InvAD: Inversion-based Reconstruction-Free Anomaly Detection with Diffusion Models

Supplementary Material

This document supplements our main paper, "InvAD: Inversion-based Reconstruction-Free Anomaly Detection with Diffusion Models". The reference indices are consistent with those used in the main paper.

7. Dataset Details

We utilize four publicly available anomaly detection (AD) benchmarks for evaluation: MVTecAD [28], VisA [29], MPDD [30], and BMAD [31]. Table 10 summarizes the key statistics of the datasets used in our experiments, including the number of classes, domain, image resolution, and sample counts for training and testing. All of these datasets follow an unsupervised AD setting. In other words, the training set only consists of defect-free normal images, and the test set consists of both normal and anomalous images.

8. Implementation Details

Our implementation is available at: https://github.com/SkyShunsuke/InversionAD.

8.1. Data Preparation

We resize all training images to 256×256 pixels and do not apply any data augmentation. Image normalization is performed using the ImageNet dataset's channel-wise mean ([0.485, 0.456, 0.406]) and standard deviation ([0.229, 0.224, 0.225]).

8.2. Model Setup

We adopt the original DiT architecture [32] as the backbone of our diffusion modeling, but remove all class-conditioning so that our method does not depend on class labels. This design allows DiT to be applied in scenarios where ground-truth annotations are unavailable or impractical (*e.g.*, inspection of diverse products on a manufacturing line). We set the number of transformer blocks and embedding dimensions to 16 and 2048 for DiT-gigant, and to 8 and 1024 for DiT-base, respectively. For the UNet-based diffusion models evaluated in Table 8 (in the main paper), we follow the architectural specifications of the original DDPM implementation [10], consistent with prior work [5, 8]. For the MLP-based diffusion models, we stack multiple MLP blocks and apply AdaLN-Zero conditioning [32], as in [45]. We summarize all major hyperparameters for diffusion modeling in Table 11.

Table 10. **Dataset statistics** used in our experiments. 'Res.' represents image resolution, where $H = W$. 'N' and 'A' denote the numbers of normal and anomalous samples, respectively.

	MVTecAD	**VisA**	**MPDD**	**BMAD**
Classes	15	12	6	6
Domain	Industrial	Industrial	Metal	Medical
Res.	1024	256	1024	1024
Train (N)	2,629	8,659	888	52742
Test (N)	467	962	176	4540
Test (A)	1,258	1,200	282	22632

8.3. Backbone Setup

For fair comparisons with previous works [7, 34], we employ pre-trained EfficientNet-B4 [33] as our feature extractor. To capture anomalies at different scales, we first extract feature maps from the 1st to the 4th blocks from EfficientNet-B4, and then apply bilinear interpolation followed by channel-wise concatenation, creating $\mathbf{z} \in \mathbb{R}^{272\times16\times16}$, as in [7, 34].

We evaluate in Table 7 in the main paper the robustness of our method to the different feature space encoding modules: ViT-B [43] and DINO-base [44]. Both modules employ the vision transformer-based architecture and use ImageNet-1k as the pre-training dataset. The only difference comes from the pre-training scheme (*i.e.*, supervised for ViT-B [43], and self-distillation for DINO-base [44]). For both models, we extract patch tokens of the last block and reorder them to construct feature maps, creating $\mathbf{z} \in \mathbb{R}^{768\times16\times16}$.

8.4. Evaluation Pipeline

We evaluate AD performance at both image- and pixel-levels using the following metrics:

- **Area Under the Receiver Operating Characteristic Curve (AU-ROC)** provides an intuition on how well the model takes a trade-off between recall and precision in a threshold-independent manner.
- **Average Precision (AP)** conducts a more practical evaluation than AU-ROC, where the test set contains far fewer anomalous samples compared to normal ones.
- **Maximum F1-score ($\mathbf{F1_{max}}$)** measures the F1 score on the best-performed threshold, which reflects actual performance on the real problem.

To assess pixel-level anomaly localization, we employ:

- **Area Under the Per-Region-Overlap Curve (AU-PRO)**

Table 11. **Hyperparameters** of diffusion modeling.

