

A Critical Evaluation of Diffusion Models: Exploring Their Limits in Open-World Anomaly Detection

Saksham Madan (Roll no. 24124040)

Kushal Srivastava (Roll no. 24124025)

Arnav Raghuvanshi (Roll no. 24124007)

Department of Mathematical Science
Indian Institute of Technology (BHU), Varanasi

Supervisor: Dr. Santwana Mukhopadhyay

Outline

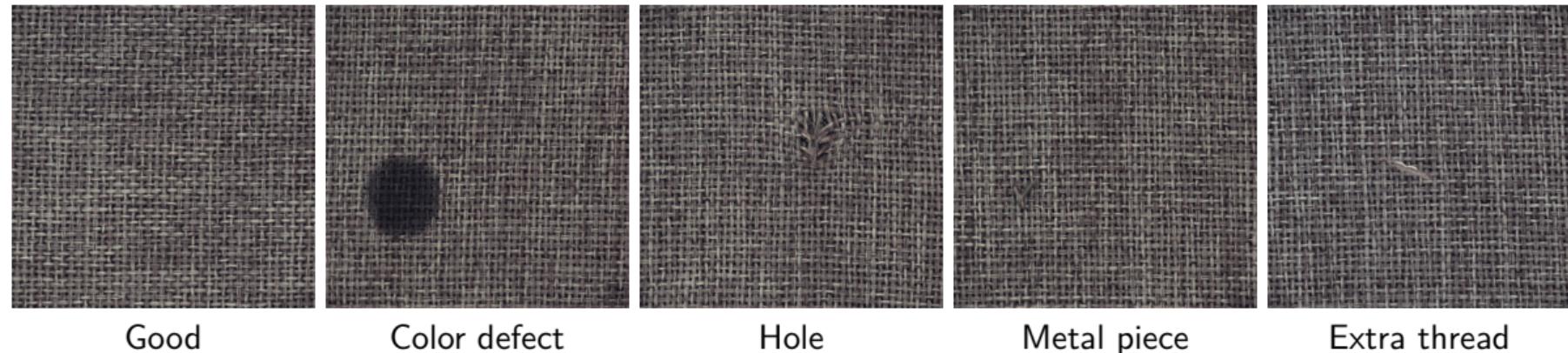
- 1 Introduction
- 2 Dataset and Research Design
- 3 Key Components
- 4 Methods
- 5 Experimental Setup
- 6 Results
- 7 Why It Fails
- 8 Practical Issues, Conclusion, and Future Work

Motivation

- Industrial inspection often deals with:
 - tiny scratches, small holes, colour shifts, loose threads, etc.
 - These are *local, high-frequency* defects on otherwise regular textures.
- Classical idea:
 - Learn what “normal” looks like,
 - Flag anything that deviates as an anomaly.
- Recent progress:
 - **Diffusion models** can generate extremely realistic images.
 - **CLIP** can compare images in a semantic feature space.
- Natural question:

Can we use these modern generative models to reconstruct a *clean* version of an image and detect anomalies from the differences?

MVTec AD Example (Carpet Category)



Good

Color defect

Hole

Metal piece

Extra thread

These defects are small, local perturbations on a largely repetitive background.

Reconstruction-Based Anomaly Detection

- Setup:
 - ① Train a model only on **normal** images.
 - ② Given a test image x , (maybe defective), ask the model for a reconstruction \hat{x} .
 - ③ Compare x and \hat{x} :
 - Large local differences \Rightarrow likely anomaly.
- Intuition:
 - The model learns the manifold of normal images.
 - Test images are projected back onto this manifold.
 - Anomalies are “pushed away” from the manifold and show up in the residual $x - \hat{x}$.
- Works relatively well with:
 - autoencoders, some GAN-based models.
- Our work: does this paradigm make sense with **latent diffusion + CLIP?**

Goals of This Project

- **Main goal:** critically evaluate latent diffusion models for open-world industrial anomaly detection.
- Sub-goals:
 - Test three practical pipelines built around Stable Diffusion:
 - ① Small UNet on top of SD latents,
 - ② SD img2img with prompts,
 - ③ LoRA fine-tuning on normal textures.
 - Combine pixel-based and CLIP-based scores into a hybrid anomaly score.
 - Analyse not just the numbers, but also:
 - reconstruction behaviour,
 - hyperparameter sensitivity,
 - and *why* the model behaves as it does.

- **MVTec Anomaly Detection** is a standard benchmark:
 - 15 categories: e.g. carpet, tile, wood, bottle, metal nut, etc.
 - Each category has:
 - many normal training images,
 - test images with annotated defects.
- Properties:
 - Defects are often small, local, and embedded in repetitive textures.
 - Good stress test for any reconstruction model:
 - It must preserve fine details,
 - but selectively remove irregularities.

