SC-JEPA: Stabilizing Latent Predictive Learning for Time-Series Anomaly Prediction

Yanan He [1] **Yunshi Wen** [2] **Xin Wang** [3]
Tengfei Ma [3]

Abstract

Time-series anomaly prediction aims to forecast future system failures before they emerge, making latent predictive models such as JEPA a promising framework for capturing precursor dynamics. However, directly applying continuous self-distillation to time-series data is often unstable and can lead to representation collapse, while also struggling to model precursors evolving at different temporal scales. To address this, we propose **SC-JEPA**, a new JEPA-based framework to model time-series anomaly prediction in a discretized predictive state space. It introduces a soft codebook bottleneck to stabilize latent predictive learning and encourage regime-level structure in the learned representations. Building on this stabilized latent space, we further design a multi-resolution predictive objective to capture precursor patterns at different temporal scales. Experiments on five benchmarks show that SC-JEPA achieves strong early-warning performance.

1. Introduction

Multivariate time series underpin many safety-critical systems, from industrial plants to server operations, where failures can propagate rapidly and incur substantial costs. In such settings, the key challenge is not merely detecting anomalies after they occur, but anticipating them early enough for intervention. This task, known as *anomaly prediction*, differs fundamentally from both anomaly detection and forecasting (Park et al., 2025). Detection is inherently reactive and identifies faults only after they manifest (Ruff et al., 2018; Xu et al., 2022), while forecasting focuses on regressing future observations or values (Zhou et al., 2021; Nie et al., 2023). Anomaly prediction, however, aims to isolate latent precursor signals within the current window that foreshadow impending instability (Jhin et al., 2023; Zhao et al., 2024; Park et al., 2025). Consequently, the primary objective is capturing the evolving system risk rather than accurately reproducing the exact future trajectory.

A natural approach to this problem is to model how system states evolve over time. Joint-Embedding Predictive Architectures (JEPA) (Assran et al., 2023) offer a compelling paradigm by learning to predict future representations directly in latent space. This objective aligns naturally with anomaly prediction, where impending failures are often marked by subtle regime shifts and semantic state drifts rather than raw signal changes. By optimizing latent predictability, JEPA provides a task-aligned framework for anticipating system instability (Ennadir et al., 2025).

However, applying JEPA-style predictive learning to anomaly prediction faces two key challenges. First, predictive learning in a continuous latent space is vulnerable to representation collapse and instability (Chen & He, 2021; Grill et al., 2020; Jing et al., 2022; Bardes et al., 2022). The encoder may map diverse inputs to a constant vector or experience partial dimensional collapse, stripping the feature space of discriminative capacity (Tian et al., 2021). The noisy, continuous nature of time-series data further exacerbates this problem. Moreover, in real-world systems, anomaly precursors typically emerge across heterogeneous time scales, ranging from abrupt shocks to slowly accumulating drifts (Wang et al., 2024; Liu et al., 2022a; Chen et al., 2024; Wang et al., 2023). A single-scale predictive design therefore struggles to capture the full spectrum of precursor dynamics. In this work, we argue that these challenges stem from a common source: the lack of structure in the predictive latent space. When representations are unconstrained, the model must simultaneously learn stable dynamics, meaningful abstractions, and scale-aware patterns, often leading to degenerate solutions.

To address these challenges, we propose SC-JEPA (**S**oft **C**odebook **JEPA**), a framework that introduces a discretized predictive state space for time-series anomaly prediction. The key idea of SC-JEPA is to constrain latent prediction through a ***soft codebook bottleneck***, which maps continuous

[1] Yale University [2] Rensselaer Polytechnic Institute [3] Stony Brook University. Correspondence to: Tengfei Ma <tengfei.ma@stonybrook.edu>.

ICML 2026 Workshop on Forecasting as a New Frontier of Intelligence.

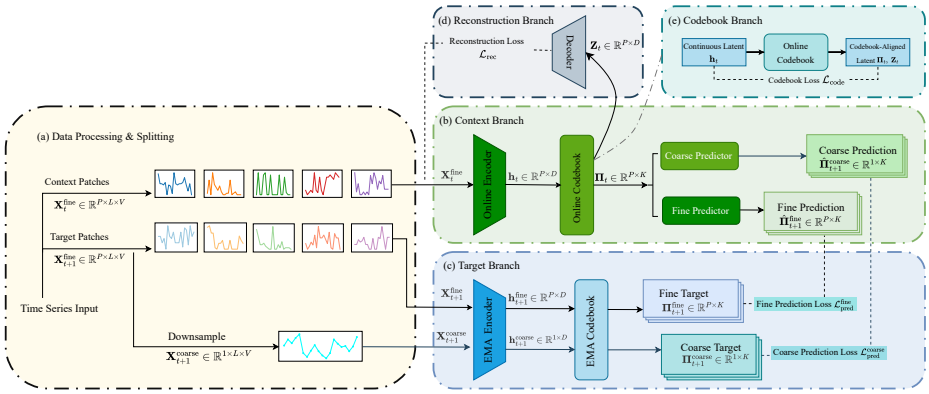


Figure 1. Overview of the SC-JEPA framework for time series representation learning.

latent features to distributions over a finite set of prototypes. This mechanism serves two purposes. This design imposes a structured geometry on the latent space, which (i) empirically stabilizes training and (ii) encourages representations to organize into regime-level states. Such structure is particularly well suited to anomaly prediction, where failures often correspond to transitions between underlying system regimes. Building on this stabilized and more structured latent space, we further introduce a ***multi-resolution*** predictive design to model system dynamics over different temporal scales. By jointly predicting fine- and coarse-grained future states, the model captures short-term fluctuations and long-term trends, enabling more robust early anomaly precursor identification.

2. Method

2.1. Problem Setup

Consider a multivariate time series $\mathbf{x} \in \mathbb{R}^{T \times V}$. Following prior anomaly prediction settings (Jhin et al., 2023; Park et al., 2025; Zhao et al., 2024), we partition $\mathbf{x}$ into non-overlapping windows $\{\mathbf{W}_t\}_{t=1}^{N}$, where $\mathbf{W}_t \in \mathbb{R}^{T_w \times V}$. The task is to predict whether the next window $\mathbf{W}_{t+1}$ is anomalous given the current window $\mathbf{W}_t$. We define the binary label $Y_{t+1} = 1$ if $\mathbf{W}_{t+1}$ contains at least one anomalous time point, and $Y_{t+1} = 0$ otherwise. Thus, the model uses only $\mathbf{W}_t$ as precursor context to predict whether an anomaly will occur in the next window.

Framework Overview. This proactive setting is challenging because precursor cues are often weak, non-stationary, and scale-variant, making stable self-supervised prediction difficult. We propose a new JEPA framework to solve this challenging task. (1) To expose scale-variant precursors, we construct dual-resolution inputs that disentangle local volatility from global trends (Section 2.2). As shown in Figure 1, SC-JEPA adopts an asymmetric teacher–student design, where an online branch predicts future latent states from the current window and a momentum-updated target branch provides supervision. As shown in Figure 1, SC-JEPA adopts an asymmetric teacher–student design, where an online branch predicts future latent states from the current window and a momentum-updated target branch provides supervision. (2) To stabilize predictive learning, we introduce a soft codebook bottleneck that maps continuous features to distributions over a finite set of prototypes (Section 2.3), constraining the latent space and promoting regime-level structure for multi-resolution prediction.

2.2. Input Formulation

To make scale-variant precursor patterns learnable under a unified predictive objective, we build dual-scale views from each raw window. Specifically, we first apply RevIN (Kim et al., 2022) to $\mathbf{W}_t$ to obtain the normalized window $\hat{\mathbf{W}}_t$, and cache the affine statistics for inversion in the reconstruction loss. We then derive two views at different resolutions:

Fine View (Patching). We tokenize $\hat{\mathbf{W}}_t$ into P patches of length L (where $T_w = P \cdot L$), forming the fine-grained representation: $\mathbf{X}_t^{\text{fine}} = \text{Patch}(\hat{\mathbf{W}}_t) \in \mathbb{R}^{P \times L \times V}$.

Coarse View (Downsampling). We generate a global view by averaging every P time points (Appendix B.2), yielding a length-L sequence treated as a single patch: $\mathbf{X}_t^{\text{coarse}} = \text{DownAvg}_P(\hat{\mathbf{W}}_t) \in \mathbb{R}^{1 \times L \times V}$.

The online branch takes only the fine view, while the target branch encodes both resolutions, introducing the cross-resolution asymmetry that underlies our architecture.

2.3. Architecture

SC-JEPA couples a shared encoder, a soft codebook, a dual-resolution predictor, and an auxiliary decoder.

Encoder. Weak and scale-variant precursors call for representations that are predictable and comparable across resolutions. We use a shared encoder $\mathcal{E}$ to map channel-independent inputs into a compact latent space, enabling both fine- and coarse-grained inputs to be processed with

shared parameters in the same embedding space.

(i) **Online encoder ($\mathcal{E}_\theta$):** $\mathbf{h}_t = \mathcal{E}_\theta(\mathbf{X}_t^{\text{fine}})$. $\mathcal{E}_\theta$ encodes only the fine-grained view and is trained to align with multi-scale targets from the momentum teacher, encouraging the representation to infer global context from local evidence.

(ii) **EMA encoder ($\mathcal{E}_\xi$):** A momentum-updated copy of the online encoder that processes both fine and coarse views to produce multi-scale targets: $\mathbf{h}_{t+1}^{\text{fine}} = \mathcal{E}_\xi(\mathbf{X}_{t+1}^{\text{fine}})$ and $\mathbf{h}_{t+1}^{\text{coarse}} = \mathcal{E}_\xi(\mathbf{X}_{t+1}^{\text{coarse}})$.

However, continuous encoder features often entangle nuisance variations. To mitigate this, we introduce a soft codebook that projects features onto a finite set of prototypes, capturing regime-like patterns while imposing geometric constraints for stability and expressiveness.

Codebook. Formally, we implement this projection via a differentiable bottleneck $\mathcal{Q}$ that maps continuous encoder features into a finite set of learnable prototypes $\{\mathbf{c}_k\}_{k=1}^K \subset \mathbb{R}^D$. For each patch representation $\mathbf{h}_{t,i} \in \mathbb{R}^D$, $\mathcal{Q}$ outputs a soft code distribution $\mathbf{p}_{t,i} \in \Delta^{K-1}$, where $\Delta^{K-1} := \{\mathbf{p} \in \mathbb{R}^K : \mathbf{p} \succeq 0,\ \mathbf{1}^\top \mathbf{p} = 1\}$. We parameterize $\mathbf{p}_{t,i}$ via temperature-scaled cosine similarity:

$$p_{t,i,k} = \frac{\exp\left\{\left\langle \bar{\mathbf{h}}_{t,i},\, \bar{\mathbf{c}}_k \right\rangle / \tau\right\}}{\sum_{j=1}^K \exp\left\{\left\langle \bar{\mathbf{h}}_{t,i},\, \bar{\mathbf{c}}_j \right\rangle / \tau\right\}}, \tag{1}$$

where $\bar{\mathbf{h}}_{t,i}$ and $\bar{\mathbf{c}}_k$ denote ℓ_2-normalized vectors. We stack these patch-wise distributions to form the window-level code sequence $\mathbf{\Pi}_t := [\mathbf{p}_{t,1}; \ldots; \mathbf{p}_{t,P}] \in \mathbb{R}^{P \times K}$, which serves as the predictor input. In parallel, we compute the expected code embedding $\mathbf{z}_{t,i} := \sum_{k=1}^K p_{t,i,k}\mathbf{c}_k \in \mathbb{R}^D$, which is used by the auxiliary decoder for reconstruction.

