

Article

ALSTMResNet: Active-Learning-Enhanced LSTMResNet as an Efficiency-Oriented Training Framework for Well Anomaly Monitoring Under Partial Labels

Feng Ge ¹, Zhi Yang ², Fan Yu ², Honglin Xiao ³, Yang Peng ², Ji Li ⁴, Yan Chen ^{4,5}, Yu Fang ^{4,6,*}  and Ke Luo ³

¹ PetroChina Southwest Oil & Gasfield Company, Chengdu 610051, China; gefeng@petrochina.com.cn

² Research Institute of Oil and Gas Process, PetroChina Southwest Oil & Gasfield Company, Chengdu 610017, China; yangzhi008@petrochina.com.cn (Z.Y.); fan@petrochina.com.cn (F.Y.); pengyang@petrochina.com.cn (Y.P.)

³ Tight Oil and Gas Exploration and Development Project Department, PetroChina Southwest Oil & Gasfield Company, Chengdu 610051, China; xiaohl2018@petrochina.com.cn (H.X.); luoke@petrochina.com.cn (K.L.)

⁴ School of Computer Science and Software Engineering, Southwest Petroleum University, Chengdu 610500, China; jli727753@gmail.com (J.L.); carly_chen@126.com (Y.C.)

⁵ Engineering Research Center for Intelligent Oil & Gas Exploration and Development of Sichuan Province, Southwest Petroleum University, Chengdu 610500, China

⁶ School of Computer and Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756, China

* Correspondence: fangyu@swpu.edu.cn

Abstract

Oil-well anomaly monitoring supports safe and efficient oil-and-gas production, but delayed recognition of abnormal operating states can reduce lifting efficiency, trigger costly interventions, and increase operational risk. Existing data-driven detectors are also vulnerable to optimistic estimates when segmentation, normalization, and train–test splitting leak information across source files. This study presents ALSTMResNet, an application-oriented active-learning workflow built on an LSTMResNet backbone for one-step-ahead anomaly screening under partial labels. The workflow converts real 3W well records into leakage-aware file-wise splits, train-only standardized sliding windows, and a fixed 15-channel representation that combines process measurements with temporal covariates; active learning is used only during training to select additional labels while leaving the deployed backbone unchanged. Experiments on the retained benchmark split obtain an F_1 -score of 0.9354, and five repeated file-wise trials give an average F_1 -score of 0.8661 ± 0.0666 while reducing retraining time relative to the full-data backbone. Label-budget and acquisition-policy analyses show that the workflow remains competitive under constrained labels, although uncertainty, entropy, margin, least-confidence, and random querying have limited separation in the present binary setting. These results indicate that ALSTMResNet can support cost-aware oil-well anomaly monitoring when labels are partially available and retraining resources are constrained.



Academic Editors: Guoqing Han and Dicho Stratiev

Received: 3 June 2026

Revised: 29 June 2026

Accepted: 5 July 2026

Published: 23 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: anomaly monitoring; well monitoring; multivariate time series; active learning; industrial artificial intelligence

1. Introduction

Anomaly detection in oil wells is a critical task in intelligent production surveillance, as incipient faults, abnormal flow regimes, and sensor disturbances can escalate into production interruptions, costly interventions, and safety hazards if they are not detected

promptly. Existing studies on oil-well monitoring have shown that these systems generate long multivariate time series characterized by heterogeneous events [1,2], making anomaly detection both practically significant and methodologically challenging [3,4]. In such settings, an effective monitoring model must capture complex temporal dependencies and nonlinear interactions among variables while remaining practical in terms of labeling requirements and deployment cost.

The engineering context is central to this problem. Oil-well monitoring systems do not produce isolated, independent samples; rather, they generate continuous operational records in which normal behavior, transient fluctuations, and abnormal events are intrinsically intertwined. Consequently, reported performance often depends as much on data preprocessing, including segmentation, normalization, and splitting, as on the classifier itself. Ignoring these factors may lead to an overestimation of model performance, particularly when data from the same source file leak across training and testing subsets or when preprocessing policies are inconsistent across comparative baselines.

Industrial deployment introduces a further constraint: labels are expensive. Although unlabeled samples are abundant, identifying whether a segment is truly anomalous often requires expert interpretation of logs and event histories. Therefore, the practical bottleneck is annotation rather than data acquisition. Pool-based active learning (AL) is well suited to this setting because it aims to allocate limited labeling effort to the most informative samples rather than revealing labels at random [5,6]. Uncertainty-guided querying has also been widely studied as an effective strategy for selecting informative samples [7,8].

To address these challenges, this study integrates uncertainty-guided active learning into a long short-term memory residual network (LSTMResNet) backbone. While recurrent architectures capture sequential dependencies [9], convolutional residual pathways enhance local feature extraction and optimization stability [10]. Although recent developments have expanded Transformer-based approaches for time-series learning [11,12], this study intentionally focuses on a tightly controlled comparison set centered on the proposed ALSTMResNet, its backbone, and selected recurrent and classical baselines. The objective is to evaluate whether active learning provides a measurable advantage when combined with a hybrid backbone under a rigorous file-wise evaluation pipeline.

This focused design is essential because the benefits of active learning are often difficult to isolate. If selective querying is evaluated under inconsistent data splits or preprocessing policies, the reported improvements become ambiguous. Furthermore, for industrial monitoring, a method that reduces labeling burden and retraining effort remains valuable even if the absolute improvement in F_1 -score is modest.

Accordingly, the contributions of this study are as follows.

- (i) This study proposes ALSTMResNet, which couples uncertainty-guided pool-based active learning with an LSTMResNet detector under a file-wise split protocol designed to eliminate cross-file data leakage.
- (ii) It demonstrates that ALSTMResNet remains competitive while substantially reducing retraining time relative to a full-data backbone under a unified evaluation framework.
- (iii) It reports repeated file-wise Monte Carlo validation, acquisition-strategy controls, and masked-label learning-curve experiments to characterize stability, label-budget sensitivity, and the limited separation among uncertainty-guided, entropy, margin, least-confidence, and random querying under the present binary setting.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the proposed framework and mathematical formulation. Section 4 reports the experimental design, results, and limitations. Finally, Section 5 concludes the paper.

2. Related Work

2.1. Oil-Well Anomaly Monitoring and Industrial Time-Series Detection

This study is situated at the intersection of oil-well monitoring and industrial multivariate anomaly detection. The 3W dataset introduced by Vargas et al. [13] has become an important public benchmark for well-event analysis, and recent work has further investigated anomaly detection on its real-world subset using one-class classifiers and other monitoring baselines [1]. Beyond the petroleum domain, comparative studies on industrial anomaly detection have shown that reported performance is strongly affected by dataset selection, evaluation protocols, and the distinction between supervised and unsupervised settings [2,14,15]. In parallel, related oilfield studies suggest that predictive maintenance is gradually shifting from rule-based surveillance to time-aware, data-driven pipelines [16].

Conventional oil-well monitoring is often implemented through engineering thresholds, expert rules, and statistical process-monitoring tools. Threshold-based alarms on pressure, temperature, and flow-rate channels are easy to deploy and interpret, but they usually depend on manually tuned limits and can miss coupled multivariate changes. Statistical process-control methods, including control charts and principal-component representations, offer a more systematic way to detect distributional departures from normal operation, but their performance can degrade under nonlinear operating regimes, drifting production conditions, and rare abnormal classes. Classical machine learning detectors such as one-class SVM and Isolation Forest relax some of these assumptions and remain useful when labeled faults are scarce, but they usually depend on handcrafted window summaries and may not fully exploit temporal dynamics in long multivariate records [1,3,4]. These trade-offs motivate controlled comparisons between classical monitors and time-aware neural backbones under a leakage-aware split.

A representative method in this line of work is the one-class support vector machine (SVM), which estimates a compact support region for normal behavior by solving [17]

$$\min_{\mathbf{w}, \rho, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu n} \sum_{i=1}^n \xi_i - \rho \quad (1)$$

subject to

$$\mathbf{w}^\top \phi(\mathbf{x}_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad (2)$$

with decision function

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \phi(\mathbf{x}) - \rho), \quad (3)$$

where $\mathbf{w}$ is the normal vector of the separating hyperplane in the feature space $\phi(\mathbf{x})$, ρ is the offset, ξ_i are slack variables, ν controls the trade-off between boundary tightness and training errors, and n is the number of training samples.

This classical formulation remains relevant because it treats anomaly detection as boundary estimation under severely limited abnormal labels, which closely matches the practical challenges of oil-well monitoring. Fernandes et al. [1] reported that one-class formulations remain competitive on real-well data, while Lian et al. [2] showed that anomaly modeling in oil and gas systems increasingly uses richer temporal representations.

Taken together, these studies suggest two points that are particularly relevant here. First, oilfield monitoring increasingly requires data-driven methods that can capture complex event patterns. Second, meaningful comparison depends on a transparent and controlled benchmark, since performance rankings may change substantially with preprocessing choices and labeling assumptions. In multivariate time-series monitoring, data leakage commonly arises when overlapping temporal windows are distributed across training and test sets, or when records from the same well or file appear in both partitions. To address

these methodological pitfalls, this study implements a strict file-wise split and a unified evaluation pipeline to enable a more rigorous leakage-aware comparison.

