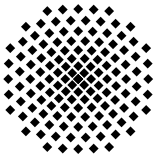


University of Stuttgart
Institute for Signal Processing and System Theory
Professor Dr.-Ing. B. Yang

Masterarbeit D1501

Generation and Evaluation of Synthetic Test Images in an Industrial Context

Author: Swathi Kumar

Date of work begin: 22.04.2024
Date of submission: 22.10.2024

Supervisor: Dominik Track, M. Sc.
Martin Wimpff, M. Sc.
Prof. Dr.-Ing. Bin Yang

Keywords: Synthetic Image Generation
Generative Models
Anomaly Detection

Abstract

Images captured in production environments exhibit various camera influences that the anomaly detector must learn to recognize. Without training on such diverse conditions, the detector is prone to misclassification and reduced performance. This master's thesis focuses on generating and evaluating synthetic test images to enhance the performance of these anomaly detectors in industrial contexts. Generative models, such as Cycle-Consistent Generative Adversarial Network (CycleGAN), were used to translate optimal setup images into ones with camera influences, while Stable Diffusion generated high-quality synthetic images that introduced subtle variations to real images. The quality and similarity of these synthetic images were evaluated through various metrics to ensure their effectiveness. The performance of the anomaly detector was evaluated through different training approaches. The results demonstrated a significant reduction in false positives when trained with images (both real and synthetic) that included those camera influences.

Bilder, die in Produktionsumgebungen aufgenommen werden, zeigen verschiedene Kameraeinflüsse, die einer Anomalieerkennung beigebracht werden müssen, um sie zu erkennen. Ohne Training auf solch vielfältige Bedingungen neigt der Detektor zu Fehlklassifikationen und einer verringerten Leistung. Diese Masterarbeit konzentriert sich darauf, synthetische Testbilder zu erzeugen und zu bewerten, um die Leistung dieser Anomaliedetektoren in industriellen Kontexten zu verbessern. Generative Modelle wie das Cycle-Consistent Generative Adversarial Network (CycleGAN) wurden verwendet, um Bilder, die mit einem optimalen Setup aufgenommen wurden, in solche mit Kameraeinflüssen zu übersetzen, während Stable Diffusion hochwertige synthetische Bilder generierte, die subtile Variationen zu echten Bildern hinzufügten. Die Qualität und Ähnlichkeit dieser synthetischen Bilder wurden anhand verschiedener Metriken bewertet, um ihre Wirksamkeit sicherzustellen. Die Leistung des Anomaliedetektors wurde anhand verschiedener Trainingsansätze bewertet. Die Ergebnisse zeigten eine signifikante Reduzierung von Falschpositiven, wenn der Detektor mit Bildern (sowohl realen als auch synthetischen) trainiert wurde, die diese Kameraeinflüsse enthielten.

Contents

1. Introduction

1.1. Motivation

Deep learning models thrive on large volumes of data, as vast and diverse datasets are essential for training algorithms that achieve high accuracy and generalization [1]. The performance of these models improves significantly with more data, allowing them to learn complex patterns and make more reliable predictions. However, acquiring such extensive datasets often presents significant challenges. Traditional data collection methods, especially in industrial settings can be costly, time-consuming, and impractical, if not impossible. Manual data annotation adds another layer of complexity, requiring considerable human effort and expertise. This creates a bottleneck in developing robust machine learning systems that require extensive, varied, and well-labelled data [2].

To overcome these challenges, synthetic data has emerged as a powerful alternative. Synthetic data is artificially generated by AI algorithms trained on real datasets, aiming to replicate the statistical properties and patterns of the original data. By modeling the probability distribution of real data and sampling from it, synthetic data generation creates new datasets that retain similar characteristics and predictive power as the original data [3]. This method enables the rapid creation of large, diverse datasets, including rare or difficult-to-capture scenarios, thus addressing the limitations of traditional data collection. As a result, synthetic data facilitates more efficient model development and training, enabling better generalization and performance of machine learning systems.

1.2. Problem Statement and Objectives

In industrial environments, anomaly detection systems are crucial for ensuring production efficiency and quality. However, these detectors often fail to reliably distinguish between defective and non-defective products when they are not trained with enough images that capture real-world variations, such as brightness, shadowing, reflections, and distortions. Without exposure to these variations during training, the detectors may misclassify normal products affected by such influences as defective, even though they are not true anomalies. This underscores the importance of training data that reflects the diverse conditions found in actual production setups.

The objective of this thesis is to enhance the accuracy and robustness of an anomaly detector by providing it with synthetically generated images that incorporate different influences. Since collecting sufficient real images to represent the full spectrum of possible influences is often time-consuming and impractical, this approach aims to reduce the reliance on extensive real-world data, by using two different generative models to create realistic synthetic data

that accurately replicate these artifacts. The generated synthetic images will then be evaluated using various metrics to assess their quality and similarity to real images before being integrated into the training dataset for the anomaly detector. By training the anomaly detector on this comprehensive dataset, the goal is to improve its ability to distinguish between genuine defects and normal variations caused by different camera effects.

1.3. Outline

The introduction provides a comprehensive overview of the significance of synthetic image generation in enhancing the efficacy of anomaly detection systems within industrial contexts, outlining the motivation and objectives of the research. Chapter 2 provides essential theoretical foundations related to deep learning and anomaly detection, starting with an overview of deep learning principles and a comparison of discriminative and generative models. It further explores synthetic image generation techniques, focusing on Generative Adversarial Networks (GANs), particularly CycleGAN, and diffusion models, while also discussing the role of autoencoders in anomaly detection. In Chapter 3, the materials and methods are detailed, describing the datasets utilized, including the Nuts, Candles, BNI, and PCB datasets. This chapter also outlines the methodologies for generating synthetic images using CycleGAN and Stable Diffusion, alongside the evaluation metrics employed to assess image quality and the anomaly detection system's setup. Chapter 4 presents the experimental results from CycleGAN and Stable Diffusion, including the results of evaluation metrics. It concludes by analyzing the impact of synthetic images on the performance of the anomaly detector, highlighting a significant reduction in false positives. Finally, Chapter 5 summarizes the thesis's key findings, discussing its contributions to the field of anomaly detection and offering recommendations for future research directions that highlight the potential of synthetic data in industrial applications.

2. Theoretical Background

This section introduces key concepts essential for understanding the generation of synthetic data and its application in anomaly detection. It begins with deep learning, a powerful method that allows models to learn complex patterns from large datasets. A distinction is drawn between discriminative models, which classify data based on learned patterns, and generative models, which produce new data resembling the original dataset. Key generative approaches, such as GANs and Diffusion Models, are covered, with a focus on CycleGAN (a conditional GAN) and Stable Diffusion, both of which are central to generating synthetic images. The significance of synthetic data in overcoming challenges related to data scarcity and variability is discussed, along with an overview of anomaly detection techniques, crucial for identifying deviations from expected patterns in industrial applications.

2.1. Deep Learning

Deep learning, a branch of machine learning, employs deep neural networks, which are multilayered architectures designed to emulate the intricate decision-making capabilities of the human brain. This approach underpins many of the artificial intelligence (AI) applications prevalent in today's technology [4].

Neural networks, or artificial neural networks (ANN), are designed to mimic the human brain by using interconnected layers of nodes (neurons) that process information in a similar manner [5]. These nodes are associated with weights and biases, which influence the input data to make accurate predictions or classifications. This structure allows neural networks to identify and interpret intricate patterns in data, making them versatile tools for various tasks involving detailed analysis and decision-making [5, 6].

Deep neural networks (DNN) consist of multiple layers, where each layer builds on the previous one to refine and optimize the predictions. This process, known as **forward propagation** [7], involves passing data through the input layer, where the deep learning model ingests the data, and through a series of hidden layers that apply transformations. The final output layer produces the prediction or classification. Each layer extracts progressively more abstract features from the data, enabling deep learning models to understand intricate patterns in images, such as shapes, textures, and complex structures.

To improve the model's accuracy, a process called **backpropagation** [8] is used. Backpropagation calculates the error between the predicted output and the actual target value, using algorithms like gradient descent to adjust the weights and biases of the network. By iteratively updating these parameters through the layers, the model learns to minimize errors and improve performance [9]. Over time, this combination of forward propagation and backpropagation enables the neural network to make increasingly accurate predictions.

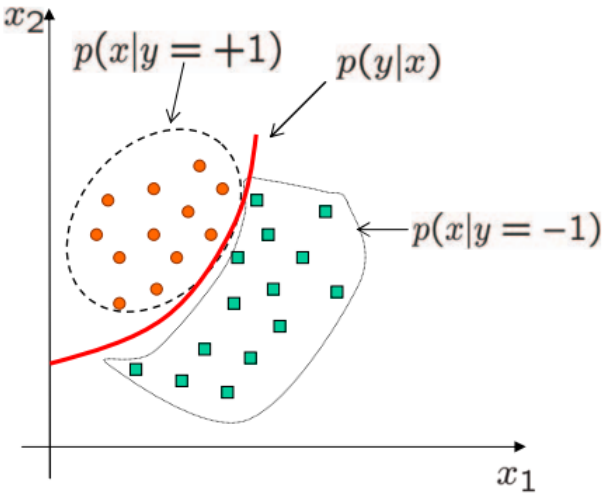


Figure 2.1.: Comparison of generative and discriminative models. Discriminative models concentrate on identifying the boundaries that separate different classes, whereas generative models are centered on producing data within each class [10]

2.2. Discriminative Vs. Generative Models

In machine learning, models are often categorized into two broad types: discriminative models and generative models. Understanding the distinction between these two approaches is crucial for selecting the appropriate model for specific tasks, such as classification or data generation.

2.2.1. Discriminative Models

Given a training dataset consisting of data points X and their corresponding labels Y, a discriminative model learns the conditional probability distribution $P(Y|X)$ [11]. This model focuses on modelling the decision boundary between classes, directly learning the mapping from input data X to the output labels Y. The goal is to find the best function that can separate or classify the data points based on their features. By estimating $P(Y|X)$, the model can predict the class of new data points by determining which class label has the highest probability given the input features.

Discriminative models are particularly useful in classification and regression tasks where the objective is to assign a label or a continuous value to each data point based on learned patterns and features [12]. These models do not attempt to model the underlying distribution of the data but rather focus on the decision boundaries between different classes.

2.2.2. Generative Models

Generative models aim to learn the underlying distribution of a dataset in order to generate new data points that resemble the original data. Given a training dataset of data points X and their corresponding labels Y, a generative model learns the joint probability distribution $P(X, Y)$ or the conditional distribution $P(Y|X)$ [11]. By modeling the data in this way, the

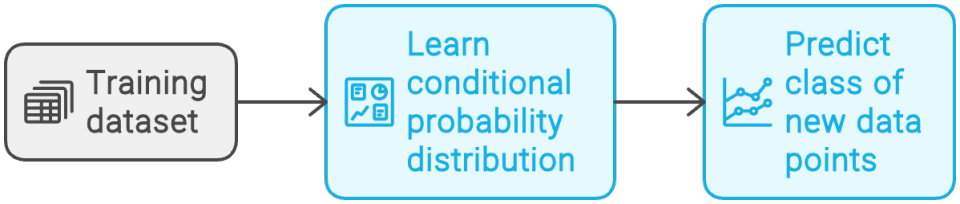


Figure 2.2.: Discriminative Model Workflow [13]

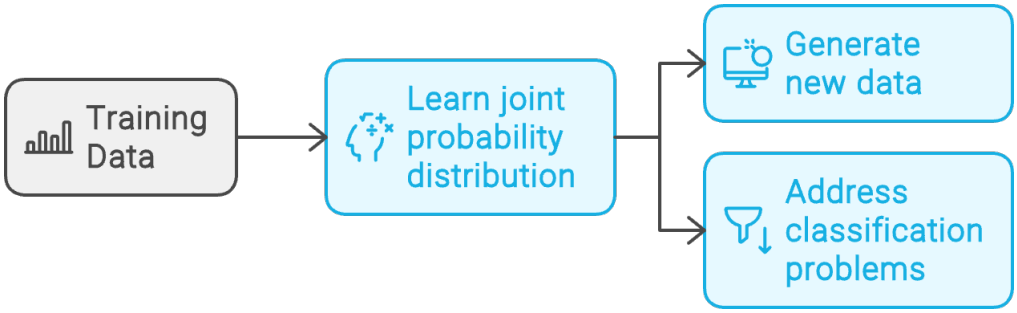


Figure 2.3.: Generative Model Workflow [13]

model can generate new samples that share characteristics with the training data, which is useful for tasks like data augmentation and simulation [14]. While generative models are primarily designed for tasks like data synthesis and simulation, they can also be used for discrimination by applying Bayes' rule [15]:

$$P(Y|X) = \frac{P(X,Y)}{P(X)} \tag{2.1}$$

However, this approach is rarely employed in practice, as discriminative models are typically more effective for classification tasks. Generative models are particularly valuable in scenarios where the goal is to understand the underlying data distribution or to generate new data samples. Unlike discriminative models, which focus solely on predicting or classifying data points, generative models attempt to model how the data was generated. This approach enables the creation of realistic synthetic data and is useful in applications such as image synthesis, data augmentation, and anomaly detection. By capturing the full distribution of the data, generative models provide a more comprehensive understanding of the data and its variations [13]. Given that this thesis centers on synthetic image generation, generative models are fundamental to accomplishing this goal.

2.3. Synthetic Image Generation

Synthetic data refers to data that is artificially generated to replicate the statistical properties and underlying patterns of real-world data [16]. For training machine learning models, synthetic data is employed to augment or substitute real-world datasets, especially when obtaining real data is constrained by scarcity, cost, or privacy concerns [17, 18]. By utilizing advanced methods such as computer simulations and generative models, synthetic data can be produced in large volumes, encompassing a wide range of scenarios and variations. This

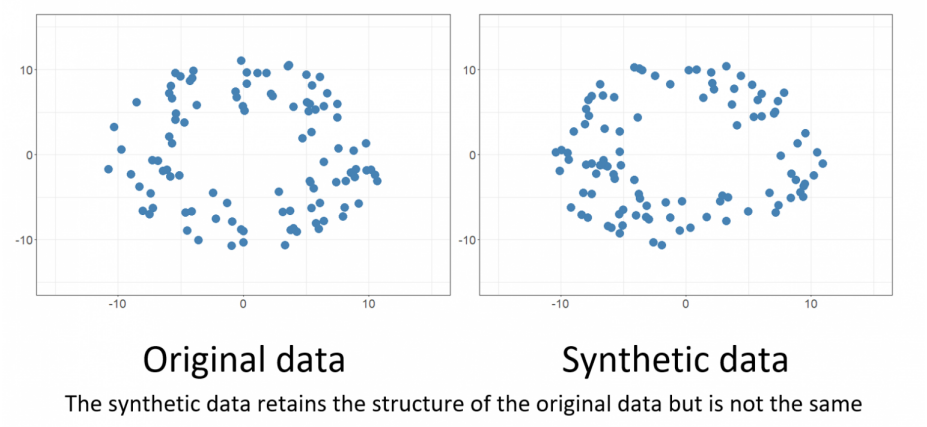


Figure 2.4.: The image shows two scatterplots: one of original data and one of synthetic data generated from the original data. The synthetic data retains the structure of the original data but is not the same. This is a common example of how generative models can be used to create new data that is similar to the original data, but not identical [19]

approach not only facilitates the training and validation of models but also ensures that sensitive or proprietary information remains protected, while still providing the comprehensive and representative data needed for robust model performance. In the context of image generation, synthetic images play an essential role in enhancing model training, especially for deep learning applications such as object detection, classification, and anomaly detection [2]. These images are generated to replicate specific features or conditions seen in real images but with the flexibility to introduce controlled variations [20]. Synthetic images are commonly used to address challenges like data imbalance, variability in test conditions, or when acquiring real images is impractical or cost-prohibitive.

Generative models, such as Generative Adversarial Networks, Diffusion Models, and Variational Autoencoders (VAEs) [21], are instrumental in the generation of synthetic images. GANs utilize a generator and a discriminator network in a competitive setting to create highly realistic images [22]. Diffusion Models work by progressively denoising random noise to generate coherent images, offering high fidelity and versatility [23]. These models excel in generating high-quality synthetic images that closely mimic real data, which enhances model robustness and accuracy in handling varied scenarios. The following Chapters 2.4 and 2.5 will provide an in-depth examination of GANs and Diffusion Models, detailing their methodologies and applications in synthetic image generation.

On the other hand, Variational Autoencoders are another type of generative model that can produce synthetic images. While VAEs are effective at learning a compact latent representation of data and generating diverse samples, they often struggle with producing high-quality images with fine details [24]. The images generated by VAEs can sometimes appear blurred or less realistic compared to those produced by GANs or Diffusion Models. This limitation makes VAEs less suitable for tasks requiring high-resolution and detailed synthetic images.

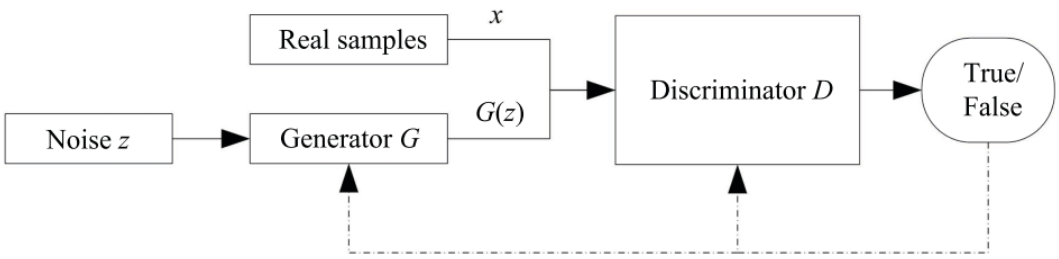


Figure 2.5.: Structure of GAN [28]

2.4. Generative Adversarial Networks

Generative Adversarial Networks, introduced by Goodfellow *et al.* in 2014 [25], represent a powerful class of generative models. Central to GANs is the concept of **adversarial training**, which is inspired by Nash equilibrium [26] in game theory. In this framework, two entities, the generator and the discriminator, engage in a competitive game. The generator's objective is to learn the distribution of real data and produce synthetic data that closely resembles it, while the discriminator's role is to distinguish between real and generated data. The game involves both entities continuously refining their strategies: the generator improves its ability to produce realistic data, and the discriminator enhances its accuracy in detecting fake data. This iterative process seeks to achieve a Nash equilibrium, where the generator's outputs are indistinguishable from real data according to the discriminator [27].

The structure of GAN is shown in Figure 2.5. Any differentiable function can be used as the generator and the discriminator. The differentiable functions D and G are used to represent the discriminator and the generator, and their inputs are real data x and random variables z, respectively. To learn the generator's distribution p_g over the data x, a prior distribution $p_z(z)$ is defined on the input noise variables. A mapping from the noise space to the data space is represented by $G(z;\theta_g)$, where G is a differentiable function modeled as a multilayer perceptron with parameters θ_g. Alongside, a second multilayer perceptron $D(x;\theta_d)$ is defined that outputs a single scalar. $D(x)$ estimates the probability that x originates from the real data distribution rather than from p_g. The discriminator D is trained to maximize the probability of correctly classifying both real data examples and samples generated by G. Simultaneously, the generator G is trained to minimize $\log(1 - D(G(z)))$ [28].

This setup creates a two-player minimax game with the following value function $V(G, D)$:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.2}$$

Here, $\mathbb{E}$ denotes the expectation over the respective distributions.

This framework captures the essence of **unconditional GANs** [29], where the primary focus is on learning the data distribution solely from random noise. In these models, the generator creates data samples based solely on the input noise z without additional context or control over the specific characteristics of the generated data. As a result, unconditional GANs are adept at learning the overall data distribution but lack the capability to control the generation of specific modes or attributes within the data [30].

To achieve more precise control over the generated data, **conditional GANs** [31, 32] utilize

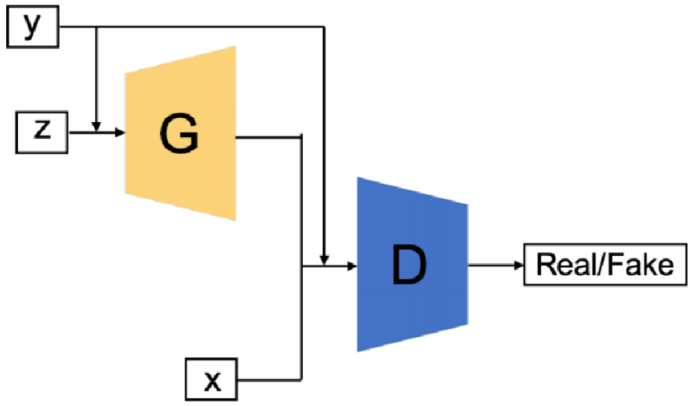


Figure 2.6.: Conditional GAN architecture [37]

additional information to guide the data generation process. This approach, which enables targeted generation of data with specific attributes, will be discussed in the following section.

Several prominent unconditional GAN variants have made substantial contributions to the development of generative techniques. The original GAN, introduced by Goodfellow *et al.* [25], laid the groundwork for using adversarial training to generate realistic data from noise. The **Deep Convolutional GAN (DCGAN)** [33] enhanced this by incorporating convolutional layers in both the generator and discriminator, leading to improved image quality. The **Wasserstein GAN (WGAN)** [34] addressed training instability by employing the Wasserstein distance as a loss metric, providing more stable and reliable convergence. The **Least Squares GAN (LSGAN)** [35] introduced the use of least squares loss instead of binary cross-entropy, which helps in achieving more stable training and high-quality images. Additionally, the **Progressive Growing GAN (PGGAN)** [36] introduced a technique of progressively increasing image resolution during training, significantly improving the quality of generated images. These developments in unconditional GANs have laid the foundation for more advanced models, including those that incorporate additional controls over the data generation process.

2.4.1. Conditional GANs

Conditional generative adversarial networks extend the traditional GAN architecture by conditioning both the generator and discriminator on additional information y, which can include class labels, attributes, or other types of data [31]. This conditioning allows the model to direct the generation process, enabling it to produce outputs that align with the given input condition. In this setup, the generator receives both a noise vector z and the condition y, combining them into a joint hidden representation to generate data. Meanwhile, the discriminator is provided with both the real or generated data x and the condition y, aiming to assess whether the generated data matches the given condition.

The objective function of cGANs modifies the original GAN formulation as follows [38]:

$$\min_{G} \max_{D} V(G,D) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x|y)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1 - D(G(z|y)))\right] \tag{2.3}$$

The incorporation of conditional information significantly enhances the adaptability of the GAN framework, making it particularly well-suited for tasks that require controlled outputs,

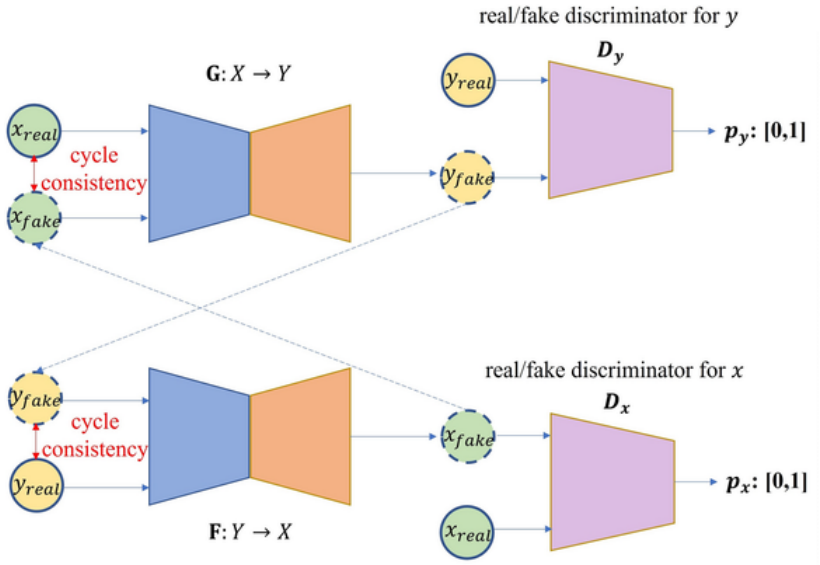


Figure 2.7.: Overview of CycleGAN architecture [42]

such as **image-to-image translation**. In these scenarios, the model is tasked with transforming an image from one domain into another based on specific input conditions. By conditioning on auxiliary data, conditional GANs allow for finer control over the generated content, ensuring that the output aligns closely with the given input while maintaining high visual fidelity.

One widely recognized cGAN for image-to-image translation is **Pix2Pix**, introduced by Isola *et al.* in 2017 [39]. This model is specifically trained on paired datasets, where each input image has a corresponding target image in the other domain. By conditioning the GAN on the input image, Pix2Pix enables the generator to learn the mapping between two domains, effectively translating an image from one domain to its target counterpart. The generator in Pix2Pix aims to produce outputs that match the target image, while the discriminator evaluates whether the generated image is the true paired image or a synthesized one. Through this adversarial process, Pix2Pix is highly effective in a wide range of image translation tasks, such as converting sketches to photographs [40], grayscale to color images, or even transforming satellite imagery into maps. However, Pix2Pix relies heavily on having paired training data, which limits its applicability in cases where paired data is scarce or unavailable.

2.4.2. CycleGANs: Unpaired Image-to-Image Translation

Cycle-Consistent Generative Adversarial Networks were introduced by Zhu *et al.* in 2017. CycleGANs extend the concept of GANs to situations where paired training data is not available by leveraging unpaired datasets [41].

Unlike Pix2Pix, which requires paired images (x_i, y_i), where x_i is an image from Domain A and y_i is its corresponding image in Domain B, CycleGANs work with two sets of unpaired images $\{x_i\}$ from Domain A and $\{x_i\}$ from Domain B.

The CycleGAN framework consists of two generator networks $G : \mathcal{X} \rightarrow \mathcal{Y}$ and $F : \mathcal{Y} \rightarrow \mathcal{X}$, and two discriminator networks D_X and D_Y. Here, G translates images from Domain A to

Domain B, while F translates images from Domain B to Domain A. The discriminators D_X and D_Y aim to differentiate between real images and generated images in their respective domains [41, 43].

