentiated)	Confusion between yellow and emergency button (chromatic shift toward red-orange; accuracy 85.3% vs. 89.7%)
Correct placement of buttons within slots (centered, proportional)	Background artifacts and noise (case contour fades irregularly)
Correct case/standalone distinction (bit 6 learned reliably)	Uniform, "flat" textures and limited material realism
High Label Accuracy (91.4%) indicates effective conditioning	

Table 4.5: GAN qualitative summary: observed strengths and limitations.

Overall, the results confirm that a DCGAN with multi-lable conditioning can learn an effective label→image mapping in the considered industrial domain, achieving reliable controllability (Label Accuracy 91.4%) and moderate-to-high structural similarity (SSIM $\approx$ 0.60). The model demonstrates that it has correctly learned the semantics of the conditioning vector: all 15 configurations are generated with appreciable fidelity, clearly distinguishing between the empty case, single-button configurations with one empty slot, multi-button combinations, and standalone components. The low SSIM standard deviation (0.057) further indicates notable generative stability, with consistent outputs within each category and no evidence of significant *mode collapse*.

However, the analysis also highlights important limitations. Global distributional quality, as measured by FID (202.06), indicates a substantial gap with respect to real images. While this value is consistent with the experimental setup, training from scratch on 5,000 images at 128×128 resolution with a classical architecture, it clearly suggests that the model struggles to capture the complexity of textures, illumination variations, and fine-grained details present in the real data. The qualitative observations renforce this pattern: although the global device structure (case shape, slot positioning, main colors) is reproduced well, surfaces often appear overly smooth or flat, with limited realism of the plastic material and occasional background artifacts. Furthermore, intra-class variability remains limited: images generated under the same conditioning tend to be highly similar, suggest-

ing that the model has learned a prototypical template for each configuration but does not produce rich variations in viewpoint, lighting, or micro-detail.

A particularly interesting aspect concerns visually ambiguous components. The yellow button exhibits the lowest accuracy (85.3%), attributable to its chromatic proximity to both the wooden background and the emergency button, whose red-orange hue overlaps with saturated yellow tones. The emergency button itself achieves 89.7% accuracy, with errors primarily driven by the generator's difficulty in consistently emphasizing its defining size difference with respect to the standard red button, which instead achieves high accuracy (94.2%) precisely because its vivid saturated hue alone is sufficient for unambiguous identification. This behavior does not indicate a failure of conditioning, but rather an intrinsic limitation in capturing subtle distinctions when the training dataset provides relatively few examples for closely related categories. With additional data or a more expressive architecture, these ambiguities could likely be reduced.

Taken together, the conditional DCGAN constitutes an appropriate and informative baseline for this thesis. It is conceptually simple, computationally efficient (approximately 2 hours of training), and clearly demonstrates the feasibility of controlled generation of images within the target industrial context. Its key strength is reliable controllability, as reflected by the high label accuracy and SSIM, which makes it suitable for data augmentation scenarios where the primary requirement is to produce controlled variants of specific configurations. At the same time, limitations in distributional quality (high FID), perceptual realism (textures and fine details), and intra-class variability leave clear room for substantial improvements with more advanced generative approaches.

4.3. Diffusion Model trained from scratch results

This section reports the evaluation results of the diffusion model trained from scratch on the EPART device image dataset. Unlike the GAN baseline, which relies on a binary conditioning vector, the diffusion model is conditioned through descriptive text prompts encoded via a CLIP text encoder, enabling a more flexible and semantically richer conditioning mechanism.

To assess a more modern approach compared to the GAN baseline, a text-conditioned diffusion model has been trained from scratch on the EPART device image dataset. The purpose of this second baseline is to verify to what extent a diffusion-based generative model can improve image quality and conditioning fidelity while preserving a purely "from scratch" setting (i.e., without fine-tuning of pre-trained models).

In contrast to the GAN, conditioning is expressed through textual captions rather than binary vectors, describing the color and number of buttons for each configuration (Section 3.3). Images were used at a resolution of 64×64 pixels to keep training time and memory usage compatible with the available resources. The dataset was split with a strongly training-oriented partition (∼96% training, 4% validation), with checkpoint selection based on training loss. Training was performed for 150 epochs (batch size 32, Adam, lr $= 1 \times 10^{-4}$, gradient clipping), with a diffusion process configured with 1,000 timesteps (linear schedule) and DDIM sampling (50 steps) at inference time.

Overall, this setup defines a from-scratch diffusion baseline that is directly comparable to the GAN baseline: both models are trained on the same domain without pre-training, but they differ in conditioning modality (text vs. discrete labels), resolution, and computational cost.

4.3.1. Quantitative results

The quantitative evaluation of the diffusion model is carried out using the same metrics adopted for the GAN baseline, FID to measure distributional quality, SSIM to assess structural fidelity, in order to ensure a direct and consistent comparison, with the addition of CLIPScore to quantify the semantic coherence between the generated image and the textual prompt, a measure not applicable to the GAN, which uses discrete binary conditioning rather than natural language descriptions.

Overall results are reported in Table 4.6. The diffusion model exhibits a substantial improvement over the GAN baseline in terms of distributional image quality, as highlighted by the reduction in FID. At the same time, interesting differences emerge between the structural and semantic metrics, suggesting a qualitatively different behavior compared to the GAN.

Metric	Value	Interpretation
FID Score	90.63	Good result
SSIM	0.562 ± 0.097	Moderate similarity, high variability
SSIM (min–max)	[0.291, 0.883]	Wide range
CLIP Score	0.293 ± 0.021	Good text–image alignment
Resolution	64×64	Low (technical limitation)
Training samples	5,000	Medium-sized dataset
Training time	∼8h	Higher complexity vs. GAN

Table 4.6: Quantitative results of the diffusion model trained from scratch.

FID. The diffusion model achieves an FID of 90.63, markedly lower than the value obtained by the GAN (202.06). This corresponds to a 55% improvement and indicates that, on average, images generated by the diffusion model are significantly closer to the distribution of real images than those produced by the DCGAN. The FID value falls within the "good" range according to common conventions in the literature (50–100), approaching the threshold of high quality while still remaining far from state-of-the-art models (FID < 50).

This improvement in FID can be interpreted in light of several architectural and methodological factors. First, the iterative denoising procedure allows the model to learn a richer and more continuous representation of the image space. Unlike GANs, which map latent noise to the final image in a single forward pass, the diffusion model progressively refines the sample over 50 denoising steps, thereby reducing the coarse artifacts that are typical of GANs trained from scratch. This multi-step process enables local error correction and smoother transitions across image regions.

Second, text conditioning via CLIP provides a semantically richer conditioning input than the binary vector used by the GAN. While the 6-bit vector encodes discrete, atomic information (presence/absence of components), the 512-dimensional CLIP embeddings capture continuous semantic relationships among visual concepts, potentially enabling better generalization and more natural compositions.

Third, the MSE-based noise prediction objective leads to a more stable training compared to adversarial optimization: GAN training is known to be sensitive to instabilities, mode

collapse, and oscillations, whereas diffusion training benefits from more robust optimization and more informative gradients. This translates directly into a better coverage of the real data distribution, which is reflected in the lower FID score observed for the diffusion model.

Nevertheless, the absolute FID value (90.63) remains far from what is commonly reported for state-of-the-art diffusion models trained at scale (e.g., FID < 30 for models such as Stable Diffusion). This is consistent with the experimental setup: the model is trained from scratch on a relatively small dataset (5,000 images, as opposed to millions for SOTA models) and at low resolution (64×64 rather than 512×512 or higher), without benefiting from large-scale pretraining. The obtained FID represents a substantial improvement over the GAN baseline, validating the effectiveness of diffusion in the considered domain, but it does not indicate a level of realism comparable to large pretrained generative models.

SSIM. The diffusion model attains a mean SSIM of 0.562 ± 0.097, computed via matched category pairing: for each generated image, a real reference image depicting the identical EPART configuration is selected, and SSIM is computed between the two, ensuring structural comparison between semantically equivalent pairs. Compared to the GAN (SSIM $= 0.597 \pm 0.057$), the diffusion model shows a different behavior in two main aspects: the mean value is slightly lower (-6%), but, more importantly, the variability is substantially higher (std $= 0.097$ vs. 0.057, +70%). This difference is both statistically and qualitatively meaningful.

The observed range [0.291, 0.883] is much wider than that of the GAN [0.434, 0.783], suggesting that the diffusion model produces outputs with greater structural diversity. This behavior is coherent with the model paradigm: GANs, through adversarial pressure, often converge to "safe" solutions that reliably fool the discriminator (sometimes resulting in more homogeneous samples), whereas diffusion models explore the image manifold in a more stochastic manner, producing higher local variety while preserving global semantic coherence.

Table 4.7 reports the configurations with the highest and lowest SSIM values, highlighting the dependence on the compositional complexity of the prompt.