2.2. Hybrid Sequence Backbones for Multivariate Time Series

Hybrid models remain attractive for multivariate time series because recurrent and convolutional components provide complementary inductive biases. Long short-term memory (LSTM) networks remain a standard choice for modeling long-range temporal dependencies [9], whereas residual convolutional pathways improve local feature extraction and optimization stability [10]. Existing studies include standard convolutional baselines [18], comprehensive reviews of deep learning for time-series classification [19], hybrid architectures such as LSTM-FCN and its multivariate variants [20,21], residual-enhanced LSTM variants [22], related residual-convolutional models for time-series classification [23], and hybrid long short-term memory residual network (LSTM-ResNet) formulations for structural damage detection [24]. In practice, the gating mechanism of LSTM networks allows the model to retain or discard historical information adaptively, which is valuable when abnormal well events evolve over extended time horizons. Residual and convolutional pathways complement this behavior by strengthening local pattern extraction and facilitating optimization in deeper temporal feature pipelines.

More recently, Transformer-based models have become increasingly influential in time-series learning. Representative forecasting-oriented architectures include Informer, Autoformer, and FEDformer [11,25,26]. Patch-based and inverted designs such as PatchTST and iTransformer have further extended this line [12,27]. Related multivariate Transformer formulations can also be traced to the original Transformer paradigm and subsequent multivariate variants [28,29]. Survey-level studies also highlight the growing importance of this model family for temporal modeling [30]. However, the objective of this study is not to identify the best temporal backbone, but to evaluate active learning under realistic industrial labeling constraints. Compared with many Transformer-based alternatives, LSTMResNet provides a well-established and computationally efficient hybrid architecture that combines recurrent modeling of temporal dependencies with residual convolutional extraction of local multivariate patterns. Its moderate model capacity also helps reduce the risk that observed gains are dominated by a substantially larger-capacity backbone rather than by the active-learning protocol under evaluation. Therefore, LSTMResNet is adopted as a representative backbone for isolating the contribution of active learning under a leakage-aware industrial benchmark. The same active-learning workflow can be evaluated with Transformer-based backbones such as PatchTST and iTransformer in subsequent studies.

2.3. Pool-Based Active Learning Under Partial Labels

Pool-based AL is well suited to industrial scenarios in which unlabeled samples are abundant but expert annotation is expensive. Early studies established the foundations of uncertainty-based querying [5,6]. More recent deep AL research has expanded toward Bayesian uncertainty and learned acquisition criteria [7,8]. Survey-level syntheses and batch-query strategies have further broadened the field [31,32]. These approaches typically balance uncertainty, diversity, and computational practicality, as illustrated by core-set selection, BADGE, and ensemble-based acquisition strategies [33–35]. In binary classification, a standard uncertainty-sampling rule selects samples whose predicted probabilities lie closest to the decision boundary. This boundary-distance view is particularly suitable for partially labeled well-monitoring data because it links sample acquisition directly to model uncertainty rather than relying on event-specific heuristics.

Nevertheless, an important gap remains. Existing AL studies predominantly focus on static image datasets and often overlook the temporal dependencies and file-level correlations inherent in multivariate industrial time series. To address this gap, the present study integrates uncertainty-guided AL with an LSTMResNet detector in a leakage-aware, file-wise framework and evaluates it under controlled backbone and labeling-budget settings.

3. Framework

3.1. Framework Overview

The proposed framework, termed ALSTMResNet, follows a single end-to-end path from raw input files to final anomaly decisions. First, real well-event CSV files are read and converted into a fixed 15-channel representation that combines seven normalized calendar covariates with eight aligned process measurements. Second, each file is transformed into overlapping sliding windows of length 96 with stride 10, and each window is assigned the binary label of the immediately following time step. Third, the resulting windows are partitioned by source file into training, validation, and test subsets, after which feature-wise standardization is fitted on the training split only and then applied to the remaining splits. Fourth, each standardized window is passed through the LSTMResNet backbone, which fuses temporal convolution, recurrent sequence modeling, and residual projection to produce an anomaly probability at the final time step. Finally, during training only, a pool-based active-learning loop selectively reveals additional labels from the unlabeled pool; at inference time, the deployed detector remains the same backbone with a validation-selected operating threshold. Figure 1 summarizes this end-to-end pipeline from file ingestion and window construction to backbone prediction, active-label acquisition, and threshold-based inference.

Each stage serves a distinct methodological role within the benchmark pipeline. The fixed input template guarantees a common feature space across heterogeneous well files, the one-step-ahead windowing rule turns continuous operational logs into a predictive monitoring task, the file-wise split and train-only normalization reduce leakage risk, and the active-learning loop targets labeling efficiency while keeping the deployed model family unchanged. The resulting framework therefore provides a controlled workflow that unifies preprocessing, representation learning, selective labeling, and binary decision-making.

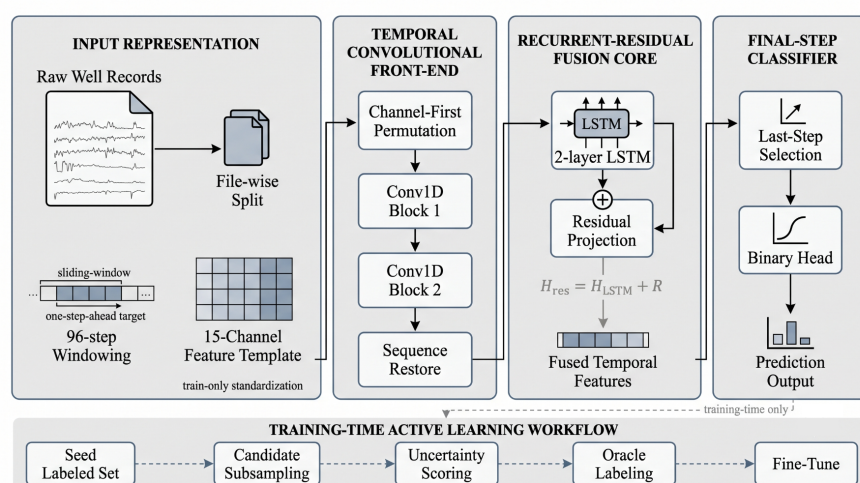


Figure 1. Architecture-level overview of ALSTMResNet. The figure shows the 96×15 input representation, the temporal convolutional front-end, the recurrent–residual fusion core, the final-step binary classifier, and the training-time active-learning workflow used to reveal additional labels while leaving the inference-time backbone unchanged. The end-to-end workflow is summarized in Figure 2.

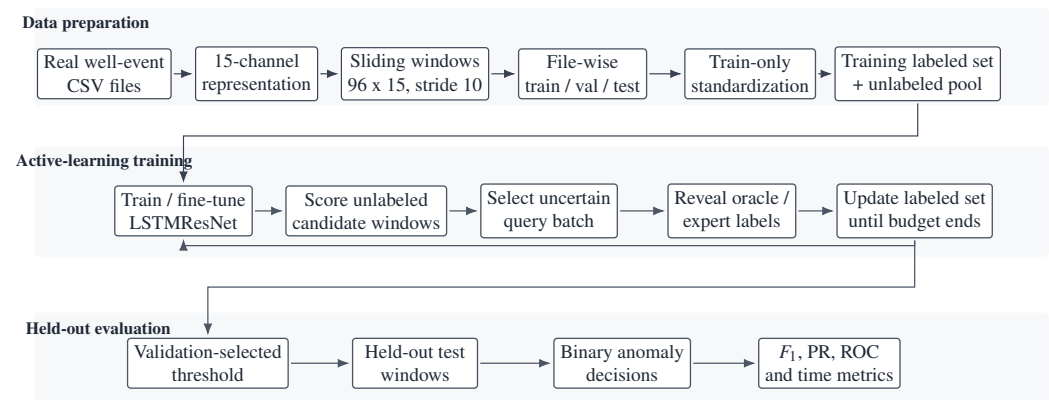


Figure 2. Flowchart of the ALSTMResNet training and evaluation workflow from raw real-well files to held-out test evaluation. The active-learning loop uses offline label masking, uncertainty-guided batch query selection, oracle/expert label reveal, and fine-tuning until the label budget is exhausted.

Table 1 lists the input variables used in the benchmark and their engineering interpretation. This representation is shared by all deep learning models, whereas the classical baselines operate on summary statistics extracted from the same windows. The process-variable abbreviations follow the original 3W dataset naming convention; in general, P and T denote pressure and temperature measurements, respectively. In practical deployment, the method requires synchronized multivariate production time series and timestamps; if a target field lacks some channels, the feature template and model would need to be recalibrated and revalidated on the available measurements.

Table 1. Input feature template and engineering interpretation used in the benchmark.

