Systems & Control Transactions

Original Research Article
Peer Reviewed Conference Proceeding
ESCAPE 36 - European Symposium on Computer Aided Process Engineering
Sheffield, UK. 21-24 June 2026

PSE PRESS

A Unified Multi-Scale TCN Framework for Batch Manufacturing Soft Sensing and Monitoring

Yee Hung Hong[a], Zhao Jinsong*

[a] Tsinghua University, Department of Chemical engineering, Beijing, China
* Corresponding Author: jinsongzhao@mail.tsinghua.edu.cn.

ABSTRACT

Batch manufacturing is central to fine chemicals, pharmaceuticals, and bioprocessing. Its operation evolves across phases and recipes, which yields high-dimensional trajectories and strong batch-to-batch variability. Meanwhile, key quality-indicative variables are often measured offline and cannot be used as online model inputs. This work presents an integrated deep learning framework that unifies soft sensing and process monitoring in a single module using only process variables as inputs. A multi-scale Temporal Convolutional Network with multiple kernel sizes extracts complementary dynamic features from sliding windows. These features are concatenated and pooled into a compact representation that feeds two task branches. A variational autoencoder branch reconstructs the input window and provides fault monitoring signals via reconstruction deviation while regularizing the latent space through KL divergence. In parallel, a prediction branch estimates the quality-indicative variable directly from the pooled temporal features without using the variational latent sample. This separation preserves a stable quality mapping while retaining a probabilistic reconstruction model for monitoring. During inference, reconstruction error and prediction error are fused into a joint state score that more comprehensively reflects system state changes than either deviation alone. Diagnostic heatmaps are produced from residual maps and optional SHAP attributions to highlight contributing variables and time segments. The framework is validated on the Industrial Penicillin Simulation, an industrial-scale penicillin fermentation benchmark. Results show stable convergence of reconstruction, prediction, and KL terms, clear fault-set monitoring trajectories, and interpretable heatmaps that support actionable diagnosis.

Keywords: Batch Process, Fermentation, Artificial Intelligence, Process Monitoring, Machine Learning

INTRODUCTION

Modern batch manufacturing increasingly operates in an environment that is both data-rich and information-poor. Hundreds of process variables are measured online, but the variables that truly define economic value and product quality are often unavailable in real time. Many quality-indicative variables require laboratory assays, which are expensive and delayed. Operators therefore face a practical dilemma: the plant generates high-frequency trajectories, yet the most critical quality information arrives late. In this work, we use the term **quality-indicative variable (QIV)** to denote a product-quality-related variable that is economically important but typically unavailable online and only measured through delayed laboratory assays.

This reality is particularly severe in dynamic batch systems. A batch is not a stationary time series. It is a sequence of phases with changing kinetics, changing control objectives, and changing degrees of freedom. Recipes evolve. Controllers are tuned. Disturbances occur. These factors produce distribution shifts across batches and time, which is now recognized as a central scientific and engineering challenge in process monitoring [1].

Two tasks are indispensable for safe and profitable operation. The first task is soft sensing, which aims to estimate quality-indicative variables from process measurements. Deep learning has become a major driver for soft sensors because it can model nonlinear and dynamic relationships and it can exploit large-scale historical data [2]. The second task is fault monitoring, which aims to

detect abnormal states early, before they propagate into off-spec product or unsafe operation. Variational autoencoders provide a principled approach to fault monitoring because reconstruction deviation can indicate departures from a learned normal operating manifold [4].

However, in many practical deployments, soft sensing and monitoring are still implemented as separate modules. This separation is convenient but it is scientifically limiting in evolving batch environments. It forces independent feature extraction mechanisms, reduces feature efficiency, and increases maintenance cost under shift. Life-cycle studies in industrial fermentation emphasize that drift handling and model maintenance become bottlenecks once a soft sensor is deployed [3]. At the same time, recent ESCAPE-related work on IndPenSim demonstrates that even predictive monitoring frameworks must address frequent operational shifts and high retraining demand, which motivates architectures that can reuse learned temporal representations [18].

This paper addresses a core scientific question:

How can we build a unified architecture that learns a shared temporal representation for batch trajectories, performs online soft sensing and process monitoring simultaneously, does not require online quality variables as inputs, and yields interpretable diagnostic outputs that operators can trust?

Bearing the core scientific question in mind, we propose this work which makes four contributions.

1. It formulates joint soft sensing and monitoring as a unified representation learning problem under the industrial constraint that QIV is not an online input stream.
2. It proposes a multi-scale multi-kernel TCN backbone that extracts complementary temporal patterns from batch windows and supports two monitoring tasks through a shared feature space.
3. It designs a dual-branch head in which a VAE branch provides probabilistic reconstruction learning with KL divergence for monitoring, while a separate prediction branch provides a stable quality mapping that does not depend on latent sampling.
4. It introduces a joint state score that fuses reconstruction deviation and prediction deviation to better capture system state changes, and it supports diagnosis with residual heatmaps and optional SHAP explanations [12].

RELATED WORK

Batch Process Monitoring under Non-stationary and Operational Diversity

Batch processes exhibit multi-phase dynamics and recipe dependent policies. These properties yield strong non-stationarity. They also yield batch to batch variability. In such settings, monitoring must remain reliable when the operating distribution shifts. Zhao discussed nonstationary monitoring in the era of industrial AI and summarized how distribution shift challenges classical condition monitoring assumptions [2].

Non-stationarity is amplified by practical realities. Operating strategies evolve, controllers are retuned, raw materials vary, equipment ages, and so on. These factors create gradual drift and abrupt change. Monitoring systems must therefore emphasize robustness and maintainability. A practical monitoring framework should reuse representations and reduce fragmentation. This is especially important for batch operations where each run is a finite trajectory that must be interpreted in context.

Soft Sensing with Scarce and Delayed Quality Measurements

Soft sensing is essential when quality is hard to measure online. Deep learning soft sensors can learn nonlinear mappings from multivariate trajectories to quality outcomes. **Sun et al.** surveyed deep learning for soft sensors and emphasized feature learning for complex dynamics [1].

A persistent barrier is label scarcity. Quality labels can be sparse and delayed. In many plants, QIV is not available as an online input stream. Semi supervised learning provides one route. It allows the model to learn from abundant process trajectories while using QIV only as supervision when it exists. **Lee et al.** developed a semi supervised latent dynamic variational framework to improve quality prediction under limited quality labels [7].

Beyond temporal modeling, some works model variable relations explicitly. Graph convolutional soft sensors embed interactions among variables to improve prediction [8]. Temporal graph approaches integrate temporal dependence with structured relationships and add transparency modules for interpretability [9]. Transfer and adversarial learning have also been studied to improve generalization across grades and modes when operating conditions change [10]. These directions highlight the need for representations that remain useful across shifting batch environments.

Fault Detection and Monitoring with Reconstruction based Generative Models

Reconstruction based monitoring is widely used in process monitoring. It learns normal behavior and detects deviation through residuals. Variational autoencoders provide a principled generative formulation. They pair a reconstruction term with KL divergence regularization of the latent distribution. Kingma and Welling introduced the core variational framework and the reparameterization method that enables stable training.

