CHENNAI MATHEMATICAL INSTITUTE

Defect:GEN

Synthetic Image Generation

Generating Synthetic Images using AI to Enhance Training Datasets

Anushka Chakraborty
Arnab Bera

Acknowledgement

We would like to take this opportunity to sincerely thank Chennai Mathematical Institute (CMI) and Algolabs for giving us this wonderful opportunity to work on such an insightful and real-world project. This experience has greatly enriched our knowledge and enhanced our practical skills.

We are deeply grateful to Mr. Suman and Mr. Arnab for their constant guidance, encouragement, and support throughout every phase of the project. Their expertise and thoughtful feedback played a key role in shaping the quality of our work.

We would also like to express our heartfelt thanks to Mr. Sudhir, Ms. Monali, and the entire AlgoLabs management team for facilitating this opportunity and providing continuous support.

Our sincere gratitude also goes to our professors at CMI, whose teaching and mentorship laid the foundation for our understanding and inspired us to apply our knowledge to practical challenges.

Finally, we are thankful to our parents for their unwavering support, motivation, and belief in us throughout this journey.

Contents

Introduction

Synthetic image generation plays a pivotal role in modern defect detection, especially when real-world datasets are severely limited—often comprising just 25–30 annotated defect images. Such small datasets are insufficient for training robust deep-learning models. Generative approaches, including GANs, VAEs, and diffusion models, have proven effective in producing diverse, high-fidelity defect samples that enrich these limited datasets while offering control over defect characteristics such as shape, location, and texture variability. The quality of generated images is evaluated using metrics like PSNR (image fidelity), SSIM (structural similarity), LPIPS (perceptual similarity), and FID (distribution matching), which collectively assess both visual realism and statistical alignment with real data. Synthetic image generation thus provides a scalable, controllable, and cost-efficient solution for enhancing defect detection systems under data-scarce conditions.

Methodology and Evaluation

Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GANs) are a class of generative models introduced by Ian Goodfellow et al. in 2014. GANs are designed to generate new data instances that resemble a given data distribution. They are based on a game-theoretic scenario involving two neural networks: a **generator** (G) and a **discriminator** (D).

Components of GAN

- **Generator (G):** Takes as input a random noise vector $z \sim p_z(z)$ from a prior distribution (e.g., Gaussian) and outputs a synthetic sample $G(z)$. Its objective is to generate data that is indistinguishable from real data.
- **Discriminator (D):** Takes an input x and outputs a probability $D(x)$ representing the likelihood that x comes from the real data distribution rather than being generated. It is a binary classifier trained to distinguish real data from generated data.

Adversarial Training

The generator and discriminator play a *minimax game* with the value function $V(G, D)$:

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

Here:

- $p_{data}(x)$ is the real data distribution.
- $p_z(z)$ is the prior distribution for latent variable z.
- $D(x)$ is the discriminator's estimate of the probability that x is real.
- $G(z)$ is the synthetic data sample generated from noise z.

Objective and Equilibrium

The final goal is to reach a Nash equilibrium where:

$$D(x) = \frac{1}{2} \quad \text{for all } x$$

This implies that the discriminator is unable to distinguish between real and fake samples, and the generator has successfully learned the data distribution:

$$p_g(x) = p_{data}(x)$$

Mathematical Insight

If the discriminator D is optimal for a fixed generator G, it can be shown that:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

Substituting into the value function gives the generator's loss:

$$C(G) = \max_D V(G, D) = -\log(4) + 2 \cdot \text{JSD}(p_{data} \| p_g)$$

where JSD denotes the Jensen-Shannon divergence. Minimizing $C(G)$ is equivalent to minimizing the JSD between the real and generated distributions.

Advantages

- Capable of generating highly realistic samples.
- Do not require explicit likelihood estimation or assumptions about data distribution.
- Widely used in image synthesis, style transfer, super-resolution, etc.

Disadvantages

- Training is unstable and sensitive to hyperparameters.
- Mode collapse: generator produces limited variety of samples.
- No explicit metric for convergence.

StyleGAN

StyleGAN is a generative adversarial network proposed by Karras et al., which introduces a novel style-based synthesis approach for high-fidelity image generation. Unlike traditional GANs that map a latent vector directly to the image through a generator, StyleGAN introduces an intermediate latent space and injects style at multiple levels of the generation process.

Architecture Overview

The StyleGAN architecture is composed of the following key components:

- **Mapping Network:** An 8-layer multilayer perceptron that transforms a latent vector $\mathbf{z} \sim \mathcal{N}(0, I)$ into an intermediate latent vector $\mathbf{w}$ in a disentangled space $\mathcal{W}$.
- **Synthesis Network:** A generator that begins from a learned constant tensor and progressively generates an image through convolutional layers. Each layer receives style information from $\mathbf{w}$ via modulation.