The key innovation in CycleGANs is the cycle-consistency loss [41, 44], which enforces that an image translated from Domain A to Domain B and then back to Domain A should approximately match the original image. This can be mathematically expressed as:

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x\sim\mathcal{X}}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y\sim\mathcal{Y}}[\|G(F(y)) - y\|_1] \tag{2.4}$$

where $\mathcal{L}_{\text{cyc}}$ denotes the cycle-consistency loss, $\|\cdot\|_1$ represents the L_1 norm, and $\mathbb{E}$ denotes the expectation over the respective domains.

In addition to the cycle-consistency loss, CycleGANs use adversarial losses [45] for each domain to ensure the generators produce outputs that are indistinguishable from real images. The adversarial losses can be defined as:

$$\mathcal{L}_{\text{adv}}(G, D_Y) = \mathbb{E}_{y\sim\mathcal{Y}}[\log D_Y(y)] + \mathbb{E}_{x\sim\mathcal{X}}[\log(1 - D_Y(G(x)))] \tag{2.5}$$

$$\mathcal{L}_{\text{adv}}(F, D_X) = \mathbb{E}_{x\sim\mathcal{X}}[\log D_X(x)] + \mathbb{E}_{y\sim\mathcal{Y}}[\log(1 - D_X(F(y)))] \tag{2.6}$$

where $\mathcal{L}_{\text{adv}}(G, D_Y)$ and $\mathcal{L}_{\text{adv}}(F, D_X)$ are the adversarial losses for the generators G and F, and their respective discriminators D_Y and D_X.

The total loss in the CycleGAN framework, combining the adversarial and cycle-consistency losses, can be expressed as:

$$\mathcal{L}_{\text{CycleGAN}}(G, F, D_X, D_Y) = \mathcal{L}_{\text{adv}}(G, D_Y) + \mathcal{L}_{\text{adv}}(F, D_X) + \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}}(G, F) \tag{2.7}$$

Here, λ_{cyc} is a hyperparameter that adjusts the relative importance of the cycle-consistency loss. Training the CycleGAN involves solving the following optimization problem:

$$\min_{G,F} \max_{D_X,D_Y} \mathcal{L}_{\text{CycleGAN}}(G, F, D_X, D_Y) \tag{2.8}$$

In this min-max game, the generators G and F attempt to create convincing images that fool the discriminators D_Y and D_X, while the discriminators are tasked with distinguishing real images from the ones generated by G and F.

To further guide the generators to preserve the content of input images that are already in the target domain, an additional identity loss $\mathcal{L}_{\text{id}}$ is introduced:

$$\mathcal{L}_{\text{id}}(G, F) = \mathbb{E}_{x\sim\mathcal{X}}[\|G(x) - x\|_1] + \mathbb{E}_{y\sim\mathcal{Y}}[\|F(y) - y\|_1] \tag{2.9}$$

This identity loss encourages the generators to output images similar to the input when the input is already from the desired domain, helping to maintain the original characteristics of the images.

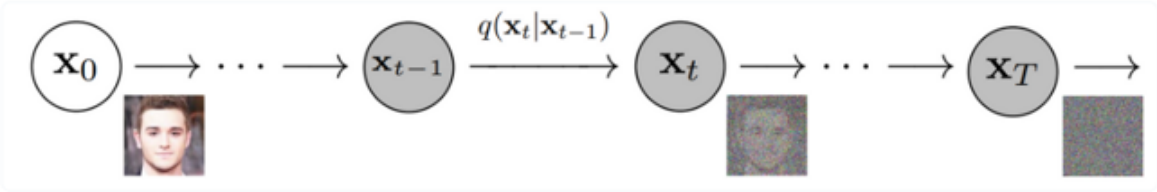


Figure 2.8.: The forward diffusion process: gradual noise addition to transform an image $\mathbf{x}_0$ into $\mathbf{x}_T$ [50].

2.5. Diffusion Models

The advent of diffusion models marks a major breakthrough in the domain of generative modeling, a field focused on learning the underlying data distribution and synthesizing new data points that resemble the original dataset. Early generative methods, such as Variational Autoencoders [46] and Generative Adversarial Networks [25], laid the groundwork for modern generative approaches by leveraging probabilistic modeling and adversarial training, respectively. Despite their success, both VAEs and GANs exhibit limitations, VAEs often produce blurry outputs due to the variational approximation, while GANs are prone to issues such as mode collapse and instability during training. Diffusion models [47, 48] have emerged as a powerful alternative, offering a more stable and interpretable generative process. By modeling the data generation as a gradual denoising process, diffusion models can achieve high-quality sample generation with theoretically grounded stability, making them a robust alternative in modern generative tasks.

The modern framework of diffusion models was formalized by Ho *et al.* (2020) with the introduction of **Denoising Diffusion Probabilistic Models (DDPM)** [48]. DDPM models the generative process as a **Markov chain** that progressively adds Gaussian noise to data in a series of discrete timesteps, forming a **forward diffusion process**. In this process, a data point $\mathbf{x}_0$ is gradually transformed into a highly noisy version $\mathbf{x}_T$ over T steps, according to the following Gaussian transition:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t;\ \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I}) \tag{2.10}$$

where α_t is a variance schedule controlling the amount of noise added at each timestep t. Figure 2.8 illustrates this **forward diffusion process** [48, 49], where noise is progressively introduced to an original image.

The core task of DDPM is to reverse this process by learning a generative model that reconstructs $\mathbf{x}_0$ from $\mathbf{x}_T$ by denoising in small, incremental steps. The **reverse diffusion process** [48, 49] is defined as:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \tag{2.11}$$

where $\mu_\theta(\mathbf{x}_t, t)$ and $\Sigma_\theta(\mathbf{x}_t, t)$ represent the mean and variance of the denoised image predicted by the neural network parameterized by θ. Figure 2.9 shows how the reverse diffusion process works to denoise an image by iteratively reconstructing $\mathbf{x}_0$ from $\mathbf{x}_T$.

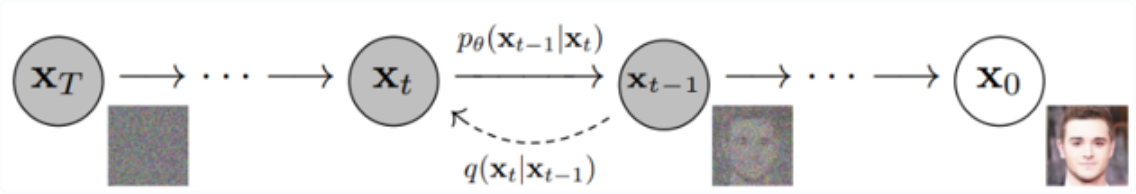


Figure 2.9.: The reverse diffusion process: reconstructing the original image $\mathbf{x}_0$ by denoising a noisy image $\mathbf{x}_T$ [48].

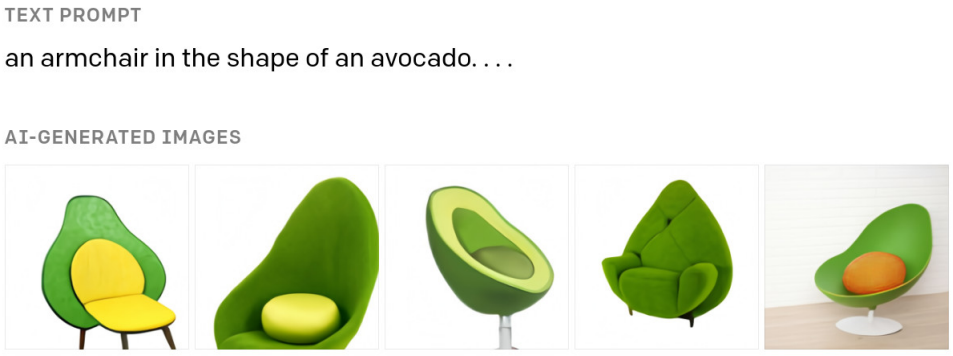


Figure 2.10.: The famous Avocado chair (Prompt: ”an armchair in the shape of an avocado”) [55]

This formulation allows DDPM to produce high-quality samples by reversing the noise addition process in a principled and stable manner. The introduction of DDPM has reignited interest in generative modeling, positioning diffusion models as a compelling alternative to Variational Autoencoders and Generative Adversarial Networks [51].

In early 2021, OpenAI introduced **DALL-E** [52] and **CLIP** [53], two landmark models that significantly advanced multimodal generative modeling. While DALL-E does not directly use diffusion processes, it laid the groundwork for subsequent models by employing a **Vision Transformer (ViT)** [54] to represent images as sequences of tokens. This tokenization allowed DALL-E to generate new images from a set of tokens, with textual tokens at the start of the sequence to condition the model on specific prompts. DALL-E's capacity to generate novel and imaginative images, such as the famous *avocado armchair* shown in Figure 2.10, showcased the potential of this approach. The model's outputs were further refined using **CLIP** (Contrastive Language-Image Pre-training), which pruned image candidates by evaluating their semantic alignment with the input text, thus functioning as an auto-cherry-picking mechanism.

In early 2021, Nichol & Dhariwal introduced an improved variant of Denoising Diffusion Probabilistic Models, focusing on enhancing the model's efficiency and image quality by refining the noise scheduling process [56]. Their work introduced a better timestep scheduler that adjusted the density of steps, making them denser toward the final stages of the image generation process. This change allowed the model to recover finer details in the generated images, reducing the number of steps required for high-quality outputs.

Later that year, in September 2021, Google introduced another significant advancement with **"Classifier-Free Diffusion Guidance"**, proposed by Ho & Salimans [57]. This method

eliminated the need for external classifiers during the diffusion process by training the model with and without auxiliary information (such as class labels or text descriptions). During generation, the model computes the difference in outputs from both conditions, guiding the process and providing greater flexibility and control over the generated images. Classifier-free guidance became a crucial tool in improving diffusion-based models, further pushing the boundaries of generative modeling.

On December 20, 2021, two groundbreaking papers were released, marking a significant milestone in the development of diffusion models. The first, titled **"GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models"**, was introduced by Nichol et al. [58]. This model from OpenAI presented a novel approach to generating and editing images using textual prompts. GLIDE employed a text-guided diffusion model capable of producing photorealistic images, significantly advancing image generation technology. Moreover, it incorporated pre-trained 'filtered' models to mitigate harmful outputs, making it a practical tool for creative applications.

Simultaneously, Rombach *et al.* from the Ommer Lab in Europe introduced **"High-Resolution Image Synthesis with Latent Diffusion Models,"** which brought a major innovation by compressing images into a latent space before applying diffusion processes [59]. By operating in a lower-dimensional latent space, Latent Diffusion Models (LDMs) greatly reduced the computational resources required while maintaining high-resolution image quality. This efficiency made LDMs more scalable and accessible for a wide range of applications, including high-resolution synthesis tasks. Both models contributed to the rapid evolution of diffusion-based image generation, setting new benchmarks for realism and computational efficiency.

As of September 15, 2022, the landscape of generative modeling was significantly shaped by three prominent models: **DALL-E 2**, developed by OpenAI, built upon the original DALL-E framework, incorporating advancements in diffusion techniques to generate highly detailed and imaginative images from textual descriptions. Trained on approximately 650 million image-text pairs, DALL-E 2 can combine concepts, attributes, and styles in novel ways, producing complex and imaginative outputs [60]. The model uses a hierarchical approach to first generate images at a resolution of 64×64, which are then progressively upscaled to 256×256 and 1024×1024 through Super Resolution techniques, enhancing both detail and visual fidelity [61]. While the DALL-E 2 code remains proprietary, limiting large-scale image generation, access to the system was available via OpenAI's portal to manually generate and save images.

Google's Imagen emerged as a direct competitor, emphasizing the quality and fidelity of generated images. This model utilized a combination of diffusion processes and large-scale language models to create photorealistic images from text prompts, focusing on both semantic alignment and visual realism [62]. Imagen's architecture incorporated advanced conditioning mechanisms, enabling it to effectively capture the intricacies of text descriptions and translate them into high-quality visual outputs.

MidJourney is a generative model developed by an independent research lab of the same name [63]. It synthesizes images from text prompts and is known for its surrealistic style, making it particularly popular among artists. The model is currently in open beta and is accessible via a **Discord server**, where users input prompts and receive generated images.

Lastly, **Stable Diffusion** [64], developed by Stability AI in 2022, revolutionized accessibil-

ity in generative modeling. Unlike its counterparts, Stable Diffusion operates within a latent space, allowing for efficient and high-resolution image synthesis with significantly lower computational requirements [59]. Stable Diffusion is primarily used to generate detailed images conditioned on text descriptions but is also versatile enough for tasks such as inpainting, outpainting, and image translation. The model is trained on 512×512 images from a subset of the LAION-5B database and uses a frozen CLIP ViT-L/14 text encoder to condition on text prompts [65]. With its 860M UNet and 123M text encoder parameters, Stable Diffusion is relatively lightweight, running efficiently on GPUs with at least 10GB VRAM. Together, these models exemplified the rapid advancements in diffusion-based generative modeling and set new standards for image generation quality and accessibility.

2.6. Anomaly Detection and Autoencoders

Anomaly detection involves identifying patterns or data points that significantly deviate from expected behavior. In industrial environments, this process is critical for maintaining operational efficiency, ensuring safety, and avoiding costly downtime [66]. Anomalies can arise from equipment malfunctions, unexpected environmental conditions, or errors in the production process. Early detection of these irregularities enables prompt intervention, reducing the risk of severe disruptions and financial losses [67].

With modern industrial systems generating vast amounts of data through sensors, machinery, and monitoring tools, traditional statistical methods often prove inadequate for detecting anomalies in complex, high-dimensional datasets [66]. Consequently, machine learning techniques have become increasingly important. Among these techniques, **autoencoders** have gained prominence due to their effectiveness in reconstructing input data and identifying deviations [68].

Autoencoders [70] are neural networks used primarily for unsupervised learning, particularly in the tasks of dimensionality reduction, feature learning, and anomaly detection. An autoencoder consists of two main components: the encoder and the decoder. The encoder compresses the input data into a latent representation, while the decoder attempts to reconstruct the original data from this compressed representation [71], as shown in Figure 2.11.

Mathematically, the **encoding process** can be represented as:

$$z = g(x; \phi) \tag{2.12}$$

where x denotes the input data (such as an image or a feature vector), g is the encoder function that maps x to the latent representation z, and ϕ represents the parameters (weights and biases) of the encoder network. The latent representation z captures the essential features of the input in a lower-dimensional space.

The **decoding process** can be written as:

$$x' = f(z; \theta) \tag{2.13}$$

where x' is the reconstructed output, f is the decoder function that maps z back to the input space, and θ are the parameters of the decoder network. The goal is to ensure that the reconstructed output x' closely matches the original input x.

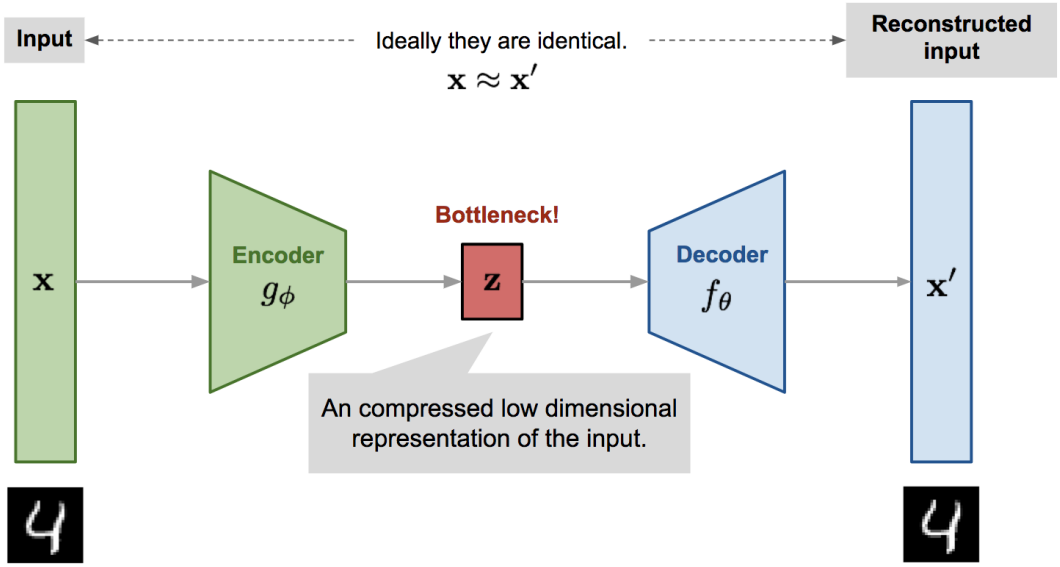


Figure 2.11.: Architecture of an autoencoder, consisting of an encoder that compresses the input data into a lower-dimensional latent representation and a decoder that reconstructs the input from this latent space [69]

To measure the quality of the reconstruction, a loss function is employed. Typically, the Mean Squared Error (MSE) is used:

$$L(x, x') = \|x - x'\|^2 \tag{2.14}$$

where $L(x, x')$ represents the loss between the original input x and the reconstructed output x'. The training process aims to minimize this loss by optimizing the parameters ϕ and θ, ensuring that the autoencoder can accurately reconstruct the input data from the latent representation.

3. Materials and Methods

3.1. Datasets

The dataset used in this thesis was collected from **Balluff Matrix Vision GmbH**, including high-resolution images (1280x960) of Nuts, Candles, Balluff Network Interfaces (BNIs), and Printed Circuit Boards (PCBs). It includes both OK (good) and NOK (not okay, defective) images, captured under two setups: optimal conditions and those with intentional camera-induced variations, such as shadow effects, plexiglass reflections, scattered sunlight, and random object placements. These variations were crucial for simulating real-world challenges, providing the necessary data for generating synthetic images that replicate these conditions. NOK images are utilized solely during the evaluation of the anomaly detector, while OK real images are essential for training models to generate synthetic images.

3.1.1. Nuts Dataset

The Nuts dataset includes both OK and NOK images, captured under different setups: Optimal and Setup with bad influences. The total number of images in each category are presented in Table 3.1.

Optimal Setup:

The images captured under the optimal setup as shown in Figure 3.1 were taken with consistent lighting and environmental conditions, ensuring high image quality. This setup provides a clear representation of both OK and NOK Nuts, free from any bad camera influences.

Setup with bad influences:

In this setup, images were captured under conditions that introduced various camera-induced variations, simulating real-world challenges. The following influences were documented:

1. **Shadow Effect:** Shadows occur due to uneven lighting conditions, leading to the formation of dark areas on the Nuts. This influence as shown in Figure 3.2 can pose a challenge for the anomaly detection system, as the detector might misclassify images with shadows as anomalies if not adequately trained on such variations. The goal is

Table 3.1.: Nuts Dataset Summary

Condition	Optimal Setup	Bad Influences
OK Images	152	305
NOK Images	48	93

(a) OK nut under optimal setup

(b) NOK nut under optimal setup

Figure 3.1.: Examples of Nuts captured under optimal setup conditions. a) OK Nut, exhibiting clear features essential for quality assessment. b) NOK Nut, showing characteristics that indicate defects. Both images are essential for training and evaluating the anomaly detection system, as they provide a clear baseline of acceptable and unacceptable product qualities.

(a) OK nut affected by shadow effect

(b) NOK nut affected by shadow effect

Figure 3.2.: Examples of Nuts affected by shadow effects. a) OK Nut and b) NOK Nut.

to train the anomaly detector to correctly classify genuine NOK images as anomalies while recognizing that OK images affected by shadows should not be labeled as such.

2. **Random Object Placement:** Random object placement refers to the misalignment of the Nuts within the image frame, where the items are not centered or are positioned at varying angles. This can affect the anomaly detection system's evaluation, as the model might mistakenly identify the irregular positioning as a defect.

   To illustrate this influence, the following images in Figure 3.3 depict both OK and NOK Nuts affected by random object placement.

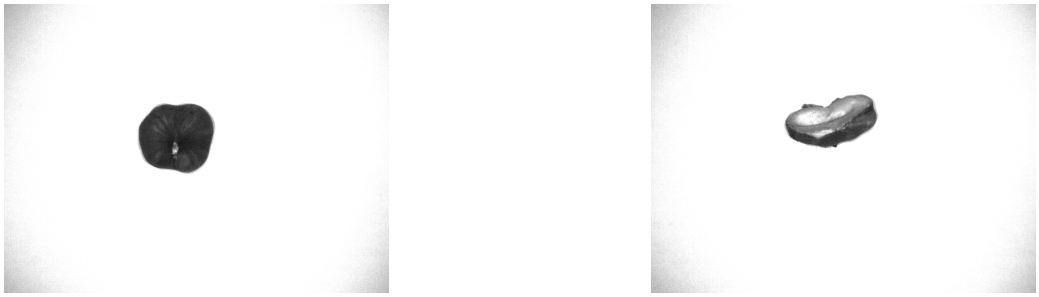
3. **High Gain Factor:** A high gain factor is applied to increase the sensitivity of the camera sensor, particularly in low-light conditions. While this adjustment can enhance the visibility of the nuts, it also amplifies any inherent noise in the image, resulting in a grainy appearance. This noise can obscure critical features, complicating the evaluation process for the anomaly detection system. If the model is not adequately trained to recognize and differentiate between noise and actual defects, it may mistakenly classify images with high gain as anomalies, leading to inaccurate assessments.

   Figure 3.4 shows both OK and NOK nuts captured with a high gain factor.

4. **Plexiglas Effect:** The presence of plexiglass can introduce reflections and distortions in the images, affecting the visibility of the nuts. These distortions can complicate

(a) OK nut with random object placement (b) NOK nut with random object placement

Figure 3.3.: Examples of Nuts affected by random object placement. a) OK Nut, and b) NOK Nut.



(a) OK nut with high gain factor (b) NOK nut with high gain factor

Figure 3.4.: Examples of nuts affected by a high gain factor. a) OK Nut, and b) NOK Nut.

the evaluation process for the anomaly detection system, as it may misinterpret the reflections as defects in the product.

Figure 3.5 showcases examples of both OK and NOK Nuts affected by the plexiglass effect.

3.1.2. Candles Dataset

The Candles dataset consists of both OK and NOK images, captured under two different setups: optimal conditions and setups with various camera-induced influences. The details of the Candles dataset are summarized in Table 3.2.

(a) OK nut affected by plexiglass effect (b) NOK nut affected by plexiglass effect

Figure 3.5.: Examples of Nuts affected by the plexiglass effect. a) OK Nut, and b) NOK Nut.

Table 3.2.: Candles Dataset Summary

Condition	Optimal Setup	Bad Influences
OK Images	145	134
NOK Images	51	86

(a) OK candle under optimal setup

(b) NOK candle under optimal setup

Figure 3.6.: Examples of Candles captured under optimal setup conditions. a) OK Candle, showing smooth and uniform features. b) NOK Candle, showing defects like dents or scratches.

Optimal Setup:

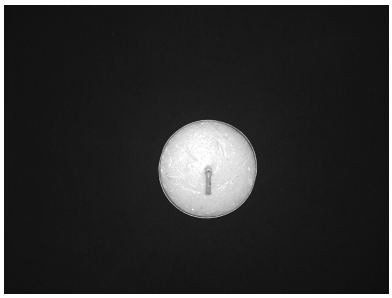
The images captured under the optimal setup were taken with consistent lighting and environmental conditions, ensuring high image quality. This setup provides a clear representation of both OK and NOK Candles. OK Candles as shown in Figure 3.6 a) typically exhibit a round shape, with no dents in the holder or scratches on the candle surface, or absence of the Candle or thread in the holder. In contrast, NOK Candles display these irregularities as shown in Figure 3.6 b) .

Setup with bad influences:

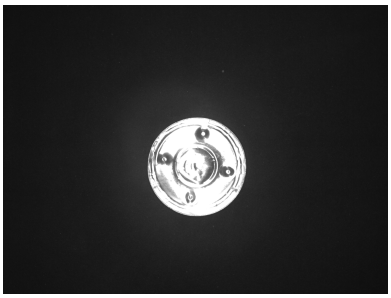
In this setup, images of Candles were taken under conditions that led to various camera artifacts, imitating the challenges encountered in practical scenarios. The following influences were observed:

1. **Scattered Sunlight:** This influence arises when sunlight is diffused or reflected by surrounding objects, leading to uneven illumination on the Candles as shown in Figure 3.7. The resulting highlights and shadows can obscure important features, potentially complicating the visual evaluation of the Candles quality.
2. **Random Object Placement:** Random object placement occurs when Candles are not centered within the image frame or are positioned at various angles. This misalignment can lead to confusion in the evaluation process, as the irregular positioning might be interpreted as a defect rather than a mere arrangement issue.

   Figure 3.8 presents examples of both OK and NOK Candles demonstrating random object placement
3. **Camera Elevation (Down):** This setup involves positioning the camera 1 cm down, leading to larger and brighter images of the Candles.

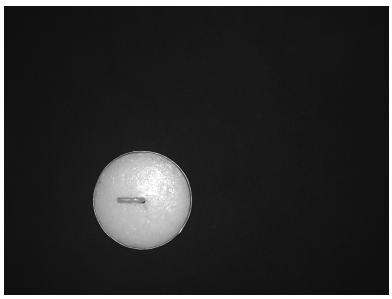


(a) OK candle affected by scattered sunlight

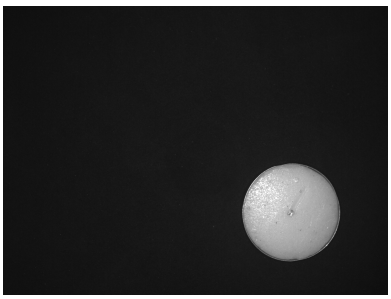


(b) NOK candle affected by scattered sunlight

Figure 3.7.: Examples of Candles affected by scattered sunlight. a) OK Candle and b) NOK Candle, which shows an absence of the candle in the holder

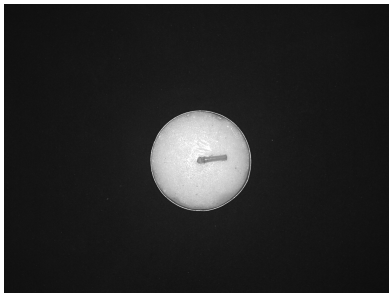


(a) OK candle with random object placement

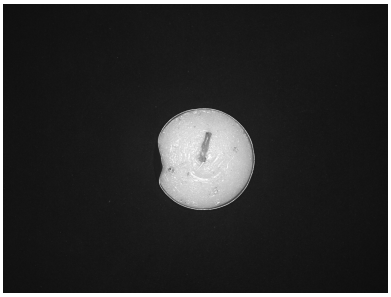


(b) NOK candle with random object placement

Figure 3.8.: Examples of Candles affected by random object placement. a) OK Candle and b) NOK Candle, which shows the absence of the thread on the candle.

