

Channel-Attentive Temporal Convolutional Networks (CA-TCN) for Interpretable Fault Diagnosis in the Tennessee Eastman Process

Pratyush Kumar Pal¹, Sumana Ray², Narottam Behera³, Somnath Chowdhury⁴, Abhiram Hens⁵, and Sandip Kumar Lahiri^{6*}

^{1,2,3,4,5,6} *National Institute of Technology Durgapur, Mahatma Gandhi Avenue, Durgapur 713209, West Bengal, India*

pkp.21ch1502@phd.nitdgp.ac.in

sr.19ch1501@phd.nitdgp.ac.in

nb.20ch1504@phd.nitdgp.ac.in

sc.19ch1103@phd.nitdgp.ac.in

ahens.che@nitdgp.ac.in

**Corresponding author: sklahiri.che@nitdgp.ac.in*

ABSTRACT

Fault detection and diagnosis (FDD), Fault classification in the Tennessee Eastman Process (TEP) involves strongly coupled, nonlinear multivariate trajectories. The present research evaluates a residual Temporal Convolutional Network (TCN) with Squeeze-and-Excitation (SE) channel attention, denoted CA-TCN. We compare CA-TCN with plain TCN, Long Short-Term Memory (LSTM), and Convolutional Neural Network-LSTM (CNN-LSTM) baselines. CA-TCN and TCN+SE refer to the same implemented architecture.

In the primary five-seed comparison, plain TCN achieved a mean accuracy of $99.8756 \pm 0.0807\%$, compared with $99.8511 \pm 0.1104\%$ for CA-TCN, $99.1711 \pm 0.9467\%$ for LSTM, and $99.8444 \pm 0.0820\%$ for CNN-LSTM. A separate matched five-seed TCN/CA-TCN ablation yielded mean accuracies of $99.8000 \pm 0.1199\%$ and $99.7756 \pm 0.1823\%$, respectively.

The mean paired CA-TCN-minus-TCN difference was -0.0244 percentage points. The exploratory paired analysis did not identify a statistically detectable difference in accuracy. Adding the SE module introduced 4,192 additional trainable parameters and increased the measured training time while providing explicit reweighting of 64 learned latent feature channels.

These coefficients are interpreted as internal latent-channel recalibration rather than sensor-importance scores. The results therefore characterize the performance–complexity trade-off associated with adding SE attention to the residual TCN backbone.

The conclusions are limited to offline whole-trajectory classification on the evaluated TEP benchmark; cross-domain generalization and online fault-detection performance were not assessed.

Keywords: Fault detection and diagnosis (FDD); Temporal Convolutional Network; Channel Attention; Tennessee Eastman Process; Deep Learning; Process Monitoring; Squeeze-and-Excitation

1. INTRODUCTION

With the advent of Industry 4.0, modern chemical plants have become increasingly complex and data-rich. In the Tennessee Eastman Process (TEP), process variables are strongly coupled, and faults exhibit nonlinear and dynamic characteristics, making first-principles modelling challenging (Yin et al., 2012; Reinartz et al., 2021; Downs and Vogel, 1993). To prevent downtime and ensure safety, one important field of research has become data-driven Fault Detection and Diagnosis (FDD) (Reinartz et al., 2021).

The first generation of data-driven approaches largely relied on multivariate statistics. Principal Component Analysis (PCA) and Partial Least Squares (PLS) are well-known tools for dimensionality reduction and anomaly detection.

These linearity and Gaussian assumptions (C. Zhang et al., 2021) impose restrictions and are unable to detect complex time-varying behaviours of dynamic chemical processes (Onel et al., 2019). Support vector machines (SVMs) and other machine learning modules are more often used to

Pratyush Kumar Pal et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

<https://doi.org/10.36001/IJPHM.2026.v17i2.4859>

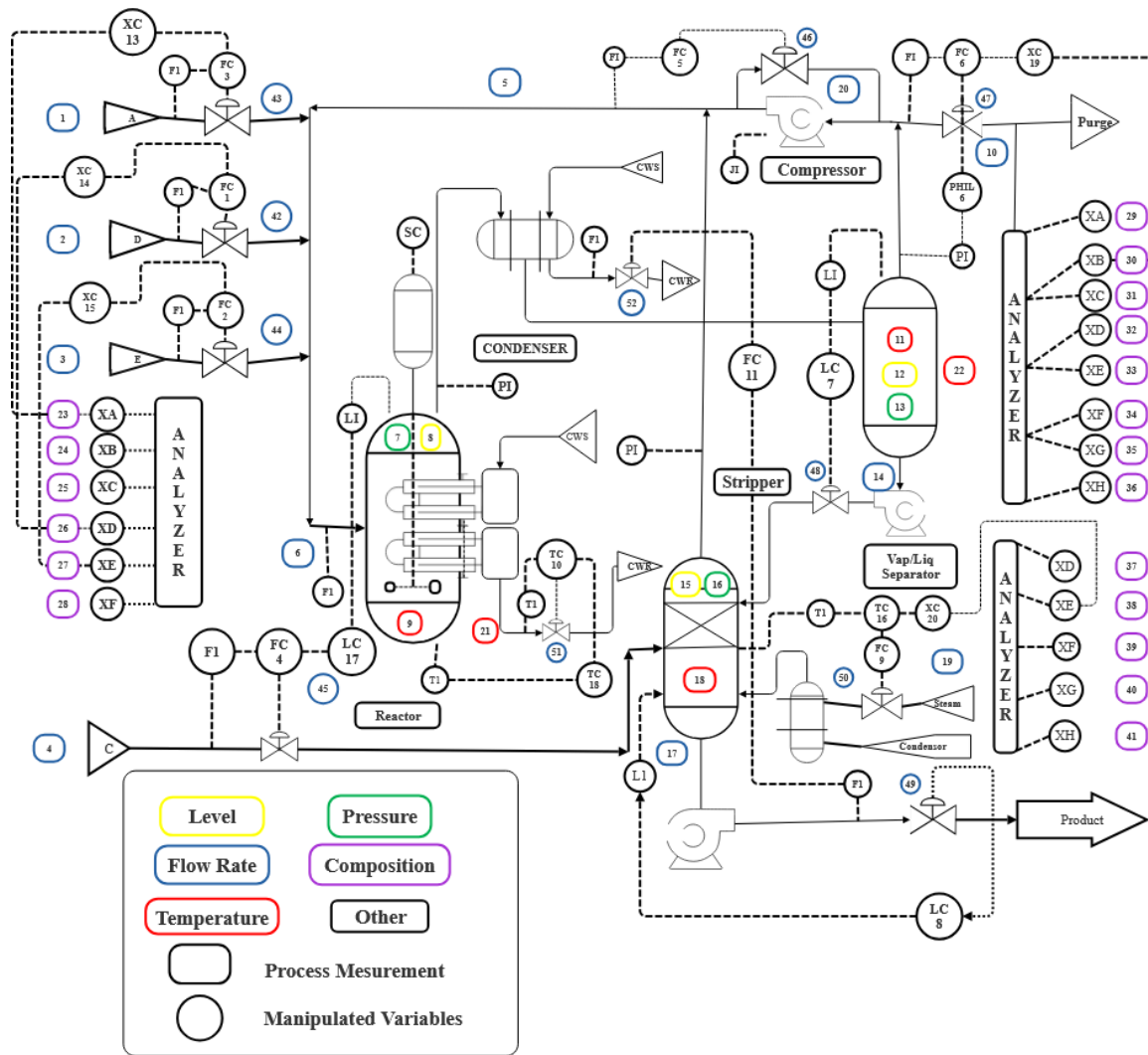


Figure 1. The TEP flow chart Schematic diagram

address these limitations. The algorithms do not only depend on manual features, but rather on those which are cheap and domain-independent (Chiang et al., 2001).

Deep Learning's (DL) capacity to learn representations from raw time-series data without explicit programming has led to its replacement of conventional shallow learning over the past few years (Agarwal et al., 2021; Lomov et al., 2021).

In particular, RNNs and a more advanced version, the LSTM networks, have been used to model temporal dependencies in process data (Hochreiter & Schmidhuber, 1997; Agarwal et al., 2022). LSTM efficiently memorizes long history for fault diagnosis, as reported in many research studies (Han et al., 2020). To improve the accuracy level, a hybrid architecture like CNN-LSTM is offered. CNN processes the sensor data to obtain local spatial characteristics, and LSTM retrieves temporal features from the CNN output (B. Chen et al., 2023). While these recurrent architectures can obtain strong

accuracy, they also have significant downsides. Models based on LSTM are mainly limited by their sequential nature.

Because LSTMs process temporal states recurrently, computation across time steps cannot be fully parallelized, which can increase training and inference latency for long sequences (Bai et al., 2018). Although LSTM gating mitigates the vanishing gradient problem of conventional RNNs, recurrent state updates still impose sequential computation and can make very long sequences computationally demanding (Hochreiter & Schmidhuber, 1997). Computational efficiency is also an important consideration for deployable fault-diagnosis models. For the TEP benchmark, Li and Shao (2021) investigated pruning, clustering, and quantization of neural networks and that substantial model compression could be achieved while largely retaining fault-detection accuracy (Li & Shao, 2021). Among other recent advances in sequential modelling architectures, TCNs emerged as a strong alternative. By including dilated causal convolutions in TCNs, one can

maintain the benefits of parallelisation afforded by the CNN formulation. This is achieved by allowing the use of very large effective history sizes (receptive fields) (Bai et al., 2018; Lea et al., 2016; Wang et al., 2022). As per Bai et al., 2018, TCNs show competitive performance with recurrent architectures in several sequence-modelling tasks. Nevertheless, RNNs remained the most popular model for industrial process monitoring applications. Industrial safety applications suffer due to the "black box" nature of deep learning models. Operators need to understand the variables underlying the fault diagnosis. To bridge such a gap, attention mechanisms (Vaswani et al., 2017) have been adapted from natural language processing to computer vision and time-series analysis (Hu et al., 2018). Attention mechanisms can dynamically adjust learned feature representations. In the current architecture of the CA-TCN model, this reweighting is applied to latent feature channels rather than to the original physical sensors.

In this paper, "CA-TCN" denotes a residual temporal convolutional network with an SE channel-attention module. As such, CA-TCN and TCN+SE refer to the same implemented architecture, not two models. To quantify the incremental contribution of SE-based latent-channel reweighting (Hu et al., 2018) we evaluate a corresponding plain TCN that possesses the same temporal-convolutional backbone and classification structure but not the SE module.