Configuration	SSIM	Interpretation
Top 3 (highest SSIM)		
Empty case	0.697	Simple structure, well-defined geometry, high consistency
Red button (standalone)	0.683	Single element, circular symmetry, easy to reproduce
Case with green button	0.658	Basic configuration, one active element
Bottom 3 (lowest SSIM)		
Case red + yellow	0.412	Two buttons with similar warm tones
Case green + emergency	0.398	Mixed sizes (standard + large button)
Case yellow + emergency	0.347	Most challenging: mixed sizes and similar warm tones
Overall mean	**0.562**	Balance across simple and complex configurations

Table 4.7: Diffusion model SSIM by category: best and worst configurations.

A clear dependence on prompt complexity can be observed: simple configurations such as "empty case" (SSIM = 0.697) or a single standalone component (SSIM = 0.683) achieve higher SSIM values, as the target structure is geometrically simple and the model converges more easily to consistent layouts. The empty case, in particular, represents the simplest configuration, a rectangular case with two empty circular slots, and correspondingly yields the highest SSIM, indicating that this base structure is robustly learned.

Conversely, prompts requiring the simultaneous presence of multiple elements yield significantly lower SSIM. The worst configuration ("case yellow + emergency", SSIM = 0.347) combines two concurrent challenges: the geometric complexity of simultaneously reproducing two buttons of significantly different sizes (standard yellow vs. larger emergency button), and the chromatic proximity of the yellow hue to both the wooden background and the red-orange tone of the emergency button, which can induce saturation variations in the generated output. Similarly, "case green + emergency" (SSIM = 0.398) appears challenging due to size management: the model struggles to consistently emphasize the relative size difference between the standard button and the emergency button, producing samples with variable proportions.

Importantly, configurations with lower SSIM are not necessarily those with the poorest qualitative outcomes. By inspecting generated samples, often the model produces visually plausible and semantically correct images (high CLIPScore), yet with local structural variations, e.g., slightly different button positions, variable proportions, or different viewpoints, which lower SSIM while preserving the intended content. This reflects the greater exploratory freedom of diffusion models compared to GANs.

It is crucial to emphasize that lower SSIM does not necessarily imply lower quality. The diffusion model is not designed to replicate pixel-wise near-identical counterparts of specific dataset images, but rather to generate plausible samples from the same distribution. Therefore, moderate SSIM values accompanied by higher diversity are expected, as long as samples remain semantically correct and visually credible. The fact that the diffusion model achieves a substantially better FID than the GAN (90.63 vs. 202.06) despite a slightly lower SSIM supports the interpretation that overall perceptual quality is improved, even though pixel-level alignment with the paired references is less consistent.

CLIPScore. The mean CLIPScore obtained by the diffusion model is 0.293 ± 0.021, exceeding the threshold for good alignment (0.28) commonly reported in the literature [20]. This result indicates a solid coherence between generated images and the provided textual prompts, validating the effectiveness of cross-attention conditioning during denoising.

A direct comparison with the GAN is methodologically non-trivial, since the GAN uses discrete binary conditioning and is evaluated via Label Accuracy (91.4%), whereas CLIPScore measures continuous semantic alignment. Nonetheless, both results indicate that the respective conditioning mechanisms operate effectively: the GAN follows the attribute vector with high reliability, while the diffusion model captures the semantic content of the natural-language prompts.

The CLIPScore value of 0.293 suggests that the model:

1. identifies the concepts mentioned in the prompt (e.g., “red button”, “empty slot”);
2. places elements in a way that remains consistent with the textual description;
3. maintains semantic consistency even for compositional prompts.

The CLIPScore analysis across prompt of increasing complexity (Table 4.8) highlights additional patterns:

Table 4.8: Diffusion model CLIP Score by prompt complexity.

Prompt type	CLIP mean	Std	Examples
Simple (1 element)	0.302	0.014	"empty case", "red button"
Moderate (2 elements)	0.287	0.019	"case with red button"
Complex (3+ elements)	0.281	0.023	"case with red button and green button"
Abstract concepts	0.245	0.031	"empty case" (absence)

Simpler prompts yield higher CLIP Scores, whereas complex prompts or those involving more abstract concepts (such as "empty", which denotes an absence) are more challenging. This pattern is consistent with known limitations of CLIP in handling negation and complex spatial relations [25].

A particularly interesting aspect is the SSIM–CLIP trade-off: moderate SSIM values (0.50–0.55) combined with higher CLIP Scores (0.30–0.35) reflect the diffusion model's tendency to generate semantically consistent but structurally diverse samples, in contrast to the GAN's more rigid adherence to training-set prototypes. In data augmentation applications, this trade-off is generally favorable: semantically correct samples with higher structural diversity are typically more useful than near-duplicates of a small number of prototypes.

In summary, the diffusion model trained from scratch demonstrates substantially stronger generative performance than the GAN baseline, achieving an FID of 90.63 (55% improvement) while maintaining good semantic alignment with textual prompts (CLIP Score 0.293). The higher structural variability (SSIM std = 0.097) reflects the exploratory nature of denoising, which tends to favor diversity and naturalness over strict pixel-wise correspondence.

The main limitations remain the low resolution (64×64, a technical/memory constraint) and the higher computational cost ($\sim$8 hours on P100 GPU of training and 50-step inference). Nevertheless, the results confirm the effectiveness of diffusion even in from-scratch settings with limited data.

4.3.2. Qualitative results

The qualitative analysis (Figure 4.2) provides a visual validation of the trends observed through the quantitative metrics. Across the 15 generated configurations, the samples are generally semantically consistent with the textual prompts and look realistic, while still showing limitations that can be directly attributed to the low resolution (64×64) imposed by computational constraints.



Figure 4.2: 5×3 grid of images generated by the diffusion model across all 15 categories (64×64 resolution). Text prompts conditioning through cross-attention mechanisms, yields samples that are semantically coherent with the input descriptions.

A first, clearly positive outcome emerges from inspection of the generated images in figure 4.2: the model correctly interprets the prompts correctly and translates them into appropriate visual configurations. Generated images consistently follow the provided descriptions, placing the requested elements in the expected slots and preserving a plausible spatial arrangement. The simplest configuration, *empty case*, shows a black case with two clearly empty circular slots, confirming that the model has learned the device's base structure. For single-button configurations, the model places the requested component in

the first slot while keeping the second slot empty, indicating that it has learned a consistent layout. When prompted with two-button combinations, such as *case with red and green button* or *case with green and yellow button*, the model distributes the two elements across the available slots in a way that is coherent both with the content and with the compositional structure implied by the text.

A particularly relevant aspect concerns relative scale, especially for the emergency button. In *case with emergency button*, the emergency component occupies most of the slot and appears noticeably larger than the standard buttons observed in other configurations. This scale distinction is also preserved in more complex prompts: in *case with red and emergency button*, the emergency button visually dominates the composition, producing a clear contrast in size with the adjacent element. The ability to capture and respect size-related attributes expressed in the prompt is a meaningful strength, also consistent with the CLIP Score (0.293), which indicates a good semantic alignment between text and image.

In terms of perceptual realism, the samples exhibit relatively smooth color transitions and limited coarse artifacts. For instance, the standalone green button shows gradual intensity variations that suggest volume and three-dimensionality, despite the strong constraints imposed by the low resolution. Similarly, the shading visible on the emergency button contributes to a sense of depth, avoiding the fully flat appearance typical of simple colored geometric primitives. Color differentiation is also generally preserved across the grid: red, green, yellow, and the emergency tone (red–orange) remain distinguishable, with plausible saturation and consistent hues within each color category.

Overall, the stochastic denoising process appears to explore multiple plausible spatial configurations rather than converging to a single rigid prototype for each category. While this diversity can reduce SSIM in pixel-level comparisons, it is typically desirable for data augmentation, where the goal is to enrich a dataset with non-identical yet plausible variants.

At the same time, the qualitative analysis reveals important limitations. The 64×64 resolution is a pervasive constraint that prevents the representation of fine-grained details. Visual cues that would characterize real industrial objects, such as mounting screws at the case corners, plastic material texture, micro-imperfections, and specular highlights, are absent or collapsed into indistinguishable gradients. The circular slot boundaries, which in real photographs would be sharp and well defined, appear instead blurred with gradual transitions. For example, in *case with red and green button*, the separation between button and slot is not clearly delineated: there is an intermediate region where the button color

blends into the black case, producing a fading effect that reduces the perceived structural definition.

More compositional configurations introduce additional difficulties. In *case with yellow and emergency button*, the scale contrast between the two components is less pronounced than in simpler prompts: the yellow button appears closer in size to the emergency button, even though the latter should clearly dominate. Additionally, the yellow hue tends to blend with the wooden background, further reducing the overall structural clarity of the generated sample. This behavior is consistent with the lowest SSIM recorded for this configuration (0.347, minimum in Table 4.7), reflecting the concurrent difficulty of satisfying size constraints and managing chromatic proximity to both the background and the emergency button's red-orange hue.

A final observation concerns scale consistency within similar categories. When comparing prompts that include the emergency button, the relative size of the emergency component with respect to standard buttons varies noticeably across samples: in *case with green and emergency*, the emergency button is clearly visible and well separated from the green button, whereas in *case with red and emergency* the two elements tend to visually merge due to their similar hue, even though the emergency button generally remains larger. This inconsistency, while partially reflecting the structural diversity characteristic of the model (high SSIM std), may be problematic when the target application requires precise and reliable scale cues for downstream computer vision tasks.