Since $\mathbf{p}_{t,i} \in \Delta^{K-1}$, the soft embedding $\mathbf{z}_{t,i} = \sum_k p_{t,i,k}\mathbf{c}_k$ constitutes a convex combination of finitely many prototypes and therefore remains within the bounded convex hull of $\{\mathbf{c}_k\}$. This boundedness imposes an explicit geometric constraint on the latent space, improving training stability and promoting recurring patterns; Appendix B.3.1 gives a formal upper bound. Appendix B.3.2 further derives a strictly positive lower bound on the batch variance, $\mathrm{Tr}(\mathrm{Cov}(\mathbf{z})) > 0$, providing a sufficient non-collapse certificate. Together, these bounds ensure that $\mathbf{\Pi}_t$ provides a compact, regime-level description of the window that is both stable and expressive.

Predictor. With the latent dynamics discretized into a stable code sequence $\mathbf{\Pi}_t$, the remaining challenge is to forecast the system's evolution. To capture anomalies across conflicting temporal scales, we employ a dual-branch predictor. Both branches operate on the same history $\mathbf{\Pi}_t$ while targeting distinct future resolutions.

(i) **Fine Predictor (Micro-Dynamics):** To capture high-frequency volatility and local point anomalies, a Transformer maps $\mathbf{\Pi}_t$ to a sequence of fine-grained predictions $\hat{\mathbf{\Pi}}_{t+1}^{\text{fine}} \in \mathbb{R}^{P \times K}$. This branch preserves the patch-wise resolution, capturing short-term fluctuations and ensuring that transient disturbances are not smoothed out.

(ii) **Coarse Predictor (Macro-Dynamics):** To distill low-frequency trends and global distribution shifts, a learnable query token $\mathbf{q}$ aggregates the entire history into a single global prediction $\hat{\mathbf{\Pi}}_{t+1}^{\text{coarse}} \in \mathbb{R}^{1 \times K}$ via cross-attention: $\hat{\mathbf{\Pi}}_{t+1}^{\text{coarse}} = \mathrm{CrossAttn}\big(\mathbf{q}, \mathbf{\Pi}_t\big)$. This mechanism forces the model to abstract away local noise and focus on slower, system-level shifts.

For supervision, the EMA branch generates the corresponding multi-scale targets $\mathbf{\Pi}_{t+1}^{\text{fine}}$ and $\mathbf{\Pi}_{t+1}^{\text{coarse}}$ using the shared codebook $\mathcal{Q}$.

Decoder. Because prediction alone can encourage overly abstract codes, an auxiliary decoder $\mathcal{D}$ reconstructs the input patches $\mathbf{X}_t^{\text{fine}}$ from the online soft-quantized embeddings $\mathbf{z}_t$ to retain signal-level semantics. The reconstruction is evaluated after inverting RevIN using the cached statistics.

3. Experiments

3.1. Experimental Setup

Datasets and Baselines. We evaluate on five multivariate anomaly detection benchmarks (MSL, SMAP, SWaT, PSM, and SMD) (Hundman et al., 2018; Goh et al., 2017; Su et al., 2019). We compare SC-JEPA against five categories of baselines: generalist sequence models: LSTM-VAE (Park et al., 2018), iTransformer (Liu et al., 2024), PatchTST (Nie et al., 2023), TS2Vec (Yue et al., 2022); LLM: Qwen2.5-3B (Team, 2024); hierarchical models: TimesNet (Wu et al., 2023), MICN (Wang et al., 2023), Pathformer (Chen et al., 2024); anomaly prediction methods: PAD (Jhin et al., 2023), A2P (Park et al., 2025), FCM (Zhao et al., 2024); and JEPA-based models: TS-JEPA (Ennadir et al., 2025).

3.2. Results

Main results. Table 6 reports anomaly prediction results on five benchmarks. SC-JEPA achieves the best AUC and F1 across all datasets, consistently outperforming all baselines under the same protocol. General-purpose and hierarchical time-series models achieve competitive results on some datasets, but are not specifically designed for future abnormal transition modeling, while Qwen2.5-3B shows reasonable AUC but unstable F1. Among anomaly prediction methods, PAD, A2P, and FCM are more competitive, yet SC-JEPA remains consistently stronger. Compared

Table 1. Full results on five benchmark datasets. Baselines are grouped by modeling characteristics. All metrics are reported in percentage (%) as the mean over 5 runs with different random seeds.

Category	Models	MSL				SMAP				SWaT				PSM				SMD			
		F1	AUC	Prec	Rec	F1	AUC	Prec	Rec	F1	AUC	Prec	Rec	F1	AUC	Prec	Rec	F1	AUC	Prec	Rec
Generalist	iTransformer (Liu et al., 2024)	27.25	64.61	16.71	75.40	33.00	60.91	22.67	64.32	70.49	82.10	98.17	55.00	54.12	63.09	37.37	98.29	14.20	57.94	8.11	62.34
	PatchTST (Nie et al., 2023)	26.98	60.43	**75.00**	16.45	30.06	61.62	21.76	49.06	70.64	81.93	94.03	57.58	58.17	75.76	**64.68**	52.86	11.79	47.76	6.58	57.02
	TS2Vec (Yue et al., 2022)	23.48	64.86	31.45	18.82	32.81	61.48	23.81	52.81	67.00	83.76	80.95	57.17	48.43	72.13	32.39	95.96	14.78	56.28	8.83	50.43
LLM	Qwen2.5-3B (Team, 2024)	21.24	55.66	17.82	26.67	0.34	63.61	1.67	0.19	17.29	56.75	41.82	14.43	47.44	65.65	50.44	46.57	13.00	60.66	8.29	31.06
Hierarchical	TimesNet (Wu et al., 2023)	28.44	59.70	30.31	36.59	27.78	57.01	17.98	62.81	67.20	83.24	90.87	54.42	52.64	56.12	35.76	**99.71**	12.51	56.33	7.64	35.96
	Pathformer (Chen et al., 2024)	21.69	61.86	13.14	68.24	23.58	58.67	14.10	**72.12**	14.90	59.87	9.20	40.98	51.77	54.41	35.04	99.10	14.32	56.49	9.41	36.60
Anomaly Prediction	PAD (Jhin et al., 2023)	21.67	55.97	12.19	**97.78**	26.92	59.79	18.15	55.00	69.75	83.71	81.54	**62.79**	57.82	73.84	57.54	58.57	12.52	58.53	6.71	**93.40**
	A2P (Park et al., 2025)	23.10	59.43	13.92	74.12	21.68	60.37	15.88	38.85	70.18	81.92	**99.10**	54.33	50.71	57.01	37.16	80.00	15.21	58.87	12.23	22.13
	FCM (Zhao et al., 2024)	24.57	61.09	14.25	90.00	30.60	61.85	**25.25**	41.41	69.09	81.66	94.25	54.83	52.34	57.25	37.00	89.85	19.25	60.44	15.59	28.94
JEPAs	TS-JEPA (Ennadir et al., 2025)	25.49	60.33	17.10	56.84	26.57	57.38	18.23	49.73	71.95	80.33	84.32	62.76	53.32	66.10	36.95	96.29	7.49	49.53	**19.11**	20.43
	SC-JEPA	**33.58**	**66.08**	35.87	40.80	**33.64**	**65.41**	24.24	56.02	**72.89**	**84.95**	98.00	58.05	**61.61**	**77.85**	55.01	72.00	**20.82**	**62.29**	18.18	31.49

Table 2. Ablation results on five benchmark datasets (%).

Model Variant	MSL		SMAP		SWaT		PSM		SMD	
Metric	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
w/o KL Divergence	29.01	53.90	28.71	60.19	67.33	79.95	54.93	70.83	12.63	53.61
w/o Reconstruction	25.96	52.68	25.15	52.80	14.77	53.30	53.03	69.84	10.18	50.44
w/o Predictive Objective	28.88	53.17	27.05	59.81	59.47	78.60	58.89	70.17	10.04	50.24
w/o Codebook Loss	31.62	58.93	31.53	62.06	72.64	82.63	60.83	75.85	12.82	57.36
w/o Codebook Module	21.82	43.02	21.69	51.00	11.51	50.00	53.03	46.61	12.45	50.00
w/o Temporal Downsampling	28.77	63.16	30.39	61.34	71.62	83.60	60.79	71.72	15.50	60.93
Full Model	**33.58**	**66.08**	**33.64**	**65.41**	**72.89**	**84.95**	**61.61**	**77.85**	**20.82**	**62.29**

Table 3. Variance ratio comparison on three datasets. Lower values indicate less representation collapse; gray text shows the relative decrease versus the variant without codebook.

Dataset	Method	Top-1 ↓	Top-5 ↓	Top-10 ↓
PSM	without codebook	0.5334	1.0000	1.0000
	with codebook (ours)	0.1765 (↓66.9%)	0.4347 (↓56.5%)	0.6778 (↓32.2%)
MSL	without codebook	0.3292	0.9088	0.9154
	with codebook (ours)	0.1804 (↓45.2%)	0.5194 (↓42.8%)	0.6607 (↓27.8%)
SWaT	without codebook	0.3251	0.9271	0.9344
	with codebook (ours)	0.1143 (↓64.8%)	0.3190 (↓65.6%)	0.4041 (↓56.8%)

with TS-JEPA, the improvement further suggests the benefit of combining multi-resolution prediction with a structured codebook for modeling diverse precursor dynamics. Full results are provided in Appendix D.

3.3. Model Analysis

3.3.1. Ablation Study

To verify that the improvement comes from the overall design rather than a single module, we conduct a systematic ablation study in Table 2. The predictive objective drives temporal dependency learning: removing it degrades performance, and KL outperforms MSE via stronger distributional alignment. Multi-resolution modeling is also crucial, as it captures coarse-grained precursors missed by single-scale models. Removing the reconstruction objective degrades performance, suggesting its role in preserving informative representations and preventing degeneracy.

Among all variants, removing the *soft codebook bottleneck* causes the largest degradation, with performance dropping to near-random, while removing auxiliary regularizers has a much smaller effect, highlighting its role as the main source of structure and stability. This naturally raises a question: what stabilizing effect does the bottleneck introduce that the auxiliary losses alone do not?

The ablations offer an empirical explanation consistent with our formulation. With the codebook, self-distillation becomes KL matching on the simplex (Eq. 1), so the predictor follows a bounded distributional trajectory rather than unconstrained regression in $\mathbb{R}^D$. Together with slowly varying EMA targets, the bounded code space provides a more stable reference for self-distillation. Without the bottleneck, this constraint disappears, making KL alignment ill-conditioned and leading to near-collapse (Zhou et al., 2022). By contrast, removing only the auxiliary codebook losses causes a smaller drop, suggesting that they mainly provide an additional performance gain rather than the stabilizing effect.