The TCN backbone is distinctively different from LSTM-based models, as it does not require recurrent time-step updates. Thus, temporal convolutions are evaluated in parallel. In CA-TCN, the temporal-convolutional backbone transforms the 52 original process variables into 64 learned latent feature channels. The SE module generates 64-channel domain-gating coefficients from the input feature maps, which are multiplied by the above learned feature maps before classification. As a result, the weights characterise adaptive latent-channel reweighting rather than the importance scores of individual TEP sensors. As the gating coefficients are generated explicitly and examined, the operation provides the internal channel-recalibration behaviour. Nonetheless, assigning a particular latent channel to a specific physical process variable would necessarily require a clear attribution method.

The experimental evaluation includes LSTM, CNN-LSTM, plain TCN, and CA-TCN baselines. A dedicated plain-TCN/CA-TCN ablation is additionally conducted to isolate the incremental contribution of the SE module. In this ablation, plain TCN and CA-TCN use the same temporal-convolutional backbone, residual projection, pooling operation, classification head, trajectory partition, and

training protocol; the SE channel-reweighting module is the only architectural difference.

The rest of the paper is structured like as, Section 2 provides a summary of relevant CNN, LSTM, CNN-LSTM, and TCN architectures, including the TEP data preparation, the CA-TCN architecture, the training configuration, and the evaluation protocol described in Section 3. Section 4 discusses and presents the results of controlled experiments, ablation studies, statistical tests, and the paper's limitations. We conclude in Section 5. The Tennessee Eastman Process benchmark is the empirical scope of the present study. As a result, the current experiments do not imply performance on additional industrial processes, sensor configurations, fault distributions, or real-time streaming environments.

2. THEORETICAL BACKGROUND: DEEP LEARNING ARCHITECTURES AT A GLANCE

In this section, we first explain the mechanisms of the baseline models (CNN, LSTM, Hybrid CNN-LSTM). Next, TCN describes its contribution to understanding the performance gains in the Tennessee Eastman Process.

2.1 Convolutional Neural Networks (CNN)

A Convolutional Neural Network (CNN) is a deep learning model primarily used for computer vision tasks. Their ability to extract "spatial" features means the networks can find local correlations among different variables at a given instant in time. For multivariate time-series data, a 1D CNN convolves along the temporal dimension, treating the process variables as input feature channels and enabling filters to learn local temporal patterns jointly across variables.

The basic 1D-CNN methodology processes a sequence by applying a filter (kernel) to the inputs. A one-dimensional convolution for a multivariate sequence computes the representation at the time step t as follows:

$$h_t = f(w * x_t + b)$$

where W_k denotes the convolution weights at temporal lag k , b is the bias term, and $\sigma(\cdot)$ is the activation function (typically ReLU). As with standard multilayer neural networks, stacking convolutional layers enables them to learn increasingly complex features. Standard CNNs, however, find it a challenge to comprehend temporal long-distance data because their receptive fields are restricted to their kernel sizes.

2.2 Long Short-Term Memory (LSTM)

To address the time-limiting factors of CNNs, LSTM networks, a variant of RNNs, are the best-known solution to the standard RNN vanishing-gradient problem.

An LSTM cell C_t regulates information flow through three principal gates: the forget gate (f_t), input gate (i_t), and output gate (o_t). These gates regulate which information is kept from the cell state, which information is added, and which part of the cell state is used as the hidden state. Even though LSTMs resolve the vanishing gradient issue of ordinary RNNs, their recurrence prevents easy temporal parallelisation of processing.

2.3 Hybrid CNN-LSTM

The CNN-LSTM aims to combine the best of both methods. This hybrid method uses CNN blocks as the initial layers to extract local, noise-robust features from raw sensor data. LSTM layers receive these high-level feature vectors to recreate how the fault will evolve. While this architecture achieves high accuracy through spatial filtering and temporal memory, the LSTM component is computationally inefficient. The sequence dependency persists, preventing the model from fully leveraging modern parallel hardware (GPUs) during training.

2.4 Temporal Convolutional Networks (TCN)

The TCN model uses convolutions to model sequential data rather than recurrent updates. In this research, two particular features are important, namely causal convolution and dilation. Causal convolution limits the information from future time steps used for the current representation, while dilated convolution widens the temporal receptive field without increasing depth. In this implementation, four causal convolutions with kernel size 3 and dilation factors 1, 2, 4, and 8 provide a local receptive field of 31 samples. Causal padding ensures that each convolutional representation depends only on the current and preceding samples. Whole-trajectory global pooling is subsequently used for SE recalibration and trajectory-level classification.

Residual connections are also employed to facilitate gradient propagation and stabilize optimization in stacked convolutional networks.

3. METHODOLOGY

To enable a fair comparison of plain TCN, CA-TCN, LSTM, and CNN-LSTM under experimental conditions, a research setup was developed.

3.1 Data and Trajectory Preparation: Tennessee Eastman Process (TEP) - Case Study

This research employs the benchmark Tennessee Eastman Process (TEP) (Downs and Vogel, 1993) which is widely recognised as the standard benchmark for evaluating algorithms used in process monitoring. The system contains a compressor, condenser, reactor, separator, and stripper. It

comprises 52 variables (supplementary Table S2). The System Model also comprises 11 manipulated (XMV) and 41 measured (XMEAS) variables, including flow, temperature, pressure, level, and composition. These variables represent realistic plant-wide dynamics. The normal operation (IDV 0) and 21 pre-defined faults (IDV 1–21) are defined by the benchmark. The current evaluation comprises 18 classes: normal operation and Faults 1, 2, 4-8, 10-14, and 16-20 (see Supplementary Table S1). Fault 21 is not available in the supplied quantitative data tables used for this article.

In agreement with previous studies (Lomov et al., 2021; Yin et al., 2012, 2014) Faults 3, 9 and 15 were not considered here. The weaknesses of the non-standard variable set do not appear to have non-negligible effects in the standard set, since their signatures are weak in the standard measurement set, and they are usually excluded from quantitative benchmarking.

The test records are not used for checkpoint selection in the controlled experiments. However, because the test set had been inspected during earlier exploratory work, it is not treated here as newly untouched confirmation data.

Data partitioning is carried out at the level of simulation runs. For each of these 18 evaluated classes, model training is assigned to training-source simulation runs 1–400. This allocation resulted in 400 trajectories per class, for a total of 7,200 training trajectories. Out of the 401–500 training-source runs, 100 trajectories per class are assigned to validation, yielding a total of 1800 validation trajectories. A distinct source for testing comprised runs 1–500 for each class, yielding 500 trajectories per class and a total of 9,000 trajectories. Retained together throughout pre-processing were source-file identity and simulation-run identity, so that trajectories from different runs are not merged.

The parameters for feature-wise Z-score normalization are computed only from the training data. More specifically, the rows originate from the training-source runs 1–400. For all 52 input variables, the mean and standard deviation are computed over the training partition and then applied unchanged to the validation and test trajectories. No validation or test observations are used to estimate the normalization parameters. The inputs to the model are entire simulation-run trajectories, not overlapping sliding windows. Every training and validation trajectory consists of 500 successive time samples. In contrast, every test trajectory has 960 samples.

As per Table 1 and the resulting run-level partition or trajectory-construction protocol, no sequence crosses a simulation-run boundary, and there are no overlapping sub-windows in the trajectory.

Table 1. Data partition and trajectory-construction protocol

Subset	Source runs per class	No. of trajectories	Samples per trajectory	Total time-step samples	Overlap
Training	1–400	7,200	500	3,600,000	0
Validation	401–500	1,800	500	900,000	0
Test	1–500, separate test source	9,000	960	8,640,000	0

Thus, a traditional sliding-window stride is not applied. The effective row advance is the entire source-trajectory length – 500 samples for training and validation and 960 samples for testing – corresponding to zero overlap. Normalization is performed independently for each input variable.

$$x_{norm} = \frac{x - \mu_{train}}{\sigma_{train}}$$

where μ_{train} and σ_{train} are estimated exclusively from the training partition. The same parameters are reused for validation and test trajectories.

Each complete simulation-run trajectory is assigned the fault-scenario label associated with its source run, including any pre-fault portion contained within that trajectory; no labels are generated independently at individual time steps.

3.2 Residual TCN with SE Channel Attention (CA-TCN)

The CA-TCN implementation comprises a residual temporal-convolutional backbone, an SE channel-attention module, and a pooled classification head, as shown in Figure 2. Thus, CA-TCN is the TCN+SE implementation that we evaluate in this work. For the direct attention ablation, the plain TCN retains the same temporal-convolutional backbone, residual projection, pooling operation, and classification head, but removes its SE module.

To capture local and progressively broader temporal patterns, four stacked 1D causal convolutional layers with dilation factors of 1, 2, 4, and 8 are used. With a kernel size of 3, this configuration provides a 31-sample local receptive field. Causal padding ensures that the representation at time t depends only on the current and preceding samples. Whole-trajectory temporal pooling subsequently aggregates the learned representations for trajectory-level classification.

The dilation factors increase exponentially across successive layers, thereby expanding the temporal receptive field without requiring a proportional increase in network depth.

The dilated causal convolution at time t for multivariate sequence $X = \{x_1, \dots, x_T\}$ is expressed as:

$$h_t = \sigma \left(\sum_{k=0}^{K-1} W_k x_{t-dk} + b \right)$$

where K is the kernel size, d is the dilation factor, W_k denotes the learnable convolution weights, and $\sigma(\cdot)$ is the nonlinear activation. The causal formulation ensures that h_t depends only on the current and preceding samples.

Each layer has 64 filters with a kernel size of 3, but the receptive field expands through exponentially increasing dilation factors as we go deeper into the network. It allows the fault evolution history to lengthen without extra depth. After the convolutional stack, we use a Squeeze-and-Excitation-based model.

The SE module recalibrates the 64 latent feature channels generated from the temporal-convolutional backbone. Initially, temporal global-average pooling summarizes all latent channels across the trajectory. The 64-dimensional representation shows the fully connected layer, which maps from 64 to 32 dimensions and uses ReLU as the activation function. Then, the second fully connected layer is shown, which performs the mapping of 32 to 64, and uses Sigmoid as the activation function, and the output is a gating vector $\alpha \in [0,1]^{64}$. The gating coefficient is broadcast along the temporal dimension and multiplied element-wise by the corresponding latent feature map.

The derived weights thus characterize the latent-channel reweighting, not the importance of sensors. After recalibrating the SE module, we use temporal global-average pooling to produce a fixed-dimensional representation.