Strengths	**Limitations**
Accurate semantic interpretation of text prompts	64×64 resolution removes fine-grained details
Appropriate handling of relative scale (larger emergency button)	Blurred edges and gradual transitions instead of sharp contours
Smooth color transitions, overall plausible perceptual appearance	Scale ambiguities in more complex configurations (e.g., red+emergency)
Clear chromatic differentiation across color categories	
Coherent spatial layout of elements within the slots	

Table 4.9: Qualitative summary: strengths and limitations.

In summary, the diffusion model trained from scratch demonstrates strong generative capabilities, achieving an FID of 90.63 while maintaining good semantic alignment with the textual prompts (CLIP Score 0.293). The qualitative inspection confirms that the model has learned meaningful mappings between text descriptions and visual configurations, producing semantically coherent samples with smooth color transitions and a degree of variability that is generally beneficial for data augmentation purpose. In particular, the ability to respect relative scale and spatial layout, enabled by cross-attention conditioning, represents a notable advantage over approaches based on discrete conditioning.

The main limitations remain the low resolution (64×64, driven by memory constraints during training) and the higher computational cost ($\sim$8 hours of training and 50-step inference). Resolution is the critical bottleneck: despite producing semantically plausible compositions, the lack of fine detail severely constrains practical usability in industrial settings, where such cues are often discriminative for downstream computer vision tasks such as object detection.

Nevertheless, the results confirm the effectiveness of diffusion even under from-scratch training with limited data (5,000 images), supporting the advantages of iterative denoising and text conditioning via cross-attention. These findings motivate the next step of this work: assessing whether fine-tuning large-scale pretrained diffusion models via LoRA can overcome the resolution and realism limitations by leveraging priors learned from billions of images, as discussed in the following section.

4.4. LoRA Fine-Tuning Results

This section reports the results of the transfer-learning approach, consisting of a LoRA (Low-Rank Adaptation) fine-tuning of Stable Diffusion XL (SDXL) Base 1.0 on the EPART device image dataset. Unlike the previous experiments, this setup does not start from randomly initialized weights. Instead, it leverages a foundation model pretrained on billions of image–text pairs and specializes it to the industrial domain by updating only a limited subset of parameters. The goal is to assess the impact of large-scale pretraining and parameter-efficient adaptation on image quality, semantic consistency, and visual detail, and to verify whether superior results can be achieved even with a dataset that is substantially smaller than those used in from-scratch training.

LoRA fine-tuning was performed on `stabilityai/stable-diffusion-xl-base-1.0` using a DreamBooth-style pipeline with Low-Rank Adaptation (rank 16, $\sim$40M trainable parameters, 1.5% of total model parameters) (Section 3.4). The dataset consists of 155 EPART images paired with descriptive captions and a fixed instance prompt *"a photo*

of EPART product case". Training was conducted at 512×512 resolution, substantially higher than GAN (128×128) and from-scratch Diffusion (64×64), to enable reproduction of fine-grained industrial details (sharp edges, material textures, mounting screws). The model was trained for 800 steps (batch size effective 4 via gradient accumulation, learning rate 1×10^{-4} for U-Net LoRA adapters) with the CLIP text encoder kept fully frozen to preserve semantic knowledge and enable chromatic generalization to unseen colors.

4.4.1. Quantitative Results

The quantitative evaluation of the LoRA model was carried out using the same metrics adopted for the GAN and the diffusion model, in order to ensure a directly comparable assessment across approaches. Specifically, we use the three metrics defined in section 4.1: the Fréchet Inception Distance (FID) to measure the distributional quality of generated images; the Structural Similarity Index (SSIM) to quantify structural fidelity with respect to real images via matched category pairing; and the CLIP Score to verify semantic alignment between textual prompts and the generated visual content. The overall results are summarized in Table 4.10.

Metric	**Value**	**Interpretation**
FID Score	61.35	Very good result
SSIM	0.507 ± 0.055	Moderate similarity, low variance
SSIM (min–max)	[0.372, 0.662]	Moderate range
CLIP Score	0.309 ± 0.025	Excellent text–image alignment
Resolution	512×512	High resolution
Training samples	155	Minimal dataset
Training time	∼1h	Efficient

Table 4.10: Quantitative results of the LoRA-fine-tuned SDXL.

At an aggregate level, the results show substantial improvements over the models trained from scratch. The FID of 61.35 is the lowest among all evaluated approaches, indicating that LoRA-generated images are the closest to the real-image distribution in the Inception-v3 feature space. The CLIP Score of 0.309 is also the highest, comfortably above the commonly used threshold for good alignment (0.28) and approaching the range typically

considered excellent (0.35), suggesting that the model follows the semantic content of the prompts with high reliability. SSIM is slightly lower than for the other models; however, it should be interpreted as consequence of the much higher output resolution (512×512).

FID. The LoRA model achieves an FID of 61.35, corresponding to a 70% improvement over the GAN baseline (from 202.06 to 61.35) and a 32% improvement over the diffusion model trained from scratch (from 90.63 to 61.35). In absolute terms, this result lies in the upper portion of the "good" range (50–100) and approaches the threshold commonly associated with excellent quality (FID < 50), despite the markedly smaller dataset and the constrained computational budget.

This improvement relative to from-scratch training can be attributed to multiple factors that operate jointly. First, large-scale pretraining provides SDXL with strong visual priors. Having been trained on billions of image–text pairs, the model already encodes general photographic and visual regularities (e.g., shading, reflections, plausible textures, and three-dimensional appearance). As a consequence, fine-tuning can focus primarily on domain adaptation, learning EPART-specific geometry, slot layout, case shape, and relative component dimensions, rather than rediscovering low-level visual phenomena from a limited dataset.

Second, the training resolution (512×512) contributes directly to the distributional match. While the from-scratch diffusion baseline was constrained to 64×64 for memory reasons, the LoRA fine-tuned SDXL model inherits an architecture designed for high-resolution synthesis. This enables the model to represent fine-grained cues that are relevant both perceptually and statistically (e.g., sharp edges, material texture, subtle reflections, and shading). Since FID is computed in Inception-v3 feature space, which is itself optimized for higher-resolution natural images, higher-resolution generations provide richer and more informative features than heavily downsampled images, supporting a closer match to the target distribution.

Third, LoRA as a parameter-efficient adaptation mechanism reduces the risk of overfitting and catastrophic forgetting. By updating only a small fraction of the total parameters (approximately 1.5%), the fine-tuning procedure preserves most of SDXL's pretrained capability for realistic image synthesis. The LoRA adapters then steer the model toward EPART device-specific structure without destabilizing the global representation learned during pretraining. This conservative update strategy makes it feasible to obtain meaningful specialization from only 155 training images, a dataset size that would be clearly insufficient for training a model of this capacity from scratch.

Finally, the DreamBooth-style conditioning setup, based on a fixed instance prompt ("a photo of EPART product case") used as a concept anchor, supports learning a coherent representation of the target domain. In practice, this trigger phrase allows the text encoder to consistently refer to the EPART concept, while additional prompt tokens (e.g., "red button", "empty slot") can be composed to specify configurations. This favors compositional generation, enabling the model to combine attributes and structures in a controlled way.

Despite the strong improvement, the result should be contextualized. An FID of 61.35, while excellent relative to the from-scratch baselines, is still higher then the values typically reported by frontier generative models trained on much larger datasets and computational resources (often FID < 30 on standard benchmarks). This gap is plausibly due to both the limited dataset size (155 images versus hundreds of thousands or millions) and the specificity of the industrial domain: EPART device images are characterized by precise geometries, uniform materials, and saturated colors that differ from the natural-image distribution on which Inception-v3 was trained, potentially making the extracted features less representative. Nevertheless, the obtained FID strongly validates transfer learning as an effective strategy to achieve high-quality synthesis under severe data constraints.

SSIM. The mean SSIM for the LoRA model is 0.507 ± 0.055, computed via matched category pairing: for each of the 155 generated images, a real reference image depicting the identical EPART configuration is selected, and SSIM is computed between the two, ensuring structural comparison between semantically equivalent pairs. At first glance, this appears lower than the GAN (0.597) and from-scratch diffusion (0.562), potentially suggesting reduced structural fidelity. However, a direct comparison is misleading unless differences in resolution and in the nature of SSIM are taken into account.

SSIM is inherently sensitive to image resolution: it evaluates local similarity over sliding windows, and the number of local comparisons scales with the total number of pixels. A 512×512 image contains 262,144 pixels, compared to 16,384 at 128×128 and 4,096 at 64×64. As a result, higher-resolution images expose many more opportunities for minor local discrepancies (illumination changes, micro-textures, reflections, subtle shading differences) that may be perceptually negligible.

Importantly, the SSIM standard deviation for LoRA (0.055) is comparable to the GAN (0.057) and substantially lower than the from-scratch diffusion model (0.097), indicating relatively stable structural behavior across categories.

A category-level analysis (Table 4.11) further clarifies the distribution of SSIM values.

