BEYOND REVERSE DENOISING: EFFICIENTLY DETECTING TIME SERIES ANOMALIES THROUGH PROTOTYPE-GUIDED DIFFUSION INVERSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Most diffusion-based time series anomaly detection methods rely on iterative reverse denoising, coupling detection with generative fidelity even though sample generation is not the goal. Expressive denoisers may reconstruct anomalous patterns, while repeated sampling incurs substantial inference overhead. We argue that the distribution-learning capability of diffusion models can instead define an efficient one-class normality criterion. To this end, we propose *PiAD*, a prototype-guided framework that detects time series anomalies through deterministic DDIM inversion trajectories without reverse denoising or test-time reconstruction. Our key insight is that a noise predictor learned from normal-dominated data defines normal latent dynamics across noise levels: normal windows tend to follow mode-consistent data-to-noise trajectories, whereas anomalous windows exhibit incompatible transitions. Because normal behavior is often multimodal, a single global trajectory prior may conflate distinct normal modes. *PiAD* therefore combines prototype-conditioned latent diffusion with prototype-specific gaussian trajectory priors and uses training-calibrated Top-$\mathbf{k}$ aggregation to retain the most discriminative deviations across inversion states. Experiments on five real-world datasets show that *PiAD* consistently outperforms existing diffusion-based approaches while requiring substantially less inference time, memory, and model capacity.

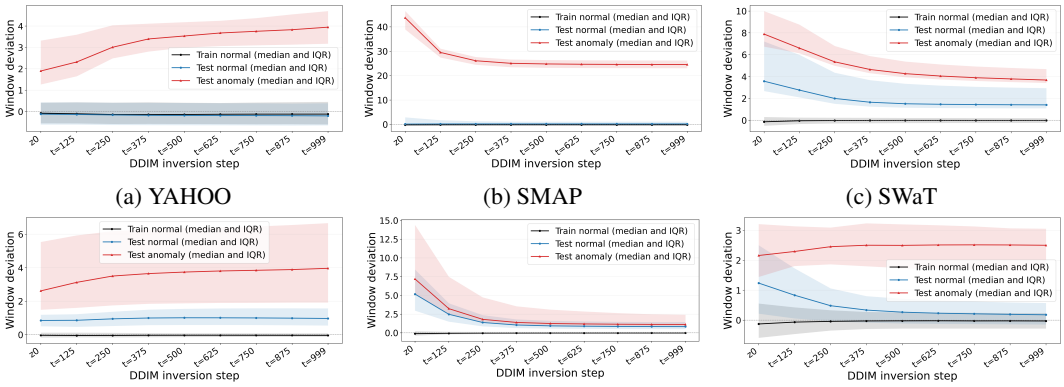
1 INTRODUCTION

Time series anomaly detection (TSAD) is essential for monitoring industrial systems, cloud services, spacecraft, and critical infrastructure (Mathur & Tippenhauer, 2016; Hundman et al., 2018; Su et al., 2019; Abdulaal et al., 2021; Zhang et al., 2022; Schmidl et al., 2022). Because anomalies are rare, diverse, and expensive to annotate, practical TSAD is commonly formulated as an unsupervised or one-class problem: a model learns regular behavior from normal-dominated training data and identifies observations incompatible with this learned normality.

Existing unsupervised TSAD methods typically measure such incompatibility using forecasting errors, reconstruction errors, or representation discrepancies. Forecasting-based approaches identify violations of predicted dynamics, while reconstruction-based methods assume that anomalous inputs are reconstructed less accurately (Audibert et al., 2020; Tuli et al., 2022; Lai et al., 2023; Feng et al., 2024). Other approaches construct anomaly criteria from temporal associations or hierarchical representations (Shen et al., 2020; Xu et al., 2022). Despite their effectiveness, these methods often reduce normality to a single prediction, reconstruction, or representation. Consequently, expressive models may reconstruct anomalies accurately, while complex but valid normal patterns may still produce large errors.

Diffusion models offer an alternative because noise-prediction training characterizes the data distribution across a broad range of noise levels (Ho et al., 2020). Recent evidence in computer vision also suggests that generative models acquire representations useful for downstream perception tasks (Gabeur et al., 2026). Accordingly, the value of diffusion models for anomaly detection need not be limited to generation or reconstruction: their learned noise predictors also encode how training-distribution observations evolve across noise scales.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) YAHOO (b) SMAP (c) SWaT

(d) SMD (e) PSM (f) MSL

Figure 1: Prototype-specific calibrated window deviations across DDIM inversion states on six datasets. (YAHOO is used only as a univariate illustrative example)

Nevertheless, existing diffusion-based TSAD methods typically use diffusion as a reconstruction or imputation engine. They estimate test windows through denoising-based procedures, starting either from gaussian noise in masked regions or from corrupted input states, and derive anomaly scores from reconstruction or imputation discrepancies (Xiao et al., 2023; Chen et al., 2023; Wang et al., 2023; Zhong et al., 2025; Li et al., 2025; Ma et al., 2026; Zhang et al., 2026). This couples detection with generative fidelity, even though anomaly detection only requires distinguishing abnormal behavior. Moreover, expressive denoisers may recover anomalous structures, while repeated denoising and decoding introduce considerable inference overhead.

We therefore ask a different question: rather than determining whether a test window can be reconstructed, can we directly assess whether its evolution under learned diffusion dynamics is compatible with normal time series? A diffusion model trained on normal-dominated data estimates a vector field describing normal latent evolution across noise levels. Deterministic DDIM inversion (Song et al., 2021) propagates an observed window from data toward noise through this learned field. Normal windows are expected to follow supported trajectories, whereas anomalous windows may encounter incompatible directions and deviate at one or more noise levels.

This view converts diffusion distribution learning into a one-class normality criterion without requiring reverse generation. Related inversion-based image anomaly detection has shown that diffusion models can map normal images toward typical regions of the noise prior and detect anomalies without reconstruction (Sakai et al., 2026). However, relying only on the terminal representation may overlook evidence that appears at earlier states and becomes attenuated near the common gaussian endpoint. As illustrated in Figure 1, progression toward noisier states amplifies anomaly evidence or improves normal–anomaly separation in some cases, whereas in others the strongest evidence appears earlier and weakens toward the terminal state. This variation motivates multi-step inversion and selective aggregation rather than reliance on a single state.

A global trajectory reference introduces another difficulty. Real-world time series often contain multiple normal operating modes associated with different workloads, phases, or control states. Although these modes may approach the same gaussian endpoint, they can follow distinct intermediate paths. Averaging them into a single global prior may therefore create an artificial trajectory that penalizes legitimate mode-specific behavior.

Motivated by these observations, we propose *PiAD*, a prototype-guided latent diffusion framework for detecting anomalies through data-to-noise trajectories. *PiAD* encodes each window into a latent sequence and extracts temporal prototypes representing heterogeneous normal patterns. A sample-adaptive prototype condition is injected into the noise predictor, while a prototype-consistency objective encourages predicted clean latents to remain compatible with the corresponding normal mode. The resulting model learns mode-aware normal dynamics rather than collapsing all normal behavior into a single representation.

After training, sparse deterministic DDIM inversion produces multi-state latent trajectories for reference and test windows. *PiAD* groups normal trajectories by prototype and estimates prototype-

specific gaussian statistics at each state. Each test trajectory is compared with its matched mode-specific prior, producing token-level deviations across noise scales. These deviations are calibrated exclusively using training trajectories, after which the Top-$\mathbf{k}$ most discriminative states are selectively aggregated. This yields a multi-scale anomaly criterion without assuming that the terminal state alone must separate every anomaly.

Because *PiAD* performs only a small number of latent-space inversion transitions, it avoids reverse denoising and test-time decoding, retaining diffusion distribution learning while removing generation as an unnecessary intermediate objective. Our contributions are summarized as follows:

- We reinterpret diffusion-based TSAD as one-class normality estimation through deterministic data-to-noise dynamics. Instead of reconstructing test windows, *PiAD* detects deviations along DDIM inversion trajectories, enabling reconstruction-free and efficient inference.
- We develop prototype-conditioned latent diffusion and prototype-specific gaussian trajectory priors to model heterogeneous normal modes. The conditioning mechanism learns mode-aware dynamics, while the corresponding priors avoid comparison with an artificial global average.
- We develop a training-calibrated Top-$\mathbf{k}$ aggregation strategy that preserves evidence from discriminative noise levels. Experiments on five real-world datasets demonstrate superior detection performance and computational efficiency over existing diffusion-based TSAD methods.

2 Related Works: Diffusion Models for TSAD

Existing diffusion-based TSAD methods predominantly formulate anomaly detection as reconstruction or imputation. DiffAD (Xiao et al., 2023) and IMDiffusion (Chen et al., 2023) employ conditional diffusion for imputing test observations, while D^{3}R (Wang et al., 2023), MODEM (Zhong et al., 2025), and PhysDiff (Li et al., 2025) introduce temporal decomposition, multiresolution modeling, or physical priors into diffusion reconstruction. EnsDiffAD (Ma et al., 2026) and ICDiffAD (Zhang et al., 2026) further improve reconstruction robustness through model ensembles and implicit conditioning. Despite these differences, existing methods generally rely on iterative reverse denoising and derive anomaly evidence from reconstruction discrepancies. In contrast, *PiAD* directly evaluates deterministic data-to-noise trajectories against prototype-specific normal priors, avoiding reverse diffusion and test-time reconstruction, substantially reducing inference time and memory consumption. A more detailed discussion is provided in Appendix A.1.