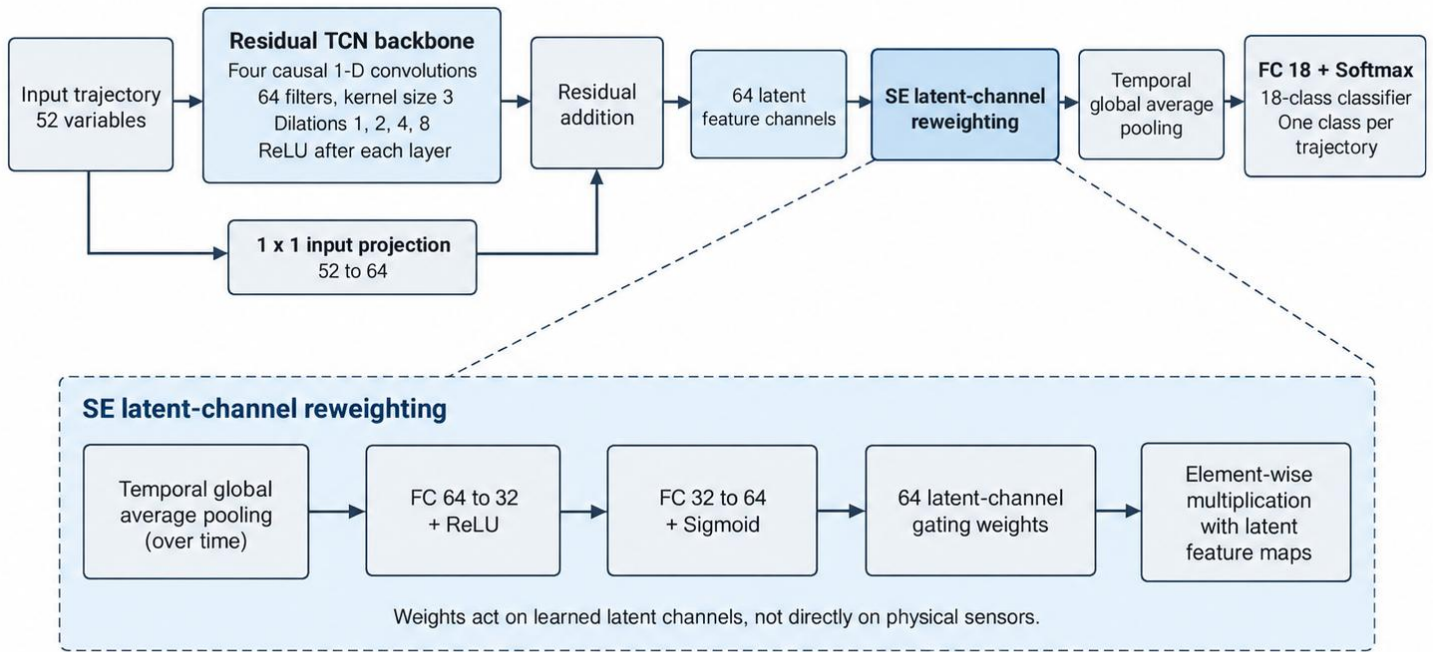


Figure 2. Executed residual TCN architecture with Squeeze-and-Excitation latent-channel reweighting.

The representation then passes through the final fully connected layer and Softmax layer. The output is a probability distribution over 18 classes for each trajectory.

TCN and Squeeze-and-Excitation are established architectural components. The current article does not claim the new attention mechanism, as it arises only from that combination. Here, the acronym CA-TCN is used descriptively to refer to the deployed residual TCN+SE model. CBAM and Transformer-based architectures use different attention mechanisms and are used only for architectural context. The additional SE input in this implementation is thus directly tested via an identical-backbone plain-TCN ablation.

The specific structural hyperparameters of CA-TCN are presented for reproducibility. The network accepts variable-length complete trajectories, with training and validation inputs of 500 time samples and a test trajectory of 960 time samples. Temporal global pooling produces fixed-dimensional representations for the classification head despite these differing trajectory lengths. The ReLU activates the intermediate fully connected layer in the squeeze-and-excitation channel attention module, producing non-linear dimensionality reduction. The last gating layer employs a

sigmoid function that restricts each SE coefficient to the range $[0,1]$.

The number of trainable parameters for the executable plain-TCN and CA-TCN networks is 51666 and 55858, respectively. As a result, the SE channel-reweighting module adds 4,192 parameters compared to the plain TCN architecture.

3.3 Training Configuration

Four controlled models (plain TCN, CA-TCN, LSTM and CNN-LSTM) were all trained using categorical cross-entropy and the Adam optimizer under a common training budget of 30 epochs. We set the mini-batch size to 64, and the initial learning rate to 1×10^{-3} , with a learning-rate reduction of 0.1 after 20 epochs, Adam decay factors of 0.9/0.999, epsilon to 10^{-8} , L2 regularization 0.0001, a gradient threshold of 1 and single precision. The models used identical training, validation, and test partitions along with normalization parameters derived from training. We performed the training with single-precision computation on our NVIDIA GeForce RTX 3050 Laptop GPU with MATLAB R2023a. The baseline network architectures are held constant while the main conditions and mechanisms for optimization and

evaluation are standardized for the comparison through a controlled trial.

3.4 Training and Evaluation Protocol

The assessment consists of LSTM, CNN-LSTM, plain TCN, and CA-TCN (TCN+SE). Alongside the primary multi-model comparison, we also conducted a matched TCN/CA-TCN ablation study to isolate the additional value of SE channel attention.

Both plain TCN and CA-TCN are investigated for the dedicated attention ablation under the same bounded hyperparameter selection procedure. Four learning-rate / L2-regularization combinations are assessed using tuning seeds 101 and 211 and, for both architectures, configuration C01 was selected which corresponds to an initial learning rate of 0.001 and L2 regularization of 0.0001. The next matched evaluation employed seeds 307, 401, 503, 601 and 701. For each pair of seeds, the same trajectory partition, training budget, shared-backbone initialization and model-selection rule are employed.

In the controlled runs, early stopping was disabled. The complete run reached the 30-epoch training budget we advised. The first checkpoint that achieves the minimum recorded validation loss is used for model selection. Checkpoint selection did not utilize test predictions.

LSTM and CNN-LSTM are re-implemented as recurrent baselines, while the plain TCN serves as the direct architectural baseline for CA-TCN. The four models utilized the controlled data partitions, and evaluation conditions described above.

Using five training seeds, we assessed robustness. The models used the same 30-epoch budget, and the returned checkpoint corresponded to the point at which the validation loss was first minimized. Checkpoint selection did not use test predictions. To identify variation related to model training as opposed to a single favourable initialization, a multi-seed design was used (Agarwal et al., 2021; B. Chen et al., 2023; H. Chen et al., 2022; Verma et al., 2022).

We empirically evaluate LSTM, CNN-LSTM, plain TCN and CA-TCN under the controlled protocol above. The purpose of the comparison is not to assume a priori that one architecture has superior classification performance and computational behaviour over another (Agarwal et al., 2021; Chadha & Schwung, 2017; Onel et al., 2019).

3.5 Validation Procedure

The evaluation of performance on the fixed test set is conducted in terms of accuracy, precision, recall and macro-averaged F1-score for the 18 evaluated classes. Table 2 contains a summary of the definitions of these metrics. Macro F1 is used to assign equal weight to each class when summarizing class-wise performance.

Table 2. Performance metrics of the four standard classifications and their mathematical form

Metric	Formula	Description / Interpretation
Accuracy	$\left(\frac{TP+TN}{TP+TN+FP+FN} \right)$	$N_{Correct}/N$: fraction of evaluated trajectories assigned the correct class.
Precision	$\left(\frac{TP}{TP+FP} \right)$	$TP/(TP+FP)$: for a given class, the fraction of predicted positives that are true positives.
Recall (Sensitivity)	$\left(\frac{TP}{TP+FN} \right)$	$TP/(TP+FN)$: for a given class, the fraction of actual positives correctly identified.
F1-Score	$\left(2 \frac{Precision \times Recall}{Precision+Recall} \right)$	$2PR/(P+R)$: harmonic mean of precision and recall; it is high when both precision and recall are high.

Legend: TP: True Positives, TN: True Negatives, FP: False Positives, FN: False Negatives (Macro F1 is calculated as the arithmetic mean of the class-wise F1-scores over the 18 evaluated classes.)

4. Results and Discussion

The experimental comparison of CA-TCN with LSTM, CNN-LSTM, and plain TCN baselines. A separate matched TCN/CA-TCN experiment utilized to quantify the incremental contribution of SE channel attention. We evaluate classification performance and computational behaviour, latent-channel reweighting, and interpretation limits of the learned representation.

4.1 Model Convergence and Training Behaviour

All four models in the controlled comparison completed a training budget of 30 epochs. As shown in Figures 3-5, the training and validation histories illustrate the results. The respective validation-loss trajectories for the same assessment run illustrated earlier for TCN, CA-TCN, LSTM, and CNN-LSTM are compared in Figure 6. These optimization curves, presented descriptively.

During training, the 500-sample trajectories are fed into the temporal convolutional network, and recurrent time-step unrolling is omitted.

The core five-seed comparison is controlled as shown in Table 3. The mean accuracy of Plain TCN was $99.8756 \pm$

0.0807% , followed by CA-TCN at $99.8511 \pm 0.1104\%$, CNN-LSTM at $99.8444 \pm 0.0820\%$, and LSTM at $99.1711 \pm 0.9467\%$. These distinctions are understood descriptively; the performance of any architecture is not uniformly superior.

Due to interruptions in the previous wall-clock, records obtained under unequal epoch budgets and certain controlled training logs are excluded from the comparative efficiency claims in the revised analysis. The computational comparison instead consists of several executable parameters and a fixed-input inference benchmark.

For the five controlled seeds, the mean batch-1 median inference times (in ms) for TCN, CA-TCN, LSTM and CNN-LSTM are 6.973, 8.452, 62.164, and 23.709, respectively, for 960-sample input trajectories. The median inference times for a batch size of 64 per trajectory were 0.655, 0.689, 1.382, and 1.006 ms, respectively. MATLAB inference calls on fixed inputs synchronized here using transfer and packing overhead, while disk reads are excluded. They define the offline inference implementation being evaluated, and should not be considered online fault-detection delay. The inference assessments for fixed inputs are indicated in Table 4.