- **Style Modulation:** Each convolutional layer is modulated using an affine transformation of the style vector:

$$\text{ModulatedConv}(x) = \gamma(\mathbf{w}) \cdot \text{Conv}(x) + \beta(\mathbf{w}),$$

  where $\gamma(\cdot)$ and $\beta(\cdot)$ are learnable transformations.

- **Noise Injection:** Uncorrelated Gaussian noise is added to each layer independently to enable stochastic variation in fine image details:

$$x_l = x_l + \sigma_l \cdot \mathcal{N}(0, I).$$

- **Discriminator:** A standard convolutional neural network trained to distinguish between real and generated images.

Training Objective

StyleGAN is trained using the standard non-saturating GAN loss. The generator and discriminator losses are:

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z}}[\log D(G(\mathbf{z}))], \tag{1}$$
$$\mathcal{L}_D = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))] \tag{2}$$

To improve training stability and image quality, StyleGAN2 introduces additional regularization terms:

- **R1 Regularization (Discriminator)**:

$$\mathcal{L}_{R1} = \lambda \cdot \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \left[\|\nabla D(\mathbf{x})\|^2 \right].$$

- **Path Length Regularization (Generator)**:

$$\mathbb{E}_{\mathbf{z}} \left[(\|\nabla_{\mathbf{w}} G(\mathbf{w})\|_2 - a)^2 \right],$$

  where a is the target path length.

Advantages

StyleGAN provides the following benefits:

- High-resolution and photorealistic image generation.
- Disentangled latent representations enabling controllable image synthesis.
- Layer-wise control over image structure and fine details.
- Effective for style mixing, interpolation, and semantic editing in the latent space.

Limitations

Despite its advantages, StyleGAN has several drawbacks:

- High computational and memory requirements for training.
- Requires large-scale datasets unless fine-tuning is used.
- Early versions (v1 and v2) suffer from spatial inconsistencies due to aliasing.
- Not inherently suited for conditional generation tasks without architectural modifications.

StyleGAN remains one of the most powerful and flexible architectures for image synthesis and has influenced many subsequent generative models. Its ability to disentangle and manipulate semantic features makes it particularly attractive for controlled image generation tasks.

DefectGAN

DefectGAN is a generative adversarial network specifically designed for synthesizing realistic defects in industrial images, such as surface scratches, dents, or anomalies, to improve defect detection models where defective samples are rare. The core motivation behind DefectGAN lies in the scarcity and cost of acquiring annotated defective data, which limits the performance of supervised learning models in quality inspection tasks. By generating synthetic yet realistic defect patterns on defect-free images, DefectGAN supports data augmentation and enhances robustness in defect detection systems.

Objective Function: DefectGAN adopts a conditional GAN framework. Let $\mathbf{x}_{\text{clean}}$ be a defect-free image, and $\mathbf{z}$ be a latent noise vector. The generator G learns to synthesize a defect image $\mathbf{x}_{\text{defect}} = G(\mathbf{x}_{\text{clean}}, \mathbf{z})$. The discriminator D aims to distinguish between real and generated defects. The objective function includes adversarial and reconstruction losses:

$$\min_G \max_D \mathcal{L}_{\text{adv}}(G, D) + \lambda \mathcal{L}_{\text{recon}}(G),$$

where $\mathcal{L}_{\text{adv}}$ is the standard GAN loss, and $\mathcal{L}_{\text{recon}} = \|\mathbf{x}_{\text{clean}} - G(\mathbf{x}_{\text{clean}}, \mathbf{z})\|_1$ enforces similarity in non-defective regions.

Components:

- **Generator:** Synthesizes realistic defect textures and integrates them naturally into the background.
- **Discriminator:** Evaluates the realism of generated defects, encouraging high-fidelity synthesis.
- **Noise Injection:** A latent variable to introduce randomness and variability in defect patterns.
- **Masking Mechanism:** Controls the spatial placement of defects in images.

Final Goal: The ultimate aim of DefectGAN is to generate diverse and realistic defect images to augment limited defective datasets, thereby improving the generalization and robustness of defect detection models in real-world industrial applications.

Advantages:

- Enables effective data augmentation for rare defects.
- Generates spatially controllable and realistic defect textures.
- Enhances the training of downstream detection or classification networks.