3.3.2. Variance Analysis.

To further examine representation collapse, we compare the full model with a variant without the codebook module on three benchmark datasets. We report top-k variance ratios for $k \in \{1, 5, 10\}$, where higher values indicate stronger collapse. Table 3 shows that removing the codebook consistently increases these ratios, while the full model keeps them much lower, providing direct evidence that the codebook mitigates representation collapse.

4. Conclusion

This work presents SC-JEPA, a new framework for time-series anomaly prediction. At its core, SC-JEPA highlights the importance of structuring the latent space in predictive learning for time series. It introduces a soft codebook bottleneck that constrains the latent prediction space, stabilizes predictive learning on continuous time series, and mitigates representation collapse without explicit negative sampling. Its multi-resolution predictive design further captures precursor patterns across temporal scales. Across five real-world benchmarks, SC-JEPA delivers strong and consistent

anomaly prediction performance.

References

Assran, M., Duval, Q., Misra, I., Bojanowski, P., Vincent, P., Rabbat, M., LeCun, Y., and Ballas, N. Self-supervised learning from images with a Joint-Embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

Bardes, A., Ponce, J., and LeCun, Y. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Bardes, A., Garrido, Q., Ponce, J., Chen, X., Rabbat, M., LeCun, Y., Assran, M., and Ballas, N. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Chen, X. and He, K. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15750–15758, 2021.

Chen, Y., Zhang, Z., Wu, H., Wang, L., Zhao, S., and Long, M. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *International Conference on Learning Representations*, 2024.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory*. Wiley-Interscience, Hoboken, N.J., 2nd edition, 2006.

Ennadir, S., Golkar, S., and Sarra, L. Joint embeddings go temporal. *arXiv preprint arXiv:2509.25449*, 2025.

Goh, J., Adepu, S., Junejo, K. N., and Mathur, A. A dataset to support research in the design of secure water treatment systems. In Havarneanu, G., Setola, R., Nassopoulos, H., and Wolthusen, S. (eds.), *Critical Information Infrastructures Security*, pp. 88–99, Cham, 2017. Springer International Publishing.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z., Azar, M. G., et al. Bootstrap your own latent-a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284, 2020.

Hundman, K., Constantinou, V., Laporte, C., Colwell, I., and Soderstrom, T. Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 387–395, 2018.

Jhin, S. Y., Lee, J., and Park, N. Precursor-of-anomaly detection for irregular time series. In *Proceedings of the 29th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 917–929, 2023.

Jing, L., Vincent, P., LeCun, Y., and Tian, Y. Understanding dimensional collapse in contrastive self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022.

Lee, D., Malacarne, S., and Aune, E. Vector quantized time series generation with a bidirectional prior model. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, 2023.

Lee, D., Malacarne, S., and Aune, E. Explainable time series anomaly detection using masked latent generative modeling. *Pattern Recognition*, 156:110826, 2024.

Liu, M., Zeng, A., Chen, M., Xu, Z., Lai, Q., Ma, L., and Xu, Q. SCINet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022a.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2022b.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. iTransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

Park, D., Hoshi, Y., and Kemp, C. C. A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3):1544–1551, 2018.

Park, M.-Y., Lee, W.-J., Kim, S. T., and Park, G.-M. When will it fail?: Anomaly to prompt for forecasting future anomalies in time series. *arXiv preprint arXiv:2506.23596*, 2025.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. Deep one-class classification. In *International Conference on Machine Learning*, pp. 4393–4402, 2018.

Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., and Pei, D. Robust anomaly detection for multivariate time series through stochastic recurrent neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Team, Q. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Tian, Y., Chen, X., and Ganguli, S. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, pp. 10268–10278. PMLR, 2021.

Van Den Oord, A., Vinyals, O., et al. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., and Chu, Y. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *International Conference on Learning Representations*, 2023.

Wang, S., Wu, H., Shi, X., Hu, T., Luo, H., Ma, L., Zhang, J. Y., and Zhou, J. TimeMixer: Decomposable multi-scale mixing for time series forecasting. In *International Conference on Learning Representations*, 2024.

Wen, Y., Ma, T., Weng, L., Nguyen, L., and Julius, A. A. Abstracted shapes as tokens-a generalizable and interpretable model for time-series classification. *Advances in Neural Information Processing Systems*, 37:92246–92272, 2025.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. TimesNet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

Xu, J., Wu, H., Wang, J., and Long, M. Anomaly transformer: Time series anomaly detection with association discrepancy. In *International Conference on Learning Representations*, 2022.

Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., and Xu, B. TS2Vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8980–8987, 2022.

Zhao, S., Wang, W., Xu, H., Yu, Z., Wen, Q., Wang, G., and Pang, G. Abnormality forecasting: Time series anomaly prediction via future context modeling. *arXiv preprint arXiv:2410.12206*, 2024.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 2021.

Zhou, J., Wei, C., Wang, H., Shen, W., Xie, C., Yuille, A., and Kong, T. Image BERT pre-training with online tokenizer. In *International Conference on Learning Representations*, 2022.

A. Related Works

Anomaly Prediction. Time-series research has largely focused on forecasting and detection. Forecasting methods predict future values (Zhou et al., 2021; Liu et al., 2024), while detection methods score the current window under reactive protocols (Ruff et al., 2018; Xu et al., 2022). More recently, anomaly prediction has emerged as an early-warning setting that aims to anticipate future abnormality or failure risk (Jhin et al., 2023; Zhao et al., 2024; Park et al., 2025). However, many existing approaches remain tied to input-space objectives or detector adaptations, motivating representation-space predictive models that emphasize latent dynamics over local noise. Another line of work related to our method is vector-quantized (VQ) representation learning (Van Den Oord et al., 2017), which has been adapted for time-series anomaly detection (Lee et al., 2023; 2024; Wen et al., 2025); however, our method differs essentially in the target task, representation granularity, and predictive framework.

Joint Embedding Predictive Architectures. Self-supervised learning for time series is commonly built on masked modeling or contrastive learning, as in PatchTST (Nie et al., 2023) and TS2Vec (Yue et al., 2022). Joint-Embedding Predictive Architectures (JEPA) instead predict future representations directly in latent space, as shown by I-JEPA and V-JEPA (Assran et al., 2023; Bardes et al., 2024), and recent efforts have begun to extend this idea to time series (Ennadir et al., 2025). However, JEPA-style predictive world models remain less explored for complex time-series systems, and stabilizing latent-space self-distillation on continuous, non-stationary signals is still challenging. Meanwhile, multi-scale temporal modeling has been explored in Pyraformer (Liu et al., 2022b), TimeMixer (Wang et al., 2024), MICN (Wang et al., 2023), and Pathformer (Chen et al., 2024), but its role in anomaly prediction via predictive latent dynamics remains underexplored.

B. Methodological Details

B.1. Notations

Table 4. Categorized notations.

Category	Symbol	Description	Category	Symbol	Description
Dimensions	T, V	Time series size ($T \times V$)	Data Inputs	$\mathbf{x}$	Input time series
	T_w	Window length		$\hat{\mathbf{W}}_t$	Normalized window
	N	Number of windows		$\mathbf{X}_t^{\text{fine}}$	Fine view (patches)
	P	Num. patches ($T_w = P \cdot L$)		$\mathbf{X}_t^{\text{coarse}}$	Coarse view (downsampled)
	L	Patch length	Representations	$\mathbf{h}_t$	Encoder latent rep.
	K	Codebook size		$\mathbf{c}_k$	Learnable prototype
	D	Embedding dimension		$\mathbf{p}_{t,i}$	Code-assignment prob.
Hyperparams	τ	Soft-max temperature		$\mathbf{\Pi}_t$	Code sequence
	$\lambda_{\dots}$	Loss balancing weights		$\mathbf{z}_{t,i}$	Soft-quantized embedding
	γ	MSE local weight		$\mathbf{q}$	Coarse query token
Architecture	$\mathcal{E}_\theta$	Online encoder			
	$\mathcal{E}_\xi$	EMA encoder	Outputs	$\hat{\mathbf{\Pi}}^{\text{fine}}$	Predicted fine code dist.
	$\mathcal{Q}$	Codebook bottleneck		$\hat{\mathbf{\Pi}}^{\text{coarse}}$	Predicted coarse code dist.
	$\mathcal{D}$	Reconstruction decoder		$\hat{\mathbf{z}}_{t,i}$	Predicted embedding
	sg	Stop-gradient operator		$\hat{\mathbf{X}}_{t,i}$	Reconstructed patch
			Objectives	$\mathcal{L}_{\text{pred/code}}$	Predictive / Code losses
				$\mathcal{L}_{\text{rec/total}}$	Recon. / Total losses

B.2. Time-Axis Downsampling

DownAvg operator. Given a normalized window $\hat{\mathbf{W}}_t \in \mathbb{R}^{T_w \times V}$ with $T_w = P \cdot L$, we define $\mathrm{DownAvg}_P(\cdot)$ as a time-axis downsampling operator that averages every P consecutive time points (stride P) for each variable:

$$\mathbf{X}_t^{\text{coarse}} = \mathrm{DownAvg}_P(\hat{\mathbf{W}}_t) \in \mathbb{R}^{1 \times L \times V}. \tag{2}$$

Equivalently, for within-patch index $\ell \in [L]$ and variable $v \in [V]$:

$$\left[\mathbf{X}_t^{\text{coarse}}\right]_{1,\ell,v} = \frac{1}{P} \sum_{j=1}^{P} \hat{\mathbf{W}}_t[(\ell - 1)P + j,\ v]. \tag{3}$$

B.3. Theoretical Analysis of Latent Stability

In this subsection, we provide the formal proofs for the stability properties of the SC-JEPA latent space. We demonstrate how the soft codebook acts as a geometric constraint that explicitly bounds representation drift. We provide two complementary guarantees: (i) an upper bound on representation drift (stability), and (ii) a lower bound that rules out the trivial constant solution (non-collapse).

B.3.1. Stability Upper Bound

This part proves an explicit upper bound on representation drift in embedding space. The argument follows a short chain: we first fix a bounded latent geometry via the soft codebook, then establish a probability-to-geometry bridge ($\ell_1 \to \ell_2$), and finally combine this bridge with EMA smoothness and predictive alignment to obtain Theorem B.3.

Preliminaries. Continuous forecasting models can exhibit collapse or explosion, making the latent geometry unstable. The soft codebook eliminates this degree of freedom by restricting every soft-quantized embedding to the convex hull of a finite prototype set. In particular, the radius constant M defined below will be the only scale factor that propagates into the final drift bound.

Definition B.1 (Soft Code Map)**.** Let $K \geq 2$ be the number of discrete prototypes and $D \geq 1$ be the embedding dimension. Let $\mathbf{C} \in \mathbb{R}^{K \times D}$ be the codebook matrix where the k-th **row** corresponds to the prototype $\mathbf{c}_k$. We define the probability simplex $\Delta^{K-1} := \{\mathbf{p} \in \mathbb{R}^K : \mathbf{p} \succeq \mathbf{0},\ \mathbf{1}^\top \mathbf{p} = 1\}$.

The **soft-quantization map** $z : \Delta^{K-1} \to \mathbb{R}^D$ is defined as the expected embedding:

$$z(\mathbf{p}) := \mathbf{C}^\top \mathbf{p} = \sum_{k=1}^{K} p_k \mathbf{c}_k. \tag{4}$$

We further define the **geometric radius** M as the maximum Euclidean norm of any prototype...