Table 3: Primary matched experiment results over five training seeds

Model	Trainable parameters	Accuracy (%)	Macro F1 (%)
Plain TCN	51,666	99.8756 ± 0.0807	99.8757 ± 0.0804
CA-TCN (TCN+SE)	55,858	99.8511 ± 0.1104	99.8512 ± 0.1101
LSTM	43,788	99.1711 ± 0.9467	99.1675 ± 0.9536
CNN-LSTM	34,094	99.8444 ± 0.0820	99.8443 ± 0.0820

Table 4. Fixed-input inference benchmark for 960-sample trajectories

Model	Batch-1 median inference (ms/trajectory)	Batch-64 amortized median (ms/trajectory)
Plain TCN	6.973	0.655
CA-TCN (TCN+SE)	8.452	0.689
LSTM	62.164	1.382
CNN-LSTM	23.709	1.006

4.2 Fault-wise Classification Performance

The plain-TCN baseline has been included to delineate the performance of the temporal-convolutional backbone from the further incremental benefit of SE attention. The SE module has added this number of parameters, which account for an increase of 4,192 trainable parameters.

Table 5 reports the mean $\pm$ standard deviation of the class-wise F1-scores across the five controlled training seeds. Synchronized warm MATLAB inference calls used as timing on 960 fixed samples. The transfer/packing overhead is

included, whereas disk I/O is excluded from measurements. Using this fixed-input benchmark, TCN models took less inference time than the evaluated recurrent baselines. The addition of SE increased the batch-1 median inference time of plain TCN from 6.973 ms to 8.452 ms for CA-TCN. Corresponding batch-64 amortized values were 0.655 and 0.689 ms per trajectory, respectively. Thus, the evaluated implementation incurs a measurable but relatively small overhead in inference time and parameter count due to SE. The values are specific to the hardware and implementation used and are not to claim universal computational superiority.

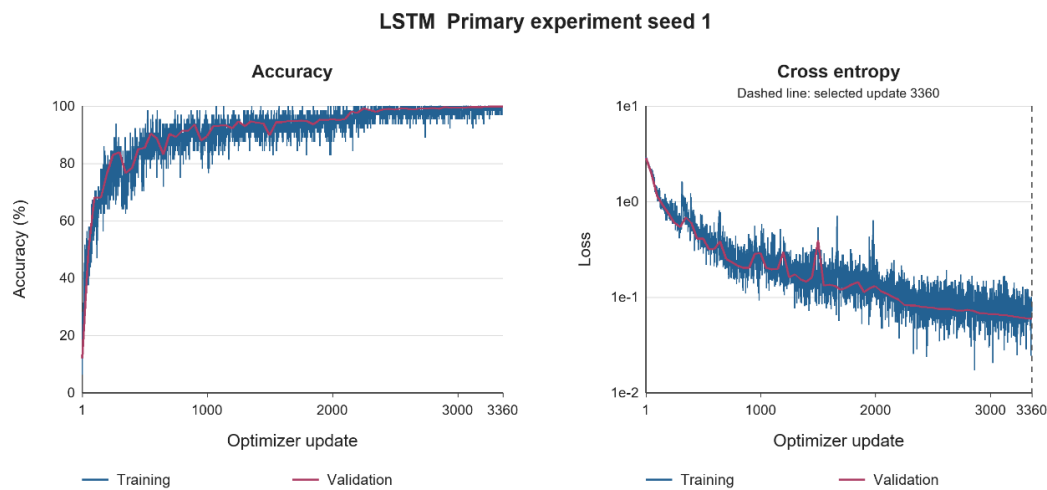


Figure 3. Primary experiment LSTM seed 1 training and validation history. The dashed line identifies the selected validation-loss minimum, not early stopping.

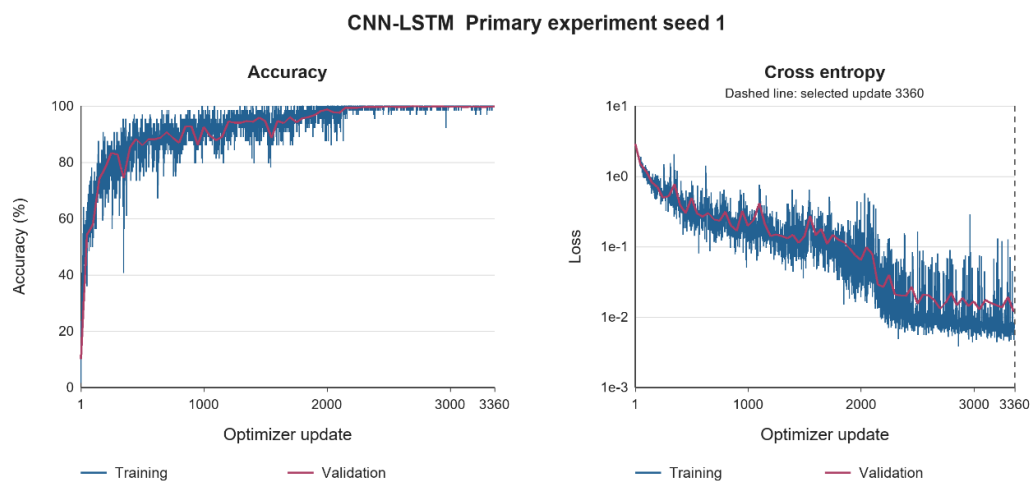


Figure 4. Primary experiment CNN-LSTM seed 1 training and validation history. All 30 epochs were completed, and the dashed line indicates the selected minimum-validation-loss checkpoint.

CA-TCN Primary experiment seed 1

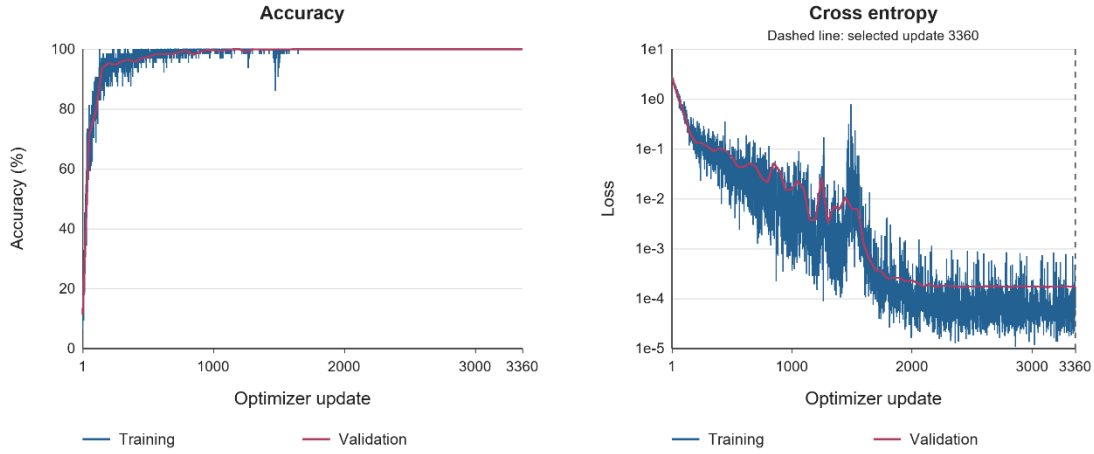


Figure 5. Primary-experiment CA-TCN seed-1 training and validation history. The dashed line identifies the selected minimum-validation-loss checkpoint; all 30 training epochs were completed

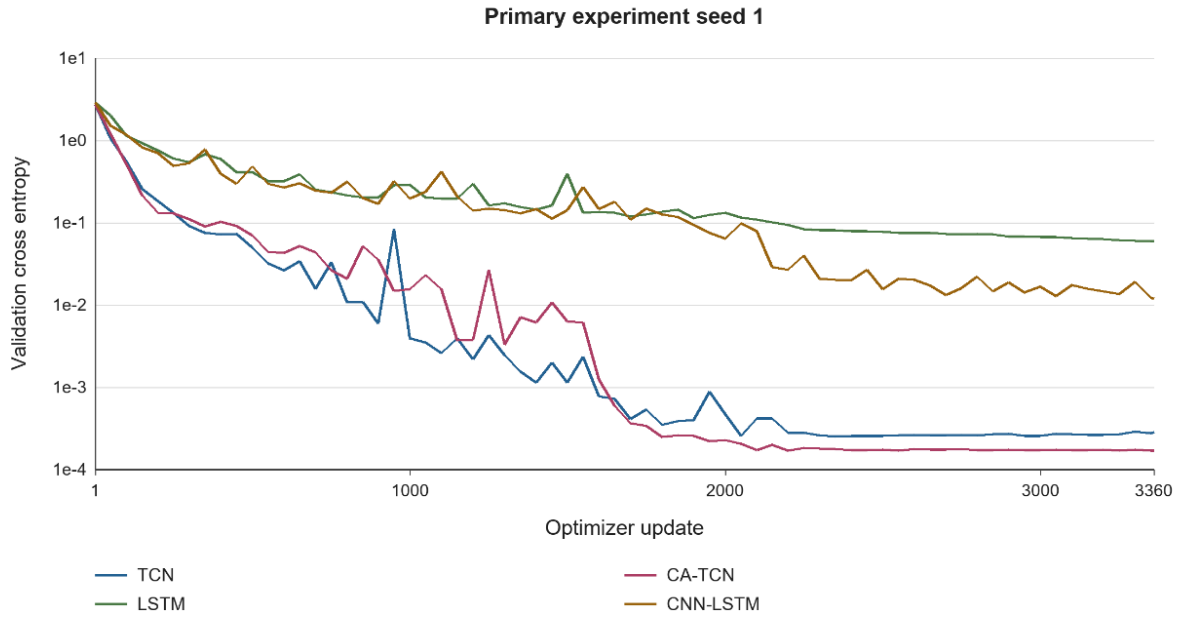


Figure 6. Validation-loss histories for TCN, CA-TCN, LSTM and CNN-LSTM in the illustrated controlled run. All models completed the common 30-epoch training budget. The curves describe optimization behaviour as a function of training iteration.

Table 5 reports the mean $\pm$ standard deviation of the class-wise F1-scores across the five controlled training seeds.