Configuration	SSIM	Interpretation
Top 3 (highest SSIM)		
Case with emergency button	0.549	Single large element, simple structure
Emergency button (standalone)	0.537	Isolated large component, well-defined geometry
Case with green button	0.534	Single button configuration, distinctive color
Bottom 3 (lowest SSIM)		
Green button (standalone)	0.427	Small isolated object; high resolution amplifies micro-variations
Yellow button (standalone)	0.435	Small object; variability in placement and fine detail
Red button (standalone)	0.463	Similar to other standalone components; positional variability
Overall mean	**0.507**	Balance across configurations

Table 4.11: SSIM of the LoRA model by category: best and worst configurations.

The highest-SSIM configurations involve single elements with relatively large size (emergency button, both standalone and within the case) or cases with a single distinctive button. These scenarios reduce compositional degrees of freedom and are easier to reproduce consistently. Conversely, the lowest SSIM values correspond to **small standalone components** (green, yellow, and red buttons), where high resolution magnifies small variations in position, orientation, shading, and micro-texture that would be largely invisible at lower resolutions. These lower SSIM values should not be interpreted as poor visual quality; rather, they reflect a different trade-off in which LoRA prioritizes detail and realism over pixel-level similarity to a specific reference.

Notably, two-button configurations involving the emergency button (case with green + emergency: 0.470, case with yellow + emergency: 0.530) show moderate SSIM values, suggesting that the model handles size contrast (standard vs. large button) with reasonable structural fidelity, though compositional complexity introduces additional variability.

Overall, the SSIM of 0.507, while numerically lower than GAN (0.597) and from-scratch

diffusion (0.562), should be interpreted in the context of the substantially higher resolution (512×512 vs. 128×128 and 64×64): higher resolution images contain exponentially more pixels and therefore exponentially more opportunities for local discrepancies, naturally resulting in lower SSIM values even when perceptual quality is superior. This resolution-dependent behavior of SSIM has been documented in the literature and does not indicate inferior structural fidelity; rather, it reflects the finer granularity at which structural comparison is being performed. The value of 0.507 is therefore consistent with high-quality 512×512 synthesis and should be evaluated alongside FID and CLIP Score for a comprehensive quality assessment.

CLIPScore. The LoRA model achieves a mean CLIP Score of 0.309 ± 0.025, representing the best result among the text-conditioned approaches (from-scratch diffusion: 0.293). The score is well above the threshold typically associated with good alignment (0.28) and approaches the range often described as excellent (0.35), confirming strong semantic consistency between prompts and generated content.

The improvement over the from-scratch diffusion model (0.309 vs. 0.293, +5.5%) can be explained by two main factors. First, the higher image resolution provides the CLIP vision encoder with richer visual evidence during embedding computation. Since CLIP is trained on substantially higher-resolution images than 64×64, extremely low-resolution generations limit its ability to recognize fine-grained attributes such as precise colors, sharp boundaries, and clear spatial relations. Second, LoRA preserves the pretrained text encoder and most of the pretrained U-Net, whereas the from-scratch diffusion model must learn to map text embeddings to images without visual priors. In the LoRA setup, the adapters mainly specialize the model to EPART-specific structure while retaining the general text-to-image alignment capabilities learned during pretraining.

Table 4.12 reports CLIP Score statistics by prompt type, showing that LoRA maintains high alignment even for compositional descriptions.

Prompt Type	CLIP Mean	Std	Examples
Color combinations	0.318	0.018	“red and yellow button”
Standalone buttons	0.317	0.020	“red button”, “green button”
Emergency + color	0.315	0.026	“emergency and green button”
Single-color with case	0.306	0.015	“case with red button”
Abstract concepts (empty)	0.290	0.008	“empty case”
Overall mean	**0.309**	**0.025**	—

Table 4.12: CLIP Score of the LoRA-fine-tuned SDXL model by prompt complexity.

The highest CLIP Scores are obtained for prompts involving clearly identifiable colors or standalone components, where semantic content is visually salient and unambiguous. Prompts involving the concept of “empty” yield lower scores, as they encode an absence rather than a presence, a known difficulty for CLIP-like models, yet the value remains above the good-alignment threshold, indicating that the LoRA model still captures this configuration reliably.

In summary, the LoRA fine-tuning achieves the best overall semantic alignment (CLIP Score 0.309) and the best distributional quality (FID 61.35) among the evaluated approaches, while SSIM appears lower primarily due to resolution-driven sensitivity and the presence of realistic fine-scale variability. Taken together, these results position the LoRA-fine-tuned SDXL model as the most effective strategy for high-quality EPART text-to-image generation.

4.4.2. Qualitative Results

The qualitative analysis of the LoRA model reveals a significant improvement in visual realism and resolution compared to the models trained from scratch, as reflected in the lower FID score (61.35 vs. 202.06 for GAN and 90.63 for Diffusion) and higher output resolution (512×512). However, this perceptual quality comes with a trade-off: the model occasionally produces compositional artifacts, including incorrect object counts (e.g., generating multiple buttons when the prompt specifies "one red button") and spatial arrangement errors (e.g., buttons positioned outside designated slots).

These errors stem from two concurrent factors. First, CLIP-based text conditioning, while effective for semantic attributes like color and object categories, has well-documented

limitations in encoding precise spatial relationships, object counting, and negations [25]. The text encoder can represent "red button" robustly, but struggles to enforce constraints like "exactly one" or "inside the left slot." Second, transfer learning from SDXL's general visual knowledge does not extend to domain-specific compositional rules such as "exactly two circular slots" or "one button per slot," which must be learned from training data. With only 155 training images, the LoRA adapters have insufficient examples to reliably encode these rigid geometric constraints, resulting in occasional violations of the intended configuration.

Inspecting the generated images (Figure 4.3) makes it possible to directly appreciate the impact of high resolution (512×512), large-scale pretraining, and parameter-efficient fine-tuning. In addition, two particularly relevant aspects emerge from this analysis: the model's ability to generalize to colors that are completely absent from the training set (Figure 4.4), and a direct comparison between generations before and after fine-tuning (Figure 4.5), which concretely illustrates what the model learns during adaptation to the EPART domain.

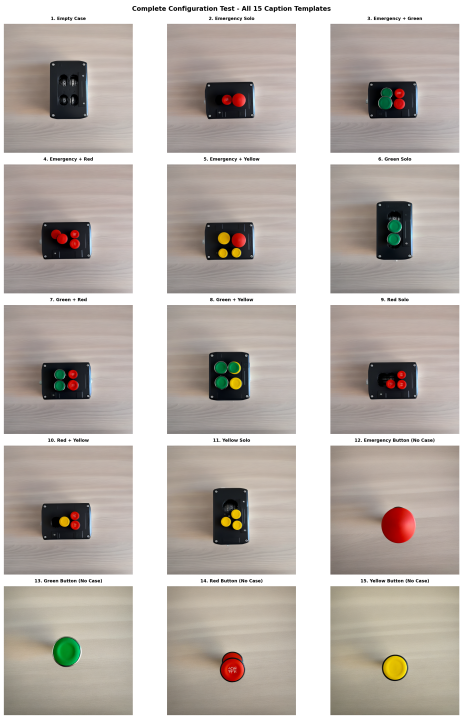


Figure 4.3: Complete grid of the 15 EPART device configurations generated by the LoRA-fine-tuned SDXL model at 512×512 resolution. The images exhibit fine details (mounting screws, sharp slot edges, realistic plastic and wood textures) and high visual realism, though compositional artifacts are present.

The first observation from the complete grid (Figure 4.3) is the exceptional level of detail captured by the model. In contrast to the GAN (128×128) and the from-scratch diffusion baseline (64×64), where many fine elements were blurred or entirely missing, the outputs of LoRA-fine-tuned SDXL model show features that bring image quality significantly closer to real photographs. Mounting screws at the corners of the case, fully absent in the previous models, are now clearly visible. The circular-slot boundaries are no longer gradual transition regions; instead, they present sharp and well-defined contours that distinctly separate the black case from the slot interiors. The plastic material of the case exhibits subtle micro-variations in brightness and slight surface imperfections that enhance realism.

The buttons benefit markedly from the higher resolution. In configurations such as "Emergency Solo" or "Green Solo", the circular surfaces display natural color gradients: brighter areas near the center suggest specular highlights, while slightly darker regions toward the edges convey three-dimensional curvature. This shading effect, largely absent at low resolution, is crucial to provide volume and depth, avoiding the flat appearance that typically characterizes lower-quality synthetic images.

Across the 15 configurations, compositional consistency is variable. Simple prompts such as single-button configurations are generally generated with correct layouts: the case structure is consistently reproduced, and single buttons are typically centered within the designated slots. However, more complex configurations reveal systematic compositional errors. In several two-button configurations, the model generates additional unwanted buttons (e.g., three or four buttons instead of two), or places buttons outside the circular slots, violating the fundamental geometric constraint of the EPART structure. Similarly, single-button prompts occasionally produce multiple instances of the same button, indicating that the text conditioning "one red button" is not reliably enforced. The emergency button, when correctly generated, is typically rendered larger than standard buttons, demonstrating that the model has learned this size distinction; however, in multi-button configurations involving the emergency button, the size difference is sometimes insufficiently pronounced or the emergency button appears duplicated.

