Drift-Aware Online Anomaly Detection in Smart Buildings via Temporal Variational Autoencoder Gradient Profile

Tran L. T. Le
Illinois Advanced Research Center at Singapore Ltd.
Singapore, Singapore
fiona.le@iarcs-create.edu.sg

Heng Chuan Tan
Illinois Advanced Research Center at Singapore
Singapore, Singapore
hc.tan@iarcs-create.edu.sg

Zhen Wei Ng
Illinois Advanced Research Center at Singapore Ltd.
Singapore, Singapore
zhenwei.ng@iarcs-create.edu.sg

Zbigniew Kalbarczyk
University of Illinois at Urbana-Champaign
Illinois, USA
kalbarcz@illinois.edu

David K. Y. Yau
Singapore University of Technology & Design
Singapore, Singapore
david_yau@sutd.edu.sg

Daisuke Mashima
Singapore University of Technology and Design
Singapore, Singapore
daisuke_mashima@sutd.edu.sg

Xin Lou
Singapore Institute of Technology
Singapore, Singapore
lou.xin@singaporetech.edu.sg

Abstract

Smart buildings operate under evolving conditions caused by changing control setpoints, and varying occupancy, while also facing increasing cyber threats. This non-stationarity makes online anomaly detection challenging, as models must adapt to benign system changes without being corrupted by malicious behavior. Existing adaptive methods can handle distribution shifts but often fail to distinguish normal drift from attacks, leading to false alarms or unsafe model updates. We propose a drift-aware online anomaly detection framework that explicitly separates normal operation, normal drift, and attack behaviors. The framework is built on a Temporal Convolutional Network–based Variational Autoencoder (TCN-VAE) trained on normal data and leverages gradient-based sensitivity representations to train a downstream classifier that enables selective and contamination-free model updates during online inference. Experiments on real-world building datasets show that selective updates outperform no-update strategies, achieving up to 0.98 AUC with contamination rates below 3%. To generalize across different floors within the same building, we apply X-means to adapt clustering granularity to increased intra-class variability. Evaluations against state-of-the-art baselines demonstrate superior performance across cross-floor datasets. The source code is available at https://github.com/illinois-arcs/Drift-Aware_AD.

CCS Concepts

• **Security and privacy → Intrusion detection systems**; • **Computer systems organization → Embedded and cyber-physical systems**; • **Computing methodologies → Machine learning approaches**.

Keywords

Anomaly Detection, Cyber-Physical Systems, Building Management Systems, Intrusion Detection, Normal Drift, Online Learning, TCN-VAE, Gradient-Based Representations

ACM Reference Format:
Tran L. T. Le, Heng Chuan Tan, Zhen Wei Ng, Zbigniew Kalbarczyk, David K. Y. Yau, Daisuke Mashima, and Xin Lou. 2026. Drift-Aware Online Anomaly Detection in Smart Buildings via Temporal Variational Autoencoder Gradient Profile. In *The 17th ACM International Conference on Future and Sustainable Energy Systems (E-Energy '26), June 22–25, 2026, Banff, AB, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3744255.3811742

E-Energy '26, Banff, AB, Canada

ACM ISBN 979-8-4007-2011-6/26/06
https://doi.org/10.1145/3744255.3811742

1 Introduction

Cyber-physical systems (CPS), such as smart buildings, rely on advanced sensor technologies to monitor physical processes and provide actionable feedback to actuators for reliable operations [1]. As these systems undergo increasing digitalization, they are increasingly vulnerable to cyber-attacks, motivating the development of various deep anomaly detection (AD) models. However, sensors and actuators in real-world systems can drift over time due to environmental conditions, and changes in operational conditions (e.g., office hours vs. weekends or setpoint adjustments at the user's request). As a result, models trained on historical "normal" data may become outdated, leading to increasing false alarms. Frequent retraining to accommodate these changes is often impractical due to high computational costs and limited labeled data, especially in safety-critical domains.

Streaming and online AD methods have been proposed to address evolving data distributions [5–7, 12, 13, 19–22]. While these methods enable continual adaptation, most treat all deviations from

the current model as anomalies and update the model using all samples, including those identified as anomalies. Crucially, they do not explicitly distinguish between true attacks and benign normal drift. As a consequence, blindly adapting to all incoming samples risks contaminating the model with attack data, ultimately degrading detection performance and system reliability.

To address this limitation, we develop an adaptive online streaming AD framework for detecting abnormal setpoint changes in heating, ventilation, and air conditioning (HVAC) systems, while distinguishing them from normal drift. The framework integrates a Variational Autoencoder (VAE) with a Temporal Convolutional Network (TCN) for multivariate sensor data. The key idea is to leverage VAE gradient-based representations for downstream anomaly characterization, while the TCN captures temporal dependencies in these gradient representations to enable adaptive AD under evolving data streams. Specifically, the gradients are defined as the sensitivity of the reconstruction loss with respect to the input features, reflecting how strongly each feature influences VAE reconstruction. Intuitively, normal changes tend to induce small and structured variations in these gradients, whereas attacks produce more abrupt and distinctive patterns.

Leveraging these gradient profiles, we design a downstream classifier based on K-Means clustering with the Hungarian algorithm [10] to explicitly map clusters to normal operation, normal drift, and attack. To improve generalization to data from other floors, we employ the X-means clustering method [16] to automatically determine an appropriate number of clusters, allowing the framework to adapt its clustering granularity to increased behavioral heterogeneity without manual tuning. This clustering enables selective and contamination-aware model updates, i.e., the model is updated only using samples classified as normal or normal drift, while samples identified as attacks are excluded.

The resulting adaptive and online framework is scalable and practical in scenarios where labeled training data are scarce or unavailable, particularly for mission-critical systems such as building management systems and microgrids. By jointly modeling temporal dynamics, gradient-based representations, and drift-aware adaptation, the proposed approach provides a practical solution for robust AD under evolving system conditions. Our main contributions:

- **Drift-aware, contamination-free online adaptation:** We propose an adaptive streaming framework that separates normal behavior, normal drift, and attacks, enabling safe and reliable model updates under non-stationary CPS.
- **Input-gradient-based sensitivity representations:** We leverage gradients extracted from a frozen TCN-VAE to capture how the model trained on normal data responds to input perturbations and enable discrimination between normal operation, normal drift, and attack behaviors beyond reconstruction error alone.
- **Adaptive clustering for cross-floor generalization:** We employ X-means to automatically select the number of clusters when transferring a pretrained model across floors, improving robustness to distribution shifts by capturing increased data variability.
- **Practical deployment with limited supervision:** The framework requires minimal labeled data, making it suitable for long-term deployment in mission-critical CPS.

2 Related Work

2.1 Gradient-based Representations for AD

Recently, a small number of works have explored backpropagated gradient representations as auxiliary features for AD in deep neural networks, with evaluations largely focused on static or non-temporal domains (e.g., computer vision) [11]. These studies show that gradient information can complement reconstruction error or latent activations. However, such approaches treat gradients independently at the sample level rather than as behavioral representations that capture evolving system dynamics.

In the context of networked systems, the GEE framework [15] demonstrates that gradient information derived from the VAE can be used for post-hoc explanation and attack fingerprinting. However, this approach generally produce non-discriminative, binary normal and abnormal decisions and do not address fine-grained separation of different attack classes.

In contrast, our work treats input-gradient profiles as behavioral representations that characterize how a learned normal model responds to temporal inputs. By clustering these gradient profiles in an unsupervised manner, our approach explicitly separates normal operation, drift, and attacks without relying on labeled attack data or reconstruction-error thresholds. This enables our model to adapt selectively in non-stationary CPS environments, while minimizing contamination from anomalous data.