3 Preliminaries: Diffusion Models and DDIM Inversion

Let $\mathbf{z}^{(0)} \sim q_{\text{data}}$ denote a clean latent sample. A diffusion model progressively perturbs $\mathbf{z}^{(0)}$ according to a predefined noise schedule $\{\beta_n\}_{n=1}^{N}$, where N is the number of diffusion steps. Defining $\alpha_n = 1 - \beta_n$ and $\bar{\alpha}_n = \prod_{j=1}^{n} \alpha_j$, the noisy state at step n can be sampled directly as

$$\mathbf{z}^{(n)} = \sqrt{\bar{\alpha}_n}\mathbf{z}^{(0)} + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{1}$$

A noise predictor ϵ_θ is trained to estimate the injected noise. For a conditional diffusion model with a generic condition $\mathbf{c}$, the standard noise-prediction objective is

$$\mathcal{L}_{\text{noise}} = \mathbb{E}_{\mathbf{z}^{(0)},n,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \epsilon_\theta(\mathbf{z}^{(n)}, n, \mathbf{c})\right\|_2^2\right], \qquad n \sim \mathcal{U}\{1,\dots,N\}. \tag{2}$$

Given the predicted noise $\hat{\boldsymbol{\epsilon}}^{(n)} = \epsilon_\theta(\mathbf{z}^{(n)}, n, \mathbf{c})$, the corresponding clean-state estimate is

$$\hat{\mathbf{z}}^{(0|n)} = \frac{\mathbf{z}^{(n)} - \sqrt{1-\bar{\alpha}_n}\hat{\boldsymbol{\epsilon}}^{(n)}}{\sqrt{\bar{\alpha}_n}}. \tag{3}$$

DDIM (Song et al., 2021) provides a deterministic transition when its stochasticity parameter is set to zero. In the inversion direction, a state at diffusion step n is mapped to a higher noise level $n' > n$ by

$$\mathbf{z}^{(n')} = \sqrt{\bar{\alpha}_{n'}}\hat{\mathbf{z}}^{(0|n)} + \sqrt{1-\bar{\alpha}_{n'}}\hat{\boldsymbol{\epsilon}}^{(n)}. \tag{4}$$

Applying this transition over an increasing sequence of diffusion steps yields a deterministic data-to-noise trajectory. *PiAD* uses the intermediate states of this trajectory directly for anomaly detection, as introduced in Section 4.

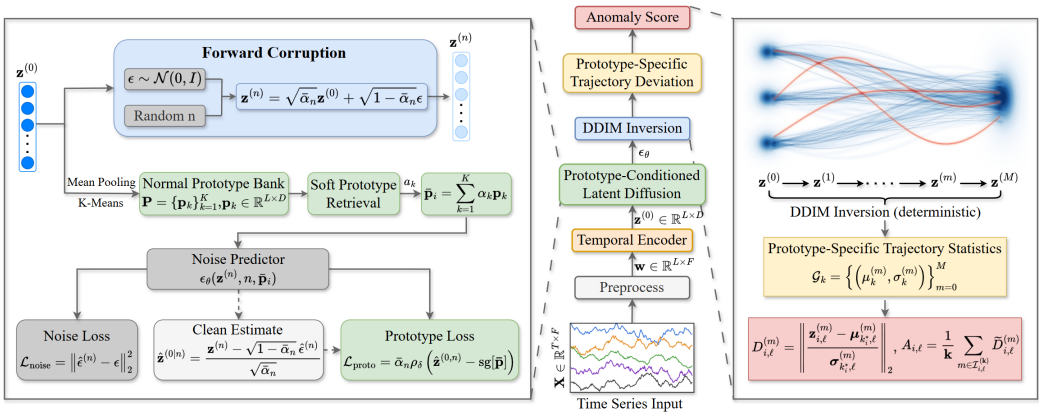


Figure 2: Workflow of the proposed *PiAD*.

4 METHODOLOGY

Let $\mathbf{X} \in \mathbb{R}^{T \times F}$ denote a multivariate time series with T timestamps and F variables. We divide $\mathbf{X}$ into overlapping windows $\mathbf{W} = \{\mathbf{w}_i\}_{i=1}^{N_\mathbf{w}}$, where $\mathbf{w}_i \in \mathbb{R}^{L \times F}$ has length L and starts at timestamp o_i. Following the standard unsupervised setting, the training split is treated as nominal reference data, and no anomaly annotations are used during training. The objective is to produce a point-level anomaly score s_t for each test timestamp t.

4.1 OVERVIEW

As illustrated in Figure 2, *PiAD* models the deterministic latent inversion trajectories of nominal time series windows. It first constructs sequence-level temporal prototypes and uses them as sample-adaptive conditions for latent diffusion learning. Deterministic DDIM inversion is then applied to the training windows to estimate prototype-specific trajectory statistics. During inference, a test trajectory is compared with its corresponding reference statistics, and deviations across inversion states are calibrated and fused into point-level anomaly scores. Detection follows only the data-to-noise inversion direction and does not require reverse denoising or test-time reconstruction. Detailed algorithm workflow is provided in the Appendix A.3.

4.2 PROTOTYPE-CONDITIONED LATENT DIFFUSION

Latent temporal representation. A lightweight temporal encoder E_ϕ maps each input window to a length-preserving latent sequence:

$$\mathbf{z}_i^{(0)} = E_\phi(\mathbf{w}_i) \in \mathbb{R}^{L \times D}, \tag{5}$$

where D is the latent dimension. The encoder can be instantiated using a recurrent, convolutional, or Transformer-based architecture and is not specific to the proposed framework.

An auxiliary decoder G_ψ is jointly optimized to retain information from the input window:

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_i \left[\frac{1}{LF} \left\| G_\psi(\mathbf{z}_i^{(0)}) - \mathbf{w}_i \right\|_2^2 \right]. \tag{6}$$

The decoder is used only during representation learning and is discarded after training.

Sequence-level temporal prototypes. For each training window, we mean-pool its latent sequence into a D-dimensional representation and apply K-Means clustering to the resulting representations. Let $\mathcal{C}_k$ denote the set of training windows assigned to cluster k. After determining cluster membership from the pooled representations, the corresponding prototype is computed by averaging the complete latent sequences:

$$\mathbf{p}_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} \mathbf{z}_i^{(0)} \in \mathbb{R}^{L \times D}, \qquad k = 1, \ldots, K. \tag{7}$$

Each prototype therefore retains the length and temporal organization of its latent sequences. Its retrieval key is defined as $\mathbf{v}_k = \mathrm{MeanPool}(\mathbf{p}_k)$.

For subsequent prototype retrieval, a latent window $\mathbf{z}_i^{(0)}$ is represented by the query $\mathbf{q}_i = \mathrm{MeanPool}(\mathbf{z}_i^{(0)}) \in \mathbb{R}^D$. Its similarity to the prototype keys is converted into a soft assignment:

$$a_{i,k} = \frac{\exp(\cos(\mathbf{q}_i, \mathbf{v}_k)/\tau)}{\sum_{j=1}^{K} \exp(\cos(\mathbf{q}_i, \mathbf{v}_j)/\tau)}, \tag{8}$$

where $\tau > 0$ is a temperature parameter. The sample-adaptive blended prototype is then

$$\bar{\mathbf{p}}_i = \sum_{k=1}^{K} a_{i,k} \mathbf{p}_k \in \mathbb{R}^{L \times D}. \tag{9}$$

Thus, $\bar{\mathbf{p}}_i$ is a convex combination of prototype sequences estimated from the training split and serves as the sample-adaptive condition.

Prototype conditioning is disabled during an initial warm-up stage because the latent space changes rapidly at the beginning of training. The prototype bank is subsequently fitted using the current training latents and periodically recomputed. The retrieved blended prototypes remain detached from the computational graph between consecutive updates.

Prototype-conditioned diffusion learning. Following the forward process in Section 3, gaussian noise is added to $\mathbf{z}_i^{(0)}$ to obtain $\mathbf{z}_i^{(n)}$. In *PiAD*, the generic condition $\mathbf{c}$ is instantiated by the sample-adaptive blended prototype $\bar{\mathbf{p}}_i$. The conditional predictor $\hat{\boldsymbol{\epsilon}}_i^{(n)} = \epsilon_\theta\left(\mathbf{z}_i^{(n)}, n, \bar{\mathbf{p}}_i\right)$ is optimized using the noise-prediction objective in Equation (2).

The predicted noise also provides a clean-latent estimate $\hat{\mathbf{z}}_i^{(0|n)}$ through Equation (3). We regularize this estimate using the retrieved prototype condition:

$$\mathcal{L}_{\mathrm{proto}} = \mathbb{E}_{i,n}\left[\frac{\bar{\alpha}_n}{LD}\sum_{\ell=1}^{L}\sum_{d=1}^{D}\rho_\delta\left(\hat{\mathbf{z}}_{i,\ell,d}^{(0|n)} - \mathrm{sg}[\bar{\mathbf{p}}_{i,\ell,d}]\right)\right], \tag{10}$$

where ρ_δ is the Huber loss and $\mathrm{sg}[\cdot]$ denotes stop-gradient. The coefficient $\bar{\alpha}_n$ assigns smaller weights to clean-state estimates obtained from more strongly corrupted latent states.