Lipschitz Bound. Training is performed in probability space (via KL divergence), whereas drift and anomaly scores are measured in embedding space. We therefore need a simple link that converts discrepancies between assignment distributions into Euclidean changes after soft quantization. Lemma B.2 provides exactly this bridge, with the codebook radius M as the controlling constant.

Lemma B.2 ($\ell_1 \to \ell_2$ Lipschitzness)**.** *For any two assignment distributions $\mathbf{p}, \mathbf{q} \in \Delta^{K-1}$, the Euclidean distance between their soft-quantized embeddings is bounded by their ℓ_1 distance, scaled by the codebook radius M:*

$$\|z(\mathbf{p}) - z(\mathbf{q})\|_2 \leq M \|\mathbf{p} - \mathbf{q}\|_1. \tag{5}$$

Proof. We expand the Euclidean distance and apply the triangle inequality.

$$\begin{aligned}\|z(\mathbf{p}) - z(\mathbf{q})\|_2 &= \left\| \sum_{k=1}^{K} p_k \mathbf{c}_k - \sum_{k=1}^{K} q_k \mathbf{c}_k \right\|_2 \\ &= \left\| \sum_{k=1}^{K} (p_k - q_k) \mathbf{c}_k \right\|_2 \\ &\leq \sum_{k=1}^{K} |p_k - q_k| \cdot \|\mathbf{c}_k\|_2.\end{aligned}$$

Since $\|\mathbf{c}_k\|_2 \leq M$ for all k, we have:

$$\sum_{k=1}^{K} |p_k - q_k| \cdot \|\mathbf{c}_k\|_2 \leq M \sum_{k=1}^{K} |p_k - q_k| = M\|\mathbf{p} - \mathbf{q}\|_1.$$

□

With this bridge in place, the remaining step is to control the probability-space differences themselves. Predictive alignment bounds the deviation between targets and predictions at each time step, while EMA smoothness bounds how fast the target moves. Combining these two controls yields the following stability bound.

Theorem B.3 (Stability Bound)**.** *Let $\{\hat{\mathbf{p}}_t\}$ and $\{\mathbf{p}_t\}$ be the sequences of predicted and target (EMA) distributions, respectively, and let $\hat{\mathbf{z}}_t = z(\hat{\mathbf{p}}_t)$. Assume the following conditions hold for time steps t and $t+1$:*

1. ***Predictive Alignment:*** *$D_{\mathrm{KL}}(\mathbf{p}_t \| \hat{\mathbf{p}}_t) \leq \varepsilon_t$ and $D_{\mathrm{KL}}(\mathbf{p}_{t+1} \| \hat{\mathbf{p}}_{t+1}) \leq \varepsilon_{t+1}$.*
2. ***Target Smoothness:*** *$\|\mathbf{p}_{t+1} - \mathbf{p}_t\|_1 \leq \delta_t$.*

Then, the representation drift $\|\hat{\mathbf{z}}_{t+1} - \hat{\mathbf{z}}_t\|_2$ is bounded by:

$$\|\hat{\mathbf{z}}_{t+1} - \hat{\mathbf{z}}_t\|_2 \leq M \left(\sqrt{2\varepsilon_{t+1}} + \delta_t + \sqrt{2\varepsilon_t} \right). \tag{6}$$

Proof. First, we recall **Pinsker's Inequality** (Cover & Thomas, 2006), which provides a bound on the ℓ_1 distance between two distributions in terms of their KL divergence:

$$\|\mathbf{p} - \mathbf{q}\|_1 \leq \sqrt{2D_{\mathrm{KL}}(\mathbf{p}\|\mathbf{q})}.$$

We now decompose the drift term $\|\hat{\mathbf{z}}_{t+1} - \hat{\mathbf{z}}_t\|_2$ using the triangle inequality by introducing the intermediate target embeddings $\mathbf{z}_{t+1}$ and $\mathbf{z}_t$:

$$\begin{aligned}\|\hat{\mathbf{z}}_{t+1} - \hat{\mathbf{z}}_t\|_2 &= \|\hat{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1} + \mathbf{z}_{t+1} - \mathbf{z}_t + \mathbf{z}_t - \hat{\mathbf{z}}_t\|_2 \\ &\leq \underbrace{\|\hat{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1}\|_2}_{\text{(A)}} + \underbrace{\|\mathbf{z}_{t+1} - \mathbf{z}_t\|_2}_{\text{(B)}} + \underbrace{\|\mathbf{z}_t - \hat{\mathbf{z}}_t\|_2}_{\text{(C)}}.\end{aligned}$$

Bounding (A) and (C): Using Lemma B.2 and Pinsker's Inequality:

$$\|\hat{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1}\|_2 \leq M\|\hat{\mathbf{p}}_{t+1} - \mathbf{p}_{t+1}\|_1 \leq M\sqrt{2\varepsilon_{t+1}}.$$

By symmetry, $\|\mathbf{z}_t - \hat{\mathbf{z}}_t\|_2 \leq M\sqrt{2\varepsilon_t}$.

Bounding (B): Using Lemma B.2 and the smoothness assumption:

$$\|\mathbf{z}_{t+1} - \mathbf{z}_t\|_2 \leq M\|\mathbf{p}_{t+1} - \mathbf{p}_t\|_1 \leq M\delta_t.$$

Summing these terms yields the final bound. □

Interpretation. This theorem implies that the sensitivity of our anomaly score is strictly controlled by the codebook radius M. Unlike unconstrained continuous models where gradients can cause unbounded output shifts (effectively $M \to \infty$), SC-JEPA guarantees that any significant representation drift arises from either a true shift in the target signal (δ_t) or a high prediction error (ε). This effectively filters out false positives caused by trivial numerical perturbations.

B.3.2. Non-Collapse Lower Bound

Goal. We provide a simple *non-collapse certificate* for the soft codebook. Namely, if a mini-batch exhibits (i) non-trivial multi-code usage, (ii) sufficiently sharp assignments, and (iii) non-negligible separation between two frequently used prototypes, then the soft-quantized embeddings $\mathbf{z}_i = \mathbf{C}^\top \mathbf{p}_i$ have strictly positive batch variance, i.e., $\mathrm{Tr}(\mathrm{Cov}(\{\mathbf{z}_i\})) > 0$.

Setup. Consider a batch of assignment probabilities $\{\mathbf{p}_i\}_{i=1}^B \subset \Delta^{K-1}$ and their average $\bar{\mathbf{p}} := \frac{1}{B}\sum_{i=1}^B \mathbf{p}_i$. Let $\mathbf{z}_i := \mathbf{C}^\top \mathbf{p}_i \in \mathbb{R}^D$, where $\mathbf{C} \in \mathbb{R}^{K\times D}$ stacks prototypes as rows.

To preclude the constant-code corner case, we assume three mild properties. Crucially, these are not arbitrary assumptions but are explicitly enforced by the optimization objectives defined in Section C.2:

- **Batch Entropy:** $\mathcal{L}_{\text{ent}}^{\text{batch}}$ maximizes $H(\bar{\mathbf{p}})$, enforcing Assumption B.4.
- **Sharpness:** $\mathcal{L}_{\text{ent}}^{\text{sample}}$ minimizes individual entropy, driving $\mathbf{p}_i$ toward one-hot vectors (Assumption B.5).
- **Separation:** $\mathcal{L}_{\text{com}}$ and EMA updates prevent prototype collapse (Assumption B.7).

Assumption B.4 (Multi-code usage / non-trivial batch marginal)**.** The batch marginal $\bar{\mathbf{p}}$ is not concentrated on a single index. Concretely,

$$H(\bar{\mathbf{p}}) \geq \eta \quad \text{for some } \eta > 0. \tag{7}$$

Assumption B.5 (Assignment sharpness)**.** For each sample, let $k(i) = \arg\max_k p_{i,k}$. There exists $\varepsilon \in [0, 1)$ such that

$$\|\mathbf{p}_i - \mathbf{e}_{k(i)}\|_1 \leq \varepsilon \quad \text{for all } i. \tag{8}$$

Remark B.6. The deviation ε is strictly controlled by the softmax temperature τ. As $\tau \to 0$, $\mathbf{p}_i$ approaches a hard assignment, ensuring $\varepsilon \to 0$.

Assumption B.7 (Separation for the two dominant codes)**.** Define

$$\beta := \frac{1-\rho^*}{K-1}, \tag{9}$$

where $\rho^* \in (1/K, 1)$ is the unique root of the equation $\mathcal{H}_b(\rho) + (1-\rho)\log(K-1) = \eta$. Let $m = \arg\max_k \bar{p}_k$ and let $r \neq m$ be any index satisfying $\bar{p}_r \geq \beta$ (guaranteed by Lemma B.8). Assume their prototypes satisfy

$$\|\mathbf{c}_m - \mathbf{c}_r\|_2 \geq \Delta_c \quad \text{for some } \Delta_c > 0. \tag{10}$$

To drive a variance lower bound, we first (Lemma B.8) extract two non-negligible codes from entropy, then (Lemma B.9) convert their marginal mass into two dominant-code groups under sharp assignments.

Lemma B.8 (Entropy lower bound implies at least two active codes)**.** *Let $m = \arg\max_k \bar{p}_k$ and $\bar{p}_{\max} = \bar{p}_m$. If $H(\bar{\mathbf{p}}) \geq \eta$, then $\bar{p}_{\max} \leq \rho^*$, where ρ^* is defined in Assumption B.7. Moreover, there exists an index $r \neq m$ such that*

$$\bar{p}_r \;\geq\; \beta \;:=\; \frac{1-\rho^*}{K-1}. \tag{11}$$

Proof. Consider the function $\phi(x) := \mathcal{H}_b(x) + (1-x)\log(K-1)$, which represents the maximum entropy achievable by a distribution with maximum probability mass x. It is well-known (from Fano's inequality analysis) that $\phi(x)$ is strictly decreasing for $x \in [1/K, 1]$. Since $H(\bar{\mathbf{p}}) \geq \eta$ and $H(\bar{\mathbf{p}}) \leq \phi(\bar{p}_{\max})$, we have $\phi(\bar{p}_{\max}) \geq \eta = \phi(\rho^*)$. By the monotonicity of ϕ, this implies $\bar{p}_{\max} \leq \rho^*$. Consequently, the remaining mass is at least $1-\rho^*$. By the pigeonhole principle, at least one non-dominant index r must satisfy $\bar{p}_r \geq (1-\rho^*)/(K-1) = \beta$. □

Lemma B.9 (Marginal mass transfers to dominant-code frequency)**.** *Define $\hat{\pi}_k := \frac{1}{B}\sum_{i=1}^B \mathbb{I}[k(i) = k]$. Under Assumption B.5, we have $\|\bar{\mathbf{p}} - \hat{\boldsymbol{\pi}}\|_1 \leq \varepsilon$, hence for all k,*

$$\hat{\pi}_k \geq \bar{p}_k - \varepsilon. \tag{12}$$

Proof.

$$\|\bar{\mathbf{p}} - \hat{\boldsymbol{\pi}}\|_1 = \left\| \frac{1}{B} \sum_{i=1}^{B} (\mathbf{p}_i - \mathbf{e}_{k(i)}) \right\|_1 \le \frac{1}{B} \sum_{i=1}^{B} \|\mathbf{p}_i - \mathbf{e}_{k(i)}\|_1 \le \varepsilon.$$

The coordinate-wise bound follows from $|\bar{p}_k - \hat{\pi}_k| \le \|\bar{\mathbf{p}} - \hat{\boldsymbol{\pi}}\|_1$. □

We will use the following prototype-proximity bound implied by sharp assignments: for any i,

$$\|\mathbf{z}_i - \mathbf{c}_{k(i)}\|_2 \le \left(\max_k \|\mathbf{c}_k\|_2 \right) \|\mathbf{p}_i - \mathbf{e}_{k(i)}\|_1 \le M\varepsilon, \tag{13}$$

where $M := \max_k \|\mathbf{c}_k\|_2$.

