VISTA: Variable-length Interpretable Scalable Temporal Analysis for Industrial Time Series Event Clustering

Hao Huang
hao.huang1@gevernova.com
GE Vernova Advanced Research Center
Niskayuna, NY, USA

Naresh Iyer
iyerna@gevernova.com
GE Vernova Advanced Research Center
Niskayuna, NY, USA

Abstract

Clustering multivariate time series by operational conditions or fault types is important across industrial domains, but existing methods struggle with long, variable-length recordings, localized signatures, and interpretability. We present VISTA (Variable-length Interpretable Scalable Temporal Analysis), an unsupervised time series clustering model for industrial events that jointly captures local and global structure. First, a convolutional encoder embeds each sliding window. Then an eviction-attention mechanism, trained end-to-end via a straight-through estimator, maintains a bounded pool of the top-P most informative embeddings per series, and truncated SVD on this pool produces a fixed-dimensional representation that captures the principal directions of variation across the pooled embeddings of each time series, naturally accommodating variable length. Finally, K-means is applied to all the representations to assign a label to each event time series. Window- and series-level contrastive losses train the pipeline end-to-end. The bounded pool keeps memory usage constant per series and computation linear with sequence length. Intra-pool self-attention scores yield faithful temporal localization ("when"), and a per-cluster channel-attribution score identifies the sensors most relevant to each cluster ("what"). On five industrial datasets, VISTA outperforms all twelve baselines in time series event clustering. Faithfulness tests confirm that these interpretable scores identify windows and channels relevant to VISTA's clustering output and align with domain-expert physical interpretations.

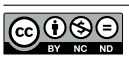
CCS Concepts

• **Computing methodologies** → **Dimensionality reduction and manifold learning**; **Cluster analysis**; • **Mathematics of computing** → **Time series analysis**; • **Applied computing** → *Engineering*.

Keywords

time series clustering, industrial event data, interpretable learning

ACM Reference Format:
Hao Huang and Naresh Iyer. 2026. VISTA: Variable-length Interpretable Scalable Temporal Analysis for Industrial Time Series Event Clustering. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management (CIKM '26), November 7–11, 2026, Rome, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3799682.3840568



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
CIKM '26, Rome, Italy.
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2539-5/2026/11
https://doi.org/10.1145/3799682.3840568

1 Introduction

Modern industrial systems—chemical plants, hydraulic rigs, wind turbines—are densely instrumented, often logging dozens to hundreds of sensors at sub-second rates. When an industrial event occurs (a process upset, a component degradation, a turbine trip), the monitoring system retains a multivariate time series of the incident. At scale, manual review of these events is too slow, costly, and infeasible—demanding automated event analysis. One natural first step in analyzing these events is to group them by time series clustering, enabling operators to spot failure modes and potential root causes. However, off-the-shelf methods face several recurring challenges.

First, *scale*. A single event recording can easily reach thousands of timestamps across dozens of channels. Full-sequence self-attention [33] costs $O(L^2)$ where L is the length of the time series, which blows up quickly, and we need to process not just one, but hundreds or thousands of events. Second, *locality*. The discriminative signature, whether produced by a transient event (e.g., a vibration spike, a pressure surge) or by a gradual process (e.g., thermal degradation, tool wear, sensor drift), is often concentrated in a limited portion of the recording, while the rest is comparatively uninformative/irrelevant. Methods that aggregate uniformly over the entire sequence risk diluting the evidence that matters most. Third, *interpretability*. Engineers want to know why two events are assigned to the same cluster, specifically which temporal segments and which sensor channels are most indicative of the underlying fault or operational signature. No existing method addresses all three, motivating VISTA.

This paper presents **VISTA** (Variable-length Interpretable Scalable Temporal Analysis), a three-stage representation pipeline, window embeddings → eviction-attention-bounded pool → SVD compression, that jointly addresses the challenges above. A convolutional encoder maps each sliding window to a local embedding. Eviction attention maintains a bounded set of the most discriminative embeddings per time series, and intra-pool self-attention followed by truncated SVD produces a fixed-dimensional representation capturing global structure regardless of recording length. Specifically:

(1) **Bounded-memory eviction with faithful interpretability.** End-to-end trainable eviction keeps memory usage constant per series. Intra-pool self-attention produces a consensus-based importance score for each surviving embedding, enabling faithful temporal localization of *when* each discriminative pattern occurs. In addition, a per-cluster channel-attribution score Γ_c, derived from an exact additive decomposition of the encoder's first convolutional layer and re-weighted by the same self-attention consensus, reveals *what* (channels) drives each cluster assignment by identifying the most influential sensor channels.

(2) **Two-level contrastive learning.** Window-level and time-series-level contrastive losses train the pipeline end-to-end, with memory banks supplying diverse negatives and similarity-based reweighting to mitigate false negatives.
(3) **Evaluation across multiple industrial datasets and SOTA methods.** We evaluate on ZeMA Hydraulic System [13] (hydraulic degradation), NASA Milling Wear [1] (manufacturing), TEP [11] (process control), N-CMAPSS [5] (turbofan engine degradation) and a proprietary wind turbine trip dataset (energy systems), demonstrating consistent improvements over twelve baselines. Variable-length datasets validate the effectiveness of VISTA's variable-length handling.

2 Related Work

2.1 Time Series Clustering and Contrastive Learning

Shape-based methods such as k-means with DTW [27] operate on raw waveforms but scale poorly to long sequences. Characteristic-based methods [36] extract global features (trend, seasonality, serial correlation) efficiently yet at a coarser abstraction. Deep approaches—DTC [22], T2Vec [19]—learn representations end-to-end but require the full sequence upfront. DTCR [21] uses an LSTM seq-to-seq autoencoder with a K-means-friendly auxiliary loss, establishing the canonical RNN-based clustering baseline; however, RNN hidden states compress arbitrarily long sequences into a fixed bottleneck, causing information loss on long industrial recordings. Contrastive learning alleviates supervision requirements: TS-TCC [12] contrasts temporal and contextual views; TS2Vec [39] performs hierarchical contrastive learning aggregated via max pooling; CDCC [25] jointly contrasts temporal and frequency domains but processes full sequences and requires k at training time; TRACE [28] targets repeated-trial neuroscience data not present in industrial settings. VISTA differs from these in using attention-based eviction and fixed-dimensional output by SVD compression for variable-length inputs. Motivated by prior work on false-negative mitigation [8], we apply similarity-based negative reweighting.

2.2 Patch-Based Models and Matrix Sketching

Dividing time series into local patches and processing them as tokens has proven effective: PatchTST [23] applies a Transformer to overlapping patches; TimeMAE [7] extends masked autoencoders to time series patches. Both share VISTA's local-window intuition but apply $O(T^2)$ self-attention over all T patches; VISTA's bounded pool avoids this quadratic cost by capping per-step attention at the pool size (details in Section 3.2). SVD-based dimensionality reduction for time series was explored in [16]. Matrix sketching [20, 37] provides efficient low-rank approximations. Differentiable SVD has been used for second-order pooling [15], and Wang et al. [35] proposed a robust backward formula via Taylor expansion.

2.3 Interpretability in Industrial Clustering

SCADA analyses use engineered statistics [17, 29]; FeatTS and Time2Feat use interpretable features and feature selection, with optional expert semi-supervision in Time2Feat [3, 30]. Shapelet methods represent clusters by discriminative subsequences, including multiview unsupervised learning [40]. VISTA instead learns end-to-end, label-free temporal and channel diagnostics from its clustering computation, while its bounded pool retains localized patterns from arbitrary-length recordings with constant per-series memory.

3 Methodology

Figure 1 illustrates the design of VISTA. Let $\mathcal{X} = \{\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(N)}\}$ denote N multivariate time series, where $\mathbf{X}^{(i)} \in \mathbb{R}^{M \times L_i}$ has M channels over L_i timestamps. Without labels, the goal is to learn a mapping $f : \mathbb{R}^{M \times L_i} \to \mathbb{R}^d$ such that same-cluster time series are close and different-cluster ones are far apart. Cluster assignments are obtained by applying K-means to the learned embeddings. K denotes the number of clusters; its automatic estimation will be discussed in Section 4.8. For complexity statements, T_i denotes the number of sliding windows extracted from series i (i.e., $T_i = \lfloor (L_i - W)/S \rfloor + 1$).

3.1 Window-Based Ingestion and Window Encoder

Rather than processing the full time series as a monolithic input, VISTA divides each recording into overlapping windows and encodes them independently. For time series $\mathbf{X}^{(i)}$, a sliding window of size W with step S extracts windows $\mathbf{w}^{(i)} \in \mathbb{R}^{M \times W}$, each of which is encoded independently. Since the window encoder g_θ is applied identically to every window, we suppress the series and window indices in this subsection and simply write $\mathbf{w}$; a window index j is re-introduced in Section 3.2 once windows are stored in the pool. Each window $\mathbf{w}$ is processed by a shared encoder g_θ which consists of three components:

Dilated Convolutional Stack. A sequence of n_{conv} 1D convolutional layers with configurable output channels, kernel sizes, strides and dilations. Each layer applies convolution, batch normalization, ReLU activation, and dropout. The dilated convolutions produce feature maps $\mathbf{h}_{\text{conv}}$, reducing the temporal dimension from W to $W' \ll W$.