Disadvantages

- Requires careful tuning to avoid unrealistic or overly repeated defect patterns.
- Generated data may introduce bias if not representative of real-world defect distribution.
- Training instability common to GAN-based approaches.

Variational Autoencoders (VAE)

Variational Autoencoders (VAEs) are probabilistic generative models that combine principles from deep learning and Bayesian inference. Introduced by Kingma and Welling (2014), VAEs aim to learn a latent representation of data while being able to generate new, similar samples by sampling from the latent space.

Components of VAE

A VAE consists of two neural networks:

- **Encoder (Recognition Model)**: Maps an input x to a distribution over latent variables z, typically a Gaussian. Denoted as $q_\phi(z|x)$, parameterized by ϕ.
- **Decoder (Generative Model)**: Reconstructs the input from a latent variable z. Denoted as $p_\theta(x|z)$, parameterized by θ.

Unlike standard autoencoders, VAEs impose a probabilistic structure on the latent space and optimize a variational lower bound.

Generative Process

Assuming a prior distribution $p(z)$ (usually standard normal $\mathcal{N}(0, I)$), the generative process is:

$$z \sim p(z), \quad x \sim p_\theta(x|z)$$

Objective Function: The ELBO

The log-likelihood of the data x is intractable, so VAEs optimize the Evidence Lower Bound (ELBO):

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x)\|p(z))$$

This consists of two terms:

- **Reconstruction Loss:** Measures how well the decoder reconstructs x from z.

$$\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$$

- **KL Divergence:** Regularizes the latent space by penalizing divergence between the approximate posterior and the prior.

$$D_{\mathrm{KL}}(q_\phi(z|x)\|p(z))$$

Reparameterization

To allow backpropagation through stochastic sampling, the latent variable is reparameterized as:

$$z = \mu(x) + \sigma(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

Here, $\mu(x)$ and $\sigma(x)$ are outputs of the encoder, and ϵ introduces stochasticity.

Final Loss Function

The total loss to be minimized is:

$$\mathcal{L}(\theta, \phi; x) = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + D_{\mathrm{KL}}(q_\phi(z|x)\|p(z))$$

Advantages

- Latent space is continuous and structured, useful for interpolation and generation.
- Tractable and efficient training using gradient-based optimization.
- Principled probabilistic framework with uncertainty estimation.

Disadvantages

- Generated samples are often blurrier compared to GANs.
- Requires a balance between reconstruction and regularization (KL divergence).
- ELBO is only a lower bound — may lead to loose approximations.

Diffusion Models

Diffusion models are a class of generative models that learn to generate data by reversing a gradual noising process. Inspired by non-equilibrium thermodynamics, they model the data generation as a Markov chain of latent variables that progressively add and then remove noise from the data.

Originally introduced in *Sohl-Dickstein et al. (2015)* and popularized by *Ho et al. (2020)*, diffusion models have recently achieved state-of-the-art performance in high-resolution image synthesis.

Components

- **Forward Process (Diffusion)**: A fixed Markov chain that gradually adds Gaussian noise to data over T time steps.
- **Reverse Process (Denoising)**: A learned generative model that removes noise step-by-step to recover the data distribution.

Forward Process

Given a data point $x_0 \sim q(x_0)$, the forward diffusion process produces a sequence of latent variables $\{x_t\}_{t=1}^T$ via:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I})$$

Here, β_t is a small variance schedule (typically linearly increasing), and the full forward process is:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

Thanks to the Markov property and Gaussian structure, one can sample x_t directly from x_0:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$.

Reverse Process

The reverse process is also a Markov chain, parameterized by a neural network, and aims to model:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

In practice, most implementations fix Σ_θ and focus on learning the mean μ_θ or the noise term ϵ_θ directly.

Training Objective

Training minimizes the variational bound on the negative log-likelihood of the data:

$$\mathcal{L}_{\text{VLB}} = \mathbb{E}_{q(x_{0:T})}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]$$

After simplification and using a reparameterization trick, the practical training loss becomes:

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

where:

- $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$
- $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
- $\epsilon_\theta(x_t, t)$ is the predicted noise by the neural network.

Sampling (Generation)

To generate data, one starts from Gaussian noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and applies the reverse process step-by-step to get x_0.

Advantages

- Capable of generating high-quality, diverse samples.
- Stable training compared to GANs.
- Probabilistic foundation with likelihood estimates.

Disadvantages

- Slow sampling due to many sequential steps (typically hundreds to thousands).
- Requires careful tuning of noise schedules and model architecture.
- High computational cost for both training and inference.