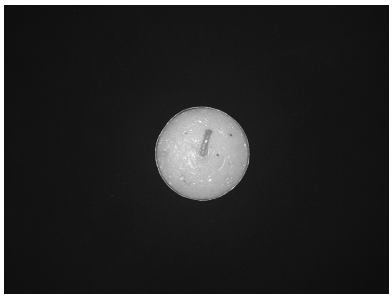


(a) OK candle with camera elevation down

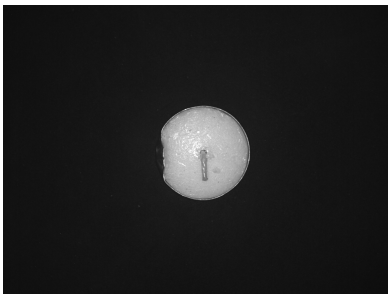


(b) NOK candle with camera elevation down

Figure 3.9.: Examples of Candles affected by camera elevation down. a) OK Candle and b) NOK Candle with dents on the candle holder.



(a) OK candle with camera elevation up



(b) NOK candle with camera elevation up

Figure 3.10.: Examples of Candles affected by camera elevation up. a) OK Candle and b) NOK Candle.

The following images in Figure 3.9 depict both OK and NOK Candles affected by this camera elevation.

4. **Camera Elevation (Up):** This setup involves positioning the camera 1 cm up, leading to smaller and darker images of the Candles.

   Figure 3.10 depict both OK and NOK Candles affected by this camera elevation.

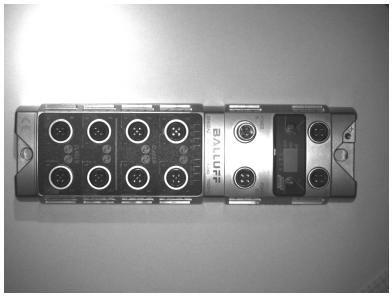
3.1.3. BNI Dataset

BNIs are essential devices in industrial automation, facilitating communication between sensors, actuators, and control systems. The BNI dataset includes both OK and NOK images captured under optimal conditions and various camera-induced influences. The number of images collected in BNI dataset are summarized in Table 3.3.

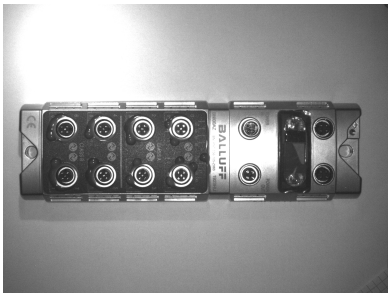
Optimal Setup:

Table 3.3.: BNI Dataset Summary

Condition	Optimal Setup	Bad Influences
OK Images	44	73
NOK Images	24	48

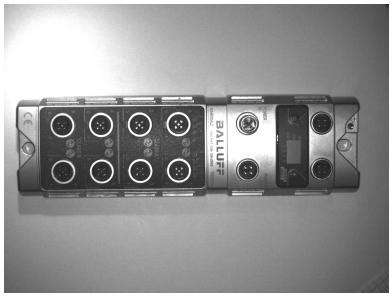


(a) OK BNI under optimal setup

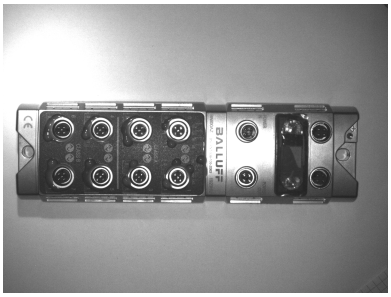


(b) NOK BNI under optimal setup

Figure 3.11.: Examples of BNIs captured under optimal setup conditions. a) OK BNI, displaying intact labels and no surface defects. b) NOK BNI, showing solidified epoxy resin on the surface.



(a) OK BNI with left inclination



(b) NOK BNI with left inclination

Figure 3.12.: Examples of BNIs affected by left inclination. a) OK BNI and b) NOK BNI with solidified epoxy resin on the surface.

Images taken in the optimal setup featured uniform lighting and controlled environmental conditions, resulting in high-quality visuals as shown in Figure 3.11. OK images display BNIs in good condition, with intact labels and defect-free surfaces, providing a clear representation of expected quality. Conversely, NOK images may lack complete labeling or exhibit irregularities, including instances where epoxy resin has solidified on the BNI surface due to leaks during the potting process.

Setup with bad influences:

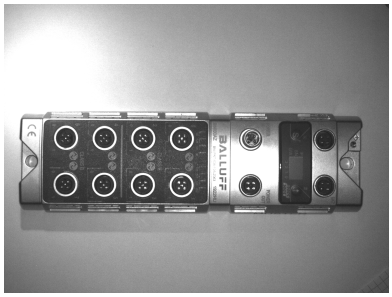
In this setup, images of BNIs were taken under conditions that introduced several camera-related distortions, replicating potential challenges in practical industrial settings. The observed influences were as follows:

1. **Inclination (Left):** In this setup, the BNI device was inclined to the left, altering its orientation in the image frame. This shift in perspective can distort the appearance of labels and surfaces, complicating defect detection.

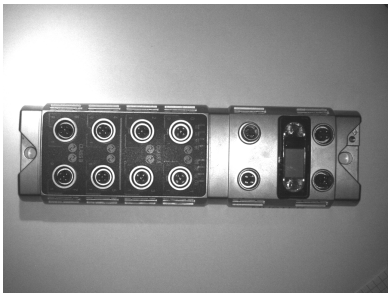
   The impact of left inclination is shown in Figure 3.12, where both OK and NOK BNIs are depicted with this orientation shift.

2. **Inclination (Right):** Similarly, the BNI device was inclined to the right, resulting in a different angle of distortion. This tilt can affect how lighting interacts with the surface, potentially hiding defects or causing glare.

   Figure 3.13 presents examples of BNIs subjected to right inclination, illustrating both

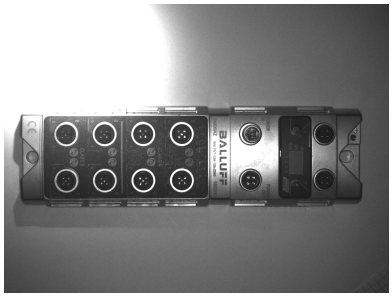


(a) OK BNI with right inclination

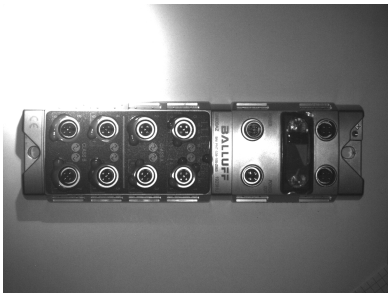


(b) NOK BNI with right inclination

Figure 3.13.: Examples of BNIs affected by right inclination. a) OK BNI and b) NOK BNI with missing labels.



(a) OK BNI under varied lighting



(b) NOK BNI under varied lighting

Figure 3.14.: Examples of BNIs affected by varied lighting. a) OK BNI and b) NOK BNI showing traces of solidified epoxy resin.

OK and NOK instances affected by this tilt

3. **Lighting:** Variations in lighting conditions were introduced, leading to shadows or overexposed areas on the BNIs. These lighting changes replicate real-world scenarios where inconsistent lighting can obscure defects or create reflections.

   The effects of lighting variations on the BNIs are demonstrated in Figure 3.14, displaying both OK and NOK examples influenced by changes in illumination.

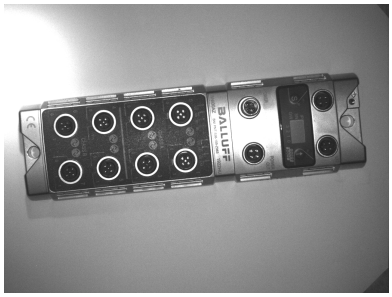
4. **Random Object Placement:** BNIs were positioned randomly within the image frame or at unusual angles, leading to inconsistency in their placement. This can confuse anomaly detection models that rely on uniform object positioning to identify defects.

   Figure 3.15 shows how random object placement affects the BNIs, highlighting both OK and NOK images impacted by this irregular positioning.

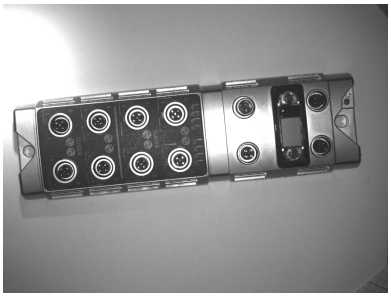
3.1.4. PCB Dataset

PCBs are crucial components in electronic devices, providing the framework for mounting and connecting electronic components. The PCB dataset includes both OK and NOK images captured under two different setups: optimal conditions and conditions influenced by additional lighting. The details of the dataset size are summarized in Table 3.4.

Figure 3.16 illustrates the PCB images: a) and b) presents the OK and NOK PCBs under optimal conditions, showcasing the expected quality and visible defects, respectively. c) and



(a) OK BNI with random placement



(b) NOK BNI with random placement

Figure 3.15.: Examples of BNIs affected by random placement. a) OK BNI and b) NOK BNI where some labels are absent.

Table 3.4.: Printed Circuit Boards (PCB) Dataset Summary

Condition	Optimal Setup	Bad Influences
OK Images	54	20
NOK Images	20	14

d) displays the OK and NOK PCBs under the setup with added lighting, which may affect the clarity and assessment of the images.

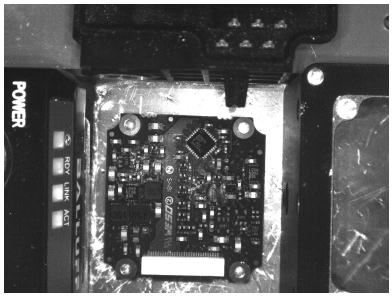
3.2. Methods

3.2.1. Model Selection

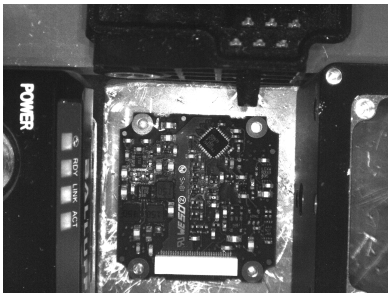
To improve the performance of the anomaly detection system, it is essential to train the model on a dataset that included various camera-induced influences such as shadow effects, lighting variations, plexiglas reflections, and random placement of objects. While real data under both optimal and influenced conditions were collected, the need arose to synthetically generate additional images that captured these artifacts. This was necessary to ensure that the anomaly detector could generalize effectively to the variations in the production environment.

Given the nature of this task, an image-to-image translation approach was selected as the most suitable method for generating these synthetic artifacts. While Pix2Pix [39] is a popular model for image translation tasks, it relies on paired datasets, which were unavailable in this case. Therefore, **CycleGAN**, introduced by Jun-Yan Zhu *et al.* (2017) [41], was chosen for its ability to perform unpaired image translation. A detailed explanation of CycleGAN is provided in Chapter 3.2.2.

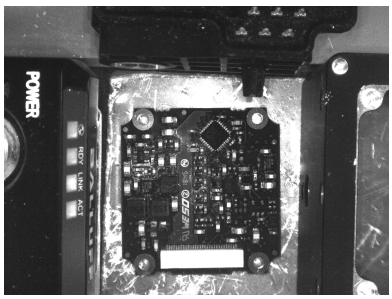
Additionally, **Stable Diffusion** was used to enhance the generation of synthetic images. Stable Diffusion, a more recent deep learning model based on diffusion processes, was employed for its ability to generate highly realistic images with a broader range of variations in image quality. This model allowed for the refinement of synthetic images by controlling the level of noise and generating realistic variations under influenced conditions. The methodology and technical details of Stable Diffusion are covered in Chapter 3.2.3.



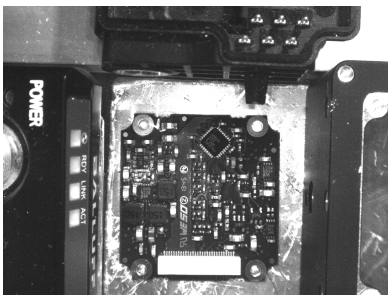
(a) OK PCB under optimal conditions



(b) NOK PCB under optimal conditions



(c) OK PCB under bad influence setup



(d) NOK PCB under bad influence setup

Figure 3.16.: Examples of PCBs captured under different setups: a) OK PCB under optimal conditions, b) NOK PCB under optimal conditions, c) OK PCB under bad influence setup with additional lighting, and d) NOK PCB under bad influence setup. The added lighting may impact the visual assessment of the PCB quality.

3.2.2. CycleGAN Method

The CycleGAN architecture consists of two generator models and two discriminator models, following a similar architecture as described in the original paper [41], facilitating the translation of images between two distinct domains: Domain-A and Domain-B. This subsection details the overall workflow, including data loading and preprocessing, data augmentation, model architecture, and training procedure.

Data Loading and Preprocessing

For the CycleGAN model implementation, the datasets chosen were the Nuts and Candles datasets, as they provide a sufficiently large number of images compared to the BNI and PCB datasets. Only the OK images from these datasets were utilized for training the CycleGAN model, while the NOK images will be employed later in the evaluation phase using the anomaly detection model.

The Nuts Dataset consists of 457 OK images, comprising:

- **Domain A (Optimal)**: 152 images
- **Domain B (with bad influences)**: 305 images

The Candles Dataset includes 279 OK images, with:

- **Domain A (Optimal)**: 145 images
- **Domain B (with bad influences)**: 134 images

(a) (b)

Figure 3.17.: (a) The original nut image and (b) the augmented nut image with a horizontal flip applied.

Images were originally in Bitmap (BMP) format with a resolution of 1280x960. The generator and discriminator models used in this work are particularly large, with the generator containing approximately ~35 million parameters and the discriminator consisting of around ~2.8 million parameters. To handle such computational demands efficiently, the images were resized to 256x256 pixels and converted to Portable Network Graphics (PNG) format, ensuring compatibility with the input requirements of the models while optimizing memory usage and processing speed. The experiments were primarily conducted on the Nuts dataset due to its well-defined influences, allowing for more precise image translation.

Data Augmentation

The CycleGAN model utilized for this task contained approximately 38 million parameters, which posed the risk of overfitting, given the relatively small dataset size [72]. To mitigate this, standard data augmentation techniques were applied using `ImageDataGenerator` [73]. The following augmentations were implemented:

1. Horizontal flipping
2. Vertical flipping
3. Rotation in the range $[-10°, +10°]$
4. Width shift in the range $[0, 0.1]$
5. Height shift in the range $[0, 0.1]$

Figure 3.17 shows an example of an original nut image alongside its augmented version, where a horizontal flip has been applied. These transformations were selected to ensure that they would not adversely affect the translation task. However, despite these augmentation strategies, the performance remained unaffected. It is crucial to note that the transformations only modify the geometry of the images without impacting the underlying information, which could limit the model's ability to learn effective translations.

Model Architecture

1. **Generator Models**:
   - **Generator-A**: This generator is responsible for creating images in Domain-A(Optimal). It takes images from Domain-B as input, performing image translation conditioned on these images. The architecture follows an encoder-decoder framework, starting with a convolutional layer that downscales the input. This is followed by a series of **nine residual blocks** for 256×256 input images, consisting two convolutional layers with 3×3 filters. Each layer uses Instance Normalization to standardize the inputs, helping the model generalize better to different

domains. The stride is set to 1×1, ensuring that the spatial dimensions of the input are preserved. The input to each residual block is added back to the output, creating a skip connection that allows the network to retain low-level features while learning more complex ones. Importantly, the second convolutional layer does not use a ReLU activation, following the official CycleGAN paper [41] implementation for better image quality. The decoder then upsamples the features through transposed convolutional layers to produce the final output image. Detailed generator architecture is provided in Appendix A.1.

- **Generator-B**: Conversely, Generator-B generates images in Domain-B(with bad influences) and takes images from Domain-A as input. It uses the same architectural principles as Generator-A, ensuring a consistent translation process.

The image translation processes can be illustrated as follows:

$$\text{Domain-B} \xrightarrow{\text{Generator-A}} \text{Domain-A}$$
$$\text{Domain-A} \xrightarrow{\text{Generator-B}} \text{Domain-B}$$

2. **Discriminator Models**: Each generator is paired with a corresponding discriminator:
   - **Discriminator-A**: This model evaluates real images from Domain-A alongside the images generated by Generator-A, predicting whether each image is real or fake. It employs a PatchGAN approach [39], processing 70×70 image patches with several convolutional layers, Instance Normalization, and LeakyReLU activations. The final layer outputs a probability map indicating the likelihood of each patch being real. An in-depth description of the generator architecture can be found in Appendix A.2.
   - **Discriminator-B**: Similarly, this discriminator assesses real images from Domain-B and those produced by Generator-B. It shares the same network architecture as Discriminator-A, ensuring consistency in evaluations.

   The interactions can be represented as:

$$\text{Domain-A} \xrightarrow{\text{Discriminator-A}} \text{[Real/Fake]}$$
$$\text{Domain-B} \xrightarrow{\text{Generator-A}} \text{Discriminator-A} \rightarrow \text{[Real/Fake]}$$

$$\text{Domain-B} \xrightarrow{\text{Discriminator-B}} \text{[Real/Fake]}$$
$$\text{Domain-A} \xrightarrow{\text{Generator-B}} \text{Discriminator-B} \rightarrow \text{[Real/Fake]}$$

3. **Composite Model**: The composite model of CycleGAN leverages multiple loss components to enable effective image translation between two unpaired domains. The main loss components include:

- **Adversarial Loss:** This loss is central to the adversarial training process. The generators are tasked with creating images that can fool the discriminators into classifying them as real. Simultaneously, the discriminators aim to distinguish between real and generated images. The adversarial loss is responsible for improving the generator's ability to create realistic images and the discriminator's ability to identify fakes. It uses the L_2 distance between the model's output and the target (real or fake).

  Adversarial Training Illustration:

$$\text{Domain-A} \xrightarrow{\text{Generator-B}} \text{Domain-B} \xrightarrow{\text{Discriminator-B}} \text{[real/fake]}$$
$$\text{Domain-B} \xrightarrow{\text{Generator-A}} \text{Domain-A} \xrightarrow{\text{Discriminator-A}} \text{[real/fake]}$$

- **Cycle Consistency Loss:** Cycle consistency ensures that if an image is translated from one domain to another and then back to the original domain, it should closely resemble the original image. This helps preserve the structure and content of the input images during translation. The cycle consistency loss is calculated as the L_1 distance between the original and reconstructed images.

  Cycle $A \rightarrow B \rightarrow A$:

$$\text{Domain-A} \xrightarrow{\text{Generator-B}} \text{Domain-B} \xrightarrow{\text{Generator-A}} \text{Domain-A}$$

  Cycle $B \rightarrow A \rightarrow B$:

$$\text{Domain-B} \xrightarrow{\text{Generator-A}} \text{Domain-A} \xrightarrow{\text{Generator-B}} \text{Domain-B}$$

- **Identity Mapping Loss (Optional):** Identity mapping is employed to preserve domain-specific properties. When an image from a domain is passed through its corresponding generator, the output should remain unchanged. This is particularly useful for tasks that require maintaining the color profiles of images. The identity loss is calculated as the L_1 distance between the input and output image.

  Identity Mapping for Domain-A:

$$\text{Domain-A} \xrightarrow{\text{Generator-A}} \text{Domain-A}$$

  Identity Mapping for Domain-B:

$$\text{Domain-B} \xrightarrow{\text{Generator-B}} \text{Domain-B}$$

These losses work together to ensure that the generators produce realistic images, preserve the structure of the original input, and maintain consistency across domains.

Table 3.5.: Training Hyperparameters

Hyperparameter	Value	Description
Epochs	100	Total number of training epochs
Batch size	1	Number of images processed per step
Learning rate(Generator)	0.0002	Learning rate for the generator
Learning rate(Discriminator)	0.0002	Learning rate for the discriminators
Momentum (β_1)	0.5	Momentum term for the Adam optimizer
Cycle-consistency loss	10× adversarial loss	Weight assigned to the cycle-consistency loss
Identity loss	5× cycle-consistency loss	Weight assigned to the identity loss
Image pool size	50	Number of fake images stored in the image pool
Training steps per epoch	$\frac{N_{\text{images}}}{B}$	Number of training steps per epoch based on dataset size

Training Procedure

The CycleGAN model is trained to perform image-to-image translation from Domain A (optimal setup) to Domain B (setup with bad influences). The training process minimizes a combined loss function, which includes the adversarial loss, cycle-consistency loss, and identity loss. The generator $G_{\text{A}\to\text{B}}$ is optimized through the composite model, while the discriminators D_{A} and D_{B} distinguish between real and generated images.

The training process runs for $N_{\text{epoch}} = 100$ epochs, with a batch size $B = 1$. The number of training steps per epoch is determined by the total number of images in the dataset, i.e., $N_{\text{steps}} = \frac{N_{\text{images}}}{B}$, where N_{images} is the size of the Domain A dataset. For each step, real images from Domain A and Domain B are sampled, and fake images are generated by the generators. Discriminators are trained on both real and fake images to distinguish them, while the generator $G_{\text{A}\to\text{B}}$) is trained to fool the discriminator.

An image pool is maintained for the discriminators to stabilize the training, where a set of 50 fake images is used to probabilistically update the discriminator. Plots of generated images are visualized during training to monitor the generator's performance, and models are saved periodically. After training, the model is selected based on iterations where the generated images appear visually appealing, allowing for the generation of new images from the training dataset.

The following table summarizes the hyperparameters used in the training procedure:

The goal is to achieve stable adversarial training where the generator produces realistic images of Domain B and the discriminators cannot easily distinguish between real and gener-

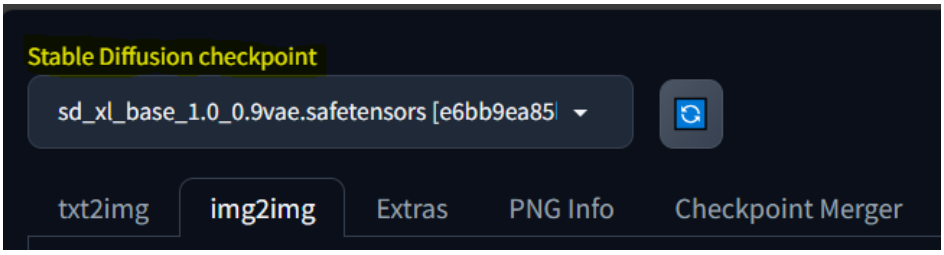


Figure 3.18.: Selecting Model Checkpoint

ated images.

3.2.3. Stable Diffusion WebUI

Stable Diffusion WebUI, particularly the `AUTOMATIC1111` version [74], offers an easy-to-use interface for generating images using Stable Diffusion models. It supports a range of image manipulation tasks, including text-to-image, image-to-image, inpainting, and more, making it suitable for various creative and technical purposes. For this thesis, the Image-to-Image (`img2img`) functionality was used to generate synthetic images using real data. The requirements necessary for setting up the Stable Diffusion model are outlined in Appendix A.3.

Using the img2img Tab in Stable Diffusion WebUI

To generate synthetic images, begin by uploading a real image, for example an image of a Nut or Candle into the `img2img` tab. Once uploaded, select the appropriate checkpoint model, such as `sd_xl_base_1.0_0.9vae.safetensors` as shown in Figure 3.18, ensuring compatibility with the model and task.

Key Settings and Configuration

- **Sampling Method (M_{sam}):** Determines the strategy for generating the image. *DPM++ 2M Karras* was selected for its ability to decrease step sizes towards the end, enhancing image quality [75].
- **Sampling Steps (S_{steps}):** This controls the number of steps for transforming noise into an image. Higher step counts increase detail but also processing time. For this process, $S_{\text{steps}} = 20$ were chosen, providing a balance between quality and efficiency.
- **Image Dimensions (D_{img}):** The width and height of the generated image are crucial for maintaining quality and consistency. A resolution of $D_{\text{img}} = 512 \times 512$ was used, which is typical for Stable Diffusion models trained on this scale.
- **CFG Scale (s_{cfg}):** The CFG Scale indicates how strongly the generated image conforms to the input prompt, lower values allow for greater creativity [76]. A CFG scale of $s_{\text{cfg}} = 7$ was used to balance creative freedom with image fidelity.
- **Denoising Strength (s_{den}):** Defines how much change should be applied to the original image [76]. At $s_{\text{den}} = 0$, no changes occur, while at $s_{\text{den}} = 1$, the image completely transforms. A value of $s_{\text{den}} = 0.2$ was used to introduce some controlled variation while retaining key features of the original image.