Intra-Window Self-Attention. The reduced feature maps $\mathbf{h}_{\text{conv}}$ of each window are treated as a sequence of W' tokens. A learnable positional encoding is added, followed by a multi-head self-attention layer with residual connection and layer normalization:

$$\begin{aligned} \tilde{\mathbf{h}}_{\text{conv}} &= \mathbf{h}_{\text{conv}} + \mathbf{E}_{\text{win}}, \\ \mathbf{h} &= \text{LayerNorm}\left(\tilde{\mathbf{h}}_{\text{conv}} + \text{Attention}(\tilde{\mathbf{h}}_{\text{conv}}, \tilde{\mathbf{h}}_{\text{conv}}, \tilde{\mathbf{h}}_{\text{conv}})\right), \end{aligned} \tag{1}$$

where $\mathbf{E}_{\text{win}} \in \mathbb{R}^{W' \times c_{n_{\text{conv}}}}$ is the learnable intra-window positional encoding.

Multi-Statistic Pooling. The W'-length sequence is collapsed into a single vector by concatenating mean, max, and standard deviation along the temporal dimension:

$$\mathbf{z}_j = \text{LayerNorm}\left(\mathbf{W}_p \left[\text{mean}(\mathbf{h}); \max(\mathbf{h}); \text{std}(\mathbf{h})\right]\right) \in \mathbb{R}^{d_e}, \tag{2}$$

where j indexes the window and $\mathbf{W}_p \in \mathbb{R}^{d_e \times 3c_{n_{\text{conv}}}}$ is a learnable projection. Max and standard deviation capture spike amplitudes and variability that mean-only pooling washes out. The compact d_e bottleneck (1) regularizes the encoder, benefiting contrastive learning [6], and (2) keeps pool memory ($P \cdot d_e$) and attention cost $O(P^2 \cdot d_e)$ low. A two-layer MLP maps $\mathbf{z}_j$ to a contrastive embedding $\mathbf{h}_j$ used only during training (refer to Window-Level Contrastive Loss in Section 3.4). At inference we use $\mathbf{z}_j$ directly.

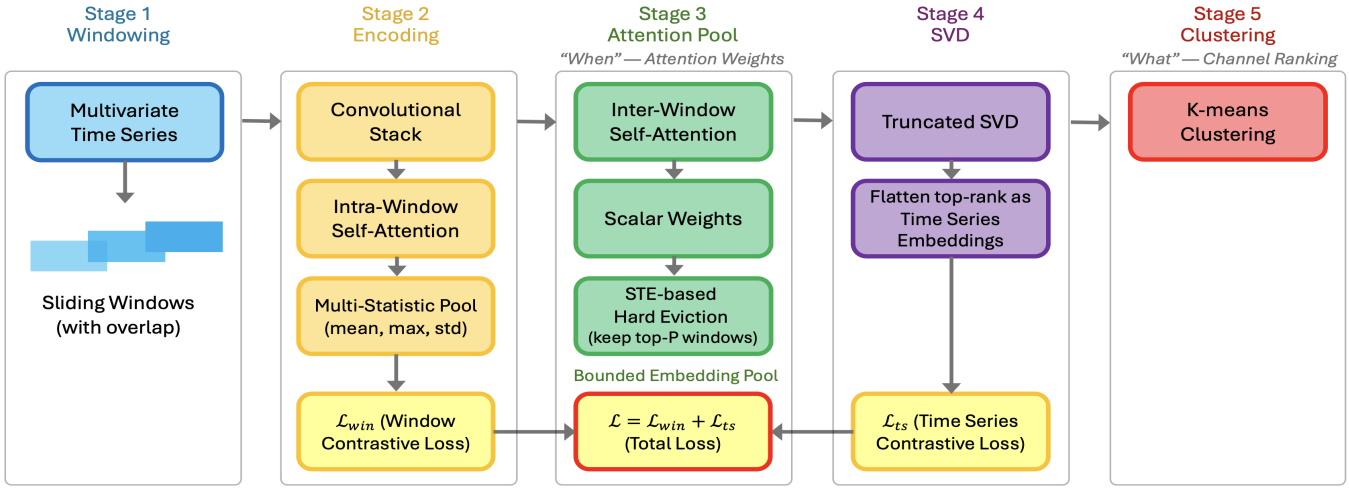


Figure 1: VISTA architecture overview. Multivariate time series are divided into sliding windows (Stage 1) and encoded into window embeddings via dilated convolution stack (Stage 2). Eviction attention—where all candidates (existing pool members and new arrivals) jointly attend over the combined set—retains the top-P most discriminative window embeddings via an end-to-end STE-trained selection step (Stage 3). Intra-pool self-attention refines each retained embedding by querying itself against all survivors. Truncated SVD on the refined embeddings produces fixed-dimensional representations as time series embeddings (Stage 4) for K-means clustering (Stage 5). Two-level contrastive losses train the pipeline end-to-end, with intra-pool self-attention importance scores providing temporal ("when") interpretability and a first-layer channel-attribution score providing sensor ("what") interpretability.

3.2 Embedding Pool with Attention-Based Eviction

Each time series maintains its own embedding pool Q of maximum capacity P. At every k-th window addition, eviction attention jointly ranks the k new arrivals and the current pool members, retaining the top-P most discriminative embeddings. The survivors are then refined by intra-pool self-attention and compressed via SVD (Section 3.3). The eviction procedure is as follows.

Positional Injection. Learnable positional encodings $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{T_{\max} \times d_e}$ are added using the memorized original position $\hat{\mathbf{z}}_j = \mathbf{z}_j + \mathbf{E}_{\text{pos}}[j]$. Thus, eviction and intra-pool attention use each window's original temporal position regardless of pool insertion order (shuffled during training and sequential at inference; Section 3.4).

Eviction Attention. Let $\hat{\mathbf{Z}}_{\text{all}} \in \mathbb{R}^{(|Q|+k) \times d_e}$ denote the concatenation of existing pool members and the k newly arrived position-injected embeddings. We apply a single self-attention block over the full candidate set so that every candidate—both incumbents and newcomers—is re-contextualized against the same pool:

$$\mathbf{C} = \text{Attention}(\hat{\mathbf{Z}}_{\text{all}}, \hat{\mathbf{Z}}_{\text{all}}, \hat{\mathbf{Z}}_{\text{all}}) \in \mathbb{R}^{(|Q|+k) \times d_e}, \tag{3}$$

yielding one context-aware vector $\mathbf{c}_j \in \mathbb{R}^{d_e}$ per candidate. Joint scoring ensures incumbents and newcomers are evaluated under the same attention weights in one forward pass, so the eviction decision is not biased by stale, un-refreshed scores for incumbents. Joint attention also tolerates small intra-step encoder drift because each candidate is content-addressably re-contextualized against the current pool, and rows whose features no longer fit can be down-weighted by the learned attention. The cost is $O((|Q|+k)^2 \cdot d_e)$ per update step, which is bounded by a constant since $|Q| \le P$ and k are fixed, and therefore linear in T_i overall (over the $\lceil T_i/k \rceil$ update steps per series). A two-layer MLP maps each $\mathbf{c}_j$ to a softmax-normalized eviction score,

$$\alpha_j = \frac{\exp(s_j)}{\sum_\ell \exp(s_\ell)}, \quad s_j = \mathbf{v}^\top \tanh(\mathbf{W}_s \mathbf{c}_j + \mathbf{b}_s), \tag{4}$$

and the $\max(0, |Q| + k - P)$ lowest-scoring candidates are removed. Hard eviction is made differentiable with a binary mask $m_j \in \{0, 1\}$ and a straight-through estimator (STE) [2], so gradients flow to all candidates while only the top-P candidates proceed to downstream stages once the pool reaches capacity. This attention block is used only for eviction scoring; a separate, independently parameterized self-attention block refines the survivors before SVD (Section 3.3).

3.3 SVD Compression

After eviction, a second multi-head self-attention block—hereafter the *intra-pool self-attention*, with independent parameters from the eviction-attention block in Eq. 3—runs over the P retained survivors to produce the refined embeddings fed to SVD. Each embedding serves as its own query and attends to all survivors as keys and values, so the refined representation $\tilde{\mathbf{z}}_j \in \mathbb{R}^{d_e}$ captures both the window's local signature and its relational context within the curated pool. The cost is $O(P^2 \cdot d_e)$ per update step, yielding $O((T_i/k) \cdot P^2 \cdot d_e)$ total per time series per epoch—linear in T_i since P, k, and d_e are constants. The self-attention-refined embeddings are stacked into a matrix $\tilde{\mathbf{Z}}_Q \in \mathbb{R}^{|Q| \times d_e}$ and compressed via a randomized rank-R decomposition: $\tilde{\mathbf{Z}} \approx \mathbf{U}\Sigma\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{|Q| \times R}$, $\Sigma \in \mathbb{R}^{R \times R}$, and $\mathbf{V} \in \mathbb{R}^{d_e \times R}$. Unlike weighted mean pooling (which captures only the centroid), SVD captures the principal directions of variation—important because two time series with the same mean embedding but different variation patterns should map to different representations. The ablation study (Table 5(a)) confirms a substantial gain over mean and mean+variance pooling.

