

RAMSeS: Robust and Adaptive Model Selection for Time-Series Anomaly Detection Algorithms

Mohamed Abdelmaksoud
BIFOLD & TU Berlin
Berlin, Germany
mohamed@tu-berlin.de

Sheng Ding
University of Stuttgart
Stuttgart, Germany
sheng.ding@ias.uni-stuttgart.de

Andrey Morozov
University of Stuttgart
Stuttgart, Germany
andrey.morozov@ias.uni-stuttgart.de

Ziawasch Abedjan
BIFOLD & TU Berlin
Berlin, Germany
abedjan@tu-berlin.de

Abstract—Time-series data vary widely across domains, making a universal anomaly detector impractical. Methods that perform well on one dataset often fail to transfer because what counts as an anomaly is context dependent. The key challenge is to design a method that performs well in specific contexts while remaining adaptable across domains with varying data complexities. We present the Robust and Adaptive Model Selection for Time-Series Anomaly Detection RAMSeS framework. RAMSeS comprises two branches: (i) a stacking ensemble optimized with a genetic algorithm to leverage complementary detectors. (ii) An adaptive model-selection branch identifies the best single detector using techniques including Thompson sampling, robustness testing with generative adversarial networks, and Monte Carlo simulations. This dual strategy exploits the collective strength of multiple models and adapts to dataset-specific characteristics. We evaluate RAMSeS and show that it outperforms prior methods on F1.

I. INTRODUCTION

Time-series are prevalent across domains such as finance, industry, and healthcare [1]–[7]. They encode complex system dynamics and often exhibit heterogeneity in seasonality, regimes, and sampling irregularities [8], [9]. The diversity in time-series structures is accompanied by a wide range of anomaly types that occur in practice, such as isolated spikes, contextual anomalies, and collective anomalies [10]–[19]. Each category captures a distinct notion of abnormality, and manifestations are domain-dependent [9], [20]. Structural diversity across time-series and the variety of anomaly types impede the realization of a “one-size-fits-all” model. Methods that excel on one series often degrade on another, and a single series can be subject to concept drifts, where the normal behavior and distribution changes over time [8], [9], [20]. In clinical patient monitoring, where, physiological signals such as ECG, blood pressure, and respiratory rate differ in scale, sampling frequency, and what constitutes anomalous behavior [8], [21]. An irregular heartbeat, a sudden spike in blood pressure, and an abnormal breathing pattern reflect different anomaly types and therefore require different models. Moreover, even within a single patient’s ECG stream, the optimal detector can shift: during rest periods, simple threshold-based methods suffice, but during exercise or stress, adaptable pattern-based detectors are needed to distinguish anomalous arrhythmia from expected elevated heart rate [2], [22].

Anomaly detection in heterogeneous setups is challenging not only because the problem is multifaceted, but also because

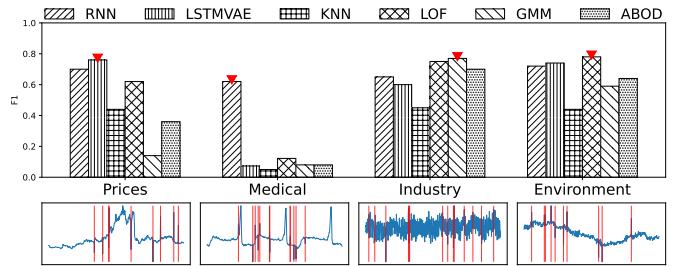


Fig. 1. F1 score for six anomaly detection models from statistical and neural-network families across four domains (Prices, Medical, Industry, and Environment) [21], [29]–[31]. Red vertical lines indicate spike anomalies. The triangle marks the best-performing model in each domain.

existing approaches face practical limitations and data scarcity. For example, distance- and density-based methods such as k-nearest neighbors and Local Outlier Factor rely on stable local neighborhoods [23], [24], in reality the series might be subject to temporal non-stationarity, i.e., different density of data points across time intervals. Reconstruction-based approaches, such as variational autoencoders, assume anomalies yield larger reconstruction errors than normal patterns [25]–[27]. Assumptions break under shifts in scale, overexpressive autoencoders, or concept drift [3], [8], [9], [28]. Figure 1 depicts performance of statistical and neural-network-based models on four different time-series, each from a different domain. No approach outperforms all other on all domains confirming the aforementioned limitations.