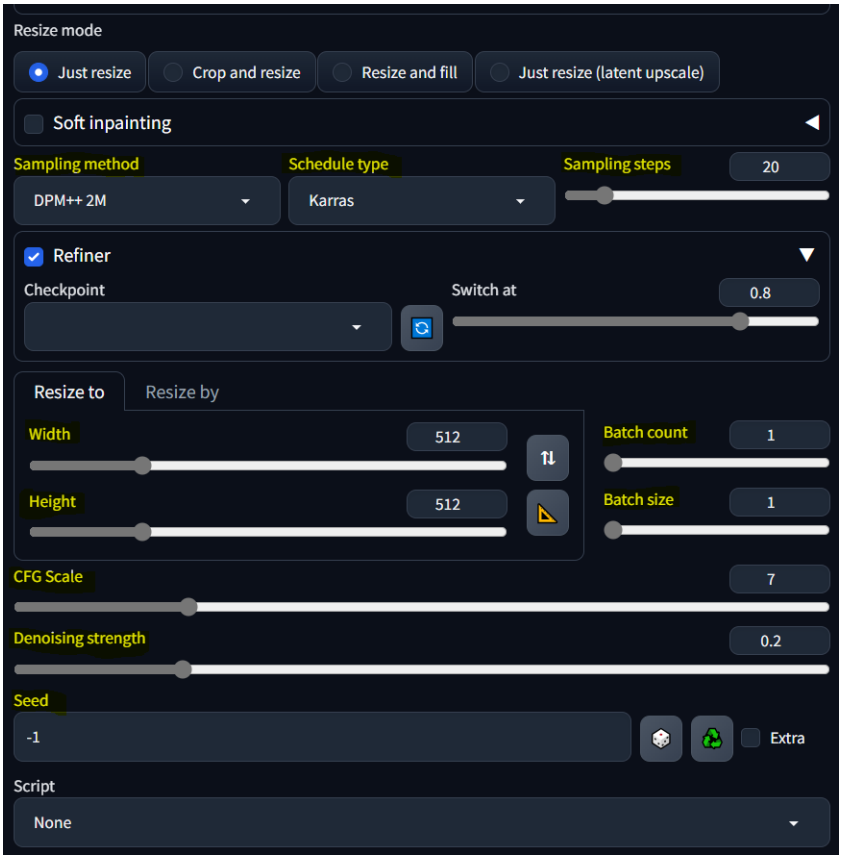


Figure 3.19.: Configuration settings in the Stable Diffusion WebUI for img2img

- **Seed (s_{seed}):** This controls the randomness of the generation process. By setting the seed to $s_{\text{seed}} = -1$, the system generates random images. Setting a specific value allows for consistent image reproduction [76].
- **Batch Count (B_{count}):** Specifies how many images are generated in one run. A count of $B_{\text{count}} = 10$ ensures the generation of multiple variations for further evaluation.
- **Batch Size (B_{size}):** Determines how many images are processed simultaneously during generation. A batch size of $B_{\text{size}} = 1$ was selected to maintain GPU efficiency while generating multiple images sequentially.

Figure 3.19 shows the Stable Diffusion WebUI interface with the img2img tab and configuration settings for image generation.

3.2.4. Evaluation Metrics

In image processing and analysis, assessing the quality of generated images is vital for understanding the performance of different algorithms and models. To achieve this, a variety of metrics are employed to quantify image quality, focusing on aspects such as similarity, distortion, and overall perceptual fidelity. Metrics like Oriented FAST and Rotated BRIEF (ORB), Structural Similarity Index (SSIM), Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and perceptual loss using VGG16 facilitate comparisons between original and synthetic images, as well as the similarity between different generated images. This aids in

optimizing techniques and ensuring that the generated results align with established quality standards. The following sections will explain these metrics in detail.

Oriented FAST and Rotated BRIEF

ORB by OpenCV is a feature detector and descriptor that efficiently identifies and describes keypoints in images, as developed by Ethan Rublee *et al.* in their 2011 paper [77], **"ORB: An efficient alternative to SIFT or SURF"**, making it particularly useful for image matching and object recognition tasks. The process begins with ORB detecting keypoints using the FAST [78] algorithm, followed by generating binary descriptors through the BRIEF [79] technique.

When comparing two images, ORB employs the Hamming distance [80, 81] metric to assess similarity by calculating the number of differing bits between the binary descriptors of matched features. A lower Hamming distance d_{H} indicates a closer match, which is essential for determining the degree of similarity between the images.

To enhance matching efficiency, ORB can be combined with a brute force matcher, which exhaustively compares each feature descriptor from one image with those from another to identify the best matches based on the lowest Hamming distance. An optimal Hamming distance threshold is crucial for improving matching accuracy; typically, values around $d_{\mathrm{H}} = 20$ to 30 are considered ideal, as they indicate strong matches while minimizing the risk of false positives.

To calculate the overall ORB score, the number of good matches (i.e., matches with a Hamming distance below the threshold) is divided by the total number of matched descriptors. This score provides a quantitative measure of how similar the two images are overall, reflecting the effectiveness of the feature matching process.

Structural Similarity Index

SSIM [82, 83] is a perceptual metric used to assess the quality of images by quantifying the degradation caused by processing tasks such as compression and transmission. Unlike traditional metrics that primarily rely on pixel-wise comparisons, SSIM evaluates changes in structural information, which reflects the way humans perceive visual information [84]. The index incorporates three key components: luminance, contrast, and structural similarity.

- **Luminance**: This component measures the brightness of the images. It calculates the average intensity of each image and compares them to determine how well the luminance levels align.
- **Contrast**: This aspect evaluates the contrast between the images, assessing how the variations in pixel intensity contribute to the overall perception of quality.
- **Structural Similarity**: This component examines the relationship between the pixels in both images, focusing on patterns of pixel intensities rather than individual pixel values.

The SSIM index produces a value between -1 and 1, where SSIM $= 1$ indicates perfect similarity between two images, and SSIM $= -1$ indicates significant differences. Due to its sensitivity to perceptual characteristics, SSIM is widely employed in various applications, including image compression, quality assessment, and computer vision tasks, as it correlates well with human visual perception.

Mean Squared Error

MSE is a widely used metric to quantify the difference between the original and generated images. It calculates the average of the squares of the differences between corresponding pixel values [84]. The formula for MSE is given by:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \left(I_{\text{original}}(i) - I_{\text{generated}}(i) \right)^2 \tag{3.1}$$

where N is the total number of pixels, $I_{\text{original}}(i)$ is the pixel value at position i in the original image, and $I_{\text{generated}}(i)$ is the pixel value at the same position in the generated image.

A lower MSE value indicates a closer resemblance between the two images, meaning better image quality. Ideally, an MSE = 0 represents perfect similarity between the images. However, MSE can sometimes be misleading as it does not consider perceptual factors, meaning that images with similar MSE values may not necessarily appear similar to human observers.

Peak Signal-to-Noise Ratio

PSNR [84] is another commonly used metric for assessing image quality, especially in the context of lossy compression. PSNR is derived from MSE and provides a measure of the maximum possible power of a signal (the original image) relative to the power of the noise (the error introduced by compression). The formula for PSNR is given by:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{I_{\text{max}}^2}{\text{MSE}} \right) \tag{3.2}$$

where I_{max} is the maximum possible pixel value of the image (typically 255 for 8-bit images).

PSNR is expressed in decibels (dB), and a higher PSNR value indicates better image quality, as it signifies a lower level of noise. An optimum PSNR value is generally considered to be PSNR = 30 dB or higher for good quality images, with values above PSNR = 40 dB indicating very high quality. While PSNR is easy to compute and widely used, it also has limitations, as it does not align perfectly with human visual perception, and higher PSNR values do not always guarantee higher visual quality.

Perceptual Loss using VGG16

Perceptual loss [85,86] measures the differences between two images based on high-level features extracted by a pre-trained deep convolutional neural network, specifically VGG16 [87]. Developed by Oxford's Visual Geometry Group, VGG16 consists of 16 layers, where the early layers focus on extracting low-level features (such as edges and textures), while the deeper layers capture high-level features (like shapes and object parts).

In the context of calculating perceptual loss between two images, the features from both the real and generated images are extracted using VGG16. The process involves loading the pre-trained model and preprocessing the images to match VGG16's input requirements. Once the features are extracted, the perceptual loss is computed by comparing these feature representations, providing a measure of similarity that aligns more closely with human visual perception. An optimal perceptual loss value is typically low, indicating a closer match between the images, which is essential for effective image quality assessment.

Comparing Two Sets of Images Using VGG16 Feature Extraction and a Classifier

This method evaluates the similarity between two sets of images: real images and those generated by a model (Stable Diffusion). By employing the pre-trained network VGG16 as a feature extractor [88], this technique captures essential semantic information while focusing on high-level features instead of pixel-wise differences.

Process Overview:

1. **Feature Extraction**: Real and generated images are preprocessed and fed into the VGG16 model to extract high-dimensional feature vectors that represent the content of each image. The feature maps produced by VGG16 are then flattened into 1D arrays to facilitate further analysis.
2. **Feature Combination**: The extracted features from both sets are combined, with labels assigned (1 for real images and 0 for generated images) to facilitate classification.
3. **Classifier Training**: A machine learning classifier, such as Logistic Regression [89] or Random Forest [90], is trained on the combined feature set. The classifier attempts to find a hyperplane that effectively separates the high-dimensional space into two distinct regions: one representing real images and the other representing generated images. The dataset is divided into training, validation, and test sets to ensure comprehensive evaluation.
4. **Model Evaluation**: The classifier's performance is measured using accuracy and ROC AUC metrics on the validation and test datasets. Additionally, confusion matrices are employed for performance visualization, providing insights into true positive, true negative, false positive, and false negative rates.
5. **Dimensionality Reduction and Visualization**: To further analyze the feature vectors, Principal Component Analysis (PCA) [91] is utilized for dimensionality reduction. This technique transforms the high-dimensional feature space into a lower-dimensional representation while preserving as much variance as possible. The reduced features allow for effective visualization of the distribution and clustering of images based on their labels, aiding in the interpretation of the classifier's performance and the inherent similarities or differences between the two sets of images.

3.2.5. Anomaly Detection on Nuts Dataset

This section discusses the evaluation of an anomaly detection model using the Nuts dataset, which consists of images categorized into OK (good) and NOK (not okay) classes captured under various conditions. As summarized in Table 3.1, the dataset includes a variety of images captured under both optimal conditions and those affected by different influences. The evaluation involves three distinct approaches to training the model, each with a different composition of training datasets. The model's performance will be assessed using a consistent test dataset, with key metrics analyzed to evaluate the effectiveness of each approach.

This test dataset as given in Table 3.6 comprises:

- All 48 NOK images captured under optimal conditions.
- All 93 NOK images with bad influences.

Table 3.6.: Test Dataset Composition

Type	Number of Images
OK Optimal	31
OK Bad Influences	60
NOK Optimal	48
NOK Bad Influences	93
Total	232

Table 3.7.: Training Dataset Composition for First Approach

Category	Number of Images
OK Optimal (Real)	121

- 31 OK images captured under optimal conditions (30% of the total 152 OK images).
- 60 OK images with bad influences, covering each of the four variations: shadow, random placement, high gain factor, and plexiglas.

The following descriptions detail the training datasets utilized for each approach and how the model's performance was evaluated.

First Approach: Training with Only Good Optimal Images

In the first approach, only the 121 OK images captured under optimal conditions were used as the training dataset. No images with bad influences were included during training. The objective of this approach was to observe the anomaly detector's performance when trained on ideal images and tested with both optimal and bad influence images in the test dataset.

Images used for training in the first approach is given in Table 3.7.

Second Approach: Adding Real Bad Influence Images

In the second approach, the remaining 245 OK images with bad influences (from the total 305) were added to the 121 OK optimal images from the first approach. This aims to evaluate if the inclusion of real bad influence images improves the performance of the anomaly detector.

Images used for training in the second approach are provided in Table 3.8.

Third Approach: Incorporating Synthetic Bad Influence Images

In the third approach, instead of using real bad influence images, 245 synthetically generated images (using Stable Diffusion) were added to the 121 OK optimal real images. This

Table 3.8.: Training Dataset Composition for Second Approach

Category	Number of Images
OK Optimal (Real)	121
OK Bad Influences (Real)	245

Table 3.9.: Training Dataset Composition for Third Approach

Category	Number of Images
OK Optimal (Real)	121
OK Bad Influences (Synthetic)	245

approach tests whether synthetically generated images can achieve similar results to real images in anomaly detection.

Images used for training in the third approach are listed in Table 3.9

Implementation Details

The implemented anomaly detection method utilizes a convolutional autoencoder to reconstruct input images and assess their similarity to the original data. A convolutional autoencoder learns efficient representations through an encoder-decoder architecture, compressing the input into a lower-dimensional latent space and reconstructing it from this representation.

The training dataset consists of images resized to 256×256 pixels and normalized to the range of $[0, 1]$. This dataset is shuffled and split into training (80%) and validation (20%) sets.

The convolutional autoencoder architecture varies slightly among the three approaches: Approach_1 employs a simpler design with fewer layers, featuring two convolutional layers in the encoder with 32 and 16 filters, and two corresponding layers in the decoder with 16 and 32 filters. In contrast, Approach_2 and Approach_3 each add an extra convolutional layer in the encoder with 64 filters and an additional corresponding layer in the decoder also with 64 filters, allowing for the capture of more complex features from the input data and enhancing feature extraction and reconstruction capabilities.

Training occurs over 100 epochs, utilizing mean squared error as the loss function. Both training and validation losses are monitored to ensure effective learning. After training, reconstruction errors for the validation set are computed to evaluate the model's performance in reconstructing normal images versus anomalous ones. A threshold for anomaly detection is established based on the 95^{th} percentile of these reconstruction errors.

Finally, the model's effectiveness is evaluated on separate test datasets containing both normal and anomalous images. Metrics such as confusion matrices, accuracy, precision, recall, F_1-score, and balanced accuracy are computed to provide a comprehensive assessment of all three approaches.

4. Results and Discussions

This chapter presents the results and discussion of experiments conducted using CycleGAN on two datasets: Nuts and Candles. These datasets were selected due to their larger size compared to the PCB and BNI datasets, allowing for more robust evaluation of the CycleGAN model. The results for the Nuts dataset are presented first, followed by the Candles dataset. Throughout the discussion, specific challenges encountered during the training process across both datasets will be highlighted. Next, the results from Stable Diffusion are presented, focusing on the impact of denoising strength on the generated images. The generated images for all four datasets are then presented and discussed in terms of their quality and realism. Subsequently, key evaluation metrics are discussed to assess the image quality and similarity between the real and generated images. Finally, the performance of the anomaly detector was evaluated using the generated images from Stable Diffusion.

4.1. CycleGAN Experiments

4.1.1. Experiments with Nuts Dataset

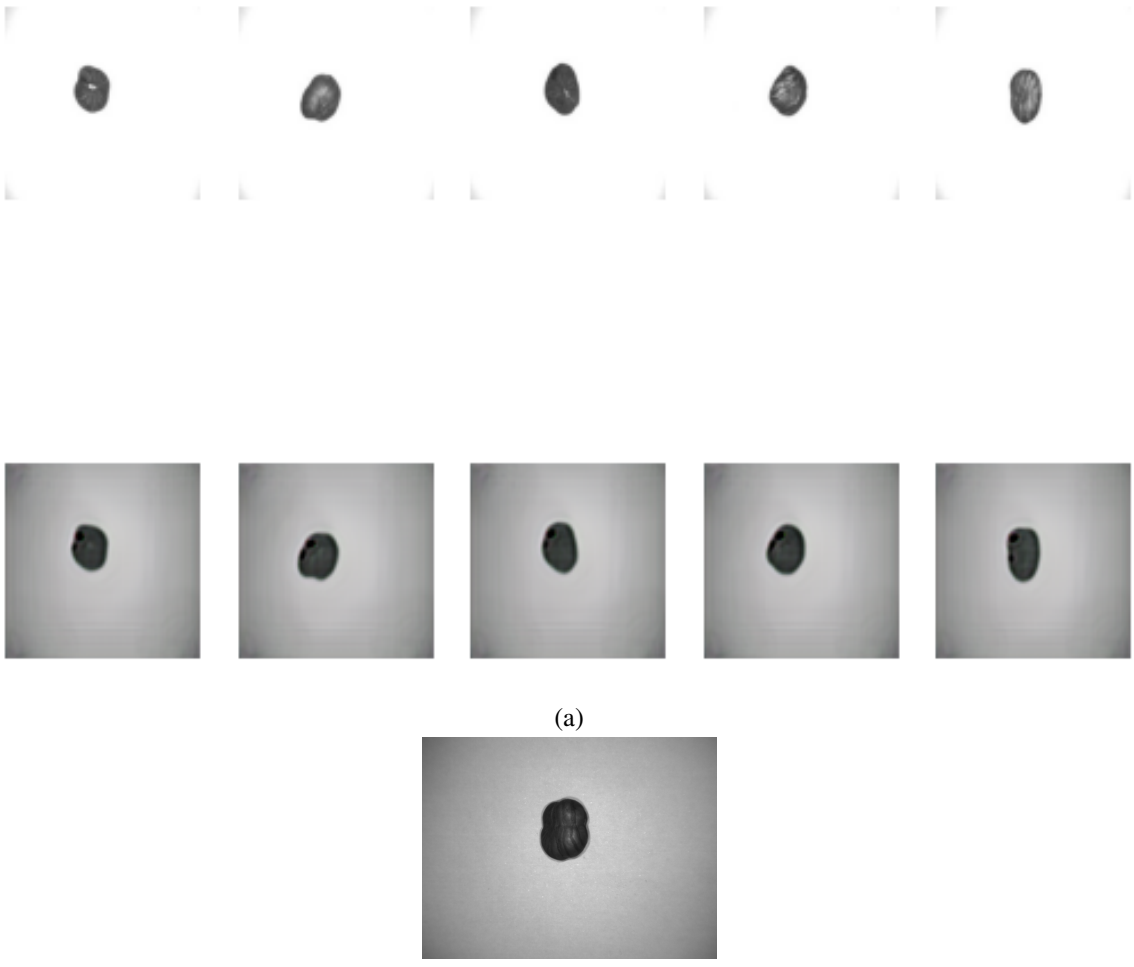
The experiments involving the Nuts dataset aimed to translate images from optimal setups to those influenced by various camera-induced factors. Specifically, this dataset encompasses four distinct types of influences: **Shadow**, **Random Placement**, **High Gain Factor**, and **Plexiglas Effect**, as discussed in Chapter 3.1.1.

Presented below are the results obtained during the training of the CycleGAN model. Certain hyperparameters, as detailed in Table 3.5, were modified to evaluate their impact on the results. Intermediate results were saved after each epoch to facilitate the assessment of model performance throughout the training process, and any changes to hyperparameters for specific results will be noted.

Shadow Effect

Figure 4.1 illustrates the image translation from optimal setup to shadow effect, obtained during CycleGAN training at the 13^{th} epoch. Figure 4.1 a) presents five images in the first row from the optimal setup (Domain A) and their corresponding translated images (Domain B) in the second row that exhibit the shadow effect. Figure 4.1 b) provides a reference image displaying the original shadow effect for comparison.

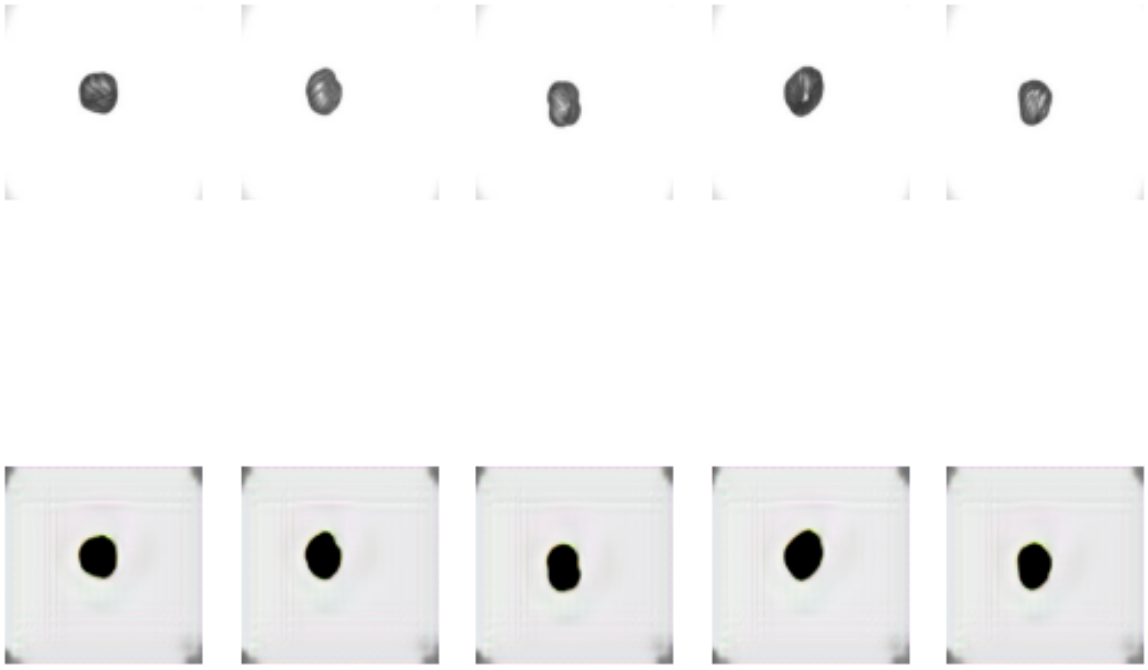
The translated images effectively capture the background and shadow characteristics seen in the reference image, with the Nuts appearing realistic. The hyperparameter settings for this experiment align with those outlined in Table 3.5, indicating that the results were derived under the original configuration.



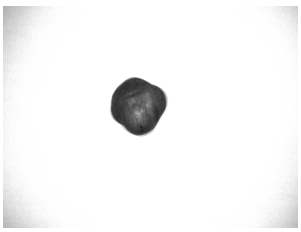
(a)

(b)

Figure 4.1.: Results of CycleGAN image translation showcasing the shadow effect. a) Translated images from optimal setups to images influenced by the shadow effect. b) Reference image of the original shadow effect.



(a)



(b)

Figure 4.2.: Results of CycleGAN image translation highlighting the high gain factor effect. a) Translated images from the optimal setup to images influenced by the high gain factor. b) Reference image demonstrating the high gain factor effect.

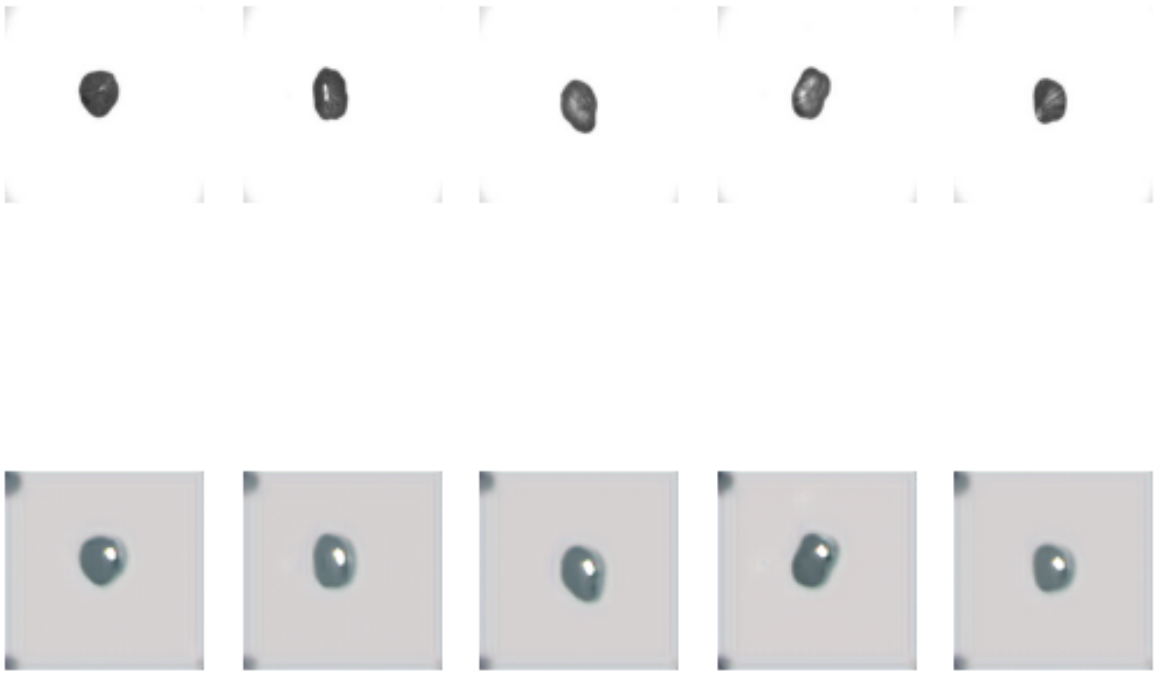
High Gain Factor Effect

Figure 4.2 illustrates the impact of translating images from the optimal setup to those influenced by a high gain factor, captured during CycleGAN training at the 61st epoch. Figure 4.2 a) displays a series of images from the optimal setup, along with the corresponding translated images that exhibit the high gain factor effect. Figure 4.2 b) presents a real image as a reference highlighting the high gain factor effect.

The results show a visible high gain factor effect, yet the realism in nut appearance is lacking, indicating that the translation process may not fully capture the nuances of the high gain factor influence. Again, the original model architecture, training procedure and same hyperparameter values as in Table 3.5 were used in this experiment.

Plexiglas Effect

Figure 4.3 illustrates the results of translating images from the optimal setup to those with Plexiglas effect, obtained during CycleGAN training. Figure 4.3 a) presents the translated images showcasing the Plexiglas effect, while Figure 4.3 b) provides a reference image (real)



(a)

(b)

Figure 4.3.: Results of CycleGAN image translation showcasing the Plexiglas effect. a) Translated images from the optimal setup to images influenced by the Plexiglas effect. b) Reference image illustrating the Plexiglas effect.

that highlights the Plexiglas effect.

For this experiment, the model complexity of both the generator and discriminator was reduced by simplifying their architecture. In the generator model, six ResNet blocks were used instead of the typical nine. Additionally, the learning rate of the discriminator was decreased from 0.0002 to 0.0001 to prevent it from becoming too dominant (loss value becoming zero) over the generator during training.

With the incorporation of these modifications, the results clearly show the Plexiglas effect as seen in the original image in Figure 4.3 b) but lack image quality.

Random Object Placement Effect

One significant limitation of CycleGAN is its inherent inability to manage tasks that require specific spatial arrangements of objects within an image. When CycleGAN processes images from Domain A, it primarily focuses on preserving the core features of the original images while incorporating artifacts from Domain B. Consequently, any modifications made during this translation are constrained by the existing structures within Domain A.

In the context of random object placement effect, the challenge stems from the fact that the training dataset for Domain A does not include any images of Nuts arranged randomly. As a result, it lacks the capability to generate images that accurately depict the random placement of Nuts.

4.1.2. Experiments with Candles Dataset

The Candles dataset was utilized to examine the translation of images from optimal setups to those impacted by different environmental and camera-induced influences like **Scattered Sunlight**, **Random Object Placement**, **Camera Elevation (Up)**, and **Camera Elevation (Down)**, as discussed in Chapter 3.1.2.

However, one of the challenges encountered with this dataset is that the visual differences between the various influences are not as pronounced as those observed in the Nuts dataset. This lack of clear distinction between influences made the analysis of generated images more difficult. As a result, the outputs produced by CycleGAN were harder to evaluate for accuracy and realism. Despite this, intermediate results saved after specific epochs during training revealed some promising translations, although the overall visual quality and differentiation were still not as satisfactory as expected.