Variable	Type	Physical Meaning	Use for Well-Status Monitoring
hour, minute, second, weekday, day, month, year	Time-derived covariates	Calendar and clock descriptors extracted from the timestamp.	Encode operating cycles and temporal context so that process changes are not interpreted independently of time.
P-PDG	Process pressure	Downhole pressure measured by the permanent downhole gauge.	Indicates downhole pressure response and helps detect pressure transients associated with abnormal well conditions.
P-TPT	Process pressure	Tubing pressure measured at the tubing pressure-temperature point.	Reflects flow and lifting conditions in the tubing and supports detection of pressure build-up or drawdown changes.
T-TPT	Process temperature	Tubing temperature measured at the tubing pressure-temperature point.	Captures thermal changes that can accompany flow-regime shifts, sensor disturbances, or abnormal production states.
P-MON-CKP	Process pressure	Production-choke pressure measurement.	Describes pressure behavior around the production choke and supports identification of restrictions or unstable flow.
T-JUS-CKP	Process temperature	Production-choke temperature measurement.	Provides complementary thermal evidence around the production choke for distinguishing normal fluctuations from abnormal events.
P-JUS-CKGL	Process pressure	Gas-lift-choke pressure measurement.	Tracks gas-lift pressure behavior, which is relevant to lift stability and gas-injection-related anomalies.
T-JUS-CKGL	Process temperature	Gas-lift-choke temperature measurement.	Captures thermal behavior near the gas-lift choke and supports diagnosis of gas-lift operating changes.
QGL	Process flow rate	Gas-lift flow-rate measurement.	Indicates the injected gas-lift rate and helps relate pressure-temperature patterns to changes in lift assistance.

3.2. Input Representation, Window Construction, and Normalization

This subsection details the preprocessing stage of the framework, including fixed-template input construction, one-step-ahead window generation, and train-only normalization.

Each input sample is represented using a fixed 15-channel template. Seven channels encode normalized calendar covariates, whereas eight channels correspond to aligned process variables. Missing channels are inserted as zero-valued columns, the multivariate sequence is linearly interpolated, and any remaining missing or non-numeric values are set to zero. This design provides a common feature space for all files and allows the same deep backbone to operate across heterogeneous well records with no file-specific feature engineering.

The original event labels are converted into a binary target, where any class value greater than zero is mapped to anomaly. This binary conversion is applied before window generation and matches the practical monitoring objective considered in this study, namely determining whether an abnormal condition is emerging rather than identifying its specific subtype at the same stage. The original multiclass folder structure is retained during file selection to preserve coverage of the available abnormal source categories in the local dataset organization.

For a source file m with aligned multivariate sequence $\mathbf{Z}^{(m)} \in \mathbb{R}^{T_m \times F}$ and binary label vector $\mathbf{c}^{(m)} = (c_1^{(m)}, \dots, c_{T_m}^{(m)})$, windows of length $S = 96$ are constructed using stride $\Delta = 10$. For a valid start index i , the i -th window from file m is defined as

$$\mathbf{X}_i^{(m)} = \mathbf{Z}^{(m)}[i : i + S - 1, :], \quad (4)$$

with supervisory label

$$y_i^{(m)} = c_{i+S}^{(m)}, \quad (5)$$

where i denotes the starting time index of the window, $\mathbf{X}_i^{(m)}$ contains the consecutive observations from time steps i to $i + S - 1$, and $y_i^{(m)}$ is the binary anomaly label at the immediately following time step $i + S$.

Accordingly, the model observes the previous 96 time steps and predicts the class of the immediately subsequent time point. Windows that do not admit a valid next-step label are discarded during construction. The task is therefore formulated as a window-conditioned one-step-ahead anomaly-state classification.

The window length and stride are chosen to balance context coverage and computational tractability. Shorter windows may omit delayed interactions among pressure, temperature, and flow variables, whereas substantially longer windows increase runtime and may weaken the local temporal patterns targeted by the convolutional blocks. Likewise, a stride of 10 preserves substantial overlap between adjacent windows and avoids degenerating into near-duplicate samples generated by stride-1 extraction. Because windows are generated independently within each source file, the resulting samples inherit the file-wise split constraints and do not cross file boundaries.

After window generation, each split is reshaped to two dimensions for feature-wise standardization. The scaler is fitted on the training split only and then applied to the validation and test splits. The standardized value of feature f at window time index t is defined as

$$\tilde{z}_{t,f} = \frac{z_{t,f} - \mu_f^{\text{train}}}{\sigma_f^{\text{train}}}, \quad (6)$$

where $z_{t,f}$ denotes the original value of feature channel f at time step t within a window, $\tilde{z}_{t,f}$ denotes the corresponding standardized value, and μ_f^{train} and σ_f^{train} are the empirical mean and standard deviation of feature f estimated from the training split only. This

train-only standardization prevents leakage from validation or test statistics into model development and ensures that all downstream models receive an identical leakage-aware input space.

3.3. ALSTMResNet Backbone and Output Prediction

After preprocessing, each standardized window is passed to the LSTMResNet backbone for temporal representation learning and anomaly probability prediction.

Each sample is represented by a multivariate window

$$\mathbf{X} \in \mathbb{R}^{S \times F}, \quad (7)$$

where $S = 96$ denotes the sequence length and $F = 15$ denotes the number of features. The LSTMResNet backbone combines a temporal convolutional front-end, recurrent sequence modeling, and a residual linear projection. After channel-first permutation, the input is passed through two one-dimensional convolutional layers with ReLU activation,

$$\mathbf{C} = \phi_{\text{conv}}(\mathbf{X}), \quad (8)$$

where $\phi_{\text{conv}}(\cdot)$ denotes the stacked Conv1D-ReLU blocks. The resulting temporal representation is then processed by a two-layer LSTM,

$$\mathbf{H}_{\text{LSTM}} = \phi_{\text{LSTM}}(\mathbf{C}). \quad (9)$$

In parallel, the convolutional output is projected into the recurrent hidden space through a residual linear layer,

$$\mathbf{R} = \mathbf{C}\mathbf{W}_r + \mathbf{b}_r, \quad (10)$$

where the linear projection is applied pointwise at each time step so that the channel dimension of the convolutional stream is aligned with the LSTM hidden dimension. The fused hidden representation is

$$\mathbf{H}_{\text{res}} = \mathbf{H}_{\text{LSTM}} + \mathbf{R}, \quad (11)$$

and the final time step is used for binary classification:

$$p_{\theta}(y = 1 \mid \mathbf{X}) = \sigma(\mathbf{w}^{\top} \mathbf{H}_{\text{res}}^{(S)} + b), \quad (12)$$

where $\mathbf{C}$ is the convolutional feature sequence, $\mathbf{H}_{\text{LSTM}}$ is the recurrent hidden sequence, $\mathbf{W}_r$ and $\mathbf{b}_r$ are the residual projection parameters, and $\mathbf{R}$ is the projected residual stream. In Equation (12), $\mathbf{H}_{\text{res}}^{(S)}$ denotes the fused hidden representation at the final time step, $\mathbf{w}$ and b are classifier parameters, and $\sigma(\cdot)$ is the sigmoid function. The resulting probability $p_{\theta}(y = 1 \mid \mathbf{X})$ is thresholded after validation-set selection to obtain the final binary output, so the overall mapping of this module is from a standardized multivariate window to an anomaly score and then to a deployment-time decision.

In the benchmark implementation, both the AL-enabled and fully supervised LSTMResNet variants use a hidden size of 256, two recurrent layers, and 64 convolutional filters, with 888,257 trainable parameters. Figure 1 presents the architecture-level view of the backbone together with the training-time active-learning loop, whereas Table 2 summarizes the backbone tensor flow. The AL-enabled and fully supervised variants therefore differ only in how labels are revealed during training, not in the deployed inference architecture.

Table 2. Tensor flow of the benchmark LSTMResNet backbone.

Stage	Operation	Output Shape
Input window	Raw multivariate sequence	$(B, 96, 15)$
Reordering	permute (0, 2, 1) for Conv1D input	$(B, 15, 96)$
Conv block 1	Conv1d (15, 64, 3, padding = 1) + ReLU	$(B, 64, 96)$
Conv block 2	Conv1d (64, 64, 3, padding = 1) + ReLU	$(B, 64, 96)$
Sequence restoration	permute (0, 2, 1)	$(B, 96, 64)$
Recurrent modeling	2-layer LSTM, hidden size 256	$(B, 96, 256)$
Residual projection	Pointwise Linear (64, 256) for residual feature mapping and dimension alignment	$(B, 96, 256)$
Fusion	Element-wise addition of recurrent and residual streams	$(B, 96, 256)$
Temporal selection	Last time step retention	$(B, 256)$
Classifier	Linear (256, 1) + sigmoid	$(B, 1)$

3.4. Training Objective

For a labeled mini-batch $\mathcal{B} = \{(\mathbf{X}_n, y_n)\}_{n=1}^N$, the benchmark optimizes binary cross-entropy loss

$$\mathcal{L}_{\text{BCE}}(\theta) = -\frac{1}{N} \sum_{n=1}^N \left[y_n \log p_{\theta}(y = 1 | \mathbf{X}_n) + (1 - y_n) \log (1 - p_{\theta}(y = 1 | \mathbf{X}_n)) \right], \quad (13)$$

where N is the batch size, $\mathbf{X}_n$ is the n -th input window, $y_n \in \{0, 1\}$ is its binary anomaly label, and $p_{\theta}(y = 1 | \mathbf{X}_n)$ is the predicted anomaly probability from Equation (12). This objective is minimized with Adam during both the fully supervised backbone training and the AL-based warm-up/fine-tuning stages, so the comparison isolates differences in label revelation and sampling policy rather than differences in the optimization criterion itself.

At deployment time, the binary anomaly decision is obtained by thresholding the predicted probability:

$$\hat{y} = \mathbb{I}[p_{\theta^*}(y = 1 | \mathbf{X}) \geq \tau^*], \quad (14)$$

where θ^* denotes the trained model parameters, τ^* is the validation-selected operating threshold, and $\mathbb{I}[\cdot]$ is the indicator function that outputs 1 when the condition inside is true and 0 otherwise.