Fault ID	TCN F1 (%)	CA-TCN F1 (%)	LSTM F1 (%)	CNN-LSTM F1 (%)
0	100.000 $\pm$ 0.000	99.980 $\pm$ 0.045	98.702 $\pm$ 1.888	99.900 $\pm$ 0.141

Fault ID	TCN F1 (%)	CA-TCN F1 (%)	LSTM F1 (%)	CNN-LSTM F1 (%)
1	99.980 $\pm$ 0.045	100.000 $\pm$ 0.000	100.000 $\pm$ 0.000	100.000 $\pm$ 0.000
2	100.000 $\pm$ 0.000	100.000 $\pm$ 0.000	99.940 $\pm$ 0.134	100.000 $\pm$ 0.000
6	99.759 $\pm$ 0.231	99.960 $\pm$ 0.089	99.960 $\pm$ 0.055	99.920 $\pm$ 0.178
12	99.228 $\pm$ 0.673	99.095 $\pm$ 1.057	98.343 $\pm$ 1.795	99.740 $\pm$ 0.167
13	99.377 $\pm$ 0.373	99.497 $\pm$ 0.326	98.455 $\pm$ 0.828	99.456 $\pm$ 0.355
16	100.000 $\pm$ 0.000	99.860 $\pm$ 0.206	94.744 $\pm$ 9.831	99.252 $\pm$ 0.853
18	99.498 $\pm$ 0.432	99.109 $\pm$ 1.002	98.273 $\pm$ 0.553	99.761 $\pm$ 0.218

Note: Faults 3, 9, and 15 were excluded under the fixed evaluation protocol

The results for each selected class reveal that the ordering of the four architectures varies by fault class. Variation across seeds is greater for some classes of LSTM, whereas TCN, CA-TCN, and CNN-LSTM cluster together for many of the reported faults. The class-wise values here are only descriptive of predictive performance and not used to infer the internal mechanism responsible for a certain error pattern.

The relative ordering of the four architectures varies across fault classes. Plain TCN achieved the highest mean F1-score for Fault 16, whereas CNN-LSTM achieved the highest mean F1-scores for Faults 12 and 18. CA-TCN achieved the highest mean F1-score for Fault 13 and tied with LSTM for the highest mean value for Fault 6. For Faults 1 and 2, multiple models achieved essentially identical or perfect mean F1-scores. These class-wise differences are treated descriptively

and not interpreted as evidence that any architecture uniformly dominates the others.

The observed class-wise differences cannot be attributed to a specific behaviour of the SE module from classification metrics alone. The present results says differences in predictive performance but do not identify the latent-feature mechanisms responsible for them.

4.3 Visualization of Fault Classification through a Confusion Matrix

Figures 7, 8, 9 for LSTM, CNN-LSTM and CA-TCN, respectively, provide the confusion matrix. The matrices show mostly diagonal predictions and small off-diagonal classifications in the runs illustrated here.

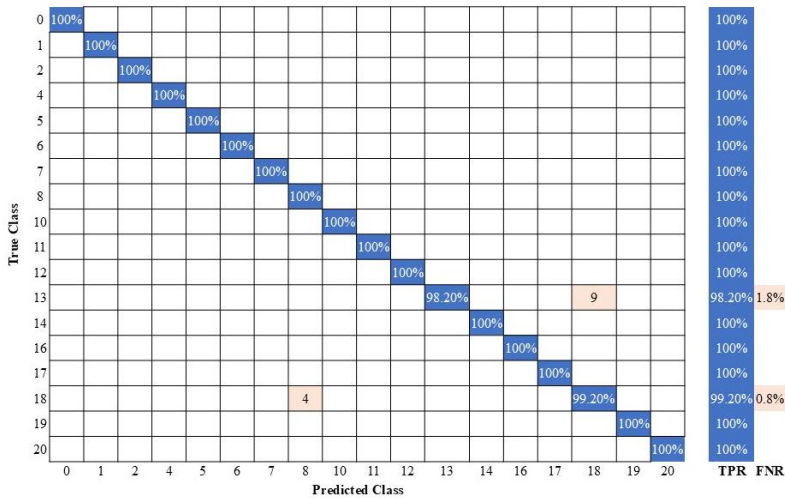


Figure 7. Confusion Matrix of LSTM

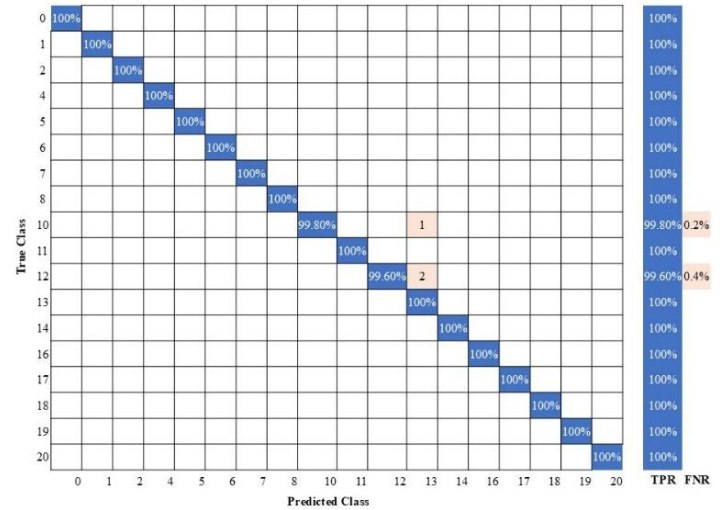


Figure 8. Confusion Matrix of CNN-LSTM

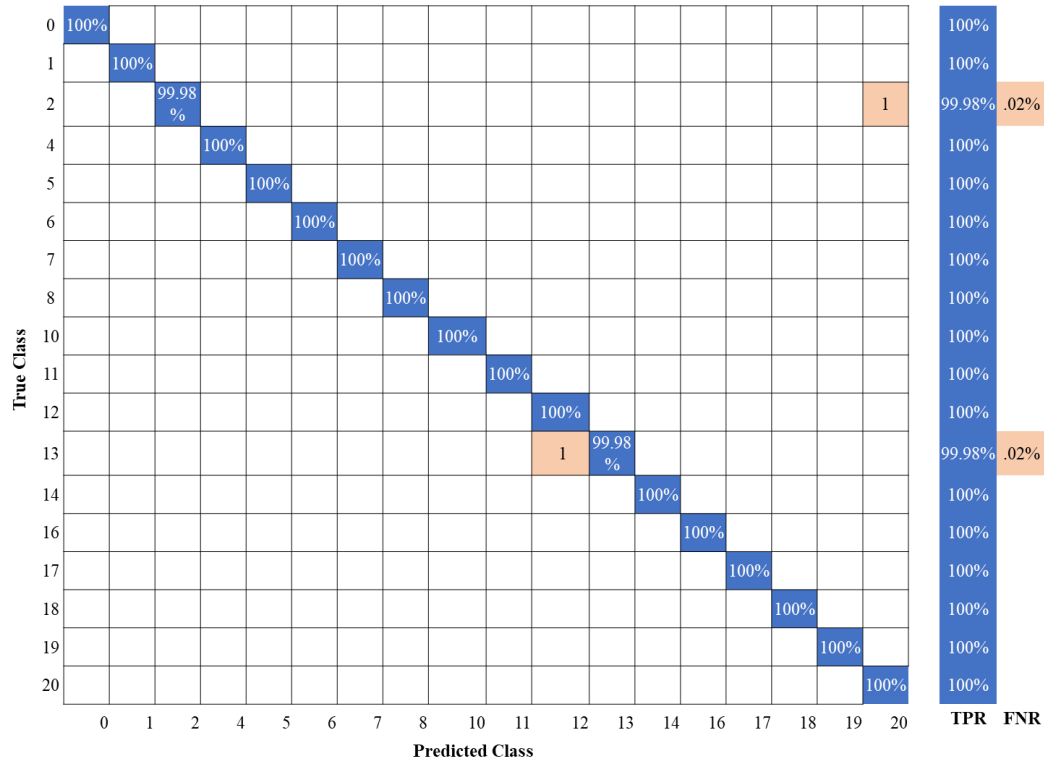


Figure 9. Confusion Matrix of CA-TCN

Confusion matrices are used to illustrate the error rates of the chosen model instances qualitatively. The particular class pairs that are confused in those runs appear in the off-diagonal entries. As a result, we use multi-seed accuracy and class-wise F1 summary as the main criteria for model comparison. (Han et al., 2020; Labbaf-Khaniki et al., 2024).

The confusion matrix displays a qualitative picture of the errors in the chosen runs. The CA-TCN matrix is highly diagonal-dominant. Still, a single confusion matrix is not taken as overall statistical superiority, since comparisons are based on aggregate results across multiple training seeds. Thus, model comparisons remain largely based on seed-level and class-wise summaries.

The t-SNE visualisation provides a qualitative projection of the learned representations onto 2D. The differences in apparent cluster compactness or separation are treated as descriptive only; the geometry of t-SNE is not used as statistical evidence that CA-TCN gives superior class separability or greater robustness to process noise.

Figure 10 presents a qualitative two-dimensional t-SNE projection of the latent representations obtained from the LSTM and CA-TCN models. Points are coloured

according to the 18 evaluated class labels. The visualization is used only for qualitative inspection.

4.4 Latent-Channel Attention and Interpretation

The Channel Attention Layer deals with an issue. As shown in Figure 10, the Attention Weights (α) learned from the TCN block vary across the 64 latent feature channels. The x-axis of this Feature Index corresponds to these 64 learned channels, not the 52 raw input sensors.

The attention mechanism functions as a continuous gating mechanism, multiplying features by weights ranging from 0 to 1 (rather than acting as a binary threshold selection). Channels that exhibit significant attention weights (e.g., >0.15) capture the most discriminative temporal patterns associated with critical physical variables, such as Reactor Pressure and Stripper Level. On the other hand, channels that process constant setpoints or uncorrelated noise receive near-zero weights.

The findings indicate that CA-TCN is not simply memorizing patterns but is actively focusing on the physical variables driving the dynamics of the process. This feature is useful to the operator, as it identifies which attributes contribute most to a failure diagnosis.

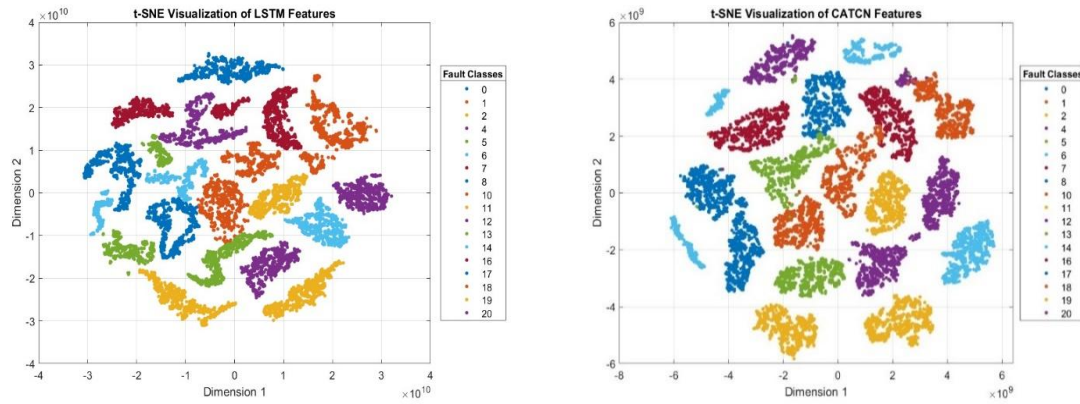


Figure 10. Qualitative t-SNE projection of LSTM and CA-TCN latent representations into two dimensions.

