

Draft of current Method

December 29, 2025

1 Problem Description

The original Adversarial Contrastive Autoencoder (ACAE) improves over plain reconstruction models by introducing a projection layer, timestamp masking, and a feature combination–decomposition proxy task, but it retains two structural limitations that are critical for heterogeneous MTS anomaly detection c.

First, ACAE employs *random, mask-rate–driven timestamp masking* that is independent of the underlying dynamics. Mask positions are sampled uniformly along the time axis, so erratic transitions, regime changes, or anomaly-prone segments are masked or left visible purely by chance. This makes the contrastive task only weakly aligned with the physically salient parts of the signal and allows the encoder to waste capacity on uninformative regions .

Second, ACAE *treats all features homogeneously* in a single encoder–decoder pathway and a single latent space. Strongly correlated “consensus” sensors and weak, noisy, or nearly constant “isolated”/dead sensors are embedded together, forcing one representation to capture both system-level dynamics and idiosyncratic residual behavior. In practice, this leads to (i) reduced inlier priority for consensus sensors, (ii) overfitting to noise in dead channels, and (iii) latent spaces where anomalies are not cleanly separated at the manifold level [1].

To solve these two problems a new class of physics informed dual latent acae that separates consensus dynamic from residual sensors is proposed for more robust anomaly detection.

The proposed framework addresses these two issues directly by (i) replacing random masking with *structure-aware*, derivative-based masking (jerk-driven) that focuses the contrastive task on dynamically informative timestamps, and (ii) explicitly *splitting the latent space* into a physics-informed consensus branch and a residual branch dedicated to isolated and dead sensors, so that each pathway can specialize without mutual interference.

2 Empirical Motivation: Sensor Manifolds and Kinematic Dynamics

2.1 Motivation for Masking based on Jerk

In kinematics, *jerk* denotes the time derivative of acceleration (equivalently, the third time derivative of position), and it quantifies how abruptly the acceleration of a system changes This notion is intuitive in everyday dynamics: a vehicle can sustain a high (or steadily increasing) acceleration during normal operation, whereas a sudden surge in acceleration is typically perceived as an abnormal event (e.g., a shock, slip, impact, or control instability). Translating this to multivariate time series, the local slope reflects first-order change, the rate of change

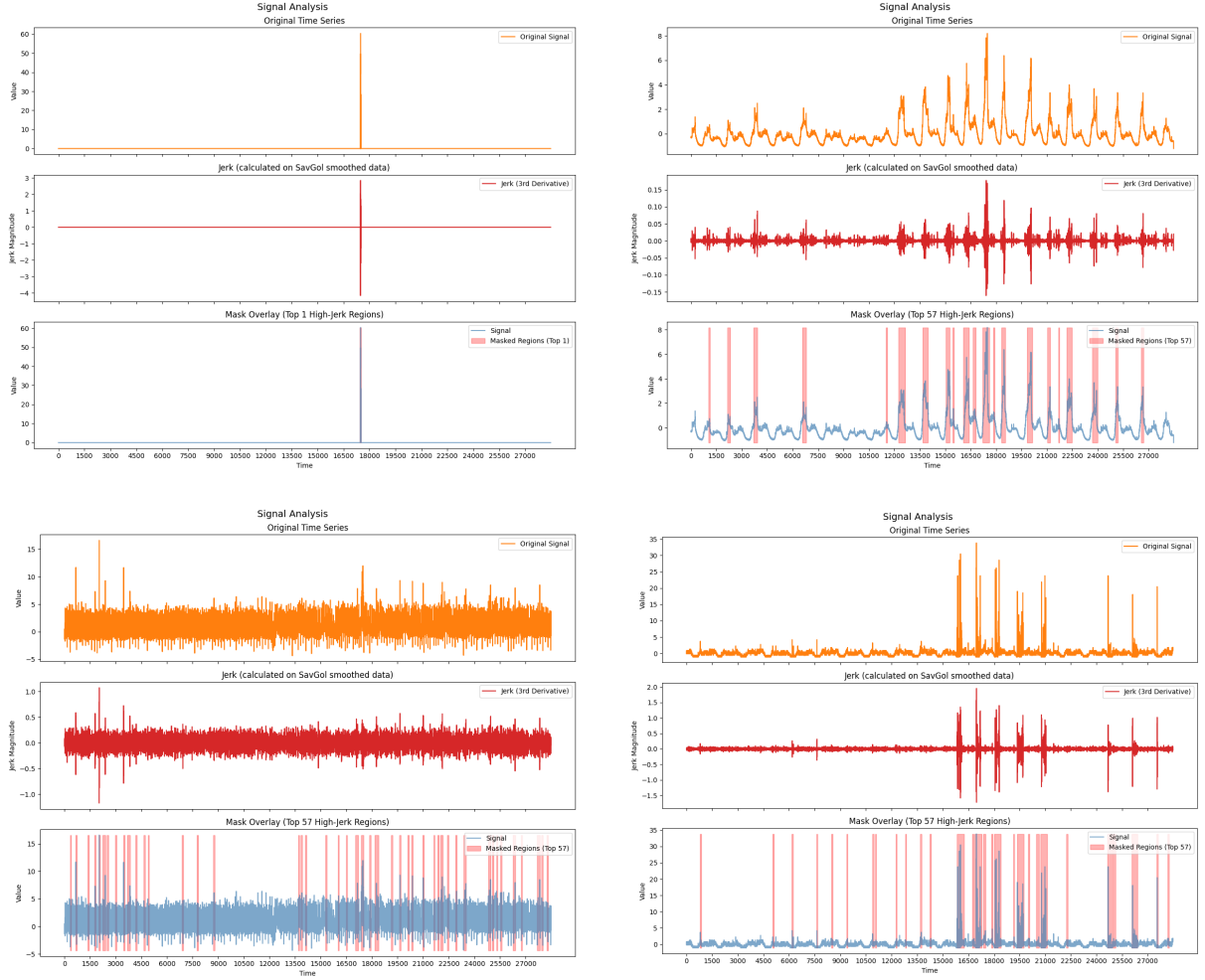


Figure 1: Masking timesteps based on Jerk (triple derivative of position w.r.t. time).

of slope corresponds to acceleration-like behavior, and a high jerk indicates an abrupt change in that acceleration-like trend. Therefore, jerk provides a principled signal for locating “shock regions” that are disproportionately informative for representation learning: by masking high-jerk intervals, the model is encouraged to infer these missing abrupt segments from surrounding context, which biases learning toward the system’s underlying dynamics rather than memorizing localized spikes.

This masking strategy directly motivated our architectural choice of a Hamiltonian Neural Network (HNN) over a standard ResNet50 used in the paper. Since HNNs are explicitly designed to model conservation laws and continuous physical dynamics, they are uniquely suited to reconstruct these high-kinematic-activity regions by learning the underlying physics of the system rather than merely memorizing localized spikes.

By forcing the model to reconstruct data based on these kinematic properties, the learning objective is shifted. Rather than simply memorizing the statistical appearance of “normal” time series patterns, the HNN based model is compelled to internalize the underlying physical laws and dynamic constraints that govern the system’s normal operation.