3.5. Pool-Based Active Learning Module

The final training-stage component of the framework is the pool-based active-learning loop, which operates on the labeled and unlabeled partitions of the training split.

For the proposed framework, the training subset is divided into an initially labeled set and an unlabeled pool. Let $\mathcal{D}_L$ denote the current labeled set and let $\mathcal{D}_U$ denote the unlabeled pool. After an initial 8-epoch warm-up on $\mathcal{D}_L$, each candidate window in $\mathcal{D}_U$ receives a boundary-distance score

$$u(\mathbf{X}) = |p_{\theta}(y = 1 | \mathbf{X}) - 0.5|, \quad (15)$$

where $u(\mathbf{X})$ denotes the uncertainty score of sample $\mathbf{X}$ and $p_{\theta}(y = 1 | \mathbf{X})$ is the model-estimated anomaly probability. Smaller values of $u(\mathbf{X})$ indicate higher uncertainty because they correspond to predictions closer to the decision boundary at 0.5. Samples with $u(\mathbf{X}) \approx 0$ are therefore treated as the most informative candidates under this query rule.

At each acquisition round, the unlabeled pool is not ranked in full. Instead, a candidate subset is first sampled without replacement from $\mathcal{D}_U$, scored by $u(\mathbf{X})$, and reduced to a shortlist containing the samples with the smallest boundary-distance scores, as controlled by the query budget and a shortlist expansion factor. In the main benchmark, this corresponds to a candidate subsample of 12,000 windows, a query size of 4000, and a

shortlist cap of 4000×6 . The final queried batch is then selected from that shortlist. When prediction-balanced querying is enabled, the shortlist is partitioned according to whether $p_\theta(y = 1 | \mathbf{X})$ is above or below 0.5, and the queried batch is assembled by approximately balancing these two predicted groups. This heuristic is used to maintain informative supervision for both sides of the current decision boundary and to reduce the risk of early acquisition rounds becoming dominated by the majority-predicted group. No hidden ground-truth labels are used to impose that balance before selection. Oracle labels are revealed only after sample selection, after which the queried windows are appended to $\mathcal{D}_L$.

The active-learning loop is implemented in an offline masked-label setting. Although all labels exist in principle, a portion of the training labels is hidden initially and revealed only when selected by the query strategy. The objective is therefore to test whether selective labeling allows the model to approach the operating point of the same backbone trained with all labels available from the outset. Queried samples are removed from the unlabeled pool after each round so that the same window cannot be selected repeatedly.

In the main benchmark, the framework uses an initial labeled ratio of 75%, equivalent to a masked-label ratio of 25%. The model performs three acquisition rounds, queries 4000 windows per round, applies eight warm-up epochs on the initially labeled subset, and performs one fine-tuning epoch after each acquisition round. A single fine-tuning epoch is adopted to limit the computational overhead of repeated retraining and to maintain a consistent update budget across acquisition rounds. This setting is used as the primary benchmark configuration for stable comparison, whereas additional sensitivity runs with 50% and 75% masked-label ratios are reported later to examine the same ALSTMResNet pipeline under more label-constrained settings. By contrast, the fully supervised LSTM-ResNet backbone is trained under the standard benchmark regime of up to 80 epochs with an early-stopping patience of 15. In addition to the boundary-distance uncertainty rule used in the main benchmark, the same acquisition schedule is also evaluated with entropy sampling, least-confidence sampling, margin sampling, and random querying to isolate the effect of the acquisition policy.

3.6. End-to-End Workflow

Figure 2 presents the training and evaluation workflow as a flowchart. The figure replaces the previous algorithm-style listing and emphasizes the operational sequence from raw files to held-out test evaluation.

4. Experiments

4.1. Dataset Description

The experiments are based on the public 3W oil-well anomaly dataset introduced by Vargas et al. [13]. The original dataset contains files associated with different source types and event categories for well-event analysis. In the present benchmark, only real-well files are retained, whereas simulated and hand-drawn files are excluded so that the resulting evaluation remains focused on field-like operating behavior.

In the benchmark implementation, the data are organized into nine class folders containing CSV files that represent individual well records or record segments. The pre-processing routine scans these folders, retains at most 30 real-well CSV files per class, and constructs the final benchmark from 142 source files. Within each class folder, eligible CSV files are first ordered by filename and truncated to the first 30 after source-type filtering. After this truncation step, the retained files are shuffled with random seed 42 for class-wise split assignment. The resulting file-wise partition contains 83 training files, 28 validation files, and 31 test files, ensuring that windows extracted from a given source file appear in only one split. After sliding-window construction with a fixed 96-step window and

stride 10, the benchmark contains 233,812 windows in total, of which 122,191, 47,537, and 64,084 are assigned to the training, validation, and test splits, respectively.

Table 3 summarizes the class types retained in the present benchmark subset. Unlike the original 3W dataset summary, this table reports the actual normal and anomalous categories used in the local experiment after real-file filtering and class-wise truncation. Including the normal class here is important because the present task is formulated as binary anomaly-state classification, and the retained benchmark therefore depends on both the abnormal taxonomy and the available normal operating records.

Table 3. Normal and anomalous event types retained in the benchmark subset.

Class	Event Type	Real Files	Total Windows
0	Normal	30	54,056
1	Abrupt Increase of BSW	5	11,783
2	Spurious Closure of DHSV	22	15,665
3	Severe Slugging	30	53,058
4	Flow Instability	30	21,576
5	Rapid Productivity Loss	12	36,090
6	Quick Restriction in PCK	6	5367
7	Scaling in PCK	4	27,135
8	Hydrate in Production Line	3	9082
Total	All retained classes	142	233,812

Table 4 reports the class-wise split statistics used for model development and evaluation. To keep the layout compact while preserving the split information, the table lists file counts and window counts in train/validation/test order for each class.

Table 4. Class-wise split statistics of the benchmark subset.

Class	Event Type	Files (Train/Val/Test)	Windows (Train/Val/Test)
0	Normal	18/6/6	32,342/11,023/10,691
1	Abrupt Increase of BSW	3/1/1	7858/2859/1066
2	Spurious Closure of DHSV	13/4/5	7127/4409/4129
3	Severe Slugging	18/6/6	32,046/10,671/10,341
4	Flow Instability	18/6/6	12,748/4575/4253
5	Rapid Productivity Loss	7/2/3	8769/5990/21,331
6	Quick Restriction in PCK	3/1/2	3916/1193/258
7	Scaling in PCK	2/1/1	15,825/4406/6904
8	Hydrate in Production Line	1/1/1	1560/2411/5111
Total	All retained classes	83/28/31	122,191/47,537/64,084

The file cap is used as a benchmark-control measure rather than as a claim about optimal sampling. In the absence of such a limit, some models would incur substantially higher memory and runtime costs, making comparison less controlled. By enforcing the same cap for all candidate methods, the benchmark preserves a common computational budget while still covering all nine event folders represented in the dataset. The present implementation controls leakage at the file level only and does not parse or verify a separate well identifier beyond the filenames themselves.

Because the retained source files differ substantially in duration, the resulting window counts are not proportional to the file counts across classes or across train/validation/test partitions. At the window level, the abnormal class accounts for 73.53% of the training split, 76.81% of the validation split, 83.32% of the test split, and 76.88% of the full benchmark. Thus, although the original events are sparse in the raw records, the resulting task is better

interpreted as a skewed binary window-level monitoring problem. This file-wise split protocol is central to the leakage-aware design of the benchmark.

4.2. Evaluation Metrics

The benchmark evaluates predictive performance using precision, recall, F_1 -score, and accuracy. Precision quantifies the reliability of positive alarms, that is, the proportion of predicted anomalies that are truly abnormal:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \quad (16)$$

Recall quantifies the sensitivity of the detector, namely the proportion of truly abnormal windows that are successfully recovered:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (17)$$

The F_1 -score summarizes the trade-off between these two objectives through their harmonic mean:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (18)$$

Accuracy measures the overall fraction of correctly classified windows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \quad (19)$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively. These four metrics capture complementary aspects of practical behavior. Precision reflects the burden of false alarms, recall measures missed abnormal conditions, F_1 -score balances the two under class skew, and accuracy provides a global but less discriminative summary when the class distribution is imbalanced [36,37]. For that reason, F_1 -score is used as the principal model-selection metric. In addition, training time reflects the cost of model development and retraining, whereas inference time reflects the cost of deployment-time execution.

4.3. Baselines

The main experimental comparison evaluates five deep-learning-based configurations and three classical machine learning baselines under a shared split. The deep-learning-based configurations are: (i) the uncertainty-guided active-learning-enhanced LSTMResNet, denoted ALSTMResNet; (ii) the same LSTMResNet backbone trained with the full labeled training split; (iii) a budget-matched LSTMResNet rerun (LSTM-EB), trained on the same final labeled budget as ALSTMResNet; (iv) a random-query active-learning control, denoted AL-Rand, which uses the same backbone and AL schedule as ALSTMResNet but replaces uncertainty-guided querying with random querying; and (v) a pure LSTM backbone with hidden size 128 and two recurrent layers. Additional acquisition-strategy variants using entropy sampling, least-confidence sampling, and margin sampling are evaluated separately in the repeated policy-level comparison. All deep models operate directly on the standardized multivariate windows.