Table 6. Contextual comparison of reported accuracy metrics in TEP studies; values should not be directly comparable head-to-head results because evaluation protocols differ

Method	Metric	Model Type	Reported Performance	Source
PCA	Detection Rate	Linear Statistical	61.77% (T ²), 74.72% (Q)	(Yin et al., 2012)
DPCA + SVM	Detection Rate	Linear + Classifier	72.35%	(Agarwal et al., 2020, 2021)
ICA-SC	Avg Accuracy	Independent Component	≈90%	(C. Zhang et al., 2021)
DSAE	Accuracy	Deep Autoencoder	91.55–93.23%	(Agarwal et al., 2020, 2021; H. Chen et al., 2022; L. Zhang et al., 2022)
DDSAE (lag1)	Accuracy	Dynamic Supervised AE	93.96–95.85%	(Agarwal et al., 2020, 2021)
DDSAE (lag2)	Accuracy	Dynamic Supervised AE	93.50–96.43%	(Agarwal et al., 2020, 2021)
DLSAE Diagnosis	Accuracy	Supervised AE	≈88.41%	(Agarwal et al., 2020, 2021)
CNN-Transformer	Accuracy	Hybrid DL	99.92%	(Pal et al., 2026)
Plain TCN	Accuracy	Temporal convolutional	99.8756 ± 0.0807%	This work
CA-TCN (TCN+SE)	Accuracy	Temporal convolutional + SE attention	99.8511 ± 0.1104%	This work
LSTM	Accuracy	Recurrent DL	99.1711 ± 0.9467%	This work
CNN-LSTM	Accuracy	Hybrid CNN–recurrent DL	99.8444 ± 0.0820%	This work

Table 7. Contextual comparison of reported F1/detection metrics in TEP studies; differences in datasets, fault subsets and evaluation protocols limit direct numerical comparison

Method	Metric	Reported Performance	Source
CNN (Chadha & Schwung)	F1-score (per fault)	50–100%	(Chadha & Schwung, 2017)
Temporal CNN1D2D (GAN)	TPR / FPR	Severe faults: 100% TPR, 0% FPR	(Lomov et al., 2021)
Nonlinear SVM	Avg F1	≈78%	(Onel et al., 2019)
BiLSTM–Attention (tanh)	Avg F1	≈87%	(Zhao et al., 2025)
Twin Transformer (all faults)	Avg F1	94%	(LabbaF-Khaniki et al., 2024)
Twin Transformer (no incipient faults)	Avg F1	97%	(LabbaF-Khaniki et al., 2024)
CNN-Transformer	Macro F1	>0.999	(Pal et al., 2026)
Plain TCN	Macro F1	99.8757 ± 0.0804%	This work
CA-TCN	Macro F1	99.8512 ± 0.1101%	This work
LSTM	Macro F1	99.1675 ± 0.9536%	This work
CNN-LSTM	Macro F1	99.8443 ± 0.0820%	This work

Note: Performance values are retained in the units reported by the cited studies

Table 8. Post-hoc exploratory statistical summary of the matched TCN/CA-TCN assessment

Quantity	Result
Matched seed pairs	5
Mean CA-TCN – TCN accuracy difference	−0.0244 pp
SD of paired differences	0.170764 pp
Approx. 95% paired-t interval	[−0.2365, +0.1876] pp
Paired t-statistic	t(4) = −0.3201
Two-sided paired t-test	p = 0.7649
Exact two-sided sign-flip sensitivity	p = 0.8750
Interpretation	No statistically detectable difference;

Note: Post hoc tests, conducted after the outcomes are known, are therefore interpreted as exploratory rather than as predeclared confirmatory evidence.

4.5 Comparative Positioning Within Literature

Tables 6 and 7 place the results of the present research in the context of previously reported TEP fault classification and detection methods. These comparisons are intended for contextual reference rather than direct ranking because the cited studies differ in dataset variants, fault subsets, simulation duration, sequence construction, train/test partitions, sampling procedures, and metric definitions (Agarwal et al., 2021).

Within the present research, plain TCN, CA-TCN, LSTM, and CNN-LSTM are evaluated using the same data partitions, preprocessing procedure, and principal evaluation conditions. The controlled four-model comparison in Table 3 and the matched plain-TCN/CA-TCN ablation in Table 9, therefore, provide the primary evidence for the conclusions drawn in this work. The protocol used here improves comparability among the models evaluated within this research, but does not standardize the experimental protocols used in the cited literature (Agarwal et al., 2020; H. Chen et al., 2022; Chiang et al., 2001; Heo & Lee, 2018, 2019; Yin et al., 2012).

4.6 Statistical Analysis and Interpretation

The primary four-model results are reported as the mean $\pm$ standard deviation across five predefined training seeds. These values describe the observed variability associated with repeated model training on the same fixed test partition; they are not confidence intervals for performance on new plants, independently sampled datasets, or different process domains.

In the primary comparison, CA-TCN achieved a mean accuracy of $99.8511 \pm 0.1104\%$, while CNN-LSTM achieved $99.8444 \pm 0.0820\%$, corresponding to a numerical difference of 0.0067 percentage points. The previously reported statistical comparison between CA-TCN and CNN-LSTM yielded ($p > 0.05$). Accordingly, this small numerical difference is treated as descriptive and not interpreted as evidence that CA-TCN is statistically superior to CNN-LSTM - no explanation based on a presumed ceiling effect is inferred from the nonsignificant result.

For the dedicated matched plain-TCN/CA-TCN ablation, an additional post-hoc exploratory paired analysis is performed across the five matched assessment seeds. The mean CA-TCN-minus-TCN accuracy difference is -0.0244 percentage points, with an approximate 95% paired-t interval of $[-0.2365, +0.1876]$ percentage points, $t(4) = -0.3201$, and two-sided $p = 0.7649$. An exact two-sided sign-flip sensitivity calculation yielded $p = 0.8750$. Neither analysis provides a statistically detectable difference in accuracy

under the evaluated conditions. However, these nonsignificant results do not establish equivalence, non-inferiority, or the absence of a possible SE effect, particularly given the small number of matched seed pairs.

The statistical unit for the paired analysis is the matched training-seed pair ($n = 5$). The 9,000 test trajectories are reused across the training seeds and therefore not treated as 45,000 independent observations. The statistical conclusions are consequently limited to training-run variability under the evaluated TEP partition and experimental protocol. The exploratory paired statistics are summarized in Table 8.

4.7 Direct Ablation of SE Channel Attention

A separate bounded experiment is conducted to isolate the incremental contribution of the SE channel-attention module from that of the residual TCN backbone. Plain TCN and CA-TCN (= TCN+SE) are evaluated using five matched assessment seeds (307, 401, 503, 601, and 701). Within each seed pair, the models used the same trajectory partition, training budget, shared-backbone initialization, and model-selection rule.

Across the five matched assessment seeds, plain TCN achieved a mean test accuracy of $99.8000 \pm 0.1199\%$, whereas CA-TCN achieved $99.7756 \pm 0.1823\%$. CA-TCN achieved the higher accuracy in three seed pairs and plain TCN in two; however, the mean paired CA-TCN-minus-TCN difference was -0.0244 percentage points. The direction and magnitude of the differences varied among seeds; therefore, the results do not establish a reproducible advantage in accuracy attributable to the SE module. They also neither establish equivalence nor imply that channel attention is universally detrimental.

The plain TCN contains 51,666 trainable parameters, whereas CA-TCN contains 55,858, corresponding to 4,192 additional parameters introduced by the SE module. In the same bounded assessment, mean synchronized training time was 194.54 ± 4.97 s for plain TCN and 228.59 ± 5.92 s for CA-TCN over the common 30-epoch budget, corresponding to 6.485 and 7.620 s per completed epoch, respectively. Thus, adding SE increased the measured mean training time by approximately 17.5% and the parameter count by 8.11% in the evaluated implementation. These measurements are hardware- and implementation-specific and are reported only to characterize the computational overhead associated with SE.

The individual paired outcomes are reported in Table 9 and visualized in Figure 11.