2.2 Online and Streaming AD in CPS

Many methods have been developed to detect anomalies in evolving data streams [5, 6]. For example, Malhotra et al. proposed an LSTM-based encoder–decoder (LSTM-ED) model that detects anomalies based on reconstruction errors [14]. The intuition is that a model trained on normal sequences would not be able to reconstruct abnormal data accurately and would therefore produce higher reconstruction errors and, in turn, higher anomaly scores. However, this approach lacks mechanisms for online adaptation and cannot distinguish between normal drift and true anomalies. Consequently, it may generate a large number of false positives. Another line of work called REBM integrates recurrent neural networks (RNN) with energy-based models (EBM) to detect anomalies in time-series data [22]. In this architecture, the RNN captures the temporal dependencies of the input stream to guide the EBM in reconstructing the observed data. If the reconstruction is good, the "energy" (i.e., reconstruction error) is low, which implies the input aligns with the learned normal patterns. On the other hand, high energy values indicate potential anomalies.

Tian et al. developed CDAOM (Concept Drift Adaptation for Online Anomaly Detection), an adaptive framework built upon the One-Class Support Vector Machine (OCSVM) [19]. CDAOM detects data drift by evaluating the distance between each incoming sample and the support vectors separating normal data from anomalies. When drift is detected, the model incrementally updates itself by adjusting the decision boundary to reflect changes in the data distribution. Yoon et al. developed ARCUS, an online deep

AD framework that employs multiple models to stay adaptive over time [21]. A new model is created when the reliability of the model pool falls below a threshold; otherwise, ARCUS keeps the current model pool and updates only the model contributing most to the pool's reliability. While both ARCUS and CDAOM are highly adaptive, they cannot differentiate true anomalies from normal drift, which may cause model drift and corruption over time.

The work [20] introduces STARE, a stationary-region-skipping strategy for streaming outlier detection. The key observation is that, in many data streams, large portions of the data space remain stable across windows. While traditional density-based methods recompute kernel densities for every window which incurs redundancy and latency, STARE updates only the regions whose distributions change. The data space is partitioned into grid cells, each with a kernel center (cell centroid) and an integer weight equal to the number of points in the cell. STARE estimates densities via Kernel Density Estimation (KDE) but avoids full recomputation by approximating changes through the grid-weight updates. Across sliding windows, it tracks arrivals and expirations to compute each cell's net weight change and triggers a density update only when this cumulative change exceeds an adaptive threshold. This grid-based design enables sublinear updates with respect to total data size, substantially reducing computation while preserving detection fidelity. However, although that approach updates only regions with distributional change, it cannot ascertain a priori whether those regions are contaminated. In contrast, we update the model exclusively with samples validated as clean, thereby minimizing contamination, which is crucial for safety-critical CPS.

Another work [7] offers an online AD via a forest of random-cut trees (RRCF) that are updated incrementally; points are scored by their structural displacement. Yet it lacks temporal modeling, cannot distinguish benign drift from attacks, and yields thresholded scores without class semantics. Our drift-aware TCN-VAE pipeline, trained on normal data, uses gradient profiles as fingerprints to separate normal drift from attacks and updates only on safe data, limiting contamination while enabling label-frugal attack grouping.

Li et al. introduced COPOD [12], a copula-based outlier detector that fits empirical copulas separately to the positive/negative tails and central regions of the data distribution, enabling precise tail probability estimation for scoring outliers without multivariate normality assumptions or predefined copula families. Building on this, Li et al. proposed ECOD [13], an unsupervised outlier detector using empirical cumulative distribution functions (ECDFs) computed on one-dimensional feature marginals. It transforms data to ECDF ranks and derives outlier scores from tail probabilities, offering parameter-free detection with strong theoretical guarantees and efficiency on high-dimensional datasets. These statistical outlier detectors prioritize scalable, assumption-light tail modeling for static or tabular data but overlook temporal dependencies and lack mechanisms for multi-class refinement during training.

Although prior works demonstrate strong performance in adapting to data drift, their solutions are typically limited to scenarios where the model incrementally adapts to newly observed samples assumed to belong to the seen distribution. These approaches do not explicitly consider whether incoming samples correspond to genuine attacks or normal operating conditions. As a result, they are less practical for our domain, where accurate abnormal detection is critical and the model must continuously learn evolving normal behaviors while avoiding contamination from attack samples.

3 Preliminaries

3.1 Smart Building

Modern smart buildings are a type of CPS in which a centralized building management system (BMS) communicates with multiple interacting physical subsystems, such as energy management, lighting, and HVAC, to provide essential services to occupants. From a CPS perspective, the BMS forms the cyber layer that collects sensor measurements from these physical systems to enable real-time monitoring and control of the building environment.

In this work, we focus on a real-world BMS that controls the HVAC system of a 16-story research facility in the western part of Singapore, specifically the HVAC serving office spaces on Level 4 and Level 14. This research facility houses multiple research centers and office spaces and is representative of a typical modern office environment.

As shown in Figure 1, the HVAC system monitors airflow, temperature, and pump status and controls the damper and valve to regulate air within the office space. Specifically, when the office temperature is higher than the temperature setpoint, the valve opens wider to allow more chilled water to cool the air in the duct before it is delivered into the office space. If the airflow in the office space falls below the setpoint, the damper opens more to increase the flow of cooled air.

Operationally, the valve and damper control actions are realized through the coordinated operation of the air handling unit (AHU) and chiller pump. The AHU typically starts first to purge stale air and supply fresh air into the variable air volume (VAV) system. The chiller pump is then activated to circulate chilled water, allowing the air temperature in the room to reach the desired setpoint. When both the AHU and pump are active, the airflow is high and remains close to the airflow setpoint. In response, the damper adjusts its position and stabilizes to maintain the required airflow. The temperature setpoint was configured to 24°C on Level 4, and to either 25°C or 23°C on Level 14 for different meeting rooms, in accordance with local operational guidelines to ensure occupant comfort and energy efficiency.

3.2 Problem Formulation

In this work, we focus on detecting anomalies arising from abnormal setpoint control in HVAC systems within BMS. Under normal operation, setpoints such as temperature and airflow are adjusted according to control logic, occupancy patterns, and operational constraints, resulting in structured and predictable system behavior over time. However, abnormal setpoint changes may arise due to user misconfiguration, faulty automation logic, or malicious manipulation of control commands. Such anomalies can lead to inefficient energy usage, occupant discomfort, or unsafe operating conditions. Detecting these anomalous behaviors is challenging due to legitimate operational drift, changing occupancy, and limited availability of labeled attack data. The goal of this work is to identify anomalous setpoint behaviors while distinguishing them from normal system drift, enabling reliable and safe adaptation of detection models under evolving system conditions.

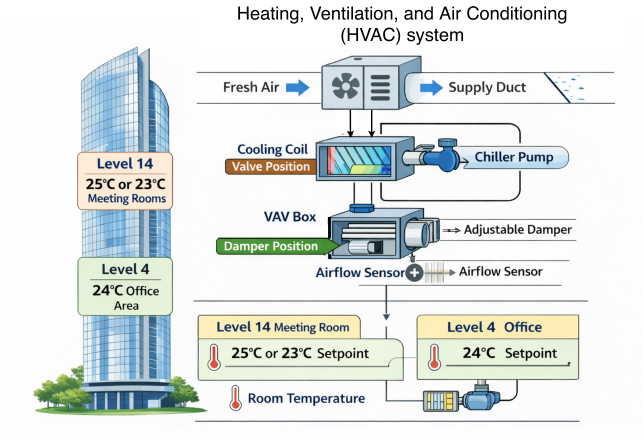


Figure 1: Overview of a BMS controlling and monitoring the HVAC system to ensure user comfort and proper functioning.