Parameter	Value
Diffusion Settings	
Training diffusion steps (T)	1000
Inference diffusion steps (S)	3
Noise scheduler	Linear, $\beta_1 = 1 \times 10^{-4}$, $\beta_T = 0.02$
Parameterization	ϵ-prediction
Noise scale (σ)	Fixed
Training Settings	
Batch size	8
Optimizer	AdamW
Gradient clipping threshold	1.0
Initial learning rate	1×10^{-6}
Peak learning rate	5×10^{-5}
Final learning rate	5×10^{-6}
Learning rate scheduler	Warmup + cosine decay
Warmup epochs	40
Total number of epochs	300
Weight decay	0

Table 12. **Computing environment** in our experiments.

Component	Training	Evaluation
CPU	Xeon 8468 ×2	i9-14900KF
GPU	NVIDIA H200 ×8	RTX 4090 (24GB)
RAM	1.0 TiB (≈1024GB)	31 GB
OS	RHEL 9.4 (Plow)	Ubuntu 24.04.2 LTS

calculates the IoU between prediction and anomalous region, which is more robust to the small anomalies [28].

8.5. Experimental Environment

Our implementations are based on Pytorch-2.7.1 [49] and Python-3.10. We show the machine specification for the training and evaluation in our experiments in Table 12.

To ensure reproducibility, we fixed random seeds for all libraries (NumPy, PyTorch, etc.) and disabled nondeterministic GPU operations if possible.

It is important to understand that we do not train any extra conditioning modules nor perform reconstruction, just train unconditional diffusion models in the standard way. This leads to significant training time reduction (e.g., for MVTecAD training time, InvAD: 2.5h on 8×H200 @28 GB/GPU— 2 × faster than DiAD [5]'s 5h). The use of 8× H200 is optional, *not required*. Specifically, training can be performed on GPUs with more than 32 GB of memory for our largest models.

Table 13. **Inversion schedule ablation** under different total diffusion steps S in multi-class MVTecAD with mAD.

$g(u)$, $u = t/T$	$S = 3$	$S = 10$	$S = 100$
Uniform $g(u) = u$	**83.35**	**82.11**	**79.35**
Quad $g(u) = u^2$	82.37	80.76	79.14
Cube $g(u) = u^3$	81.68	80.84	79.18
Exp $g(u) = \frac{e^{5u}-1}{e^5-1}$	81.48	80.85	79.23

Table 14. **Feature resolution ablation** under different anomaly sizes in multi-class MVTecAD with mAD.

$h \times w$	**Tiny**	**Small**	**Medium**	**Large**	**All**
8×8	73.38	76.38	79.17	83.57	82.05
16×16	**78.54**	**80.65**	**81.76**	**84.94**	**83.70**
24×24	76.33	78.76	80.00	81.97	81.44

9. Addtional ablations

Inversion schedule. To accelerate inversion computation, we uniformly set $S = 3$ to skip intermediate steps within the original $T = 1000$ steps, and create the subset of timesteps $\tau_3 = [333, 666, 999]$ (*i.e.*, uniform scheduler). We include an ablation by varying the scheduling policy $g(u)$ for examining the impact of alternative schedules. The numerical results are shown in Table 13. We can see that the accuracy across all schedules is mostly consistent, with the uniform schedule performing best for S=3. We thus adopt this commonly used uniform schedule in our implementation.

Feature resolutions. To utilize the high-level semantic features of backbones, we follow the latent diffusion manner by extracting the feature maps $\mathbf{z} \in \mathbb{R}^{C \times h \times w}$ and operating diffusion modeling on $\mathbf{z}$. While this contributes to the improvements in both accuracy and inference speed, backbones may overly compress the anomalous information, especially for small $h \times w$. In Table 14, we conduct a design ablation on feature resolution ($h \times w$) and report MVTecAD mAD by different anomaly sizes (anomalous pixel ratio) for inspecting optimal feature resolution. The default 16×16 is best overall (83.70) and on Tiny/Small (78.54/80.65), exceeding both 8×8 and 24×24 settings. This suggests performance is not simply limited by upsampling, and increasing resolution is not monotonically beneficial; there is an optimal resolution–efficiency balance.