These limitations stem from multiple concurrent factors. First, the training dataset is both small and insufficiently diverse: while 155 real images were used, expanding to the larger augmented dataset (5,000 images) used for GAN and Diffusion training did not improve compositional accuracy, as the augmented samples (rotations, flips, color jitter) do not introduce the semantic diversity needed to learn strict geometric constraints, augmentation replicates existing configurations rather than providing new compositional examples. Second, CLIP-based text conditioning has documented limitations on count-

ing, spatial relations, and negations [25]: the phrase "one red button" is encoded as a semantic concept ("presence of red button") rather than an exact cardinality constraint ("exactly one, no more"), allowing the stochastic diffusion sampling process to generate outputs with incorrect multiplicities while still aligning with the prompt embedding. Third, transfer learning from SDXL does not extend to domain-specific compositional rules such as "buttons must be placed inside circular slots" or "exactly two slots per case," which are unique to the EPART product and must be learned from training data alone. The combination of these factors, limited semantic diversity in the dataset, CLIP's compositional blindness, and the absence of geometric priors in the foundation model, results in occasional violations of structural constraints despite high perceptual realism.

A second contributing factor is the probabilistic nature of diffusion sampling. Even when a model captures the target distribution well on average, each generation remains intrinsically stochastic, and mid-probability failure modes, such as duplicated or mispositioned buttons, can occasionally occur. This behavior could be mitigated by increasing the number of training examples or by introducing explicit constraints at inference time (e.g., post-processing checks on cardinality and placement), but under the current setup it represents a relevant limitation.

Despite these issues, the LoRA approach exhibits distinctive capabilities that make it the most promising and versatile strategy among those evaluated. Beyond the superior visual quality enabled by the high resolution (512×512), a remarkable property emerges that is entirely absent in the from-scratch models: the ability to generalize to colors never seen during fine-tuning.

The original set of 155 images includes only four button-color categories: red, green, yellow, and emergency (red–orange). When prompted with colors that are completely absent from the fine-tuning data, the model generates buttons in the requested hues while preserving the EPART device structure (Figure 4.4).



Figure 4.4: Color generalization test: the LoRA model correctly generates buttons in colors entirely absent from the training dataset. This capability stems from SDXL's pre-trained visual knowledge: the frozen CLIP encoder preserves semantic understanding of all colors, while the pre-trained U-Net knows how to visually render them.

From Figure 4.4, it is evident that the model not only produces the requested colors, but does so with surprisingly accurate and well-separated tones. This color generalization constitutes a qualitative advantage over both the GAN and the from-scratch diffusion baseline. In the GAN setup, conditioning is expressed through discrete labels tied to the colors present in the dataset; consequently, generating entirely unseen colors is structurally impossible.

For the diffusion model trained from scratch, the situation is similar in practice. Although it is text-conditioned via CLIP and can in principle receive prompts specifying arbitrary colors, the U-Net is trained from random initialization on a dataset containing only red, green, yellow, and emergency buttons. While the frozen CLIP text encoder can produce meaningful embeddings for prompts such as "cyan button", the U-Net has never observed visual examples corresponding to those embeddings. During training, the model learns to associate the embeddings of "red", "green", "yellow", and "emergency" with the corresponding pixel patterns in real images; for an embedding such as "cyan", it lacks any grounded visual mapping. As a result, attempts to generate unseen colors with the from-scratch diffusion model would tend to yield ambiguous outputs, incorrect hues, or uncontrolled interpolations among known colors.

In contrast, LoRA inherits a U-Net pretrained on billions of images spanning the full visible color spectrum. During SDXL pretraining, the model observes countless instances of cyan, purple, mint, orange, rose gold, blue, and many other hues across diverse contexts, learning robust internal representations of how each color manifests visually in terms of RGB composition, saturation, brightness, and interaction with illumination. This general

color knowledge is encoded in the pretrained U-Net weights and remains largely intact during LoRA fine-tuning.

From an application standpoint, this color generalization greatly expands the model's utility. Rather than being restricted to the four colors present in the original dataset, one can generate EPART variants in dozens or hundreds of different colors simply by modifying the prompt, without collecting additional real images, labeling new data, or retraining the model.

It is important to emphasize that this color generalization, while remarkable, does not remove the structural limitations discussed above: cyan or purple buttons may still be duplicated, misplaced, or geometrically distorted, exactly as can occur for the red or green buttons seen during training. Generalization here concerns the chromatic dimension, not geometric precision or compositional correctness. Nonetheless, it constitutes a distinctive advantage of LoRA over from-scratch approaches and concretely illustrates the value of transfer learning: with a minimal investment (155 images, 800 steps, approximately one hour of training), the resulting model not only specializes to a specific domain but also retains generalization capabilities inherited from large-scale pretraining.

To further understand the effectiveness of LoRA fine-tuning, it is useful to directly compare generations produced by SDXL base (without LoRA) and by the same model after fine-tuning (with LoRA enabled), using identical prompts and a fixed seed (Figure 4.5).



Figure 4.5: Direct comparison between SDXL base and LoRA-fine-tuned SDXL generations for five representative EPART device configurations.

The difference is both evident and substantial. In the top row, SDXL base interprets the prompts in a way that is largely unrelated to the target domain, producing transparent suitcases or generic containers with colored elements inside, effectively treating "product case" as a generic storage case rather than an industrial control panel. The bottom row shows a clear transformation: after only 800 fine-tuning steps on 155 images, each

configuration is rendered in a manner consistent with the EPART domain, with a black rectangular case, well-defined circular slots, a realistic wooden background, and visible mounting screws.

This transformation, achieved by updating about 40 million parameters (approximately 1.5% of the total), highlights the efficiency of LoRA fine-tuning. With minimal compute and an extremely small dataset, the model becomes effectively specialized to a specific domain. By comparison, training SDXL from scratch would require massive computational resources and millions of images; the from-scratch diffusion baseline required eight hours on 5,000 images for lower qualitative results; and the GAN baseline required two hours on 5,000 images with an FID roughly three times worse.

4.5. Comparison between Models

This section provides a systematic quantitative comparison across the three evaluated generative approaches: a multi-label-conditioned DCGAN, a text-conditioned diffusion model trained from scratch and a LoRA fine-tuning of SDXL. The goal is to highlight strengths and weaknesses of each approach, thereby guiding model selection under different constraints such as computational budget, dataset size, and the quality requirements of the target industrial domain.

4.5.1. Overall quantitative comparison

Table 4.13 summarizes the quantitative results for all models across the considered metrics, together with training and inference characteristics.

Metric / Characteristic	GAN	Diffusion	LoRA
Distributional quality			
FID Score	202.06	90.63	**61.35**
Improvement vs. GAN	—	-55%	-70%
Structural fidelity			
SSIM (mean ± std)	**0.597**±0.057	0.562±0.097	0.507±0.055
SSIM (range)	[0.434, 0.783]	[0.291, 0.883]	[0.372, 0.662]
Semantic alignment			
Conditioning metric	91.4% (Label)	0.293 (CLIP)	**0.309 (CLIP)**
Conditioning type	6D binary	Text prompts	Text prompts
Color generalization	Impossible	Limited	Complete
Resolution and details			
Resolution	128×128	64×64	**512×512**
Fine details	Moderate	Absent	High
Training efficiency			
Dataset size	5,000 img	5,000 img	**155 img**
Training time with P100 GPU	2h	8h	**1h**
Trainable params	100% model	100% U-Net	1.5% (LoRA)
Pre-training	No	No	SDXL
Inference efficiency			
Time per image with P100 GPU	**0.1s**	5s (50 steps)	24s (30 steps)
Relative speed vs GAN	1×	0.02×	0.004×
Main limitations			
Quality	High FID	Low resolution	Compositional artifacts
Flexibility	Rigid cond.	Limited general.	CLIP counting limits
Scalability	Re-training req.	Re-training req.	Lightweight adapt.

Table 4.13: Overall quantitative comparison across GAN, diffusion model trained from scratch, and LoRA-fine.tuned SDXL.

The quantitative comparison highlights clear and fundamental trade-offs. LoRA achieves the best overall distributional quality (FID 61.35), the highest resolution (512×512), the strongest semantic alignment (CLIP 0.309), and the most efficient training regime (155 images, $\sim$1 hour). However, it exhibits a lower SSIM (0.507) and compositional artifacts (e.g., duplicated buttons or incorrect placement), plausibly due to the extremely limited fine-tuning dataset. The GAN attains the highest SSIM (0.597), reflecting structurally consistent outputs, but suffers from a very high FID (202.06) and a complete lack of chromatic flexibility. The diffusion model provides a balance between quality (FID 90.63) and diversity (SSIM std 0.097), but is strongly constrained by the 64×64 resolution, which removes essentially all fine details.

FID is the most informative indicator of overall perceptual quality in this setting, as it measures how close the generated samples are to the real-image distribution. The progression GAN (202.06) → diffusion (90.63) → LoRA (61.35) reflects a systematic improvement as more powerful generative paradigms are adopted and, crucially, as large-scale pretraining is leveraged.