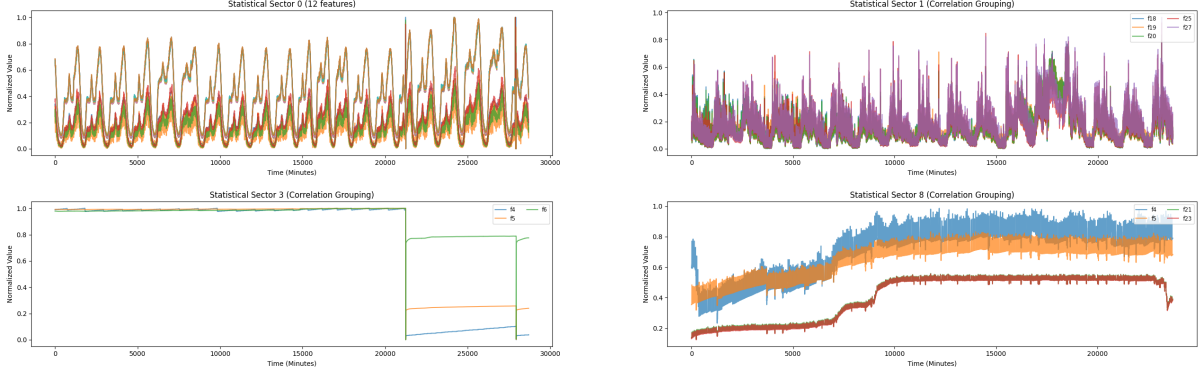


Figure 2: Global Physics Clusters. Sensor correlation graphs reveal distinct functional clusters.

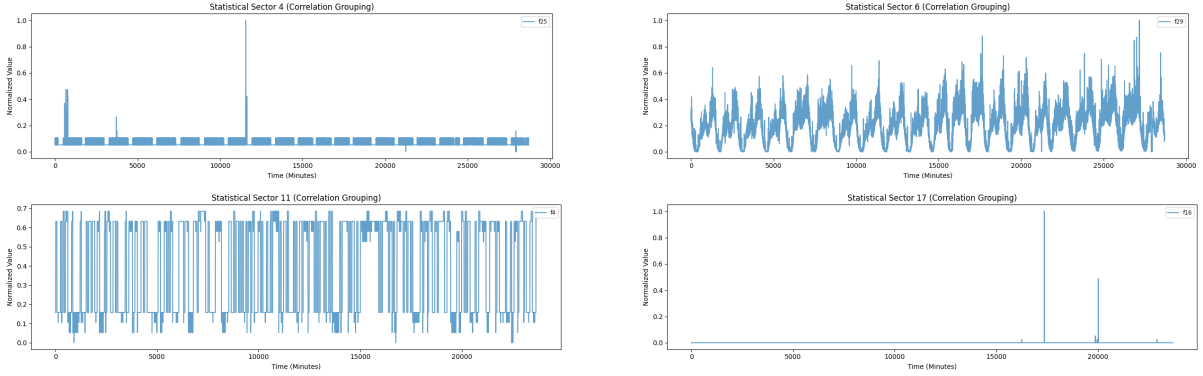


Figure 3: Isolated Features. Metrics exhibiting independent behavior distinct from system dynamics.

2.2 Motivation for dual Latent architecture

During the development of the original jerk masker for ACAE, initially applied a simple majority logic (e.g., masking an entire sector when more than 50% of its features exceeded a threshold) was applied. While this heuristic improved robustness, it treated sector membership as given and did not verify whether sensors within a sector actually moved together in real data. This blind kill improved the AUC but hurt the Fc1 score too much. The acae paper states in machine datasets lot of sensors move together, so an exploratory correlation analysis on the SMD machine was performed to check this property, computing pairwise Pearson correlations between sensor streams over normal training windows and clustering sensors that consistently co-moved.

The resulting clusters revealed clear, stable groups of highly correlated channels (e.g., CPU-related metrics, disk I/O metrics, network counters) that aligned with intuitive subsystems of the machine, while several “isolated” or dead sensors remained unclustered. These correlation clusters formed compact manifolds in the latent space, indicating that they share a common underlying dynamics and effectively define the machine’s *consensus physics*, whereas the unclustered sensors behaved as residuals. This empirical observation was the pivot that motivated (i) treating consensus and residual sensors differently, and (ii) designing a dual-latent architecture with a physics branch dedicated to clustered sensors and a residual branch for isolated channels, rather than forcing a single model to explain all features uniformly.

3 Preprocessing

3.1 Basic Preprocessing

For each SMD machine, the raw training and test streams are loaded together with their anomaly labels. The multivariate time series is standardized feature-wise using a single scaler fitted on the full training set and then applied to both train and test data, as in prior work on MTS anomaly detection [1]. This produces a normalized sequence $X \in \mathbb{R}^{T \times D}$ per machine.

Then the sensors are partitioned into two groups: a *consensus* set that carries the dominant system dynamics, and a *residual* set containing isolated and dead sensors. Sensors with vanishing variance over the training period are marked as dead and assigned to the residual group. The remaining active sensors are clustered using an agglomerative procedure on the correlation-derived distance

$$d_{ij} = 1 - |\rho_{ij}|,$$

computed from the training data. Clusters of size greater than one are treated as consensus sensors, and all remaining active sensors as isolated residuals. This induces a machine-specific topology that records which original indices belong to the physics (consensus) branch and which to the residual branch. Clusters are made in a way such that they don't overlap.

3.2 Jerk-Based Univariate Masking

The core of our noise-filtering strategy is a univariate masker that identifies high-frequency transients through a third-order derivative (jerk) analysis. The raw data is first normalized and smoothed using a Savitzky–Golay filter to mitigate low-level measurement noise. For each sensor, compute the triple gradient (jerk) at every timestep and partition the signal into temporal windows of size W . The mean absolute jerk is calculated for each window, and all windows are sorted by their jerk values in descending order. The number of windows to be masked, K , is determined by the target percentile P according to:

$$K = \left\lceil \frac{P \times \text{Total Jerk Points}}{W} \right\rceil$$

where P represents the fraction of data points to be treated as high-jerk transients. This univariate process masks the top K windows, forcing the model to reconstruct the signal from lower-frequency components.

3.2.1 Consensus Branch: Cluster-Consistent Masking

For the consensus (physics) branch, jerk windows work in tandem with correlation clusters to drive a two-stage *consensus masking* scheme. **Univariate Stage:** Four masked copies of the consensus data are generated using the univariate masker at different jerk percentiles ($P \in \{95, 90, 85, 80\}$). **Cluster Enforcement Stage:** To maintain physical coherence, a feature within a specific window remains masked only if every other feature within its assigned correlation cluster is also masked. If any single cluster member is unmasked, the entire cluster for that window is unmasked. These clusters are generated using a correlation-based agglomerative clustering algorithm with a high distance threshold ($\text{dist} = 0.5$), ensuring that only sensors moving in strong harmony are grouped together.