Time Series Representation. The final representation is constructed by scaling the right singular vectors by the singular values: $\mathbf{r} = \text{flatten}(\mathbf{V\Sigma}) \in \mathbb{R}^{R \cdot d_e}$. $\mathbf{U}$ encodes contributions of pool rows, whose ordering changes under eviction during training with shuffled windows and therefore has no physical temporal meaning. We therefore exclude $\mathbf{U}$; temporal order is modeled earlier through original-position encodings in eviction and intra-pool attention. The output dimension $R \cdot d_e$ is fixed regardless of recording length. We emphasize that fixed dimensionality alone is generic—any global pool (mean/max) achieves it. SVD's specific advantage lies in preserving the principal directions of variation across the pool. This effect is isolated by the ablation in Table 5(a): using NMI (higher is better), mean pooling achieves 0.781, mean+variance pooling 0.800, and SVD 0.887, all at the same output dimension and with the same bounded pool.While the pool contains fewer than R embeddings ($|Q| < R$, cold start), rank-R truncated SVD is undefined, so $\mathcal{L}_{\text{ts}}$ is skipped and only $\mathcal{L}_{\text{win}}$ trains the encoder (Section 3.4).

SVD Backward Pass and Training Stability. End-to-end training requires gradients through the low-rank decomposition, which are known to be unstable when singular values are close [15, 35]. Since $\mathbf{U}$ is excluded from the forward pass, its gradient contribution vanishes, removing the most ill-conditioned term. We further stabilize the remaining $\mathbf{V}$- and Σ-components by clamping pairwise squared-singular-value differences at ϵ=10^{-6} and gradient-norm-clipping at 1.0. Implementation uses PyTorch's differentiable `torch.svd_lowrank` with q=R and two subspace iterations. Across all five datasets and 100 runs total, the median SVD-gradient ℓ_2-norm was 0.12 with 99th percentile 0.41 (well below the clip threshold of 1.0) and zero divergences were observed.

3.4 Two-Level Contrastive Learning

VISTA trains with two-level contrastive objectives: for each window and each time series, an augmented view forms a positive pair, and the model pulls positives together while pushing apart negatives drawn from the rest of the mini-batch and two FIFO memory banks.

Training: global shuffling. At the start of each epoch, all windows from all time series are shuffled into a single global stream and processed in mini-batches of size B. Shuffling both diversifies in-batch negatives and regularizes the eviction scorer, forcing it to rank candidates by content and positional encoding rather than by which series or which arrival batch they came from. Table 5(a) confirms that shuffled training outperforms sequential training. **Inference: sequential ordering.** At inference, windows are added in natural order, ensuring deterministic representations.

Data Augmentation. For each input window $\mathbf{w}_j$, we generate an augmented view $\mathbf{w}'_j$ by applying a composition of augmentations: (1) additive Gaussian noise, (2) random channel dropout with fixed probability, and (3) random cropping that retains a contiguous subsegment of length $\rho \cdot W$ (with ρ uniformly sampled from $[0.8, 1.0]$), followed by linear interpolation back to length W. The augmented window is processed through the same encoder (Section 3.1) to produce the augmented embedding and its projected counterpart $\mathbf{h}'_j$ (used in window-level contrastive loss).

NT-Xent Loss with False-Negative Mitigation. Some in-batch negatives may share the anchor's cluster (false negatives), degrading contrastive learning. Motivated by [8], we downweight negatives with high positive cosine similarity to the anchor while retaining full weight for nonpositively correlated negatives. Let $\mathbf{a}_b$ and $\mathbf{a}'_b$ denote a generic positive pair of ℓ_2-normalized embeddings ($\mathbf{a}_b$ is the anchor and $\mathbf{a}'_b$ its augmented view); the same loss form is instantiated at the window level ($\mathbf{a}_b$=$\mathbf{h}_j$, $\mathbf{a}'_b$=$\mathbf{h}'_j$, $\mathcal{N}$=$\mathcal{M}_{\text{win}}$) and at the series level ($\mathbf{a}_b$=$\mathbf{r}^{(i)}$, $\mathbf{a}'_b$=$\mathbf{r}'^{(i)}$, $\mathcal{N}$=$\mathcal{M}_{\text{ts}}$). For positive pairs $\{(\mathbf{a}_b, \mathbf{a}'_b)\}_{b=1}^{B}$ and memory-bank negatives $\mathcal{N}$:

$$\ell_b = -\log \frac{\exp(s_{bb'}/\tau)}{Z_b},$$
$$Z_b = \exp(s_{bb'}/\tau) + \sum_{b'' \neq b}^{B} w_{bb''} \exp(s_{bb''}/\tau) + \sum_{\mathbf{n} \in \mathcal{N}} w_{bn} \exp(s_{bn}/\tau), \tag{5}$$

where $\text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\top \mathbf{v}/(\|\mathbf{u}\|\|\mathbf{v}\|)$ is cosine similarity, $\mathbf{a}_{b''}$ (for $b'' \neq b$) denotes the anchor of a different in-batch pair used as a negative for $\mathbf{a}_b$, $\mathbf{n} \in \mathcal{N}$ denotes a memory-bank negative embedding, $s_{bb'} = \text{sim}(\mathbf{a}_b, \mathbf{a}'_b)$, $s_{bb''} = \text{sim}(\mathbf{a}_b, \mathbf{a}_{b''})$, $s_{bn} = \text{sim}(\mathbf{a}_b, \mathbf{n})$, and $w_{bb''} = \max(1 - [\max(s_{bb''}, 0)]^\gamma, \epsilon_w)$ (similarly for w_{bn}) with $\gamma = 2$, $\epsilon_w = 0.05$, and temperature $\tau = 0.5$. The total loss averages over both directions: $\mathcal{L}_{\text{NT-Xent}} = \frac{1}{2B} \sum_b (\ell_b + \ell'_b)$, where ℓ'_b is the symmetric loss obtained by swapping the roles of $\mathbf{a}_b$ and $\mathbf{a}'_b$ as anchor and positive.

Window-Level Contrastive Loss. Within each batch of B windows, projected embeddings $\{\mathbf{h}_j, \mathbf{h}'_j\}$ serve as positives and in-batch negatives, supplemented by a global FIFO memory bank $\mathcal{M}_{\text{win}}$ of size B_{win}. The loss is defined by $\mathcal{L}_{\text{win}} = \mathcal{L}_{\text{NT-Xent}}(\{\mathbf{h}_j\}, \{\mathbf{h}'_j\}; \mathcal{M}_{\text{win}})$.

Time-Series-Level Contrastive Loss. Every k window additions, the eviction–self-attention–SVD pipeline produces a time-series representation $\mathbf{r}^{(i)}$ and its augmented counterpart $\mathbf{r}'^{(i)}$, compared against a global FIFO memory bank $\mathcal{M}_{\text{ts}}$: $\mathcal{L}_{\text{ts}} = \mathcal{L}_{\text{NT-Xent}}(\mathbf{r}^{(i)}, \mathbf{r}'^{(i)}; \mathcal{M}_{\text{ts}})$. The augmented representation $\mathbf{r}'^{(i)}$ is produced by routing the augmented windows $\{\mathbf{w}'_j\}$ through the same encoder and attention parameters but into a separate pool $Q'^{(i)}$ with its own eviction, self-attention, and SVD state, so that $\mathbf{r}^{(i)}$ and $\mathbf{r}'^{(i)}$ represent different stochastic augmentation views of the same series.

Total Loss and Training Protocol. At each batch, the total loss is:

$$\mathcal{L} = \lambda_{\text{win}} \cdot \mathcal{L}_{\text{win}} + \lambda_{\text{ts}} \cdot \mathcal{L}_{\text{ts}} \cdot \mathbb{1}[\text{update step}], \tag{6}$$

where $\mathbb{1}[\text{update step}]$ is 1 every k window additions and 0 otherwise. The N time series are split 75%/25% into training and validation sets. Early stopping selects the checkpoint with the lowest validation contrastive loss. At inference, the best model encodes all N time series to produce final SVD representations for K-means clustering. At the start of each epoch, every pool $Q^{(i)}$ is cleared while the memory banks $\mathcal{M}_{\text{win}}$ and $\mathcal{M}_{\text{ts}}$ are not cleared.

Why pools require clearing. Pool embeddings from a single time series are stacked row-wise and jointly decomposed by SVD, which treats all P rows of $\bar{\mathbf{Z}}$ as samples from a single distribution and, unlike attention, has no learned mechanism to down-weight rows from a stale snapshot. Within one epoch, encoder drift between consecutive optimizer steps is small and the eviction / intra-pool self-attention layers tolerate it via content-addressed re-weighting (Section 3.2). Across an entire epoch, however, drift accumulates over hundreds of steps, so if some rows come from the current encoder g_{θ_t} and others from a much-earlier snapshot $g_{\theta_{t-\Delta}}$, the matrix mixes two misaligned embedding spaces and SVD returns principal directions of a distribution that may never have existed. Empirically, disabling

pool clearing degrades NMI by 6.7 points on average and by 9.2 on ZeMA-Cooler, where the 3-class distinction depends on subtle principal-direction differences.