VAE based monitoring has been extended in

chemical processes. Interpretable variants embed attention to support fault identification and variable contribution analysis. **Bi et al.** proposed an orthogonal self-attentive VAE for interpretable fault detection and identification [4]. Distributed VAE monitoring has also been proposed for large scale systems and variable group structures [5].

A limitation of purely reconstructive monitoring is that reconstruction deviation does not always align with quality relevance. A deviation may be detectable in variables yet have limited effect on quality. A quality critical deviation may also be subtle in measured variables. This motivates quality related monitoring and combined monitoring statistics. **Tang et al.** proposed a nonlinear quality related method that combines variational information bottleneck ideas with VAE learning [6]. This line of work supports monitoring metrics that reflect both operating deviation and quality impact.

Unified Learning across Soft Sensing and Monitoring

Soft sensing and monitoring are often implemented separately. Yet both tasks aim to infer a latent process state from trajectories. Separate modeling duplicates feature extraction and increases lifecycle burden. It also creates inconsistencies across modules when the operating distribution shifts.

Lifecycle and deployment studies in industrial fermentation highlight this problem. **Metcalfe et al.** discussed MLOps oriented soft sensors and showed that drift handling and maintenance can dominate the cost of deployment [3]. This motivates unified architectures that share temporal representations and reduce retraining effort.

Recent work on IndPenSim also reflects community interest in transferable monitoring. **Hung HY et al.** reported predictive monitoring enhanced by transfer learning on IndPenSim [13]. This underscores the importance of representation reuse under operational shift. A remaining gap is a unified architecture that performs both online quality estimation and monitoring while respecting the constraint of no online quality inputs.

Explainability and Actionable Diagnosis

In industrial operation, an alarm alone is not sufficient. Operators require diagnosis that supports corrective actions. Explainability is therefore critical.

SHapley Additive exPlanations (SHAP) provides an additive attribution framework grounded in Shapley values from cooperative game theory. It offers a consistent approach for interpreting model outputs through feature contributions. Explainable diagnosis in chemical processes increasingly combines data driven learning with domain structure. **Wu et al.** proposed a topology aware diagnosis model that integrates process topology knowledge with self attention mechanisms [11]. Comprehensive fault detection and diagnosis studies continue to highlight interpretability as a key requirement for industrial adoption.

Positioning of this work: This paper proposes a unified batch monitoring module. It uses process variables only as inputs. It learns one shared temporal representation. It supports soft sensing and monitoring simultaneously. It provides interpretable diagnosis artifacts that can be used in practice.

PRELIMINARIES

Variable Screening with Lagged Mutual Information and Redundancy Control

Batch trajectories are often high dimensional. Many variables can be weakly related to quality. Many variables can also be redundant. Feeding all variables into a deep model can increase noise sensitivity and reduce interpretability. We therefore perform variable screening using nominal batches only.

We quantify nonlinear relevance to QIV using lagged mutual information. For each candidate variable x_i, we compute mutual information between $x_i(t-k)$ and $y(t)$ for lags $k = 0, \dots, K$. We take the maximum over lags as the relevance score. We also record the lag that achieves the maximum. This captures delayed effects of process variables on quality.

We control redundancy using an mRMR style greedy selection. Each candidate is penalized by its redundancy with already selected variables. Redundancy is approximated through correlation. The procedure yields a compact variable subset that preserves relevance and reduces collinearity.

Pseudocode is available in Algorithm 1.

Temporal Convolutional Networks for Causal Multi Scale Dynamics

Batch monitoring requires models that can learn long range temporal dependence. Training must remain stable. We adopt Temporal Convolutional Networks based on causal dilated convolutions and residual blocks. Causal convolution enforces that the representation at time t depends only on present and past samples within a window. Dilation expands the receptive field without recurrent computation. Residual connections stabilize optimization. We implement a multi scale TCN using parallel branches with different kernel sizes. Each branch captures temporal patterns at a different scale. Branch outputs are concatenated and pooled. This yields a compact shared temporal representation that is used by both downstream tasks.

Inputs: Nominal batches $D_0 = \{(X, y) \mid Fault = 0\}$, Candidate variables $X = \{x_1, \ldots, x_d\}$, QIV y, Maximum lag K, Redundancy weight λ Number of selected variables k.

Outputs: Selected variable set S, Relevance scores $Rel(x_i)$ and optimal lags k_i^*.

Step 1: Compute lagged MI relevance

1. for each $x_i \in X$ do
2. for $k = 0$ to K do
3. Construct paired samples $\{(x_i(t-k), y(t))\}_t$ using nominal batches D_0
4. $I_{i,k} \leftarrow MI(x_i(t-k), y(t))$
5. end for
6. $Rel(x_i) \leftarrow max_{k\in[0,K]} I_{i,k}$
7. $k_i^* \leftarrow argmax_{k\in[0,K]} I_{i,k}$
8. end for

Step 2: Greedy mRMR-style selection

9. $S \leftarrow \emptyset$
10. while $|S| < k$ *do*
11. for each $x_i \in X \backslash S$ *do*
12. if $S = \emptyset$ *then* $R_i \leftarrow 0$
13. else
14. $R_i \leftarrow \frac{1}{|S|} \sum_{x_j \in S} |corr(x_i, x_j)|$
15. end if
16. $Score(x_i) \leftarrow Rel(x_i) - \lambda R_i$
17. end for
18. $x^* \leftarrow argmax_{x_i \in X\backslash S} Score(x_i)$
19. $S \leftarrow S \cup \{x^*\}$
20. end while
21. *return* $S, \{Rel(x_i), k_i^*\}$

Algorithm 1: Variable Screening with Lagged Mutual Information and mRMR-style selection

Semi Supervised Training with Sparse Quality Labels

In realistic plants, QIV labels are sparse and delayed. QIV is not used as an online input channel. It is used only as a training label when it exists. We adopt a semi supervised training protocol.

A reconstruction objective is trained on nominal windows. It uses only process variables. This objective is always available. A prediction objective is trained only when a QIV label exists. We use a label mask $m_{b,t} \in \{0, 1\}$ so that unlabeled windows do not contribute to the prediction loss. This enables joint training on mixed labeled and unlabeled data.

PROBLEM FORMULATION

Let $x_{b,t} \in \mathbb{R}^D$ denote process variables at time t for batch b. We use a sliding window

$$X_{b,t} = [x_{b,t-W+1}, \ldots, x_{b,t}] \in \mathbb{R}^{W\times D}$$

Let $y_{b,t}$ denote a quality indicative variable aligned with the window end. In the target setting, $y_{b,t}$ is sparse and delayed. It is not an online input channel.

We aim to build one module that:

1 takes $X_{b,t}$ as input.
2 outputs a reconstructed window $\hat{X}_{b,t}$ for monitoring.
3 outputs a quality estimate $\hat{y}_{b,t}$ for soft sensing.
4 computes a joint state score by fusing monitoring deviation and quality deviation.
5 supports interpretable diagnosis through heatmaps and optional SHAP attributions.