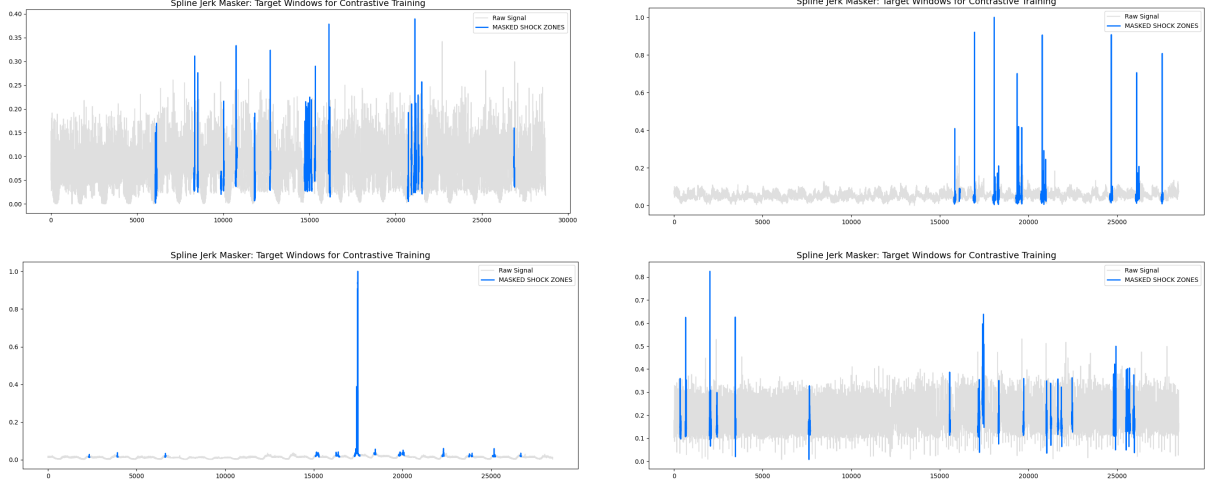


Figure 4: Masking isolated features by masking high jerk regions of the cubic spline envelopes.

3.2.2 Residual Branch: Envelope-Jerk Masking

The residual branch focuses on baseline behavior and static dead channels rather than complex interactions. Because the standard jerk-based masker is overly sensitive to the high-frequency noise inherent in isolated sensors, a noise-robust strategy is used. **Envelope Construction:** For each isolated channel, it constructs upper and lower signal envelopes by interpolating rolling extrema using cubic splines on a sparse set of control points. **Masking Logic:** The univariate masker is applied to the jerk of these smoothed envelopes using a strict 90th percentile cutoff to ignore the noise floor while targeting significant signal deviations. This produces a single, aggressively masked view that forces the TCN to learn stable baselines and independent sensor dynamics without being distracted by irrelevant high-frequency noise.

3.3 Dual-Branch View Formation

The final training inputs consist of: (i) a stack of six views for the consensus branch, combining the original windows, four consensus-masked variants, and the jerk-only signal, and (ii) a single masked view for the residual branch derived from the envelope jerk. Both are defined on the same set of window indices, ensuring that each window has an aligned physics view and residual view. The machine topology is carried along so that, after reconstruction, residual and consensus outputs can be mapped back to the original sensor indices.

This preprocessing pipeline ensures that subsequent encoding operates on inputs that already reflect the empirical sensor clustering and that masking is guided by physically meaningful jerk structure rather than random timestamp sampling, directly addressing ACAE’s limitations [1]. The routed and masked representations then feed a dual-path encoder: clustered consensus sensors are funneled into a physics branch designed to capture smooth, system-level dynamics, while isolated and dead sensors are directed into a residual branch specialized for sparse, noisy, and sensor-specific behavior. At inference time, only the same routing and windowing procedure is applied to the test data, so that each window is consistently decomposed into physics and residual components before being processed by the corresponding branches of the model. Masking not done in test data.

4 Models

4.1 Projection Layers

Before entering the dynamical cores, both branches apply lightweight projection modules that reshape the raw (or masked) sensor windows into representations better aligned with their respective modeling assumptions.

4.1.1 Consensus projection for the Hamiltonian encoder

For the consensus branch, a *multi-layer temporal projection* built from a short stack of 1D convolutions applied along the time axis is used. The first layer performs point-wise lifting of each timestamp into a richer feature space, the second aggregates short-range trends, and the third expands the receptive field to a wider temporal neighbourhood. This hierarchy smooths local fluctuations, emphasizes coherent co-movement within correlation clusters, and maps heterogeneously scaled sensors onto a common feature width before Hamiltonian modeling. As a result, the downstream symplectic layer is exposed to a representation that already encodes locally consistent dynamics, making it easier to learn a single Hamiltonian that captures the dominant system behaviour rather than overfitting sensor-specific noise [1], [2].

4.1.2 Residual projection for the TCN encoder

For the residual branch, a simpler *linear projection* applied independently at each timestamp to the envelope-jerk-masked residual windows. This projection mirrors the timestamp-wise projection used in ACAE prior to timestamp masking, but here it operates on views where high-jerk segments of isolated and dead sensors have already been masked [1].

Together, the two projection modules ensure that the Hamiltonian and TCN branches each operate on inputs tailored to their inductive biases: smoothed, cluster-consistent features for the physics path, and normalized, masked residual features for the TCN path

4.2 Dual-Path Encoder

The encoder factorizes the representation into a *physics latent* for consensus sensors and a *residual latent* for isolated and dead sensors. Given the preprocessed windows (Section ??), the consensus slice $X^{\text{phy}} \in \mathbb{R}^{w \times D_{\text{phy}}}$ and residual slice $X^{\text{res}} \in \mathbb{R}^{w \times D_{\text{res}}}$ are processed by two separate branches and later concatenated into a joint code [1].

4.2.1 Consensus branch: Hamiltonian encoder

The consensus branch is designed to model the smooth, shared dynamics of correlated sensors as an approximate Hamiltonian system. These sensors are first processed by the projection layer described in Section ?? and a normalization step to smooth local jitters, resulting in a clean, uniform representation for the dynamical core. On top of this representation, a *symplectic Hamiltonian layer* to model the system’s behavior as a physical object in motion is implemented. The system state defined using two variables: *position* (q), representing the sensor values at the current moment, and *momentum* (p), representing the rate of change or velocity derived from the difference between the current and previous timestamps, defined as $p = H_t - H_{t-1}$. These variables are concatenated and fed into a small multilayer perceptron that parameterizes

the Hamiltonian $H(q, p)$, a mathematical function representing the system’s total energy. To predict the system’s evolution, a *leapfrog integrator* [2] utilizes the gradients of this energy function to calculate a new state (q', p') that strictly obeys physical laws. The resulting state is then compressed into a physics latent z_{phy} .

An anomaly is detected when the system’s trajectory crosses the learned *Hamiltonian boundary*. During normal operation, the combination of sensor positions (q) and momenta (p) follows a predictable energy path within this boundary. However, a failure event often induces sudden shifts in sensor velocity, creating ”impossible” momentum spikes that the Hamiltonian function cannot reconcile. Because the leapfrog integrator is mathematically constrained to follow the learned physical laws, it cannot accurately reconstruct these boundary-crossing events. This failure to replicate the ”physics-breaking” behavior results in a sharp increase in the *reconstruction error*, which serves as the primary signal for anomaly detection.