For the classical methods, each window is converted into a handcrafted feature vector composed of six statistics per channel: mean, standard deviation, minimum, maximum, median, and variance. Because each window contains 15 channels, the resulting feature vector has 90 dimensions. The retained classical baselines are a radial-basis-function (RBF) SVM [17], Isolation Forest [38], and a stratified Dummy classifier. Isolation Forest

predictions are remapped so that anomaly remains the positive class. This comparison set preserves the main axes needed for interpretation: paired backbone control, label-budget control, acquisition-policy control, a simpler recurrent baseline, a discriminative classical baseline, an unsupervised anomaly baseline, and a trivial lower-bound classifier.

Classical baselines remain relevant in industrial monitoring because low-dimensional statistical summaries can still capture substantial discrimination signal, especially when the target decision is binary and the operational patterns are relatively structured. A deep model result is therefore more convincing when evaluated against competent classical comparators rather than against neural references alone.

4.4. Experimental Settings

The experiments are run in a Conda environment using Python 3.8.20, PyTorch 2.4.1, and scikit-learn 1.3.2. The benchmark employs random seed 42 for reproducibility. Deep models use a batch size of 64, a learning rate of 10^{-3} , binary cross-entropy (BCE) loss, and a maximum of 80 epochs with an early-stopping patience of 15. For the proposed AL model under the main 25% masked-label setting, the initial labeled set contains 91,643 windows, the unlabeled pool contains 30,548 windows, and the final labeled set grows to 103,643 windows after three query rounds. The anomaly class is treated as the positive class throughout the benchmark.

In the retained main comparison, all probability-output deep models, namely AL-STMResNet, LSTMResNet, LSTM-EB, AL-Rand, and Pure LSTM, use the same validation-threshold search over the interval $[0.35, 0.75]$ with step size 0.01, and the selected operating point is the one that maximizes validation F_1 -score. This unified threshold-selection protocol is used to keep the comparison fair across deep models, since threshold choice directly affects ROC/precision–recall trade-offs and can materially change the reported F_1 outcome [39,40]. Classical baselines retain their native decision rules on the same train/validation/test split. The random-query active-learning control AL-Rand and the LSTM-EB are included under the same evaluation protocol. Training and inference times are wall-clock measurements intended for within-study comparison, and the inference time corresponds to a full forward pass over the held-out test loader after training.

In this study, fair comparison refers specifically to using the same file-wise split, the same train-only normalization, and validation-based operating-point selection for the probability-output deep models, together with explicit backbone, label-budget, and acquisition-policy controls. All deep models therefore receive the same window representation and the same data partition for training, validation, and testing. Classical baselines are evaluated on statistical descriptors derived from exactly the same windows rather than from a different handcrafted dataset. These choices reduce the chance that a method appears stronger simply because it benefited from a more favorable preprocessing or evaluation setup [41–43].

Table 5 summarizes the main comparison subset and configurations used throughout the later analysis.

4.5. Comparison Results

Table 6 presents the selected main comparison on the retained file-wise split. The table is used as the main operating-point comparison because it contains the paired backbone control, the random-query control, and the retained baseline models under the same train-only standardization and validation-threshold protocol. All training and inference times are reported in seconds under the same runtime environment.

Table 5. Main comparison subset and configurations used in the core discussion.

Model	Configuration
ALSTMResNet	Hidden dimension = 256; recurrent layers = 2; convolution filters = 64; kernel size = 3; masked-label ratio = 0.25 in the main setting; warm-up epochs = 8; AL rounds = 3; query size = 4000 per round; pool subsample = 12,000; shortlist factor = 6; prediction-balanced uncertainty shortlist sampling based on predicted probabilities; one fine-tuning epoch per round; common validation-threshold grid [0.35, 0.75] with step size 0.01, selected by validation F_1 .
LSTMResNet	Hidden dimension = 256; recurrent layers = 2; convolution filters = 64; kernel size = 3; Adam learning rate = 10^{-3} ; maximum epochs = 80; early-stopping patience = 15; same validation-threshold grid as ALSTMResNet, selected by validation F_1 .
LSTM-EB	Same backbone as LSTMResNet, but trained on a random 103,643-window subset to match the final ALSTMResNet labeled budget; same validation-threshold grid as ALSTMResNet, selected by validation F_1 .
AL-Rand	Same backbone and AL schedule as the proposed model, but queried samples are selected uniformly at random rather than by uncertainty; same validation-threshold grid as ALSTMResNet, selected by validation F_1 .
Pure LSTM	Hidden dimension = 128; recurrent layers = 2; same validation-threshold grid as ALSTMResNet, selected by validation F_1 .
(RBF) SVM	RBF kernel; gamma = scale.
Isolation Forest	Number of estimators = 200; contamination = 0.3; random seed = 42.
Dummy classifier	Stratified prediction; random seed = 42.

Table 6. Selected main comparison including ALSTMResNet, its paired controls, and the retained baseline models used in the core discussion. All timing results are reported in seconds under the same runtime environment; best values in each metric column are bold, with lower values preferred for timing metrics.

Model	Prec. $\uparrow$	Rec. $\uparrow$	F_1 $\uparrow$	Acc. $\uparrow$	Train Time (s) $\downarrow$	Infer Time (s) $\downarrow$
ALSTMResNet (ours)	0.92	0.95	0.94	0.90	268.70	6.14
LSTMResNet	0.93	0.92	0.92	0.88	752.70	6.04
LSTM-EB	0.92	0.96	0.94	0.90	1004.72	6.02
AL-Rand	0.92	0.95	0.93	0.89	260.28	6.04
Pure LSTM	0.93	0.91	0.92	0.88	170.66	2.47
(RBF) SVM	0.91	0.84	0.88	0.82	202.74	93.05
Isolation Forest	0.90	0.61	0.73	0.64	1.34	0.47
Dummy classifier	0.78	0.65	0.71	0.59	<0.01	0.00

Under this retained split, ALSTMResNet reaches an F_1 -score of 0.9354 while requiring substantially less retraining time than the full-data LSTMResNet backbone and the budget-matched rerun LSTM-EB. The single-split result should be interpreted together with the repeated file-wise validation in Table 7, which characterizes rank stability under alternative file partitions.

Table 7. Monte Carlo file-wise validation over five repeated partitions. Values are reported as mean $\pm$ standard deviation; the best value in each metric column is bolded once, with lower values preferred for variance and training time.

Model	Prec. $\uparrow$	Rec. $\uparrow$	F_1 $\uparrow$	F_1 Var. $\downarrow$	Train Time (s) $\downarrow$
ALSTMResNet (ours)	0.8978 $\pm$ 0.0461	0.8388 $\pm$ 0.0941	0.8661 $\pm$ 0.0666	0.0044	266.28 $\pm$ 11.92
AL-Rand	0.8985 $\pm$ 0.0443	0.8400 $\pm$ 0.0946	0.8670 $\pm$ 0.0663	0.0044	263.55 $\pm$ 13.26
LSTMResNet	0.8978 $\pm$ 0.0359	0.8306 $\pm$ 0.0873	0.8610 $\pm$ 0.0546	0.0030	886.01 $\pm$ 340.84
LSTM-EB	0.8894 $\pm$ 0.0378	0.8340 $\pm$ 0.0711	0.8601 $\pm$ 0.0521	0.0027	522.59 $\pm$ 134.71

4.6. Repeated File-Wise Validation

To address the stability of the reported comparison, Table 7 reports five repeated file-wise Monte Carlo trials using split seeds 42–46. In each trial, source files are repartitioned before window construction, standardization is fitted on the training split only, and the same model configuration is retrained. This repeated protocol is not intended to replace the detailed retained-split analysis in Table 6; rather, it estimates how much the operating point varies when the file partition changes.

The repeated trials refine the interpretation of the retained-split result. ALSTMResNet remains competitive in average F_1 -score, but it does not show a meaningful mean advantage over the random-query control in this binary setting, while providing strong evidence of a computational benefit: relative to the full-data LSTMResNet backbone, the average training time is reduced by approximately 69.9%, while the average F_1 -score remains within 0.006 of the paired controls. A policy-level comparison under alternative query rules is included to test whether this pattern changes when the acquisition score is varied.

4.7. Acquisition-Strategy Comparison

Table 8 compares five active-learning query policies under the same five repeated file-wise partitions. The compared policies include boundary-distance uncertainty sampling used in the main ALSTMResNet workflow, entropy sampling, least-confidence sampling, margin sampling, and random querying. All five variants use the same LSTMResNet backbone, warm-up schedule, query budget, pool subsampling size, validation-threshold search, and file-wise leakage-aware evaluation protocol; only the acquisition score used to rank candidate windows is changed.

Table 8. Comparison of active-learning acquisition strategies over five repeated file-wise partitions. Values are reported as mean $\pm$ standard deviation; the best value in each metric column is bolded once, with lower values preferred for variance and training time.