Threat Model. We consider both remote and insider attackers. We assume that remote attackers gain access to the BMS workstation via compromised credentials or remote services, while insider attackers (e.g., malicious vendors or operators) have legitimate system access. Once access is established, attackers can move laterally within the BMS network and exploit industrial protocols (e.g., BACnet, Modbus) to manipulate data exchanged between sensors, PLCs, and the BMS. Specifically, attackers can launch false data injection attacks and false command injection attacks, where sensor readings are manipulated to mislead control logic and malicious commands are issued to actuators such as dampers and valves, respectively. These attacks can result in occupant discomfort, equipment wear and tear, or even system failure.

3.3 Primer on Variational Autoencoders (VAE)

VAEs share the same encoder-decoder architecture as a conventional autoencoder [9]. The encoder compresses the input data into a latent representation, while the decoder uses this latent representation to reconstruct the input. However, unlike a conventional autoencoder that maps an input to a fixed point in latent space, a VAE maps the input to a latent distribution, as shown in Figure 2.

More formally, given an input x, the encoder network parameterized by ϕ outputs a variational posterior distribution $q_\phi(z \mid x)$, which is typically modeled as a diagonal Gaussian with mean μ and variance σ. A latent vector z is sampled from this distribution and passed to the decoder network $f_\theta(\cdot)$, which reconstructs the input $\hat{x} = f_\theta(z)$ that parameterizes the conditional likelihood $p_\theta(x \mid z)$. Because VAEs model latent distributions, their loss function includes a KL-divergence term (in addition to the reconstruction loss) to regularize the latent space. This term ensures that the learned representations follow a standard normal prior with zero mean and variance of 1.

Since the latent sample z is drawn randomly from a probability distribution, the sampling operation is non-differentiable with respect to mean μ and standard deviation σ, preventing gradients from the reconstruction loss from being directly backpropagated to the encoder during training. VAEs resolve this issue via the

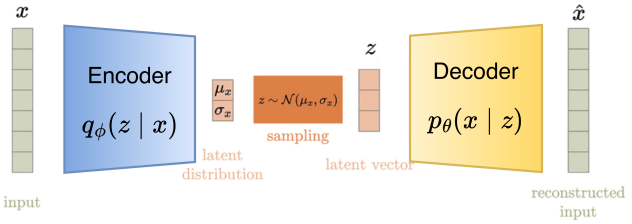


Figure 2: VAE Architecture [2]

reparameterization trick

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \tag{1}$$

where a standard normal noise variable ϵ is sampled and then deterministically scaled by the standard deviation σ and shifted by the mean μ, enabling gradients to propagate through the sampling operation to the encoder.

3.4 Temporal Convolutional Networks (TCN)

Temporal Convolutional Networks (TCNs) are convolutional architectures designed for sequence and time-series modeling. Unlike recurrent neural networks, TCNs use one-dimensional convolutions without recurrence, enabling parallel computation and stable training. TCNs were introduced by Bai et al. [3] and have been shown to be competitive with recurrent models across a wide range of sequence modeling tasks.

A key property of TCNs is causality, which ensures that the output at time step t depends only on current and past inputs. To efficiently model long-range temporal dependencies, TCNs employ dilated causal convolutions, which apply convolutional filters over past time steps with fixed dilation factors to expand the receptive field without increasing model depth. By stacking layers with increasing dilation factors, TCNs can capture both short- and long-term temporal patterns, as illustrated in the Figure 3a.

In practice, TCNs are implemented using residual blocks composed of dilated causal convolutions and nonlinear activations. Compared to recurrent models such as LSTMs, TCNs offer parallelizable training, flexible receptive field control, and robustness to vanishing gradients, making them well suited for multivariate time-series modeling in CPS, as illustrated in the Figure 3b, c.

4 Proposed Framework

Figure 4 shows the proposed framework, where a TCN is used as both the encoder and decoder within a VAE and serves as the backbone of our AD framework. We use a TCN to capture temporal dependencies efficiently, which makes it suitable for streaming time-series data. We further choose a VAE instead of a deterministic autoencoder because its latent regularization encourages more stable and generalized representations of normal behavior, making the model less sensitive to minor distributional changes caused by non-malicious shifts from the original normal operating conditions. Furthermore, VAE enables the use of gradient-based representations for anomaly characterization to support adaptive model updates without the need to build or retrain a new model from scratch, which is computationally expensive and difficult to scale.

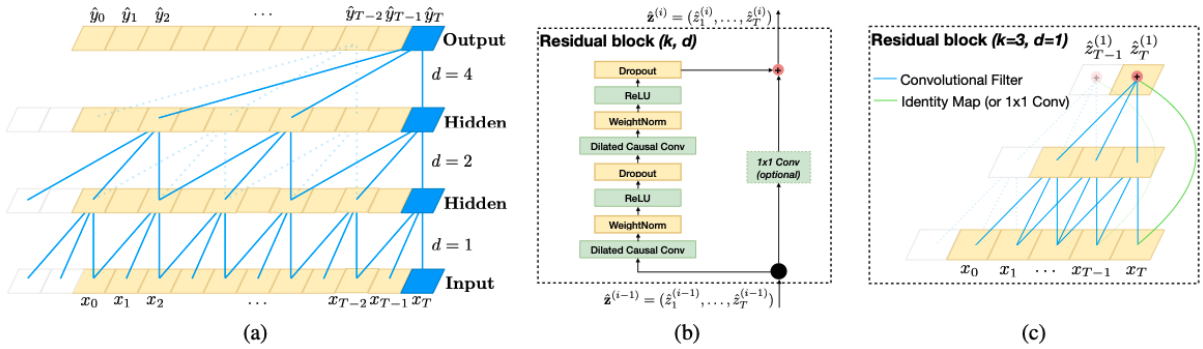


Figure 3: TCN architecture, illustrating dilated causal convolutions and residual blocks, adopted from [3]. (a) Dilated causal convolution with dilation factors d=1,2,4 and filter size k=3. (b) TCN residual block, where a 1×1 convolution is used when the input and output dimensions differ. (c) TCN residual connection with blue lines denoting convolutional filters and green lines representing identity mappings.

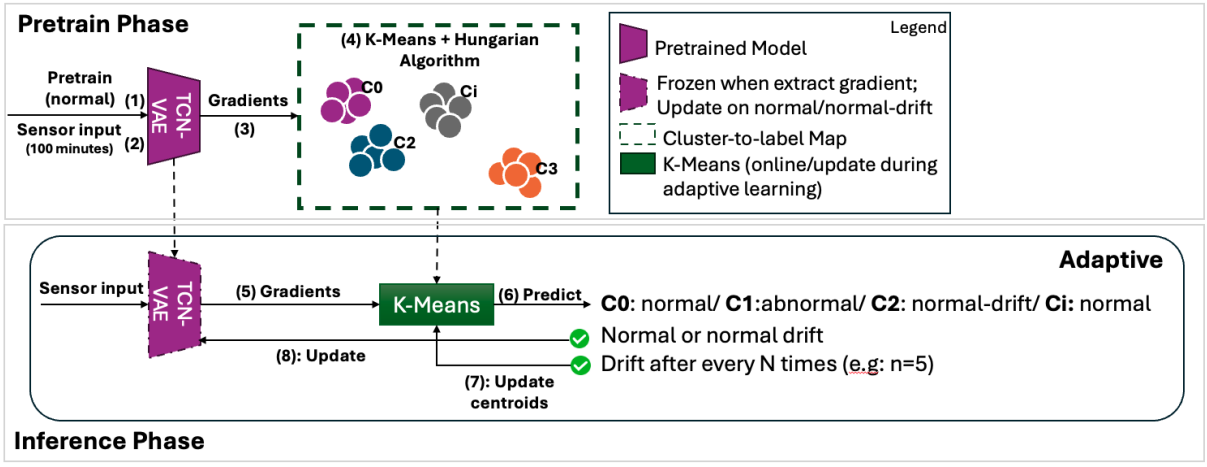


Figure 4: TCN-VAE with Gradient Profiles for Anomaly Detection

The TCN-VAE is first trained on normal time-series data to establish a baseline of normal behavior. Once trained, the model weights are frozen to preserve the learned representation. To bootstrap the downstream classifier, a small set of labeled samples from each class (i.e. normal, normal drift, and attack) is fed into the frozen model. The gradients extracted from these samples are then used to train a k-means classifier. Class labels are assigned after clustering to determine whether each cluster represents normal behavior, normal drift, or an attack. While this clustering stage requires labeled samples to initialize the cluster-to-label mapping, the labeling requirement is substantially lower than that of conventional supervised approaches. In real-world deployment, the proposed (pretrained) framework operates without labeled data and does not assume prior knowledge of the attack windows. Instead, it adapts online as (unlabeled) operational data becomes available over time.