Theorem B.10 (Explicit variance lower bound / non-collapse)**.** *Under Assumptions B.4–B.7, let β be defined as in Assumption B.7 (i.e., $\beta = \frac{1-\rho^*}{K-1}$), and define*

$$\alpha := \max\{\beta - \varepsilon,\, 0\}.$$

Let (m, r) be the indices specified in Assumption B.7. If $\alpha > 0$ and $2M\varepsilon < \Delta_c$, then the batch covariance of $\{\mathbf{z}_i\}_{i=1}^{B}$ obeys

$$\mathrm{Tr}\big(\mathrm{Cov}(\{\mathbf{z}_i\})\big) \;\ge\; \alpha^2 \,(\Delta_c - 2M\varepsilon)^2 \;>\; 0. \tag{14}$$

Proof. Step 1 (two codes have non-trivial dominant frequency). Let $A = \{i : k(i) = m\}$ and $R = \{i : k(i) = r\}$. By Lemma B.8, $\bar{p}_r \ge \beta$. Since $m = \arg\max_k \bar{p}_k$, we also have $\bar{p}_m \ge \bar{p}_r \ge \beta$. Applying Lemma B.9 yields:

$$\Pr(i \in A) = \hat{\pi}_m \ge \bar{p}_m - \varepsilon \ge \beta - \varepsilon = \alpha,$$
$$\Pr(i \in R) = \hat{\pi}_r \ge \bar{p}_r - \varepsilon \ge \beta - \varepsilon = \alpha.$$

Step 2 (dominant-code membership implies proximity to the corresponding prototype). By (13) and Assumption B.5, we have tight concentration around prototypes:

$$i \in A \implies \|\mathbf{z}_i - \mathbf{c}_m\|_2 \le M\varepsilon, \qquad j \in R \implies \|\mathbf{z}_j - \mathbf{c}_r\|_2 \le M\varepsilon.$$

Step 3 (two separated prototype-centered clusters imply positive covariance trace). For any $i \in A$ and $j \in R$, the triangle inequality gives:

$$\begin{aligned} \|\mathbf{z}_i - \mathbf{z}_j\|_2 &\ge \|\mathbf{c}_m - \mathbf{c}_r\|_2 - \|\mathbf{z}_i - \mathbf{c}_m\|_2 - \|\mathbf{z}_j - \mathbf{c}_r\|_2 \\ &\ge \Delta_c - 2M\varepsilon. \end{aligned}$$

Now view $\mathbf{Z}$ as a random variable obtained by uniformly sampling an index from the batch, and let $\mathbf{Z}'$ be an independent copy. The pairwise-variance identity yields $\mathrm{Tr}(\mathrm{Cov}(\mathbf{Z})) = \frac{1}{2}\,\mathbb{E}\big[\|\mathbf{Z} - \mathbf{Z}'\|_2^2\big]$. Restricting the expectation to the event $\{i \in A,\ j \in R\}$ yields the bound:

$$\begin{aligned} \mathrm{Tr}(\mathrm{Cov}(\mathbf{Z})) &\ge \frac{1}{2} \cdot 2\, \Pr(i \in A) \Pr(j \in R)\, (\Delta_c - 2M\varepsilon)^2 \\ &\ge \alpha^2 (\Delta_c - 2M\varepsilon)^2. \end{aligned}$$

If $\alpha > 0$ and $2M\varepsilon < \Delta_c$, the right-hand side is strictly positive, ruling out collapse. □

Remark. Theorem B.10 provides a certificate against collapse. The bound becomes vacuous only if (i) $\alpha = 0$ (low batch entropy), or (ii) $\Delta_c \le 2M\varepsilon$ (low sharpness or prototype separation). However, as shown empirically in Appendix **??**, the model maintains high code diversity and sharpness throughout training, ensuring these conditions are met in practice.

C. Reproducibility Details

C.1. Dataset Statistics and Preprocessing

Table 5 summarizes the dataset statistics. For preprocessing, we remove constant channels identified from the training split and apply the same channel mask to the test split, ensuring consistent effective dimensions across splits. All remaining variables are instance-normalized independently within each window.

We construct training pairs $(\mathbf{X}_t, \mathbf{X}_{t+1})$ from two consecutive non-overlapping windows sampled from the time series. Each window has a fixed length T and is partitioned into $P{=}T/L$ patches with patch length $L{=}20$.

Table 5. **Dataset Statistics. Raw Dim.** is the original dimension; **Eff. Dim.** is the dimension after removing constant features.

Dataset	Raw	Eff.	Constant Indices (Zero Variance)	Train	Test	Anom (%)
MSL	55	33	[1, 4, 8, 10, 18, 21, 22, 24, 25, 26, 30, 32, 34, 36, 37, 38, 40, 42, 44, 50, 51, 52]	58,317	73,729	10.5
SMAP	25	24	[16]	135,183	427,617	12.8
SWaT	51	40	[4, 10, 11, 13, 15, 29, 31, 32, 43, 48, 50]	495,000	449,919	12.1
PSM	25	25	–	132,481	87,841	27.8

C.2. Optimization Objective

We first introduce the notation used to define the three coupled objectives. Let $\mathbf{p}_{t,i} \in \mathbb{R}^K$ denote the online soft code distribution for patch i at window t (Eq. 1), satisfying $\mathbf{p}_{t,i} \geq \mathbf{0}$ and $\mathbf{1}^\top \mathbf{p}_{t,i} = 1$. Let $\mathbf{z}_{t,i} = \sum_{k=1}^{K} p_{t,i,k}\mathbf{c}_k$ be the corresponding soft-quantized latent. The fine predictor outputs $\hat{\mathbf{p}}^{\text{fine}}_{t+1,i}$ and latent $\hat{\mathbf{z}}_{t+1,i}$ for each patch, while the coarse predictor outputs a single distribution $\hat{\mathbf{p}}^{\text{coarse}}_{t+1}$. All targets $\mathbf{p}^{\text{fine}}_{t+1,i}, \mathbf{p}^{\text{coarse}}_{t+1}$ and $\mathbf{z}_{t+1,i}$ are produced by the EMA (target) branch.

Overall objective. We optimize the online encoder, codebook, predictors, and decoder using three terms:

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{code}} + \lambda_r \, \mathcal{L}_{\text{rec}}. \tag{15}$$

With the overall objective in place, we detail each term next.

(A) Predictive Objective ($\mathcal{L}_{\text{pred}}$). We supervise online predictions using EMA targets at both fine (patch) and coarse (window) resolutions, so that the representation is jointly shaped by local precursors and global trends.

$$\mathcal{L}_{\text{pred}} = \underbrace{\lambda_f \left(\mathcal{L}^{\text{fine}}_{\text{KL}} + \gamma \mathcal{L}^{\text{fine}}_{\text{MSE}}\right)}_{\text{Local Structure}} + \underbrace{\lambda_c \mathcal{L}^{\text{coarse}}_{\text{KL}}}_{\text{Global Dynamics}} . \tag{16}$$

The fine-grained components align code distributions and stabilize latent features, while the coarse objective captures high-level semantics:

$$\mathcal{L}^{\text{fine}}_{\text{KL}} = \sum_i D_{\text{KL}}(\mathbf{p}^{\text{fine}}_{t+1,i} \parallel \hat{\mathbf{p}}^{\text{fine}}_{t+1,i}), \tag{17}$$

$$\mathcal{L}^{\text{fine}}_{\text{MSE}} = \sum_i \|\mathbf{z}_{t+1,i} - \hat{\mathbf{z}}_{t+1,i}\|_2^2, \tag{18}$$

$$\mathcal{L}^{\text{coarse}}_{\text{KL}} = D_{\text{KL}}(\mathbf{p}^{\text{coarse}}_{t+1} \parallel \hat{\mathbf{p}}^{\text{coarse}}_{t+1}). \tag{19}$$

Here, D_{KL} denotes the Kullback–Leibler divergence. λ_f and λ_c balance patch-level supervision with window-level consistency. Since the targets $\mathbf{p}$ are probability vectors produced by the softmax codebook (i.e., $\mathbf{p} \in \Delta^{K-1}$), KL provides a principled objective for distributional alignment in the code space, rather than treating codes as unconstrained Euclidean regressands. In addition, the momentum-updated EMA branch yields slowly varying targets, which stabilizes self-distillation and makes KL matching better conditioned during training.

(B) Codebook Objective ($\mathcal{L}_{\text{code}}$). A well-behaved bottleneck requires more than defining soft assignments: during training, the code space must stay synchronized with the encoder features, and the assignment distributions must avoid becoming either overly diffuse or collapsed to a few indices. Accordingly, we regularize the soft codebook with (i) prototype–feature alignment and (ii) dual-entropy calibration:

$$\mathcal{L}_{\text{code}} = \underbrace{\lambda_{\text{emb}}\mathcal{L}_{\text{emb}} + \lambda_{\text{com}}\mathcal{L}_{\text{com}}}_{\text{Codebook alignment}} + \underbrace{\lambda_{\text{ent}}^{\text{sample}}\mathcal{L}_{\text{ent}}^{\text{sample}} - \lambda_{\text{ent}}^{\text{batch}}\mathcal{L}_{\text{ent}}^{\text{batch}}}_{\text{Entropy}}. \tag{20}$$

The alignment terms couple continuous encoder features $\mathbf{h}_{t,i}$ with their soft-quantized embeddings $\mathbf{z}_{t,i}$ via a bidirectional objective:

$$\mathcal{L}_{\text{emb}} = \textstyle\sum_i \| \operatorname{sg}(\mathbf{z}_{t,i}) - \mathbf{h}_{t,i}\|_2^2, \tag{21}$$

$$\mathcal{L}_{\text{com}} = \textstyle\sum_i \|\mathbf{z}_{t,i} - \operatorname{sg}(\mathbf{h}_{t,i})\|_2^2, \tag{22}$$

where $\operatorname{sg}(\cdot)$ denotes the stop-gradient operator.

Meanwhile, we calibrate the assignment distributions through dual-entropy control:

$$\mathcal{L}_{\text{ent}}^{\text{sample}} = \mathbb{E}[H(\mathbf{p})], \quad \mathcal{L}_{\text{ent}}^{\text{batch}} = H(\mathbb{E}[\mathbf{p}]). \tag{23}$$

In particular, minimizing $\mathcal{L}_{\text{ent}}^{\text{sample}}$ sharpens per-sample assignments and encourages decisive code selection, whereas maximizing $\mathcal{L}_{\text{ent}}^{\text{batch}}$ promotes diverse code usage across the dataset, mitigating index collapse. Together, these terms keep the prototype set aligned with the evolving feature space while maintaining stable, informative code distributions for downstream prediction.