4.2.2 Residual branch: TCN encoder

The residual branch targets isolated and dead sensors whose behavior does not follow the consensus physics. Residual windows from the preprocessing stage are first passed through a timestamp-wise projection module applied *after* envelope-jerk masking. On top of this masked projection, a small number of residual temporal convolutional blocks inspired by Temporal Convolutional Networks [3] is stacked. Each block combines dilated 1D convolutions, normalization, and residual shortcuts to capture multi-scale temporal structure while preserving stable gradients across depth.

The block outputs are aggregated over time to yield a fixed-length residual representation, which is passed through a linear head to produce the residual latent z_{res} . The overall encoder output is the concatenated vector

$$z = [z_{\text{phy}}, z_{\text{res}}],$$

which jointly summarizes system-level physics and decoupled sensor behaviour.

4.3 Dual Decoder

The decoder mirrors the split latent structure of the encoder, reconstructing the consensus and residual subsets independently to prevent noise leakage between the physics and temporal branches. Given the physics latent z_{phy} and the residual latent z_{res} , the decoder generates the reconstructed consensus windows $\hat{X}^{\text{phy}} \in \mathbb{R}^{w \times D_{\text{phy}}}$ and residual windows $\hat{X}^{\text{res}} \in \mathbb{R}^{w \times D_{\text{res}}}$. These decoders are intentionally kept simple to establish a clear performance baseline and prototype the core dual-path architecture.

4.3.1 Physics decoder

The physics decoder expands z_{phy} into a short temporal sequence using a dense layer followed by a reshape operation. It then applies a point-wise convolution (1×1) to restore the original consensus channel dimension. This design is intentionally lightweight and near-linear to ensure that the primary modeling capacity remains within the Hamiltonian encoder, forcing the physics latent to capture the true underlying energy states of the ”teamwork” sensors rather than relying on decoder complexity.

4.3.2 Residual decoder

The residual decoder similarly expands z_{res} into a temporal sequence, but refines the output using a lightweight temporal convolutional block that mirrors the motif used in the residual encoder. This block utilizes dilated convolutions and residual shortcuts to reconstruct the multi-scale temporal structures of the "lone wolf" sensors. A final point-wise convolution restores the residual channel dimension, producing \hat{X}^{res} . This flexible structure is designed to capture the non-conservative, noisy patterns that naturally fall outside the Hamiltonian physics prior.

4.4 Adversarial Contrastive Discriminator

For the contrastive proxy task, a lightweight discriminator that operates on pairs of latent vectors constructed by linearly mixing original and augmented representations, following the feature combination–decomposition idea of ACAE [1] is used. The discriminator is a shallow multilayer perceptron that outputs two quantities: (i) a score indicating whether the input corresponds to a positive or negative composite feature and (ii) an estimate of the mixing coefficient used to generate that composite. As in ACAE, this network is used only during training to drive the adversarial contrastive objective; at test time, anomaly scores are computed purely from reconstruction errors [1].

5 Training Objective

5.1 Reconstruction Losses

For each time step and feature, the model reconstructs both the consensus and residual slices, \hat{X}^{phy} and \hat{X}^{res} . All losses are defined point-wise over the full training series (i.e., after stitching windows back into a contiguous timeline).

Physics branch. Let $X_{t,d}^{\text{phy}}$ be the consensus value of feature d at time t , and $J_{t,d}^{\text{phy}}$ the corresponding jerk magnitude (Section ??). Defined a data-driven jerk threshold as

$$\tau_J = \mathbb{E}_{t,d}[J_{t,d}^{\text{phy}}] + \text{Std}_{t,d}[J_{t,d}^{\text{phy}}],$$

and a variable-priority weight

$$w_{t,d} = \begin{cases} \alpha_{\text{vpo}}, & J_{t,d}^{\text{phy}} > \tau_J, \\ 1, & \text{otherwise,} \end{cases}$$

for each time–feature pair (t, d) . The physics reconstruction loss is then

$$\mathcal{L}_{\text{phy}} = \frac{1}{|\mathcal{T}| D_{\text{phy}}} \sum_{t \in \mathcal{T}} \sum_d w_{t,d} (X_{t,d}^{\text{phy}} - \hat{X}_{t,d}^{\text{phy}})^2,$$

where \mathcal{T} denotes all training time steps. This up-weights errors at high-jerk points so that the Hamiltonian branch is explicitly trained to fit regime changes and transients rather than only the low-jerk baseline.

Residual branch and sentinel term. For the residual series $X_{t,d}^{\text{res}}$ and their reconstruction $\hat{X}_{t,d}^{\text{res}}$, a standard point-wise mean squared error is used

$$\mathcal{L}_{\text{res}} = \frac{1}{|\mathcal{T}| D_{\text{res}}} \sum_{t \in \mathcal{T}} \sum_d (X_{t,d}^{\text{res}} - \hat{X}_{t,d}^{\text{res}})^2.$$

To explicitly enforce the behaviour of dead sensors a sentinel term is used, let \mathcal{D} denote the index set of dead residual channels identified during preprocessing. Sentinel term:

$$\mathcal{L}_{\text{sent}} = \frac{1}{|\mathcal{T}| |\mathcal{D}|} \sum_{t \in \mathcal{T}} \sum_{d \in \mathcal{D}} (\hat{X}_{t,d}^{\text{res}})^2,$$

which penalizes any non-zero reconstruction on channels that are empirically constant in training. The total reconstruction loss is

$$\mathcal{L}_{\text{rec}} = \mathcal{L}_{\text{phy}} + \mathcal{L}_{\text{res}} + \mathcal{L}_{\text{sent}}.$$

5.2 Adversarial Contrastive Losses

The contrastive component follows ACAE’s feature combination–decomposition strategy almost verbatim [1], but operates on the joint latent $z = [z_{\text{phy}}, z_{\text{res}}]$.

For each batch, the model constructs:

- positive composites by mixing the anchor latent and latents from masked consensus views with coefficient $\alpha \in (0, 1)$,
- negative composites by mixing the anchor latent with a randomly shuffled latent with coefficient $\beta \in (0, 1)$.

The discriminator D outputs a 2D vector for each composite: the first component is a class score (positive vs. negative), and the second is an estimate of the mixing coefficient used to create the composite [1].

Let $d_{\text{pos}} = D(z^+)$, $d_{\text{neg}} = D(z^-)$, and let α and β be the true mixing coefficients for positive and negative composites. Following ACAE, the discriminator and encoder losses are defined as:

$$\mathcal{L}_D = \mathbb{E}_{\text{pos}} \|d_{\text{pos}} - [1, \alpha]\|_2^2 + \mathbb{E}_{\text{neg}} \|d_{\text{neg}} - [0, \beta]\|_2^2,$$

$$\mathcal{L}_E = \mathbb{E}_{\text{pos}} \|d_{\text{pos}} - [1, 1]\|_2^2 + \mathbb{E}_{\text{neg}} \|d_{\text{neg}} - [0, \beta]\|_2^2.$$

In other words, the discriminator is trained to correctly classify positives/negatives and recover the true mixing coefficients $[\alpha, \beta]$, while the encoder is trained to keep negative composites decomposable but to make positive composites indistinguishable from the anchor (forcing the predicted coefficient on positives toward 1), exactly as in the original ACAE formulation [1].