4.1 Pretraining Phase

We consider multivariate time windows $x \in \mathbb{R}^{T\times d}$ extracted from a stream of sensor measurements, where each window contains T time steps and d features. We adopt a Temporal Convolutional Network variational autoencoder (TCN-VAE) as a backbone for gradient-based feature extraction, where features are derived from the sensitivity of the reconstruction loss with respect to the input. Specifically, the TCN-VAE is pretrained on normal samples denoted as $\mathcal{D}_{\text{norm}} = \{x_i\}_{i=1}^{N}$, and gradients computed from this pretrained model are used to characterize deviations from normal behavior. All input features are min–max normalized to $[0, 1]$.

Given an input window, the encoder maps the input data x to the mean μ and variance σ^2 of a diagonal Gaussian distribution in the latent space.

$$q_\phi(z \mid x) = \mathcal{N}\big(\mu_\phi(x),\ \mathrm{diag}(\sigma^2_\phi(x))\big), \tag{2}$$

where the diagonal covariance assumes conditional independence among latent dimensions. This design choice simplifies inference and enables efficient optimization via the reparameterization trick, while remaining sufficient for modeling normal system behavior in practice. A standard normal prior $p(z) = \mathcal{N}(0, I)$ is adopted for the latent variable z, which regularizes the latent space by

encouraging the learned variational posterior to remain close to a known reference distribution.

In training the model, the objective is to maximize the evidence lower bound (ELBO) on normal samples by minimizing the negative ELBO (β-VAE) in Eqn. 3, which consists of a reconstruction error term and a KL-divergence term that regularizes the latent space toward a standard normal prior.

$$\mathcal{L}_{\text{VAE}}(x) = \mathbb{E}_{q_\phi(z|x)}\big[\|x - f_\theta(z)\|_2^2\big] + \beta\, D_{\text{KL}}\big(q_\phi(z \mid x) \,\|\, \mathcal{N}(0, I)\big), \tag{3}$$

where $q_\phi(z \mid x)$ maps an input x to the parameters $(\mu_\phi(x), \sigma_\phi(x))$ of the latent distribution. The function $f_\theta(\cdot)$ reconstructs the input from the latent variable z. The hyperparameter β controls the trade-off between reconstruction fidelity and latent regularization.

We set $\beta = 0.001$, following common practice in VAE-based AD, where small β values are preferred to preserve accurate reconstruction of normal operational patterns while imposing only mild regularization on the latent space. Stronger KL regularization may oversmooth representations and reduce anomaly sensitivity.

The expectation in Eqn. 3 is estimated using the reparameterization trick: $z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$. After training on normal data $\mathcal{D}_{\text{norm}}$, we freeze the encoder and decoder parameters (ϕ, θ) and use the model solely in inference mode for anomaly scoring and downstream analysis.

4.2 Gradient Profile

We extract *input gradients*, defined as the gradients of the reconstruction loss with respect to the input features [17, 18], by backpropagating the loss $L(x)$ to the input while keeping the model parameters (ϕ, θ) fixed:

$$\nabla_x L(x) = \frac{\partial}{\partial x}\,\|x - g(x;\theta)\|_2^2 = 2\,r(x) \;-\; 2\,J_g(x)^\top \,\text{vec}\big(r(x)\big), \tag{4}$$

where $r(x)$ is defined as the residual, i.e., the difference between the input and the reconstructed input, $J_g(x) = \frac{\partial g(x;\theta)}{\partial x}$ is the Jacobian and $\text{vec}(\cdot)$ is an operation that flattens a matrix into a vector. Eqn. (4) follows directly from the chain rule of backpropagation, with gradients computed only with respect to the input features and no updates applied to the model parameters.

Given these gradients, we vectorize and ℓ_2-normalize them to obtain the *gradient profile* $\varphi(x) = \text{norm}(\text{vec}(\nabla_x L(x)))$, which serves as the input to a downstream classifier that distinguishes among normal, drift, and attack samples. Intuitively, $\nabla_x L(x)$ quantifies the sensitivity of the reconstruction loss to each input feature, indicating how perturbations to individual features would affect the reconstruction error. These sensitivity patterns can differ substantially for normal, drift, and attack samples, even when their reconstruction errors are similar, making gradient profiles ideal for classifying samples. Representative gradient profiles under normal, drift, and attack conditions are shown in Figure 1 in Appendix A.

4.3 Clustering

To train the downstream classifier, we use a small labeled set with 100 samples per class (normal, normal drift, and attack). These samples are passed through the frozen model to obtain their gradient profiles. Based on this labeled subset, we compute the mean gradient vector for each class, which serves as the anchor centroid.

Using majority voting alone to assign labels to clusters may miss cluster representations for minority classes. In particular, when one class (e.g., normal) dominates the data, multiple clusters may be assigned to that class, leaving "normal drift" and "attack" classes without dedicated cluster representations. To avoid this issue, we adopt a two-stage label-to-cluster assignment strategy. First, we use the Hungarian algorithm [10] to ensure each label gets at least one dedicated "anchor" cluster. The algorithm works by constructing a cost matrix where entry (i, j) represents how many samples from label j appear in cluster i. The Hungarian algorithm then finds the optimal one-to-one assignment that maximizes the total number of correctly assigned samples, ensuring each label is matched to exactly one cluster—the cluster where it has the strongest presence. This guarantees balanced representation across all labels.

After this assignment, the remaining clusters are labeled using majority voting based on the class distribution of samples within each cluster. This allows multiple clusters to correspond to the same class, capturing variations within a class, such as normal behavior under different operating conditions, while ensuring that every class has at least one clear representative. The Hungarian algorithm with majority voting enables fine-grained clustering while preserving one cluster per label.

While three clusters are sufficient to represent normal, normal drift, and attack, the underlying data distribution and system behaviors may change when a pretrained model is deployed on a different floor. As shown in our cross-floor evaluation in Section 6.4, fixing (K=3) can lead to suboptimal performance because the new target environment may exhibit finer-grained operating modes or previously unseen variations. To address this issue, we employ X-means [16] as a preprocessing step for each new location to estimate the number of clusters prior to deployment. This enables the clustering stage of our framework to adapt to distributional shifts and improve generalization across floors. Specifically, X-means is applied to gradient features collected from the target environment, and the optimal number of clusters is determined automatically by maximizing the Bayesian Information Criterion (BIC).

4.4 Inference Phase

For each incoming window x, the reconstruction loss is first computed, after which the input gradient $\nabla_x L(x)$ is obtained via backpropagation according to Eqn. 4. The resulting gradient is then vectorized and ℓ_2-normalized to form the gradient profile. This gradient profile serves as a learned feature representation that captures how the model responds to different window inputs, i.e., normal, normal drift, and attack samples. Each window is assigned to a K-Means cluster, which is mapped to one of three semantic classes (normal, normal drift, or attack) using the learned cluster-to-label mapping. During online adaptation, we update the pretrained model using data predicted as normal or normal drift, while excluding predicted attack samples to prevent contamination of model parameters.