10. Extended Background of Diffusion Models

10.1. Denoising Diffusion Probabilistic Models

Denoising diffusion probabilistic models (DDPM) [10] are a class of latent generative models which involve progressively collapsed T random variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$. Let us assume we have *i.i.d.* data samples $\mathcal{X} = \{\mathbf{x}_0 \mid \mathbf{x}_0 \sim q_0(\mathbf{x})\}$

where $q_0(\mathbf{x})$ is an unknown data distribution. DDPM aims to approximate this data distribution by tracing the data collapsing process (forward process) in reverse order. In general, the forward process $q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)$ is represented as a Markov process with the Gaussian transition kernel:

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}), \tag{9}$$

$$\text{where} \quad q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}\Big(\sqrt{\frac{\alpha_t}{\alpha_{t-1}}}\,\mathbf{x}_{t-1}, (1 - \frac{\alpha_t}{\alpha_{t-1}})\mathbf{I}\Big). \tag{10}$$

The noise schedule is defined by the monotonically decreasing sequences $\{\alpha_t\}_{t=1}^T$ in the above equation. Notably, from the reproducibility of the Gaussian distribution, we have:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, (1 - \alpha_t)\mathbf{I}). \tag{11}$$

Since we can efficiently compute latent variables $\mathbf{x}_t$ given a clean sample $\mathbf{x}_0$, it provides a GPU-friendly diffusion model training. Also, we define the generative process $p_\theta(\mathbf{x}_{0:T})$ as the Markov process with a learnable Gaussian transition kernel:

$$p_\theta(\mathbf{x}_{0:T}) := p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta^{(t)}(\mathbf{x}_{t-1} \mid \mathbf{x}_t), \tag{12}$$

$$\text{where} \quad p_\theta^{(t)}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) := \mathcal{N}\big(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta^{(t)}(\mathbf{x}_t), \sigma_t^2\mathbf{I}\big). \tag{13}$$

In DDPM, the posterior mean is modeled using a noise prediction neural network $\epsilon_\theta^{(t)}(\mathbf{x}_t)$:

$$\boldsymbol{\mu}_\theta^{(t)}(\mathbf{x}_t) = \frac{\sqrt{\alpha_{t-1}}}{\sqrt{\alpha_t}}\big(\mathbf{x}_t - \frac{1 - (\alpha_t/\alpha_{t-1})}{\sqrt{1 - \alpha_t}}\epsilon_\theta^{(t)}(\mathbf{x}_t)\big). \tag{14}$$

The diffusion objective is expressed as:

$$\mathcal{L}_\gamma(\epsilon_\theta) := \sum_{t=1}^{T} \gamma_t \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\left\| \epsilon_\theta^{(t)}(\mathbf{x}_t) - \epsilon_t \right\|_2^2 \right], \tag{15}$$

where $\mathbf{x}_t$ is efficiently sampled by Equation (11), and $\gamma := [\gamma_1, \ldots, \gamma_T]^T$ denotes the weighting vector.

10.2. Denoising Diffusion Implicit Models

The forward process in Equation (9) is a Markov process where the current state $\mathbf{x}_t$ solely depends on the previous state $\mathbf{x}_{t-1}$. Denoising diffusion implicit models (DDIM) [24] is a non-Markov diffusion model where each forward process is conditioned on both $\mathbf{x}_0$ and $\mathbf{x}_{t-1}$. Specifically, the forward posterior is defined as:

$$q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_0 + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0}{\sqrt{1 - \alpha_t}}, \sigma_t^2 \boldsymbol{I}\right), \tag{16}$$

where $\{\sigma_t\}_{t=1}^T$ controls the uncertainly of the diffusion trajectories. Since this posterior is designed to match $q(\mathbf{x}_t \mid \mathbf{x}_0)$ as DDPM, the generative process can be trained with the same objective described in Equation (15), for any $\{\sigma_t\}_{t=1}^T$. When $\sigma_t \to 0$, the generative process becomes deterministic and can be represented as:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\boldsymbol{f}_\theta(\mathbf{x}_t, t) + \sqrt{1 - \alpha_{t-1}}\epsilon_\theta^{(t)}(\mathbf{x}_t), \tag{17}$$

where $\boldsymbol{f}_\theta(\mathbf{x}_t, t) = (\mathbf{x}_t - \sqrt{1 - \alpha_t}\epsilon_\theta^{(t)}(\mathbf{x}_t))/\sqrt{\alpha_t}$. For more detailed proof and derivation of DDIM, see [24].