The complete training objective is

$$\mathcal{L} = \lambda_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}} + (1 - \lambda_{\mathrm{rec}})\left(\mathcal{L}_{\mathrm{noise}} + \lambda_{\mathrm{proto}}\mathcal{L}_{\mathrm{proto}}\right), \tag{11}$$

where $\lambda_{\mathrm{rec}} \in [0, 1]$ balances reconstruction and diffusion learning, and $\lambda_{\mathrm{proto}} \geq 0$ controls prototype regularization. After optimization, the learned diffusion dynamics are used to construct the nominal trajectory references.

4.3 Prototype-Specific Prior over DDIM Inversion Trajectories

Deterministic DDIM inversion trajectory construction. Let $0 = n_0 < n_1 < \cdots < n_M$ denote the network time indices used for DDIM inversion. The target indices $n_1, \ldots, n_M$ are selected approximately uniformly from the complete diffusion schedule and include its terminal index.

We distinguish the trajectory-state index m from the scheduler index n_m. The trajectory is initialized using the encoded clean state $\mathbf{z}_i^{(0)}$. To represent the clean starting boundary, we define $\gamma_0 = 1$ and $\gamma_m = \bar{\alpha}_{n_m}$ for $m \geq 1$. The blended prototype $\bar{\mathbf{p}}_i$ is computed once from $\mathbf{z}_i^{(0)}$ and remains fixed throughout inversion.

At trajectory state m, the predictor evaluates $\hat{\boldsymbol{\epsilon}}_i^{(m)} = \epsilon_\theta(\mathbf{z}_i^{(m)}, n_m, \bar{\mathbf{p}}_i)$. The corresponding clean-latent estimate is

$$\hat{\mathbf{z}}_i^{(0|m)} = \frac{\mathbf{z}_i^{(m)} - \sqrt{1 - \gamma_m}\,\hat{\boldsymbol{\epsilon}}_i^{(m)}}{\sqrt{\gamma_m}}. \tag{12}$$

The next trajectory state is obtained through the deterministic DDIM inversion:

$$\mathbf{z}_i^{(m+1)} = \sqrt{\gamma_{m+1}}\,\hat{\mathbf{z}}_i^{(0|m)} + \sqrt{1-\gamma_{m+1}}\,\hat{\boldsymbol{\epsilon}}_i^{(m)}. \tag{13}$$

Repeated application of Equation (13) produces

$$\mathcal{T}_i = \left\{\mathbf{z}_i^{(m)}\right\}_{m=0}^{M}. \tag{14}$$

The trajectory contains the initial encoded state and M progressively noisier states. ***PiAD*** directly retains these states for anomaly scoring and does not subsequently execute a reverse denoising trajectory.

Prototype-specific trajectory statistics. Each training trajectory is assigned to its highest-weight prototype, written as $g_i = \arg\max_k a_{i,k}$. For every prototype k, trajectory state m, temporal position ℓ, and latent dimension d, we estimate

$$\mu_{k,\ell,d}^{(m)} = \mathrm{Mean}_{i:g_i=k}\,\mathbf{z}_{i,\ell,d}^{(m)}, \quad \sigma_{k,\ell,d}^{(m)} = \mathrm{Std}_{i:g_i=k}\,\mathbf{z}_{i,\ell,d}^{(m)}. \tag{15}$$

These statistics define a prototype-specific diagonal reference distribution at every trajectory state and temporal position.

For a test window, we select the reference prototype $k_i^* = \arg\max_k a_{i,k}$. Its standardized deviation at state m and temporal position ℓ is

$$D_{i,\ell}^{(m)} = \left\| \frac{\mathbf{z}_{i,\ell}^{(m)} - \boldsymbol{\mu}_{k_i^*,\ell}^{(m)}}{\boldsymbol{\sigma}_{k_i^*,\ell}^{(m)}} \right\|_2. \tag{16}$$

This measures the displacement of the test state after normalizing each latent dimension by its variation in the assigned training trajectories.

4.4 Training-Calibrated Multistate Scoring

The numerical ranges of $D_{i,\ell}^{(m)}$ can vary across trajectory states. For each state m, we therefore estimate $b^{(m)} = \mathrm{Mean}_{i,\ell}\,D_{i,\ell,\mathrm{train}}^{(m)}$ and $r^{(m)} = \mathrm{Std}_{i,\ell}\,D_{i,\ell,\mathrm{train}}^{(m)}$ using only the training trajectories. A test deviation is calibrated as

$$\widetilde{D}_{i,\ell}^{(m)} = \frac{D_{i,\ell}^{(m)} - b^{(m)}}{r^{(m)}}. \tag{17}$$

The fitted calibration parameters remain fixed during testing.

For each temporal position ℓ, let $\mathcal{I}_{i,\ell}^{(\mathbf{k})}$ denote the indices of the $\mathbf{k}$ largest values in $\{\widetilde{D}_{i,\ell}^{(m)}\}_{m=0}^{M}$, where $1 \le \mathbf{k} \le M+1$. The within-window anomaly score is

$$A_{i,\ell} = \frac{1}{\mathbf{k}} \sum_{m \in \mathcal{I}_{i,\ell}^{(\mathbf{k})}} \widetilde{D}_{i,\ell}^{(m)}. \tag{18}$$

The candidate states include $m = 0$, corresponding to the initial encoded representation.

Finally, let $\mathbf{W}(t) = \{i \mid o_i \le t \le o_i + L - 1\}$ denote the windows covering timestamp t. Their contributions are averaged to obtain

$$s_t = \frac{1}{|\mathbf{W}(t)|} \sum_{i \in \mathbf{W}(t)} A_{i,t-o_i+1}. \tag{19}$$

Because the aggregation uses the actual starting timestamp o_i of each window, it applies directly to arbitrary sliding-window strides.

At inference, ***PiAD*** retains the temporal encoder, prototype bank, conditional noise predictor, prototype-specific trajectory statistics, and calibration parameters. Each test window requires one deterministic pass through the selected inversion states. The auxiliary decoder is discarded, and neither a reverse diffusion trajectory nor a nominal reconstruction of the test input is generated.

Table 1: Results on the five real-world datasets. All values are the average percentages over three random seeds. The best is highlighted in **bold** and the second best is underlined.

Model	SMD			SWaT			PSM			MSL			SMAP			Avg.
	VP	VR	AF	VP	VR	AF	VP	VR	AF	VP	VR	AF	VP	VR	AF	rank
KNN	20.67	73.23	86.61	10.87	36.47	55.77	10.39	43.16	71.74	12.38	65.12	89.07	15.23	53.28	63.52	16.53
IForest	12.76	75.19	32.10	28.96	72.98	21.11	16.65	48.14	15.79	70.90	94.56	99.78	42.01	69.50	97.09	13.00
PCA	18.19	77.63	18.16	18.61	61.13	30.49	15.93	53.42	22.76	89.69	**99.66**	99.89	21.35	43.51	95.45	13.27
MAD-GAN	22.65	79.65	81.02	51.97	80.66	79.30	15.96	44.59	70.00	14.66	82.23	85.42	75.32	90.05	85.82	11.07
BeatGAN	28.70	84.31	87.79	52.04	83.84	75.32	30.91	**72.66**	73.82	8.68	77.24	81.46	41.66	82.86	80.00	9.00
USAD	15.12	72.35	18.15	24.50	73.65	51.45	12.99	44.93	20.79	89.42	99.64	**99.94**	20.03	61.73	58.23	14.60
THOC	12.51	70.69	76.88	41.89	78.94	79.13	18.70	49.50	71.77	14.57	71.76	81.49	31.40	66.92	62.69	14.40
FGANomaly	19.70	80.76	81.77	53.04	84.99	77.99	21.38	54.86	32.60	36.97	79.77	97.57	67.14	85.30	98.72	9.07
TranAD	23.78	86.14	85.52	57.59	86.88	79.35	22.71	55.22	29.22	37.24	80.92	97.57	65.62	77.97	95.35	8.07
A.T.	3.69	47.32	64.35	13.26	49.97	20.56	11.43	47.95	46.12	2.15	49.97	75.43	15.50	42.05	87.15	19.13
TimesNet	32.07	87.41	20.61	15.89	54.37	49.91	11.77	44.84	40.65	20.87	95.48	86.52	21.56	69.56	58.14	14.80
MEMTO	14.46	85.89	75.83	39.36	72.54	75.74	17.60	66.59	69.07	14.45	66.87	77.68	26.79	81.74	77.88	13.80
NPSR	28.08	83.91	83.79	42.17	84.37	78.67	22.44	63.41	78.75	39.84	91.60	99.36	95.15	97.93	95.87	5.83
SensitiveHUE	25.40	80.96	83.82	46.67	80.81	79.47	25.39	63.65	73.20	15.93	76.35	88.20	55.61	73.39	95.48	8.60
D^3R	50.48	88.47	85.97	38.99	73.51	80.04	19.11	54.51	69.43	60.34	87.42	99.56	95.05	97.81	98.80	6.60
DiffAD	13.23	72.53	71.40	37.65	68.20	78.46	22.44	58.69	69.48	40.46	81.31	97.37	23.58	51.32	68.93	13.10
IMDiffusion	41.57	84.77	86.51	43.86	71.73	78.87	20.25	52.54	69.04	40.33	81.64	97.27	50.05	89.00	86.50	9.53
PhysDiff	26.36	72.21	83.03	47.76	82.04	76.45	21.37	62.28	71.04	36.40	78.03	97.58	91.57	97.34	95.44	8.93
EnsDiffAD	27.90	74.93	81.48	39.53	75.93	77.68	18.40	57.79	70.44	25.51	79.54	88.12	38.83	78.09	84.69	11.93
ICDiffAD	16.31	76.74	87.14	43.89	78.28	80.10	21.05	53.02	72.82	37.54	82.12	97.57	81.07	95.20	88.85	8.33
Ours	**56.20**	**95.24**	**91.40**	**64.82**	**90.57**	**81.23**	**34.62**	70.96	**80.45**	**92.17**	98.49	99.66	**99.70**	**99.94**	**99.84**	1.40