METHODOLOGY

Architecture Overview

The framework consists of a shared temporal backbone and two task branches.

1. Shared backbone. A multi scale TCN uses multiple kernel sizes in parallel.

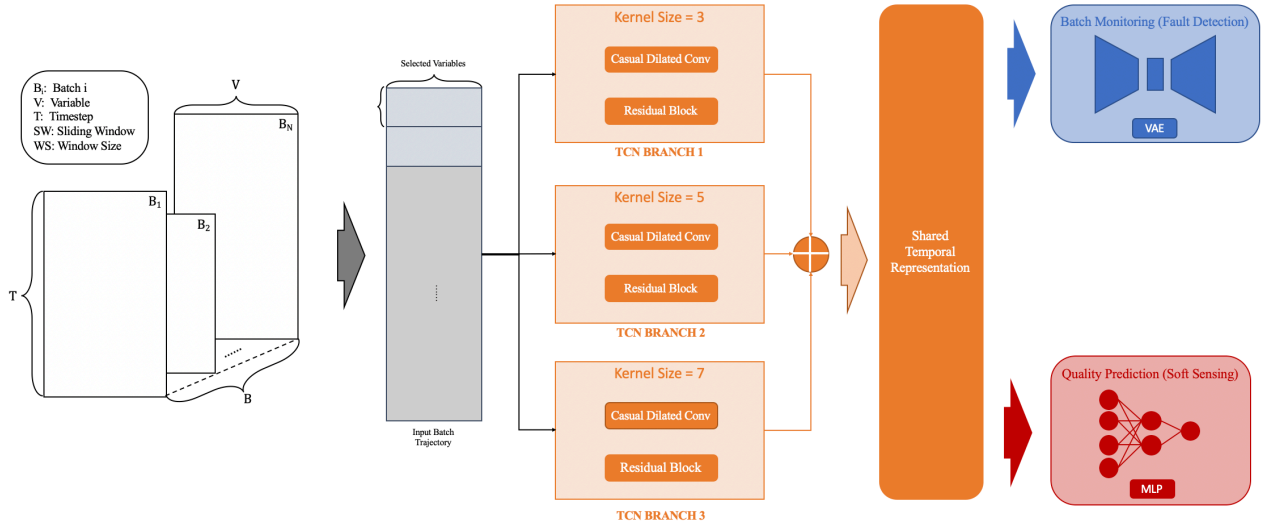

Figure 1. Multi scale multi kernel TCN backbone with two branches

2. Monitoring branch. A VAE reconstructs the input window. Monitoring is based on reconstruction deviation.
3. Soft sensing branch. A predictor estimates QIV from pooled temporal features. It does not use the latent sample.

This design enforces controlled sharing. Both tasks reuse temporal features. The predictor avoids stochastic sampling to preserve stable quality mapping.

Multi Scale TCN Backbone with Multiple Kernel Sizes

Batch trajectories contain patterns at different time scales. Early phases can exhibit fast transients. Later phases can exhibit slow drifts. A single kernel size can under represent this diversity. We therefore build parallel TCN branches with kernel sizes $k_1, k_2, \ldots \ldots$.

Each branch uses causal dilated convolutions. Dilation increases receptive field efficiently. Residual connections improve stability for deep temporal stacks. Let $f_{b,t}^{(k)}$ denote features from kernel branch k. We concatenate:

$$F_{b,t} = Concat(f_{b,t}^{(k_1)}, f_{b,t}^{(k_2)}, \ldots \ldots)$$

We pool across time to obtain compact shared features:

$$g_{b,t} = Pool(F_{b,t})$$

VAE Monitoring Branch with KL Divergence Regularization

The VAE branch maps $g_{b,t}$ into a latent distribution $q(z_{b,t} \mid g_{b,t})$ with parameters $\mu_{b,t}$ and $\sigma_{b,t}$. We sample using reparameterization:

$$z_{b,t} = \mu_{b,t} + \sigma_{b,t} \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$

A decoder reconstructs the window and outputs $\hat{X}_{b,t}$ Reconstruction loss:

$$\mathcal{L}_{rec} = MSE(X_{b,t}, \hat{X}_{b,t})$$

KL divergence:

$$\mathcal{L}_{KL} = D_{KL}(q(z_{b,t} \mid g_{b,t}) \,||\, \mathcal{N}(0, I))$$

Monitoring deviation:

$$e_{rec} = MSE(X_{b,t}, \hat{X}_{b,t})$$

Prediction Branch for Soft Sensing without Latent Sampling

The prediction branch estimates QIV from pooled temporal features:

$$\hat{y}_{b,t} = f_{pred}(g_{b,t})$$

The prediction branch does not take $z_{b,t}$ as input. This avoids sampling noise in the quality mapping. QIV is not an input stream. It is only used as a label when available.

We use a robust prediction loss:

$$\mathcal{L}_{pred} = m_{b,t} \bullet Huber(y_{b,t}, \hat{y}_{b,t})$$

where $m_{b,t}$ is the label mask defined in the Semi-Supervised Training section.

Joint Multi Task Objective

The overall objective is:

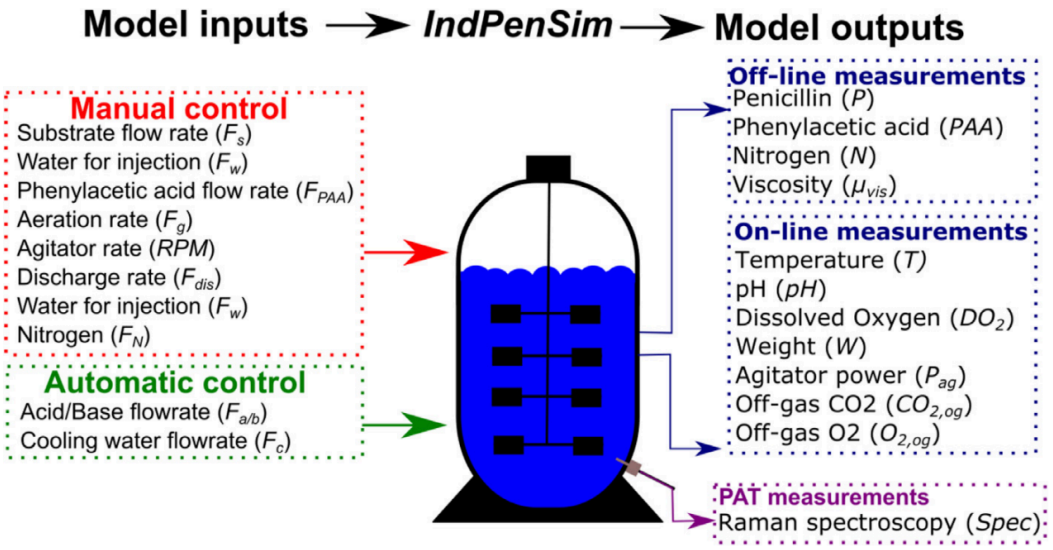

Figure 2. Summary of Inputs and Outputs of IndPenSim

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{pred}\mathcal{L}_{pred} + \beta\mathcal{L}_{KL}$$

This objective forces the backbone to learn shared temporal features. It also maintains a regularized latent space for stable reconstruction learning.