Stable Diffusion

Stable Diffusion is a type of latent diffusion model (LDM) developed by the CompVis team. It leverages the power of diffusion models for high-quality image synthesis while addressing their computational inefficiency. The key idea is to train the diffusion model in a lower-dimensional latent space instead of pixel space.

Key Components

- **VAE (Autoencoder):** Used to encode high-dimensional images into a lower-dimensional latent space $z = \text{Enc}(x)$ and decode back $x = \text{Dec}(z)$.
- **U-Net Backbone:** Operates on the latent space z during the diffusion process. The U-Net denoises latent variables over time steps t.
- **Text Encoder (e.g., CLIP or BERT):** Converts the input text prompt c into an embedding that conditions the diffusion process via cross-attention layers.
- **Scheduler:** Controls the forward and reverse noise schedule (e.g., linear, cosine, or DDIM).

Latent Diffusion Process

Instead of diffusing pixel space images x, Stable Diffusion applies the diffusion process to the latent variables z obtained from an autoencoder:

$$z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

A conditional U-Net model $\epsilon_\theta(z_t, t, c)$ is trained to predict the noise ϵ added at time t given the text condition c.

Training Objective

The simplified loss function (denoising score matching) is:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{z_0,\epsilon,t,c}\left[\|\epsilon - \epsilon_\theta(z_t, t, c)\|^2\right]$$

Where:

- $z_0 = \text{Enc}(x)$ is the latent of the real image.
- z_t is the noisy version of z_0 at timestep t.
- c is the conditioning embedding from the text encoder.

The model learns to recover z_0 from z_t across all timesteps t.

Sampling (Generation)

To generate an image from a prompt:

1. Encode the prompt into text embedding c.
2. Sample Gaussian noise $z_T \sim \mathcal{N}(0, I)$.
3. Iteratively denoise using the learned model:
$$z_{t-1} = \text{ReverseStep}(z_t, t, c)$$
4. Decode the final latent z_0 using the decoder: $x = \text{Dec}(z_0)$.

Optimized samplers like DDIM or DPM are used to reduce the number of steps from thousands to as few as 20–50.

Advantages

- Efficient training and inference by operating in latent space.
- Supports powerful text-to-image generation via cross-attention.
- Open-source and scalable (unlike proprietary models).
- Modular architecture allows plug-and-play conditioning (e.g., ControlNet).

Disadvantages

- Still slower than GANs for real-time applications.
- Requires high compute resources (GPUs with large memory) during training.
- Output quality is sensitive to prompt phrasing and tokenization.

ControlNet

ControlNet is an advanced neural network architecture designed to augment pre-trained diffusion models by enabling more precise and controllable image generation. The primary motivation behind ControlNet is to address the limitations of existing diffusion models, which often lack mechanisms to enforce spatial conditions or external guidance during the image synthesis process. In tasks such as image-to-image translation, pose-guided synthesis, and depth-to-image generation, ControlNet provides a mechanism to incorporate structured conditioning inputs (e.g., edges, segmentation maps, depth maps) into the generative pipeline.

Objective Function: ControlNet builds upon the standard denoising diffusion probabilistic model (DDPM) objective. Let $\mathbf{x}_0$ denote the original data, and $\mathbf{x}_t$ the noisy version at timestep t. The training objective is typically formulated as:

$$\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{\mathbf{x}_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c})\|^2\right],$$

where ϵ is Gaussian noise, ϵ_θ is the model prediction, and $\mathbf{c}$ is the conditioning input. ControlNet introduces a duplicate copy of the original network, allowing gradients to flow through both the main network and the control branch. The additional conditioning is injected in a way that does not interfere with the original network's learned weights, ensuring stability and flexibility.

Components: ControlNet consists of the following key components:

- **Frozen Pretrained Diffusion Model:** A stable backbone that maintains the quality of generation.
- **Trainable Control Branch:** A copy of the backbone, initialized identically but trained to process and inject the conditional information.
- **Zero Convolution Layers:** Used to regulate the contribution of the control signal, initialized to output zero and trained gradually.

- **Control Inputs:** Structured inputs such as canny edges, pose estimations, semantic maps, etc.

Final Goal: The overarching goal of ControlNet is to enable high-fidelity, high-resolution image synthesis that adheres to user-provided guidance signals, while maintaining the diversity and realism associated with diffusion models. This allows users to precisely control the content and structure of generated images, enabling applications in art generation, human pose-guided synthesis, and more.

Advantages:

- Enables precise control over image generation using various structured inputs.
- Maintains the quality and diversity of outputs from pre-trained diffusion models.
- Flexible integration with different types of conditions (e.g., edges, depth, sketches).
- Efficient training by freezing the base model and updating only the control branch.