10.3. Derivation of DDIM Inversion

From Equation (17), we have:

$$\mathbf{y}_{t-1} - \mathbf{y}_t = (p_{t-1} - p_t)\epsilon_\theta^{(t)}(\mathbf{x}_t), \tag{18}$$

where $\mathbf{y}_t := \mathbf{x}_t/\sqrt{\alpha_t}$ and $p_t := \sqrt{1/\alpha_t - 1}$. In the limit of $T \to \infty$, the above differential equation converges to a continuous-time ODE:

$$\mathrm{d}\mathbf{y}_t = \epsilon_\theta^{(t)}\,\mathrm{d}p_t. \tag{19}$$

Based on the ODE formulation of DDIM, we can derive Equation (17) by applying Euler integration to Equation (19) with the step size $\Delta p_t^+ := p_{t-1} - p_t (> 0)$. Similarly, by applying the forward Euler method to the Equation (19) within the range $[t-1, t]$, we have:

$$\mathbf{y}_t = \mathbf{y}_{t-1} + \Delta p_t^- \epsilon_\theta^{(t-1)}(\mathbf{x}_{t-1}), \tag{20}$$

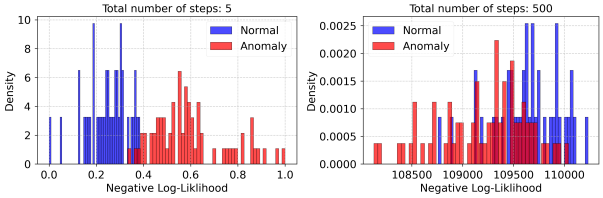
where $\Delta p_t^- := p_t - p_{t-1}$. Next, replacing $\mathbf{y}_t$ with $\mathbf{x}_t/\sqrt{\alpha_t}$ in Equation (20) yields:

$$\mathbf{x}_t = \sqrt{\alpha_t}\big(\frac{\mathbf{x}_{t-1}}{\sqrt{\alpha_{t-1}}} + \Delta p_t^- \epsilon_\theta^{(t-1)}(\mathbf{x}_{t-1})\big). \tag{21}$$

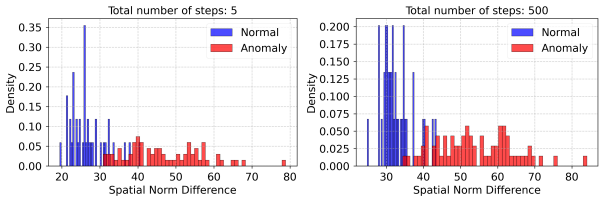
It can be easily noticed that Equation Equation (21) is essentially equivalent to Equation (6). A similar discussion applies to the subset of timesteps $\boldsymbol{\tau}_S = [\tau_1, \tau_2, \ldots, \tau_S = T] \subset \{1, 2, \ldots, T\}$ by considering arbitray time interval $[\tau_s, \tau_{s+1}]$.

11. Reverse Scoring Problem

As shown in Tables 4 and 9 (in the main paper), we observe a performance drop when the number of diffusion steps S increases in conjunction with NLL-based anomaly scoring. To investigate the underlying cause of this phenomenon, we visualize the NLL distribution across different values of S in Figure 4. Interestingly, the histogram of NLL exhibits counterintuitive results: as S increases by 500, normal images are assigned lower likelihoods, suggesting that learned normal samples occupy lower-density regions than anomalous ones. This observation aligns with



(a) Histogram of normal and anomalous samples with NLL scoring.



(b) Histogram of normal and anomalous samples with *our NLL+Diff* scoring.

Figure 4. **Comparison of the histogram** of normal and anomalous samples, with the conventional NLL scoring (a) and our proposed *NLL+Diff* scoring (b), on the test set of *hazelnut* in MVTecAD.

findings in flow-based out-of-distribution (OOD) detection literature [47, 48], where a similar tendency arises due to the nature of high-dimensional data distributions. This is known as the *reverse-scoring* issue, which is widely acknowledged in dealing with high-dimensional space.