Joint State Score by Fusing Reconstruction and Prediction Deviation

At inference, the model provides reconstruction deviation:

$$e_{rec}(t) = MSE(X_{b,t}, \hat{X}_{b,t})$$

The model also provides a quality estimate $\hat{y}_{b,t}$. When a lab assay arrives, prediction deviation can be computed:

$$e_{pred}(t) = |y_{b,t} - \hat{y}_{b,t}|$$

We define a joint state score:

$$S(t) = \alpha \bullet norm\left(e_{rec}(t)\right) + (1-\alpha) \bullet norm\left(e_{pred}(t)\right)$$

When $y_{b,t}$ is not yet available, the system runs in early warning mode with $S(t) = norm\left(e_{rec}(t)\right)$. This supports online monitoring under delayed quality measurement.

Interpretable Inference with SHAP Heatmaps

For diagnosis, we compute the residual map:

$$R_{b,t} = (X_{b,t} - \hat{X}_{b,t})^2$$

We visualize $R_{b,t}$ as a variable time heatmap for alarm windows. This highlights which variables and which time segments contribute most to reconstruction deviation.

If attribution based interpretation is required, we apply SHAP to the prediction output or to the joint score. This yields variable contribution values that support human interpretation.

CASE STUDY: INDPENSIM BATCH PENICILLIN FERMENTATION

We validate the framework on IndPenSim. IndPenSim simulates an industrial scale fed batch penicillin fermentation process and includes realistic variability, disturbances, and sensor noise.[16] It is widely used as a benchmark for monitoring and soft sensing studies. Unlike the traditional PenSim benchmark [15], IndPenSim models a 100, 000 L bioreactor, bringing it closer to real-world industrial conditions and validated by historical industrial data. Detailed process descriptions and operational specifics can be found at the IndPenSim website and in the original publications [14]. **Figure 2** illustrates the summary of IndPenSim.

Recent ESCAPE community work also used IndPenSim to study predictive monitoring with transfer learning under changing control strategies [13]. This supports the view that transferable temporal representations are important. Our focus differs in one key constraint. We enforce that QIV is not an online input. We unify soft sensing and monitoring within one module and fuse reconstruction and prediction deviations for system state tracking.

RESULTS AND DISCUSSION

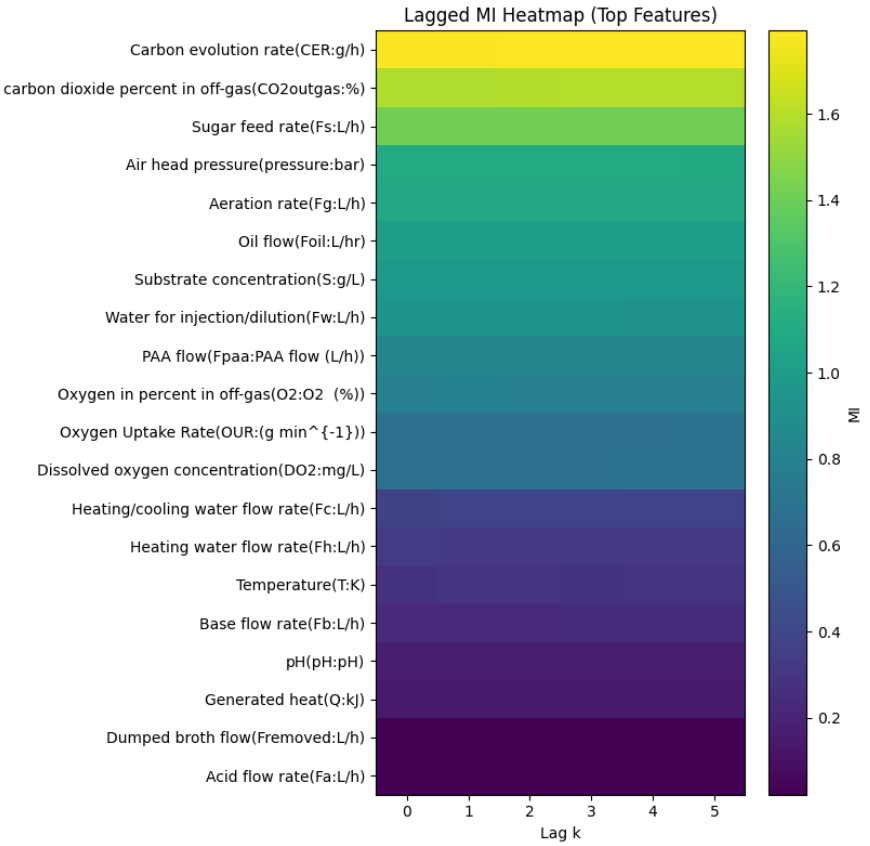

Figure 3. Lagged MI heatmap of top variables

Variable Screening Results

To reduce input dimensionality while retaining quality relevance, we performed variable screening using **nominal batches only**. For each candidate variable x_i, we computed **lagged mutual information (MI)** between $x_i(t-k)$ and the quality-indicative variable $y(t)$ over lags $k = 0, \dots, 5$. The **maximum** MI across lags was used as the relevance score $Rel(x_i) = max_k I(x_i(t-k), y(t))$, and the corresponding **optimal lag** k_i^* was recorded.

Figure 3 above visualizes the lag-dependent MI patterns for the most informative variables. Clear delayed effects can be observed: several key variables achieve their peak relevance at $k^* = 4$~5, including **CER** ($k^* = 4$), **CO**_2 in off-gas ($k^* = 4$), **substrate concentration** ($k^* = 5$), **OUR** ($k^* = 5$), and **temperature** ($k^* = 5$). This is consistent with fed-batch fermentation where operational actions (e.g., feeding/aeration) and physiological responses propagate to quality with a time delay.

Figure 4 summarizes the **max-MI ranking**, where variables related to **metabolic activity and gas exchange** (CER and off-gas CO_2) dominate the relevance ranking, followed by **feeding** and **aeration/pressure** variables. These variables are physically meaningful: gas-out measurements and CER reflect respiratory activity and carbon balance, which are closely tied to the process state and product formation, while feed and aeration govern substrate availability and oxygen transfer.

Because high-dimensional trajectories often contain strong collinearity, we further applied an **mRMR-style greedy selection** to control redundancy. Each candidate variable was penalized by its average absolute correlation with already selected variables, and the selection score was computed as

$$Score(x_i) = Rel(x_i) - \lambda Red(x_i)$$

This step prevents the final input set from being dominated by highly correlated variables. For example, although **CO**_2 in off-gas has the second highest max-MI, it is selected after **CER** and **sugar feed rate**, reflecting redundancy suppression against already selected strong predictors.