- **Type:** comparative exploratory study.
- High-level plan:
 - ① Implement multiple reconstruction pipelines using latent diffusion.
 - ② For each pipeline:
 - generate reconstructions on MVTec test images,
 - compute anomaly scores (pixel + CLIP),
 - visualize heatmaps showing where the model “thinks” anomalies are.
 - ③ Compare with a CLIP feature-based baseline.
 - ④ Use mathematical and architectural reasoning to interpret:
 - why certain defects are missed,
 - why textures drift.

Implementation Details

- **Software**

- Python, PyTorch.
- HuggingFace Diffusers (Stable Diffusion, DDIM scheduler).
- OpenAI CLIP.

- **Compute**

- Kaggle GPU environment (limited VRAM).
- Required optimizations:
 - attention-slicing,
 - half-precision (fp16),
 - small batch sizes.

- **Models**

- Stable Diffusion v1.5,
- CLIP ViT-B/32 as image encoder.

- Forward process (noising):
 - Start from a clean sample x_0 ,
 - Iteratively add Gaussian noise:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I).$$

- For large t , x_t becomes nearly pure noise.
- Reverse process (denoising):
 - A neural network is trained to *predict the noise* at each step.
 - Informally: “Given a slightly noisy image, what clean image could have produced it?”
 - Sampling: start from noise, run the learned reverse steps to get a new image.
- Important: this reverse process is **generative**, not an exact inverse map.

Latent Diffusion (Stable Diffusion) – Details

- Directly applying diffusion in pixel space is expensive.
- Latent diffusion uses a VAE:
 - Encoder: $\text{Enc}(x) = z$ (compressed latent representation).
 - Decoder: $\text{Dec}(z) \approx x$.
- Diffusion runs on z instead of x :
 - Much smaller dimensionality,
 - Faster and cheaper.
- But:
 - The VAE encoding is *lossy*.
 - High-frequency details (tiny scratches, small holes) are partially discarded.
 - This is the **information bottleneck** of latent diffusion.

CLIP – A Semantic Distance Between Images

- CLIP has:
 - an image encoder $f(x)$,
 - a text encoder $g(t)$,
 - both map into a shared feature space.
- Training idea:
 - Pairs (x_i, t_i) that belong together should have high similarity,
 - Mismatched pairs should have low similarity.
- In our use:
 - We focus on $f(x)$, the image embeddings.
 - Given input x and reconstruction \hat{x} :

$$\text{CLIP distance} = 1 - \frac{\langle f(x), f(\hat{x}) \rangle}{\|f(x)\| \|f(\hat{x})\|}.$$

- If CLIP believes they are semantically different, this distance is large.

LoRA – Parameter-Efficient Fine-Tuning

- Full fine-tuning of a diffusion model is expensive (millions of parameters).
- LoRA modifies each weight matrix W_0 by a low-rank correction:

$$W = W_0 + BA, \quad B \in \mathbb{R}^{d \times r}, \quad A \in \mathbb{R}^{r \times k}, \quad r \ll \min(d, k).$$

- Interpretation:
 - W_0 encodes general knowledge,
 - BA encodes a small adaptation to a new domain (here: clean industrial textures).
- For us:
 - LoRA is applied to the diffusion UNet,
 - The VAE encoder/decoder remain unchanged,
 - So the fundamental compression bottleneck still exists.

Three Reconstruction Pipelines

- **1. Small UNet on SD Latents**

- Take $z = \text{Enc}(x)$ from Stable Diffusion.
- Add a smaller UNet that tries to “repair” z before decoding.
- Hope: recover some of the lost fine texture.

- **2. Stable Diffusion img2img**

- Start from the encoded version of x ,
- Add noise controlled by a parameter called *strength*,
- Run the diffusion process with a prompt like “a flawless carpet”,
- Try to keep the structure while removing defects.

- **3. LoRA Fine-Tuning**

- Train a small LoRA module on normal images,
- Encourage the model to generate the typical clean texture of each category.

Hybrid Anomaly Scoring

- Given input x and reconstruction \hat{x} , we compute:
 - Pixel error: $\text{MSE}(x, \hat{x})$,
 - CLIP distance: $1 - \cos(f(x), f(\hat{x}))$.
- Hybrid score:

$$S_{\text{hybrid}}(x) = \alpha \cdot \text{MSE}(x, \hat{x}) + (1 - \alpha) \cdot (1 - \cos(f(x), f(\hat{x}))),$$

with $\alpha \in [0, 1]$.

- Interpretation:
 - Pixel term: sensitive to small local changes.
 - CLIP term: sensitive to semantic/texture-level changes.
 - Tuning α changes how “strict” we are about pixel vs semantic alignment.