Strategy	Prec. $\uparrow$	Rec. $\uparrow$	F_1 $\uparrow$	F_1 Var. $\downarrow$	Train Time (s) $\downarrow$
Boundary dist. (ours)	0.8978 $\pm$ 0.0461	0.8388 $\pm$ 0.0941	0.8661 $\pm$ 0.0666	0.0044	266.28 $\pm$ 11.92
Entropy	0.8969 $\pm$ 0.0478	0.8395 $\pm$ 0.0937	0.8660 $\pm$ 0.0669	0.0045	257.27 $\pm$ 12.49
Least conf.	0.8976 $\pm$ 0.0464	0.8401 $\pm$ 0.0927	0.8667 $\pm$ 0.0657	0.0043	255.29 $\pm$ 12.03
Margin	0.8971 $\pm$ 0.0476	0.8389 $\pm$ 0.0941	0.8658 $\pm$ 0.0671	0.0045	256.60 $\pm$ 13.85
Random	0.8985 $\pm$ 0.0443	0.8400 $\pm$ 0.0946	0.8670 $\pm$ 0.0663	0.0044	263.55 $\pm$ 13.26

The acquisition-strategy comparison shows that the five policies are nearly indistinguishable under the present binary probability-output setting. The mean F_1 -scores range only from 0.8658 to 0.8670, and the standard deviations overlap across all strategies. This result is consistent with the mathematical relationship among binary uncertainty scores: boundary distance, entropy, least-confidence, and margin sampling are monotonic or near-monotonic transformations of the same predicted class probability. Thus, in this benchmark, changing the uncertainty score has little effect on the operating point compared with the effects of file-wise splitting, label budget, and retraining schedule.

4.8. Ablation Results

To isolate the contribution of the AL design itself on the retained split, Table 9 focuses on the proposed model and its two primary paired controls, namely the full-data backbone and the budget-matched rerun used here as the *equal labeled budget* control (LSTM-EB). This view separates two questions: whether the partial-label workflow can approach the full-data backbone, and whether the achieved operating point is obtained with lower retraining cost. Figure 3 additionally includes the random-query control as a policy-level reference.

Table 9. Ablation-style paired comparison focused on ALSTMResNet and its two primary paired controls. Best values in each metric column are bolded, with lower values preferred for timing metrics.

Configuration	Prec. $\uparrow$	Rec. $\uparrow$	F_1 $\uparrow$	Acc. $\uparrow$	Train Time (s) $\downarrow$	Infer Time (s) $\downarrow$
w/o AL	0.93	0.92	0.92	0.88	752.70	6.04
Equal labeled budget	0.92	0.96	0.94	0.90	1004.72	6.02
ALSTMResNet (Full)	0.92	0.95	0.94	0.90	268.70	6.14

On the retained file-wise split, ALSTMResNet improves F_1 -score by +0.0118 relative to the full-data backbone while reducing training time by approximately 64.3%, with nearly unchanged inference cost. Relative to the budget-matched rerun, ALSTMResNet reaches a closely comparable operating point but reduces retraining time by approximately 73.3%. The repeated trials in Table 7 show that this single-split ranking should not be overinterpreted as stable dominance across file partitions.

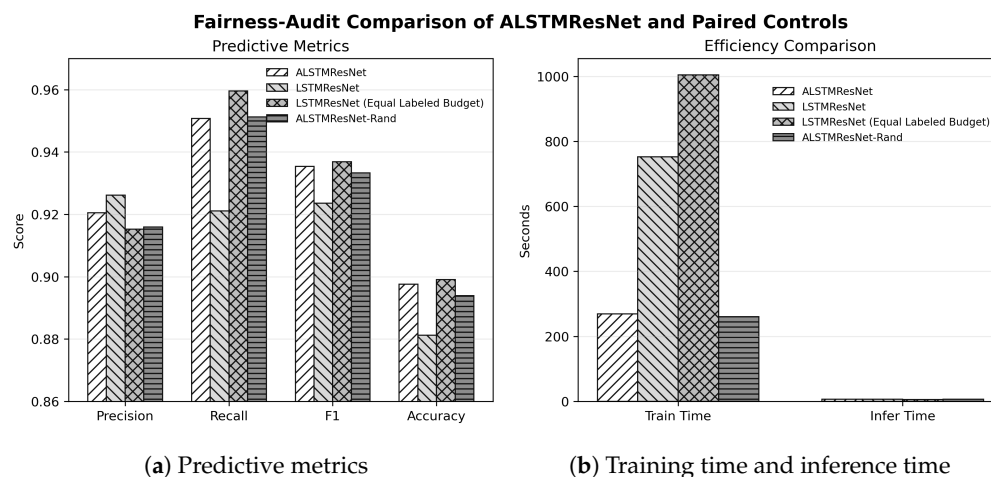


Figure 3. Extended ablation-style fair comparison among ALSTMResNet, the full-data LSTMResNet backbone, the budget-matched LSTMResNet rerun used as the *w/o guided selection* control (LSTM-EB), and the random-query control (AL-Rand). (a) Predictive metrics. (b) Training time and inference time.

4.9. Discussion and Analysis

The retained-split and repeated-trial results point to a training-cost trade-off rather than a large acquisition-policy effect. On the retained split, ALSTMResNet reduces training time by more than half relative to the full-supervision backbone while reaching a competitive operating point under unified threshold tuning. Across the five repeated file-wise trials, however, the uncertainty-guided variant and AL-Rand are nearly indistinguishable in mean F_1 -score.

Although LSTM-EB achieves a marginally higher F_1 -score on the retained split, it requires a substantially longer retraining process. In time-sensitive industrial update scenarios, this small predictive gap may be outweighed by the reduction in computational overhead and retraining burden.

Table 6 summarizes the retained-split trade-off numerically, and Figure 4 presents the same comparison in the performance–time space. The purpose of the figure is not to isolate the numerically highest score alone, but to show how the main comparison set is distributed in the performance–time space. Under this view, ALSTMResNet lies close to the high- F_1 operating region while requiring far less retraining time than either the full-data backbone or the budget-matched supervised rerun.

This pattern is also reflected in the retained comparison set as a whole. Pure LSTM achieves performance close to that of the full LSTMResNet backbone with lower training and inference cost, whereas (RBF) SVM remains a competitive classical baseline but incurs a much larger inference cost. The lower inference time of Pure LSTM is expected because it omits the convolutional front-end and residual projection used in ALSTMResNet. These additional components increase forward-pass cost, but they also provide richer local temporal feature extraction and recurrent-residual fusion, which is consistent with the solid predictive behavior of the proposed framework. Isolation Forest and the Dummy classifier define substantially weaker reference points in predictive terms, although both remain computationally inexpensive. Taken together, these results indicate that, within the

selected comparison set, the main practical separation lies in the training cost required to reach a competitive operating region while maintaining predictive quality.

The confusion-matrix comparison in Figure 5 provides a complementary view at the error-pattern level. The three retained high-performing rows from the fair-comparison setting operate in a very similar error region, but they reach that region with markedly different training costs. This result reinforces the same practical interpretation: among methods with similar deployment-time behavior, the main distinction lies in the cost required to obtain that operating point.

For industrial monitoring, this framing is especially relevant because engineering teams often care about retraining burden, annotation effort, and update speed alongside small differences in a final predictive metric. A strategy that preserves the deployed model while reaching a competitive operating region can therefore remain useful even when the budget-matched supervised rerun is marginally stronger in one split.

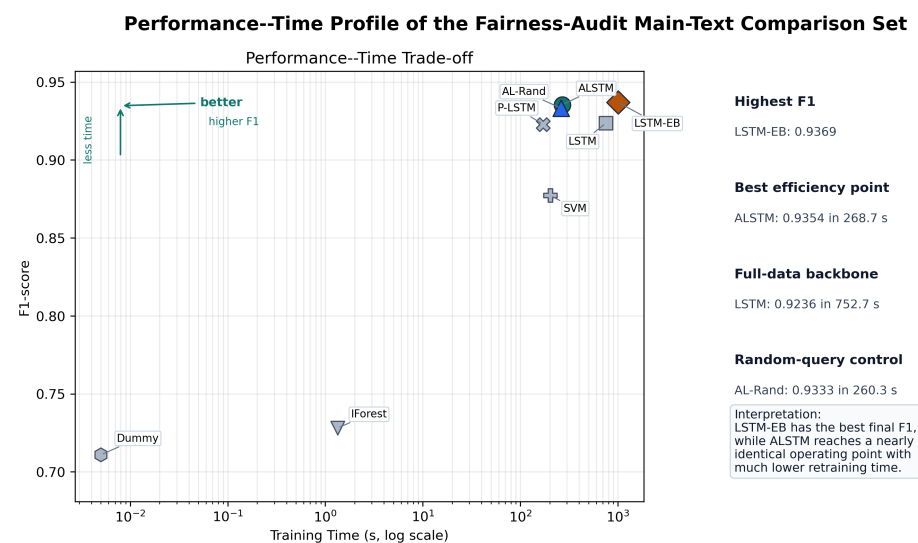


Figure 4. Performance–time profile of the compact main comparison set. The x-axis reports training time on a log scale and the y-axis reports F_1 -score. ALSTMResNet lies in the high- F_1 region while requiring less retraining time than the marginally higher-scoring budget-matched rerun LSTM-EB. Each point is labeled directly; nearby models are separated with callout labels to avoid overlap.