Disadvantages:

- Increased memory and computation due to the dual-branch architecture.
- Requires task-specific control data, which may not always be available.
- May suffer from overfitting or reduced diversity if the control is too strong or poorly aligned.

Overall, ControlNet represents a significant advancement in controllable image generation, offering a practical and scalable solution for incorporating external guidance into powerful generative models.

Evaluation Metrics

To evaluate the performance and quality of generative models, several quantitative metrics are employed. These metrics assess fidelity, perceptual similarity, and structural accuracy between generated and reference images.

PSNR (Peak Signal-to-Noise Ratio)

PSNR measures the ratio between the maximum possible pixel value and the magnitude of reconstruction error. It is widely used to quantify the quality of image reconstruction or generation. Mathematically, it is defined as:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{\text{MSE}}\right),$$

where MAX_I is the maximum possible pixel value (typically 255), and MSE is the mean squared error between the generated and ground truth images. Higher PSNR values indicate better quality and lower distortion.

SSIM (Structural Similarity Index Measure)

SSIM measures the structural similarity between two images by comparing their luminance, contrast, and structural features. It is defined as:

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

where μ_x and μ_y are the means, σ_x^2 and σ_y^2 are the variances, and σ_{xy} is the covariance of images x and y. The constants C_1 and C_2 stabilize the division. SSIM values range from 0 to 1, with 1 indicating perfect structural similarity.

LPIPS (Learned Perceptual Image Patch Similarity)

LPIPS is a perceptual metric that compares deep feature representations extracted from pretrained neural networks. Unlike PSNR or SSIM, LPIPS aligns more closely with human visual perception. A lower LPIPS score indicates that the generated image is perceptually closer to the reference image. It is particularly useful when small pixel-wise errors do not affect visual quality significantly.

FID (Fréchet Inception Distance)

FID evaluates the quality of generated images by comparing the distributions of real and generated image features extracted using the Inception network. It computes the Fréchet distance between two multivariate Gaussians:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}),$$

where (μ_r, Σ_r) and (μ_g, Σ_g) are the mean and covariance of features from real and generated images, respectively. Lower FID values indicate better generation quality and diversity.

Results

This section presents the qualitative and quantitative evaluation of generative models. The original dataset is shown, followed by visual results from each model. Finally, we compare models using various evaluation metrics.

Original Dataset Samples

Figure 1 shows representative samples from the original dataset used for training and evaluation.