The same hyperparameters, as detailed in Table 3.5, were used for training the Candles dataset. The discriminator model's complexity was intentionally reduced to diminish its capacity for distinguishing between real and generated images. Specifically, the original discriminator architecture as in Appendix A.2 was modified by decreasing the number of convolutional filters in the initial layers from 64, 128, and 256 to 32, 64, and 128, respectively and by removing the final layer that utilized 512 filters. This simplification aimed to establish a better balance between the generator and discriminator during adversarial training, enhancing the generator's ability to produce more realistic outputs. Key components such as LeakyReLU activations and Instance Normalization were retained as recommended in [41] to stabilize training and improve convergence. Results were obtained at different epochs, with intermediate outputs saved throughout the training process, allowing for an assessment of the generated images over time.

Scattered Sunlight Effect

Figure 4.4 illustrates how images from optimal setups are translated into those affected by scattered sunlight, with results obtained during CycleGAN training at the 56^{th} epoch.

Camera Elevation (Down)

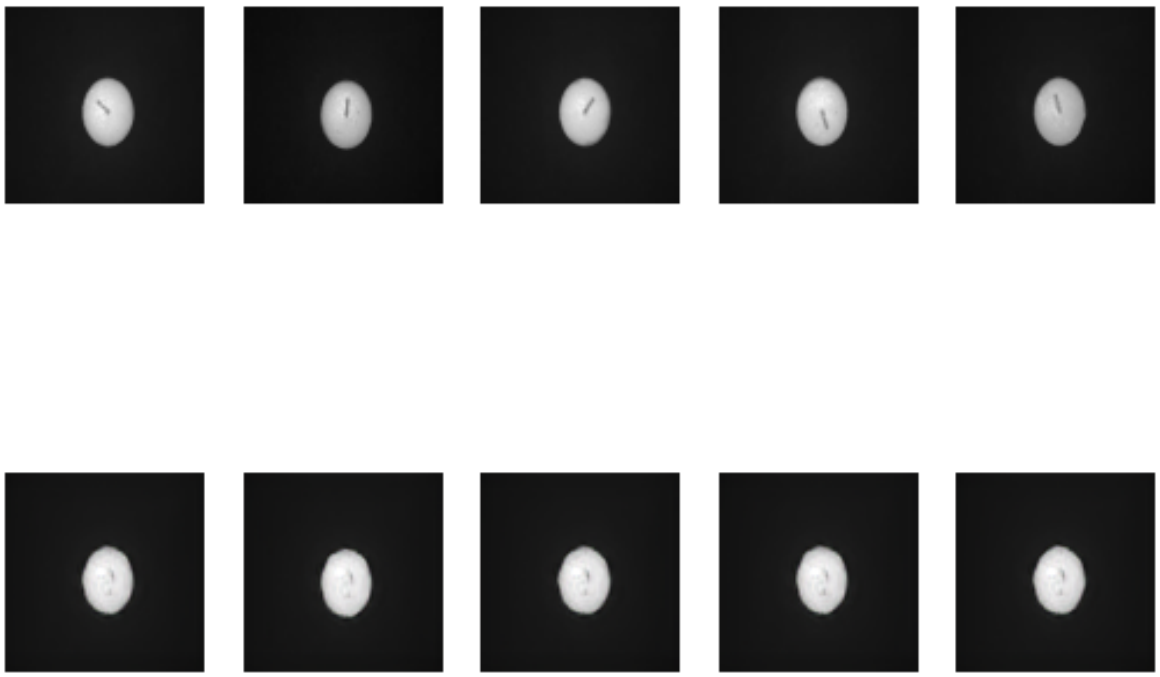
Figure 4.5 focuses on the translation of images from optimal setups to those impacted by a downward camera elevation effect, showcasing results achieved at the 67^{th} epoch.

Camera Elevation (Up)

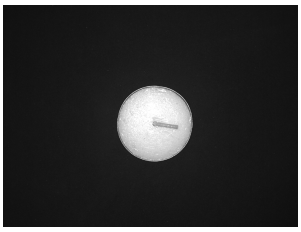
Figure 4.6 highlights the image translation from optimal setups to those influenced by camera elevation (up), obtained at the 84^{th} epoch.

Analysis of Results

The **Scattered Sunlight Effect** demonstrates the presence of bright sunlight; however, the generated images lack realism. In the case of **Camera Elevation (Down)**, the translated images appear slightly brighter, similar to the reference image, but do not exhibit the expected

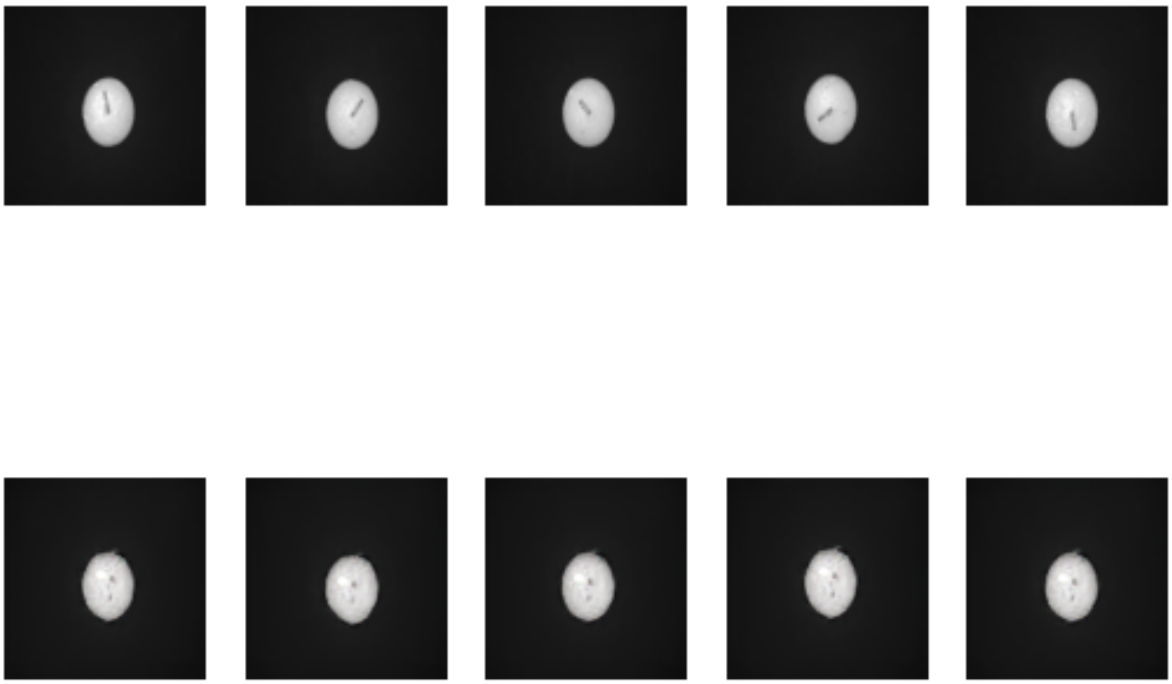


(a)

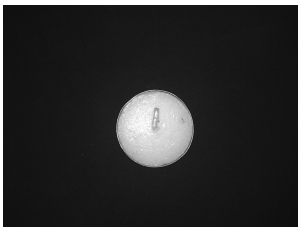


(b)

Figure 4.4.: Results of CycleGAN image translation showcasing the scattered sunlight effect. a) Translated images from the optimal setup to images influenced by scattered sunlight. b) Reference image illustrating the original scattered sunlight effect.

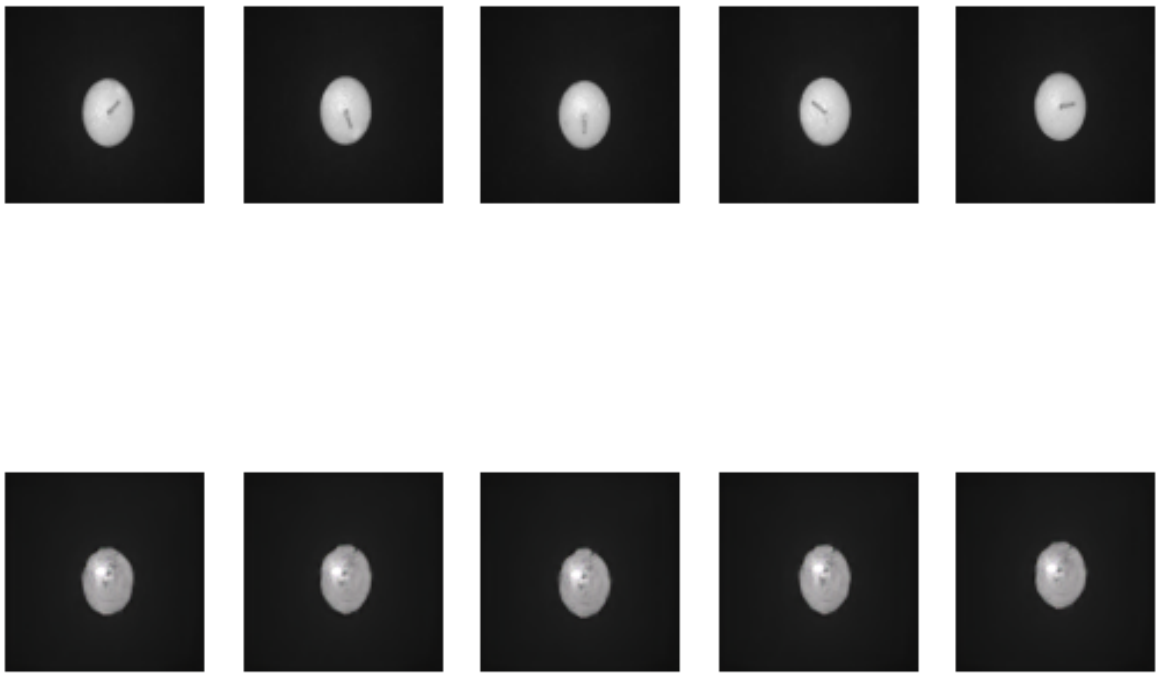


(a)



(b)

Figure 4.5.: Results of CycleGAN image translation showcasing the camera elevation (down) effect. a) Translated images from the optimal setup to images influenced by camera elevation (down). b) Reference image illustrating the original camera elevation (down) effect.



(a)

(b)

Figure 4.6.: Results of CycleGAN image translation showcasing the camera elevation (up) effect. a) Translated images from the optimal setup to images influenced by camera elevation (up). b) Reference image illustrating the original camera elevation (up) effect.

larger dimensions. Conversely, for **Camera Elevation (Up)**, the images appear darker but fail to convey the smaller size anticipated from this effect. As discussed in Chapter 4.1.1, CycleGAN preserves the integrity of images from Domain A while attempting to infuse artifacts from Domain B. This inherent characteristic explains the inability to achieve larger images in the case of Camera Elevation (Down) and smaller images in the case of Camera Elevation (Up). Additionally, the model is unable to generate the effect of **Random Object Placement** for the same reasons. Overall, the results reveal that the generated images across all conditions are not particularly realistic, with candle threads not being generated clearly. Furthermore, the translated images in the Scattered Sunlight Effect, Camera Elevation (Down), and Camera Elevation (Up) show a notable similarity. This lack of diversity in outputs may suggest a potential modal collapse during training, leading to repetitive results across these conditions.

4.1.3. Model Training Analysis

The training process for the CycleGAN model can be evaluated by monitoring key losses over time. Three main loss functions were tracked to assess the performance: generator loss L_{G} for domain A to B translation, and discriminator losses $L_{\text{D,real}}$ for real images and $L_{\text{D,fake}}$ for fake images in Domain B. The loss values showed similar trends while training with Nuts dataset and Candles dataset.

Generator Loss for domain A to B Translation

The generator loss for domain A to B translation, shown in Figure 4.7, which is an average of three components: adversarial loss, identity loss, and cycle consistency loss and it gradually decreases over the course of training, as expected. This trend indicates that the generator is progressively improving its ability to produce realistic images of Domain B from Domain A. However, the training process shows fluctuations, with occasional spikes in the loss. This suggests that while the generator improves on average, there are certain points where the quality of the generated images temporarily drops. These fluctuations could be caused by instability in adversarial training, where the discriminator becomes briefly better at identifying fake images, forcing the generator to adapt. Additionally, the results do not improve in a steady manner, with epochs of high-quality results followed by others where the quality deteriorates. This inconsistency highlights the challenges associated with training GANs [92]. Training was stopped when the loss values began to stabilize, indicating that further improvements in quality were minimal.

Discriminator Losses

The discriminator losses $L_{\text{D,real}}$ and $L_{\text{D,fake}}$ as shown in Figure 4.8, initially decreases, but this is characterized by significant fluctuations rather than a smooth decline. Both loss curves experience abrupt spikes at various training epochs, indicating moments where the discriminator momentarily becomes better at distinguishing between real and generated images. Over time, both losses approach near-zero values, suggesting that the discriminator effectively learns to classify the images. However, despite this stabilization, the presence of sudden spikes demonstrates the ongoing challenges in adversarial training, where the balance between the generator and discriminator must be carefully managed to prevent either model from overpowering the other.

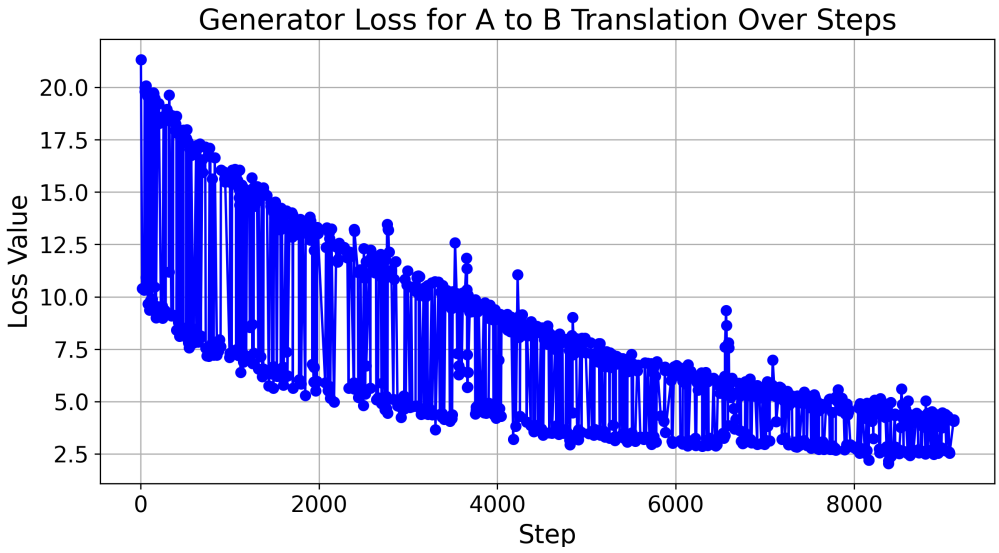


Figure 4.7.: Generator loss for domain A (Optimal) to domain B (Bad influences) translation during training

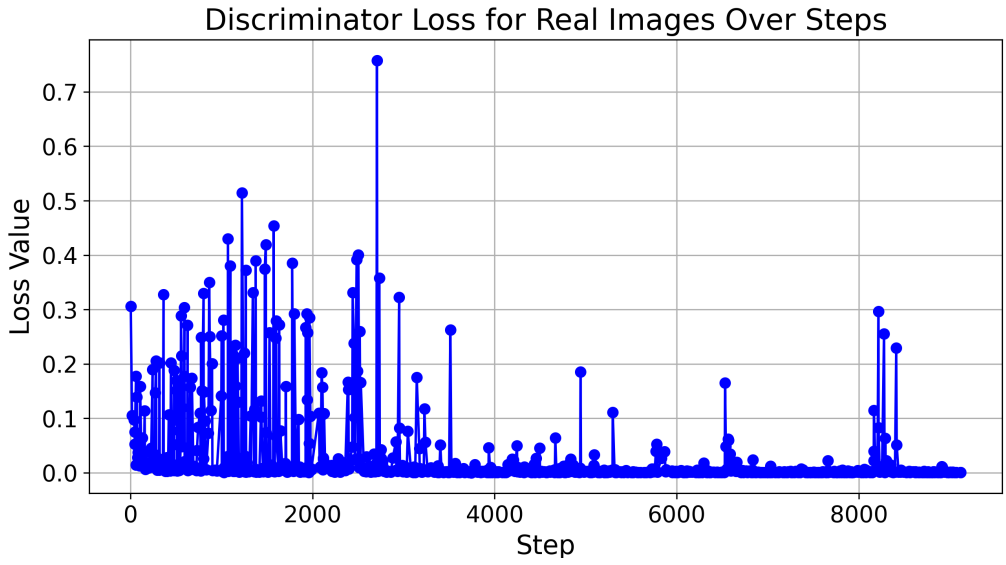
4.1.4. Challenges Faced in CycleGAN Training

While CycleGAN offers a powerful framework for unpaired image-to-image translation, several challenges emerged during training that impacted the quality and effectiveness of the generated images.

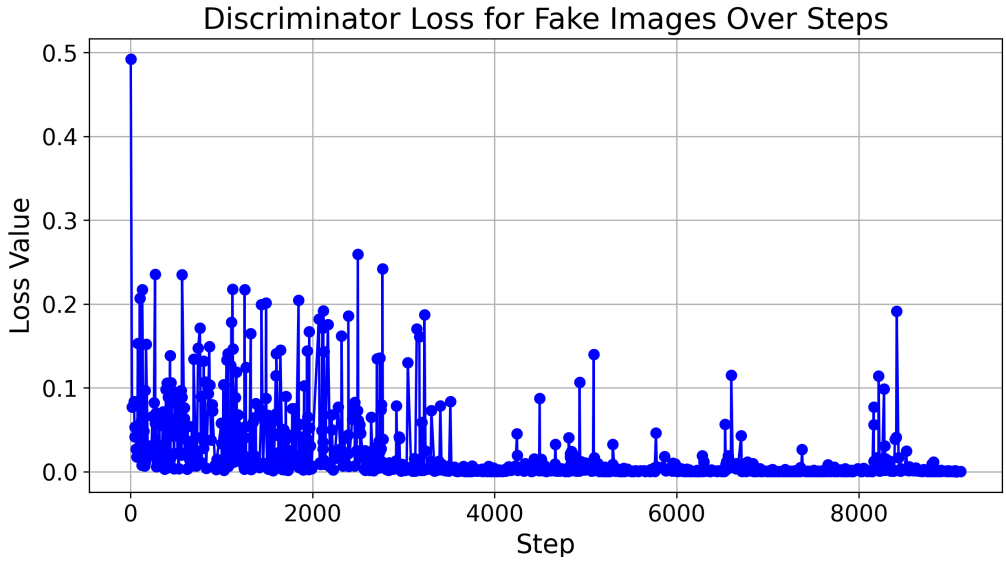
One of the key challenges encountered during the training of CycleGAN was **mode collapse** [93]. This problem, common in adversarial training, occurs when the generator produces a restricted range of outputs, failing to capture the full diversity of the training data. This results in repetitive or similar images regardless of the variations in the input, leading to a significant reduction in the overall output quality. As evidenced by the results, Candle images generated under different conditions, such as the Scattered Sunlight Effect and variations in Camera Elevation, exhibited striking similarities, suggesting that the model experienced mode collapse at various stages of training. This issue can often be attributed to the inherent instability of GAN architectures, wherein the delicate balance between the generator and discriminator is disrupted, causing the generator to latch onto specific modes of the data distribution while neglecting others. Factors such as discriminator overfitting and catastrophic forgetting can exacerbate this issue, ultimately hindering the generator's ability to produce varied and realistic outputs [94].

Another significant challenge arose from the **nature of the data**. Generative models, including CycleGAN, typically require large and diverse datasets to generalize well and produce varied results. However, in this case, the datasets were limited both in size and diversity. The Nuts and Candles datasets, although containing different samples, lacked significant variation in appearance. Many images of Nuts and Candles looked similar in terms of color, shape, and background. Although data augmentation techniques were applied to increase the diversity of the dataset, the inherent similarity between the images meant that these augmentations were not particularly helpful in improving the model's ability to capture meaningful variations.

A further notable challenge was the **instability of the training process**. Even though the



(a)



(b)

Figure 4.8.: Discriminator losses during training: a) loss for real images and b) loss for fake images from domain B

losses showed a gradual decline over time, there were consistent fluctuations and occasional spikes in both generator and discriminator losses. This instability in the training dynamics is a characteristic of GAN models, where the adversarial nature of the training can lead to periods of rapid improvement followed by sudden regressions. In this case, the instability made it difficult to reach a point where the generator and discriminator could both perform optimally. This unstable training environment, combined with the limited and homogeneous data, ultimately hindered the ability of CycleGAN to produce realistic and varied outputs across different conditions.

4.2. Stable Diffusion Experiments

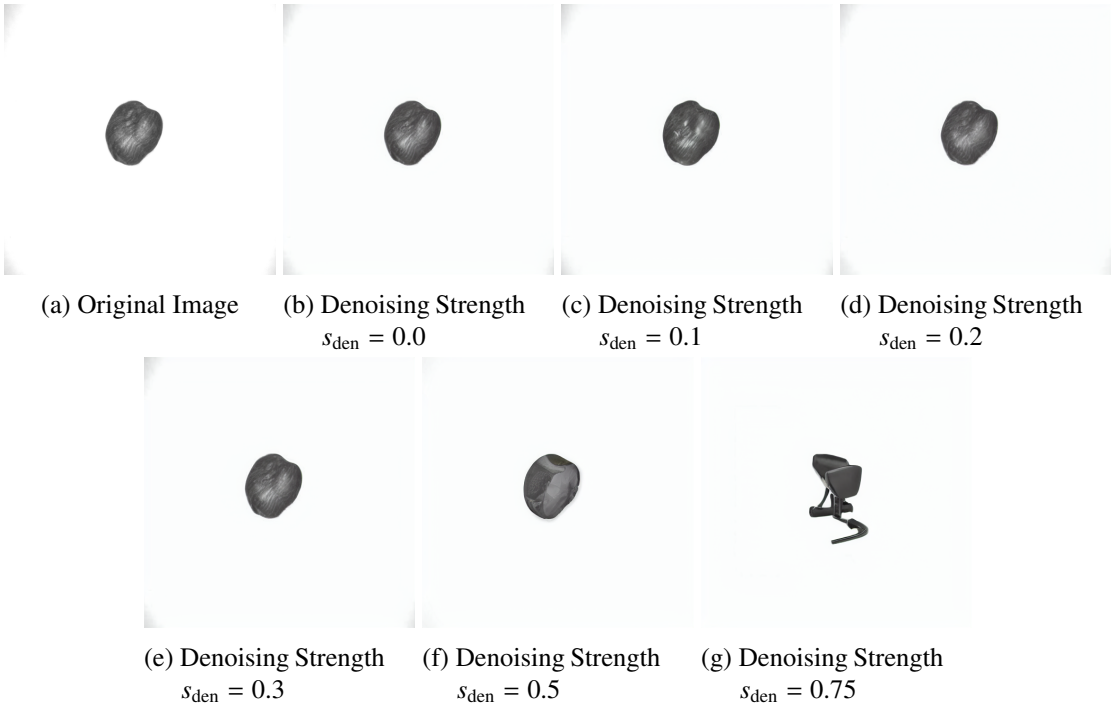
Initially, the goal was to introduce various camera influences into the images using generative models, particularly CycleGAN, for image-to-image translation. Although CycleGAN was capable of generating images with these camera influences, its limitations, including mode collapse and training instability, resulted in images that lacked the necessary quality and realism. As a result, these images could not be reliably used to train the anomaly detector effectively.

Consequently, Stable Diffusion, specifically the `img2img` functionality was used to generate synthetic images. While direct image-to-image translation from optimal images to images with bad influences was not possible with Stable Diffusion, this approach allowed for the generation of synthetic images that incorporated the necessary camera influences. By uploading each real image with bad camera influence from the Nuts dataset and setting a low denoising strength $s_{\text{den}} = 0.2$ and all the other configurations as mentioned in Chapter 3.2.3, it was possible to generate similar images with subtle variations in texture, which were considered more suitable for testing. This approach was applied to all the images from the Domain B dataset of Nuts, creating a set of synthetic images corresponding to each real image. These synthetic images were integrated into the anomaly detection process to test whether they could replicate the performance improvements observed when real images with bad influences were used.

Thus, while the initial objective was to use generative models to introduce various artifacts, Stable Diffusion offered a more practical way to generate image variations. Although real images with bad influences still need to be collected, the advantage of Stable Diffusion is that it requires a limited quantity of these images compared to CycleGAN, which necessitates a larger dataset for good results. This makes the synthetic image generation process more efficient.

4.2.1. Impact of Denoising Strength on Image Generation

In the Stable Diffusion experiments, the influence of varying denoising strengths on the generated images was systematically evaluated. Denoising strength is a crucial parameter that dictates the extent to which the original image is modified during the generation process. By manipulating this parameter, the changes in the characteristics of the output images can be observed.



(a) Original Image (b) Denoising Strength $s_{\text{den}} = 0.0$ (c) Denoising Strength $s_{\text{den}} = 0.1$ (d) Denoising Strength $s_{\text{den}} = 0.2$

(e) Denoising Strength $s_{\text{den}} = 0.3$ (f) Denoising Strength $s_{\text{den}} = 0.5$ (g) Denoising Strength $s_{\text{den}} = 0.75$

Figure 4.9.: Generated images at varying denoising strengths.

For this analysis, a single original image was used as a base, and images were generated with denoising strengths set to 0.0, 0.1, 0.2, 0.3, 0.5, and 0.75. As shown in Figure 4.9, the original image provides a reference point for understanding the alterations made at each denoising level.

At a denoising strength $s_{\text{den}} = 0.0$, the generated image closely mirrors the original, maintaining its features and details. However, as the denoising strength increases, noticeable changes in texture and structure emerge. For instance, at a denoising strength $s_{\text{den}} = 0.1$, the output begins to exhibit slight variations.