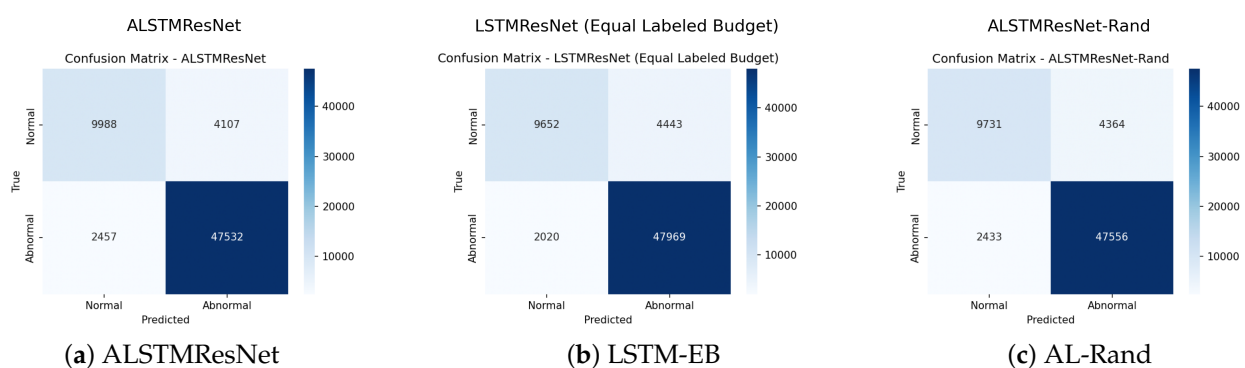


Figure 5. Confusion-matrix comparison for three retained high-performing fair-comparison rows. (a) ALSTMResNet. (b) LSTM-EB. (c) AL-Rand. The panels highlight that these methods operate in a very similar error region despite markedly different retraining costs.

Label-Efficiency Learning Curve

Figure 6 and Table 10 report the label-efficiency learning-curve analysis under progressively tighter label budgets. Masked-label ratios of 25%, 50%, and 75% denote the proportion of training labels hidden from the initial labeled set before the AL loop begins.

The final numbers of labeled windows therefore decrease from 103,643 to 73,095 and to 42,547 as the initial label budget becomes tighter.

The learning curve indicates that the AL variants remain in a competitive F_1 range under reduced labeled budgets, but no acquisition policy consistently dominates all budgets. At the 25% masked-label setting, the five AL policies obtain nearly identical F_1 -scores around 0.938, while the budget-matched supervised rerun obtains 0.9003. At the 50% and 75% settings, the budget-matched rerun obtains higher F_1 -scores than most AL variants but requires substantially longer training time. These results support the engineering claim that the proposed workflow can operate under constrained labels and reduce retraining time, while also showing that the acquisition policy itself should not be presented as uniformly better.

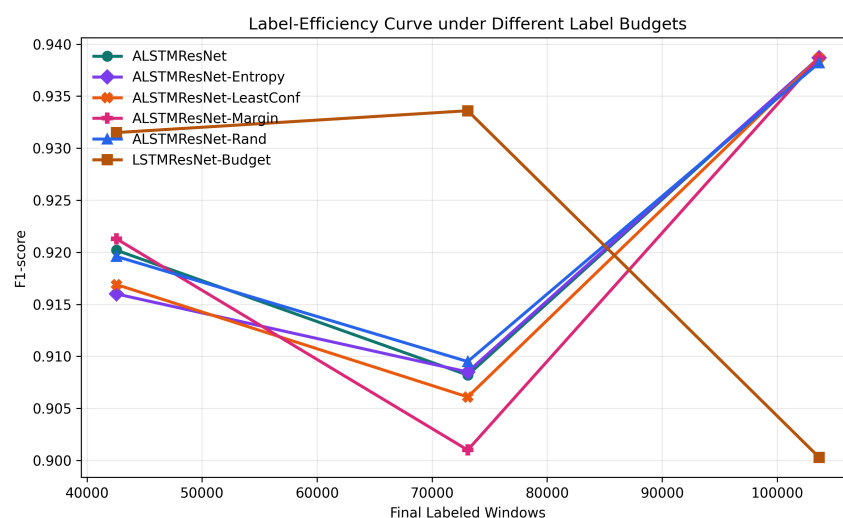


Figure 6. Label-efficiency learning curve under three masked-label ratios. The curve compares uncertainty-guided, entropy, least-confidence, margin, and random-query ALSTMResNet variants with a budget-matched supervised LSTMResNet rerun using the same final labeled-window budgets.

Table 10. Label-efficiency results under different final labeled-window budgets on the retained split. Each final labeled-window budget reports F_1 and training time in seconds; the best value in each metric column is bolded once, and ties in F_1 are resolved in favor of the boundary-distance ALSTMResNet variant.

Strategy	103,643 Windows		73,095 Windows		42,547 Windows	
	$F_1 \uparrow$	Time (s) $\downarrow$	$F_1 \uparrow$	Time (s) $\downarrow$	$F_1 \uparrow$	Time (s) $\downarrow$
Boundary-distance uncertainty (ours)	0.9387	251.42	0.9082	174.48	0.9202	100.14
Entropy sampling	0.9387	308.20	0.9085	204.08	0.9160	114.96
Least-confidence sampling	0.9387	287.50	0.9061	212.14	0.9169	122.08
Margin sampling	0.9387	304.12	0.9010	210.50	0.9213	128.06
Random querying	0.9382	248.42	0.9095	169.35	0.9196	96.36
LSTM-EB	0.9003	433.90	0.9336	345.72	0.9315	246.35

4.10. Limitations

Several limitations remain in this study. First, the repeated validation is performed as file-wise Monte Carlo resampling on the retained public benchmark, but it is not a live field deployment or a cross-field prospective trial. The results should therefore be interpreted as offline benchmark evidence on public real-well records rather than proof of field performance. Practical deployment would still require online data ingestion, alarm-delay evaluation, cross-well validation, long-term drift monitoring, and periodic recalibration on target wells.

Second, the active-learning loop is simulated through offline label masking and oracle reveal. This protocol makes acquisition strategies reproducible and comparable, but it does not model annotation delay, human errors, or annotator disagreement. In a field workflow, uncertain windows would need to be submitted to domain experts in batches, and disputed labels would require dual review or senior-engineer arbitration before being used for retraining.

Third, the task is formulated as binary anomaly-state prediction. This formulation is suitable for first-stage alarm screening, where the immediate operational need is to distinguish abnormal from normal operation, but it does not replace fine-grained fault-type diagnosis. A multiclass extension would require replacing the sigmoid output and BCE loss with a softmax classifier and cross-entropy loss, together with macro- F_1 , per-class recall, and class-balanced acquisition analysis.

Finally, the acquisition-policy effect is limited in the present binary setting. The uncertainty-guided rule, entropy sampling, margin sampling, and least-confidence sampling are closely related for binary probability outputs, and the empirical comparison with AL-Rand shows little separation in mean F_1 -score.

5. Conclusions

This paper presented ALSTMResNet for oil-well anomaly monitoring under a leakage-aware file-wise evaluation protocol. On the retained benchmark split, ALSTMResNet achieved a test F_1 -score of 0.9354 with a training time of 268.70 s, remaining close to the full-data and budget-matched LSTMResNet controls while requiring substantially less retraining time. Five repeated file-wise Monte Carlo trials further showed that ALSTMResNet remained competitive on average, with an F_1 -score of 0.8661 ± 0.0666 and an average training time of 266.28 s, compared with 886.01 s for the full-data LSTMResNet backbone.

Overall, the study provides a reproducible workflow that combines source-file-level splitting, train-only standardization, partial-label training, paired controls, repeated validation, and label-efficiency analysis for oil-well anomaly monitoring. Within this scope, the framework offers a practical way to study retraining efficiency and label-budget constraints while keeping the limitations of offline simulation and binary alarm screening explicit.

Author Contributions: Conceptualization, F.G., Y.C. and Y.F.; methodology, F.G., Z.Y. and F.Y.; software, J.L., Y.C. and Y.F.; validation, J.L., Y.C. and Y.F.; formal analysis, F.G., Z.Y. and J.L.; investigation, F.G., Z.Y., F.Y., H.X., Y.P. and K.L.; data curation, F.Y., H.X., Y.P., J.L. and K.L.; writing—original draft preparation, J.L., Y.C. and Y.F.; writing—review and editing, F.G., Z.Y., F.Y., H.X., Y.P., J.L., Y.C., Y.F. and K.L.; visualization, J.L., Y.C. and Y.F.; supervision, Y.C. and Y.F.; project administration, F.G., Z.Y., F.Y., H.X., Y.P. and K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Oil & Gas Major Project “Efficient Wellbore Deliquification Control, Intelligent Production Technology, and Integrated Demonstration for Low-Pressure Gas Fields”, PetroChina Southwest Oil & Gasfield Company (Grant No. 2025ZD1408000), and the Oil & Gas Major project “Research and Application of Intelligent Analysis Technology for Shale Gas Production Process”, PetroChina Southwest Oil & Gasfield Company (Grant No. 2025D00311).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: https://github.com/ricardovargas/3w_dataset.