(C) Reconstruction Objective ($\mathcal{L}_{\text{rec}}$). To anchor latent representations and prevent collapse, we minimize the reconstruction error between the input patches and their reconstructions (after denormalization):

$$\mathcal{L}_{\text{rec}} = \sum_{i=1}^{P} \left\|\hat{\mathbf{X}}_{t,i} - \mathbf{X}_{t,i}\right\|_2^2. \tag{24}$$

Here $\hat{\mathbf{X}}_{t,i}$ is denormalized using the cached RevIN statistics of window t before computing $\mathcal{L}_{\text{rec}}$.

C.3. Implementation Details

Training & Evaluation Protocol. Each multivariate series is converted into non-overlapping context–target pairs with $T_c = 100$ and $T_t = 100$, where each context window is split into $P = 5$ patches of length $L = 20$. We pre-train SC-JEPA on the official training set in a self-supervised manner, and evaluate downstream performance using a chronological 6:2:2 split of the official test set for classifier training, threshold selection, and final testing. We report window-level Precision, Recall, F1, and AUC. Further details are deferred to Appendix C.3 and Appendix C.5.

All methods are implemented in PyTorch and trained on a single NVIDIA GeForce RTX 4090 GPU. We use Adam for all optimizations, and unless otherwise specified, we keep the hyperparameters reported below fixed across datasets to ensure a controlled comparison. We set the latent dimension to D=256 and the codebook size to K=128, and maintain EMA target networks for both the encoder and the codebook to stabilize training.

After each optimization step on the online parameters θ, the corresponding target parameters ξ are updated by $\xi \leftarrow \rho\xi + (1-\rho)\theta$ with decay ρ=0.996; gradients are not back-propagated through the EMA targets (stop-gradient).

Model Architectures.

- **Encoder:** Patch size L=20. The projection head is a 2-layer MLP (hidden 64→32) mapping to latent dimension D=256. The backbone is a 6-layer Transformer (8 heads, dropout 0.1).
- **Codebook:** Codebook size K=128, dimension D=256. Soft quantization temperature τ=0.1.
- **Predictors:** Both fine and coarse predictors are 2-layer Transformers (4 heads, hidden dimension 128).

Optimization & Hyperparameters.

- **Training:** Adam optimizer (lr 5×10^{-4}, weight decay 10^{-5}), batch size 128. Target networks are updated via EMA (decay 0.996). Gradient clipping norm is 0.5. We train for a maximum of 100 epochs, initiating model selection after 50 epochs with an early stopping patience of 10.
- **Loss Weights:** Predicted objective weights λ_f=1.0, λ_c=0.5, and γ=0.1. Codebook alignment weights λ_{emb}=1.0 and λ_{com}=0.25. Entropy regularization weights $\lambda_{\text{ent}}^{\text{sample}}$=0.005 and $\lambda_{\text{ent}}^{\text{batch}}$=0.01. Reconstruction weight λ_r is annealed linearly from 0.5 to 0.1 during training.

C.4. SC-JEPA Pre-training Algorithm

We provide the detailed formulation of the total training objective for reproducibility. Aligning with the notation in Section C.2, the total loss $\mathcal{L}$ is composed of three parts:

$$\begin{aligned}\mathcal{L} = &\underbrace{\lambda_f \left(\mathcal{L}_{\text{KL}}^{\text{fine}} + \gamma \mathcal{L}_{\text{MSE}}^{\text{fine}}\right) + \lambda_c \mathcal{L}_{\text{KL}}^{\text{coarse}}}_{\text{Predictive Objective}} \\ &+ \underbrace{\lambda_{\text{emb}}\mathcal{L}_{\text{emb}} + \lambda_{\text{com}}\mathcal{L}_{\text{com}} + \lambda_{\text{ent}}^{\text{sample}}\mathcal{L}_{\text{ent}}^{\text{sample}} - \lambda_{\text{ent}}^{\text{batch}}\mathcal{L}_{\text{ent}}^{\text{batch}}}_{\text{Codebook Objective}} \\ &+ \underbrace{\lambda_r \mathcal{L}_{\text{rec}}}_{\text{Reconstruction}}\end{aligned}$$

Algorithm 1 summarizes the full pre-training pseudocode.

Algorithm 1 SC-JEPA Pre-training Procedure

Require: Dataset $\mathcal{D}$, Online parameters θ, EMA parameters ξ, EMA rate τ

Sample batch $(\mathbf{X}_t, \mathbf{X}_{t+1}) \sim \mathcal{D}$ ▷ fine-scale future
$\mathbf{X}_{t+1}^{\text{coarse}} \leftarrow \text{Downsample}(\mathbf{X}_{t+1})$ ▷ coarse-scale future
while not converged **do**
  $\mathbf{h}_t \leftarrow \text{Encoder}_\theta(\mathbf{X}_t)$ ▷ online encoding
  $\mathbf{p}_t, \mathbf{z}_t \leftarrow \text{Codebook}_\theta(\mathbf{h}_t)$ ▷ online soft codes
  $\mathcal{L}_{\text{code}} \leftarrow$ codebook loss ▷ on $(\mathbf{h}_t, \mathbf{z}_t, \mathbf{p}_t)$
  $\hat{\mathbf{X}}_t \leftarrow \text{Decoder}_\theta(\mathbf{z}_t)$ ▷ reconstruction branch
  $\mathcal{L}_{\text{rec}} \leftarrow$ reconstruction loss ▷ compare $\hat{\mathbf{X}}_t$ vs. $\mathbf{X}_t$
  $\hat{\mathbf{p}}_{t+1}^{\text{fine}}, \hat{\mathbf{z}}_{t+1} \leftarrow \text{Predictor}_\theta^{\text{fine}}(\mathbf{p}_t)$ ▷ fine prediction
  $\hat{\mathbf{p}}_{t+1}^{\text{coarse}} \leftarrow \text{Predictor}_\theta^{\text{coarse}}(\mathbf{p}_t)$ ▷ coarse prediction
  $\mathbf{p}_{t+1}^{\text{fine}}, \mathbf{z}_{t+1} \leftarrow \text{EMACodebook}_\xi(\mathbf{X}_{t+1})$ ▷ stop-grad targets
  $\mathbf{p}_{t+1}^{\text{coarse}} \leftarrow \text{EMACodebook}_\xi(\mathbf{X}_{t+1}^{\text{coarse}})$ ▷ stop-grad targets
  $\mathcal{L}_{\text{pred}}^{\text{fine}} \leftarrow$ prediction loss ▷ $(\hat{\mathbf{p}}_{t+1}^{\text{fine}}, \hat{\mathbf{z}}_{t+1})$ vs. $(\mathbf{p}_{t+1}^{\text{fine}}, \mathbf{z}_{t+1})$
  $\mathcal{L}_{\text{pred}}^{\text{coarse}} \leftarrow$ prediction loss ▷ $\hat{\mathbf{p}}_{t+1}^{\text{coarse}}$ vs. $\mathbf{p}_{t+1}^{\text{coarse}}$
  $\mathcal{L}_{\text{pred}} \leftarrow \mathcal{L}_{\text{pred}}^{\text{fine}} + \mathcal{L}_{\text{pred}}^{\text{coarse}}$
  $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{quant}} + \lambda_r \mathcal{L}_{\text{rec}}$
  Update θ; update ξ with EMA ▷ optimizer step + EMA update
end while

C.5. Downstream Anomaly Prediction Protocol

Problem Formulation. We formulate the downstream task as Anomaly Prediction, distinguishing it from standard point-wise anomaly detection. The objective is to utilize a historical context window $\mathbf{X}_t$ to predict the anomaly status of the subsequent non-overlapping future window $\mathbf{X}_{t+1}$. This setup evaluates the model's ability to identify early warning signs

(precursors) rather than merely reacting to ongoing failures. Each dataset provides anomaly annotations as a binary indicator array aligned with the time axis, denoted by $\mathbf{a} \in \{0,1\}^{T_{\text{full}}}$. Let $\mathcal{I}_{t+1}$ be the set of time indices covered by the future window $\mathbf{X}_{t+1}$. We define the window-level label as

$$y_{t+1} \;=\; \mathbb{I}\left(\sum_{\tau \in \mathcal{I}_{t+1}} \mathbf{a}_\tau > 0\right) \in \{0,1\}. \tag{25}$$

Let $\hat{s}_{t+1} \in [0,1]$ denote the predicted anomaly probability and let

$$\hat{y}_{t+1} \;=\; \mathbb{I}(\hat{s}_{t+1} > \delta^*) \tag{26}$$

be the binary prediction using threshold δ^* selected on the validation set to maximize the window-level F1 score.

Notation. We denote by $\mathbf{X}_t$ the context window and by $\mathbf{X}_{t+1}$ the subsequent non-overlapping future window. The supervision signal is the *future-window* label $y_{t+1} \in \{0,1\}$, which indicates whether $\mathbf{X}_{t+1}$ is anomalous under the window-level definition above. Thus, each downstream training example is a pair $(\mathbf{X}_t, y_{t+1})$, and the goal is anomaly prediction.

We report window-level Precision, Recall, and F1-score:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \quad \text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{27}$$

We additionally report the Area Under the Receiver Operating Characteristic curve (AUC) computed from the raw probabilities $\hat{s}_{t+1}$.

Adaptation Strategy. We utilize the pre-trained SC-JEPA as a frozen feature extractor. Specifically, we discard the projection heads and freeze the parameters of the encoder and codebook. For a given input $\mathbf{X}_t$, the encoder yields soft code probabilities $\mathbf{p}_t \in \mathbb{R}^{V \times P \times K}$, where V denotes variables, P patches, and K the codebook size. To capture the most salient anomalies across the multivariate dimensions while preserving temporal structure, we apply *Variable-wise Max-Pooling*:

$$\mathbf{h}_t = \max_{v \in \{1 \ldots V\}} \mathbf{p}_t^{(v)} \in \mathbb{R}^{P \times K}. \tag{28}$$

The aggregated representation $\mathbf{h}_t$ is flattened and passed to an MLP classifier C_ψ, trained to estimate $\hat{s}_{t+1} = P(y_{t+1} = 1 \mid \mathbf{h}_t)$.

Algorithm 2 Downstream Anomaly Prediction Protocol

Require: Frozen Encoder θ, frozen Codebook, train split $\mathcal{S}_{\text{tr}}$, val split $\mathcal{S}_{\text{val}}$, test split $\mathcal{S}_{\text{te}}$
Initialize MLP classifier C_ψ
Train:
for each batch $(\mathbf{X}_t, y_{t+1}) \in \mathcal{S}_{\text{tr}}$ **do**
  $\mathbf{p}_t \leftarrow \text{Codebook}(\text{Encoder}_\theta(\mathbf{X}_t))$ ▷ $\mathbf{p}_t \in \mathbb{R}^{V \times P \times K}$
  $\mathbf{h}_t \leftarrow \max_{v \in \{1,\ldots,V\}} \mathbf{p}_t^{(v)}$ ▷ variable-wise max-pooling, $\mathbf{h}_t \in \mathbb{R}^{P \times K}$
  $\hat{s}_{t+1} \leftarrow C_\psi(\mathbf{h}_t)$ ▷ anomaly probability
  Update ψ using binary cross-entropy loss $\mathcal{L}_{\text{cls}}(\hat{s}_{t+1}, y_{t+1})$
end for
Validate: choose δ^* on $\mathcal{S}_{\text{val}}$ to maximize window-level F1 using $\hat{y}_{t+1} = \mathbb{I}(\hat{s}_{t+1} > \delta)$
Test: report Precision/Recall/F1 on $\mathcal{S}_{\text{te}}$ using δ^*, and AUC using raw $\hat{s}_{t+1}$

D. Experimental Results

D.1. Main Benchmark

Table 6. Full results on five benchmark datasets. All metrics are reported in percentage (%) as mean ± standard deviation over 5 runs with different random seeds. Best baselines are underlined.