- **Diffusion Hyperparameters**

- Strength: 0.1–0.7 (amount of noise added in img2img).
- Guidance scale: 3.5–9 (how strongly we enforce the text prompt).
- Steps: 20–50 (number of denoising steps).

- **Scoring Hyperparameters**

- α : 0.3–0.9 (balance between pixel and CLIP contributions).

- **Evaluation Metrics**

- AUROC (normal vs anomalous),
- MSE, SSIM (reconstruction quality),
- CLIP cosine similarity,
- Qualitative heatmaps of $|x - \hat{x}|$.

Summary of Experimental Findings

- **Small UNet on SD Latents**

- Reconstructions remained blurry,
- High-frequency textures lost in the VAE encoding were not recovered.

- **Stable Diffusion img2img**

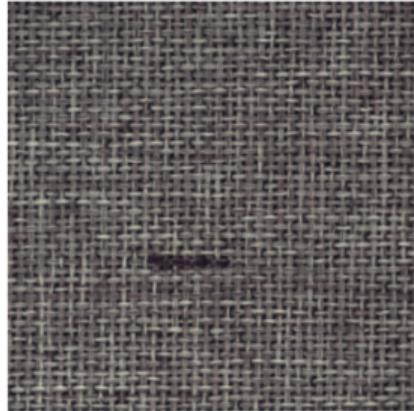
- Low strength:
 - The reconstruction is almost identical to the input,
 - Defects remain.
- High strength:
 - Defects vanish,
 - But textures and sometimes shapes change significantly.

- **LoRA Fine-Tuning**

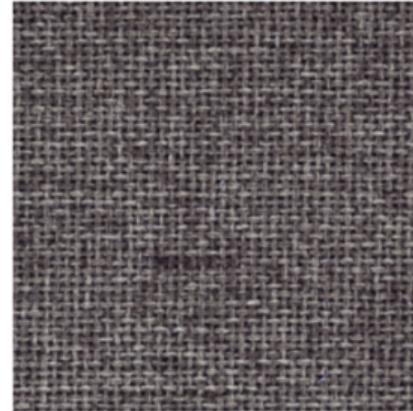
- Some defects are softened or removed,
- But we see colour shifts and style changes (identity drift).

Effect of Diffusion Strength

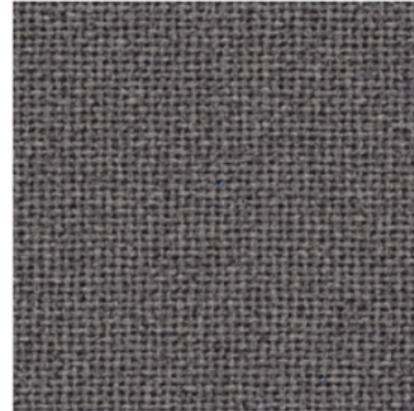
Sample 0: Original



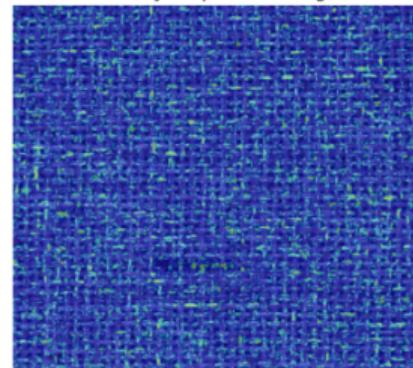
Low strength = 0.3



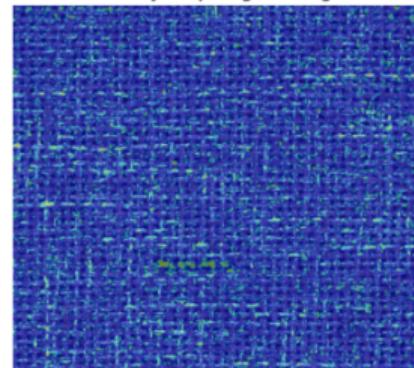
High strength = 0.5



Anomaly map (low strength)



Anomaly map (high strength)



Quantitative Metrics and Their Limitations

- AUROC values sometimes looked “acceptable”.
- However, deeper inspection showed:
 - Good AUROC sometimes came from global style changes,
 - Not from precise removal of defects on an otherwise unchanged background.
- In other words:

The model was often correct for the wrong reasons: it changed the whole texture, which correlated with anomalies, but did not give faithful reconstructions.

Information Bottleneck in the VAE

- Stable Diffusion uses $x \mapsto z = \text{Enc}(x)$, z is much smaller than x .
- Approximate decoding: $x \approx \text{Dec}(z)$.
- The reconstruction error $x - \text{Dec}(\text{Enc}(x))$ is not negligible:
 - It contains high-frequency texture details,
 - Exactly the information that distinguishes small defects from normal patterns.
- If an anomaly is smaller than this “VAE noise floor”:

It gets washed out before the diffusion process even starts.