For K-Means drift adaptation, we update cluster centroids periodically at the batch level. Each batch consists of 64 window-level gradient vectors. A batch is assigned a label using a majority vote over the 64 window-level K-Means predictions. If the majority label is normal drift, the standardized gradient vectors from that batch

are buffered. Once five normal-drift batches have been accumulated, the centroids corresponding to the normal-drift clusters are updated using the buffered data. Centroid updates are performed at the batch level rather than for individual windows to reduce sensitivity to noise and outliers. Updating centroids on the individual window would increase sensitivity to outliers, potentially destabilizing cluster boundaries and leading to incorrect decisions.

5 Data Collection in BMS

5.1 Experiment Setup

Experiments were conducted by perturbing the temperature setpoints in the office and meeting rooms to study the HVAC system's response under abnormal conditions. These perturbations were introduced by modifying the PLC control logic to toggle the temperature setpoint at fixed intervals. This manipulation resulted in frequent valve actuations, preventing the room temperature from stabilizing at the desired level and causing user discomfort. Repeated opening and closing of the valve can also cause excessive stress on the mechanical components, leading to wear and a reduced operational lifespan.

Such changes to PLC control logic or behavior can result from either unintentional user misconfiguration or malicious activity. In the case of misconfiguration, errors may occur during routine maintenance or firmware updates, where incorrect parameter values or improper scheduling settings are deployed to the PLC. On the other hand, attackers may exploit vulnerabilities in control protocols or engineering workstations to modify PLC parameters or control logic as described in our threat model.

5.2 Datasets and Labeling

Three datasets were collected from the 16-story research facility: one from an office space on Level 4 (L4) and two from separate meeting rooms (MR) on Level 14 (L14-MR1 and L14-MR2). The L4 office is vacant, while both meeting rooms on Level 14 are occupied with varying occupancy levels over time. For the L4 office, the temperature setpoint was cycled among 21°C, 24°C, and 27°C every 2 hours. In L14-MR1, it was toggled among 25°C, 28°C, and 30°C every 10 minutes, while in L14-MR2, it was toggled among 23°C, 27°C, and 30°C every 10 minutes. These temperature setpoints and update intervals were deliberately configured differently to generate diverse representative scenarios for stress-testing the proposed TCN-VAE model.

Each dataset contains ten features: three related to damper control (airflow, airflow setpoint, and damper feedback), three related to valve control (room temperature, temperature setpoint, and valve feedback), two pump status features indicating the chilled water system operating hours, one feature for outdoor temperature, and one condensation alarm feature. The datasets were manually labeled to facilitate performance benchmarking against baseline approaches.

To enable this evaluation, each dataset was labeled into normal (class 0), normal drift (class 1) and attack (class 2) based on the operating conditions of the air handling unit (AHU), chiller pump system, temperature setpoint and observed data distributions.

The valve position is determined by the room temperature and the temperature setpoint. When the room temperature is too high, the valve opens to allow more chilled water into the cooling coil to cool the air passing over it. When the temperature is too low, the valve closes to reduce cooling. Once the room temperature reaches the desired setpoint, the valve position stabilizes and remains constant. Samples corresponding to these conditions represent normal operation and are labeled as normal.

During weekends and after office hours, the AHU and pumps are turned off, indicating non-operational conditions. Although this represents a deviation from the normal operating profile, it is still considered part of normal operation. To differentiate these cases from normal operation, we classify the samples as normal drift, regardless of the valve and damper positions.

Samples are labeled as attacks when the temperature setpoint deviates from its expected value (i.e., 24°C on level 4 and either 23°C or 25°C on level 14, depending on the meeting room configuration) and the valve behavior is inconsistent with observed normal operation. However, we only consider these conditions as attacks during office hours, when the AHU and pump are operating and airflow is high. Setpoint changes outside office hours are considered normal drift since they have no impact on the occupants. Table 1 summarizes the dataset characteristics. Further details of the dataset are presented in Appendix B (Figure 2), where we provide an illustration of the time-varying HVAC system behavior under normal, normal drift and attack conditions to motivate the use of gradient representations.

6 Evaluation

In this section, we evaluate the performance of our TCN-VAE framework under three updating strategies—no update, full update, and selective update—to demonstrate the benefit of the proposed selective adaptation mechanism. Next, we evaluate our model on multiple HVAC datasets, analyze its performance across datasets collected from different floors, and compare our approach with state-of-the-art baselines.

6.1 Parameter Settings

Model Architecture / Configuration. Our TCN-VAE comprises three TCN blocks with 64 filters, processing sequences of 20 time steps across 10 sensor dimensions. The encoder maps inputs to a 10-dimensional latent space using ReLU activation and dropout (0.2). Each block uses causal convolutions with kernel size 3. The model is trained with the Adam optimizer (learning rate 1×10^{-3}) and batch size 64.

Computational Environment. Experiments were conducted on an Apple M4 Pro (12-core CPU, 24 GB unified memory), using Python 3.9.6 with TensorFlow 2.20.0 and Keras 3.10.0. The source code is available on GitHub (https://github.com/illinois-arcs/Drift-Aware_AD). Our framework requires no specialized hardware and can be deployed on commercial off-the-shelf (COTS) computing devices.

Dataset Splitting. We follow a chronological split between training and evaluation data. The TCN-VAE model is first trained using 5000 normal samples collected during earlier time periods. Subsequent time windows containing unseen normal data, normal drift and attack data are then used exclusively for evaluation.

Table 1: Summary of datasets used for model evaluation

Dataset	Collection Period	Total	Normal	Normal Drift	Abnormal
Level 4 Office area (vacant)	Jul 2 – Sep 8, 2025	97,062	27,256 (28.08%)	66,918 (68.94%)	2,888 (2.98%)
Level 14 MR-1 (occupied)	Sep 24 – Oct 8, 2025; Oct 15 – Dec 1, 2025	87,010	26,737 (30.72%)	55,765 (64.09%)	4,508 (5.18%)
Level 14 MR-2 (occupied)	Sep 17 – Oct 8, 2025; Oct 15 – Dec 1, 2025	98,141	23,426 (23.87%)	62,635 (63.82%)	12,080 (12.31%)

6.2 Performance under Different Update Strategies

To demonstrate the effectiveness of our proposed framework, we evaluate its performance under three different updating scenarios: (1) no update, (2) update all, and (3) selective update. The first scenario corresponds to a static model where the model is trained once and kept fixed throughout the inference phase. The second scenario refers to updating the model on every sample regardless of whether it is normal, normal drift, or attack. The third scenario refers to the proposed approach where the model updates only when samples are classified as normal or normal drift, while attack samples are excluded to avoid model corruption.

We evaluated the different model update scenarios in terms of macro AUC, F2-score and contamination rate. Specifically, we used macro AUC, F2-score to quantify the classification performance, while the contamination rate was used to assess the performance of the model updates. When analyzing the classification metrics, we treat abnormal samples as the positive class, i.e., True Positive (TP), whereas we treat normal and normal drift samples as the positive class when evaluating the model update performance.

F2-score is used because we prioritize recall over precision. We want the model to detect as many abnormal samples as possible, even if it means having some false positives. A high false positive rate means that the model is not up-to-date with the current data distribution but at least the model is not corrupted with bad samples. On the other hand, contamination rate (CR) quantifies the fraction of updates that were actually abnormal. A low contamination rate means that the model is not corrupted with abnormal data samples.