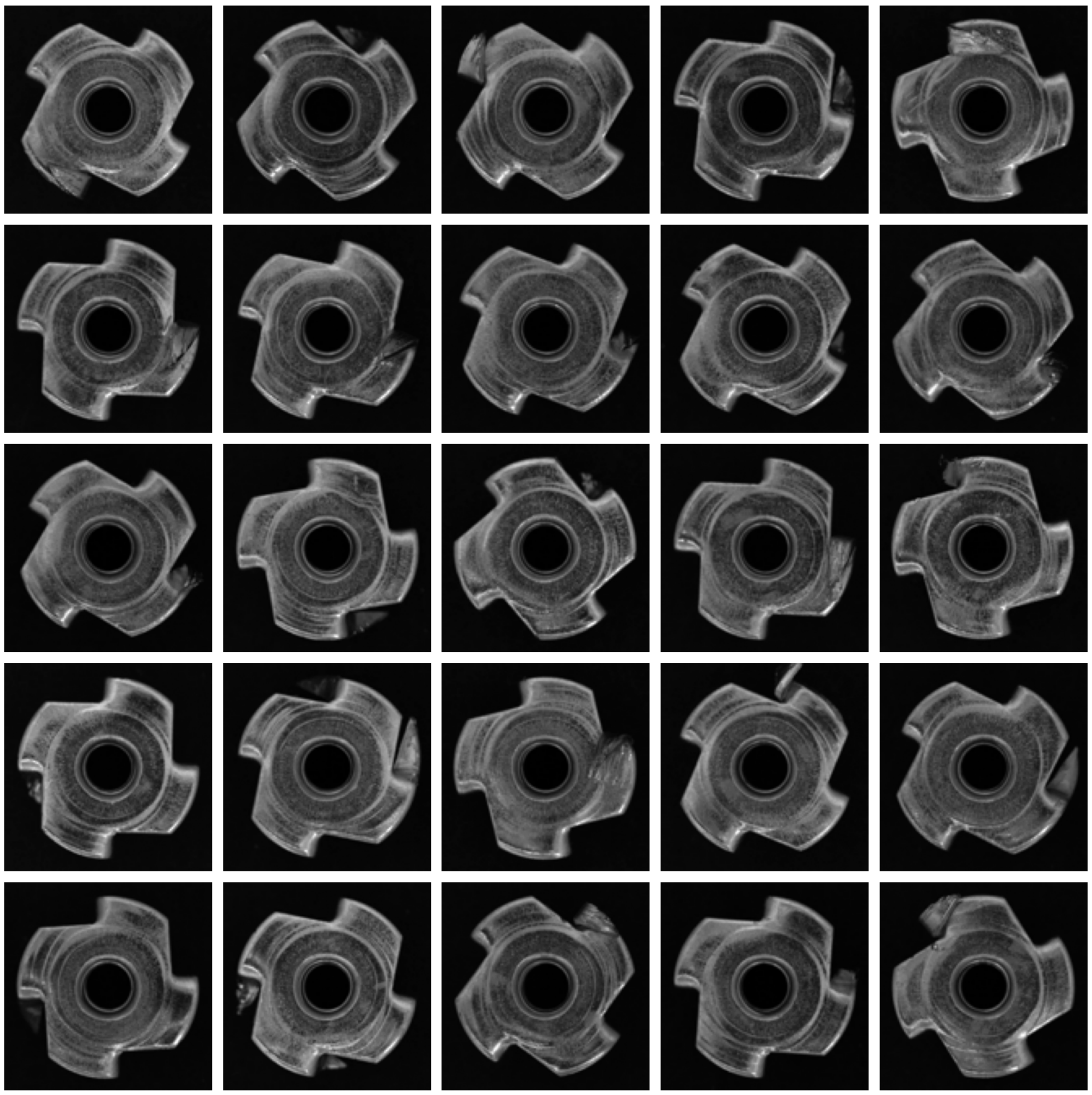

Figure 1: Sample images from the original dataset.

Model Outputs

Generative Adversarial Networks (GAN)

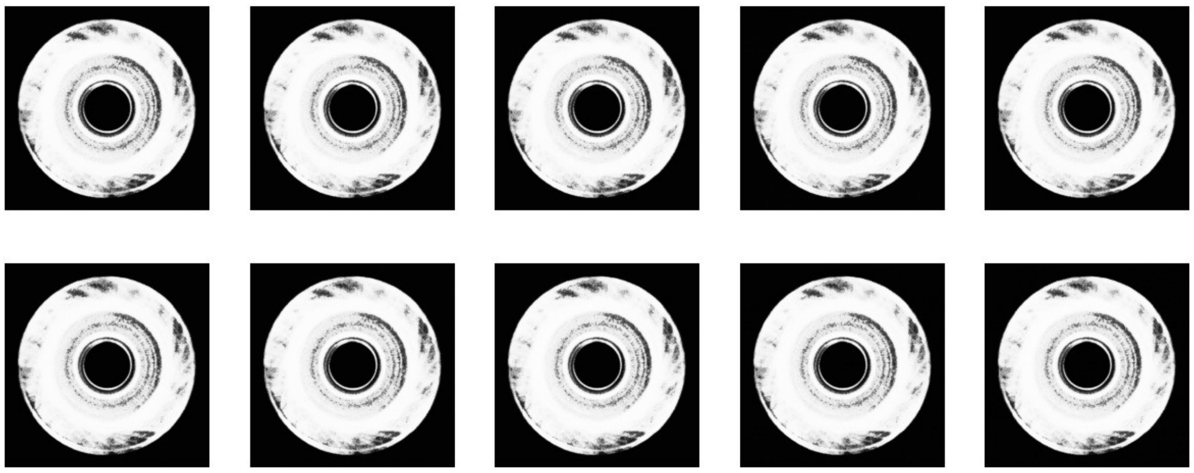

Figure 2: Generated images using GAN.