At a denoising strength $s_{\text{den}} = 0.5$, the generated image becomes significantly more abstract, with the original image's details becoming increasingly obscured. Finally, at a denoising strength $s_{\text{den}} = 0.75$, the output diverges dramatically from the original, resulting in an image that preserves subtle features of its source while introducing a new, creative interpretation.

For the experiments, a denoising strength $s_{\text{den}} = 0.2$ was selected, as it introduced subtle but meaningful variations to the images while still preserving the key characteristics of the original. This balance allowed for slight texture changes that could simulate real-world variations without drastically altering the content of the images.

4.2.2. Generated Images Using Stable Diffusion

Below are the generated images for each of the four datasets using Stable Diffusion. These images include examples from both optimal setups and those affected by various camera-induced influences, demonstrating how the model introduces subtle variations while maintaining the essential characteristics of the original images.

Figure 4.10 compares the real and generated images for the Nuts dataset across different setups, where each pair consists of a real image on the left and its corresponding generated image on the right.

Figure 4.11 presents a comparison between real and generated images for the Candles dataset, with each pair displaying a real image followed by its corresponding synthetically generated version.

Figure 4.12 illustrates the comparison of real and generated images for the BNI dataset under different setup.

Figure 4.13 shows the comparison of real and generated images for the PCB dataset. First pair shows images from the optimal setup, while the second pair shows images affected by bad camera influences, with real images on the left and generated images on the right.

For all four datasets, only OK (good) images were generated, as the goal of anomaly detection is to add synthetically generated images that include bad camera influences to the training dataset and to check if the model performs similarly when using real images with bad camera influences. There is no need to generate synthetic NOK (anomalous) images, as real NOK images are used solely for testing.

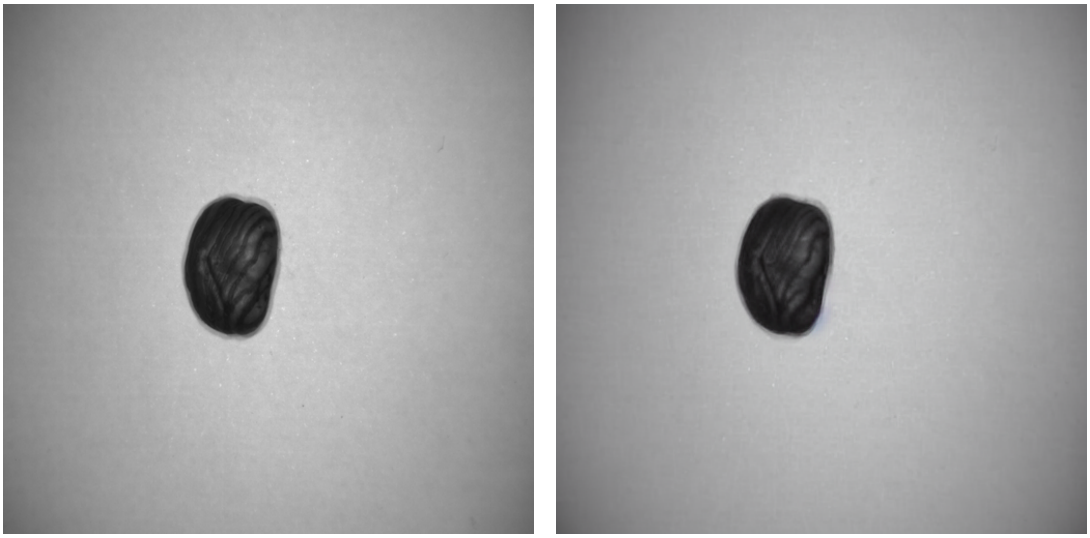
As observed in the generated images, the model perform exceptionally well for the Nuts and Candles datasets. These datasets do not contain intricate details like text or fine structures, which makes it easier for the model to generate high-quality synthetic images with minimal deviations from the originals. The essential features, such as the shapes and textures, are preserved.

In contrast, the BNI and PCB datasets present more challenges due to the presence of fine details, such as text or small components, which are not accurately reproduced in the generated images. For example, in the BNI dataset as in Figure 4.12, the words and specific markings are not clearly reproduced in the synthetic images. Similarly, the PCB dataset as shown in Figure 4.13, which contains detailed features like labels or small circuits, also shows inconsistencies, with the generated images failing to capture these precise elements. However, the overall shape and structure of the objects remain consistent.

4.3. Evaluation Metrics for Assessing Image Quality and Similarity

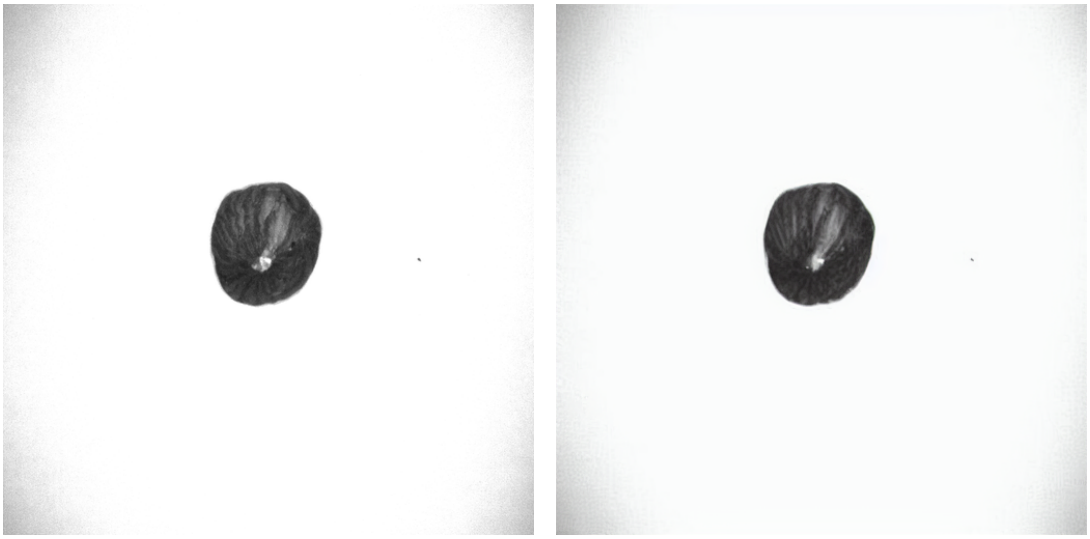
In this evaluation, images generated using CycleGAN are compared to real images to assess the quality and fidelity of the synthetic outputs. Following this, the generated images using Stable diffusion (img2img) from the Nuts, Candles, BNI, and PCB datasets are assessed using various metrics. Comparisons are made between real and generated images to evaluate how accurately the synthetic images replicate the original. Additionally, since multiple synthetic images are generated from a single real image using the Stable Diffusion img2img method, a pair of generated images are also being compared to analyze the differences and assess how much variation exists between the generated outputs.

(a) Optimal setup



(b) Shadow effect

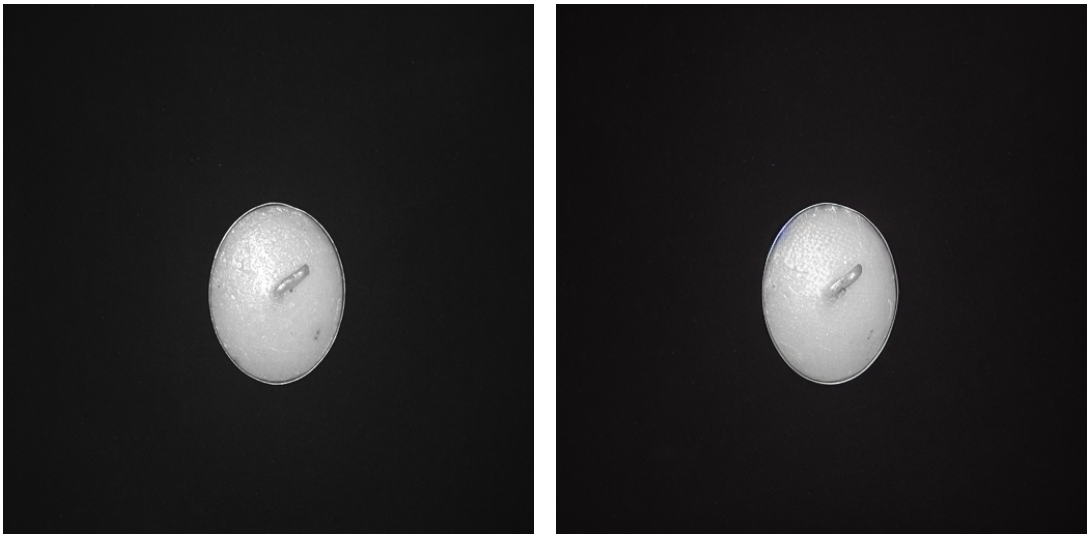
(c) Random placement effect



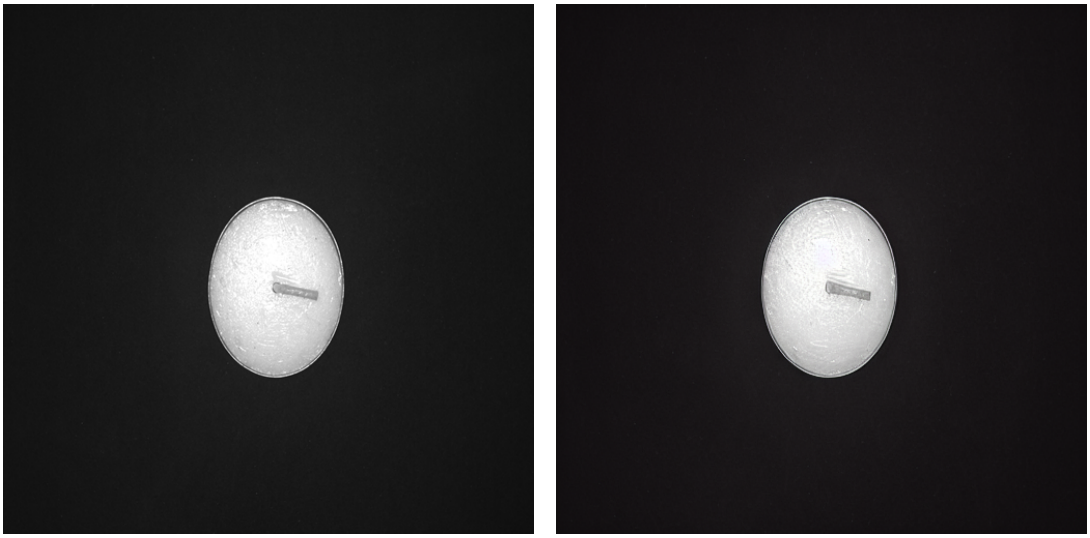
(d) High gain factor effect

(e) Plexiglass effect

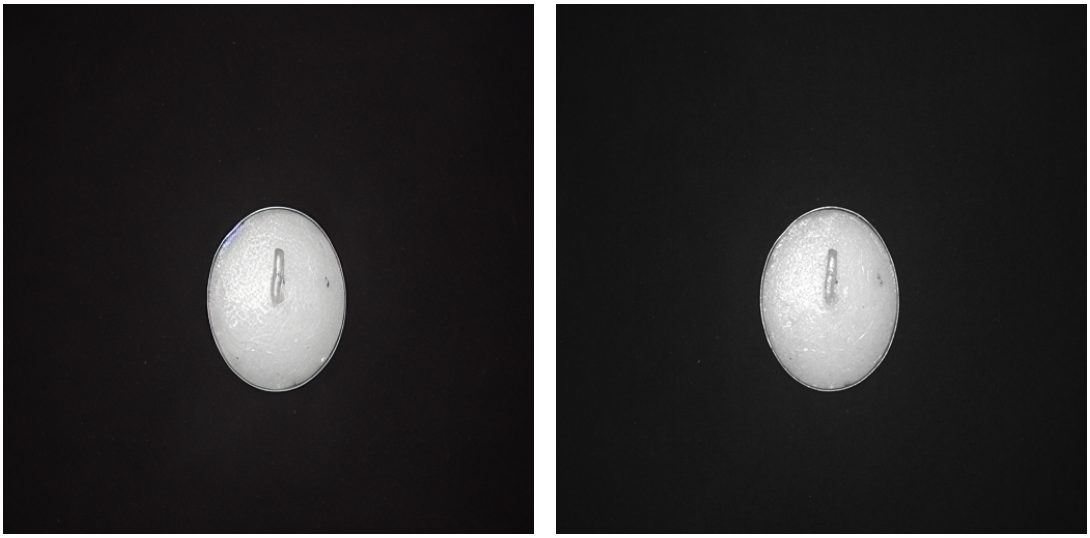
Figure 4.10.: Comparison of real and generated images for Nuts dataset



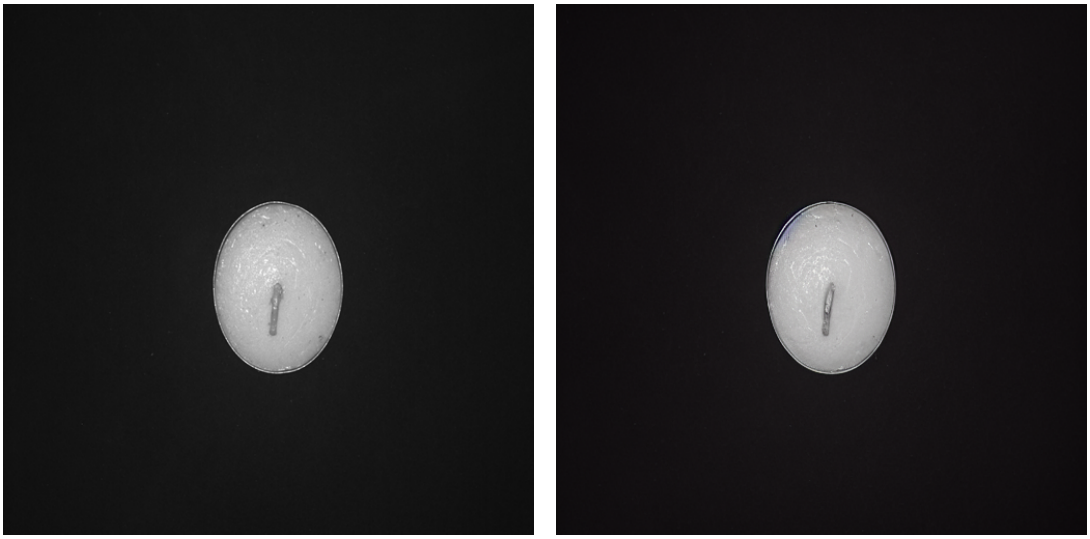
(a) Optimal setup



(b) Scattered sunlight effect



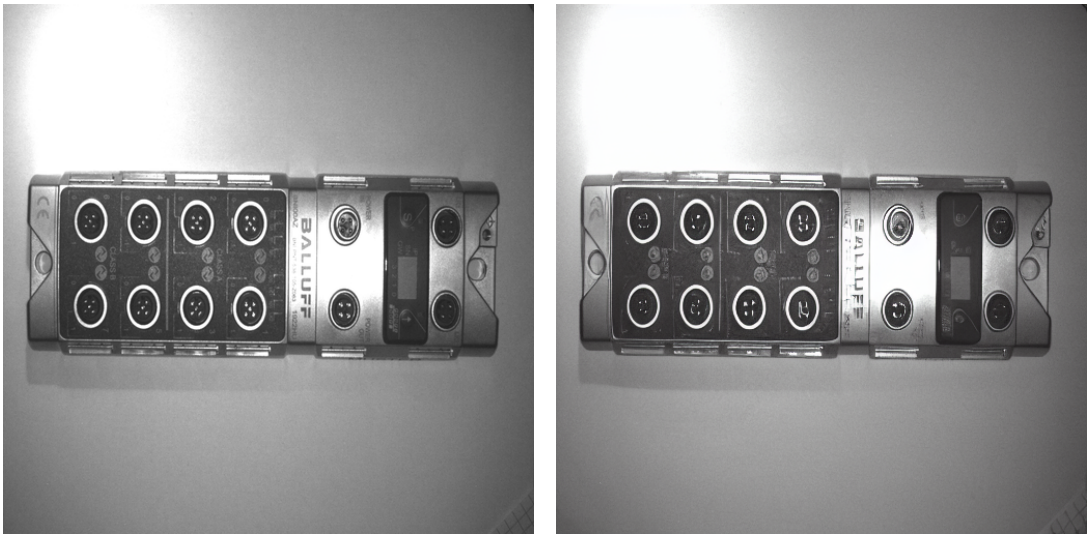
(c) Camera elevation (down)



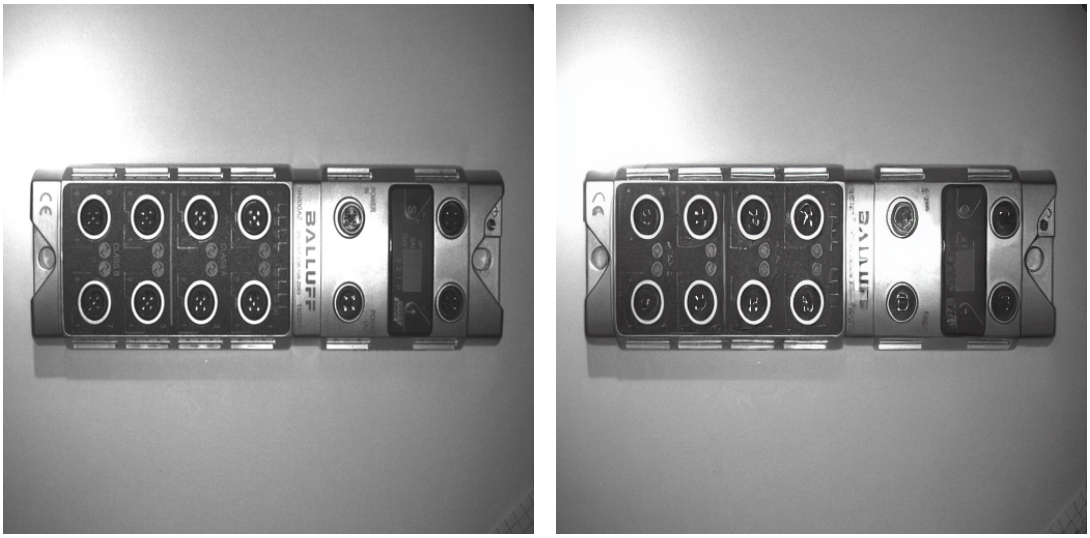
(d) Camera elevation (up)

(e) Random placement effect

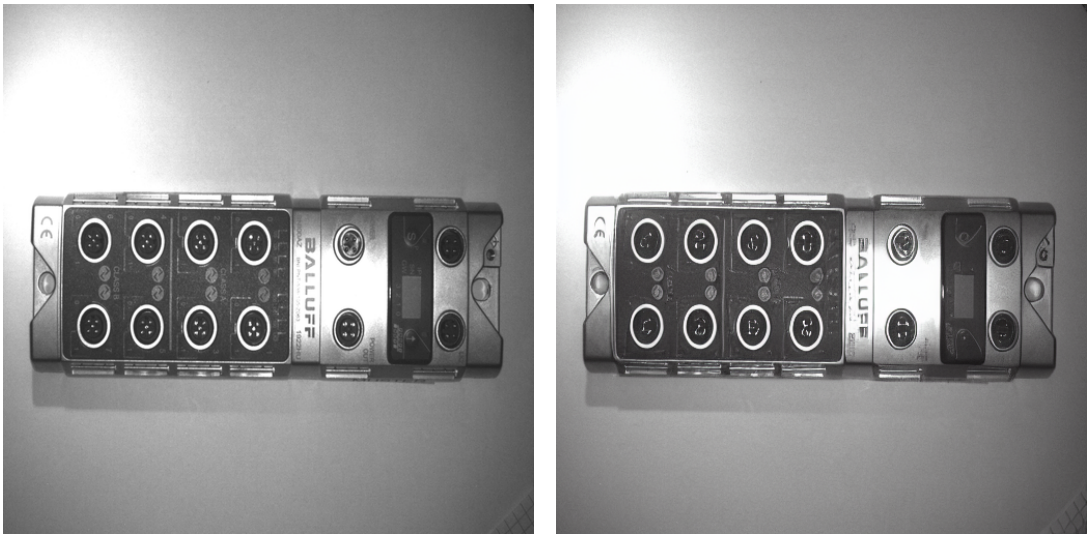
Figure 4.11.: Comparison of real and generated images for Candles dataset



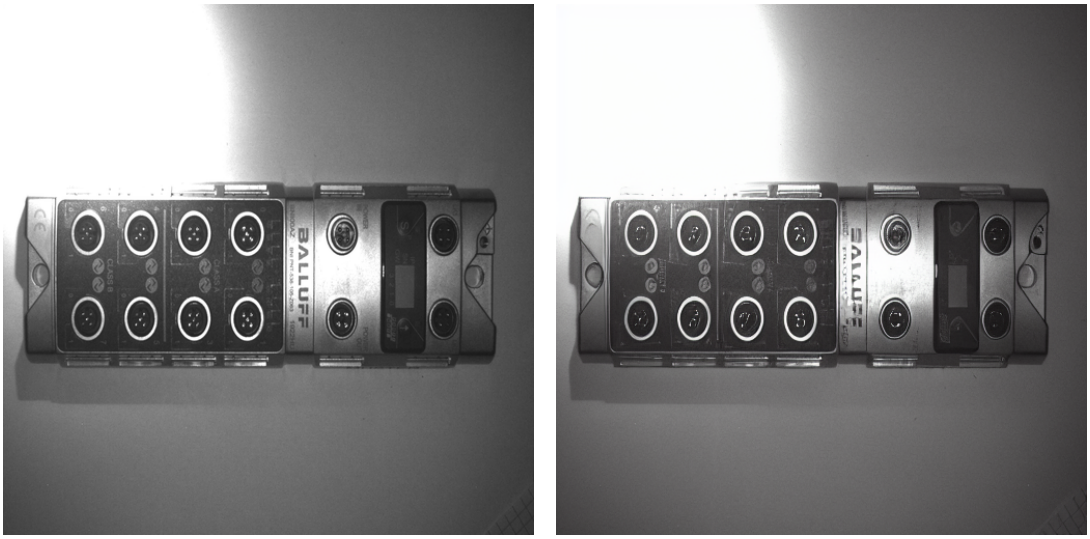
(a) Optimal setup



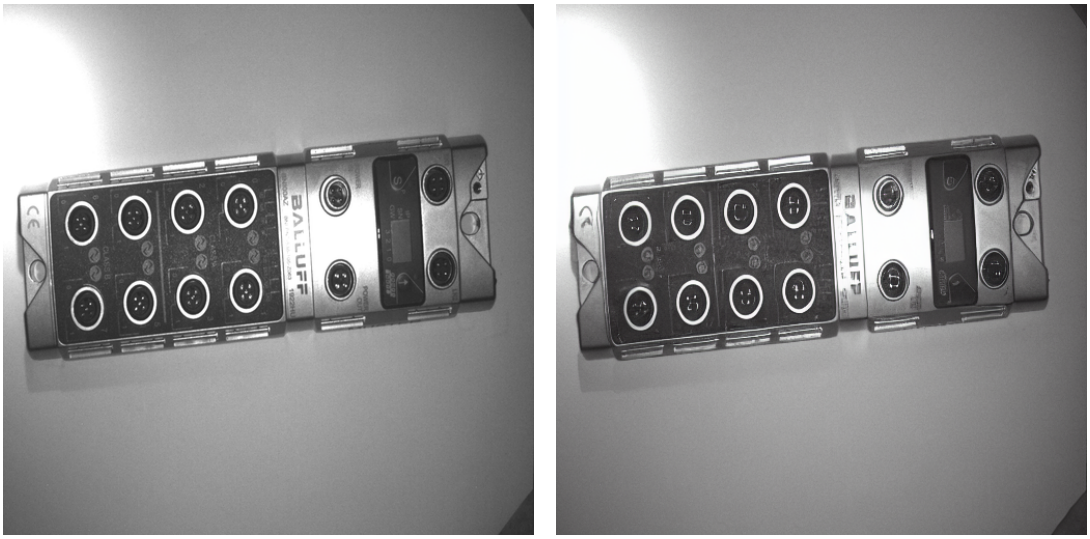
(b) Inclination (left)



(c) Inclination (right)

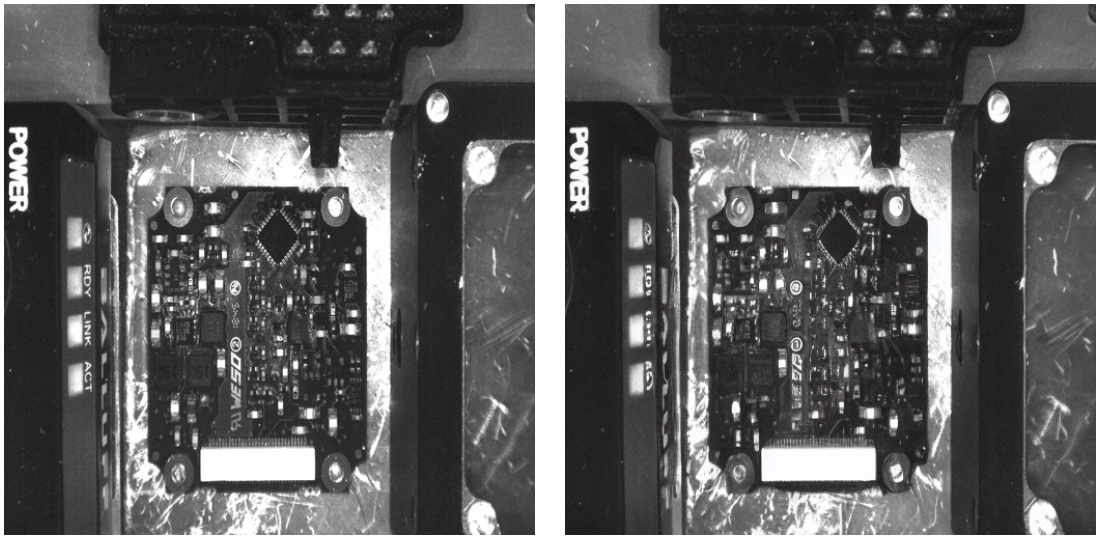


(d) Lighting effect

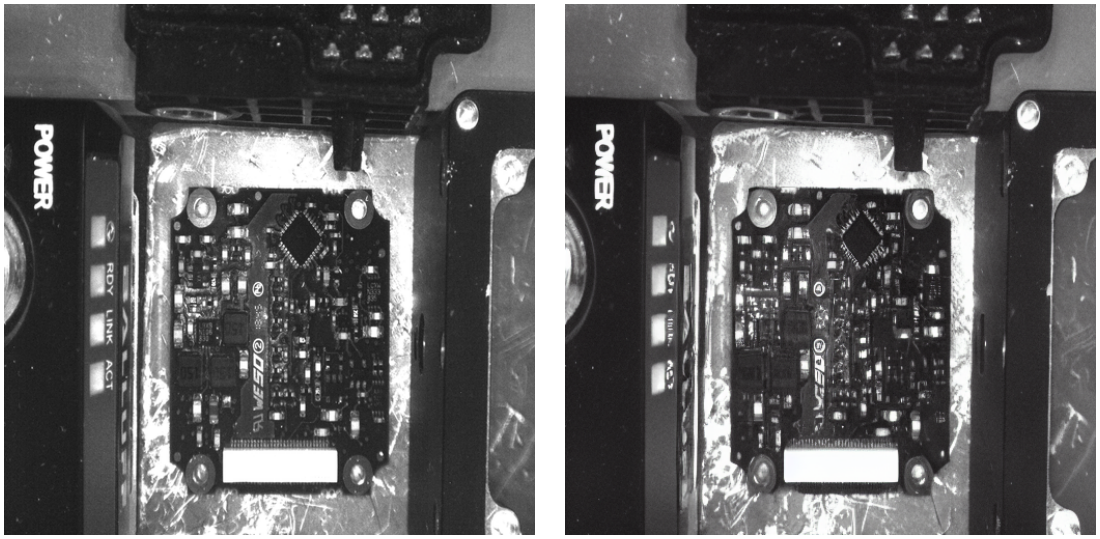


(e) Random placement effect

Figure 4.12.: Comparison of real and generated images for BNI dataset



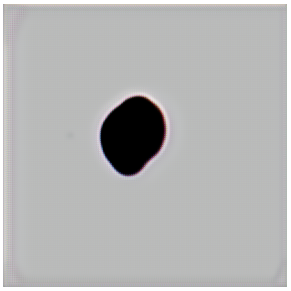
(a) Optimal setup



(b) Bad influence setup

Figure 4.13.: Comparison of real and generated images for PCB dataset

(a) Real Image



(b) CycleGAN-Generated Image

Metric	Value
ORB	0.6000
SSIM	0.7936
MSE	1105.6633
PSNR in dB	17.2401
Perceptual Loss P_L	0.6042

Figure 4.14.: Comparison of real a) and CycleGAN-generated nut image b) with corresponding metric values.