The results in Table 2 show that the proposed TCN-VAE with selective updates achieves AUC, F2-score, and recall that are comparable to full updates across all datasets. The high performance of the full update strategy can be attributed to the small percentage of attack samples in each dataset such that updates on these samples do not immediately degrade detection accuracy. In contrast, no update has the worst performance in terms of F2-score and recall metrics. These findings highlight the importance of online adaptation and show that selective updating preserves detection performance while reducing the risk of model contamination.

6.3 Latency Performance

Figure 5 compares the model update latency under the update all and selective update strategies. Across all evaluated settings (L4, L14-MR1, and L14-MR2), selective update consistently achieves lower update latency, reducing the average model update time from approximately 130–135 ms to 85–100 ms. This reduction stems from the fact that selective update avoids unnecessary parameter updates on contaminated windows, thereby reducing computational overhead during online operation. These results demonstrate that drift-aware selective updating improves security and robustness while reducing model update latency by approximately 25–35%.

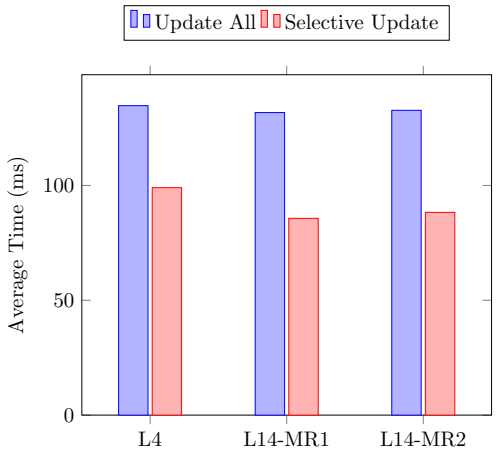


Figure 5: Average model update time under different update strategies.

Table 2: Model Performance under different update strategies ($K = 3$) (✗: no update, ✓: full update, S: selective update)

Datasets	Update	AUC ↑	F2 ↑	Recall ↑	CR ↓
L4	✗	0.940	0.862	0.879	–
	✓	0.980	0.949	0.942	0.55%
	S	**0.980**	**0.950**	**0.943**	**0.54%**
L14-MR1	✗	0.810	0.781	0.795	–
	✓	0.830	0.788	0.794	3.10%
	S	**0.830**	**0.788**	**0.795**	**2.81%**
L14-MR2	✗	0.700	0.610	0.690	–
	✓	0.760	**0.709**	**0.757**	0.35%
	S	**0.760**	0.701	0.752	**0.29%**

Contamination rate (CR) is not applicable for the no-update model.s

6.4 Same-room/Cross-room Evaluation

Same-room Evaluation. Table 2 shows that using K-Means with a fixed number of clusters (K=3) is effective when the model is evaluated on the same floor it was pretrained on, as the three clusters naturally correspond to normal, normal-drift, and abnormal behaviors. Selective update on normal and normal-drift data consistently improves detection performance over the no-update baseline

Table 3: Cross-floor evaluation of pretrained models with adaptive cluster selection (X-means).

Pretrain Floor	Test Floor	K	AUC ↑	F2 ↑	Recall ↑
L4	L14-MR1	3	0.780	0.687	0.731
		15	**0.80**	**0.681**	**0.726**
	L14-MR2	3	0.87	0.761	0.779
		38	**0.89**	**0.835**	**0.845**
L14-MR1	L14-MR2	3	0.82	0.747	0.761
		15	**0.87**	**0.849**	**0.847**
	L4	3	0.98	0.942	0.937
		41	**0.98**	**0.793**	**0.863**
L14-MR2	L14-MR1	3	0.76	0.682	0.708
		17	**0.81**	**0.681**	**0.711**
	L4	3	0.96	0.731	0.785
		25	**0.97**	**0.780**	**0.827**

Values shown in bold were obtained using adaptive cluster selection via X-means.

(e.g., F2 increases from 0.862 to 0.950 on L4) while maintaining low contamination rates (e.g., less than 1% on L4 and L14-MR2).

When comparing AUC, F2, and recall across the two levels, the selective update model performs best on L4. This is because the L4 office space is vacant and exhibits relatively stable system behavior with little variability, making anomalies easier to detect. In contrast, people enter and leave the meeting rooms on L14, which introduces greater variability in system dynamics. Even with this added complexity, the selective update model achieves comparable performance on L14-MR1 and L14-MR2, demonstrating strong detection performance under more variable operating conditions.

Cross-room/cross-floor Evaluation. However, when a pretrained model is directly applied to a different floor, the assumption of a fixed clustering granularity becomes suboptimal due to shifts in data distribution and behavioral complexity, as illustrated in Figure 6. This motivates the use of X-means to automatically determine an appropriate number of clusters for cross-floor evaluation. As shown in Table 3, using a fixed K=3 leads to substantial performance variation across different floor pairs, indicating that behavioral complexity differs between floors. In contrast, adaptive cluster selection via X-means identifies more suitable clustering granularities for the target floor, resulting in improved or comparable AUC and F2 in most cross-floor settings, as illustrated in Figure 6c. Notably, these gains are achieved without modifying the pretrained representation model during adaptation. Overall, the results suggest that adapting the clustering stage alone is sufficient to accommodate cross-floor heterogeneity, making X-means a practical mechanism for improving generalization.

6.5 Comparison with State-of-the-Art Methods

We compare our approach with several well-established baseline approaches. Since most baseline approaches are designed for binary classification, we employed the AUC (abnormal vs. rest) metric to ensure a fair comparison with our approach. This metric evaluates abnormal detection by taking abnormal class as the positive class and treating the remaining classes as negative. Table 4 reports the AUC results when each approach is evaluated on the same floor on which the model is trained. Under this setting, the proposed gradient-based approach consistently achieves strong performance, obtaining the highest AUC on L4 (0.980) and L14-MR1 (0.910), and competitive performance on L14-MR2 (0.880). Compared to baselines such as ARCUS [21], CDAOM [19], and LSTM-ED [14], our approach demonstrates a clear advantage, indicating that gradient-based representations are more effective at separating abnormal behavior from normal operation. Although ARCUS performs well for drift detection in its original online-learning scenarios, it exhibits inferior performance on our datasets due to differences in domain characteristics and model design. ARCUS relies on an autoencoder with a fixed latent representation, making it highly sensitive to distributional shifts even when such shifts originate from normal operating conditions. Moreover, ARCUS is designed to trigger updates upon detecting drift, without distinguishing whether the drift arises from normal variation or abnormal operation. As a result, it is less effective for abnormal detection in our system.

Table 4: AUC (abnormal vs. rest) Comparison Across Datasets

Scheme	L4	L14-MR1	L14-MR2
Our approach	**0.980**	**0.910**	0.88
ARCUS (RAPP) [21]	0.479	0.579	0.588
CDAOM [19]	0.463	0.227	0.282
LSTM-ED [14]	0.528	0.583	0.573
REBM [22]	0.979	0.861	0.752
RRCF [7]	0.736	0.510	0.560
STARE [20]	0.672	0.672	0.724
ECOD [13]	0.973	0.831	0.871
COPOD [12]	0.961	0.835	**0.902**

Notably, our approach yields substantial improvements on the more challenging L14 benchmarks, with gains of +4.9% over REBM and +7.5% over ECOD on L14-MR1, and +11.6% over REBM on L14-MR2. Although COPOD attains slightly higher performance on L14-MR2 (by 2.4%), this advantage does not generalize across datasets, where our approach maintains more balanced and stable results, including on L4. These results highlight the robustness of our approach in scenarios where statistical tail-based methods such as ECOD and COPOD struggle with complex distributions, and where energy-based models like REBM are less effective under pronounced spatiotemporal variability.

Unlike these baselines, which compute anomaly scores using fixed decision rules or pretrained models without explicitly distinguishing evolving behavioral patterns or supporting selective adaptation under drift, our adaptive pipeline leverages input-gradient based representations and clustering to characterize normal operation, normal drift, and attacks, and to selectively adapt the model over time. Performance gains are most pronounced on the harder L14 benchmarks, underscoring the superior generalization of our method to realistic, multi-class anomaly scenarios compared to static scoring approaches.