StyleGAN

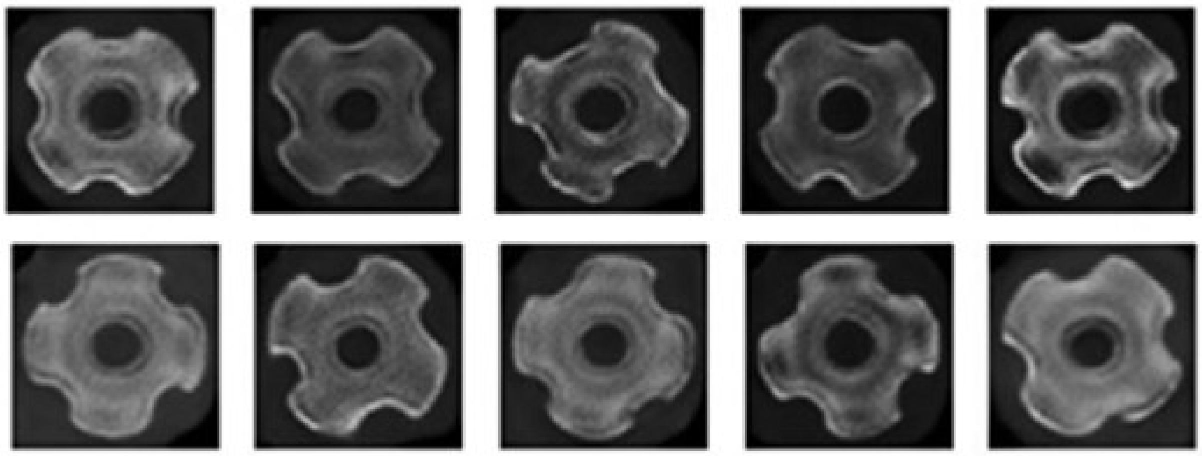

Figure 3: Generated images using Diffusion Models.

DefectGAN

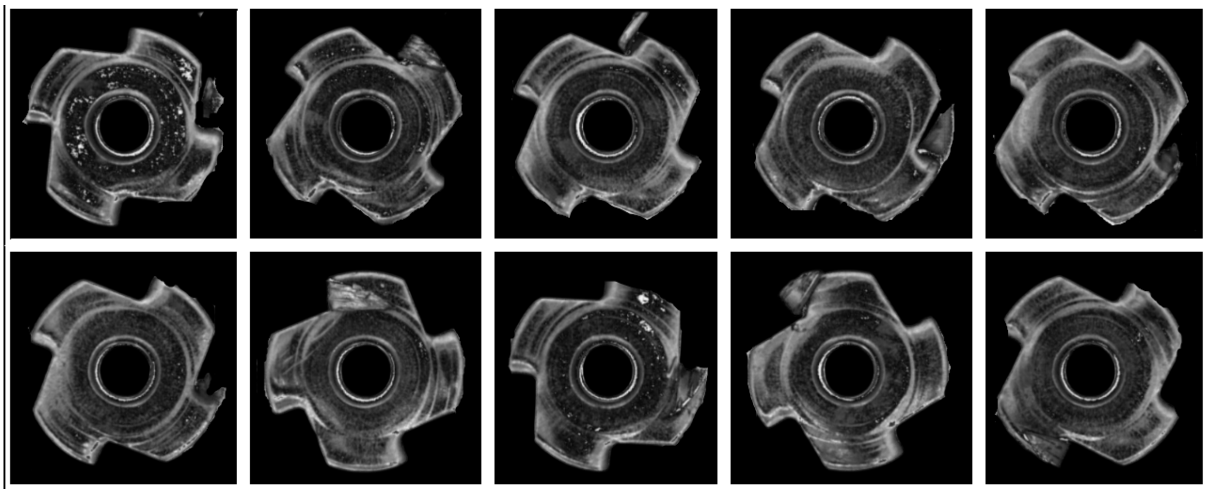

Figure 4: Generated images using Stable Diffusion.