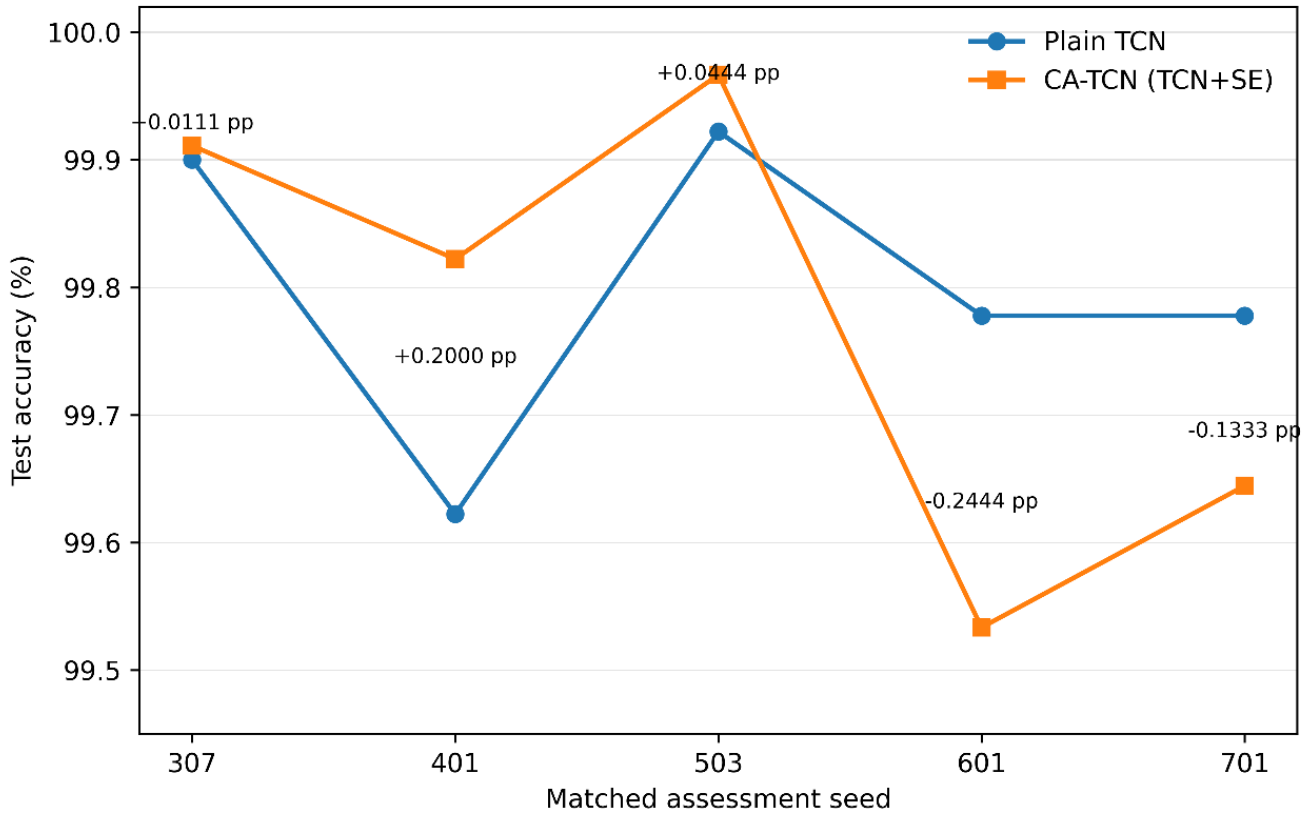


Figure 11. Paired test accuracy of plain TCN and CA-TCN across the five matched assessment seeds. Each line connects models evaluated with the same seed, trajectory partition, training budget, shared-backbone initialization, and model-selection rule. The paired outcomes do not show a consistent accuracy advantage from adding SE attention

Table 9. Paired Study Outcome

Seed	TCN accuracy (%)	CA-TCN accuracy (%)	CA-TCN – TCN (pp)
307	99.9000	99.9111	+0.0111
401	99.6222	99.8222	+0.2000
503	99.9222	99.9667	+0.0444
601	99.7778	99.5333	–0.2444
701	99.7778	99.6444	–0.1333
Mean	99.8000	99.7756	–0.0244

4.8 Scope, Generalization, and Deployment Limitations

All empirical evaluations in this article are conducted on the Tennessee Eastman Process, using the fixed-trajectory

partitions and fault classes described in Section 3.1. Accordingly, the reported classification performance, statistical comparisons, and computational measurements characterize the evaluated TEP setting.

The fully convolutional architecture accommodates the 500-sample training/validation trajectories and the 960-sample test trajectories through temporal global pooling. This compatibility with different trajectory lengths within the evaluated TEP protocol. Performance under different process dynamics, sensor configurations, fault taxonomies, sampling conditions, noise characteristics, or operating regimes was not examined in the present research.

Similarly, the inference-time measurements in Table 4 characterize fixed-input offline execution in the recorded MATLAB/GPU environment. The research is not evaluating streaming inference, fault-onset detection delay, continuous online monitoring, or deployment on an operating industrial plant. Therefore, real-time applicability is regarded as a direction for future evaluation rather than a capability.

Future work will evaluate plain TCN and CA-TCN on additional multivariate industrial benchmarks and, where available, on independently collected real-process datasets. Such studies are required before conclusions can be drawn regarding cross-domain robustness, transferability, streaming fault detection, or real-time industrial deployment.

5. Conclusion

This research evaluated a residual Temporal Convolutional Network with and without Squeeze-and-Excitation channel attention for offline TEP trajectory classification. Both plain TCN and CA-TCN achieved high classification accuracy under the evaluated protocol. However, the controlled multi-seed comparison and matched ablation do not establish a reproducible or statistically detectable accuracy advantage from adding SE attention. Similarly, the small numerical difference between CA-TCN and CNN-LSTM is not interpreted as statistical superiority.

Adding the SE module increased the trainable parameter count from 51,666 to 55,858 and increased measured training time in the matched implementation. In return, the SE block provides explicit adaptive reweighting of 64 learned latent feature channels. These coefficients provide insight into the internal channel-recalibration process but do not constitute validated sensor-level attribution or a causal explanation of the fault. Under the evaluated conditions, the principal contribution of SE is therefore an additional latent-channel reweighting mechanism with measurable computational overhead, rather than an improvement in accuracy.

The conclusions of this article are limited to the evaluated TEP benchmark and offline whole-trajectory classification protocol. Cross-domain robustness, streaming fault detection, fault-onset latency, and real-time industrial deployment were not evaluated. Future work will examine these questions using additional multivariate industrial datasets, independently defined evaluation protocols, and online or streaming fault-monitoring settings.

References:

- Agarwal, P., Gonzalez, J. I. M., Elkamel, A., & Budman, H. (2022). Hierarchical Deep LSTM for Fault Detection and Diagnosis for a Chemical Process. *Processes*, 10(12). <https://doi.org/10.3390/pr10122557>
- Agarwal, P., Tamer, M., & Budman, H. (2020). Assessing observability using supervised autoencoders with an application to the Tennessee Eastman process. *IFAC-PapersOnLine*, 53(2), 206–211. <https://doi.org/10.1016/j.ifacol.2020.12.122>
- Agarwal, P., Tamer, M., & Budman, H. (2021). *Explainability: Relevance based Dynamic Deep Learning Algorithm for Fault Detection and Diagnosis in Chemical Processes*. <https://doi.org/10.1016/j.compchemeng.2021.107467>
- Bai, S., Kolter, J. Z., & Koltun, V. (2018). *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling* (arXiv:1803.01271). arXiv. <https://doi.org/10.48550/arXiv.1803.01271>
- Chadha, G. S., & Schwung, A. (2017). Comparison of deep neural network architectures for fault detection in Tennessee Eastman process. *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–8. <https://doi.org/10.1109/ETFA.2017.8247619>
- Chen, B., Zhang, H., He, F., Zhang, C., Chen, Y., Liao, H., & Zheng, S. (2023). Modified Q- σ Rule for Incipient Fault Detection in Industrial Processes on Analysis of Intermittent Process Variation. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.4375935>
- Chen, H., Cen, J., Yang, Z., Si, W., & Cheng, H. (2022). Fault Diagnosis of the Dynamic Chemical Process Based on the Optimized CNN-LSTM Network. *ACS Omega*, 7(38), 34389–34400. <https://doi.org/10.1021/acsomega.2c04017>
- Chiang, L. H., Russell, E. L., & Braatz, R. D. (2001). *Fault Detection and Diagnosis in Industrial Systems*. Springer London. <https://doi.org/10.1007/978-1-4471-0347-9>
- Downs, J.J., Vogel, E.F., 1993 A plant-wide industrial process control problem *Comput. Chem. Eng.* 17 (3), 245–255 [https://doi.org/10.1016/0098-1354\(93\)80018-I](https://doi.org/10.1016/0098-1354(93)80018-I)
- Han, Y., Ding, N., Geng, Z., Wang, Z., & Chu, C. (2020). An optimized long short-term memory network based fault diagnosis model for chemical processes. *Journal of Process Control*, 92, 161–168. <https://doi.org/10.1016/j.jprocont.2020.06.005>
- Heo, S., & Lee, J. H. (2018). Fault detection and classification using artificial neural networks. *IFAC-PapersOnLine*, 51(18), 470–475. <https://doi.org/10.1016/j.ifacol.2018.09.380>
- Heo, S., & Lee, J. H. (2019). Statistical process monitoring of the Tennessee Eastman process using parallel autoassociative neural networks and a large dataset. *Processes*, 7(7). <https://doi.org/10.3390/pr7070411>
- Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. *Neural Computation* 9. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Hu, J., Shen, L., & Sun, G. (2018). *Squeeze-and-Excitation Networks*. <https://doi.org/10.48550/arXiv.1709.01507>
- Labba-Khaniki, M. A., Manthouri, M., & Ajami, H. (2024). *Twin Transformer using Gated Dynamic Learnable Attention mechanism for Fault Detection and Diagnosis in the Tennessee Eastman Process*. <https://doi.org/10.48550/arXiv.2403.10842>
- Lea, C., Vidal, R., Reiter, A., & Hager, G. D. (2016). *Temporal Convolutional Networks: A Unified Approach to Action Segmentation* (arXiv:1608.08242). arXiv. <https://doi.org/10.48550/arXiv.1608.08242>
- Li, M., & Shao, Y. (2021). *Deep Compression of Neural Networks for Fault Detection on Tennessee Eastman Chemical Processes*. <http://arxiv.org/abs/2101.06993>
- Lomov, I., Lyubimov, M., Makarov, I., & Zhukov, L. E. (2021). Fault detection in Tennessee Eastman process with temporal deep learning models. *Journal of Industrial Information Integration*, 23, 100216. <https://doi.org/10.1016/j.jii.2021.100216>
- Onel, M., Kieslich, C. A., & Pistikopoulos, E. N. (2019). A nonlinear support vector machine-based feature selection approach for fault detection and diagnosis: Application to the Tennessee Eastman process. *AIChE Journal*, 65(3), 992–1005. <https://doi.org/10.1002/aic.16497>
- Pal, P. K., Ray, S., Behera, N., Chowdhury, S., Hens, A., & Lahiri, S. K. (2026). Compact CNN-Transformer Model for Robust Fault Diagnosis in Large-Scale Chemical Processes. *International Journal of Prognostics and Health Management*, 17(1). <https://doi.org/10.36001/ijphm.2026.v17i1.4744>
- Reinartz, C., Kulahci, M., & Ravn, O. (2021). An extended Tennessee Eastman simulation dataset for fault-detection and decision support systems. *Computers and Chemical Engineering*, 149. <https://doi.org/10.1016/j.compchemeng.2021.107281>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is All you Need*.
- Verma, R., Yerolla, R., & Besta, C. S. (2022). Deep Learning-based Fault Detection in the Tennessee