The final selected subset contains **18 variables**, reported in **Table 1**, which are used as the input channels for the subsequent multi-kernel TCN backbone. Overall, the screening results indicate that the quality-relevant information is concentrated in variables representing **respiration/gas exchange**, **feeding**, and **oxygen-transfer-related operations**, while low-MI variables (e.g., control

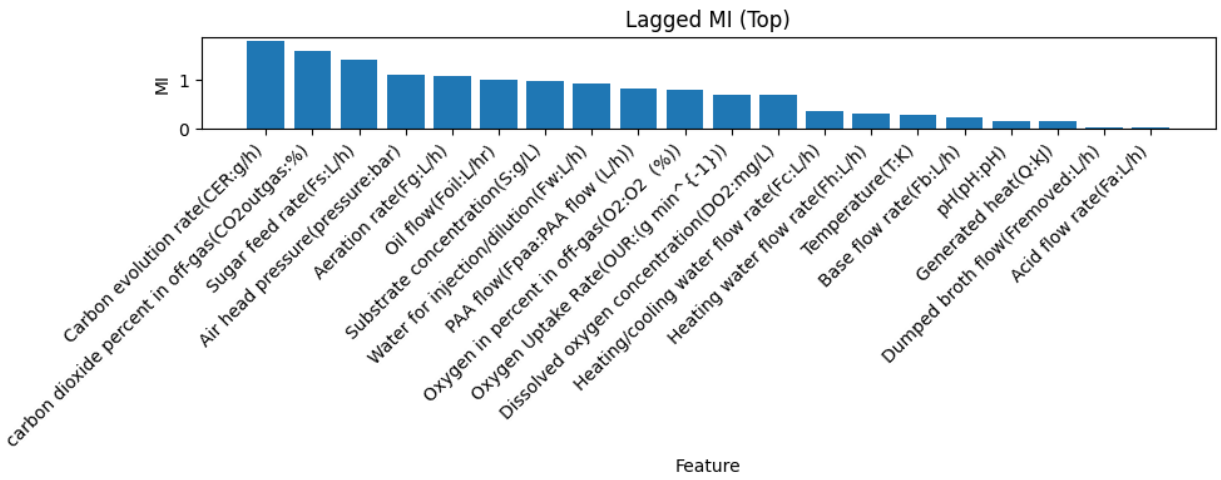

Figure 4. Max lagged MI ranking of candidate variables

Table 1. Selected variables from lagged MI screening and redundancy-controlled selection (k=18)

Rank	Variable	Max lagged MI	Best lag k^*	mRMR score
1	Carbon evolution rate(CER:g/h)	1.793	4	1.793
2	Sugar feed rate(Fs:L/h)	1.417	0	1.343
3	carbon dioxide percent in off-gas(CO2outgas:%)	1.597	4	1.312
4	Air head pressure(pressure:bar)	1.107	0	0.962
5	Aeration rate(Fg:L/h)	1.091	1	0.889
6	Substrate concentration(S:g/L)	0.982	5	0.875
7	Water for injection/dilution(Fw:L/h)	0.940	0	0.854
8	Oil flow(Foil:L/hr)	1.017	1	0.804
9	Oxygen in percent in off-gas(O2:O2 (%))	0.797	1	0.701
10	PAA flow(Fpaa:PAA flow (L/h))	0.829	3	0.647
11	Dissolved oxygen concentration(DO2:mg/L)	0.694	4	0.648
12	Oxygen Uptake Rate(OUR:(g min^{-1}))	0.695	5	0.520
13	Heating water flow rate(Fh:L/h)	0.326	0	0.274
14	Heating/cooling water flow rate(Fc:L/h)	0.376	4	0.263
15	Temperature(T:K)	0.288	5	0.231
16	Base flow rate(Fb:L/h)	0.244	4	0.181
17	pH(pH:pH)	0.174	0	0.153
18	Generated heat(Q:kJ)	0.150	1	0.128

reference flags) are naturally deprioritized.

Baseline Comparison with the Proposed Framework

This section compares the proposed unified predictive monitoring framework with two recurrent baselines. The baselines are **Baseline-1 (LSTM)** and **Baseline-2 (GRU)**, both following the same predictive monitoring pipeline with a fixed alarm threshold. Here, **LSTM** denotes **Long Short-Term Memory**, and **GRU** denotes **Gated Recurrent Unit**. Both are recurrent neural network architectures for sequential modeling, while GRU uses a simpler gating mechanism with fewer parameters than LSTM.

The proposed method combines a shared temporal representation with a reconstruction-based monitoring

branch and a deterministic quality prediction branch, and triggers alarms using a unified monitoring score.

To ensure a fair comparison, all methods were trained and evaluated under an identical experimental specification. The dataset was split into 80%/10%/10% train/validation/test batches. All models used sliding windows of length $W = 10$ with stride 1. The same feature screening procedure was applied: lagged mutual information was computed up to max lag $K = 5$, followed by redundancy control with weight $\lambda = 0.5$, and the **top** $k = 18$ variables were retained as input channels for every model.

All networks were optimized with **Adam** using a learning rate of 1×10^{-3} and weight decay 1×10^{-4}, with batch size 64 and up to 50 epochs. Early stopping was applied with patience = 6 on the validation set. Input variables were scaled using robust scaling (clipping at 5.0), and the target was scaled during training (scale_y = True). For the joint learning objective, we used the same loss weights across runs:$\lambda_{rec} = 1.0, \lambda_{pred} = 1.0$, and $\lambda_{pred} = 0.5$. During inference, the alarm threshold was computed consistently as the 0.99 quantile of nominal scores for all methods. We note that this choice is not claimed to be a universal or golden standard. Rather, it is a practical empirical calibration rule that uses the upper tail of the nominal score distribution to control false alarms. In this study, we selected a high nominal quantile because batch trajectories contain strong phase evolution and transient variability, and a conservative threshold helps suppress nuisance alarms during nominal operation. More importantly, using the same thresholding principle for all methods ensures a fair comparison, so that the observed differences reflect the quality of the learned monitoring statistic rather than method-specific threshold tuning.

We evaluated all methods on eight fault batches generated by simulator. For each batch, the result sheet reports the fault interval $[t_{fault\_start}, t_{fault\_end}]$, the first alarm time $t_{alarm,}$ (FAT, the first time index where the score exceeds the threshold), and the detection delay $\Delta t = t_{alarm,} - t_{fault\_start}$. If a method never crosses the threshold during the entire batch, then $t_{alarm,}$ is undefined and the table records "-", which also makes Δt undefined.

According to **Table 2**, across the eight fault batches, the LSTM baseline does not produce valid alarms and therefore has 0/8 detection. At the same time, it generates frequent nuisance alarms before the fault (average 19.75 pre-fault false alarms), which indicates that its monitoring statistic is strongly affected by early-batch transients or phase evolution. The GRU baseline achieves 8/8 detection, but the detection delay is highly variable, with a large worst case delay (maximum 631). In contrast, the proposed framework achieves 8/8 detection with zero pre-fault false alarms, and yields substantially smaller delays (mean 17.5, median 6.5, maximum 54). Overall, the proposed method improves both reliability (consistent detection) and timeliness (lower delay) while maintaining clean monitoring behavior.