The GAN FID (202.06) reflects the intrinsic limits of from-scratch adversarial training under a constrained dataset: the model learns coarse regularities (case shape and dominant colors) but produces images with flat textures, background artifacts, and limited intra-class variation, which remain statistically distant from real photographs. The improvement achieved by the DDPM (90.63, -55%) is consistent with iterative denoising, which tends to reduce coarse artifacts and yields smoother transitions, but it remains capped by the 64×64 resolution that drastically limits the available visual information. LoRA (61.35, -70% vs. GAN, -32% vs. diffusion) benefits from the combination of large-scale pretraining, high-resolution generation, and an SDXL architecture optimized for high-fidelity outputs.

At the same time, these values should be interpreted in context: none of the three approaches reaches the sub-30 FID often reported by frontier diffusion models on standard benchmarks, which is consistent with the intrinsic difficulty of the domain (small datasets, highly structured industrial objects) and the specificity of the task. Among the evaluated settings, LoRA is the closest to the "excellent" regime (often associated with FID < 50).

The SSIM analysis reveals an apparent paradox: the GAN achieves the highest SSIM (0.597), followed by diffusion (0.562) and LoRA (0.507), i.e., the reverse order compared to FID and perceived realism. This counterintuitive behavior is explained by two concurrent factors. First, SSIM is strongly resolution-dependent: LoRA operates at 512×512 pixels (16 times more pixels than GAN's 128×128, 64 times more than Diffusion's 64×64), and higher resolution images naturally exhibit lower SSIM due to more opportunities for local discrepancies in fine details, even when perceptual quality is superior. Second, the nature of the generative process plays a role: GAN's adversarial training produces outputs that closely adhere to training-set prototypes with low variability, resulting in high pixel-level similarity (high SSIM) but limited diversity and distributional quality (high FID). In contrast, diffusion-based models (both from-scratch and LoRA) employ iterative denoising, which introduces controlled stochastic variability that increases perceptual realism and distribution coverage (low FID) at the cost of reduced pixel-perfect similarity to specific reference images (lower SSIM).

This inverse relationship between SSIM and perceptual quality underscores the impor-

tance of multi-metric evaluation: SSIM provides meaningful information within each model (e.g., which configurations are structurally harder to reproduce), but cross-model SSIM comparisons must account for resolution differences. A low SSIM value at 512×512 resolution does not indicate inferior structural fidelity compared to a high SSIM at 128×128; rather, it reflects the finer spatial granularity at which structural comparison is being performed. For this reason, SSIM is evaluated alongside FID (which captures distributional quality and is less resolution-sensitive) and CLIP Score (which measures semantic alignment independently of pixel-level details), ensuring a comprehensive assessment that balances structural fidelity, perceptual realism, and semantic correctness. From an application standpoint, the choice between GAN outputs (higher SSIM at low resolution) and LoRA outputs (lower SSIM at high resolution) should prioritize the latter whenever the downstream task requires fine-grained visual cues such as texture recognition, edge detection, or precise geometric measurements.

Finally, the conditioning mechanisms differ substantially across approaches. The GAN uses a 6D binary conditioning vector (Label Accuracy 91.4%), while diffusion and LoRA use text prompts (CLIPScore 0.293 and 0.309, respectively). Although these conditioning metrics are not directly comparable, both indicate effective alignment between inputs and generated content. However, the qualitative implications are markedly different.

Binary conditioning in the GAN is rigid but precise: it specifies exactly which components should be present. Yet it is not scalable: introducing a new color or component requires extending the label space and re-training the model, and chromatic generalization is structurally impossible. The from-scratch diffusion model is text-conditioned and therefore more expressive, but the U-Net trained from random initialization effectively learns only the associations supported by the dataset (red, green, yellow, emergency). Prompts involving unseen colors would likely yield ambiguous outputs or uncontrolled interpolations because the model has never observed such colors during training. In contrast, LoRA inherits a pretrained U-Net with broad semantic coverage; as a result, it can generalize to unseen colors in a controlled manner, substantially expanding the practical utility of the system. With only 155 images covering four colors, LoRA-fine-tuning enables the generation of potentially hundreds of chromatic variants by prompt editing alone.

4.5.2. Comparative Summary

The comparative analysis shows that the three approaches occupy distinct points in the quality–efficiency–flexibility trade-off space, each offering specific advantages depending

on the target application scenario.

GAN excels in inference speed (∼0.1s per image) and structural consistency (SSIM 0.597), making it well suited for real-time deployment or resource-constrained hardware where moderate resolution (128×128) is acceptable. Its discrete conditioning provides precise control (Label Accuracy 91.4%); however, the rigidity of the binary attribute vector and the impossibility of chromatic generalization make it unsuitable for scenarios requiring high flexibility.

Diffusion from scratch provides an interesting compromise: higher overall quality than the GAN (FID -55%), greater generative diversity (SSIM std 0.097), and flexible text conditioning, without relying on pretrained foundation models. Nonetheless, the 64×64 resolution removes fine-grained details entirely, limiting its applicability to tasks where visual detail is secondary.

LoRA achieves the strongest distributional quality (FID 61.35, best overall), the highest resolution (512×512, preserving fine details), the best semantic alignment (CLIP 0.309), and high training efficiency (155 images). Its unique ability to generalize to unseen colors makes it the most versatile option for data augmentation. The main limitation is the presence of compositional artifacts (e.g., duplicated or mispositioned buttons), which can be attributed to the extremely small fine-tuning set and can be mitigated through post-processing or a moderate expansion of the training data.

For the EPART industrial domain considered in this work, LoRA emerges as the most promising solution in the majority of practical scenarios, combining high quality, chromatic flexibility, and efficient adaptation. The GAN remains competitive for edge or real-time settings where latency is critical, whereas the from-scratch diffusion model is a viable alternative when independence from pretrained models is a strict requirement. Ultimately, the optimal choice depends on task constraints: if quality and flexibility are prioritized, LoRA is preferable; if speed is critical, GAN; if full autonomy is required, diffusion. The concluding chapter discusses broader implications of these findings and outlines future directions for further improving the evaluated approaches.

5 | Conclusions and Future Developments

This thesis has systematically investigated the application of generative models for image synthesis in an industrial domain, using the EPART device as a case study. Three major paradigms of visual generation were studied: multi-label-conditioned GANs, diffusion models trained from scratch with text conditioning, and parameter-efficient fine-tuning of stable diffusion models via LoRA. The main contribution of this work does not lie solely in reporting quantitative scores, but rather in clarifying the structural trade-offs between visual quality, controllability, semantic flexibility, and computational efficiency in a realistic industrial setting, characterized by limited data availability and strict operational constraints. The experimental evidence clearly suggests that there is no universally "best" model; instead, each approach becomes optimal under different practical requirements.

Multi-label-conditioned GANs prove to be a robust and predictable solution, particularly effective when the domain is well defined and the number of configurations is limited. Their main strengths are the structural consistency of the generated images and the very fast inference time, which makes them suitable for real-time or resource-constrained applications. However, the rigidity of discrete conditioning and the difficulty in capturing fine visual details significantly limit both scalability and generalization.

Diffusion models trained from scratch represent a substantial step forward in terms of distributional image quality and semantic flexibility. Text conditioning enables a more natural and compositional description of configurations compared to binary attribute vectors, while iterative denoising supports higher structural diversity. Nevertheless, the high computational cost and the need to drastically reduce resolution to make training feasible constrain their applicability in practical industrial scenarios where fine details are often critical.

The LoRA-based fine-tuning approach emerge as the most effective in balancing quality, flexibility, and computational cost. Knowledge transfer from a foundation model such as SDXL enables high-resolution and semantically consistent image generation even with an

extremely small dataset. The ability to generalize to colors never observed during training is a particularly significant result, highlighting the value of large-scale pre-training. At the same time, the presence of compositional errors shows that, in domains with strict geometric constraints, knowledge transfer alone is not sufficient to guarantee perfect structural control.

Overall, these results suggest that, in industrial settings, the choice of a generative model should be guided not only by visual realism, but also by the type of control required (e.g. multilabels attributes or textual prompts), the available data, and the computational resources.

From an application standpoint, each approach appears particularly suitable for specific problem categories.

Conditional GANs are appropriate when deterministic and strict control over the structure of generated objects is required, for instance in synthetic dataset generation for industrial vision systems with discrete, well-defined configurations. In such contexts, the ability to produce coherent images with limited variability can represent an advantage rather than a drawback.

From-scratch diffusion models are better suited to exploratory or research-oriented scenarios, where the goal is to study model behavior without relying on pre-trained models. They enable controlled analyses of the impact of text conditioning and iterative denoising, but they are unlikely to be an efficient solution for real industrial applications under tight time and resource constraints.

LoRA, instead, appears particularly promising for advanced industrial applications, such as high-resolution synthetic dataset generation, product-variant simulation, and support for virtual prototyping. The possibility of obtaining high visual realism with a small amount of real data (155 images) opens practical opportunities for adopting generative models in contexts where data collection is costly or impractical. However, the compositional artifacts, such as incorrect button counts or spatial arrangement errors, indicate that the approach is most suitable for batch generation workflows where automated or manual filtering can be applied to select structurally correct samples, or for applications where exact geometric precision is not critical (e.g., visual exploration, concept design) rather than safety-critical industrial contexts requiring strict compositional guarantees.