Why memory banks tolerate drift. Memory-bank entries serve only as contrastive negatives in the denominator of Eq. 5, where encoder drift shifts all negatives approximately uniformly so relative dissimilarity is preserved. The NT-Xent similarity is cosine (unit-normalized), so per-entry magnitude drift is absorbed and only angular ranking matters, and FIFO replacement further bounds staleness to one epoch. Clearing memory banks each epoch in fact degrades NMI by 3.9 points on average, primarily by reducing negative-pool diversity. We therefore adopt this asymmetric clearing strategy as a practical heuristic validated by ablation.

3.5 Interpretability

VISTA provides model-native interpretability via two scores—a temporal localization score read off from intra-pool self-attention, and a channel-attribution score read off from an exact additive decomposition of the encoder's first convolutional layer—both derived from computations the forward pass already performs to produce the clustering embedding: **1) Temporal Localization ("when").** Let $\mathbf{A} \in \mathbb{R}^{P\times P}$ denote the attention weight matrix from the intra-pool self-attention block (the survivor-refinement block in Section 3.3), where A_{ij} is the weight that embedding i places on embedding j. We define the importance score of embedding j as its column-sum:

$$\beta_j = \frac{1}{P}\sum_{i=1}^{P} A_{ij}, \tag{7}$$

which measures how much embedding j is attended to by all other pool members. Unlike the eviction scores α_j, which rank candidates against the pre-eviction pool composition (including windows about to be discarded), the importance scores β_j are computed after eviction over the final P survivors, providing a stable consensus signal for interpretability. Since the pool memorizes each embedding's original positional index, high-β_j windows can be mapped back to the original time axis, localizing when discriminative patterns occur. **2) Channel Attribution ("what").** Channel identity is carried by the input only at the first convolutional layer of the encoder (Section 3.1). Because that first layer is linear in the input-channel axis, its pre-activation for window j of series i admits an exact additive decomposition

$$\mathbf{h}^{(i)}_{\text{pre},j} = \sum_{m=1}^{M} \underbrace{\mathbf{W}^{(1)}_{:,m,:} * \mathbf{w}^{(i)}_{j,m,:}}_{\mathbf{c}^{(i)}_{j,m}} + \mathbf{b}^{(1)}, \tag{8}$$

where $\mathbf{W}^{(1)}$ is the first-layer kernel and $\mathbf{w}^{(i)}_{j,m,:}$ is channel m of window j. We define the per-window channel weight as the normalized magnitude of each additive term,

$$s^{(i)}_{j,m} = \frac{\left\|\mathbf{c}^{(i)}_{j,m}\right\|_F}{\sum_{m'=1}^{M}\left\|\mathbf{c}^{(i)}_{j,m'}\right\|_F}, \tag{9}$$

which quantifies how much channel m contributes to the encoder's first-layer response at window j. Aggregating over the retained pool windows using the temporal-consensus weights β_j (Eq. 7) yields a per-series channel score, and averaging within each K-means cluster c yields the cluster-level channel attribution

$$\gamma^{(i)}[m] = \sum_{j\in\mathcal{Q}^{(i)}} \beta^{(i)}_j s^{(i)}_{j,m}, \qquad \Gamma_c[m] = \frac{1}{|C_c|}\sum_{i\in C_c}\gamma^{(i)}[m], \tag{10}$$

where $C_c = \{i : c(i) = c\}$ is the set of series assigned to K-means cluster c. $\Gamma_c \in \mathbb{R}^M$ ranks sensors by how much they drive the encoder features of the windows the model itself attended to for cluster c. Like β_j, Γ_c is read off the internal state of the trained model without extra forward passes, additional parameters, or ground-truth labels (the per-cluster aggregation in Eq. 10 uses only the predicted K-means assignments). Unlike gradient-based saliency, it relies on an exact decomposition rather than a linearization. The decomposition is exact only at the first 1-D conv pre-activation, so downstream nonlinearities, pooling, intra-pool self-attention, STE eviction, and SVD compression may further amplify or dampen individual channels; the β_j re-weighting in Eq. 10 partially accounts for the intra-pool self-attention step, but Γ_c remains a first-layer attribution and should not be read as a full layer-wise causal decomposition. We therefore validate Γ_c's relevance to the final clustering output—not just the first-layer response—with an input-level masking test that re-runs the entire frozen pipeline (Section 4.5).

4 Experiments

4.1 Dataset and Baselines

ZeMA Hydraulic System—Cooler Condition (ZeMA-Cooler) [13]: 60-second cycles from an industrial hydraulic test rig. We focus on 3 cooler degradation states. **NASA Milling Wear (Milling) [1]:** CNC milling runs with spindle-current, vibration, and acoustic-emission sensors. There are 8 clusters over depth-of-cut, feed rate, and workpiece material. **Tennessee Eastman Process (TEP) [11]:** fault detection on the TEP simulation (41 measured + 11 manipulated variables, 12 fault clusters). We upsample each series to L=5,000 timestamps to stress-test long-sequence behavior. **Wind Turbine Trip Events (WindTrip):** proprietary, variable-length SCADA trip recordings from an operational wind turbine fleet. There are 9 fault categories, including pitch, generator, and grid-related trips. **N-CMAPSS [5]:** publicly available turbofan run-to-failure data (DS01–DS03) spanning seven degradation modes (single- and multi-mode HPT/LPT/fan failures), recorded at flight-cycle resolution. Recording lengths vary naturally from $L \approx$ 6,000 to $L \approx$ 38,000. After filtering units with fewer than 50 cycles, 274 units remain. Full statistics appear in Table 1. The five datasets span ≈2.0 billion data points in total.

Preprocessing. For every dataset, all channels are z-scored using per-channel mean/std computed on the 75% training partition and applied to both splits to prevent leakage. Identical inputs are fed to VISTA and all baselines. All methods are fitted without labels. The 75%/25% split is used for leakage-safe normalization and, where applicable, unsupervised model selection and early stopping based on validation loss. Labels are accessed solely at evaluation time to compute NMI/ARI.

Baselines. We compare against 12 methods spanning four families: (i) *shape/statistical*: DTW+K-means [27], K-Shape [24], ML-GSSM [32]; (ii) *contrastive/SSL(Self-Supervised Learning)*: TS-TCC [12], TS2Vec [39], SoftCLT [18], CDCC [25], TimeMAE [7];

Table 1: Dataset statistics. N: number of time series; M: number of channels; L: length (timestamps); K: number of clusters. Total points = $N \times M \times L$ (approximate for variable-length datasets, using midpoint of L).

Dataset	N	M	L	K	Total points
ZeMA-Cooler	2,205	7	6,000	3	92.6 M
Milling	167	6	9,000	8	9.0 M
TEP	6,000	52	5,000	12	1.56 B
WindTrip	907	32	6K–10K	9	≈232 M
N-CMAPSS	274	20	6K–38K	7	≈121 M

(iii) *temporal modeling*: TimesNet [38]; (iv) *low-rank / recurrent clustering*: DTSS [14] (matrix sketching), DTCR [21] (canonical LSTM seq-to-seq clustering), and **LSTM-CL**—a Bi-LSTM encoder we train with VISTA's exact NT-Xent loss (Eq. 5), augmentations, and false-negative mitigation, using the final hidden state as the series embedding. Like TS2Vec, LSTM-CL handles variable length natively, isolating whether VISTA's advantage stems from its specific architecture rather than from avoiding fixed-length preprocessing.

Baselines Implementation Details. For trainable baselines, we use the same 75%/25% train/validation split as VISTA and apply early stopping based on validation loss. Non-neural clustering and statistical baselines, including DTW+K-means, K-Shape, and ML-GSSM, are fitted using their standard optimization or convergence procedures and do not use validation-loss-based early stopping. For baselines requiring fixed-length input, variable-length recordings (WindTrip, N-CMAPSS) are zero-padded to the maximum length within each dataset; attention-based baselines additionally receive a padding mask. Two baselines handle variable length natively without padding: TS2Vec (via per-series hierarchical pooling) and LSTM-CL (via recurrent sequential processing). To verify that zero-padding does not systematically disadvantage fixed-length baselines, we additionally evaluate six representative baselines under truncation and resampling strategies (Section 4.4). DTCR is trained with the auxiliary K-means-friendly losses from [21]. For every baseline—including the encoder depth and width of all neural baselines—hyperparameters are tuned with the same leave-one-dataset-out (LODO) protocol used for VISTA (Section 4.2): for each baseline–target-dataset pair, the configuration is selected by lowest average validation loss over the other four datasets, with the target's validation loss excluded. Each baseline receives the same tuning budget as VISTA—at least as many hyperparameter axes and grid points—searched within the ranges recommended by the baseline's original paper, to avoid over-tuning VISTA or under-tuning baselines. All baselines were run for 20 random trials, and mean and standard deviation were recorded.