Table 2. Overall comparison

Method	Detection rate	Mean delay	Median delay	Max delay	Avg. false alarms (pre-fault)
Baseline-1 (LSTM)	0/8 (0%)	–	–	–	19.75
Baseline-2 (GRU)	8/8 (100%)	99.75	29.0	631	0.125
Proposed	8/8 (100%)	17.5	6.5	54	0.0

The stark difference between GRU and LSTM suggests that, under the present data scale and training budget, the simpler recurrent parameterization of GRU is easier to optimize than LSTM. GRU uses a lighter gating structure and may therefore adapt more effectively to the nonstationary batch trajectories and limited fault examples considered here. By contrast, the LSTM monitoring score appears to be more strongly affected by early-batch transients and phase evolution, which leads to poor separation between nominal and faulty score distributions. This interpretation is consistent with the high pre-fault false alarm count of LSTM (19.75 on average) and its failure to produce any valid alarm in the actual fault intervals.

Table 3. Batch Wise Result

batch_id	fault_start_t	fault_end_t	LSTM FAT	LSTM delay	GRU FAT	GRU delay	Proposed FAT	Proposed delay
1	379	459	–	–	407	28	386	7
2	349	449	–	–	417	68	403	54
3	499	549	–	–	503	4	503	4
4	379	459	–	–	409	30	387	8
5	379	459	–	–	409	30	385	6
6	499	549	–	–	503	4	504	5
7	499	549	–	–	502	3	501	2
8	349	449	–	–	980	631	403	54

The batch-wise results clarify where the baselines fail. For the LSTM baseline, the first alarm time and delay are missing ("-") because the score never exceeds the threshold; hence, detection cannot be declared. This is not a reporting issue but a direct consequence of the

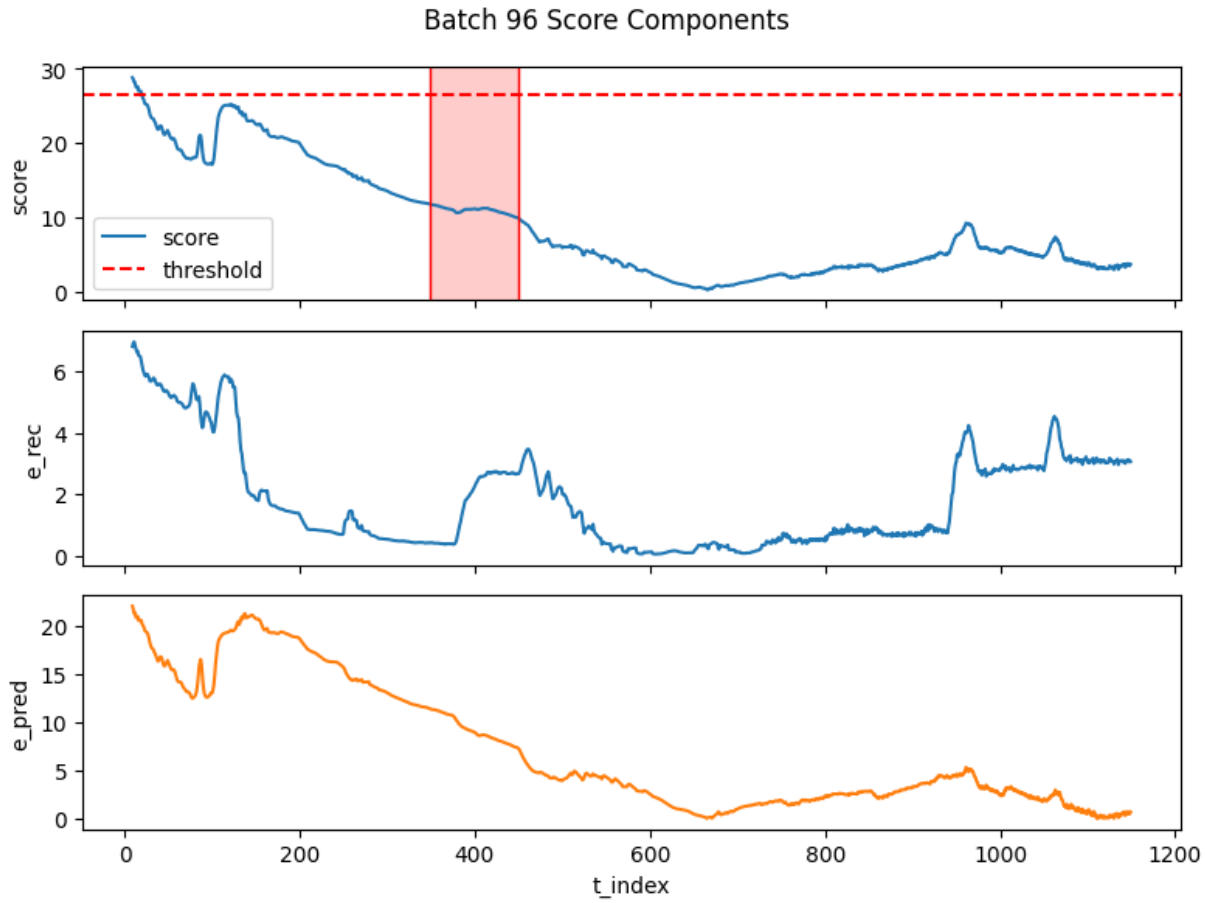

Figure 5. Score components for Batch 8 under the LSTM baseline. The shaded region denotes the fault interval and the dashed line denotes the fixed alarm threshold.

alarm rule. Batch 8 provides a representative example and is used to illustrate the mechanism behind these missing entries.

Table 4. Fault Batch 8 case study result

Batch ID	Fault Type	FDS	FAR	EDT
8	Coolant-Flowrate Fault	1	0	54

Figure 5 shows the monitoring score and its components for Batch 8. The score (top) stays below the fixed threshold throughout the highlighted fault interval, so the LSTM baseline never triggers an alarm; therefore t_{alarm} and the detection delay are recorded as "-". This indicates that the LSTM score is strongly influenced by early-stage variability and phase transitions, which raises the calibrated threshold and makes mid-batch faults difficult to detect. Therefore, the underperformance of LSTM in this study is not merely a consequence of one difficult batch, but reflects a systematic calibration issue: nominal early-phase variability inflates the threshold-relevant score distribution and reduces sensitivity to mid-batch faults.

The reduced delay of the proposed framework can be attributed to two design properties. First, the multi-scale TCN backbone extracts temporal patterns at different receptive fields, which is beneficial for batch processes where short transients and slow drifts coexist. Second, the proposed monitoring statistic fuses reconstruction deviation with prediction deviation, so the model can respond not only to trajectory abnormality but also to quality-relevant inconsistency. This dual evidence appears particularly useful in difficult cases such as Batch 8, where a single recurrent monitoring pathway may detect the fault only after a substantial lag.