Discussion. Existing diffusion inversion detectors have mainly been studied in the image domain. DiffGuard (Gao et al., 2023) and FakeInversion (Cazenavette et al., 2024) use inversion to support subsequent reconstruction or supervised classification, rather than directly deriving anomaly evidence from the inversion process. DiffPath (Heng et al., 2024) exploits the data-to-noise path but compresses it into global derivative statistics, thereby overlooking local structure and heterogeneous normal modes. InvAD (Sakai et al., 2026) is more closely related to our proposed method, but it primarily scores the terminal inverted feature with respect to a shared gaussian prior, leaving intermediate dynamics and mode-dependent normality unmodeled. Despite these advances in vision, deterministic diffusion inversion remains largely unexplored in the time series domain and has not been investigated as a reconstruction-free inference mechanism for time series anomaly detection. The proposed *PiAD* fills this gap by learning prototype-specific distributions for token-level states at multiple inversion timesteps, avoiding reliance on a single global terminal prior while accounting for heterogeneous normal regimes. To the best of our knowledge, *PiAD* is the first framework to employ deterministic data-to-noise diffusion inversion as the primary inference mechanism for time series anomaly detection. By eliminating reverse denoising and test-time reconstruction, *PiAD* provides an effective and computationally efficient alternative to existing diffusion-based detectors.

5 Experiments

5.1 Experimental Settings

Datasets. We evaluate our method on five real-world multivariate time series anomaly detection datasets covering industrial control, service monitoring, and spacecraft telemetry: SMD (Su et al., 2019), SWaT (Mathur & Tippenhauer, 2016), PSM (Abdulaal et al., 2021), MSL and SMAP (Hundman et al., 2018). We follow the official training/test splits provided by (Liu & Paparrizos, 2024) and the training data is further split into train/validation sets with an 8:2 ratio to enable hyperparameter tuning and early stopping. Detailed information about the datasets is provided in Appendix A.4.1.

Baselines. We compare our method with 20 baselines, which can be broadly divided into three categories: i) traditional machine learning methods, including KNN (Ramaswamy et al., 2000), IForest (Liu et al., 2008), and PCA (Aggarwal, 2016); ii) non-diffusion deep learning methods, including MAD-GAN (Li et al., 2019), BeatGAN (Zhou et al., 2019), USAD (Audibert et al., 2020), THOC (Shen et al., 2020), FGANomaly (Du et al., 2021), TranAD (Tuli et al., 2022), Anomaly Transformer (A.T.) (Xu et al., 2022), TimesNet (Wu et al., 2023), MEMTO (Song et al., 2023), NPSR (Lai et al., 2023), and SensitiveHUE (Feng et al., 2024); and iii) diffusion-based methods, including D^3R (Wang et al., 2023), DiffAD (Xiao et al., 2023), IMDiffusion (Chen et al., 2023), PhysDiff (Li et al., 2025), EnsDiffAD (Ma et al., 2026), and ICDiffAD (Zhang et al., 2026).

Evaluation Metrics. We evaluate anomaly detection models using VUS-PR (VP) and VUS-ROC (VR) (Paparrizos et al., 2022) and Affiliation-F1 (AF) (Huet et al., 2022). VUS-PR and VUS-ROC provide threshold-independent, range-aware evaluations across different temporal tolerance levels, while Affiliation-F1 measures localized event-level agreement between predicted and ground-truth anomaly ranges. Together, they provide a comprehensive assessment of ranking performance, temporal robustness, and anomaly localization quality.

5.2 Main Results

Table 1 summarizes the results on five real-world datasets. Our method achieves the best performance in 12 of 15 dataset-metric combinations and the best average rank of 1.40. The improvements are particularly pronounced on SMD, where it exceeds the strongest competing results by 5.72%, 6.77%, and 3.61% in VUS-PR, VUS-ROC, and Affiliation F1, respectively. It also achieves near-perfect performance on SMAP and remains highly competitive on PSM and MSL. Notably, our method consistently outperforms all diffusion-based baselines, demonstrating the advantages of prototype-conditioned inversion and prototype-specific normal trajectory priors.

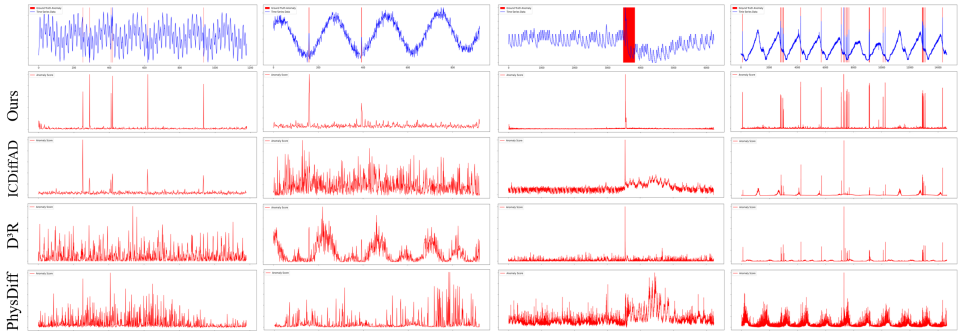


Figure 3: Comparison of anomaly score with ICDiffAD (Zhang et al., 2026), D^{3}R (Wang et al., 2023), and PhysDiff (Li et al., 2025) on four representative time series. Our method produces sharper anomaly responses while maintaining more stable scores in normal regions.

Figure 3 qualitatively compares our method with three strong diffusion-based baselines on four representative time series. Our method produces sharp peaks aligned with ground-truth anomalies while maintaining stable, low scores in normal regions. In contrast, competing methods frequently respond to normal oscillations, periodic turning points, or distribution shifts, resulting in noisier scores and potential false alarms. This separation arises because prototype-specific trajectory priors compare each window with its corresponding normal mode, suppressing legitimate temporal variations, while training-calibrated Top-**k** aggregation emphasizes the inversion steps at which anomalies deviate most strongly. Overall, the visualization confirms that our method provides clearer separation between normal and anomalous regions than other diffusion-based detectors.

5.3 Ablation Studies

In this section, we conduct ablation studies to analyze: i) the effects of the reconstruction loss $\mathcal{L}_{\text{rec}}$ and prototype-consistency loss $\mathcal{L}_{\text{proto}}$, and ii) the roles of prototype conditioning (PC) and the prototype-specific prior (PSP). The standard diffusion noise-prediction loss $\mathcal{L}_{\text{noise}}$ is retained in all variants, as it constitutes the fundamental objective for training the diffusion model rather than an auxiliary component.

Effects of $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{proto}}$. We evaluate: (1) the reconstruction loss $\mathcal{L}_{\text{rec}}$, which preserves temporal information in the latent representations, and (2) the prototype-consistency loss $\mathcal{L}_{\text{proto}}$, which encourages the predicted clean latents to approach their corresponding normal prototypes. As shown in Table 2(a), removing either loss generally decreases performance, while excluding both produces the lowest overall results. Combining the two losses consistently achieves the best performance across all five datasets. The improvements are particularly evident on MSL and SMAP, demonstrating that structural preservation and prototype regularization provide complementary supervision for learning compact and discriminative normal representations.

Effects of PC and PSP. We compare four variants: (1) removing both prototype components, (2) using prototype conditioning (PC) with a global trajectory prior, (3) using unconditional inversion with prototype-specific priors (PSP), and (4) combining PC with PSP. As shown in Table 2(b), PC alone degrades performance because a global gaussian prior cannot adequately represent its mode-dependent trajectories. PSP alone improves PSM and MSL but remains inconsistent across datasets without prototype-conditioned dynamics. Combining PC with PSP achieves the best results on all datasets, improving PSM and MSL from 30.12% and 82.26% to 34.62% and 92.17%, respectively. Notably, neither component yields consistent gains alone, confirming their complementarity: PC establishes mode-aware inversion dynamics, while PSP evaluates trajectories against the corresponding normal modes.

Table 2: Ablation studies of training objectives and prototype mechanisms in terms of VUS-PR.