4.3.1. Analysis of CycleGAN-Generated Images

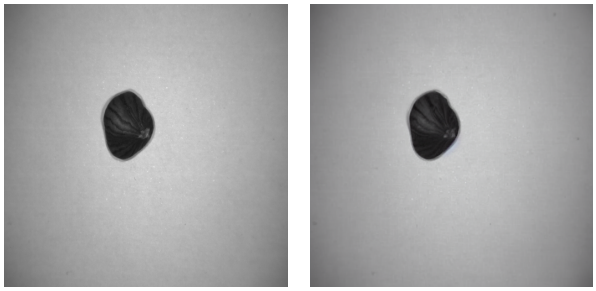
For CycleGAN, the image-to-image translation process involves transforming nut images captured under optimal conditions into nut images affected by various bad influences. In this framework, images from Domain A (optimal setup) remain intact, while artifacts and features from Domain B images are applied to these optimal images. Consequently, the generated images retain the shape and characteristics of Domain A nut images but incorporate elements such as shadow effects from Domain B. To evaluate the metrics for CycleGAN-generated images, a real image from Domain B that closely resembles the generated image was selected. This real image as shown in Figure 4.14 a) was compared to one of the generated outputs as given in Figure 4.14 b), which was created using a model saved at the 11th epoch during training.

The comparison of the generated images from CycleGAN with their corresponding real counterparts reveals some notable discrepancies in quality. Specifically, the evaluation metrics indicate an ORB = 0.6000 and SSIM = 0.7936, which signify a moderate level of feature retention and structural similarity between the real and generated images. However, the MSE = 1105.6633 and PSNR = 17.2401 dB suggest significant pixel differences and lower visual fidelity. Furthermore, a Perceptual Loss P_L = 0.6042 indicates that the generated images fail to retain essential perceptual characteristics of the real images. These factors collectively indicate that the CycleGAN-generated images are not suitable for evaluating the anomaly detector and are consequently not used in the evaluation process, as their reduced similarity to the real images could lead to misleading assessments of the model's performance.

4.3.2. Assessment of Outputs from Stable Diffusion

Nuts Dataset

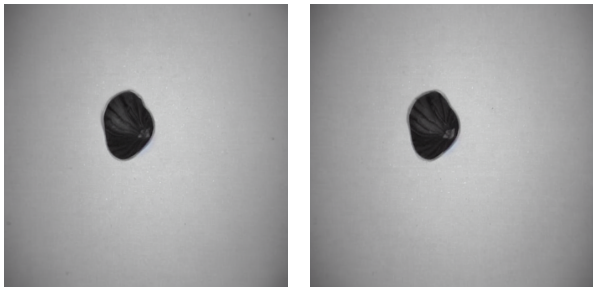
The comparison of the Nuts dataset between the real image (Real) and the generated image (Generated_1) as given in Figure 4.15, highlights effective image synthesis, with ORB = 0.9697 and SSIM = 0.9420, indicating strong feature retention and structural similarity. An MSE = 18.3361 suggests minor pixel differences, while a PSNR = 33.5168 dB reflects high



(a) Real (b) Generated_1

Metric	Value
ORB	0.9697
SSIM	0.9420
MSE	18.3361
PSNR in dB	33.5168
Perceptual Loss P_L	0.0625

Figure 4.15.: Metric values for real a) and generated b) nut images



(a) Generated_1 (b) Generated_2

Metric	Value
ORB	1.0000
SSIM	0.9678
MSE	6.2752
PSNR in dB	38.1736
Perceptual Loss P_L	0.0551

Figure 4.16.: Metric values for two generated nut images

visual quality. A Perceptual Loss $P_L = 0.0625$ further demonstrates that the generated image retains significant perceptual characteristics of the real image.

In contrast, the comparison between the two generated images (Generated_1 and Generated_2) as in Figure 4.16, shows perfect feature matching with an ORB = 1.0000 and high structural similarity (SSIM = 0.9678). An MSE = 6.2752 indicates minimal pixel discrepancies, and a PSNR = 38.1736 dB points to exceptional image quality. The lower Perceptual Loss $P_L = 0.0551$ confirms that both generated images are highly similar. Overall, while the generated image is similar to the real one, the two generated images are even more closely aligned.

The method of **comparing two sets of images using VGG16 feature extraction and a classifier** was applied to the Nuts dataset. For all 152 real OK images and 305 images captured under bad camera effects, corresponding synthetic images were generated using Stable Diffusion, resulting in a total of 914 images for analysis. The dataset was divided into training, validation, and test sets, with 80% allocated for training, 10% for validation, and 10% for testing. The composition of the dataset is summarized in Table 4.1.

In this analysis, VGG16 was utilized as a feature extractor to evaluate the similarity between real and generated Nut images. The images were resized to 224×224 pixels and normalized using the standard mean and standard deviation values specific to the VGG16 architecture. To extract features, the model processed images from two directories: one containing real images and the other containing generated images. The features were stored as flattened vectors, which were subsequently used for training a Random Forest classifier.

The Random Forest classifier was initialized with 100 estimators and trained on the extracted features. As given in Table 4.2, validation results indicated a high accuracy of approximately

Table 4.1.: Dataset Composition

Dataset	**Number of Images**
Training	639
Validation	137
Test	138
Total	914

Table 4.2.: Performance Metrics of the Random Forest Classifier

Metric	**Value**
Validation Accuracy	98.54%
Validation ROC AUC	0.9996
Test Accuracy	92.75%
Test ROC AUC	0.9275

98.54% and a ROC AUC score of 0.9996, demonstrating the classifier's effectiveness in distinguishing between real and generated images. However, the model's performance on the test dataset revealed an accuracy of approximately 92.75% and a ROC AUC score of 0.9275.

The confusion matrix as shown in Figure 4.17 offers a clear summary of the classifier's performance. For the Random Forest classifier, the matrix indicated 63 true positives, 65 true negatives, 5 false positives, and 5 false negatives. These results reflect the classifier's ability to distinguish between real and generated images, with minimal misclassification.

In this evaluation, the primary focus is to assess the similarity of synthetic images to real images. Ideally, the classifier should fail to distinguish between these two types of images, indicating that the generated images are sufficiently similar to the real ones. However, the classifier does not compare feature vectors in a pairwise manner (i.e., it does not directly compare each real image's features with those of a corresponding generated image). Instead, it learns to separate the overall distributions of real and generated images based on the feature vectors extracted from each image.

This method allows the classifier to achieve high accuracy, as it identifies patterns that consistently differentiate the two distributions, even if these patterns are subtle and spread across many images. Essentially, the classifier attempts to find a decision boundary in the feature space that best separates real images from generated images based on the features extracted by the model.

The PCA plot as shown in Figure 4.18 illustrates the distribution of feature vectors extracted from both real and generated images. In the plot, three distinct clusters can be observed, each representing different underlying features within the dataset. The red dots denote real images, while the blue dots signify generated images.

The presence of overlapping clusters suggests that the synthetic images share significant similarities with the real images in the feature space. Ideally, for an effective evaluation, the generated images should closely resemble the real images, as indicated by their proximity

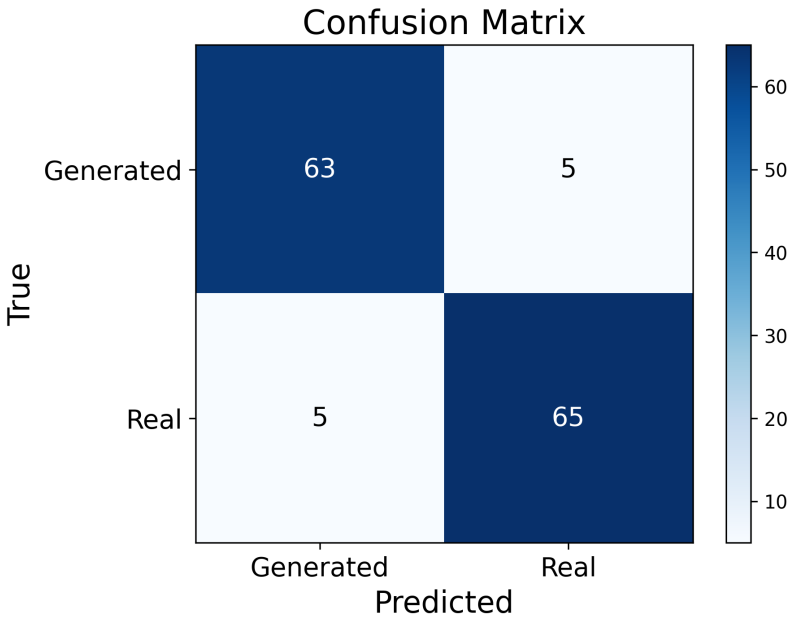


Figure 4.17.: Confusion matrix illustrating the performance of the Random Forest classifier in distinguishing between real and generated images.

in the PCA plot. This visual representation emphasizes the goal of this analysis, which is to assess how well the synthetic images capture the characteristics of the real images, suggesting that the synthetic images can be effectively utilized for evaluating anomaly detector performance.

Candles Dataset

The analysis of the candle dataset shows a similar trend to that observed in the Nuts dataset. The metric values presented in the Figure 4.19, indicate a strong similarity between the real and generated images. Notably, the two generated images exhibit even higher similarity at pixel level as per the metric values in Figure 4.20

BNI Dataset

The analysis of the BNI dataset reveals a slightly different trend compared to the Nuts and Candle datasets. As shown in Figure 4.21, an ORB = 0.9754 indicates strong feature matching between the real and generated images. However, a value of SSIM = 0.8312, which is notably lower than in previous datasets, suggests that while key features are preserved, there are more structural differences between the real and generated images. This is further supported by the relatively high MSE = 118.6083 and a PSNR = 26.8631 dB, indicating increased pixel discrepancies and lower image quality. The perceptual loss $P_L = 0.1834$ also reflects greater deviation in perceptual characteristics between the real and generated BNI images.

In contrast, the comparison of two generated images as in Figure 4.22 shows a closer alignment. ORB = 0.9647 and SSIM = 0.8621 reflect better structural similarity. Additionally, the MSE drops and the PSNR improves (MSE = 87.2969, PSNR = 28.1942 dB), indicating fewer pixel-level discrepancies. The perceptual loss also decreases to $P_L = 0.1371$, highlighting the greater similarity between the two generated images, although the differences

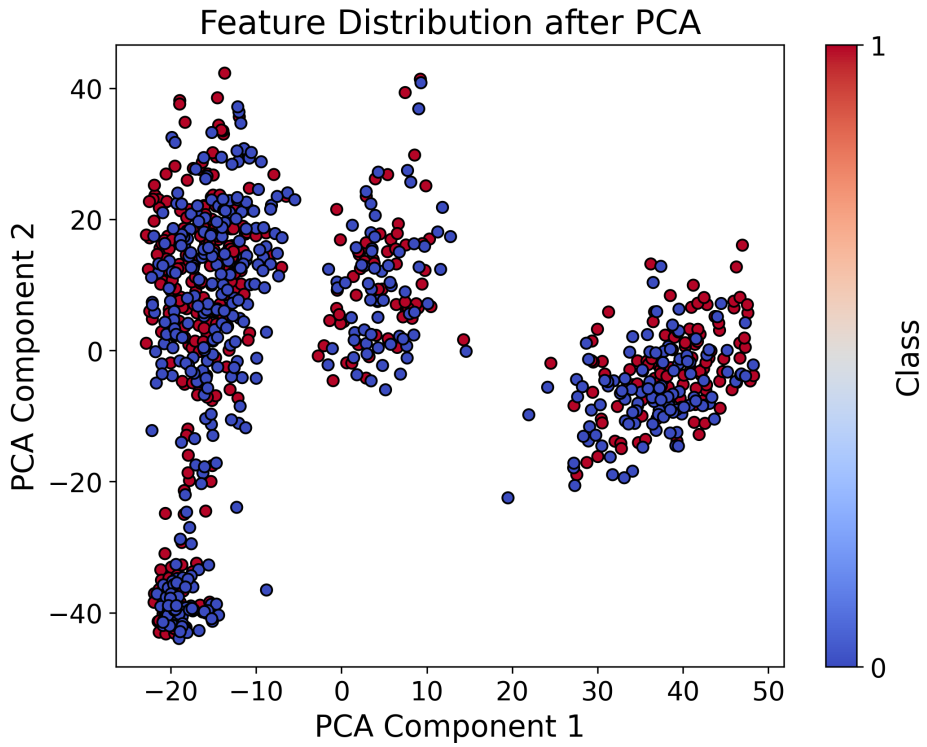
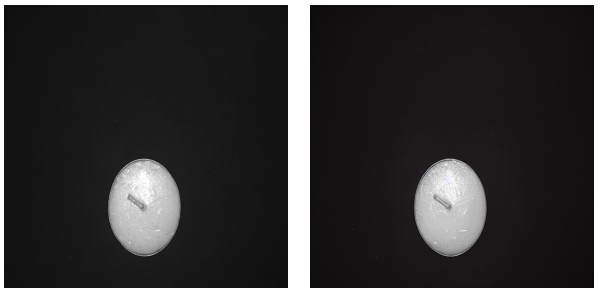


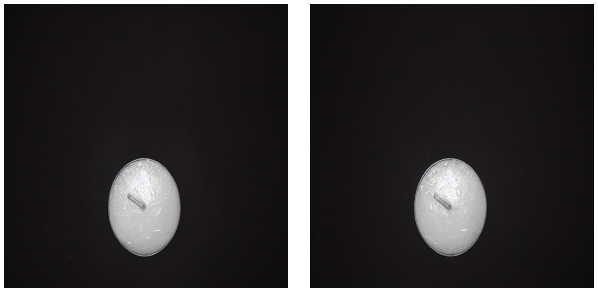
Figure 4.18.: PCA plot illustrating the distribution of feature vectors from real and generated images.



(a) Real (b) Generated_1

Metric	Value
ORB	0.9492
SSIM	0.9365
MSE	19.7778
PSNR in dB	35.1349
Perceptual Loss P_L	0.0849

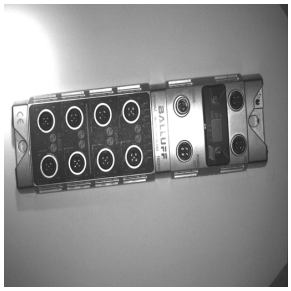
Figure 4.19.: Metric values for real a) and generated b) candle images



(a) Generated_1 (b) Generated_2

Metric	Value
ORB	0.9474
SSIM	0.9525
MSE	12.2984
PSNR in dB	36.9554
Perceptual Loss P_L	0.0708

Figure 4.20.: Metric values for two generated candle images



(a) Real

(b) Generated_1

Metric	Value
ORB	0.9754
SSIM	0.8312
MSE	118.6083
PSNR in dB	26.8631
Perceptual Loss	0.1834

Figure 4.21.: Metric values for real a) and generated b) BNI images

(a) Generated_1

(b) Generated_2

Metric	Value
ORB	0.9647
SSIM	0.8621
MSE	87.2969
PSNR in dB	28.1942
Perceptual Loss	0.1371

Figure 4.22.: Metric values for two generated BNI images

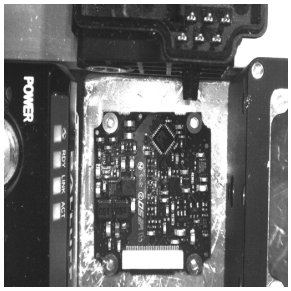
are still more pronounced compared to previous datasets.

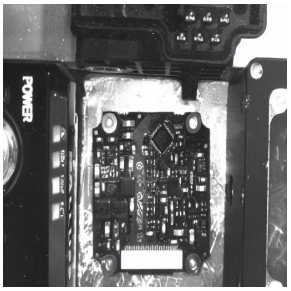
PCB Dataset

The analysis of the PCB dataset shows a similar trend to the BNI dataset. As given in Figure 4.23, an ORB = 0.8587 indicates moderate feature matching between the real and generated images. However, SSIM = 0.7087, coupled with a high MSE = 366.2571 and a relatively low PSNR = 22.0384 dB, suggests more significant structural and pixel-level differences. The perceptual loss P_L = 0.2796 reflects noticeable perceptual discrepancies between the real and generated PCB images.

The comparison between the two generated images in Figure 4.24 shows a slight improvement. The values of ORB and SSIM increases with ORB = 0.8815 and SSIM = 0.7564, indicating better structural similarity. However, the mean squared error remains high at MSE = 312.3232, and the PSNR = 22.4011 dB showed a minor improvement, and the perceptual loss remains relatively high at P_L = 0.3354, suggesting that while the two generated images are closer, perceptual differences are still evident.

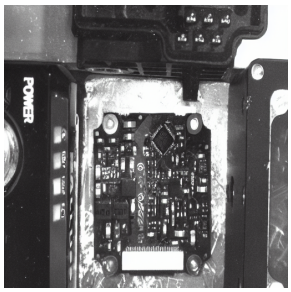
The BNI and PCB datasets, as discussed in Chapter 4.2.2, are more complex compared to the Nuts and Candles datasets, primarily due to their fine details such as text, labels, and small components. This intricacy is reflected not only in the visual analysis of the generated images, where elements like words, markings, and small circuits are not accurately reproduced, but also in the metric values. In both the BNI and PCB datasets, while the overall shape and structure of objects remain consistent, the finer details are lost in the synthetic images. This contrasts with the Nuts and Candles datasets, where the generated images exhibited stronger similarity both visually and metrically, as evidenced by higher ORB, SSIM, and PSNR values. The lower SSIM and higher MSE values for BNI and PCB highlight the

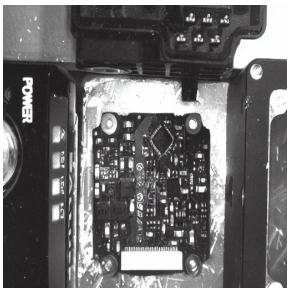

(a) Real


(b) Generated_1

Metric	Value
ORB	0.8587
SSIM	0.7087
MSE	366.2571
PSNR in dB	22.0384
Perceptual Loss	0.2796

Figure 4.23.: Comparison of real a) and generated b) PCB images along with their metric values.


(a) Generated_1


(b) Generated_2

Metric	Value
ORB	0.8815
SSIM	0.7564
MSE	312.3232
PSNR in dB	22.4011
Perceptual Loss	0.3354

Figure 4.24.: Metric values for two generated PCB images

difficulty in preserving the intricate details of these datasets, making them more challenging to synthesize accurately compared to the relatively simpler datasets like Nuts and Candles.

4.4. Anomaly Detection Results on the Nuts Dataset

This section presents the results of the anomaly detection on the Nuts dataset using three different approaches. These approaches were designed to assess how the anomaly detector's performance varies when trained with only optimal images, with the addition of real bad influence images, and with synthetic bad influence images. The choice to focus solely on the Nuts dataset is based on its comprehensive representation of various influences commonly encountered in industrial environments, such as shadow effects, high gain factors, and the plexiglass effect. In contrast, other datasets do not exhibit such clear representations of these influences, limiting their utility for robust anomaly detection training and evaluation. By analyzing the performance on the Nuts dataset, more precise insights can be drawn regarding the effectiveness of the proposed methodologies before potentially extending the analysis to other datasets.

4.4.1. Analysis of Loss Plots

The loss plots in Figure 4.25 illustrate the training and validation performance for the three approaches. While the minimum validation losses for all three approaches are relatively

small ($L_{val,1}$ = 0.0004 for Approach_1 and $L_{val,2}$ = $L_{val,3}$ = 0.0002 for Approach_2 and Approach_3), the key differences lie in the overall learning behavior as discussed in the following Chapter 4.4.2.

In Approach_1, which only utilized optimal images, the model's validation loss converged quickly, but the training loss ($L_{train,1}$) remained relatively higher as shown in Figure 4.25 a). This suggests that the model had limited exposure to diverse data and struggled to generalize effectively across the dataset.

Approach_2, which incorporated real bad influence images, demonstrates a stable decrease in both training loss ($L_{train,2}$) and validation loss as shown in Figure 4.25 b), reflecting better generalization as the model was exposed to a wider variety of conditions. This leads to a more balanced performance across both training and unseen data, as evidenced by the lower and more stable validation loss.

Approach_3, which used synthetic bad influence images, exhibits a loss curve that is very similar to Approach_2 as shown in Figure 4.25 c). The validation loss is on par with that of Approach_2, indicating that synthetic data can effectively support the model's generalization capabilities. The final validation loss shows that synthetic images were as effective as real bad influence images in enhancing model robustness.

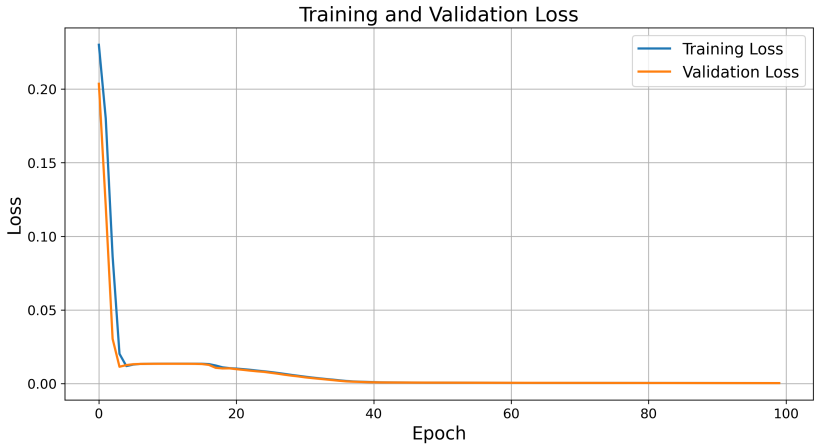
Overall, the latter two approaches performed notably better than the first, emphasizing the necessity of incorporating diverse data, including real-world camera influences. Notably, the findings indicate that synthetic data can significantly enhance model performance, achieving results comparable to those attained with real images.

4.4.2. Interpretation of Confusion Matrix and Metric Results

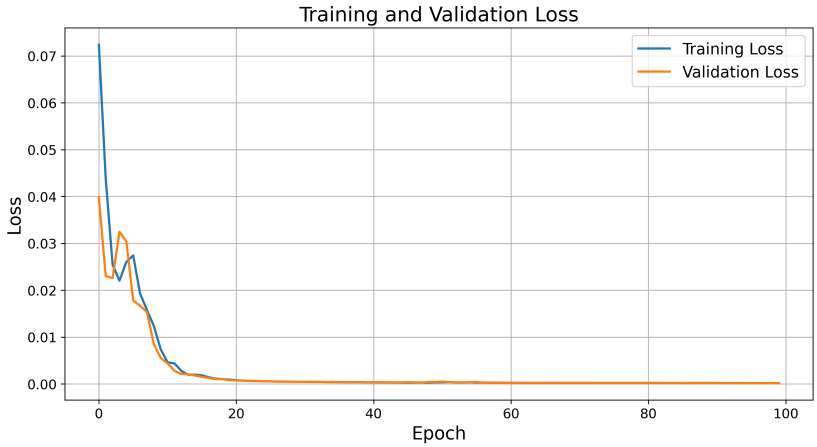
The results of the anomaly detection model demonstrate distinct differences in performance across the three approaches, as reflected in the confusion matrices in Figure 4.26 and metric values in Table 4.3. In the first approach, where only good images captured under optimal conditions were used for training, the model struggled to correctly classify good images with bad influences, resulting in a high number of false positives. This is evident in the confusion matrix shown in Figure 4.26 a), where 48 normal images were misclassified as anomalous, contributing to the relatively low accuracy A = 0.73. The recall for normal images was particularly low at r_{normal} = 0.47, indicating that many OK images with bad influences were mistakenly labeled as anomalies.