4.2 Evaluation Metrics and Implementation

Evaluation Metrics. We use two standard clustering evaluation metrics: **Normalized Mutual Information (NMI)**: Measures the mutual dependence between predicted and ground-truth partitions, normalized to $[0, 1]$. **Adjusted Rand Index (ARI)**: Measures pairwise agreement between partitions, adjusted for chance, with 1 indicating perfect agreement. For final evaluation of our VISTA, the best model (selected via early stopping on 25% validation loss) re-encodes all N time series—including training samples—to produce final SVD representations. K-means is then run on all N embeddings with the ground-truth number of clusters K, and NMI/ARI are computed over all N assignments, following standard practice in unsupervised clustering benchmarks. Representation-learning baselines similarly encode all N time series before final K-means clustering, while direct clustering baselines are fitted to all N series using their respective procedures, including DTW+K-means. Automatic K estimation is included in Section 4.8.

Label-free hyperparameter selection via leave-one-dataset-out (LODO). A single candidate grid (over P, R, W, S, etc.) is defined a priori. For each candidate configuration, VISTA is trained on all five datasets independently (each with its own 75%/25% train/val split). For each target dataset we then select the configuration with the lowest average validation loss over the other four datasets—a separate selection per LODO fold, so the reported configuration can differ across target datasets, and the target dataset's validation loss is never used for selection. **No ground-truth labels are used at any stage** of training, early stopping, or hyperparameter selection. Labels are used exclusively for computing the final NMI and ARI metrics.

VISTA Implementation Details. VISTA is implemented in PyTorch. The configuration below is the LODO-selected setting for the ZeMA-Cooler fold; it also serves as the reference configuration for the ablation and sensitivity studies in Section 4.7. The encoder uses three dilated convolutional layers with output channels $[64, 64, 64]$, kernel sizes $[5, 5, 5]$, strides $[1, 1, 2]$, and dilations $[1, 2, 2]$. The window size is $W = 120$, step size $S = 60$, window embedding dimension $d_e = 32$ (projected from $3c_{n_{\text{conv}}} = 192$ via $\mathbf{W}_p$), and SVD rank $R = 3$, yielding a time-series embedding dimension of $R \times d_e = 96$. The pool capacity is $P = 40$. Memory banks hold 256 (window) and 128 (time-series) embeddings. The update interval is $k = 5$ window additions. Training uses Adam (lr 10^{-4}, weight decay 10^{-5}), batch size $B = 32$, gradient norm clipping at 1.0, and SVD backward clamping at 10^{-6}. Loss weights are $\lambda_{\text{win}} = 1.0$, $\lambda_{\text{ts}} = 5.0$. All tests are repeated 20 times with different random seeds for both model initialization and the 75%/25% train/validation split.

4.3 Clustering Results

Table 2 presents the NMI and ARI clustering performance comparison, reported as mean ± standard deviation over 20 runs. All metrics are computed on all N time series per dataset (see Section 4.2 for protocol details). VISTA ranks first on all five datasets in both NMI and ARI.

For TEP, ZeMA-Cooler and Milling, VISTA leads on all three: NMI 0.941 on TEP, 0.929 on ZeMA-Cooler (an 8-point lead over DTSS), and 0.824 on Milling (a 10-point lead over CDCC). Shape-based methods (K-Shape, DTW) collapse on the 52-channel TEP input. For WindTrip, VISTA achieves the largest gap, leading second-best (DTSS) by 12 NMI points (0.864 vs. 0.746). For N-CMAPSS, VISTA achieves NMI 0.878, a 12-point lead over DTSS (0.761).

Statistical testing. Our primary significance evidence is a per-dataset Wilcoxon signed-rank test over 20 paired runs against the respective second-best baseline, which yields $p < 0.01$ on TEP, WindTrip, ZeMA-Cooler, and N-CMAPSS and $p < 0.05$ on Milling—well powered at n=20 and not subject to the "few-datasets" limitation. As a cross-dataset complement, we additionally follow the standard

Table 2: Clustering performance comparison (12 baselines). Best results in bold, second best underlined. Results are mean ± std over 20 runs (varying both initialization seeds and train/val splits); metrics are computed on all N time series per dataset. †Statistically significant improvement over the per-dataset second-best method ($p < 0.01$ on TEP, WindTrip, ZeMA-Cooler, N-CMAPSS; $p < 0.05$ on Milling; Wilcoxon signed-rank test over paired runs). A Friedman test across all methods and datasets is significant ($p < 10^{-6}$); see Section 4.3 for Nemenyi post-hoc results.

	TEP		WindTrip		ZeMA-Cooler		Milling		N-CMAPSS	
Method	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
VISTA (ours)	**.941±.022**†	**.923±.025**†	**.864±.014**†	**.803±.018**†	**.929±.023**†	**.908±.020**†	**.824±.029**†	**.742±.034**†	**.878±.019**†	**.821±.023**†
CDCC	.901±.015	.863±.028	.712±.031	.672±.028	.841±.043	.856±.033	.721±.032	.635±.037	.738±.028	.672±.033
DTSS	.877±.020	.842±.023	.746±.026	.702±.032	.851±.035	.830±.029	.708±.037	.591±.045	.761±.025	.698±.030
LSTM-CL	.857±.027	.742±.045	.691±.028	.529±.043	.798±.038	.721±.042	.696±.036	.552±.044	.723±.030	.624±.036
DTCR	.838±.031	.708±.051	.647±.030	.478±.047	.778±.040	.695±.046	.672±.041	.518±.050	.685±.033	.575±.040
TS-TCC	.871±.030	.753±.056	.654±.033	.477±.055	.787±.039	.706±.049	.677±.042	.529±.051	.689±.035	.581±.042
TS2Vec	.885±.025	.779±.042	.671±.029	.503±.048	.819±.036	.745±.044	.689±.038	.547±.046	.706±.031	.602±.038
TimeMAE	.853±.028	.728±.048	.649±.032	.478±.052	.774±.041	.692±.047	.661±.040	.507±.049	.681±.034	.563±.041
SoftCLT	.831±.045	.692±.055	.608±.027	.425±.059	.754±.046	.654±.052	.639±.039	.477±.047	.654±.037	.535±.044
TimesNet	.721±.040	.499±.069	.621±.025	.503±.013	.718±.037	.607±.041	.627±.035	.403±.023	.607±.033	.476±.039
MLGSSM	.752±.029	.689±.071	.648±.014	.441±.062	.663±.020	.564±.032	.602±.042	.410±.047	.593±.030	.452±.036
DTW	.718±.032	.373±.068	.544±.041	.533±.073	.308±.021	.245±.034	.560±.051	.375±.059	.481±.043	.328±.051
K-Shape	.382±.057	.134±.065	.270±.093	.189±.093	.079±.027	.034±.010	.284±.097	.108±.069	.215±.072	.098±.058

Demšar [9] protocol: a Friedman omnibus test strongly rejects equal NMI ranks ($p < 10^{-6}$). With five datasets and 13 methods, the Nemenyi critical difference is ≈8.2 ranks; at α=0.05, VISTA is significantly better than SoftCLT, TimesNet, MLGSSM, DTW, and K-Shape.

4.4 Variable Length Evaluation

To verify that VISTA's advantage on variable-length datasets is not an artifact of suboptimal baseline preprocessing, we evaluate six representative baselines under three preprocessing strategies: (1) *zero-padding* to the maximum length (the default in Table 2), (2) *truncation* to the minimum length within each dataset, and (3) *resampling* (linear interpolation) to a fixed common length (the median length). The natively variable-length baselines (e.g. TS2Vec and LSTM-CL) are additionally evaluated without any preprocessing. All other hyperparameters and evaluation protocols remain identical.

Zero-padding consistently outperforms resampling across fixed-length baselines, while truncation is uniformly worst, especially on N-CMAPSS: resampling distorts event timing, whereas zero-padding preserves native resolution. We therefore use zero-padding in Table 2. Under this setting, VISTA surpasses the strongest baseline, DTSS, by 11.8 and 11.7 NMI points on WindTrip and N-CMAPSS. Native TS2Vec and LSTM-CL slightly outperform their fixed-length variants but remain 15–19 points below VISTA, confirming that VISTA's advantage is architectural rather than a preprocessing artifact.

Table 3: Fair baseline comparison on variable-length datasets. All three preprocessing strategies (Zero-pad, Resample, Truncate) are shown. "Native" denotes methods that handle variable length without preprocessing. Results are mean ± std over 20 runs (varying both initialization seeds and train/val splits). VISTA result (bold) and the Native rows for TS2Vec/LSTM-CL are repeated from Table 2 for reference.