Table 5: Comparison of Gradient Representation Quality between TCN-VAE, AE, and TCN

Datasets	Model	AUC ↑	F2 ↑	Recall ↑	CR ↓
L4	TCN-VAE (ours)	0.980	0.950	0.943	0.54%
	AE	0.960	0.773	0.764	2.26%
	TCN	0.950	0.872	0.886	0.19%
L14-MR1	TCN-VAE (ours)	0.760	0.701	0.752	0.29%
	AE	0.610	0.469	0.566	2.30%
	TCN	0.770	0.663	0.667	6.86%
L14-MR2	TCN-VAE (ours)	0.830	0.788	0.795	2.81%
	AE	0.610	0.437	0.522	2.13%
	TCN	0.780	0.823	0.829	3.56%

6.6 Ablation Study: Gradient Quality Analysis.

We conducted additional experiments comparing our proposed TCN-VAE with gradient-based baselines that do not use VAE regularization, namely standard Autoencoder (AE) and a TCN predictor, while keeping all configurations identical (K=3, centroid update buffer=5). The results in Table 5 show that TCN-VAE consistently achieves the strongest overall performance. For example, on L4, TCN-VAE achieves AUC = 0.980 and F2 = 0.950, compared with AUC = 0.960, F2 = 0.773 for AE, and AUC = 0.950, F2 = 0.872 for TCN. On L14-MR1 and L14-MR2, AE shows significant degradation (e.g., AUC = 0.610), while TCN also performs worse than TCN-VAE in terms of AUC. These results indicate that latent regularization in the VAE improves the stability and discriminative quality of gradient representations. While TCN captures temporal dependencies, its gradients are less effective for distinguishing operational modes without the structured latent space provided by the VAE. Overall, this comparison confirms that both temporal modeling and latent regularization are important, and the proposed TCN-VAE provides the most effective gradient representations for our task.

6.7 Hyperameters Analysis

6.7.1 Centroid update buffer sizes. Table 1 in Appendix C.1 evaluates the impact of different centroid update buffer sizes (from 5 to 20) on detection performance. The results indicate that a small buffer size of 5 batches provides the best balance between detection performance and CR. On the L4 dataset, all buffer sizes achieve the same recall and CR (0.5%), suggesting that increasing the buffer size offers no additional benefit. On the more variable L14 floors, larger buffers yield only marginal recall improvements (e.g., L14-MR1), but they also lead to a substantially higher CR of 1.3% compared to 0.3%. A similar trend is observed on L14-MR2, where larger buffers provide negligible recall improvement while slightly increasing contamination. These results indicate that a small buffer enables faster adaptation to normal drift while reducing the risk of incorporating attack samples into centroid updates. Therefore, we select buffer = 5 as the default configuration in our framework.

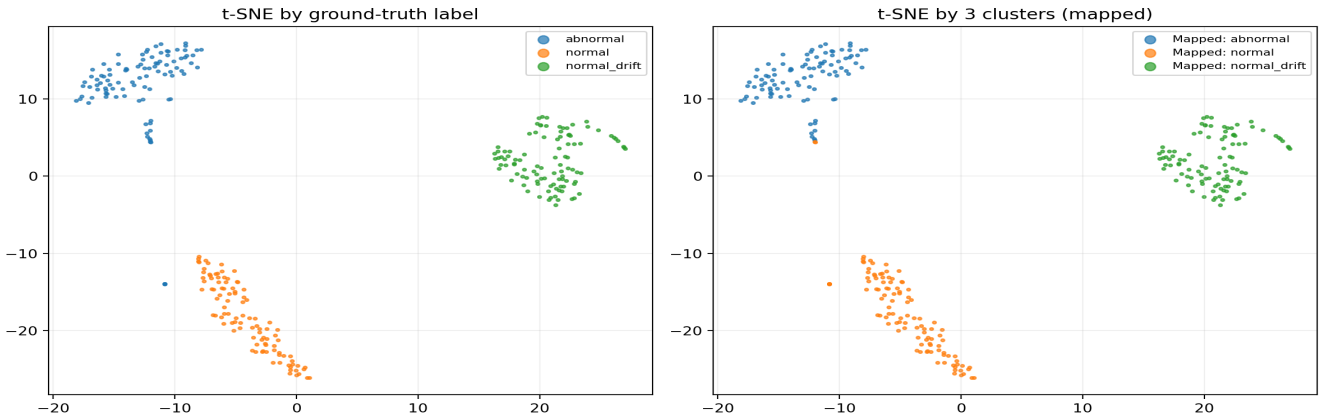
6.7.2 KL Weight (β). We evaluate the impact of the KL divergence weight by testing three values, $\beta \in 1E-3, 1E-2, 1E-1$, across multiple floors. The results in Table 2 of Appendix C.1 show a non-monotonic effect: increasing β does not consistently degrade performance but shifts the balance between reconstruction sensitivity and latent regularization. On the relatively stable and vacant L4 floor, $\beta = 1E-3$ achieves the best performance (AUC = 0.98, F2 = 0.95), indicating that weaker regularization preserves sharper reconstruction gradients. In contrast, the more variable and occupied L14 floors benefit from stronger regularization, where $\beta = 1E-1$ performs better by stabilizing gradient representations. Notably, $\beta = 1E-2$ consistently yields the weakest performance across all floors, suggesting that moderate regularization may blur gradients without sufficient latent stabilization. This task-dependent sensitivity is consistent with prior β-VAE studies [4, 8].

6.8 Limitations

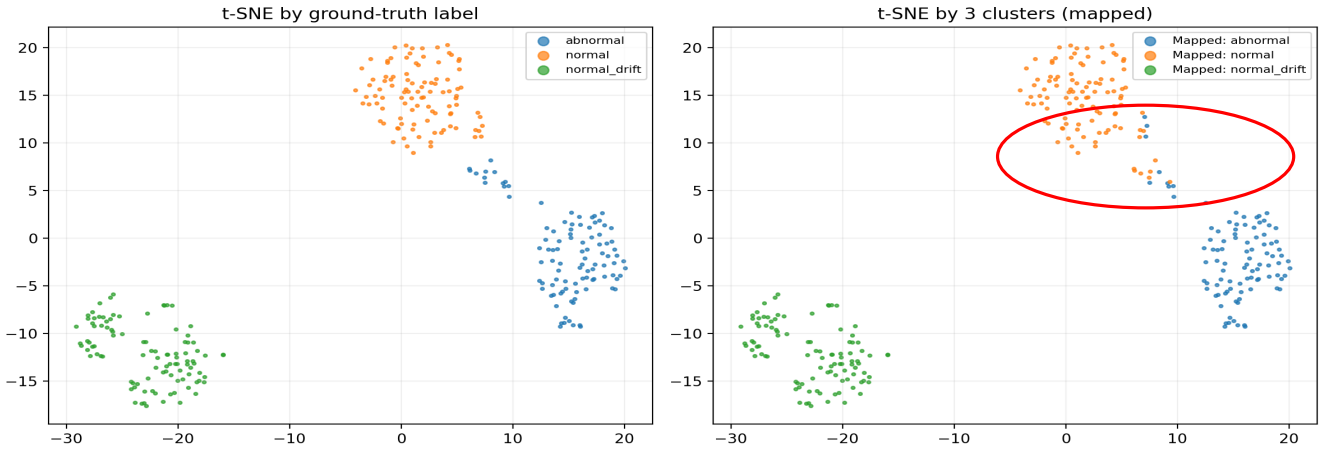
While our results demonstrate strong performance in distinguishing normal drift from attack behaviors in HVAC systems, the evaluation was conducted within a specific smart building environment. Furthermore, our dataset does not span a long enough period to capture sensor aging, whereas seasonal changes are insignificant in our geographical region. Despite this, normal drifts caused by day-to-day and time-of-day variations are indeed generally unseen after the initial training of our model. Our results show that such unseen drifts can be effectively handled during online deployment as new online data becomes available. The core idea of leveraging temporal VAE gradient profiles is general and can potentially be applied to multi-class classification tasks in other cyber-physical system domains. Future work will focus on extending the evaluation to more diverse environments with varying operational and environmental conditions.