Models	MSL				SMAP				SWaT				PSM				SMD			
	F1	AUC	Prec	Rec	F1	AUC	Prec	Rec	F1	AUC	Prec	Rec	F1	AUC	Prec	Rec	F1	AUC	Prec	Rec
K-Means	20.63 ± 2.30	52.17 ± 5.65	19.27 ± 7.16	31.34 ± 15.40	8.74 ± 3.39	39.82 ± 2.57	10.52 ± 8.17	10.16 ± 4.92	14.45 ± 0.38	62.20 ± 0.16	7.99 ± 0.32	77.05 ± 8.16	42.75 ± 1.73	51.37 ± 3.80	30.40 ± 2.34	78.89 ± 21.18	14.79 ± 5.41	57.49 ± 0.22	35.83 ± 34.03	20.64 ± 11.95
DeepSVDD	23.46 ± 0.98	53.67 ± 4.80	14.78 ± 2.14	73.87 ± 26.12	22.05 ± 2.97	44.07 ± 2.61	12.90 ± 1.19	81.99 ± 24.23	15.28 ± 1.75	58.22 ± 2.34	12.65 ± 1.90	20.00 ± 3.34	44.45 ± 1.56	49.95 ± 3.10	30.31 ± 3.33	89.38 ± 16.00	9.41 ± 2.42	47.31 ± 4.44	6.25 ± 1.61	19.15 ± 4.85
LSTM-VAE	22.42 ± 7.02	55.22 ± 10.27	20.84 ± 10.10	32.43 ± 15.96	22.77 ± 1.73	52.97 ± 2.60	14.35 ± 1.21	55.39 ± 4.47	54.60 ± 1.17	79.91 ± 1.00	64.92 ± 14.30	50.07 ± 8.30	56.73 ± 1.14	71.95 ± 0.50	49.32 ± 1.59	66.86 ± 2.10	10.04 ± 1.10	56.18 ± 0.86	5.78 ± 0.58	38.29 ± 6.92
iTransformer	27.25 ± 2.03	64.61 ± 1.19	16.71 ± 1.25	75.40 ± 11.80	33.00 ± 2.05	60.91 ± 1.58	22.67 ± 2.83	64.32 ± 10.77	70.49 ± 1.00	82.10 ± 0.36	98.17 ± 1.82	55.00 ± 1.18	54.12 ± 0.92	63.09 ± 0.75	37.37 ± 1.17	98.29 ± 2.10	14.20 ± 0.75	57.94 ± 0.85	8.11 ± 0.76	62.34 ± 13.63
TimesNet	28.44 ± 5.79	59.70 ± 5.15	30.31 ± 7.40	36.59 ± 27.00	27.78 ± 1.48	57.01 ± 1.81	17.98 ± 0.60	62.81 ± 10.10	67.20 ± 3.94	83.24 ± 1.64	90.87 ± 15.87	54.42 ± 6.09	52.64 ± 0.02	56.12 ± 2.66	35.76 ± 0.09	99.71 ± 0.57	12.51 ± 0.49	56.33 ± 0.87	7.64 ± 0.42	35.96 ± 6.64
Qwen2.5-3B	21.24 ± 0.52	55.66 ± 0.84	17.82 ± 1.28	26.67 ± 2.22	0.34 ± 0.69	63.61 ± 1.05	1.67 ± 3.33	0.19 ± 0.38	17.29 ± 1.85	56.75 ± 7.35	41.82 ± 21.26	14.43 ± 6.83	47.44 ± 9.10	65.65 ± 5.93	50.44 ± 6.09	46.57 ± 14.36	13.00 ± 1.78	60.66 ± 4.59	8.29 ± 1.41	31.06 ± 3.65
PAD	21.67 ± 0.03	55.97 ± 6.96	12.19 ± 0.06	97.78 ± 4.44	26.92 ± 1.78	59.79 ± 0.30	18.15 ± 1.21	55.00 ± 11.80	69.75 ± 4.83	83.71 ± 0.61	81.54 ± 15.17	62.79 ± 4.32	57.82 ± 5.96	73.84 ± 1.85	57.54 ± 3.18	58.57 ± 9.17	12.52 ± 0.34	58.53 ± 2.35	6.71 ± 0.20	93.40 ± 2.46
A2P	23.10 ± 0.85	59.43 ± 1.28	13.92 ± 1.18	74.12 ± 17.29	21.68 ± 4.10	60.37 ± 1.42	15.88 ± 1.26	38.85 ± 18.15	70.18 ± 0.26	81.92 ± 0.10	99.10 ± 0.74	54.33 ± 0.33	50.71 ± 0.96	57.01 ± 0.38	37.16 ± 0.97	80.00 ± 3.48	15.21 ± 0.73	58.87 ± 1.55	12.23 ± 1.48	22.13 ± 6.33
FCM	24.57 ± 0.73	61.09 ± 0.76	14.25 ± 0.54	90.00 ± 7.37	30.60 ± 3.08	61.85 ± 0.86	25.25 ± 2.76	41.41 ± 9.44	69.09 ± 1.08	81.66 ± 0.37	94.25 ± 7.92	54.83 ± 1.86	52.34 ± 0.89	57.25 ± 1.18	37.00 ± 1.39	89.85 ± 4.37	19.25 ± 1.46	60.44 ± 0.65	15.59 ± 2.43	28.94 ± 10.43
MICN	20.00 ± 0.96	60.18 ± 2.26	11.56 ± 0.57	75.29 ± 10.12	20.81 ± 2.38	51.16 ± 1.72	13.75 ± 3.01	55.58 ± 22.01	48.31 ± 3.19	79.38 ± 0.30	77.99 ± 7.96	35.33 ± 3.71	48.61 ± 1.38	54.97 ± 1.12	34.64 ± 1.33	82.09 ± 6.61	12.05 ± 0.70	52.21 ± 0.37	7.72 ± 0.53	28.72 ± 5.83
Pathformer	21.69 ± 2.29	61.86 ± 0.66	13.14 ± 1.43	68.24 ± 18.45	23.58 ± 1.27	58.67 ± 0.38	14.10 ± 0.68	72.12 ± 7.37	14.90 ± 1.72	59.87 ± 1.24	9.20 ± 1.04	40.98 ± 9.16	51.77 ± 0.84	54.41 ± 3.09	35.04 ± 0.76	99.10 ± 1.19	14.32 ± 0.50	56.49 ± 1.71	9.41 ± 1.24	36.60 ± 15.82
TS-JEPA	25.49 ± 4.22	60.33 ± 4.65	17.10 ± 3.91	56.84 ± 18.65	26.57 ± 1.67	57.38 ± 0.72	18.23 ± 0.74	49.73 ± 8.05	71.95 ± 0.56	80.33 ± 0.69	84.32 ± 1.67	62.76 ± 0.33	53.32 ± 2.25	66.10 ± 3.11	36.95 ± 2.50	96.29 ± 2.65	7.49 ± 2.65	49.53 ± 8.44	19.11 ± 27.96	20.43 ± 14.37
PatchTST	26.98 ± 0.51	60.43 ± 1.13	75.00 ± 0.00	16.45 ± 0.38	30.06 ± 1.20	61.62 ± 0.66	21.76 ± 0.93	49.06 ± 2.10	70.64 ± 3.57	81.93 ± 1.27	94.03 ± 4.25	57.58 ± 3.94	58.17 ± 0.73	75.76 ± 0.11	64.68 ± 0.59	52.86 ± 0.90	11.79 ± 0.43	47.76 ± 0.95	6.58 ± 0.22	57.02 ± 3.40
TS2Vec	23.48 ± 2.14	64.86 ± 0.54	31.45 ± 2.46	18.82 ± 2.35	32.81 ± 0.32	61.48 ± 0.46	23.81 ± 0.30	52.81 ± 1.95	67.00 ± 0.83	83.76 ± 0.23	80.95 ± 2.10	57.17 ± 0.41	48.43 ± 1.01	72.13 ± 0.43	32.39 ± 0.58	95.96 ± 2.90	14.78 ± 1.46	56.28 ± 0.31	8.83 ± 1.28	50.43 ± 11.51
SC-JEPA	**33.58 ± 4.34**	**66.08 ± 3.25**	**35.87 ± 10.90**	**40.80 ± 15.68**	**33.64 ± 1.45**	**65.41 ± 2.06**	**24.24 ± 1.28**	**56.02 ± 8.18**	**72.89 ± 0.70**	**84.95 ± 0.82**	**98.00 ± 1.00**	**58.05 ± 1.20**	**61.61 ± 4.32**	**77.85 ± 1.28**	**55.01 ± 2.47**	**72.00 ± 12.96**	**20.82 ± 2.61**	**62.29 ± 0.90**	**18.18 ± 5.24**	31.49 ± 13.48

D.2. Generality

We evaluate generality by comparing in-domain and cross-domain pre-training while keeping the downstream evaluation exactly the same.

For a target dataset, in-domain pre-training uses its official training set. Cross-domain pre-training uses the union of the other three datasets, with the target dataset completely excluded during pre-training. Since the source datasets may have different channel dimensions in this cross-domain setting, we treat each channel as an independent univariate series during pre-training, which avoids any explicit channel matching across datasets. After pre-training, we follow the same downstream protocol and evaluate on the same target test split in both settings, so the only difference is the source of pre-training data.

Table 7. Comparison of in-domain versus cross-domain generalization.

Models	Settings	MSL		SMAP		SWaT		PSM	
		F1	AUC	F1	AUC	F1	AUC	F1	AUC
PatchTST	In-Domain	26.98±0.51	60.43±1.13	30.06±1.20	61.62±0.66	70.64±3.57	81.93±1.27	58.17±0.73	75.76±0.11
	Cross-Domain	27.08±0.44	60.80±1.17	26.21±0.24	58.77±1.63	54.34±5.96	68.76±0.87	50.07±0.19	53.72±0.93
	Abs. Diff.	+0.10	+0.37	-3.85	-2.85	-16.30	-13.17	-8.10	-22.04
	Rel. Diff.	+0.37%	+0.61%	-12.81%	-4.63%	-23.07%	-16.07%	-13.92%	-29.09%
TS2Vec	In-Domain	23.48±2.14	64.86±0.54	32.81±0.32	61.48±0.46	67.00±0.83	83.76±0.23	48.43±1.01	72.13±0.43
	Cross-Domain	22.87±2.67	62.00±0.74	26.99±0.75	57.54±0.35	65.08±0.53	82.91±0.02	48.15±7.73	56.22±3.49
	Abs. Diff.	-0.61	-2.86	-5.82	-3.94	-1.92	-0.85	-0.28	-15.91
	Rel. Diff.	-2.60%	-4.41%	-17.74%	-6.41%	-2.87%	-1.01%	-0.58%	-22.06%
SC-JEPA	In-Domain	33.58±4.34	66.08±3.25	33.64±1.45	65.41±2.06	72.89±0.70	84.95±0.82	61.61±4.32	77.85±1.28
	Cross-Domain	33.48±7.26	67.13±6.15	33.81±5.13	68.88±1.45	71.10±0.76	82.96±0.40	56.87±4.31	66.80±6.02
	Abs. Diff.	-0.10	+1.05	+0.17	+3.47	-1.79	-1.99	-4.74	-11.05
	Rel. Diff.	-0.30%	**+1.59%**	+0.51%	**+5.30%**	-2.46%	**-2.34%**	-7.69%	**-14.19%**

D.3. Ablation Studies

Table 8 reports the complete ablation results. All variants follow the same training and evaluation protocol as the full model, and we modify only the targeted component while keeping the rest of the architecture and training pipeline unchanged. When the standard deviation across random seeds is (near) zero while the mean score is low, this typically indicates model collapse, i.e., the model makes nearly identical (often single-class) predictions across runs and loses discriminative signal.