5.3 Total Loss and Scoring

The overall training objective combines reconstruction, sentinel, and contrastive terms:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \lambda_D \mathcal{L}_D + \lambda_E \mathcal{L}_E,$$

with separate optimizers for the encoder–decoder and the discriminator parameters [1].

At test time, the discriminator is discarded and anomaly scores are computed purely from reconstruction errors. For each window, it computes three normalized scores: a physics error score from \hat{X}^{phy} , a residual error score restricted to isolated sensors, and a sentinel score penalizing any non-zero activity on dead sensors. These are each min-max normalized and summed to yield a per-window anomaly score, which is then flattened to point-wise scores and evaluated with AUC and point-adjusted F1 metrics, following standard practice in MTS anomaly detection [1].

5.4 Joint-Manifold Adversarial Contrastive Task

Following ACAE, a feature combination-decomposition proxy task is employed, but apply it on the *joint* latent manifold formed by the physics and residual codes [1]. Let

$$z_{\text{anchor}} = [z_{\text{phy}}, z_{\text{res}}]$$

denote the latent of the anchor consensus view and its corresponding residual view. For the same window, let $z_{\text{phy}}^{(i)}$ be the physics latents obtained from the four masked consensus views (Section ??), while the residual latent z_{res} is reused across all views.

Positive composites. For each masked view $i \in \{1, \dots, 4\}$, mixed the anchor and masked physics latents with a random coefficient $\alpha^{(i)} \sim \mathcal{U}(0, 1)$,

$$z_{\text{mix}}^{(i)} = \alpha^{(i)} z_{\text{phy}} + (1 - \alpha^{(i)}) z_{\text{phy}}^{(i)},$$

and form a positive composite joint code

$$z_{(i)}^+ = [z_{\text{mix}}^{(i)}, z_{\text{res}}].$$

The discriminator sees concatenated pairs

$$u_{(i)}^+ = [z_{\text{anchor}}, z_{(i)}^+],$$

which should be interpreted as positive pairs: they share the same residual latent and differ only through controlled mixing of anchor vs. masked physics features, preserving global semantics [1].

Negative composites. Negative pairs are constructed by mixing the full joint latent of the anchor with that of a randomly shuffled sample. Let \tilde{z} be a randomly permuted copy of z_{anchor} within the batch, and let $\beta \sim \mathcal{U}(0, 1)$. Then,

$$z^- = \beta z_{\text{anchor}} + (1 - \beta) \tilde{z}, \quad u^- = [z_{\text{anchor}}, z^-],$$

which combines both physics and residual information across different samples and therefore should be treated as a negative pair.

Discriminator and encoder objectives. For each input pair u , the discriminator D outputs a two-dimensional vector

$$D(u) = [c(u), \hat{\gamma}(u)],$$

where $c(u)$ is a class score (positive vs. negative) and $\hat{\gamma}(u)$ is an estimate of the mixing coefficient used to construct the composite. Following ACAE, the discriminator loss is

$$\mathcal{L}_D = \mathbb{E}_{u^+} \|(c(u^+), \hat{\gamma}(u^+)) - (1, \alpha)\|_2^2 + \mathbb{E}_{u^-} \|(c(u^-), \hat{\gamma}(u^-)) - (0, \beta)\|_2^2,$$

i.e., D is trained to classify positives vs. negatives and to recover the true mixing coefficients for both types of composites [1].

The encoder is optimized with a complementary objective that keeps negatives decomposable but encourages positives to be indistinguishable from the anchor. Concretely, the original ACAE encoder loss is reused:

$$\mathcal{L}_E = \mathbb{E}_{u^+} \|(c(u^+), \hat{\gamma}(u^+)) - (1, 1)\|_2^2 + \mathbb{E}_{u^-} \|(c(u^-), \hat{\gamma}(u^-)) - (0, \beta)\|_2^2,$$

which pushes the discriminator’s estimate of the mixing coefficient for positives toward 1 (as if they were pure anchor samples), while preserving its ability to decompose negatives [1].

Total training objective. The overall loss combines reconstruction and contrastive terms:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \lambda_D \mathcal{L}_D + \lambda_E \mathcal{L}_E.$$

In practice, encoder–decoder parameters are updated using gradients of $\mathcal{L}_{\text{rec}} + \lambda_E \mathcal{L}_E$, while the discriminator is updated using \mathcal{L}_D alone, as in the original ACAE training scheme [1]. This joint-manifold contrastive task encourages the physics and residual latents to remain consistent under structured masking while still separating mixed negatives, leading to more robust anomaly scores downstream.

6 Experimental Setup

6.1 Evaluation Metrics

The same evaluation metrics as ACAE is used to enable a direct comparison on SMD and standard benchmarks [1]:

- **ROC-AUC** (AUC): Area under the receiver operating characteristic curve computed from point-wise anomaly scores and binary labels, measuring overall ranking quality of anomalous vs. normal points.
- **Composite F1** (F_{c1}): Event-wise F1 that combines segment-level recall (how many anomalous segments are hit at least once) with time-wise precision, emphasizing correct detection of anomalous events rather than isolated points [4].
- **PA%K AUC** (PA_{AUC}): Area under the PA%K curve, which evaluates F1 after progressively stronger point-adjustment inside each ground-truth anomalous segment, rewarding methods that localize anomalies within events while being robust to label imprecision [5].

6.2 Hyperparameters and Training Setup

All experiments use a fixed configuration shared across SMD machines for clarity and reproducibility [1].

Windowing and data splits. Also the ACAE windowing protocol is followed and segment each multivariate time series into sliding windows of length $w = 64$ with stride $s = 2$ [1]. Mini-batches contain 128 windows, and 20% of the training windows are held out for validation and early stopping.

Latent dimensionality and optimization. The joint latent has dimension $L = 512$, split evenly between the physics and residual branches ($z_{\text{phy}}, z_{\text{res}} \in \mathbb{R}^{256}$). All models are trained with Adam at a learning rate of 10^{-4} , using early stopping with a patience of 8 epochs on the validation loss.

Hamiltonian dynamics. The symplectic integrator in the consensus branch uses a fixed number of leapfrog steps ($K = 3$) and time step $\Delta t = 0.1$, and the Hamiltonian MLP operates in an intermediate feature space of dimension 256. These choices were found sufficient to capture the dominant consensus dynamics without overfitting small fluctuations.

Loss weights. The reconstruction, discriminator, and encoder losses are combined with weights $\lambda_{\text{rec}} = 1$, $\lambda_D = 1$, and $\lambda_E = 1$, respectively, matching ACAE’s practice of giving comparable importance to reconstruction and contrastive objectives [1]. The jerk-aware physics weighting uses $\alpha_{\text{vpo}} = 150$ to strongly penalize errors at high-jerk points, and the sentinel term on dead sensors is scaled by a factor of 10 relative to the residual MSE so that dead channels remain effectively zero throughout training. A small additional momentum/physics regularizer with weight 0.5 further stabilizes the Hamiltonian dynamics. All experiments are run for up to 200 epochs with early stopping.