5.1. Extension to Robotic Applications and Automated Assembly

A further application perspective of considerable relevance concerns integrating generated images with robotic systems for automated assembly of industrial components. High-quality synthetic images produced by generative models, should not be viewed only as a tool for dataset augmentation. They can also be interpreted as a visual representation of the final product, potentially usable as input for robotic perception and planning modules.

In an industrial scenario, a generated image of a complete device could serve as a visual reference for a robotic system tasked with assembling the device from its elementary components. Starting from a representation of the final object, a robot equipped with computer vision modules could identify constituent parts, infer their spatial arrangement, and plan an assembly sequence. In this setting, the generative model would act as a producer of "visual prototypes", while the robotic system would use these prototypes as guidance for action.

This perspective sits at the intersection of image generation, computer vision, and cognitive robotics. Synthetic images could be used to train component recognition models, pose estimation algorithms, and learning-based planning systems, reducing the need to manually acquire large amounts of real data. Moreover, the ability to generate multiple product configurations would allow the simulation of diverse assembly scenarios, improving system robustness to changes in layout, color, and configuration.

In an even more advanced direction, the process could be inverted: rather than only generating images from text, a generative model could support design and industrial planning. Starting from functional specifications or a desired layout, the system could generate plausible visual representations of the product, which could then be translated into operational instructions for a robot. In this sense, generative models could form a bridge between conceptual design and physical execution, contributing to reducing the gap between digital prototyping and automated production.

Although this scenario was not implemented in this thesis, the results obtained indicate that the quality and controllability of generated images are fundamental prerequisites for such applications. In particular, the capability of LoRA fine-tuned diffusion models to produce realistic, high-resolution images suggests that these models could become key components of future robotic assembly pipelines based on visual perception and learning.

5.2. Limitations of the Study

Despite the positive results, this work presents several intrinsic limitations that should be highlighted.

First, the dataset sizes used, while representative of a realistic industrial context, limit the generalizability of the results. In particular, generative model behavior may change substantially with larger or more heterogeneous datasets. The LoRA dataset, composed of only 155 images, is sufficient to demonstrate the effectiveness of transfer learning but remains near the lower bound of practical applicability. The observed compositional artifacts, such as duplicated buttons, incorrect placements, and geometric distortions, are directly attributable to insufficient semantic diversity in the training set rather than merely dataset size. While a larger augmented dataset (5,000 images) was available and used for GAN and Diffusion training, augmentation techniques (rotations, flips, color jitter) replicate existing configurations without introducing new compositional examples, e.g., augmenting "one red button" produces rotated/flipped versions of the same configuration, not examples of "two buttons" or "button outside slot." These artifacts could likely be substantially reduced with 500–1000 semantically diverse real images covering a wider range of compositional variations (e.g., more examples of edge cases, partially occluded buttons, varied slot occupancy patterns), while still preserving the efficiency and data requirements advantages of fine-tuning compared to training large models from scratch.

Second, the adopted metrics, although covering different dimensions of generative quality, are not free of limitations. FID is influenced by the domain bias of Inception-v3, trained on ImageNet and optimized for natural images rather than industrial components; SSIM is strongly resolution-dependent, making cross-resolution comparisons problematic; CLIPScore does not fully capture structural relations and geometric constraints, and may behave like a bag-of-words measure for complex prompts. This suggests that evaluating generative models in industrial settings requires more domain-oriented metrics, potentially based on feature extractors trained on industrial data or on directly measurable downstream tasks (e.g., accuracy of classifiers trained on synthetic data, detector performance on generated images).

5.3. Future Work

Building on the results obtained, several research and development directions emerge that could significantly extend and improve the present work.

A first direction concerns improving structural control in diffusion- and LoRA-based mod-

els. Integrating explicit geometric constraints, for instance through segmentations, depth maps, or layout guidance, could substantially reduce the compositional errors observed. Methods such as ControlNet, T2I-Adapter, or multimodal conditioning represent natural extensions: instead of providing only text prompts, one could jointly provide sketches of the desired layout, segmentation maps specifying where each component should appear, or bounding boxes constraining object positions. Such structured conditioning would preserve the semantic flexibility of text (e.g., colors, styles, materials) while adding hard geometric constraints that enforce compositional correctness.

Another promising direction is to train neural post-processing modules acting as "geometric correctors": networks that, given a generated image with potential artifacts, automatically detect and correct structural violations (extra buttons, incorrect placements, deformations). This approach separates visual generation from geometric validation, simplifying both tasks.

A second direction concerns iteratively expanding the synthetic dataset. A self-training strategy, where high-quality generated images are used to augment the training set, could progressively improve stability and accuracy while reducing reliance on real data. The iterative loop would operate as follows: generate a large batch of synthetic images with the current model, automatically filter out samples with evident artifacts using detectors or geometric validators, add the high-quality samples to the training set, and continue fine-tuning on the expanded dataset. This bootstrapping process could progressively reduce artifacts without requiring additional real data collection.

A natural methodological extension is to explore hybrid generative systems that combine the deterministic control of GANs with the semantic flexibility of diffusion models. For instance, architectures where a GAN enforces strict geometric consistency (case shape, number and position of slots) while a diffusion module generates high-frequency visual details (textures, colors, lighting) could offer a favorable trade-off. Moreover, leveraging more recent foundation models (e.g., Stable Diffusion 3, DALL-E 3) or models specialized for industrial content (fine-tuned on CAD datasets, technical renderings, or industrial photography) could further improve generation quality.

Finally, computational sustainability is highly relevant. The comparative analysis showed that LoRA-based transfer learning achieves superior results at a substantially lower computational cost than training from scratch (1 hour vs 8 hours, 155 images vs 5000). This suggests that future industrial adoption of generative models will increasingly rely on parameter-efficient and modular strategies rather than full model training.

Future directions include distilling LoRA fine-tuned models into smaller and faster ar-

chitectures while preserving high quality and reducing inference latency for real-time applications; developing multi-domain LoRA adapters that can be dynamically loaded depending on the target product, enabling a single base model to serve multiple production lines; and optimizing deployment on edge hardware (embedded GPUs, TPUs, or specialized accelerators) to enable on-device generation in distributed industrial settings.

5.4. General Conclusion

In conclusion, this thesis demonstrates that generative models can be successfully applied to industrial domains even under limited-data conditions, while highlighting that the choice of architecture and training strategy is decisive for the final outcome.

These results suggest that the future of industrial image generation will not rely on a single paradigm, but on an intelligent integration of different approaches capable of combining structural control, semantic flexibility, and pre-trained knowledge. The convergence between generative models, computer vision, and cognitive robotics, as discussed in the perspective on robotic applications, indicates that these systems will not be used only for data augmentation, but may become active components of integrated production pipelines, contributing to the design, simulation, and automation of complex industrial processes.

Bibliography

[1] Christoforos Aristeidou, Nikos Dimitropoulos, and George Michalos. Generative AI and neural networks towards advanced robot cognition. *CIRP Annals - Manufacturing Technology*, 73(1):21–24, 2024.

[2] Bojan Obrenovic, Xiao Gu, Guoyu Wang, Danijela Godinic, and Ilimdorjon Jakhongirov. Generative AI and human–robot interaction: implications and future agenda for business, society and ethics. *AI & Society*, 40:677–690, 2025.

[3] Kun Zhang, Peng Yun, Jun Cen, Junhao Cai, Didi Zhu, Hangjie Yuan, Chao Zhao, Tao Feng, Michael Yu Wang, Qifeng Chen, Jia Pan, Wei Zhang, Bo Yang, and Hua Chen. Generative artificial intelligence in robotic manipulation: A survey, 2025.

[4] Andrea Maria Zanchettin. End-to-end action model learning from demonstration in collaborative robotics. *Robotics and Autonomous Systems*, 193, 2025.

[5] Susmita Ray. A quick review of machine learning algorithms. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMIT-Con)*, pages 35–39, India, 2019. IEEE.

[6] Darpan Pandey, Kamal Niwaria, and Bharti Chourasia. Machine learning algorithms: A review. *International Research Journal of Engineering and Technology (IRJET)*, 6(2):2395–0056, 2019.

[7] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models, 2017.

[8] Su Wang. Generative adversarial networks (GAN): A gentle introduction. Tutorial document, 2017. Tutorial on GAN in LIN395C: Research in Computational Linguistics, University of Texas at Austin.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[10] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Dif-

fusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10850–10869, 2023.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020. NeurIPS 2020.

[12] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. Workshop paper at NeurIPS 2021.

[13] Hyunwoo Yoo. Fine tuning text-to-image diffusion models for correcting anomalous images, 2024. Drexel University.

[14] Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021.

[15] Youssef Sultan, Jiangqin Ma, and Yu-Ying Liao. Fine-tuning stable diffusion XL for stylistic icon generation: A comparison of caption size, 2024.

[16] Samar Pratap, Alston Richard Aranha, Divyanshu Kumar, Gautam Malhotra, Anantharaman Palacode Narayana Iyer, and S. S. Shylaja. The fine art of fine-tuning: A structured review of advanced LLM fine-tuning techniques. *Natural Language Processing Journal*, 11:100144, 2025.