- Eastman Process. *Proceedings of the 2nd International Conference on Artificial Intelligence and Smart Energy, ICAIS 2022*, 228–233. <https://doi.org/10.1109/ICAIS53314.2022.9743021>
- Wang, J., Li, Y., Gao, R. X., & Zhang, F. (2022). Hybrid physics-based and data-driven models for smart manufacturing: Modelling, simulation, and explainability. *Journal of Manufacturing Systems*, 63, 381–391. <https://doi.org/10.1016/j.jmsy.2022.04.004>
- Woo, S., Park, J., Lee, J.-Y., & Kweon, I. S. (2018). CBAM: Convolutional Block Attention Module. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer Vision – ECCV 2018* (Vol. 11211, pp. 3–19). Springer International Publishing. https://doi.org/10.1007/978-3-030-01234-2_1
- Yin, S., Ding, S. X., Haghani, A., Hao, H., & Zhang, P. (2012). A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process. *Journal of Process Control*, 22(9), 1567–1581. <https://doi.org/10.1016/j.jprocont.2012.06.009>
- Yin, S., Gao, X., Karimi, H. R., & Zhu, X. (2014). Study on support vector machine-based fault detection in Tennessee Eastman process. *Abstract and Applied Analysis*, 2014. <https://doi.org/10.1155/2014/836895>
- Zhang, C., Guo, Q., & Li, Y. (2020). Fault detection in the Tennessee eastman benchmark process using principal component difference based on k-nearest neighbors. *IEEE Access*, 8, 49999–50009. <https://doi.org/10.1109/ACCESS.2020.2977421>
- Zhang, C., Zheng, X., & Li, Y. (2021). A novel fault detection and diagnosis scheme based on independent component analysis-statistical characteristics: Application on the tennessee eastman benchmark process. *Journal of Chemical Engineering of Japan*, 54(6), 304–312. <https://doi.org/10.1252/jcej.20we045>
- Zhang, H., Liu, Q., Chen, X., Zhang, D., & Leng, J. (2017). A digital twin-based approach for designing and multi-objective optimization of hollow glass production line. *IEEE Access*, 5, 26901–26911.
- Zhang, L., Song, Z., Zhang, Q., & Peng, Z. (2022). Generalized transformer in fault diagnosis of Tennessee Eastman process. *Neural Computing and Applications*, 34(11), 8575–8585. <https://doi.org/10.1007/s00521-021-06711-2>
- Zhao, S., Duan, Y., Roy, N., & Zhang, B. (2025). A novel fault diagnosis framework empowered by LSTM and attention: A case study on the Tennessee Eastman process. *The Canadian Journal of Chemical Engineering*, 103(4), 1763–1785. <https://doi.org/10.1002/cjce.25460>

BIOGRAPHIES

A. Pratyush is currently a Sr. Technical Officer at CSIR CMERI, Durgapur, India. He did his M.Sc. in Computer Science and B.Sc. in Computer Application, and is Currently Pursuing a PhD.

B. Sumana completed her M.Tech in Microelectronics and VLSI and is currently pursuing a PhD.

C. Narottam is a metallurgical engineering professional with over 24 years of experience in steel industry Operations, R&D, quality assurance, and major project execution. He holds Bachelor's and Master's degrees in Metallurgical Engineering from NIT Warangal and a Master's degree from IIT (BHU) Varanasi, and is currently pursuing a PhD.

D. Somnath is currently working as a Technical Services Engineer at Scientific Design LLC in Dubai, United Arab Emirates. He did his M.Tech in Chemical Engineering and is currently pursuing a PhD

E. Abhiram is currently a faculty member in the Department of Chemical Engineering of the National Institute of Technology Durgapur, India. He did his B.Tech. and M.Tech. in Chemical Engineering before completing a PhD in Engineering Science. His research area includes process modelling and simulation using computational fluid dynamics and molecular dynamics. Recently, he began working in AI/ML-based process optimization.

F. Sandip is currently a faculty member in the Department of Chemical Engineering of the National Institute of Technology Durgapur, India. He did his M.Tech in Chemical Engineering and PhD from NIT Durgapur. He was awarded the title of Innovation Ambassador by the Ministry of Education, Government of India.

Supplementary Information

This supplementary provides additional reproducibility and robustness analyses supporting the main manuscript.

Table S1. Summary of fault scenarios in the Tennessee Eastman Process (Faults 3, 9 and 15 are typically excluded from quantitative benchmarking because of their weak statistical signatures (Downs and Vogel, 1993; Agarwal et al., 2021; Chiang et al., 2001; Reinartz et al., 2021)).

Fault ID	Process variable or unit affected	Fault type	Typical description	Detectability
1	A/C feed ratio, B composition constant (stream 4)	Step change	Change in reactant feed ratio causing composition imbalance	High
2	B composition, A/C ratio constant (stream 4)	Step change	Shift in B feed composition	High
3	D feed temperature (stream 2)	Step change	Small change in feed temperature	Low (incipient)
4	Reactor cooling water inlet temperature	Step change	Cooling disturbance affecting reactor temperature control	High
5	Condenser cooling water inlet temperature	Step change	Cooling disturbance in the condenser loop	High
6	A feed loss (stream 1)	Step change	Reduction or loss of A feed stream	High
7	C header pressure loss (stream 4)	Step change	Reduced availability of C feed gas	High
8	A/B/C feed composition (stream 4)	Random variation	Fluctuating feed composition	Moderate
9	D feed temperature (stream 2)	Random variation	Small random thermal disturbance	Low (incipient)
10	C feed temperature (stream 4)	Random variation	Temperature fluctuations in the C feed	Moderate
11	Reactor cooling water inlet temperature	Random variation	Cooling water temperature fluctuation	Moderate
12	Condenser cooling water inlet temperature	Random variation	Variation in condensing temperature	Moderate
13	Reaction kinetics	Slow drift	Gradual catalyst deactivation or reaction-rate change	High
14	Reactor cooling water valve	Sticking fault	Actuator stiction affecting heat removal	Moderate
15	Condenser cooling water valve	Sticking fault	Valve stiction in the condensing loop	Low (incipient)
16	Unknown disturbance 1	Unknown	Unspecified composite fault	Moderate
17	Unknown disturbance 2	Unknown	Unspecified process disturbance	Moderate

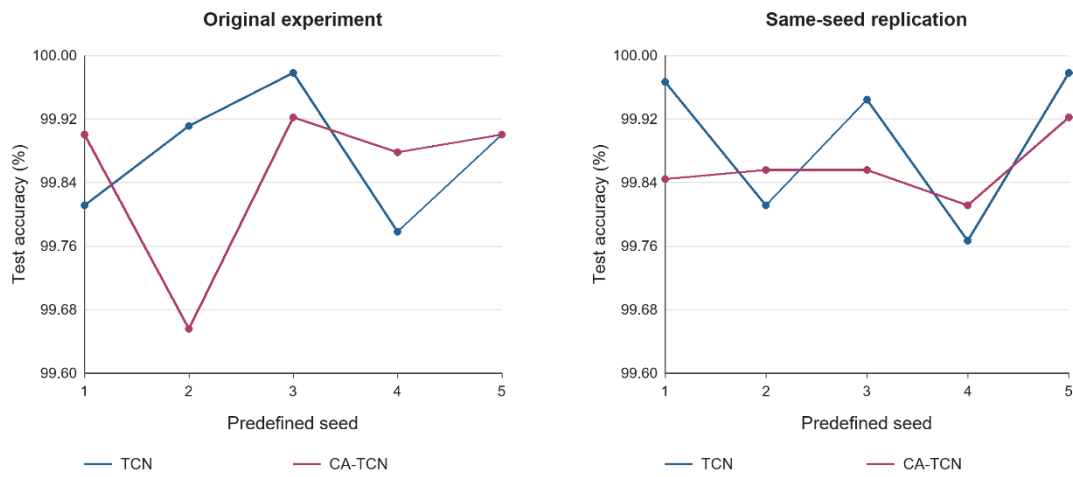
18	Unknown disturbance 3	Unknown	Unspecified process disturbance	Moderate
19	Unknown disturbance 4	Unknown	Unspecified process disturbance	Moderate
20	Unknown disturbance 5	Unknown	Unspecified process disturbance	Moderate
21	Valve or instrument bias	Bias fault	Sensor drift or bias in measurement	High

Note: Faults 3, 9, and 15 excluded under the fixed benchmark protocol; Fault 21 was not available in the supplied quantitative data tables used in this study.

Table S2. Key measured (XMEAS) and manipulated (XMV) variables in the Tennessee Eastman Process (Variable designations follow the original notation by (Downs and Vogel, 1993; Agarwal et al., 2021; Reinartz et al., 2021).

Variable ID	Description	Variable ID	Description
XMEAS 1	A feed (flow)	XMEAS 22	Stripper steam flow
XMEAS 2	D feed (flow)	XMEAS 23	Stripper level
XMEAS 3	E feed (flow)	XMEAS 24	Stripper pressure
XMEAS 4	A and C feed (flow)	XMEAS 25	Stripper temperature
XMEAS 5	Recycle flow	XMEAS 26	Stripper steam temperature
XMEAS 6	Reactor feed rate	XMEAS 27	Condenser cooling water outlet temperature
XMEAS 7	Reactor pressure	XMEAS 28	Condenser level
XMEAS 8	Reactor level	XMEAS 29	Separator temperature
XMEAS 9	Reactor temperature	XMEAS 30	Separator level
XMEAS 10	Purge rate	XMEAS 31	Compressor work
XMEAS 11	Product flow	XMEAS 32	Product cooling water flow
XMEAS 12	Separator pressure	XMEAS 33	Product temperature
XMEAS 13	Separator level	XMEAS 34	Recycle temperature
XMEAS 14	Separator temperature	XMEAS 35	Cooling water inlet temperature
XMEAS 15	Stripper pressure	XMEAS 36	Cooling water outlet temperature
XMEAS 16	Stripper level	XMEAS 37	Process gas molecular weight
XMEAS 17	Stripper temperature	XMEAS 38	Purge gas composition (A)
XMEAS 18	Compressor work	XMEAS 39	Purge gas composition (B)
XMEAS 19	Reactor cooling water outlet temperature	XMEAS 40	Product composition (G)

XMEAS 20	Condenser cooling water outlet temperature	XMEAS 41	Product composition (H)
XMV 1	D feed flow	XMV 7	Purge rate
XMV 2	E feed flow	XMV 8	Compressor recycle valve
XMV 3	A feed flow	XMV 9	Condenser cooling water valve
XMV 4	A and C feed flow	XMV 10	Separator cooling water valve
XMV 5	Recycle valve	XMV 11	Stripper steam valve
XMV 6	Reactor cooling water valve		



Supplementary Figure S1: The same five predefined seeds repeated in the batches and therefore not interpreted as ten independent seed runs.