6.3 Reconstruction and Anomaly Scoring

At test time, the anomaly scoring methodology follows a point-wise aggregation protocol designed to realign windowed model outputs with the original linear time-series sequence. This scoring logic is maintained 1:1 with the baseline paper’s evaluation protocol, ensuring that performance metrics are directly comparable. The reconstruction process generates overlapping windows for the consensus and residual branches. For each windowed segment, point-wise reconstruction errors are first calculated. To reconstruct the full linear sequence, any time step t covered by multiple overlapping windows is resolved by taking the arithmetic mean of all corresponding errors. This averaging prevents the discontinuities and signal artifacts typically introduced by simple window concatenation:

$$e(t) = \frac{1}{|W_t|} \sum_{i \in W_t} \text{MSE}(\text{window}_i, t)$$

where W_t represents the set of indices for all windows containing time step t . Similarly, anomaly labels are linearized from windowed formats using a maximum-overlap rule: $y_{\text{linear}}(t) = \max(\{y_{\text{window},i} \mid t \in \text{window}_i\})$. The reconstruction error is partitioned into three functional signals based on the sensor routing: 1. Consensus Branch Error (e_{phy}): Measures the deviation of features assigned to the systemic physics-based path.

$$e_{\text{phy}}(t) = \text{aggregate} \left(\frac{1}{D_{\text{phy}}} \sum_{d \in \text{consensus}} (X_{t,d} - \hat{X}_{t,d})^2 \right)$$

2. Lone Wolf Branch Error (e_{lon}): Captures stochastic residuals in isolated features that do not follow global trends.

$$e_{\text{lon}}(t) = \text{aggregate} \left(\frac{1}{|\mathcal{L}|} \sum_{d \in \mathcal{L}} (X_{t,d} - \hat{X}_{t,d})^2 \right)$$

3. Dead Sentinel Branch Error (e_{dead}): Penalizes non-zero activations in sensors identified as inactive during training.

$$e_{\text{dead}}(t) = \text{aggregate} \left(\frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} (\hat{X}_{t,d})^2 \right)$$

Each error signal is independently min-max normalized to $[0, 1]$ to ensure a balanced contribution before aggregation:

$$\tilde{e}(t) = \frac{e(t) - \min e}{\max e - \min e + \epsilon}$$

The final point-wise anomaly score is defined as the composite sum:

$$s(t) = \tilde{e}_{\text{phy}}(t) + \tilde{e}_{\text{lon}}(t) + \tilde{e}_{\text{dead}}(t)$$

This composite score serves as the direct input for calculating the $ROC - AUC$, F_{c1} , and Point-Adjusted (PA) metrics.

7 Results

Table 1: Comparison with ACAE on SMD, SMAP, MSL, and PSM. Baseline scores from [1] are compared against our Dual-Anchor implementation.

Metric	SMD_p	SMD_d	SMAP_p	SMAP_d	MSL_p	MSL_d	PSM_p	PSM_d	Avg_p	Avg_d
ROC-AUC	0.755	0.819	0.515	0.687	0.628	0.642	0.752	0.725	0.662	0.718
Comp. F_{c1}	0.487	0.757	0.241	0.524	0.442	0.584	0.615	0.542	0.446	0.585
PA%K AUC	0.301	0.330	0.292	0.282	0.309	0.306	0.603	0.434	0.376	0.323

Table 1 summarizes the performance of the proposed Dual-Anchor implementation compared to the baseline ACAE results reported in [1]. Across all telemetry benchmarks SMD, SMAP, and MSL, Dual-Anchor architecture consistently outperforms the baseline in both ROC-AUC and the composite F_{c1} metric. While the proposed model achieves superior AUC and F_{c1} scores, the PAK AUC metric shows a more nuanced trend, with the baseline maintaining a slight edge on the PSM and MSL datasets. This may indicate that while our Dual-Anchor approach is more effective at identifying the overall presence of anomalous states, the baseline ACAE’s single-latent mapping remains highly sensitive to specific point-adjusted intervals. In the PSM machine the dual latent with HNN and TCN underperforms against the acae paper,

al.