(a) Training objectives

$\mathcal{L}_{\text{rec}}$	$\mathcal{L}_{\text{proto}}$	SMD	SWaT	PSM	MSL	SMAP
×	×	47.28	57.27	27.80	79.79	87.86
✓	×	54.92	58.31	33.74	82.36	94.12
×	✓	51.35	62.78	29.08	89.37	96.94
✓	✓	**56.20**	**64.82**	**34.62**	**92.17**	**99.70**

(b) Prototype mechanisms

PC	PSP	SMD	SWaT	PSM	MSL	SMAP
×	×	51.39	60.14	30.12	82.26	99.12
✓	×	48.86	58.52	29.38	71.66	92.16
×	✓	50.51	57.15	31.23	86.07	97.89
✓	✓	**56.20**	**64.82**	**34.62**	**92.17**	**99.70**

5.4 Computational Efficiency Analysis

Table 3: Computational efficiency across five datasets. Parameter counts are in millions, and each dataset entry reports train time (s) / test time (s) / peak memory (GB), averaged over three runs. Reductions are relative to the second-best results.

Model	Params. (M)	SMD	SWaT	PSM	MSL	SMAP
		Train / Test / Mem.	Train / Test / Mem.	Train / Test / Mem.	Train / Test / Mem.	Train / Test / Mem.
D^3R	52.2	192.9 / 111.3 / 1.5	510.7 / 66.0 / 1.5	4438.1 / 1008.4 / 1.5	174.4 / 24.6 / 1.5	203.7 / 36.7 / 1.5
DiffAD	38.9	91.3 / 235.8 / 3.4	50.2 / 70.6 / 5.8	133.4 / 1461.3 / 4.2	36.9 / 30.8 / 3.0	31.3 / 40.9 / 2.9
IMDiffusion	0.4	199.3 / 100.1 / 2.2	343.4 / 61.7 / 4.5	1037.3 / 295.1 / 1.5	129.4 / 21.7 / 2.6	41.5 / 12.3 / 1.5
PhysDiff	10.6	460.8 / 337.1 / 1.6	1088.6 / 316.4 / 2.4	2587.1 / 1782.8 / 1.4	141.6 / 104.1 / 2.0	250.8 / 74.7 / 1.4
EnsDiffAD	13.5	736.6 / 191.3 / 12.2	2598.3 / 577.4 / 14.3	6252.8 / 1273.4 / 9.5	1464.9 / 62.4 / 12.0	323.0 / 50.7 / 9.5
ICDiffAD	32.2	100.2 / 67.4 / 1.3	166.6 / 117.5 / 6.7	673.3 / 64.1 / 1.0	33.5 / 38.7 / 2.8	104.4 / 259.6 / 5.2
Ours	**0.3**	**15.8 / 5.3 / 0.3**	**25.9 / 6.8 / 0.2**	**98.0 / 18.1 / 0.2**	**21.6 / 2.5 / 0.1**	**20.4 / 3.2 / 0.1**
Reduction vs. 2nd (%)	*25.0*	*82.7 / 92.1 / 76.9*	*48.4 / 89.0 / 86.7*	*26.5 / 71.8 / 80.0*	*35.5 / 88.5 / 93.3*	*34.8 / 74.0 / 92.9*

Table 3 compares computational efficiency across five datasets. *PiAD* uses only 0.3M parameters and consistently achieves the lowest testing time and peak memory consumption. Compared with the corresponding second best results, testing time is reduced by 71.8% to 92.1%, while peak memory consumption is reduced by 76.9% to 93.3%. These gains mainly result from avoiding the reconstruction-based inference procedure used by existing diffusion-based anomaly detectors, which typically requires iterative reverse denoising and, in some cases, multiple stochastic reconstructions. In contrast, our method performs deterministic DDIM inversion at a sparse set of timesteps and directly evaluates latent-trajectory deviations against prototype-specific diagonal gaussian priors. Operating in a compact latent space and using lightweight prototype assignment and element-wise distance computations further reduce computational and memory overhead.

6 Conclusion

In this work, we propose *PiAD*, a reconstruction-free framework that detects time series anomalies by evaluating their compatibility with learned normal data-to-noise dynamics. It combines deterministic DDIM inversion, prototype-conditioned diffusion, prototype-specific trajectory priors, and training-calibrated Top-**k** aggregation to model heterogeneous normal patterns and capture discriminative deviations without reverse denoising or test-time reconstruction. Experiments on five real-world datasets show that *PiAD* achieves the best overall average rank while substantially reducing model size, inference time, and memory consumption. Ablation and trajectory analyses validate its main components and show that anomaly evidence varies across noise levels, with the terminal state not necessarily being the most discriminative. These results establish diffusion inversion trajectories as an effective and efficient alternative to reconstruction for time series anomaly detection.

AI USE STATEMENT

We used generative AI for translation, polishing, and consistency checks. Generative AI was not used to develop the proposed methods, generate data, conduct experiments, or make theoretical claims. All AI-generated content has been rigorously reviewed by humans. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide an anonymous link to the source code in Appendix A.6. Sections 3 and 4 present the mathematical formulation and implementation workflow of *PiAD*, while Section 5.1 describes the evaluation metrics and experimental protocol. Appendix A.2 details data splitting, normalization, and sliding window construction, and Appendix A.3 provides the complete training, trajectory prior estimation, and inference algorithm. Appendices A.4.1 and A.4.2 document the datasets and baseline implementations, while Appendix A.4.3 specifies the computational environment, model architecture, training settings, and hyperparameters.

REFERENCES

Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. Practical approach to asynchronous multivariate time series anomaly detection and localization. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2485–2494, 2021.

Charu C Aggarwal. An introduction to outlier analysis. In *Outlier analysis*, pp. 1–34. Springer, 2016.

Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 3395–3404, 2020.

George Cazenavette, Avneesh Sud, Thomas Leung, and Ben Usman. Fakeinversion: Learning to detect images from unseen text-to-image models by inverting stable diffusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10759–10769, 2024.

Yuhang Chen, Chaoyun Zhang, Minghua Ma, Yudong Liu, Ruomeng Ding, Bowen Li, Shilin He, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. Imdiffusion: Imputed diffusion models for multivariate time series anomaly detection. *Proceedings of the VLDB Endowment*, pp. 359–372, 2023.

Bowen Du, Xuanxuan Sun, Junchen Ye, Ke Cheng, Jingyuan Wang, and Leilei Sun. Gan-based anomaly detection for multivariate time series using polluted training set. *IEEE Transactions on Knowledge and Data Engineering*, pp. 12208–12219, 2021.

Yuye Feng, Wei Zhang, Yao Fu, Weihao Jiang, Jiang Zhu, and Wenqi Ren. Sensitivehue: Multivariate time series anomaly detection by enhancing the sensitivity to normal patterns. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 782–793, 2024.

Valentin Gabeur, Shangbang Long, Songyou Peng, Paul Voigtlaender, Shuyang Sun, Yanan Bao, Karen Truong, Zhicheng Wang, Wenlei Zhou, Jonathan T Barron, et al. Image generators are generalist vision learners. *arXiv preprint arXiv:2604.20329*, 2026.

Ruiyuan Gao, Chenchen Zhao, Lanqing Hong, and Qiang Xu. Diffguard: Semantic mismatch-guided out-of-distribution detection using pre-trained diffusion models. In *IEEE/CVF International Conference on Computer Vision*, pp. 1579–1589, 2023.

Alvin Heng, Alexandre H Thiery, and Harold Soh. Out-of-distribution detection with a single unconditional diffusion model. *Advances in Neural Information Processing Systems*, pp. 43952–43974, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, pp. 6840–6851, 2020.

Alexis Huet, Jose Manuel Navarro, and Dario Rossi. Local evaluation of time series anomaly detection algorithms. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 635–645, 2022.

Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 387–395, 2018.

Chih-Yu Andrew Lai, Fan-Keng Sun, Zhengqi Gao, Jeffrey H Lang, and Duane Boning. Nominality score conditioned time series anomaly detection by point/sequential reconstruction. *Advances in Neural Information Processing Systems*, pp. 76637–76655, 2023.

Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In *International conference on Artificial Neural Networks*, pp. 703–716, 2019.

Long Li, Wencheng Zhang, Shi Yuan, Hongle Guo, and Wanghu Chen. Physdiff: A physically-guided diffusion model for multivariate time series anomaly detection. *Advances in Neural Information Processing Systems*, pp. 105720–105750, 2025.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proceedings of the IEEE International Conference on Data Mining*, pp. 413–422, 2008.

Qinghua Liu and John Paparrizos. The elephant in the room: Towards a reliable time-series anomaly detection benchmark. *Advances in Neural Information Processing Systems*, pp. 108231–108261, 2024.

Qian Ma, Yanyang Li, Mei Bai, Xite Wang, Shikai Guo, Yu Gu, and Ge Yu. Ensdiffad: Ensemble diffusion models for multivariate time series anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 2026.

Aditya P Mathur and Nils Ole Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *Proceedings of the International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, pp. 31–36, 2016.

John Paparrizos, Paul Boniol, Themis Palpanas, Ruey S Tsay, Aaron Elmore, and Michael J Franklin. Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection. *Proceedings of the VLDB Endowment*, pp. 2774–2787, 2022.

Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 427–438, 2000.

Shunsuke Sakai, Xiangteng He, Chunzhi Gu, Leonid Sigal, and Tatsuhito Hasegawa. Invad: Inversion-based reconstruction-free anomaly detection with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21389–21398, 2026.

Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *Proceedings of the VLDB Endowment*, pp. 1779–1797, 2022.

Lifeng Shen, Zhuocong Li, and James Kwok. Timeseries anomaly detection using temporal hierarchical one-class network. *Advances in Neural Information Processing Systems*, pp. 13016–13026, 2020.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

Junho Song, Keonwoo Kim, Jeonglyul Oh, and Sungzoon Cho. Memto: Memory-guided transformer for multivariate time series anomaly detection. *Advances in Neural Information Processing Systems*, pp. 57947–57963, 2023.

Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2828–2837, 2019.

Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *Proceedings of the VLDB Endowment*, pp. 1201–1214, 2022.

Chengsen Wang, Zirui Zhuang, Qi Qi, Jingyu Wang, Xingyu Wang, Haifeng Sun, and Jianxin Liao. Drift doesn't matter: Dynamic decomposition with diffusion reconstruction for unstable multivariate time series anomaly detection. *Advances in Neural Information Processing Systems*, pp. 10758–10774, 2023.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

Chunjing Xiao, Zehua Gou, Wenxin Tai, Kunpeng Zhang, and Fan Zhou. Imputation-based time-series anomaly detection with conditional weight-incremental diffusion models. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2742–2751, 2023.

Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. In *International Conference on Learning Representations*, 2022.

Fan Zhang, Sinchee Chin, Jing-Hao Xue, and Wenming Yang. Icdiffad: Implicit conditioning diffusion model for time series anomaly detection. In *International Conference on Learning Representations*, pp. 54492–54510, 2026.

Shenglin Zhang, Zhenyu Zhong, Dongwen Li, Qiliang Fan, Yongqian Sun, Man Zhu, Yuzhi Zhang, Dan Pei, Jiyan Sun, Yinlong Liu, et al. Efficient kpi anomaly detection through transfer learning for large-scale web services. *IEEE Journal on Selected Areas in Communications*, pp. 2440–2455, 2022.

Guojin Zhong, Pan Wang, Jin Yuan, Zhiyong Li, and Long Chen. Multi-resolution decomposable diffusion model for non-stationary time series anomaly detection. In *International Conference on Learning Representations*, pp. 63630–63656, 2025.

Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. Beatgan: Anomalous rhythm detection using adversarially generated time series. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 4433–4439, 2019.

A APPENDIX

A.1 DETAILED DISCUSSION OF RELATED WORKS

Existing diffusion-based methods for time series anomaly detection primarily formulate detection as a reconstruction or imputation problem. DiffAD (Xiao et al., 2023) introduces conditional weight-incremental diffusion for time series imputation, where selected observations are retained as conditions and the imputation discrepancies are used for anomaly detection. IMDiffusion (Chen et al., 2023) employs neighboring observations to guide diffusion-based imputation and aggregates the intermediate denoising outputs to improve detection robustness.

Subsequent studies incorporate additional temporal structures into diffusion reconstruction. D^3R (Wang et al., 2023) combines dynamic decomposition with diffusion reconstruction to model non-stationary temporal patterns. MODEM (Zhong et al., 2025) introduces a coarse-to-fine diffusion framework that captures cross-resolution dependencies and progressively reconstructs temporal patterns at different resolutions. PhysDiff (Li et al., 2025) integrates physics-guided signal decomposition with conditional diffusion reconstruction, using the resulting frequency components as explicit temporal priors.

Recent methods further improve the robustness and consistency of diffusion-based reconstruction. EnsDiffAD (Ma et al., 2026) ensembles multiple TiT-Diffusion models, each of which separately models temporal and inter-variable dependencies, to improve robustness under distribution shifts. ICDiffAD (Zhang et al., 2026) introduces an SNR-based noise scheduler and an implicit conditioning mechanism that initializes reverse diffusion from partially corrupted inputs, thereby improving reconstruction consistency.

Despite their different conditioning mechanisms and temporal modeling strategies, these approaches primarily depend on iterative reverse denoising to reconstruct or impute a test sequence and derive anomaly evidence from the resulting discrepancies. *PiAD* adopts a different detection paradigm. Instead of generating a nominal reconstruction, it deterministically maps each test window through a compact DDIM inversion trajectory and directly measures its deviation from normal latent trajectory distributions. This reconstruction-free design eliminates costly reverse-denoising chains and repeated generation of candidate reconstructions, substantially reducing inference time and memory consumption. Anomaly detection is thus performed efficiently along the data-to-noise trajectory, without reverse diffusion or test-time reconstruction.

A.2 DATA PREPROCESSING

```
# Step 1: Load the original time series
X, y = load_data(data_source)        # X: (T, F), y: (T,)
T_train = get_train_length(data_source)

# Step 2: Split the training and test sequences
X_train_all = X[:T_train]
y_train_all = y[:T_train]
X_test = X[T_train:]
y_test = y[T_train:]

# Step 3: Construct the validation split
val_start = int((1 - val_ratio) * T_train)

X_train = X_train_all[:val_start]
y_train = y_train_all[:val_start]
X_val = X_train_all[val_start:]
y_val = y_train_all[val_start:]

# Step 4: Normalize using training statistics only
mean = mean_per_feature(X_train)
std = safe_std_per_feature(X_train)

X_train = (X_train - mean) / std
X_val = (X_val - mean) / std
X_test = (X_test - mean) / std

# Step 5: Slice each split into sliding windows
def build_windows(X_split, y_split, win_size, stride):
    if len(X_split) < win_size:
        return

    starts = range(0, len(X_split) - win_size + 1, stride)

    for start in starts:
        end = start + win_size
        window = X_split[start:end]      # shape: (win_size, F)
        label = max(y_split[start:end])  # used only for evaluation

        yield window, label, start

# Step 6: Construct the model inputs
W_train = list(build_windows(X_train, y_train, win_size, stride))
W_val = list(build_windows(X_val, y_val, win_size, stride))
W_test = list(build_windows(X_test, y_test, win_size, stride))

# Step 7: Form mini-batches
train_loader = DataLoader(W_train, batch_size=64, shuffle=True)
val_loader = DataLoader(W_val, batch_size=64, shuffle=False)
test_loader = DataLoader(W_test, batch_size=64, shuffle=False)

# Output batch shape: (batch_size, win_size, F)
```

A.3 ALGORITHM OF *PiAD*

Algorithm 1 Algorithm Flow of *PiAD*

Input: Training series $\mathbf{X}_{\text{train}}$, test series $\mathbf{X}_{\text{test}}$, window length L, stride s, number of prototypes K, diffusion schedule $\{\bar{\alpha}_n\}_{n=1}^{N}$, inversion steps M, Top-$\mathbf{k}$ parameter $\mathbf{k}$.
Output: Point-level anomaly scores $\{s_t\}_{t=1}^{T_{\text{test}}}$.

Initialization:
Divide $\mathbf{X}_{\text{train}}$ and $\mathbf{X}_{\text{test}}$ into training windows $\mathbf{W}_{\text{train}}$ and test windows $\mathbf{W}_{\text{test}}$ with starting timestamps $\{o_i\}$.
Initialize encoder E_ϕ, decoder G_ψ, noise predictor ϵ_θ, and prototype bank $\mathbf{P}$.
/ Phase 1: Prototype-Conditioned Diffusion Learning */*
while not converged **do**
  // Periodically update the prototype bank after warm-up
  if prototype update is required **then**
    Encode $\mathbf{W}_{\text{train}}$ and cluster the pooled latent representations.
    Construct $\mathbf{P} = \{\mathbf{p}_k\}_{k=1}^{K}$ using Equation (7).
  end if
  for each training mini-batch $\mathcal{B} \subset \mathbf{W}_{\text{train}}$ **do**
    $\mathbf{z}_i^{(0)} \leftarrow E_\phi(\mathbf{w}_i), \quad \mathbf{w}_i \in \mathcal{B}$.
    Retrieve the soft prototype condition $\bar{\mathbf{p}}_i$ using Equations (8) and (9).
    Sample diffusion index n and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
    Construct $\mathbf{z}_i^{(n)}$ through forward diffusion and predict $\hat{\boldsymbol{\epsilon}}_i^{(n)} \leftarrow \epsilon_\theta(\mathbf{z}_i^{(n)}, n, \bar{\mathbf{p}}_i)$.
    Compute $\mathcal{L}_{\text{noise}}$, $\mathcal{L}_{\text{rec}}$, and $\mathcal{L}_{\text{proto}}$.
  end for
  Compute the total objective $\mathcal{L}$ using Equation (11).
  Update ϕ, ψ, and θ to minimize $\mathcal{L}$.
end while
/ Phase 2: Prototype-Specific Trajectory Prior */*
Refit the final prototype bank using all training windows.
for each training window $\mathbf{w}_i \in \mathbf{W}_{\text{train}}$ **do**
  $\mathbf{z}_i^{(0)} \leftarrow E_\phi(\mathbf{w}_i)$.
  Retrieve $\bar{\mathbf{p}}_i$ and assign $g_i \leftarrow \arg\max_k a_{i,k}$.
  Construct the deterministic inversion trajectory $\mathcal{T}_i = \{\mathbf{z}_i^{(m)}\}_{m=0}^{M}$ using Equations (12) and (13).
end for
Estimate prototype-specific trajectory statistics $\{\boldsymbol{\mu}_{k,\ell}^{(m)}, \boldsymbol{\sigma}_{k,\ell}^{(m)}\}$ using Equation (15).
Compute training trajectory deviations and fit the calibration parameters $\{b^{(m)}, r^{(m)}\}_{m=0}^{M}$.
/ Phase 3: Anomaly Inference */*
for each test window $\mathbf{w}_j \in \mathbf{W}_{\text{test}}$ **do**
  $\mathbf{z}_j^{(0)} \leftarrow E_\phi(\mathbf{w}_j)$.
  Retrieve $\bar{\mathbf{p}}_j$ and select $k_j^* \leftarrow \arg\max_k a_{j,k}$.
  Construct $\mathcal{T}_j$ through deterministic DDIM inversion ▷ No reverse denoising
  Compute standardized deviations $\{D_{j,\ell}^{(m)}\}_{m=0}^{M}$ using Equation (16).
  Calibrate the deviations using Equation (17).
  Fuse the $\mathbf{k}$ largest deviations into $\{A_{j,\ell}\}_{\ell=1}^{L}$ using Equation (18).
end for
Aggregate overlapping window scores according to their starting timestamps using Equation (19).
Return $\{s_t\}_{t=1}^{T_{\text{test}}}$.