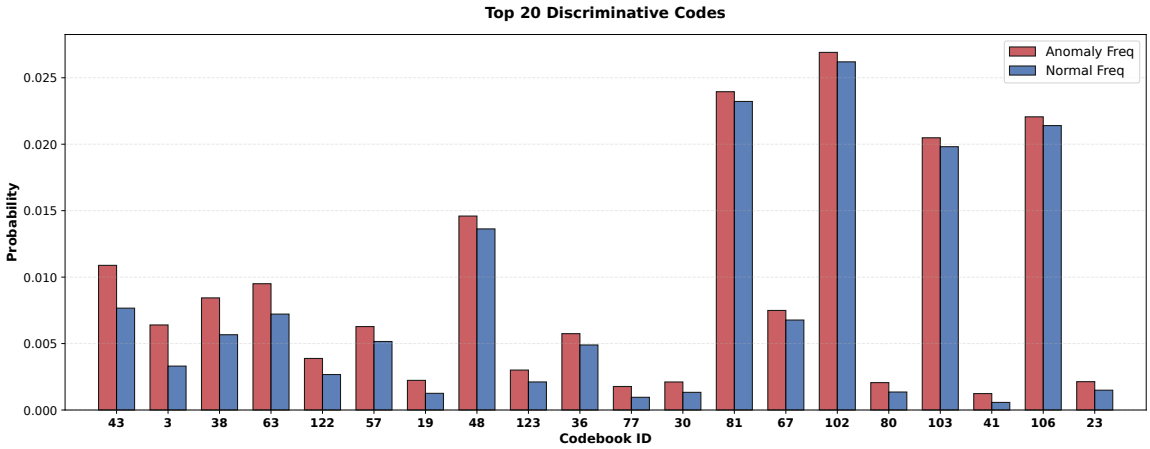


Figure 2. Histogram of latent code occurrence in anomalous and normal PSM windows, sorted by anomaly–normal frequency gap.

Table 8. Ablation results on five benchmark datasets.

Datasets	**MSL**		**SMAP**		**SWaT**		**PSM**		**SMD**	
Metrics	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
w/o KL Regularization	29.01 ± 6.64	53.90 ± 5.25	28.71 ± 1.71	60.19 ± 0.95	67.33 ± 3.97	79.95 ± 0.97	54.93 ± 2.54	70.83 ± 1.25	12.63 ± 3.75	53.61 ± 8.33
w/o Reconstruction Decoder	25.96 ± 1.82	52.68 ± 1.68	25.15 ± 0.31	52.80 ± 1.88	14.77 ± 1.42	53.30 ± 2.45	53.03 ± 0.00	69.84 ± 7.74	10.18 ± 1.65	50.44 ± 1.44
w/o Predictive Objective	28.88 ± 6.41	53.17 ± 1.03	27.05 ± 2.31	59.81 ± 1.76	59.47 ± 1.55	78.60 ± 0.59	58.89 ± 5.50	70.17 ± 4.73	10.04 ± 3.42	50.24 ± 5.07
w/o Codebook Loss	31.62 ± 1.71	58.93 ± 1.81	31.53 ± 7.88	62.06 ± 0.60	72.64 ± 1.24	82.63 ± 0.63	60.83 ± 4.09	75.85 ± 0.65	12.82 ± 2.01	57.36 ± 4.25
w/o Codebook Module	21.82 ± 0.00	43.02 ± 9.30	21.69 ± 0.00	51.00 ± 1.23	11.51 ± 0.00	50.00 ± 0.00	53.03 ± 0.00	46.61 ± 6.79	12.45 ± 0.00	50.00 ± 0.00
w/o Downsampling	28.77 ± 6.95	63.16 ± 3.83	30.39 ± 2.47	61.34 ± 0.15	71.62 ± 0.28	83.60 ± 0.40	60.79 ± 3.31	71.72 ± 4.68	15.50 ± 1.54	60.93 ± 0.10
Full Model	$\mathbf{33.58 \pm 4.34}$	$\mathbf{66.08 \pm 3.25}$	$\mathbf{33.64 \pm 1.45}$	$\mathbf{65.41 \pm 2.06}$	$\mathbf{72.89 \pm 0.70}$	$\mathbf{84.95 \pm 0.82}$	$\mathbf{61.61 \pm 4.32}$	$\mathbf{77.85 \pm 1.28}$	$\mathbf{20.82 \pm 2.61}$	$\mathbf{62.29 \pm 0.90}$

E. Visualizations and Qualitative Analysis

E.0.1. Visualization of Learned Regimes

As motivated in the Introduction and formalized in Section 2.3, the codebook $\mathcal{Q}$ is designed to capture regime-like patterns through a discrete bottleneck with finite prototypes $\{\mathbf{c}_k\}$. Here, we empirically verify whether the learned representations align with these methodological motivations.

First, we examine the activation statistics of the codebook. Figure 2 shows latent code activation statistics on PSM, ranked by the frequency gap between anomalous and normal windows. A small subset of codes exhibits clear activation differences, suggesting that the model partitions the latent space into condition-dependent regions without explicit supervision. Figure 3 further maps the most discriminative codes back to time-series patches, where different codes align with different physical behaviors. This suggests that the codebook disentangles local dynamics into identifiable regimes and helps separate precursor-like patterns from normal dynamics.

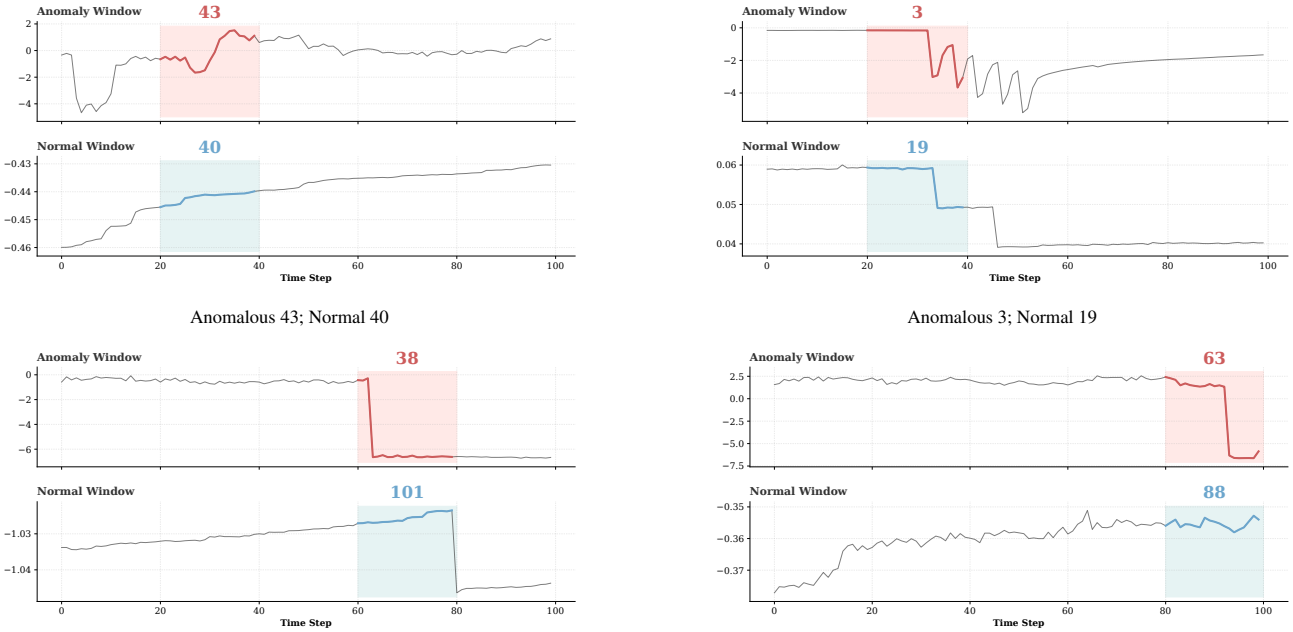
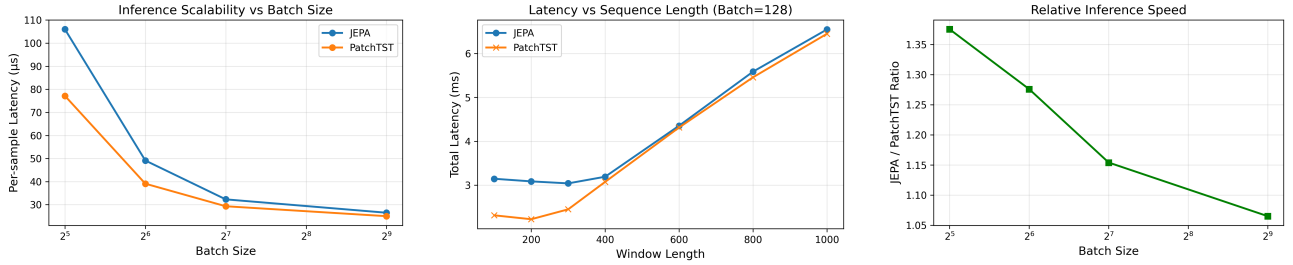


Figure 3. Window-level comparisons of dominant latent codes between anomalous and normal windows on the PSM dataset. Shaded regions indicate patch-level segments where the dominant code is selected.

E.1. Inference Efficiency



(a) Latency vs. Batch *(b)* Latency vs. Window *(c)* Latency Ratio

Figure 4. Inference-time scaling comparisons across batch size, window length, and latency ratios.

Figure 4 evaluates the inference efficiency of SC-JEPA compared with PatchTST from three complementary perspectives. First, Figure 4a reports the per-sample latency vs. batch size: SC-JEPA is slower, but the gap shrinks at larger batches. Figure 4b then reports latency vs. window length (batch fixed): both scale similarly, with SC-JEPA slightly slower. Figure 4c confirms this trend: the latency ratio decreases as batch size increases.

Overall, these results highlight a limitation of our approach: inference efficiency is not a primary contribution of SC-JEPA, as its richer architecture incurs higher computational cost than a single-backbone baseline. Our design instead prioritizes early-warning performance and representation stability, with the runtime overhead becoming more manageable in deployment scenarios that support moderate to large batch sizes.