		WindTrip		N-CMAPSS	
Method	**Preprocessing**	NMI	ARI	NMI	ARI
VISTA (ours)	Native	**.864±.014**	**.803±.018**	**.878±.019**	**.821±.023**
DTSS	Zero-pad	.746±.026	.702±.032	.761±.025	.698±.030
	Resample	.731±.027	.658±.034	.742±.027	.647±.032
	Truncate	.697±.030	.641±.037	.674±.031	.618±.036
CDCC	Zero-pad	.712±.031	.672±.028	.738±.028	.672±.033
	Resample	.693±.033	.628±.030	.715±.030	.667±.035
	Truncate	.658±.036	.603±.033	.669±.033	.571±.038
TS2Vec	Zero-pad	.648±.031	.475±.050	.684±.033	.577±.040
	Resample	.635±.033	.479±.053	.662±.035	.569±.043
	Truncate	.609±.036	.431±.056	.609±.038	.531±.047
	Native	.671±.029	.503±.048	.706±.031	.602±.038
LSTM-CL	Zero-pad	.668±.030	.503±.045	.700±.032	.599±.038
	Resample	.655±.032	.487±.048	.686±.034	.563±.040
	Truncate	.624±.035	.456±.052	.634±.037	.533±.044
	Native	.691±.028	.529±.043	.723±.030	.624±.036
TS-TCC	Zero-pad	.654±.033	.477±.055	.689±.035	.581±.042
	Resample	.637±.035	.454±.058	.667±.037	.527±.045
	Truncate	.603±.038	.428±.062	.614±.040	.488±.048
TimeMAE	Zero-pad	.649±.032	.478±.052	.681±.034	.563±.041
	Resample	.638±.034	.462±.055	.663±.036	.542±.044
	Truncate	.605±.037	.432±.059	.597±.039	.468±.047

4.5 Interpretability Analysis and Evaluation

We evaluate whether the two model-native scores β_j and Γ_c faithfully identify windows and channels relevant to VISTA's clustering output, using masking tests that require no ground-truth fault-region annotations, following the rationale of the ERASER benchmark [10]. These intervention-based tests establish faithfulness to VISTA's clustering computation, but they do not establish that the selected windows or sensors are physical causes of the underlying industrial event. We then illustrate how these scores provide interpretable "when and what" diagnostics. Domain experts also confirmed that the identified windows and channels carry physically grounded signatures distinguishing each cluster (e.g., the cooling-transition interval and outlet-temperature/cooling-efficiency channels for the near-failure cooler state).

Quantitative Evaluation. *Attention faithfulness (pool-window mask-and-predict).* If β_j (Eq. 7) genuinely identifies cluster-discriminative

Table 4: Faithfulness of β_j (attention, window axis) and Γ_c (channel axis). Average NMI (mean ± std) after masking; no-masking baseline is .887 ± .021. Lower is stronger faithfulness and is bold. Averaged over five datasets (20 runs for β_j, 5 seeds for Γ_c). Γ_c uses absolute channel counts because M varies per dataset.

	Attention (β_j, window %)			**Channel** (Γ_c, # channels)		
Mask	10%	20%	30%	k=1	k=2	k=3
Top	**.841±.019**	**.793±.024**	**.738±.023**	**.822±.028**	**.754±.029**	**.687±.038**
Random	.872±.017	.854±.015	.829±.022	.859±.020	.833±.027	.803±.025
Bottom	.884±.013	.876±.014	.868±.018	.878±.016	.866±.022	.849±.019

windows, then removing those windows from the pool should degrade clustering quality more than removing random or low-scored windows. We mask (replace with zero vectors) the top-k%, random-k%, or bottom-k% of pool windows ranked by β_j, recompute SVD on the masked pool, and re-run K-means. We evaluate $k \in \{10, 20, 30\}$% across all five datasets, averaged over 20 runs.

Channel faithfulness (input-channel mask-and-predict). We evaluate Γ_c with a structurally identical test along the channel axis. For each test series we rank channels by $\Gamma_{c(i)}$ where $c(i)$ is its predicted cluster, mean-impute the top-k, random-k, or bottom-k channels at the input, and re-run the frozen encoder → pool → SVD → K-means pipeline. We use $k \in \{1, 2, 3\}$ (absolute counts, since M varies per dataset), averaged over five seeds per dataset. Although Γ_c is computed at the first conv layer, masking is applied at the input and NMI is read from the full frozen pipeline, so the test validates Γ_c end-to-end.

Table 4 presents both tests. For attention, masking the top-10% most important windows degrades NMI by 4.6 points (.887 → .841) vs. 1.5 for random and 0.3 for bottom; at k=30% the gap widens to 14.9 vs. 5.8 vs. 1.9. For channels, removing the highest-Γ_c channel alone degrades NMI by 6.5 points (.887 → .822) while removing the lowest changes NMI by only 0.9; at k=3 the gap widens to 20.0 vs. 8.4 vs. 3.8. Both axes show the monotonic ordering top ≫ random ≫ bottom at every masking level, confirming that β_j and Γ_c provide faithful, interpretable indicators of the windows and channels driving VISTA's output rather than arbitrary signals or proxies for signal magnitude. The larger top-mask drops along the channel axis reflect a single channel being a substantially larger fraction of the input on small-M datasets (e.g., 1/6 on Milling). The grouped-ablation results at $k \in \{2, 3\}$ also empirically mitigate the standard limitation of single-channel attributions, namely that two highly correlated channels can share credit at the first conv layer.

Case Study. Figure 2 illustrates what the faithful scores reveal on ZeMA-Cooler. Each column is one cluster, while each row is the rank-1/2/3 channel for that column's cluster ranked by Γ_c (Eq. 10), giving engineers a "when and what" summary per cluster.

Temporal localization patterns. The β_j distributions reveal physically meaningful patterns across the three degradation states: the near-failure cluster concentrates ∼70% of score mass in the mid-cycle cooling transition (sample indices $[1500, 2400]$, i.e., $t \in [15, 24]$ s of the 60 s cycle), the full-efficiency cluster spreads scores uniformly ($\beta \leq 0.06$), and the reduced-efficiency cluster lies between the two—consistent with a gradual efficiency loss.

Channel attribution across clusters. Γ_c shows that different clusters rely on different sensor channels: the near-failure cluster is dominated by TS1 (cooler outlet temperature) and CE (cooling efficiency), while the reduced-efficiency cluster is dominated by PS3 (secondary pressure) and EPS1 (motor power).

As an external, ground-truth-based cross-check, we compute the Fisher discriminant ratio per channel using the dataset's ground-truth cluster labels (used here only for evaluation, never during training or clustering). The Spearman rank correlation between Γ_c and this ground-truth Fisher ranking is $\rho = 0.82$ on ZeMA-Cooler (mean $\rho = 0.78$ across all five datasets), showing that Γ_c—which is computed purely from the trained encoder and requires no labels—recovers the same channels that a principled truth-label-based discriminability statistic identifies as the most class-separating. This channel-specificity lets engineers immediately identify which sensors to monitor for each degradation mode, enabling targeted condition-based maintenance.

4.6 Scalability Analysis

VISTA's pool-based design bounds memory at $P \cdot d_e$ per time series regardless of recording length T. Window encoding costs $O(T \cdot M \cdot W)$. Attention and SVD are computed only every k pool additions, so attention scales as $O(T/k \cdot P^2 \cdot d_e)$ and the randomized rank-R decomposition has leading cost $O(T/k \cdot P \cdot d_e \cdot R)$—both linear in T since P, d_e, k, and R are constants—compared to $O(T^2)$ for full inter-window attention. Several baselines also have linear-time sequence encoders, including DTSS, LSTM-CL, DTCR, and fixed-state MLGSSM inference, but their memory and representation mechanisms differ from VISTA's bounded pool. TimeMAE incurs quadratic inter-patch attention, while DTW has quadratic alignment time under the standard dynamic-programming formulation, making them impractical in the N-CMAPSS length range. In VISTA, variable-length inputs are handled jointly by the top-P pool and SVD: the bounded pool ensures the matrix fed to SVD has at most $P \times d_e$ entries regardless of T, and the low-rank decomposition then produces a fixed $R \cdot d_e$ output dimension.

Table 5(c) shows wall-clock training time. VISTA trains in 1.0 h on TEP (6,000 series) and 0.2 h on Milling (167 series), comparable to the fastest baselines (DTSS, TS2Vec). N-CMAPSS takes 1.8 h despite having only 274 units, reflecting the long variable-length recordings (up to 38,000). ZeMA-Cooler takes 1.7 h despite fewer data points because the subtle 3-class distinction requires ∼2.5× more epochs to converge—a dataset-intrinsic property observed across all baselines. Methods with heavier architectures (TS-TCC, TimeMAE, CDCC) are 2–7× slower; DTW exceeds the 72-hour cutoff on TEP (the largest N^2 workload) and approaches it on N-CMAPSS (68 h) and WindTrip (64 h) due to the $O(L^2)$ cost at L up to 38,000, even with GPU-accelerated soft-DTW. The asymptotic separation ($O(T)$ vs. $O(T^2)$ for full inter-window attention) means VISTA's advantage over quadratic baselines grows with T on any backend.

4.7 Ablation and Sensitivity Analysis

Table 5(a) shows the ablation results. Removing the encoder causes the largest drop (.887→.253). The two contrastive losses (Eq. 6) are complementary, not antagonistic: Win-only already beats TS-only at

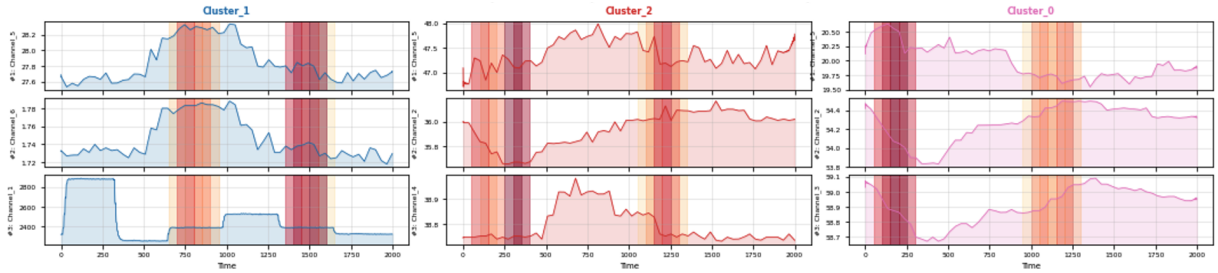


Figure 2: ZeMA interpretability. Columns show clusters and rows their top-three channels by Γ_c (Eq. 10). Each signal is the centroid sample; shading marks the ten highest-β_j windows (dark red to light yellow). Clusters exhibit distinct temporal and channel patterns.