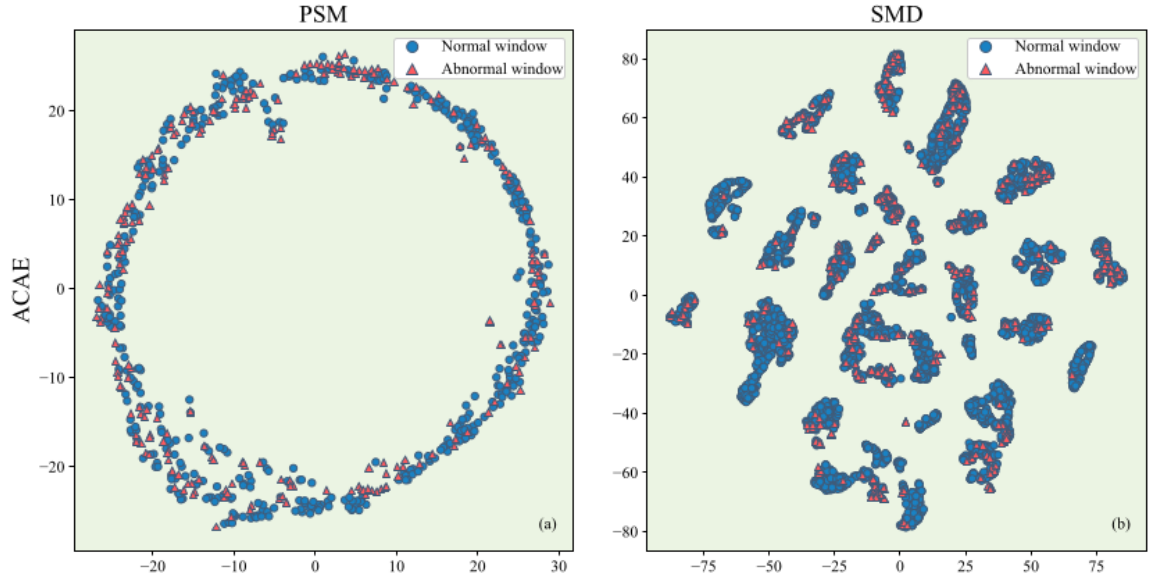
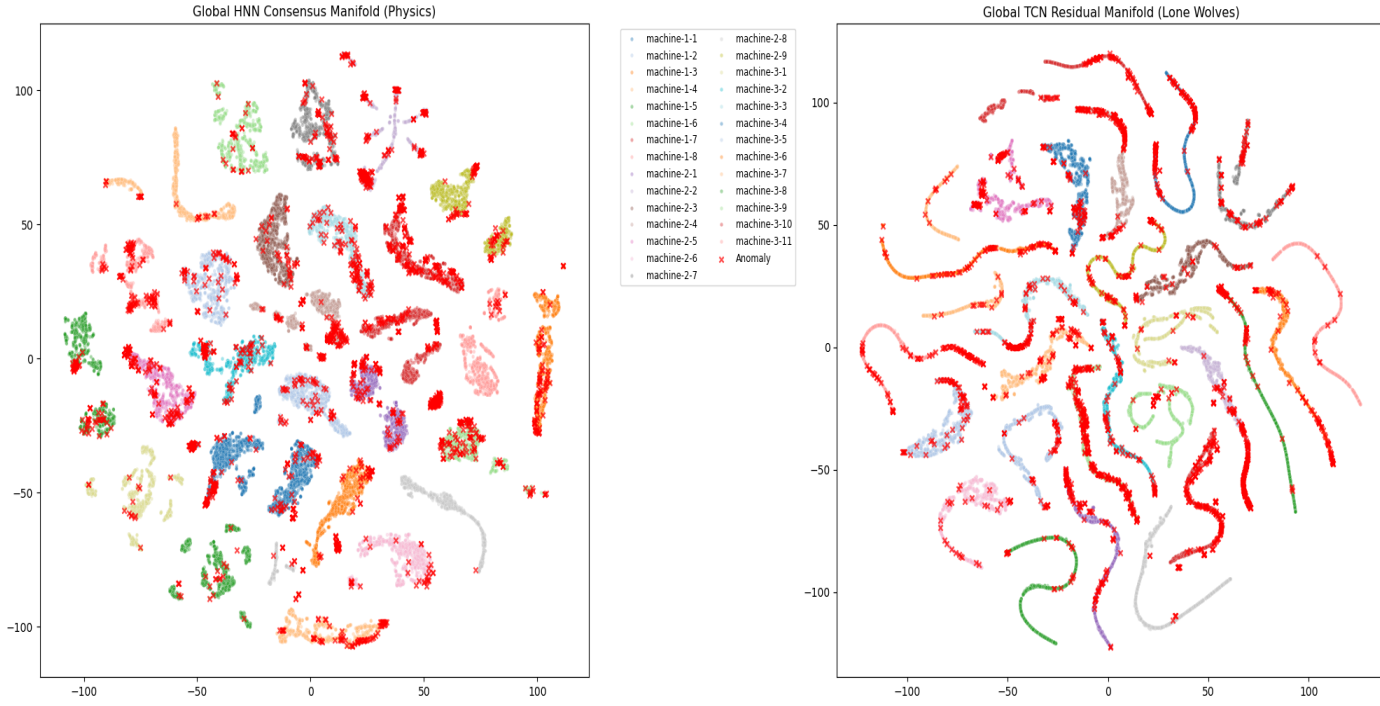
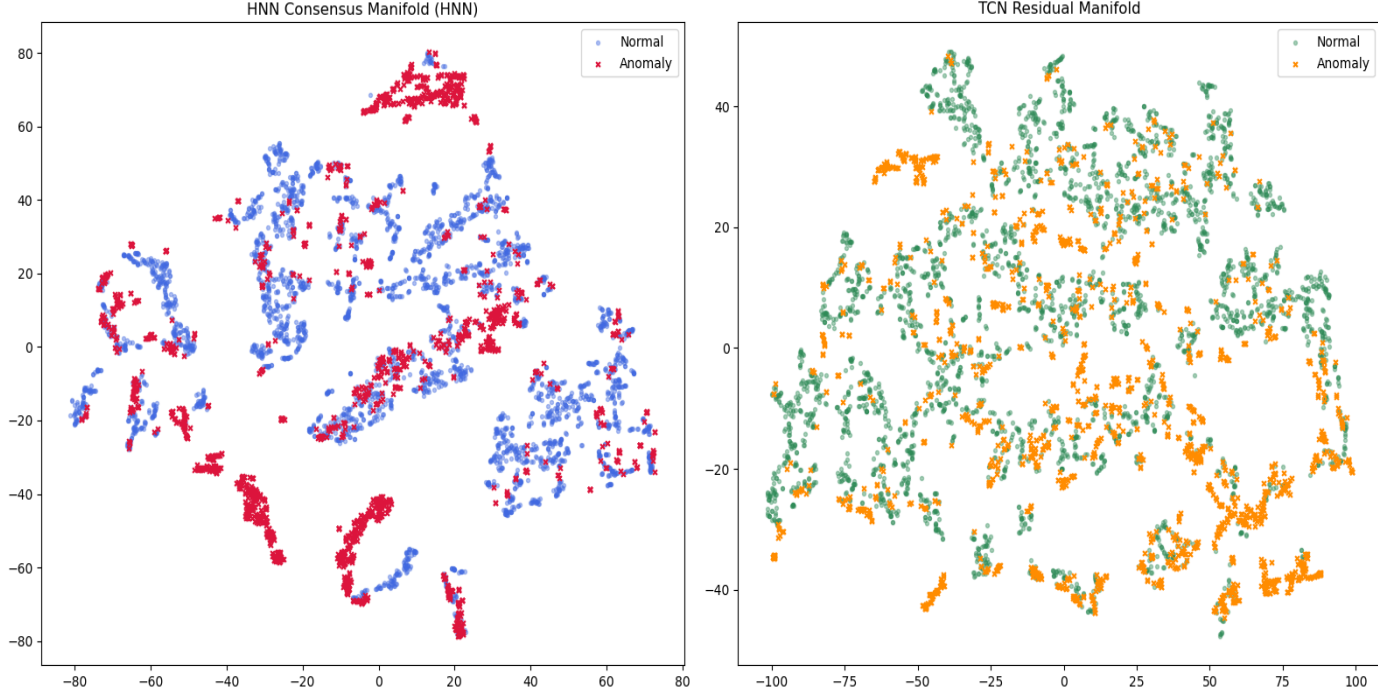


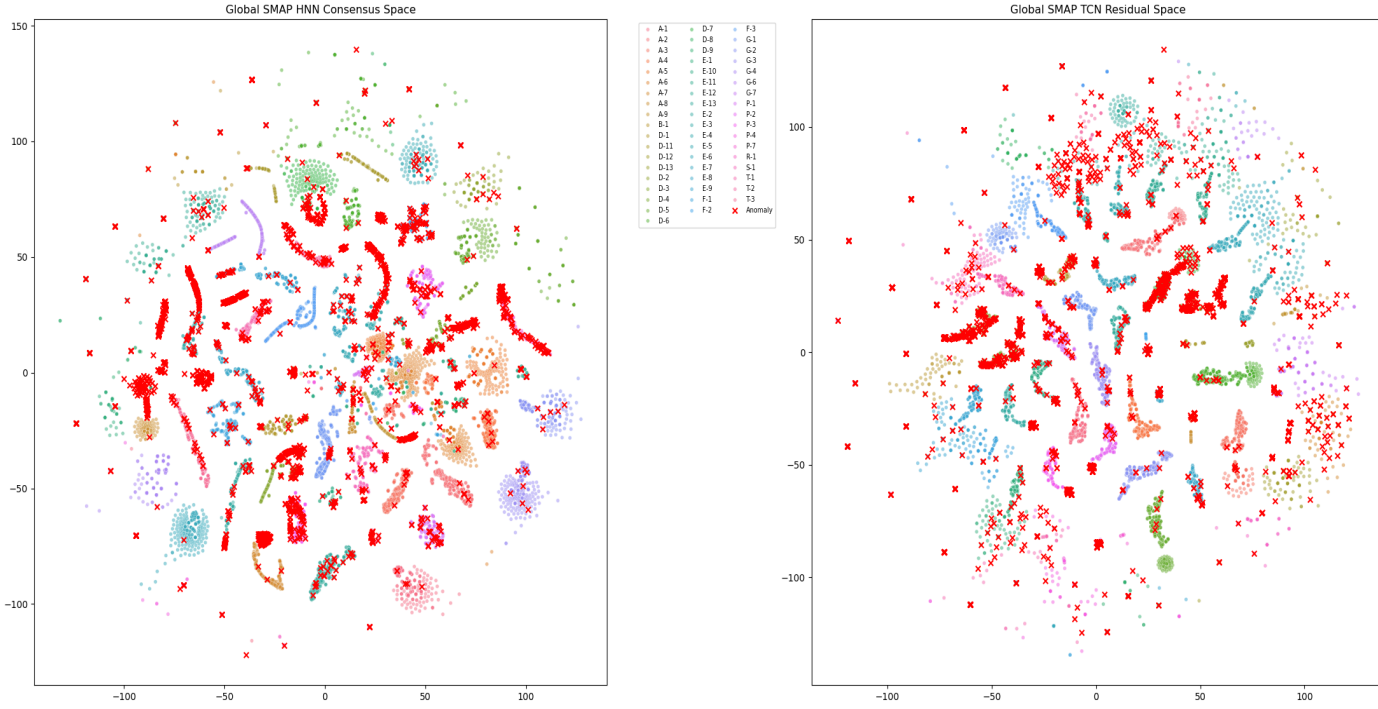
Figure 5: Latent space of SMD and PSM reported by paper