Conflicts of Interest: Authors Feng Ge, Zhi Yang, Fan Yu, Yang Peng, Honglin Xiao and Ke Luo were employed by the company PetroChina Southwest Oil & Gasfield Company. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. PetroChina Southwest Oil & Gasfield

Company had no role in the design of the study; in the collection, analysis, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this paper:

AL	Active Learning
AL-Rand	Random-Query Active Learning
ALSTMResNet	Active-Learning-Enhanced Long Short-Term Memory Residual Network
BCE	Binary Cross-Entropy
CSV	Comma-Separated Values
FN	False Negative
FP	False Positive
LSTM	Long Short-Term Memory
LSTM-EB	Equal-Budget Long Short-Term Memory Residual Network
LSTMResNet	Long Short-Term Memory Residual Network
RBF	Radial Basis Function
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

References

1. Fernandes, W., Jr.; Komati, K.S.; Gazolli, K.A.S. Anomaly detection in oil-producing wells: A comparative study of one-class classifiers in a multivariate time series dataset. *J. Pet. Explor. Prod. Technol.* **2024**, *14*, 343–363. <https://doi.org/10.1007/s13202-023-01710-6>. [CrossRef]
2. Lian, Y.; Geng, Y.; Tian, T. Anomaly detection method for multivariate time series data of oil and gas stations based on digital twin and MTAD-GAN. *Appl. Sci.* **2023**, *13*, 1891. <https://doi.org/10.3390/app13031891>. [CrossRef]
3. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **2009**, *41*, 15. <https://doi.org/10.1145/1541880.1541882>. [CrossRef]
4. Blázquez-García, A.; Conde, A.; Mori, U.; Lozano, J.A. A review on outlier/anomaly detection in time series data. *ACM Comput. Surv.* **2021**, *54*, 56. <https://doi.org/10.1145/3444690>. [CrossRef]
5. Settles, B. Active Learning. In *Synthesis Lectures on Artificial Intelligence and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 6, pp. 1–114. <https://doi.org/10.2200/S00429ED1V01Y201207AIM018>. [CrossRef]
6. Cohn, D.A.; Ghahramani, Z.; Jordan, M.I. Active learning with statistical models. *J. Artif. Intell. Res.* **1996**, *4*, 129–145. <https://doi.org/10.1613/jair.295>. [CrossRef]
7. Gal, Y.; Islam, R.; Ghahramani, Z. Deep Bayesian active learning with image data. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 1183–1192.
8. Yoo, D.; Kweon, I.S. Learning loss for active learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 93–102. <https://doi.org/10.1109/CVPR.2019.00018>. [CrossRef]
9. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>. [CrossRef] [PubMed]
10. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. <https://doi.org/10.1109/CVPR.2016.90>. [CrossRef]
11. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 11106–11115. <https://doi.org/10.1609/aaai.v35i12.17325>. [CrossRef]
12. Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; Long, M. iTransformer: Inverted transformers are effective for time series forecasting. In Proceedings of the Twelfth International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.

13. Vargas, R.E.V.; Munaro, C.J.; Ciarelli, P.M.; Medeiros, A.G.; do Amaral, B.G.; Barrionuevo, D.C.; de Araújo, J.C.D.; Ribeiro, J.L.; Magalhães, L.P. A realistic and public dataset with rare undesirable real events in oil wells. *J. Pet. Sci. Eng.* **2019**, *181*, 106223. <https://doi.org/10.1016/j.petrol.2019.106223>. [CrossRef]
14. Kim, B.; Alawami, M.A.; Kim, E.; Oh, S.; Park, J.; Kim, H. A comparative study of time series anomaly detection models for industrial control systems. *Sensors* **2023**, *23*, 1310. <https://doi.org/10.3390/s23031310>. [CrossRef] [PubMed]
15. Panza, M.A.; Pota, M.; Esposito, M. Anomaly detection methods for industrial applications: A comparative study. *Electronics* **2023**, *12*, 3971. <https://doi.org/10.3390/electronics12183971>. [CrossRef]
16. Khalili, Y.; Ahmadi, M.; Moraveji, M.K. Time-aware predictive maintenance of electrical submersible pumps using catboost ensemble learning and trend-based labeling. *J. Pet. Explor. Prod. Technol.* **2025**, *15*, 147. <https://doi.org/10.1007/s13202-025-02070-z>. [CrossRef]
17. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. <https://doi.org/10.1007/BF00994018>. [CrossRef]
18. Wang, Z.; Yan, W.; Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 1578–1585. <https://doi.org/10.1109/IJCNN.2017.7966039>. [CrossRef]
19. Fawaz, H.I.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.-A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. <https://doi.org/10.1007/s10618-019-00619-1>. [CrossRef]
20. Karim, F.; Majumdar, S.; Darabi, H.; Chen, S. LSTM fully convolutional networks for time series classification. *IEEE Access* **2018**, *6*, 1662–1669. <https://doi.org/10.1109/ACCESS.2017.2779939>. [CrossRef]
21. Karim, F.; Majumdar, S.; Darabi, H.; Harford, S. Multivariate LSTM-FCNs for time series classification. *Neural Netw.* **2019**, *116*, 237–245. <https://doi.org/10.1016/j.neunet.2019.04.014>. [CrossRef] [PubMed]
22. Shojaadini, S.V.; Beirami, M.J. Mobile sensor based human activity recognition: Distinguishing of challenging activities by applying long short-term memory deep learning modified by residual network concept. *Biomed. Eng. Lett.* **2020**, *10*, 419–430. <https://doi.org/10.1007/s13534-020-00160-x>. [CrossRef] [PubMed]
23. Zou, X.; Wang, Z.; Li, Q.; Sheng, W. Integration of residual network and convolutional neural network along with various activation functions and global pooling for time series classification. *Neurocomputing* **2019**, *367*, 39–45. <https://doi.org/10.1016/j.neucom.2019.08.023>. [CrossRef]
24. Wang, H.; Lan, K.; Xia, G.; Geng, S. Time-varying structural damage detection based on LSTM-ResNet model. *J. Telem. Track. Command.* **2022**, *43*, 8–17. <https://doi.org/10.12347/j.jcyk.20211108001>. [CrossRef]
25. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Proceedings of the Advances in Neural Information Processing Systems 34, Virtual Event, 6–14 December 2021; pp. 22419–22430.
26. Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; Jin, R. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; Volume 162, pp. 27268–27286.
27. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. In Proceedings of the Eleventh International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.
28. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems 30, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
29. Zerveas, G.; Jayaraman, S.; Patel, D.; Bhamidipaty, A.; Eickhoff, C. A transformer-based framework for multivariate time series representation learning. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Virtual Event, 14–18 August 2021; pp. 2114–2124. <https://doi.org/10.1145/3447548.3467401>. [CrossRef]
30. Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; Sun, L. Transformers in time series: A survey. In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI), Macao, China, 19–25 August 2023; pp. 6778–6786.
31. Ren, P.; Xiao, Y.; Chang, X.; Huang, P.-Y.; Li, Z.; Gupta, B.B.; Chen, X.; Wang, X. A survey of deep active learning. *ACM Comput. Surv.* **2022**, *54*, 180. <https://doi.org/10.1145/3472291>. [CrossRef]
32. Kirsch, A.; van Amersfoort, J.; Gal, Y. BatchBALD: Efficient and diverse batch acquisition for deep Bayesian active learning. In Proceedings of the Advances in Neural Information Processing Systems 32, Vancouver, BC, Canada, 8–14 December 2019.
33. Sener, O.; Savarese, S. Active learning for convolutional neural networks: A core-set approach. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
34. Ash, J.T.; Zhang, C.; Krishnamurthy, A.; Langford, J.; Agarwal, A. Deep batch active learning by diverse, uncertain gradient lower bounds. In Proceedings of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 26–30 April 2020.
35. Beluch, W.H.; Genewein, T.; Nürnberger, A.; Köhler, J.M. The power of ensembles for active learning in image classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 9368–9377. <https://doi.org/10.1109/CVPR.2018.00976>. [CrossRef]

36. Saito, T.; Rehmsmeier, M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE* **2015**, *10*, e0118432. <https://doi.org/10.1371/journal.pone.0118432>. [CrossRef] [PubMed]
37. Chicco, D.; Tötsch, N.; Jurman, G. The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *Biodata Min.* **2021**, *14*, 13. <https://doi.org/10.1186/s13040-021-00244-z>. [CrossRef] [PubMed]
38. Liu, F.T.; Ting, K.M.; Zhou, Z.-H. Isolation forest. In Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM), Pisa, Italy, 15–19 December 2008; pp. 413–422. <https://doi.org/10.1109/ICDM.2008.17>. [CrossRef]
39. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. <https://doi.org/10.1016/j.patrec.2005.10.010>. [CrossRef]
40. Lipton, Z.C.; Elkan, C.; Naryanaswamy, B. Optimal thresholding of classifiers to maximize F1 measure. In *Machine Learning and Knowledge Discovery in Databases*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8725, pp. 225–239. https://doi.org/10.1007/978-3-662-44851-9_15. [CrossRef] [PubMed]
41. Kaufman, S.; Rosset, S.; Perlich, C.; Stitelman, O. Leakage in data mining: Formulation, detection, and avoidance. *ACM Trans. Knowl. Discov. Data* **2012**, *6*, 15. <https://doi.org/10.1145/2382577.2382579>. [CrossRef]
42. Arlot, S.; Celisse, A. A survey of cross-validation procedures for model selection. *Stat. Surv.* **2010**, *4*, 40–79. <https://doi.org/10.1214/09-SS054>. [CrossRef]
43. Cerqueira, V.; Torgo, L.; Mozetič, I. Evaluating time series forecasting models: An empirical study on performance estimation methods. *Mach. Learn.* **2020**, *109*, 1997–2028. <https://doi.org/10.1007/s10994-020-05910-7>. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.