A.4 Detailed Experimental Settings

A.4.1 Datasets

Table 4 summarizes the statistics of the five multivariate time series datasets used in our experiments. MSL and SMAP consist of spacecraft telemetry measurements collected from multiple subsystems and are used to identify abnormal spacecraft operations. SWaT contains sensor and actuator measurements from a water treatment system, representing anomaly detection in industrial control environments. SMD and PSM comprise multivariate monitoring metrics collected from server systems, where anomalies are associated with abnormal service or machine behaviors. These datasets exhibit substantial diversity in dimensionality, sequence length, and anomaly prevalence: their dimensionalities range from 25 to 66, their total lengths range from 6,100 to 217,624 timestamps, and their test anomaly ratios range from 2.86% to 18.21%. This diversity enables a comprehensive evaluation of model performance across heterogeneous application domains and different levels of data imbalance.

Table 4: Statistics of the main datasets.

Dataset	Domain	Dimension	Length	Train	Test	Anomaly Ratio
MSL	Spacecraft	55	6,100	1,525	4,575	2.86%
SMAP	Spacecraft	25	8,209	2,052	6,157	18.21%
SWaT	Water Treatment	66	14,996	3,749	11,247	17.66%
SMD	Server Machine	38	21,203	2,272	18,931	4.13%
PSM	Server Machine	25	217,624	50,000	167,624	14.55%

A.4.2 Baselines

We compare our method with 20 representative baseline methods, covering traditional anomaly detectors, non-diffusion deep learning methods, and diffusion-based methods[1]. All results are from our own runs, using the same computational environment and the official or publicly available implementations listed below. We follow the hyperparameter settings recommended by the original papers or code repositories. For iterative deep models, we retain the checkpoint with the lowest unlabeled validation loss and do not use validation or test anomaly labels during model selection.

- KNN (Ramaswamy et al., 2000): https://github.com/thedatumorg/TSB-AD
- IForest (Liu et al., 2008): https://github.com/thedatumorg/TSB-AD
- PCA (Aggarwal, 2016): https://github.com/thedatumorg/TSB-AD
- MAD-GAN (Li et al., 2019): https://github.com/LiDan456/MAD-GANs
- BeatGAN (Zhou et al., 2019): https://github.com/imbingox/BeatGAN
- USAD (Audibert et al., 2020): https://github.com/manigalati/usad
- THOC (Shen et al., 2020): https://github.com/carrtesy/THOC-Pytorch
- FGANomaly (Du et al., 2021): https://github.com/sxxmason/FGANomaly
- TranAD (Tuli et al., 2022): https://github.com/imperial-qore/TranAD
- AnomalyTransformer (A.T.) (Xu et al., 2022): https://github.com/thuml/Anomaly-Transformer
- TimesNet (Wu et al., 2023): https://github.com/thuml/Time-Series-Library
- MEMTO (Song et al., 2023): https://github.com/gunny97/MEMTO
- NPSR (Lai et al., 2023): https://github.com/andrewlai61616/NPSR
- SensitiveHUE (Feng et al., 2024): https://github.com/yuesuoqingqiu/SensitiveHUE
- D^3R (Wang et al., 2023): https://github.com/ForestsKing/D3R
- DiffAD (Xiao et al., 2023): https://github.com/ChunjingXiao/DiffAD

[1]MODEM (Zhong et al., 2025) is not included in our comparison because we could not identify an official public implementation or sufficient reproducibility resources.

- IMdiffusion (Chen et al., 2023): https://github.com/17000cyh/IMDiffusion
- PhysDiff (Li et al., 2025): https://anonymous.4open.science/r/PhysDiff-4726
- EnsDiffAD (Ma et al., 2026): https://github.com/Steptember/EnsDiffAD
- ICDiffAD (Zhang et al., 2026): https://github.com/ZhangFan21/ICDiffAD

A.4.3 Implementation Details

We implemented our method in Python 3.10 using PyTorch 2.6.0 and conducted all experiments on a single NVIDIA GeForce RTX 5060 Ti GPU with 16 GB of memory. The sliding window length and stride were selected according to the dataset length T. Specifically, we used a window length of 100 with a stride of 1 for $T < 10{,}000$, 150 with a stride of 1 for $10{,}000 \leq T < 20{,}000$, 200 with a stride of 2 for $20{,}000 \leq T \leq 100{,}000$, and 200 with a stride of 5 for $T > 100{,}000$. The batch size was set to 64. The temporal encoder E_ϕ was implemented as a two-layer LSTM with a hidden dimension of 64 and a latent dimension of $D = 64$, while the auxiliary decoder G_ψ was implemented as a two-layer MLP with GELU activation. The conditional noise predictor ϵ_θ was implemented using a 1D U-Net with a hidden dimension of 64 and two bottleneck residual blocks. The number of diffusion time steps was set to 1000 with a linear variance schedule from $\beta_1 = 10^{-4}$ to $\beta_{1000} = 0.02$. For DDIM inversion, we used $M = 8$ inversion steps, resulting in $M + 1 = 9$ latent states including $\mathbf{z}^{(0)}$. The training-calibrated trajectory scores were aggregated using Top-$\mathbf{k}$ fusion with $\mathbf{k} = 3$. The prototype bank contained $K = 8$ temporal prototypes, and the temperature of the soft assignment function was set to $\tau = 1$. Prototype conditioning was activated after five warm-up epochs, and the prototypes were refitted every five epochs. The model was trained for at most 50 epochs using Adam with a learning rate of 10^{-3} and weight decay of 10^{-5}. The loss coefficients were set to $\lambda_{\text{rec}} = 0.3$ and $\lambda_{\text{proto}} = 0.1$, while the Huber-loss parameter was set to $\delta = 1$. Early stopping was applied with a patience of five epochs and a minimum improvement of 10^{-4}, and the best model parameters were restored for final inference.

A.5 More Experimental Results

A.5.1 Comparison Results with Mean and Standard Deviation

Table 5: Results on the five real-world datasets. All values are the average percentages over three random seeds. The best is highlighted in **bold** and the second best is underlined. VP: VUS-PR, VR: VUS-ROC, AF: Affiliation-F1.