In the second approach, when real images with bad influences were introduced, the model's performance improved, with the accuracy rising to A = 0.75. The recall for normal images increased significantly to r_{normal} = 0.97, showing that the model became much better at recognizing good images with bad influences as normal. However, this improvement came with a trade-off in the number of false negatives, as 56 anomalous images were misclassified as normal as shown in shown in Figure 4.26 b), reducing the recall for anomalous images to $r_{anomalous}$ = 0.60. Despite the inclusion of real bad influence images, the accuracy did not improve substantially due to this increase in false negatives, highlighting the model's ongoing difficulty in distinguishing subtle anomalies.

Third approach as shown in Figure 4.26 c), which involved the use of synthetic images with bad influences, yielded results similar to second approach, with the accuracy remain-



(a) Approach_1



(b) Approach_2

(c) Approach_3

Figure 4.25.: Comparison of Training and Validation Loss across Different Approaches

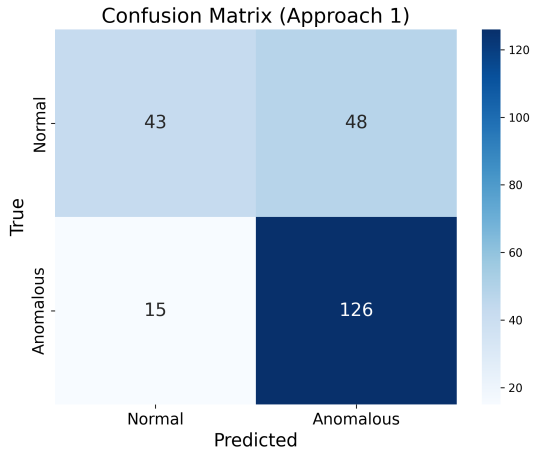
Table 4.3.: Summary of Metric Values across Different Approaches

Metric	**Approach 1**	**Approach 2**	**Approach 3**
Accuracy	0.73	0.75	0.75
Precision (Normal)	0.74	0.61	0.61
Precision (Anomalous)	0.72	0.97	0.95
Recall (Normal)	0.47	0.97	0.95
Recall (Anomalous)	0.89	0.60	0.62
F1-Score (Normal)	0.58	0.75	0.74
F1-Score (Anomalous)	0.80	0.74	0.75
Balanced Accuracy	0.68	0.78	0.78

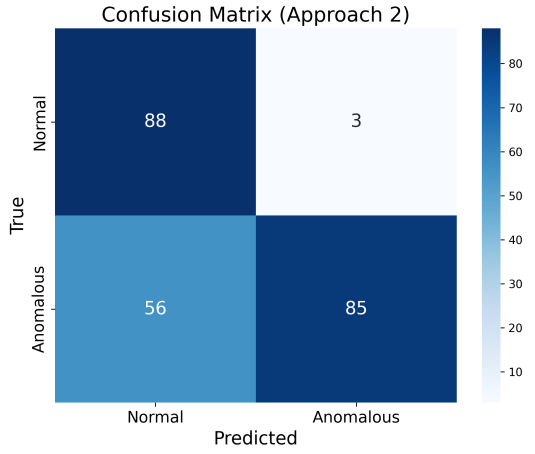
ing at A = 0.75. Both the recall for normal images $r_{\text{normal}} = 0.95$ and anomalous images $r_{\text{anomalous}} = 0.62$ were comparable, suggesting that the synthetic images were effective in helping the model recognize OK images with bad influences as normal. However, the issue of misclassifying anomalous images as normal persisted, indicating that while synthetic images helped with false positives, they did not solve the challenge of false negatives.

The balanced accuracy (B_{A}) metric, which accounts for both the recall of normal and anomalous images, provides a more comprehensive view of the model's performance. In the first approach, the balanced accuracy was $B_{\text{A}} = 0.68$, reflecting the model's poor recall for normal images. In both second and third approaches, the balanced accuracy improved to $B_{\text{A}} = 0.78$, showing that the addition of real or synthetic bad influence images helped balance the model's performance across both normal and anomalous classes.

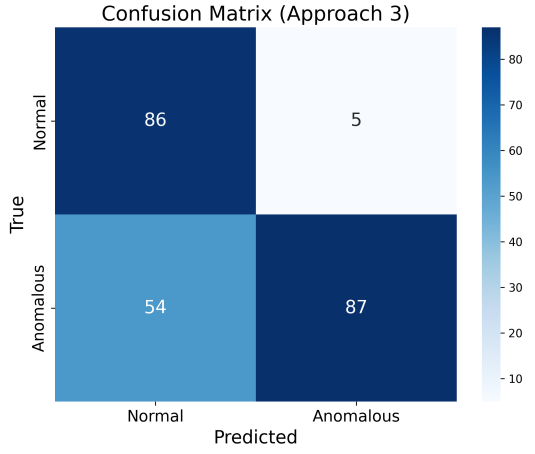
The reason for this limitation in accuracy improvement lies in the nature of the data. Some anomalous nut images closely resemble OK images with some bad influences like background, lighting and shadows. The anomalies, such as subtle shape imperfections in the Nuts, can be difficult for the model to distinguish from normal variations, especially when the defects are minor. As a result, even with the inclusion of real or synthetic bad influence images, the model still faces challenges in reliably identifying these visually subtle anomalies, leading to a plateau in accuracy improvements.



(a) Approach_1



(b) Approach_2



(c) Approach_3

Figure 4.26.: Comparison of Confusion Matrices across Different Approaches

5. Conclusion

In this master thesis, an approach for enhancing anomaly detection in industrial settings through synthetic image generation has been presented and evaluated. The research specifically addressed the challenges of acquiring sufficient labeled training data by utilizing generative models to create realistic synthetic images that replicate real-world variations. For this purpose, two generative models, CycleGAN and Stable Diffusion (using the `img2img` functionality), were employed to generate synthetic images. Initially, CycleGAN was utilized to facilitate image-to-image translation, aiming to introduce various camera influences into the images. However, significant challenges were encountered, including mode collapse and unstable training dynamics, primarily due to the limited variety and volume of available training data. As a result, the images generated by CycleGAN often lacked the necessary quality and realism for effective training of the anomaly detector.

In response to these limitations, Stable Diffusion was implemented as a more effective alternative. Although direct image-to-image translation was not possible with Stable Diffusion, meaning it couldn't explicitly introduce artifacts like bad camera influences in the same way CycleGAN does, Stable Diffusion WebUI allowed for the generation of multiple images by uploading a single real image to the `img2img` tab . Each generated image exhibited slight variations in texture, lighting, and other features, effectively simulating the desired camera influences. This approach enabled the creation of a diverse set of synthetic images from a limited number of real ones, thus expanding the dataset.

Evaluation metrics were then applied to compare the quality and similarity between the real and generated images, as well as between different generated samples. The synthetic images generated by Stable Diffusion showed a strong resemblance to real ones, while the variations between different synthetic images aligned with the intended goal of introducing subtle changes in texture and lighting.

A set of Nut images is generated corresponding to real Nut images in the Domain B dataset (setup with bad influences) of OK images. The anomaly detection performance was then evaluated using three different approaches: training with only optimal images, adding real bad influence images, and incorporating synthetic bad influence images. Adding real and synthetic bad influence images showed significant improvement in model performance, particularly in reducing false positives. Initially, when the model was trained with only optimal images, it struggled to classify normal images affected by bad camera influences accurately. However, once the model was trained with the inclusion of these bad influence images, it became adept at recognizing them as normal. Despite this progress, the model experienced an increase in false negatives, misclassifying some anomalies as normal. This issue arose because many good images with bad camera influences closely resembled certain NOK images, with only negligible differences, such as a slight defect in the nut shape. This similarity made it difficult for the model to identify true anomalies, underscoring the challenges presented by the nature of the data in achieving reliable anomaly detection.

While this research has made significant advancements in enhancing anomaly detection through synthetic image generation, several areas require further exploration. One potential direction is to investigate other techniques to improve anomaly detection by employing more powerful models that can accurately identify the subtle differences between normal and anomalous images. Enhancing the model architecture, such as incorporating advanced neural networks or ensemble methods, could lead to better feature extraction and classification performance.

A. Models

A.1. CycleGAN - Generator Model

Table A.1.: Model Summary - Generator Model

Layer (type)	**Output Shape**	**Param #**
InputLayer (input_1)	(None, 256, 256, 3)	0
Conv2D (conv2d)	(None, 256, 256, 64)	9,472
InstanceNormalization (instance_norm)	(None, 256, 256, 64)	128
Activation (activation)	(None, 256, 256, 64)	0
Conv2D (conv2d_1)	(None, 128, 128, 128)	73,856
InstanceNormalization (instance_norm)	(None, 128, 128, 128)	256
Activation (activation_1)	(None, 128, 128, 128)	0
Conv2D (conv2d_2)	(None, 64, 64, 256)	295,168
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Activation (activation_2)	(None, 64, 64, 256)	0
Conv2D (conv2d_3)	(None, 64, 64, 256)	590,080
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Activation (activation_3)	(None, 64, 64, 256)	0
Conv2D (conv2d_4)	(None, 64, 64, 256)	590,080
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Concatenate (concatenate)	(None, 64, 64, 512)	0
Conv2D (conv2d_5)	(None, 64, 64, 256)	1,179,904
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Activation (activation_4)	(None, 64, 64, 256)	0
Conv2D (conv2d_6)	(None, 64, 64, 256)	590,080
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Concatenate (concatenate_1)	(None, 64, 64, 768)	0
Conv2D (conv2d_7)	(None, 64, 64, 256)	1,769,728
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Activation (activation_5)	(None, 64, 64, 256)	0
Conv2D (conv2d_8)	(None, 64, 64, 256)	590,080
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Concatenate (concatenate_2)	(None, 64, 64, 1024)	0
Conv2D (conv2d_9)	(None, 64, 64, 256)	2,359,552
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Activation (activation_6)	(None, 64, 64, 256)	0
Conv2D (conv2d_10)	(None, 64, 64, 256)	590,080

InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Concatenate (concatenate_3)	(None, 64, 64, 1280)	0
Conv2D (conv2d_11)	(None, 64, 64, 256)	2,949,376
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Activation (activation_7)	(None, 64, 64, 256)	0
Conv2D (conv2d_12)	(None, 64, 64, 256)	590,080
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Concatenate (concatenate_4)	(None, 64, 64, 1536)	0
Conv2D (conv2d_13)	(None, 64, 64, 256)	3,539,200
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Activation (activation_8)	(None, 64, 64, 256)	0
Conv2D (conv2d_14)	(None, 64, 64, 256)	590,080
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Concatenate (concatenate_5)	(None, 64, 64, 1792)	0
Conv2D (conv2d_15)	(None, 64, 64, 256)	4,129,024
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Activation (activation_9)	(None, 64, 64, 256)	0
Conv2D (conv2d_16)	(None, 64, 64, 256)	590,080
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Concatenate (concatenate_6)	(None, 64, 64, 2048)	0
Conv2D (conv2d_17)	(None, 64, 64, 256)	4,718,848
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Activation (activation_10)	(None, 64, 64, 256)	0
Conv2D (conv2d_18)	(None, 64, 64, 256)	590,080
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Concatenate (concatenate_7)	(None, 64, 64, 2304)	0
Conv2D (conv2d_19)	(None, 64, 64, 256)	5,308,672
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Activation (activation_11)	(None, 64, 64, 256)	0
Conv2D (conv2d_20)	(None, 64, 64, 256)	590,080
InstanceNormalization (instance_norm)	(None, 64, 64, 256)	512
Concatenate (concatenate_8)	(None, 64, 64, 2560)	0
Conv2D (conv2d_21)	(None, 64, 64, 128)	2,949,248
InstanceNormalization (instance_norm)	(None, 64, 64, 128)	256
Activation (activation_12)	(None, 64, 64, 128)	0
Conv2D (conv2d_22)	(None, 64, 64, 3)	9,411
InstanceNormalization (instance_norm)	(None, 64, 64, 3)	6

Table A.2.: Model Details

Total Params	35,276,553
Trainable Params	35,276,553
Non-trainable Params	0

A.2. CycleGAN - Discriminator Model

Table A.3.: Model Summary - Discriminator Model

Layer	Output Shape	Param #
InputLayer (Input)	(None, 256, 256, 3)	0
Conv2D (Conv2D)	(None, 128, 128, 64)	3,136
LeakyReLU (LeakyReLU)	(None, 128, 128, 64)	-
Conv2D (Conv2D)	(None, 64, 64, 128)	131,200
InstanceNormalization (InstNorm)	(None, 64, 64, 128)	256
LeakyReLU (LeakyReLU)	(None, 64, 64, 128)	-
Conv2D (Conv2D)	(None, 32, 32, 256)	524,544
InstanceNormalization (InstNorm)	(None, 32, 32, 256)	512
LeakyReLU (LeakyReLU)	(None, 32, 32, 256)	-
Conv2D (Conv2D)	(None, 16, 16, 512)	2,097,664
InstanceNormalization (InstNorm)	(None, 16, 16, 512)	1,024
LeakyReLU (LeakyReLU)	(None, 16, 16, 512)	-
Conv2D (Conv2D)	(None, 16, 16, 1)	8,193

Table A.4.: Model Details

Total Params	2,766,529
Trainable Params	2,766,529
Non-trainable Params	0

A.3. Requirements for Stable Diffusion Setup

- **GPU:** A computer or laptop with a dedicated GPU, requiring at least 4GB of VRAM and 8GB of RAM.
- **Python 3.10.6:** Install the appropriate version of Python to ensure compatibility with the Stable Diffusion WebUI.
- **Stable Diffusion Model:** Download the Stable Diffusion XL (SDXL) model from https://stability.ai. Specifically, download the `sd_xl_base_1.0_0.9vae.safetensors` file for image generation.
- **Stable Diffusion WebUI:** Download the Stable Diffusion WebUI from the https://github.com/AUTOMATIC1111/stable-diffusion-webui GitHub repository. After downloading the ZIP file and extracting it, run the `webui-user.bat` file (for Windows) or `webui-user.sh` (for Linux) to launch the interface.

List of Figures

List of Tables

Bibliography

[1] Z. Yang, F. Zhan, K. Liu, M. Xu and S. Lu, "Ai-generated images as data source: The dawn of synthetic era," 2023. [Online]. Available: https://arxiv.org/abs/2310.01830

[2] J. W. Anderson, M. Ziolkowski, K. Kennedy and A. W. Apon, "Synthetic image data for deep learning," *arXiv preprint arXiv:2212.06232*, 2022.

[3] K. El Emam, L. Mosquera and R. Hoptroff, *Practical synthetic data generation: balancing privacy and the broad availability of data*. O'Reilly Media, 2020.

[4] L. Wuttke, "Machine learning vs. deep learning: What's the difference?" April 2024. [Online]. Available: https://datasolut.com/machine-learning-vs-deep-learning/

[5] S. B. Maind, P. Wankar *et al.*, "Research paper on basic of artificial neural network," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 2, no. 1, pp. 96–100, 2014.

[6] L. Hardesty, "Explained: neural networks," *MIT News*, vol. 14, 2017.

[7] V. Luhaniwal, "Forward propagation in neural networks — simplified (math and code version)," May 2019. [Online]. Available: https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250

[8] Y. LeCun, D. Touresky, G. Hinton and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, vol. 1, 1988, pp. 21–28.

[9] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[10] Z. Tu, "Learning generative models via discriminative approaches," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[11] L. Deng and N. Jaitly, "Deep discriminative and generative models for speech pattern recognition," in *Handbook of pattern recognition and computer vision*. World Scientific, 2016, pp. 27–52.

[12] Wikipedia contributors, "Discriminative model — wikipedia, the free encyclopedia," https://en.wikipedia.org/wiki/Discriminative_model, last edited: 4 October 2024.

[13] N. Arun, "Generative vs discriminative models: Differences & use cases," https://www.datacamp.com/blog/generative-vs-discriminative-models, Sep. 2020.

[14] C. Zheng, G. Wu and C. Li, "Toward understanding generative data augmentation," *Advances in neural information processing systems*, vol. 36, pp. 54 046–54 060, 2023.

[15] A. Ng and M. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, vol. 14, 2001.

[16] T. E. Raghunathan, "Synthetic data," *Annual review of statistics and its application*, vol. 8, no. 1, pp. 129–140, 2021.

[17] V. Bolón-Canedo, N. Sánchez-Maroño and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," *Knowledge and information systems*, vol. 34, pp. 483–519, 2013.

[18] J. M. Abowd and L. Vilhuber, "How protective are synthetic data?" in *International Conference on Privacy in Statistical Databases*. Springer, 2008, pp. 239–246.

[19] K. Walker, "Synthetic data: Unlocking the power of data and skills for machine learning," https://dataingovernment.blog.gov.uk/2020/08/20/synthetic-data-unlocking-the-power-of-data-and-skills-for-machine-learning/, August 2020.

[20] Y. Lu, M. Shen, H. Wang, X. Wang, C. van Rechem and W. Wei, "Machine learning for synthetic data generation: a review," *arXiv preprint arXiv:2302.04062*, 2023.

[21] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," *Advances in neural information processing systems*, vol. 29, 2016.

[22] E. Schönfeld, "Improving quality and controllability in gan-based image synthesis," 2022.

[23] B. Wang and J. J. Vastola, "Diffusion models generate images like painters: an analytical theory of outline first, details later," *arXiv preprint arXiv:2303.02490*, 2023.

[24] L. Cai, H. Gao and S. Ji, "Multi-stage variational auto-encoders for coarse-to-fine image generation," in *Proceedings of the 2019 SIAM international conference on data mining*. SIAM, 2019, pp. 630–638.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[26] N. Giocoli, "Nash equilibrium," *History of political economy*, vol. 36, no. 4, pp. 639–666, 2004.

[27] S. Zhang, "On the nash equilibrium of moment-matching gans for stationary gaussian processes," in *Mathematical and Scientific Machine Learning*. PMLR, 2022, pp. 113–128.

[28] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng and F.-Y. Wang, "Generative adversarial networks: introduction and outlook," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[29] M. Georgopoulos, J. Oldfield, G. G. Chrysos and Y. Panagakis, "Cluster-guided image synthesis with unconditional models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 543–11 552.

[30] G. Lee, H. Kim, J. Kim, S. Kim, J.-W. Ha and Y. Choi, "Generator knows what discriminator should learn in unconditional gans," in *European Conference on Computer Vision*. Springer, 2022, pp. 406–422.

[31] T. DeVries, A. Romero, L. Pineda, G. W. Taylor and M. Drozdzal, "On the evaluation of conditional gans," *arXiv preprint arXiv:1907.08175*, 2019.

[32] M. Mirza, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[33] A. Radford, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[34] M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[35] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2794–2802.

[36] T. Karras, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[37] F. Eckerli and J. Osterrieder, "Generative adversarial networks in finance: an overview," *arXiv preprint arXiv:2106.06364*, 2021.

[38] J. Gauthier, "Conditional generative adversarial nets for convolutional face generation," *Class project for Stanford CS231N: convolutional neural networks for visual recognition, Winter semester*, vol. 2014, no. 5, p. 2, 2014.

[39] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[40] P. Sangkloy, J. Lu, C. Fang, F. Yu and J. Hays, "Scribbler: Controlling deep image synthesis with sketch and color," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5400–5409.

[41] J.-Y. Zhu, T. Park, P. Isola and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[42] E. Baykal Kablan, "Regional realness-aware generative adversarial networks for stain normalization," *Neural Computing and Applications*, vol. 35, 05 2023.

[43] A. Almahairi, S. Rajeshwar, A. Sordoni, P. Bachman and A. Courville, "Augmented cyclegan: Learning many-to-many mappings from unpaired data," in *International conference on machine learning*. PMLR, 2018, pp. 195–204.

[44] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet and A. Zisserman, "Temporal cycle-consistency learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1801–1810.

[45] H.-W. Dong and Y.-H. Yang, "Towards a deeper understanding of adversarial losses," *arXiv preprint arXiv:1901.08753*, 2019.

[46] D. P. Kingma, M. Welling *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[47] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[48] J. Ho, A. Jain and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[49] L. Weng, "What are diffusion models?" *lilianweng. github. io*, p. 21, 2021.

[50] Yainnoware, "Decoding the math behind9 diffusion models," https://yainnoware.blogspot.com/2022/11/decoding-math-behind-diffusion-models.html, November 20 2022.

[51] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International conference on machine learning*. PMLR, 2021, pp. 8162–8171.

[52] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen and I. Sutskever, "Zero-shot text-to-image generation," in *International conference on machine learning*. Pmlr, 2021, pp. 8821–8831.

[53] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[54] D. Alexey, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv: 2010.11929*, 2020.

[55] OpenAI, "Dall·e: Creating images from text," https://openai.com/index/dall-e/, January 2021.

[56] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International conference on machine learning*. PMLR, 2021, pp. 8162–8171.

[57] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[58] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741*, 2021.

[59] R. Rombach, A. Blattmann, D. Lorenz, P. Esser and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[60] OpenAI, "Dall·e 2: A new era of image generation," 2022. [Online]. Available: https://openai.com/dall-e-2/

[61] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[62] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in neural information processing systems*, vol. 35, pp. 36 479–36 494, 2022.

[63] MidJourney, "Midjourney: Surrealistic text-to-image generation," 2022, accessed: 2022-09-15. [Online]. Available: https://www.midjourney.com/

[64] J. Rando, D. Paleka, D. Lindner, L. Heim and F. Tramèr, "Red-teaming the stable diffusion safety filter," *arXiv preprint arXiv:2210.04610*, 2022.

[65] A. Borji, "Generated faces in the wild: Quantitative comparison of stable diffusion, midjourney and dall-e 2," *arXiv preprint arXiv:2210.00586*, 2022.

[66] L. Martí, N. Sanchez-Pi, J. M. Molina and A. C. B. Garcia, "Anomaly detection based on sensor data in petroleum industry applications," *Sensors*, vol. 15, no. 2, pp. 2774–2797, 2015.

[67] M. A. Panza, M. Pota and M. Esposito, "Anomaly detection methods for industrial applications: A comparative study," *Electronics*, vol. 12, no. 18, p. 3971, 2023.

[68] H. Torabi, S. L. Mirtaheri and S. Greco, "Practical autoencoder based anomaly detection by using vector reconstruction error," *Cybersecurity*, vol. 6, no. 1, p. 1, 2023.

[69] K. Renu, "Anomaly detection using autoencoders," https://towardsdatascience.com/anomaly-detection-using-autoencoders-5b032178a1ea, Jan. 2021.

[70] U. Michelucci, "An introduction to autoencoders," *arXiv preprint arXiv:2201.03898*, 2022.

[71] D. Bank, N. Koenigstein and R. Giryes, "Autoencoders," *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pp. 353–374, 2023.

[72] Y. Wang, C. Wu, L. Herranz, J. Van de Weijer, A. Gonzalez-Garcia and B. Raducanu, "Transferring gans: generating images from limited data," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 218–234.

[73] TensorFlow, "Imagedatagenerator." [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

[74] AUTOMATIC1111, "Stable diffusion web ui," August 2022. [Online]. Available: https://github.com/AUTOMATIC1111/stable-diffusion-webui

[75] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li and J. Zhu, "Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models," *arXiv preprint arXiv:2211.01095*, 2022.

[76] N. Diffusion, "Transform images into stunning ai art with stable diffusion," 2024, updated: 2024-05-16. [Online]. Available: https://www.nextdiffusion.ai/tutorials/transform-images-into-stunning-ai-art-with-stable-diffusion

[77] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[78] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. Springer, 2006, pp. 430–443.

[79] M. Calonder, V. Lepetit, C. Strecha and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. Springer, 2010, pp. 778–792.

[80] A. Bookstein, V. A. Kulyukin and T. Raita, "Generalized hamming distance," *Information Retrieval*, vol. 5, pp. 353–375, 2002.

[81] M. Norouzi, D. J. Fleet and R. R. Salakhutdinov, "Hamming distance metric learning," *Advances in neural information processing systems*, vol. 25, 2012.

[82] I. Bakurov, M. Buzzelli, R. Schettini, M. Castelli and L. Vanneschi, "Structural similarity index (ssim) revisited: A data-driven approach," *Expert Systems with Applications*, vol. 189, p. 116087, 2022.

[83] D. Brunet, E. R. Vrscay and Z. Wang, "On the mathematical properties of the structural similarity index," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1488–1499, 2011.

[84] U. Sara, M. Akter and M. S. Uddin, "Image quality assessment through fsim, ssim, mse and psnr—a comparative study," *Journal of Computer and Communications*, vol. 7, no. 3, pp. 8–18, 2019.

[85] M. Han, H. Shim and J. Baek, "Perceptual ct loss: implementing ct image specific perceptual loss for cnn-based low-dose ct denoiser," *IEEE Access*, vol. 10, pp. 62 412–62 422, 2022.

[86] M. S. Rad, B. Bozorgtabar, U.-V. Marti, M. Basler, H. K. Ekenel and J.-P. Thiran, "Srobb: Targeted perceptual loss for single image super-resolution," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 2710–2719.

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

[88] M. Florkowski, "Anomaly detection, trend evolution, and feature extraction in partial discharge patterns," *Energies*, vol. 14, no. 13, p. 3886, 2021.

[89] M. P. LaValley, "Logistic regression," *Circulation*, vol. 117, no. 18, pp. 2395–2399, 2008.

[90] S. J. Rigatti, "Random forest," *Journal of Insurance Medicine*, vol. 47, no. 1, pp. 31–39, 2017.

[91] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[92] D. Saxena and J. Cao, "Generative adversarial networks (gans) challenges, solutions, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–42, 2021.

[93] N. Kodali, J. Abernethy, J. Hays and Z. Kira, "On convergence and stability of gans," *arXiv preprint arXiv:1705.07215*, 2017.

[94] H. Thanh-Tung and T. Tran, "Catastrophic forgetting and mode collapse in gans," in *2020 international joint conference on neural networks (ijcnn)*. IEEE, 2020, pp. 1–10.

Declaration

Herewith, I declare that I have developed and written the enclosed thesis entirely by myself and that I have not used sources or means except those declared.

This thesis has not been submitted to any other authority to achieve an academic grading and has not been published elsewhere.

Stuttgart, 22.10.2024 Swathi Kumar