7 Conclusion

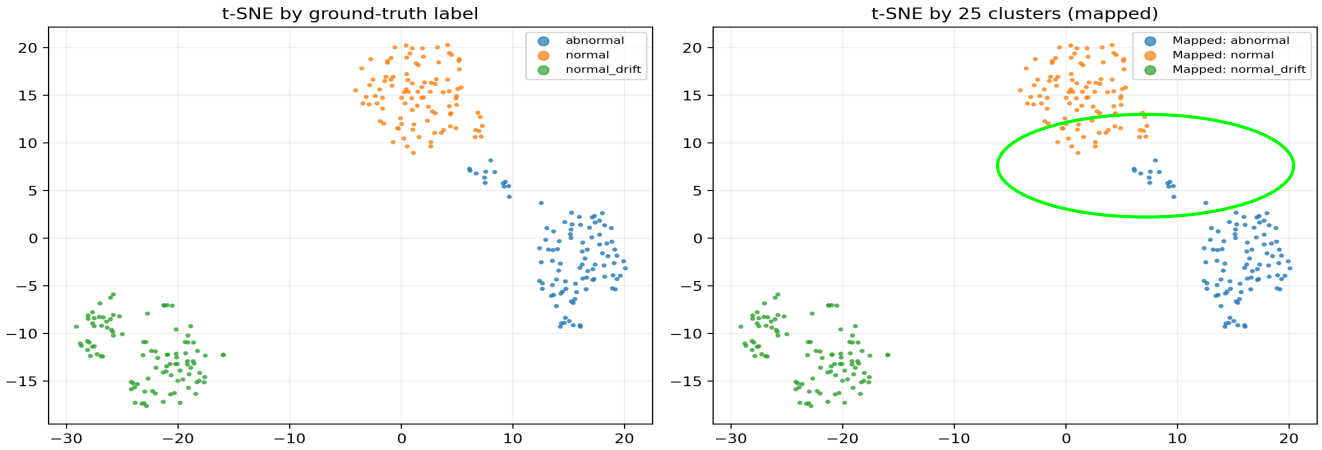
We proposed a drift-aware online AD framework for energy and building cyber-physical systems that explicitly distinguishes among normal operation, normal drift, and attack behaviors. By combining a Temporal Convolutional Network–based Variational Autoencoder with gradient-based sensitivity representations, the framework captures behavioral changes beyond reconstruction error and enables selective, contamination-free model updates under non-stationary conditions. To support cross-floor deployment, we introduced an adaptive clustering strategy that automatically adjusts clustering granularity to account for increased intra-class variability caused by distribution shifts. Experiments on real-world building management system datasets demonstrate improved robustness to normal drift, reliable anomaly classification, and low contamination rates under both same-floor and cross-floor scenarios. Overall, the proposed approach offers a practical and scalable solution for long-term AD in mission-critical energy systems.



(a) Same-floor evaluation: L4 → L4



(b) Cross-floor evaluation: L14–MR2 → L4 (3 clusters)



(c) Cross-floor evaluation: L14–MR2 → L4 (X-means, 25 clusters)

Figure 6: t-SNE visualizations of gradient profiles under same-floor and cross-floor evaluation. Samples from L4 are clearly separable under the three-cluster setting when using a model pretrained on the same floor, as shown in (a). In contrast, when a model pretrained on a different floor (L14–MR2) is evaluated on L4, the cluster boundaries become less distinct (highlighted by the red circle), as shown in (b). To address this issue, the X-means method is employed to determine the optimal number of clusters, enabling finer-grained modeling of intra-class variations near ambiguous boundaries and improving classification performance, as illustrated in (c).

Acknowledgments

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. This research is also supported in part by the National Research Foundation, Singapore, under its National Satellite of Excellence Programme "Design Science and Technology for Secure Critical Infrastructure: Phase II" (Award No: NRF-NCR25- NSOE05-0001). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Research Foundation.

References

[1] Wael Alsafery, Omer Rana, and Charith Perera. 2023. Sensing within smart buildings: A survey. *Comput. Surveys* 55, 13s (2023), 1–35.

[2] Aqeel Anwar. 2021. Difference Between Autoencoder (AE) and Variational Autoencoder (VAE). Towards Data Science. https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2/ (accessed Jan. 23, 2026).

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. In *International Conference on Learning Representations (ICLR)*.

[4] Christopher P. Burgess, Irina Higgins, Arka Pal, et al. 2018. Understanding disentangling in beta-VAE. *arXiv preprint arXiv:1804.03599* (2018).

[5] Yang Cao, Yixiao Ma, Ye Zhu, and Kai Ming Ting. 2024. Revisiting streaming anomaly detection: benchmark and evaluation. *Artificial Intelligence Review* 58, 1 (2024), 8.

[6] Daniel Fährmann, Laura Martín, Luis Sánchez, and Naser Damer. 2024. Anomaly detection in smart environments: A comprehensive survey. *IEEE access* 12 (2024), 64006–64049.

[7] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. 2016. Robust random cut forest based anomaly detection on streams. In *International conference on machine learning*. PMLR, 2712–2721.

[8] Irina Higgins, Loic Matthey, Arka Pal, et al. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations*.

[9] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[10] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[11] HyeongJun Kwon, MyungHee Kim, and Juhyun Kim. 2020. Backpropagated Gradient Representations for Anomaly Detection. In *European Conference on Computer Vision (ECCV)*. 553–569. https://arxiv.org/abs/2007.09507

[12] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. 2020. COPOD: Copula-Based Outlier Detection. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1118–1123. doi:10.1109/icdm50108.2020.00135

[13] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George H. Chen. 2023. ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions. *IEEE Transactions on Knowledge and Data Engineering* 35, 12 (2023). doi:10.1109/tkde.2022.3159580

[14] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148* (2016).

[15] Quoc Phong Nguyen, Kar Wai Lim, Dinil Mon Divakaran, Kian Hsiang Low, and Mun Choon Chan. 2019. GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection. *CoRR* abs/1903.06661 (2019). arXiv:1903.06661 http://arxiv.org/abs/1903.06661

[16] Dan Pelleg, Andrew W Moore, et al. 2000. X-means: Extending k-means with efficient estimation of the number of clusters.. In *Icml*, Vol. 1. Stanford, CA, 727–734.

[17] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv:1312.6034 [cs.CV] https://arxiv.org/abs/1312.6034

[18] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks *(ICML'17)*. JMLR.org, 3319–3328.

[19] Hongda Tian, Nguyen Lu Dang Khoa, Ali Anaissi, Yang Wang, and Fang Chen. 2019. Concept drift adaption for online anomaly detection in structural health monitoring. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2813–2821.

[20] Susik Yoon, Jae-Gil Lee, and Byung Suk Lee. 2020. Ultrafast local outlier detection from a data stream with stationary region skipping. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1181–1191.

[21] Susik Yoon, Youngjun Lee, Jae-Gil Lee, and Byung Suk Lee. 2022. Adaptive model pooling for online deep anomaly detection from a complex evolving data stream. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 2347–2357.

[22] Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Zhang. 2016. Deep structured energy based models for anomaly detection. In *International conference on machine learning*. PMLR, 1100–1109.