Table 5: (a) Ablation study averaged over all five datasets. The "No intra-pool self-attn" row removes only the intra-pool self-attention layer; eviction attention is kept. (b) Sensitivity to SVD rank R and pool capacity P on ZeMA-Cooler. (c) Wall-clock training time (hours) per dataset; bold marks fastest runtime only, not overall model quality; classical baselines (K-Shape, DTW, MLGSSM) use GPU-accelerated FFT/BLAS implementations; ">72" denotes runs terminated after 72 hours. (d) Automatic cluster number estimation via HDBSCAN: each cell shows the median $\hat{K}$ / $\mathrm{NMI_H}$ over 20 runs; K^* is the ground-truth cluster count, and the best result is in bold.

(a) Ablation

Configuration	Avg NMI	Avg ARI
VISTA (full)	**.887**	**.839**
− No encoder	.253	.164
− Win CL only	.801	.751
− TS CL only	.729	.638
− No FN mitig.	.810	.743
− No intra-pool self-attn	.770	.712
− No SVD (mean)	.781	.708
− No SVD (m+v)	.800	.734
Hard, no STE	.789	.716
Sequential train	.849	.793

(b) Hyperparameter Sensitivity

SVD rank R (P = 40)

R	Embed. dim	NMI	ARI
1	32	0.872	0.825
2	64	0.911	0.878
3	**96**	**0.929**	**0.908**
5	160	0.924	0.893
8	256	0.907	0.869

Pool capacity P (R = 3)

P	Eviction?	NMI	ARI
10	yes	0.863	0.819
20	yes	0.901	0.871
30	yes	0.918	0.893
40	**yes**	**0.929**	**0.908**
60	yes	0.925	0.893
80	yes	0.916	0.881
99 (all)	no	0.892	0.852

(c) Wall-Clock Training Time (hours)

	VISTA	DTSS	TS2Vec	LSTM-CL	DTCR	SoftCLT	TS-TCC	TimeMAE	TimesNet	CDCC	MLGSSM	K-Shape	DTW
TEP	1.0	0.8	0.9	0.6	**0.5**	1.2	2.1	2.5	2.3	4.1	2.9	1.0	>72
WindTrip	1.3	1.1	1.5	0.9	**0.7**	1.8	3.7	4.3	3.5	7.3	4.9	1.7	64
ZeMA	1.7	1.4	1.5	1.0	**0.8**	1.8	2.7	3.0	2.6	4.9	3.6	1.3	6.5
Milling	0.2	0.2	0.2	**0.1**	**0.1**	0.2	0.3	0.4	0.3	0.5	0.4	**0.1**	2.5
N-CMAPSS	1.8	1.5	1.9	1.2	**0.9**	2.3	4.1	4.9	3.9	8.3	5.6	2.1	68

(d) Automatic Cluster Number Estimation via HDBSCAN

Dataset (K^*)	VISTA	CDCC	DTSS	TS-TCC	TS2Vec	TimeMAE	SoftCLT	TimesNet	MLGSSM	DTW	K-Shape
TEP (K^*=12)	**12 / .928**	10 / .847	11 / .866	9 / .784	10 / .816	9 / .793	8 / .738	7 / .629	8 / .677	6 / .571	4 / .284
WindTrip (K^*=9)	**9 / .851**	7 / .654	8 / .739	6 / .572	7 / .608	6 / .563	5 / .504	5 / .538	6 / .559	4 / .432	3 / .203
ZeMA (K^*=3)	**3 / .918**	3 / .821	3 / .834	3 / .769	3 / .781	3 / .758	2 / .697	2 / .672	2 / .608	2 / .273	2 / .061
Milling (K^*=8)	**7 / .786**	6 / .689	6 / .655	5 / .593	5 / .617	5 / .578	4 / .489	4 / .451	4 / .424	3 / .361	2 / .144
N-CMAPSS (K^*=7)	**7 / .862**	5 / .673	6 / .729	4 / .604	5 / .638	4 / .594	3 / .541	3 / .521	4 / .512	2 / .381	2 / .163

the series level (.801 vs. .729), which means that local discrimination is a useful prior rather than a competing objective. Within those losses, removing the false-negative reweighting ($w_{bb''}$ in Eq. 5) costs 8 NMI points, with the largest effect on ZeMA and Milling. Removing intra-pool self-attention (No intra-pool self-attn) drops NMI by 12 points on average (WindTrip: −14, N-CMAPSS: −13); the complementary contribution of learned eviction is isolated in Section 4.8. SVD outperforms mean (.781) and mean+var (.800). Disabling the STE gradient pathway while keeping hard top-P selection ("Hard, no STE") loses 9.8 NMI (.887→.789): without gradients flowing to the scoring head, α_j never learns to identify discriminative windows and hard top-P degenerates toward random selection (refer to Random selection in Table 6, .795). This confirms STE is what makes discrete eviction trainable. Finally, switching to sequential (unshuffled) window ordering during training ("Sequential train") drops NMI by 3.8 points (.887→.849), confirming that global shuffling acts as an effective insertion-order augmentation: positional encodings preserve each window's temporal identity, while shuffled insertion provides more diverse contrastive negatives per batch.

Table 5(b) shows hyperparameter sensitivity. Performance peaks at $R = 3$ (−6 NMI at R=1, noisy at $R \geq 5$, moderate sensitivity with NMI > 0.90 for $R \in [2, 5]$). Notably, all cases outperform mean

Table 6: Eviction faithfulness: NMI under alternative pool composition policies at inference. All components except eviction are identical. The Learned–Anti gap shows that α_j identifies windows relevant to VISTA's output. Results averaged over five datasets and 20 runs.

Eviction Policy	Avg NMI	Avg ARI	ΔNMI
Learned α_j (VISTA)	**.887±.021**	**.839±.024**	—
Reservoir sampling	.808±.029	.756±.033	−.079
Random selection	.795±.031	.733±.035	−.092
FIFO (most recent P)	.781±.033	.714±.038	−.106
Anti-eviction (lowest α_j)	.762±.034	.691±.039	−.125

pooling (Avg NMI 0.781), confirming that singular-value weighting—scaling each principal direction by σ_r—captures discriminative variation structure that the centroid discards with stability across varying values of R. Pool capacity plateaus at $P = 40$, and retaining all windows is worse due to score dilution.

4.8 Eviction and Auto-K Analysis

Eviction Faithfulness Test. The ablation study (Table 5(a)) demonstrates that intra-pool self-attention is essential while keeping eviction attention unchanged. To isolate the contribution of the eviction mechanism specifically, we design an intervention-based faithfulness test for the eviction scores α_j (Eq. 4), analogous to the attention faithfulness test for β_j (Table 4).

We train VISTA with its default configuration, then at inference override the pool composition policy while keeping all other components (encoder, self-attention, SVD, K-means) identical. For each time series, the encoder first produces embeddings for all T_i windows, and eviction scores α_j are computed for every window. We compare five pool-composition policies: (i) **Learned** (top-P by α_j, i.e., normal VISTA inference); (ii) **Anti-eviction** (bottom-P by α_j); (iii) **Random**; (iv) **FIFO** (last-P encountered); and (v) **Reservoir sampling** [34]. Random and Reservoir are averaged over 20 draws. After pool construction, the standard self-attention refinement → SVD compression → K-means pipeline is applied identically. Results are averaged over all five datasets and 20 random seeds.

Learned eviction exceeds every alternative by at least 7.9 NMI, while Anti-eviction is worst, confirming that α_j ranks discriminative windows. The 9.2-point Learned–Random gap isolates learned eviction because both retain the bounded pool and intra-pool self-attention. Combined with the "No intra-pool self-attn" ablation in Table 5(a), the results attribute ∼11.7 NMI to intra-pool self-attention and ∼9.2 to learned eviction. Anti-eviction drops NMI most on N-CMAPSS (15.1 points) and WindTrip (14.8), showing that learned eviction is especially valuable for long, variable-length recordings with sparse discriminative windows.

Automatic Cluster Number Estimation. The experiments above use the ground-truth K, following standard practice. In practice K is unknown, so we evaluate automatic K estimation from two complementary angles on VISTA's embeddings (all medians over 20 runs).

Model-selection criteria. Silhouette score [26] and gap statistic [31], swept over $K \in \{2, \ldots, 2K_{\text{true}}\}$. Silhouette recovers K_{true} exactly on 4 of 5 datasets (TEP 12, ZeMA 3, WindTrip 9, N-CMAPSS 7; Milling 7 vs. true 8), and the gap statistic on 3 of 5 (ZeMA, Milling, N-CMAPSS exact; TEP 11 vs. 12, WindTrip 8 vs. 9).