(a) Dual Latent Space of SMD.



(b) Dual Latent Space of PSM.



(c) Dual Latent Space of SMAP.

Figure 6: Dual latent space visualizations across datasets.

The latent space analysis shows the difference in the latent space of the acae paper latens and the dual latents, degradation of the circular psm latent space might be the reason for lower scores. A circual latent performing better might indicate perdiocity in data that the hnn + tcn

couldnt capture

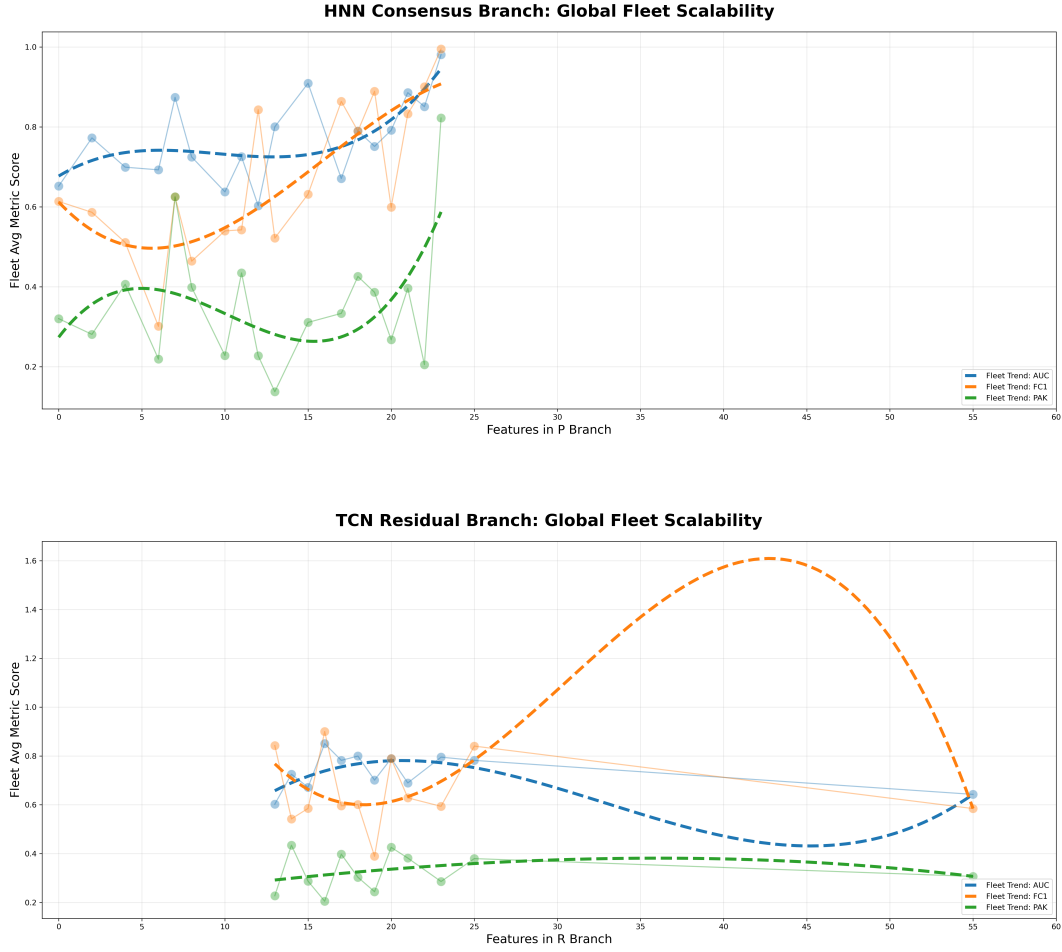


Figure 7: Plot of feature numbers of each latent and corresponding scores. Upper plot is no of features entering HNN path vs scores and Lower plot is no of features entering the TCN path

The scalability analysis shows that the HNN branch performs better as more consensual features are added, which lifts the entire system’s performance scores. This confirms that the physics-based Hamiltonian path handles higher complexity well without losing accuracy.

On the other hand, the TCN branch tanks the system-wide scores when too many features are pushed into it. Performance generally peaks around 20 features before the extra noise from high-dimensional sensors starts to overwhelm the decoder.

Seeing all the individual machine points together confirms that this bottleneck is a consistent pattern across the whole fleet rather than an isolated issue.

Currently the TCN branch is the most probable bottleneck. It starts underperforming when given more features also it failed to capture the highly peridoic patterns of the PSM which i had hoped its speciality will be.It uses dialated convolutions with windows to get a receptive field of the data, but since the residual branch gets lost of noisy features, it probably getting

lost with a smaller window(64 to match the hnn) in the local fluctuations so it underperforms overall, either we have to increase the window or change it. Replacing it with a light mlp has shown better scores in smd dataset, indicating for highly noisy features temporal model that maps the sequence of events might break if noise breaks that order.

refs

References

- [1] J. Yu et al., “An adversarial contrastive autoencoder for robust multivariate time series anomaly detection,” *Expert Systems with Applications*, vol. 245, p. 123010, 2024.
- [2] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou, “Symplectic recurrent neural networks,” in *International Conference on Learning Representations (ICLR)*, 2020.
- [3] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, “Temporal convolutional networks for action segmentation and detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 156–165. DOI: 10.1109/CVPR.2017.113
- [4] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo, “An evaluation of anomaly detection and diagnosis in multivariate time series,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 6, pp. 2520–2537, 2021.
- [5] S. Kim, K. Choi, H.-S. Choi, B. Lee, and S. Yoon, “Towards a rigorous evaluation of time-series anomaly detection,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 7194–7201.