Overall, under identical training and inference settings, the proposed framework delivers more actionable

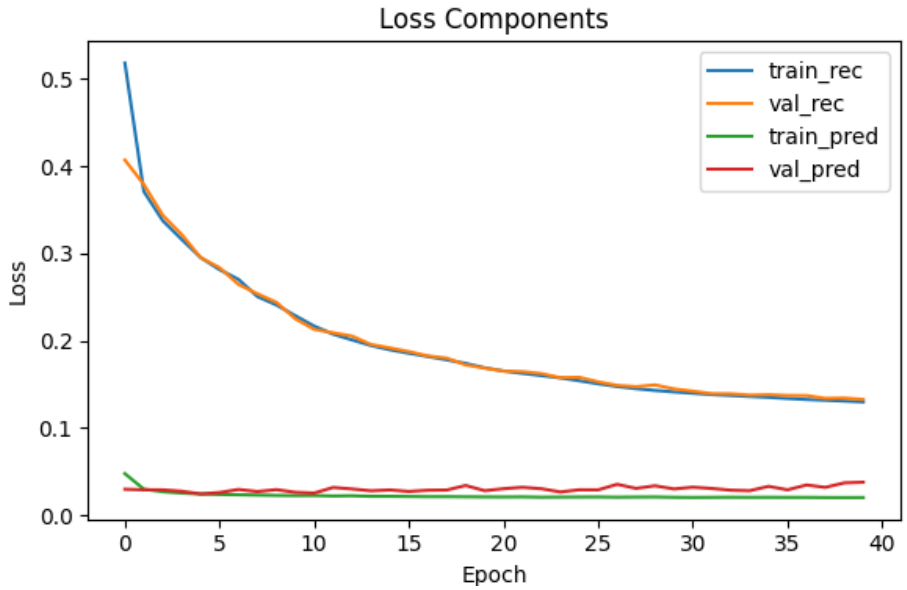

Figure 6. Training Loss Component

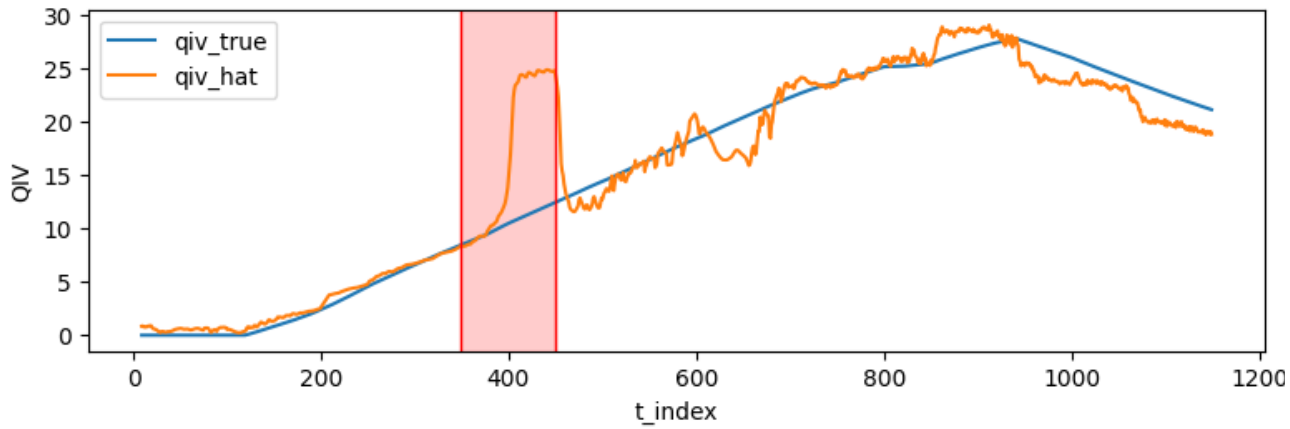

Figure 7. QIV prediction vs. true QIV for Batch 8

monitoring than the recurrent baselines: LSTM suffers from frequent pre-fault false alarms and missed detections, while GRU can detect faults but may alarm very late on hard cases (e.g., Batch 8). The proposed method achieves reliable detection with lower delays and cleaner alarm trajectories.

Case Study of Batch 8: Monitoring, Soft-Sensing and SHAP Diagnosis

This subsection presents a representative fault-batch case study to illustrate how the proposed framework behaves during inference and how the diagnostic module supports interpretation. In our experimental outputs, the plots run corresponds to fault batch_id = 8 in the evaluation list. For this batch, the fault-metric summary reports Fault Detect Success (FDS) = 1, False Alarm Rate (FAR) = 0, and Event/early detection delay (EDT) = 54 time steps, indicating that the fault is detected successfully with no pre-fault alarms and a moderate detection delay.

We first verify that the joint learning objective is optimized stably. The overall loss curve (**Figure 6**) decreases smoothly for both training and validation, and the gap remains small, suggesting stable convergence without severe overfitting. The loss component plot further shows that the reconstruction term dominates the objective, while the prediction term remains comparatively small yet consistent across epochs. This behavior is expected under semi-supervised operation: reconstruction learning is always available from process trajectories, while prediction learning is constrained by sparse quality labels. Together, these curves support that the shared temporal representation can be trained reliably and that both task branches are optimized simultaneously.

We then examine the online soft-sensing behavior for this batch. **Figure 7** plots the ground-truth QIV (y) and

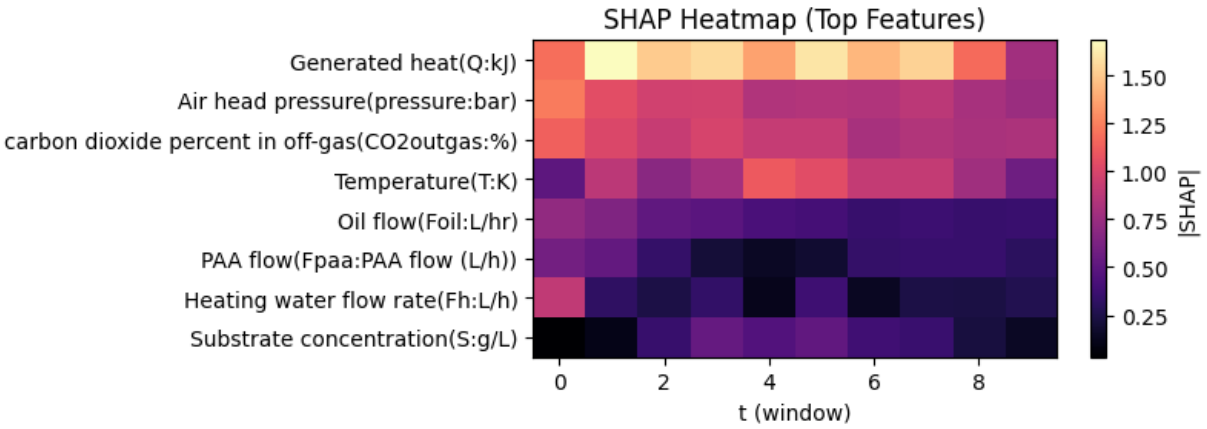

Figure 8. SHAP heatmap for the alarm window of Batch 8. Warmer colors indicate larger positive contributions to the score.

the model estimate ($\hat{y}$) over time, with the fault interval highlighted. Outside the fault window, $\hat{y}$ generally tracks the smooth trend of y, indicating that the deterministic prediction head provides a stable quality mapping from the pooled temporal features. In the highlighted fault interval, the predicted QIV shows a pronounced deviation from the true trajectory, which is consistent with the idea that faults can manifest as quality-relevant inconsistencies even when process trajectories are still within a plausible operating range. This mismatch later becomes a key contributor to the monitoring statistic through the prediction deviation component.