Models	SMD			SWaT			PSM			MSL			SMAP		
	VP	VR	AF	VP	VR	AF	VP	VR	AF	VP	VR	AF	VP	VR	AF
KNN	20.67 ±0.00	73.23 ±0.00	86.61 ±0.00	10.87 ±0.00	36.47 ±0.00	55.77 ±0.00	10.39 ±0.00	43.16 ±0.00	71.74 ±0.00	12.38 ±0.00	65.12 ±0.00	89.07 ±0.00	15.23 ±0.00	53.28 ±0.00	63.52 ±0.00
IForest	12.76 ±0.00	75.19 ±0.00	32.10 ±0.00	28.96 ±0.00	72.98 ±0.00	21.11 ±0.00	16.65 ±0.00	48.14 ±0.00	15.79 ±0.00	70.90 ±0.00	94.56 ±0.00	99.78 ±0.00	42.01 ±0.00	69.50 ±0.00	97.09 ±0.00
PCA	18.19 ±0.00	77.63 ±0.00	18.16 ±0.00	18.61 ±0.00	61.13 ±0.00	30.49 ±0.00	15.93 ±0.00	53.42 ±0.00	22.76 ±0.00	89.69 ±0.00	**99.66** ±0.00	99.89 ±0.00	21.35 ±0.00	43.51 ±0.00	95.45 ±0.00
MAD-GAN	22.65 ±0.24	79.65 ±0.41	81.02 ±0.44	51.97 ±2.33	80.66 ±2.74	79.30 ±1.04	15.96 ±2.38	44.59 ±7.10	70.00 ±1.08	14.66 ±4.55	82.23 ±2.69	85.42 ±3.68	75.32 ±5.23	90.05 ±2.02	85.82 ±4.16
BeatGAN	28.70 ±0.59	84.31 ±0.45	87.79 ±4.96	52.04 ±5.32	83.84 ±2.12	75.32 ±2.97	30.91 ±1.82	**72.66** ±2.09	73.82 ±2.61	8.68 ±0.35	77.24 ±1.61	81.46 ±0.12	41.66 ±7.11	82.86 ±4.24	80.00 ±2.56
USAD	15.12 ±1.23	72.35 ±2.60	18.15 ±0.00	24.50 ±0.10	73.65 ±0.15	51.45 ±0.00	12.99 ±1.21	44.93 ±5.26	20.79 ±3.29	89.42 ±0.03	99.64 ±0.00	**99.94** ±0.00	20.03 ±0.03	61.73 ±0.04	58.23 ±0.00
THOC	12.51 ±2.38	70.69 ±5.93	76.88 ±2.12	41.89 ±4.94	78.94 ±1.94	79.13 ±1.57	18.70 ±1.58	49.50 ±3.92	71.77 ±1.20	14.57 ±12.17	71.76 ±4.37	81.49 ±0.10	31.40 ±5.53	66.92 ±2.37	62.69 ±10.61
FGANomaly	19.70 ±0.12	80.76 ±0.19	81.77 ±0.02	53.04 ±0.14	84.99 ±0.05	77.99 ±0.01	21.38 ±0.24	54.86 ±0.62	32.60 ±0.91	36.97 ±0.04	79.77 ±0.09	97.57 ±0.00	67.14 ±0.19	85.30 ±0.25	98.72 ±0.15
TranAD	23.78 ±3.18	86.14 ±2.69	85.52 ±2.78	57.59 ±3.05	86.88 ±1.29	79.35 ±1.91	22.71 ±1.29	55.22 ±2.91	29.22 ±1.82	37.24 ±0.20	80.92 ±0.07	97.57 ±0.00	65.62 ±6.18	77.97 ±9.15	95.35 ±6.80
A.T.	3.69 ±0.01	47.32 ±3.06	64.35 ±8.81	13.26 ±0.02	49.97 ±0.03	20.56 ±6.41	11.43 ±0.18	47.95 ±4.12	46.12 ±3.76	2.15 ±0.01	49.97 ±0.34	75.43 ±2.19	15.50 ±1.75	42.05 ±9.46	87.15 ±12.86
TimesNet	32.07 ±5.72	87.41 ±3.51	20.61 ±5.46	15.89 ±0.33	54.37 ±1.47	49.91 ±0.44	11.77 ±4.18	44.84 ±3.17	40.65 ±1.28	20.87 ±2.50	95.48 ±0.81	86.52 ±0.43	21.56 ±3.19	69.56 ±4.81	58.14 ±4.45
MEMTO	14.46 ±3.65	85.89 ±4.95	75.83 ±5.47	39.36 ±2.59	72.54 ±3.65	75.74 ±1.01	17.60 ±0.63	66.59 ±3.92	69.07 ±0.09	14.45 ±0.54	66.87 ±5.45	77.68 ±3.02	26.79 ±5.21	81.74 ±2.50	77.88 ±2.80
NPSR	28.08 ±1.54	83.91 ±1.28	83.79 ±3.51	42.17 ±2.69	84.37 ±1.77	78.67 ±2.78	22.44 ±0.87	63.41 ±0.47	78.75 ±0.56	39.84 ±8.33	91.60 ±1.06	99.36 ±0.14	95.15 ±0.52	97.93 ±0.28	95.87 ±0.14
SensitiveHUE	25.40 ±0.81	80.96 ±0.51	83.82 ±4.37	46.67 ±7.69	80.81 ±6.34	79.47 ±4.41	25.39 ±0.83	63.65 ±0.54	73.20 ±2.13	15.93 ±0.10	76.35 ±0.84	88.20 ±0.35	55.61 ±3.58	73.39 ±2.00	95.48 ±5.09
D^3R	50.48 ±1.64	88.47 ±1.84	85.97 ±1.85	38.99 ±2.34	73.51 ±3.18	80.04 ±3.34	19.11 ±0.34	54.51 ±0.40	69.43 ±0.00	60.34 ±8.44	87.42 ±2.63	99.56 ±0.22	95.05 ±0.28	97.81 ±0.32	98.80 ±0.03
DiffAD	13.23 ±0.14	72.53 ±1.53	71.40 ±0.44	37.65 ±0.18	68.20 ±0.08	78.46 ±0.38	22.44 ±0.52	58.69 ±0.77	69.48 ±0.06	40.46 ±0.50	81.31 ±0.24	97.37 ±0.16	23.58 ±0.24	51.32 ±0.65	68.93 ±0.01
IMDiffusion	41.57 ±2.08	84.77 ±3.55	86.51 ±3.31	43.86 ±2.09	71.73 ±2.23	78.87 ±0.72	20.25 ±0.91	52.54 ±1.43	69.04 ±0.73	40.33 ±0.23	81.64 ±0.14	97.27 ±0.20	50.05 ±1.51	89.00 ±0.73	86.50 ±1.10
PhysDiff	26.36 ±0.48	72.21 ±1.84	83.03 ±0.14	47.76 ±1.08	82.04 ±0.97	76.45 ±4.19	21.37 ±0.51	62.28 ±1.44	71.04 ±2.07	36.40 ±0.11	78.03 ±0.55	97.58 ±0.00	91.57 ±2.94	97.34 ±0.71	95.44 ±1.94
EnsDiffAD	27.90 ±2.56	74.93 ±3.09	81.48 ±0.80	39.53 ±1.63	75.93 ±0.02	77.68 ±2.38	18.40 ±0.53	57.79 ±0.35	70.44 ±0.14	25.51 ±1.89	79.54 ±0.36	88.12 ±0.92	38.83 ±0.39	78.09 ±1.27	84.69 ±0.31
ICDiffAD	16.31 ±0.38	76.74 ±1.42	87.14 ±2.35	43.89 ±0.10	78.28 ±0.24	80.10 ±0.02	21.05 ±0.03	53.02 ±0.08	72.82 ±0.00	37.54 ±0.02	82.12 ±0.02	97.57 ±0.00	81.07 ±3.26	95.20 ±1.32	88.85 ±2.72
Ours	**56.20** ±1.63	**95.24** ±0.35	**91.40** ±0.24	**64.82** ±1.04	**90.57** ±0.48	**81.23** ±0.48	**34.62** ±1.07	70.96 ±0.79	**80.45** ±0.43	**92.17** ±4.56	98.49 ±1.48	99.66 ±0.30	**99.70** ±0.11	**99.94** ±0.03	**99.84** ±0.10

A.5.2 HYPERPARAMETER SENSITIVITY ANALYSIS

In this section, we analyze the sensitivity of our model to three key hyperparameters: i) the number of normal prototypes K, ii) the number of DDIM inversion steps M, and iii) the number of selected trajectory states $\mathbf{k}$ in Top-$\mathbf{k}$ aggregation. Each parameter is varied while the others are fixed. Detailed parameter settings are provided in Appendix A.4.3.

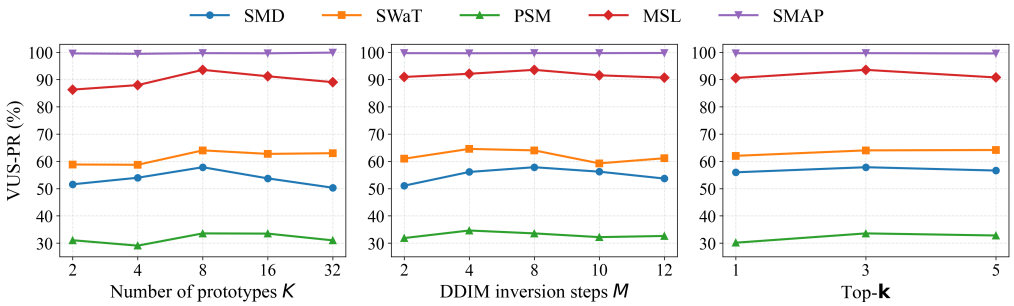


Figure 4: Hyperparameter sensitivity in terms of VUS-PR on five datasets.

As shown in Figure 4 (left), performance varies moderately with the number of prototypes. Using too few prototypes may merge distinct normal modes, whereas excessive partitioning may reduce the statistical reliability of each prototype-specific trajectory prior. The relatively stable results over a broad range of K indicate that *PiAD* is not highly sensitive to the precise prototype count.

Figure 4 (middle) shows that performance remains generally stable across different numbers of DDIM inversion steps. Increasing M initially provides a denser characterization of the inversion trajectory, but further increases yield limited additional information and may introduce redundant states. This suggests that a sparse inversion trajectory is sufficient to capture useful anomaly evidence.

As shown in Figure 4 (right), selecting multiple trajectory states generally provides more robust performance than relying on a single state. However, including too many states may incorporate less discriminative deviations and dilute the anomaly evidence. Overall, the results support selective multistate aggregation while demonstrating limited sensitivity to the exact value of $\mathbf{k}$.

A.6 CODE AVAILABILITY AND REPRODUCIBILITY

To support reproducibility, we provide a simplified implementation of the core *PiAD* model at https://anonymous.4open.science/r/PiAD-D07C, covering its primary training and inference mechanisms. The complete codebase will be made publicly available immediately upon the paper's acceptance.