Small Defects vs Model Variance

- Suppose $x_{\text{anom}} = x_{\text{norm}} + \delta$, with δ a tiny defect.
- The model's own reconstruction noise and randomness is larger than δ :

$$\|x_{\text{anom}} - \hat{x}\| \approx \text{model drift} \gg \|\delta\|.$$

- Then:
 - The residual $x - \hat{x}$ is dominated by generative drift,
 - Not by the anomaly.
- This makes it hard to interpret high residuals as “true anomalies”.

Generative vs Invertible

- The reverse diffusion is trained to approximate:

$$p_{\theta}(x_0 \mid x_T = \text{noise}),$$

i.e. the distribution of plausible clean images.

- It is *not* trained to invert a specific x .
- In practice:
 - Small changes in random seed or guidance lead to noticeably different images.
 - So the map $x \mapsto \hat{x}$ is not a stable, near-identity map.
- For anomaly detection we would like:
 - \hat{x} to be exactly x minus the defect,
 - but diffusion gives us “another clean example from the same class”.

Contrast With Autoencoders

- Autoencoders are trained with an objective of the form:

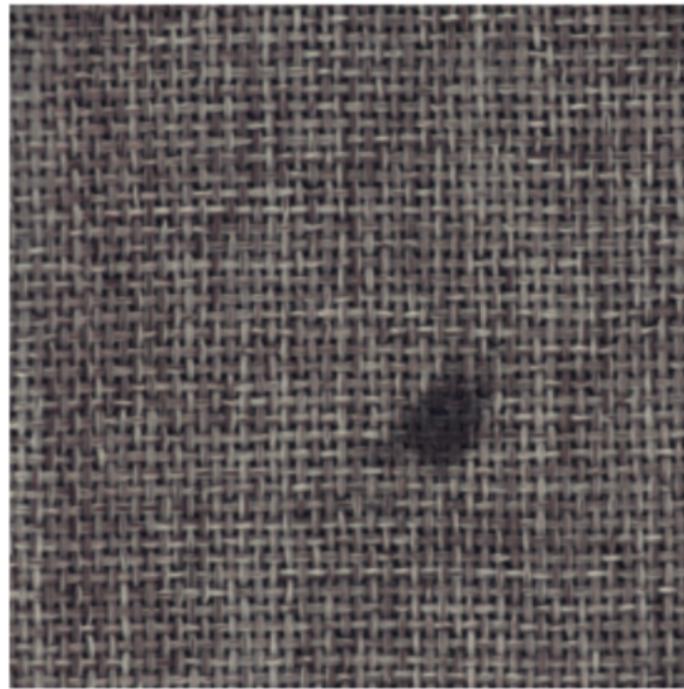
$$\min \|x - \text{AE}(x)\|^2,$$

which explicitly encourages identity-preserving reconstruction.

- Diffusion models minimize a denoising loss, not a direct reconstruction loss in pixel space.
- As a result:
 - Autoencoders are biased toward copying the input (on normal data),
 - Diffusion models are biased toward sampling from the learned distribution.
- For reconstruction-based anomaly detection, this mismatch is fundamental rather than just an implementation detail.

Identity Drift – Visual Example

Original



Reconstruction



Output looks plausible and defect-free, but the exact texture and structure have changed: the identity of

- **Compute constraints**
 - Kaggle GPUs with limited VRAM,
 - Out-of-memory issues for large models and high resolution.
- **Approximate inversion**
 - Methods like DDIM inversion approximate the latent corresponding to x ,
 - Not a mathematically exact inverse map, which adds additional drift.
- These limitations influenced:
 - how many experiments we could run,
 - and how thoroughly we could sweep hyperparameters.

Conclusion

Main Conclusion

Latent diffusion models (Stable Diffusion family) are **not well-suited** for reconstruction-based industrial anomaly detection that requires identity-preserving, pixel-accurate results.

- VAE compression removes the very fine-scale information we care about.
- The diffusion reverse process is generative, not a stable inverse of x .
- LoRA and CLIP improve scoring and texture adaptation, but do not remove the fundamental bottlenecks.
- Numeric metrics alone (like AUROC) can be misleading if we ignore these structural limitations.

Future Work

- Implement and compare with stronger baselines:
 - PatchCore, WinCLIP, and related methods.
- Explore pixel-space diffusion with more reconstruction-oriented objectives.
- Combine CLIP features with discriminative models:
 - e.g. training a classifier or segmenter on CLIP embeddings.
- Investigate invertible architectures:
 - Normalizing flows specifically tailored to industrial textures.

Project Artifacts

- Full written report.
- This slide deck summarizing the project.
- Experiment code and Kaggle scripts.
- Public repository:
 - github.com/s4kr3d-w0r1d/exploratory_project

The End

Thank You!