Finally, we demonstrate interpretability using SHAP attributions on an alarm window. **Figure 8** visualizes a SHAP heatmap for the top contributing variables within the local inference window (the horizontal axis corresponds to time positions inside the sliding window). The heatmap indicates that the score at this alarm moment is dominated by Generated heat (Q), Heating/cooling-related signals, Temperature, and closely coupled variables such as Air head pressure and CO_2 in off-gas, with additional contributions from Oil flow, PAA flow, and Heating water flow rate. Importantly, this attribution pattern is not only interpretable but also diagnostically specific: the concentration of contributions on thermal-balance and heat-removal pathways is consistent with the known fault label for Batch 8, i.e., a Coolant-Flowrate Fault. A reduction or perturbation in coolant flow directly weakens heat removal, which manifests as abnormal thermal behavior (Q and temperature), and subsequently propagates to gas-exchange and pressure dynamics through shifts in metabolic state and mass-transfer conditions. Therefore, the SHAP heatmap provides an actionable diagnosis that can localize the event to a cooling/coolant subsystem disturbance, rather than suggesting a feed-driven or aeration-driven root cause, effectively aligning the model's explanation with the ground-truth fault type (Coolant-Flowrate Fault) for Batch 8.

In summary, Batch 8 illustrates the full workflow of the proposed framework: (i) stable joint training with simultaneous reconstruction and prediction learning; (ii) online quality tracking with fault-induced prediction deviation during the fault window; and (iii) SHAP-based explanation that localizes the alarm to a small set of physically interpretable variables, supporting actionable diagnosis.

CONCLUSIONS

This work advances batch analytics beyond isolated "soft sensor + monitor" toolchains by introducing a unified, explainable framework that learns a reusable process-state representation for nonstationary operations while respecting the industrial reality that quality measurements are delayed and unavailable online. Looking forward, this architecture provides a strong foundation for the next step in industrial AI: integrating topology or physics constraints for sharper root-cause localization, adopting uncertainty-aware and adaptive fusion to handle evolving label availability, and enabling transfer/continual learning across recipes and plants to minimize retraining. Together, these directions point toward monitoring systems that are not just accurate, but trustworthy, maintainable, and decision-shaping in order to close the loop from data to diagnosis to intervention in modern batch manufacturing.

ACKNOWLEDGEMENTS

This work was supported by the National Key R&D

Program of China under Grant 2024YFC3013601.

REFERENCES

1. Sun Q, Ge Z. A survey on deep learning for data-driven soft sensors. IEEE Trans. Ind. Inf. 17:5853-5866 (2021). https://doi.org/10.1109/tii.2021.3053128
2. Zhao Z, Lee JM. Soft sensing and fault detection of batch processes using a multi-task deep learning framework. Comput Chem Eng 163:107827 (2022) https://doi.org/10.1016/j.compchemeng.2022.107827
3. Metcalfe B, Acosta-Pavas JC, Robles-Rodriguez CE, Georgakilas GK, Dalamagas T, Aceves-Lara CA, Daboussi F, Koehorst JJ, Corrales DC. Towards a machine learning operations (mlops) soft sensor for real-time predictions in industrial-scale fed-batch fermentation. Computers & Chemical Engineering 194:108991 (2025). https://doi.org/10.1016/j.compchemeng.2024.108991
4. Zhao C. Perspectives on nonstationary process monitoring in the era of industrial artificial intelligence. Journal of Process Control 116:255-272 (2022). https://doi.org/10.1016/j.jprocont.2022.06.011
5. Bi X, Zhao J. A novel orthogonal self-attentive variational autoencoder method for interpretable chemical process fault detection and identification. Process Safety and Environmental Protection 156:581-597 (2021). https://doi.org/10.1016/j.psep.2021.10.036
6. Tang P, Peng K, Dong J. Nonlinear quality-related fault detection using combined deep variational information bottleneck and variational autoencoder. ISA Transactions 114:444-454 (2021). https://doi.org/10.1016/j.isatra.2021.01.002
7. Lee YS, Chen J. Developing semi-supervised latent dynamic variational autoencoders to enhance prediction performance of product quality. Chemical Engineering Science 265:118192 (2023). https://doi.org/10.1016/j.ces.2022.118192
8. Jia M, Xu D, Yang T, Liu Y, Yao Y. Graph convolutional network soft sensor for process quality prediction. Journal of Process Control 123:12-25 (2023). https://doi.org/10.1016/j.jprocont.2023.01.010
9. Guo W, Zhu J, Yu X, Jia M, Liu Y. Temporal graph convolutional network soft sensor for molecular weight distribution prediction. Chemometrics and Intelligent Laboratory Systems 252:105196 (2024). https://doi.org/10.1016/j.chemolab.2024.105196
10. Dai Y, Yang C, Zhu J, Liu Y. Adversarial transferred data-assisted soft sensor for enhanced multigrade quality prediction. ACS Omega 8:19900-19911 (2023). https://doi.org/10.1021/acsomega.3c01832
11. Wu D, Bi X, Zhao J. Protopormer: toward understandable fault diagnosis combining process topology for chemical processes. Ind. Eng. Chem. Res. 62:8350-8361 (2023). https://doi.org/10.1021/acs.iecr.3c00206
12. Rao S, Wang J. A comprehensive fault detection and diagnosis method for chemical processes. Chemical Engineering Science 300:120565 (2024). https://doi.org/10.1016/j.ces.2024.120565
13. Hung HY, Jinsong Z. Enhancing batch chemical manufacturing via development of deep learning based predictive monitoring with transfer learning. Systems and Control Transactions 4:1548-1554 (2025). https://doi.org/10.69997/sct.188760
14. Goldrick S, Ştefan A, Lovett D, Montague G, Lennox B. The development of an industrial-scale fed-batch fermentation simulation. Journal of Biotechnology 193:70-82 (2015). https://doi.org/10.1016/j.jbiotec.2014.10.029
15. Birol G, Ündey C, Çinar A. A modular simulation package for fed-batch fermentation: penicillin production. Computers & Chemical Engineering 26:1553-1565 (2002). https://doi.org/10.1016/s0098-1354(02)00127-8