ControlNET

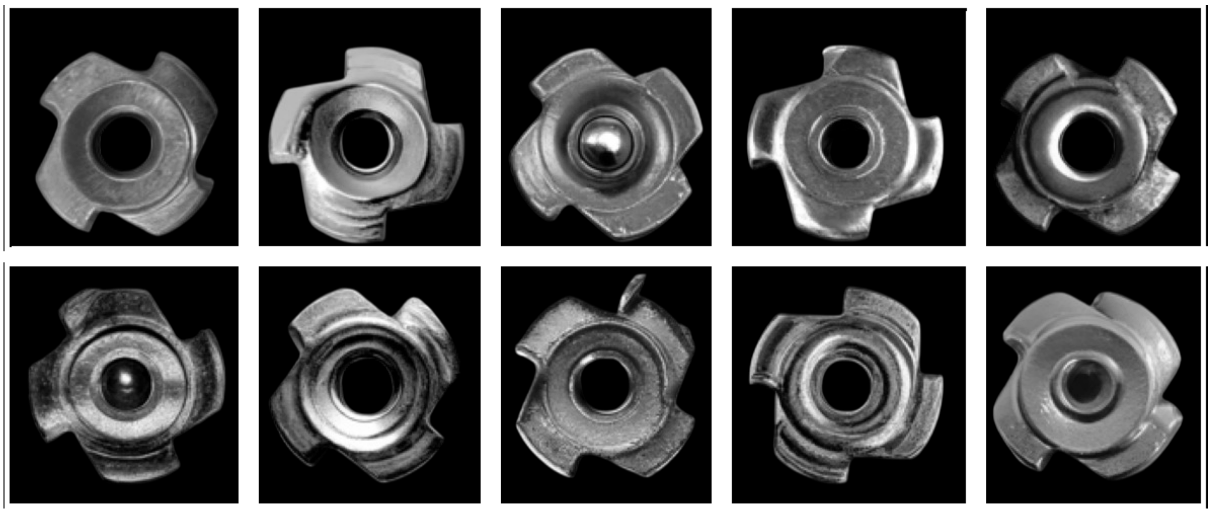

Figure 5: Generated images using ControlNET.

Quantitative Evaluation

Model	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Standard GAN	17.2 ± 2.5	0.62 ± 0.07	0.55 ± 0.07	82.4 (65 − 98)
StyleGAN	22.5 ± 1.8	0.78 ± 0.04	0.32 ± 0.05	45.1 (36 − 58)
DefectGAN	25.8 ± 1.2	0.85 ± 0.03	0.18 ± 0.03	20.7 (15 − 28)
ControlNet	**27.3** ± 1.0	**0.89** ± 0.02	**0.14** ± 0.02	**15.2** (11 − 21)

Table 1: Quantitative evaluation of generative models.

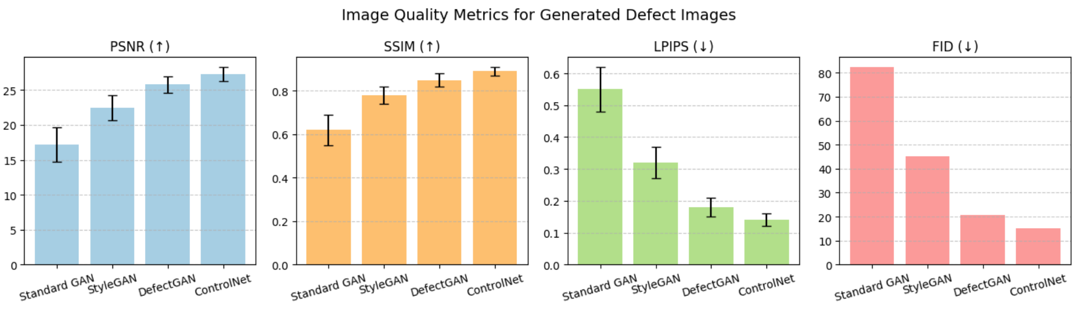

Figure 6: Bar charts comparing evaluation metrics across models.

Conclusion

The experimental results clearly demonstrate that ControlNET achieves the best overall performance across all evaluation metrics, attaining the highest PSNR (27.3) and SSIM (0.89), alongside the lowest LPIPS (0.14) and FID (15.2). This indicates that ControlNET not only produces outputs with superior perceptual quality but also preserves structural similarity more effectively than competing models. DefectGAN ranks second, with strong scores in both PSNR (25.8) and SSIM (0.85) and comparatively low LPIPS (0.18) and FID (20.7), showing its capability in generating high-quality results, albeit slightly behind ControlNET. In contrast, GAN and StyleGAN lag significantly behind in all metrics, reflecting lower reconstruction accuracy and perceptual realism. Overall, ControlNET emerges as the most effective model, followed by DefectGAN, with GAN and StyleGAN performing less favorably.

Tools Used

The following tools and frameworks were used in the development and implementation of this project:

- **Python** – Core programming language used for model development and experimentation.
- **PyTorch** – Deep learning framework used to implement and train generative models.
- **NumPy** – Library for numerical computations and array manipulations.
- **OpenCV** – Library for computer vision and image preprocessing.
- **Pillow** – Python library for image loading, processing, and format conversion.
- **Matplotlib** – Visualization library used to plot results and evaluation metrics.

Bibliography

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. arXiv:1406.2661.

[2] Zongwei Zhou, et al. DefectFill: Defect-Guided Image Completion for Industrial Anomaly Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. CVPR 2021 PDF.

[3] Andrew Brock, Jeff Donahue, Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1809.11096.

[4] Tero Karras, Samuli Laine, Timo Aila. Analyzing and Improving the Image Quality of StyleGAN. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. arXiv:1912.04958.

[5] Jake Snell, et al. Learning to Generate Images with Perceptual Similarity Metrics. 2015. arXiv:1511.06409.

[6] Prafulla Dhariwal, Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2105.05233.

[7] Lvmin Zhang, Maneesh Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models (ControlNet). 2023. arXiv:2302.05543.