Density-based discovery without K (baseline comparison). To further test whether the learned representations support K-free cluster discovery, we apply HDBSCAN [4], which infers K from data density, to the per-series representation of each method (Table 5(d)). For embedding-producing methods (all deep baselines and VISTA), HDBSCAN operates on the $N \times d$ embedding matrix with Euclidean distance; for the distance-only baselines DTW and K-Shape, which do not produce embeddings, HDBSCAN operates on the precomputed $N \times N$ DTW / shape-based-distance matrix (metric='precomputed'). All HDBSCAN hyperparameters (min_cluster_size, min_samples) are swept on the validation split and selected by the highest internal density-based cluster validity (DBCV) score, identically across all methods.

As shown in Table 5(d), VISTA recovers the correct K on four of the five datasets and underestimates by one on Milling ($\hat{K}$=7 vs. K^*=8), where two similar machining conditions are merged, while it also achieves the highest NMI_H on all five. Baselines (even the strongest DTSS and CDCC) consistently underestimate K on higher-cardinality tasks, indicating less-separated clusters. Together with the silhouette/gap-statistic evidence above, this confirms VISTA's embeddings form clusters compact and well-separated for both model-selection criteria and density-based discovery, practically important for industrial deployment where the fault type number is typically unknown a priori.

5 Conclusion

We presented VISTA, a scalable unsupervised framework for industrial multivariate time series clustering that jointly addresses long and variable-length recordings, localized discriminative signatures, and result interpretability. A convolutional window encoder, an attention-based eviction pool with bounded capacity, and SVD compression together produce fixed-dimensional embeddings with constant memory per series and linear computational cost in sequence length. The model is trained end-to-end using two-level contrastive learning with false-negative mitigation. Across five industrial datasets, VISTA outperforms all twelve baselines, especially suitable on the variable-length benchmarks. A fair-preprocessing comparison confirms the advantage is architectural rather than a data-handling artifact, and three faithfulness tests—on α_j (eviction), β_j (intra-pool attention), and Γ_c (channel attribution)—validate that VISTA's model-native "when and what" diagnostics are interpretable, faithful to its clustering output, label-free, and aligned with domain-expert physical interpretations.

Limitations and Future Work. VISTA requires specifying window size W and step S. Adaptive windowing could improve generalization across sampling rates. Γ_c captures first-layer channel usage. Extending it to fully layer-wise attribution would refine the "what" diagnostic for highly correlated channels. Our experiments evaluate offline clustering of completed event recordings. Although the bounded pool supports incremental processing, real-time deployment requires latency measurements and an online clustering/update protocol, which we leave for future work.

6 GenAI Usage Disclosure

The author used Claude Opus 4.7 solely to check spelling and improve grammar during the writing of this paper. The tool was not used to generate scientific content, analyses, results, code, or data.

References

[1] Alice Agogino and Kai Goebel. 2007. Milling Data Set. NASA Ames Prognostics Data Repository. [Online]. Available: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432* (2013).

[3] Angela Bonifati, Francesco Del Buono, Francesco Guerra, and Davide Tiano. 2022. Time2Feat: Learning Interpretable Representations for Multivariate Time Series Clustering. *Proceedings of the VLDB Endowment* (2022).

[4] Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-Based Clustering Based on Hierarchical Density Estimates. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. 160–172. doi:10.1007/978-3-642-37456-2_14

[5] M. A. Chao, C. Kulkarni, K. Goebel, and O. Fink. 2021. Aircraft Engine Run-to-Failure Dataset Under Real Flight Conditions for Prognostics and Diagnostics. *Data* 6, 1 (2021), 5.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the International Conference on Machine Learning (ICML)*. 1597–1607.

[7] Mingjin Cheng, Xiaoyu Tao, Zhiwen Liu, Qingsong Liu, Huan Zhang, Renhe Zhang, and Enhong Chen. 2024. TimeMAE: Self-Supervised Representations of Time Series with Decoupled Masked Autoencoders. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*. 498–508.

[8] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. 2020. Debiased Contrastive Learning. In *Advances in Neural Information Processing Systems*, Vol. 33. 8765–8775.

[9] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 7 (2006), 1–30.

[10] Jay DeYoung, Sarthak Jain, Nazneen F. Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A Benchmark to Evaluate Rationalized NLP Models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. 4443–4458.

[11] James J. Downs and Ernest F. Vogel. 1993. A Plant-Wide Industrial Process Control Problem. *Computers & Chemical Engineering* 17, 3 (1993), 245–255.

[12] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. 2021. Time-Series Representation Learning via Temporal and Contextual Contrasting. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2352–2359.

[13] N. Helwig, E. Pignanelli, and A. Schütze. 2015. Condition Monitoring of a Complex Hydraulic System Using Multivariate Statistics. In *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. Pisa, Italy, 210–215. doi:10.1109/I2MTC.2015.7151267

[14] H. Huang, T. Shah, and S. Yoo. 2022. Deep Time Series Sketching and Its Application on Industrial Time Series Clustering. In *Proceedings of the IEEE International Conference on Big Data*. 1997–2006.

[15] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. 2015. Matrix Backpropagation for Deep Networks with Structured Layers. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2965–2973.

[16] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. 2001. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems* 3, 3 (2001), 263–286.

[17] Andrew Kusiak and Anoop Verma. 2013. Monitoring Wind Farms with Performance Curves. *IEEE Transactions on Sustainable Energy* 4, 1 (2013), 192–199.

[18] S. Lee, T. Park, and K. Lee. 2024. Soft Contrastive Learning for Time Series. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[19] Xiucheng Li, Kai Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. 2018. Deep Representation Learning for Trajectory Similarity Computation. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 617–628.

[20] Edo Liberty. 2013. Simple and Deterministic Matrix Sketching. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 581–588.

[21] Qianli Ma, Jiandong Zheng, Sen Li, and Garrison W. Cottrell. 2019. Learning Representations for Time Series Clustering. In *Advances in Neural Information Processing Systems*, Vol. 32. 3781–3791.

[22] N. S. Madiraju, S. M. Sadat, D. Fisher, and H. Karimabadi. 2018. Deep Temporal Clustering: Fully Unsupervised Learning of Time-Domain Features. *arXiv preprint arXiv:1802.01059* (2018).

[23] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series Is Worth 64 Words: Long-Term Forecasting with Transformers. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[24] John Paparrizos and Luis Gravano. 2015. *k*-Shape: Efficient and Accurate Clustering of Time Series. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1855–1870.

[25] F. Peng, J. Luo, X. Lu, S. Wang, and F. Li. 2024. Cross-Domain Contrastive Learning for Time Series Clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 8921–8929.

[26] Peter J. Rousseeuw. 1987. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.* 20 (1987), 53–65.

[27] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, 1 (1978), 43–49.

[28] L. Schmors, D. Gonschorek, J. N. Böhm, Y. Qiu, N. Zhou, D. Kobak, A. Tolias, F. Sinz, J. Reimer, K. Franke, S. Damrich, and P. Berens. 2025. TRACE: Contrastive Learning for Multi-Trial Time-Series Data in Neuroscience. In *Advances in Neural Information Processing Systems*, Vol. 39.

[29] J. Tautz-Weinert and S. J. Watson. 2017. Using SCADA Data for Wind Turbine Condition Monitoring—A Review. *IET Renewable Power Generation* 11, 4 (2017), 382–394.

[30] Davide Tiano, Angela Bonifati, and Raymond Ng. 2021. FeatTS: Feature-Based Time Series Clustering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

[31] Robert Tibshirani, Guenther Walther, and Trevor Hastie. 2001. Estimating the Number of Clusters in a Data Set via the Gap Statistic. *Journal of the Royal Statistical Society: Series B* 63, 2 (2001), 411–423.

[32] R. Umatani, T. Imai, K. Kawamoto, and S. Kunimasa. 2023. Time Series Clustering with an EM Algorithm for Mixtures of Linear Gaussian State Space Models. *Pattern Recognition* 138 (2023), 109375.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, A. N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30.

[34] Jeffrey S. Vitter. 1985. Random Sampling with a Reservoir. *ACM Trans. Math. Software* 11, 1 (1985), 37–57.

[35] Wei Wang, Zheng Dang, Yuliang Hu, Pascal Fua, and Mathieu Salzmann. 2022. Robust Differentiable SVD. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 9 (2022), 5472–5487.

[36] Xiaozhe Wang, Kate Smith, and Rob J. Hyndman. 2006. Characteristic-Based Clustering for Time Series Data. *Data Mining and Knowledge Discovery* 13, 3 (2006), 335–364.

[37] David P. Woodruff. 2014. Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends in Theoretical Computer Science* 10, 1–2 (2014), 1–157.

[38] Haixu Wu, Tao Hu, Yong Liu, Hang Zhou, Jie Wang, and Mingsheng Long. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[39] Zhihan Yue, Yan Wang, Jianshu Duan, Tiexin Yang, Cheng Huang, Yong Tong, and Bin Xu. 2022. TS2Vec: Towards Universal Representation of Time Series. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 8980–8987.

[40] N. Zhang and S. Sun. 2023. Multiview Unsupervised Shapelet Learning for Multivariate Time Series Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).