[17] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[19] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[20] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7514–7528, 2021.

[21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini

Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[22] Min Jin Chong and David Forsyth. Effectively unbiased fid and inception score and where to find them. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6070–6079, 2020.

[23] Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking fid: Towards a better evaluation metric for image generation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[24] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[25] Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic compositionality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5238–5248, 2022.

[26] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.

List of Figures

2.1 Training and testing of a machine learning model [6] 8
2.2 Schematic of a fully connected artificial neural network 9
2.3 Schematic of the convolution operation in a Convolutional Neural Network. A kernel is applied locally to an input tensor: an element-wise product, followed by a summation generate the values of the output feature map. . . 11
2.4 Conceptual diagram of an encoder–decoder architecture. The encoder compresses the input into a lower-dimentional representation (the latent space), while the decoder reconstructs the output from the latent space. . . 14
2.5 Conceptual diagram of a Generative Adversarial Network. The black dotted curve represents the true data distribution p_{data}, while the green curve denotes the distribution of samples generated by the generator p_g. The blue dashed curve corresponds to the discriminator output, i.e., an estimate of the probability that a given sample originates from the real data distribution rather than from the generator. The generator maps a latent variable z to a synthetic sample, while the discriminator evaluates the probability that a sample comes from the real data distribution. The training process occurs through an adversarial mechanism that guides the alignment between p_g and p_{data} [9]. 18
2.6 Example of the forward diffusion process. Starting from the clean data x_0, Gaussian noise is progressively added until, after T iterations, a distribution close to a standard isotropic Gaussian is obtained. Each intermediate state x_t represents a partially degraded version of the original data. [10] 20

2.7 Effect of Classifier-Free Guidance on the generation process. The leftmost image corresponds to the unguided case ($s = 0$), in which the model generates unconditioned samples characterized by high diversity but weak adherence to the conditioning signal (i.e., limited coherence with the target class or text prompt). Moving to the right, increasing the guidance parameter s progressively strengthens the influence of the condition, making the generated samples more consistent with the specified class/text content, at the cost of reduced generative variability [12]. 23
2.8 High-level schema of the fine-tuning cycle for the text-to-image Stable Diffusion Model 27
2.9 Comparison between generation with base model (a) and fine-tuned model (b). Fine-tuning significantly improves image quality, with more realistic details and a more natural composition. [13] 29
2.10 LoRA architecture: the pre-trained weights W_0 remain frozen, while adaptation occurs through the low-rank decomposition BA with $r \ll d$. The typical initialization is $B = 0$, $A \sim \mathcal{N}(0, \sigma^2)$. [14] 31

3.1 Examples of different configurations present in the dataset: case with an emergency button (a), combinations of colored standard buttons (b), empty cases (c), and buttons without a case (d). 36
3.2 Text-conditioning mechanism in diffusion models. The prompt is encoded into semantic embeddings and injected into the U-Net denoising network through cross-attention layers (Q, K, V). At each denoising step, the textual representation conditions the reconstruction of the image, guiding the generation process toward the desired semantic content. [17] 48
3.3 Examples of EPART training samples paired with their corresponding captions. The structured text explicitly encodes the configuration (slots, button type/color, and case), providing a stable conditioning signal during LoRA fine-tuning. 56

4.1 Samples generated by the GAN model across the 15 configurations. Resolution: 128×128 pixels. 76
4.2 5×3 grid of images generated by the diffusion model across all 15 categories (64×64 resolution). Text prompts conditioning through cross-attention mechanisms, yields samples that are semantically coherent with the input descriptions. 85

4.3 Complete grid of the 15 EPART device configurations generated by the LoRA-fine-tuned SDXL model at 512×512 resolution. The images exhibit fine details (mounting screws, sharp slot edges, realistic plastic and wood textures) and high visual realism, though compositional artifacts are present. 95
4.4 Color generalization test: the LoRA model correctly generates buttons in colors entirely absent from the training dataset. This capability stems from SDXL's pre-trained visual knowledge: the frozen CLIP encoder preserves semantic understanding of all colors, while the pre-trained U-Net knows how to visually render them. 98
4.5 Direct comparison between SDXL base and LoRA-fine-tuned SDXL generations for five representative EPART device configurations. 99

List of Tables

3.1 Distribution of images in the dataset after augmentation 37
3.2 Semantic mapping of the multi-label vector 39
3.3 Generator architecture for the synthesis of 128 × 128 images 40
3.4 Discriminator architecture for the synthesis of 128 × 128 images 41
3.5 Summary of training hyperparameters for the Conditional GAN 44
3.6 Conditional diffusion U-Net architecture adopted for EPART images generation (64 × 64) 46
3.7 Components of the text-conditioned Diffusion model 49
3.8 Training hyperparameters adopted for the diffusion model 51
3.9 Examples of captions used for SDXL LoRA fine-tuning, with semantic-control segments highlighted in bold. 55
3.10 Hyperparameter configuration for LoRA fine-tuning of Stable Diffusion XL. 60

4.1 Summary of evaluation metrics 69
4.2 Examples of images in dataset used by the GAN model under different multi-label conditioning vectors. 70
4.3 Quantitative results for the GAN model (DCGAN). 71
4.4 Per-component Label Accuracy (GAN). 75
4.5 GAN qualitative summary: observed strengths and limitations. 77
4.6 Quantitative results of the diffusion model trained from scratch. 80
4.7 Diffusion model SSIM by category: best and worst configurations. 82
4.8 Diffusion model CLIP Score by prompt complexity. 84
4.9 Qualitative summary: strengths and limitations. 87
4.10 Quantitative results of the LoRA-fine-tuned SDXL. 89
4.11 SSIM of the LoRA model by category: best and worst configurations. . . . 92
4.12 CLIP Score of the LoRA-fine-tuned SDXL model by prompt complexity. . 94
4.13 Overall quantitative comparison across GAN, diffusion model trained from scratch, and LoRA-fine.tuned SDXL. 101

Acknowledgements

In quest'ultima pagina desidero ringraziare tutte le persone che mi hanno accompagnato e sostenuto in questo importante percorso.

In primo luogo, vorrei ringraziare il Prof. Zanchettin per l'opportunità che mi ha offerto .

Un ringraziamento speciale va a Eleonora, con la quale ho condiviso questi ultimi mesi, a volte difficili, ma che con pazienza e dedizione mi ha sempre aiutato a superare gli ostacoli e a migliorare.

Ai miei genitori, Laura ed Eugenio, da sempre esempio da seguire e punto di riferimento solido su cui poter fare affidamento in ogni momento. Grazie per il vostro supporto costante e per avermi trasmesso, attraverso i vostri insegnamenti, i valori che mi hanno reso la persona che sono contento di essere oggi. Grazie per avermi insegnato a credere in me stesso, a non arrendermi di fronte alle difficoltà e a riconoscere ciò che è davvero importante nella vita.

A mio fratello Gabriele, per rendermi ogni giorno una persona migliore, per i suoi consigli preziosi e per essere sempre presente nei momenti di difficoltà. Anche con poche parole, è sempre riuscito ad aiutarmi a superare ansie e paure. Grazie per riuscire a farmi ridere anche nelle situazioni più difficili e per essere sempre un punto di riferimento.

Alle mie care nonne, Gabriella e Maria Teresa, per avermi sempre dato l'affetto incondizionato che solo le nonne sanno dare. Grazie per aver gioito insieme a me, e forse più di me, per ogni traguardo raggiunto.

A mio zio Gianfranco e a mia zia Alessandra, per aver condiviso con me ogni risultato e per aver sempre creduto in me.

A tutti i miei amici. Agli amici che conosco ormai da dieci anni, con i quali ho condiviso una parte fondamentale della mia vita come il liceo: grazie per tutte le serate in cui, ridendo e scherzando, mi avete fatto dimenticare gli stress quotidiani. Grazie per tutte le esperienze vissute insieme e che spero di continuare a vivere per molti anni ancora. Grazie perché, nonostante la distanza che spesso ci separa, mi dimostrate ogni giorno

quanto per voi sia importante mantenere il nostro legame e rivederci appena gli impegni ce lo permettono.

Agli amici che quotidianamente mi sostengono e mi fanno divertire, grazie. Grazie a Gorazza Vibes per avermi fatto comprendere ogni giorno il significato profondo dell'amicizia, e per avermi insegnato che anche con le cose più semplici ci si può sempre divertire. Grazie per tutti i picnic al MID, i weekend trascorsi insieme, le serate passate a chiacchierare o guardare un film. Grazie per tutte le proposte che hanno trasformato giornate grigie in momenti carichi di un'energia speciale.

Infine, a Emma, mia compagna di vita. Questi anni universitari, percorsi interamente al suo fianco, sono stati resi indimenticabili dalla sua presenza certa e dal suo supporto incondizionato. Grazie per aver condiviso con me gioie e difficoltà, per non avermi mai fatto dubitare di me stesso, e per ricordarmi sempre, attraverso una frase, bacio o abbraccio, quanto sia bella la vita che costruiamo ogni giorno insieme. Grazie per aver reso questo percorso memorabile, e per continuare ogni giorno a migliorare ciò che siamo.

Grazie

Fra