Existing work has attempted to address these limitations either by selecting the best fit single model per time series or by aggregating multiple models in ensembles [3], [8], [9], [28], [32], [33]. Nevertheless, significant open challenges remain. Current single-model selection algorithms are efficient and interpretable on stable segments but degrade under regime changes, non-stationarity, and borderline anomalies [9]. Prior ensemble techniques exploit the collective signals of multiple detectors but are highly sensitive to subset model selection and often assume comparable score scales [9]. To bridge this gap, we propose RAMSeS (Robust and Adaptive Model Selection for Time-Series Anomaly Detection). RAMSeS introduces two novel approaches within the same framework: a genetic-algorithm-based stacking ensemble that uses self-supervised learning to effectively search through possible ensemble candidates and a single-model selection module that combines linear

Thompson sampling with robustness testing. In summary, we make the following contributions:

- *Genetic-algorithm-based stacking ensemble (GA-Ens)*. Our approach combines a genetic algorithm with a meta-learner. The entanglement helps to guide the subset selection process of the genetic algorithm via scores learned by the meta-learner. As a result, our approach converges significantly faster than exhaustive grid and Bayesian search [34].
- *Single model selection* traverses models using contextual linear Thompson sampling (LinTS) and ε -greedy exploration: LinTS is conditioned on window-level context and balances exploration and exploitation, yielding rankings that follow local regime changes. Simultaneously, robustness is assessed via three tests: Generative Adversarial Network (GAN)-based perturbations, Monte Carlo simulations, and an off-by-threshold sensitivity analysis. Then, the LinTS-based rankings and robustness signals are combined via Markov-based rank aggregation to produce a stable consensus order.
- We present an experimental evaluation on known benchmark datasets [21], [29], [30] and show that each of our proposed approaches, ensemble and model selection, significantly outperform the state-of-the-art counter parts [8], [9].

II. RELATED WORK

There are six lines of related work: *time series anomaly detection* and model selection targeting the no one-size-fits-all challenge, *model selection via surrogate metrics* that avoid labels, *learning-based adaptive selection* with reinforcement learning and bandits, *meta-learning* for per-dataset model choice, *model ensembling* for combining multiple base learners' scores, and *benchmarking frameworks*.

A. Time Series Anomaly Detection and Model Selection

Time series anomaly detection (TSAD) spans statistical forecasting, residual analysis, density-based, and deep learning approaches including autoencoders, recurrent networks, and Transformers [23], [35]–[38]. Recent work has explored online detection, model-based diagnostics, and toolchain integration for cyber-physical systems [39]–[44], as well as adaptive model selection and automated evaluation [45], [46]. However, these methods treat model evaluation as a static ranking task without incorporating robustness testing or temporal reliability. In contrast, RAMSeS integrates adaptive model selection with robustness-aware ensemble optimization, enabling consistent performance under diverse and evolving conditions.

B. Surrogate Metrics for Model Selection

In the absence of labels, surrogate metrics estimate detector quality by combining multiple signals. Goswami et al. [8] propose a composite score that aggregates prediction error, detector centrality, and performance on synthetic anomalies via rank aggregation to approximate supervised orderings. Although effective on curated datasets, this method assumes that synthetic anomalies resemble real deviations and may overemphasize detectors with high centrality. Jung et al. [27] present LaF-AD, which weights detectors by the variance of

anomaly probabilities across bootstraps to reward stability. However, it does not explicitly test robustness to distribution shift or adversarial noise and lacks online adaptivity. RAMSeS incorporates robustness evaluations GAN-based perturbations, off-by-threshold tests, and Monte Carlo simulations, to stress-test and rank detectors under realistic conditions.

C. Learning-Based Adaptive Selection

Adaptive selectors learn policies that switch between models over time. Zhang et al. [47] formulate model selection as a Markov decision process (MDP) over contextual features such as detector scores and inter-model agreement, training a policy with confidence-based rewards. This approach captures temporal adaptation but relies on crafted states and requires tuning and training effort. Ngo et al. [48] apply contextual bandits in hierarchical edge environments to balance accuracy and latency. Their selection is trained offline and optimizes deployment cost rather than robustness or generalization. In contrast, RAMSeS integrates an adaptive selection policy based on LinTS with ε -greedy, allowing contextual adaptation while preserving robustness to noise and shifting conditions.

D. Meta-Learning for Time Series Model Choice

Meta-learning approaches infer the most suitable detector from dataset characteristics. Sylligardos et al. [49] demonstrate that dataset-specific model choice can be learned and incorporated into AutoML pipelines, but the resulting selection remains static at inference time and ignores intra-series regime changes. Boniol et al. [50] present ADecimo, a practical selector for time-series anomaly detection, which similarly focuses on dataset-level choice rather than online adaptation. To overcome such static behavior, RAMSeS extends beyond per-dataset meta-learning by enabling continuous, context-aware model adaptation throughout the time series.

E. Ensemble Selection

At a high level, ensembles differ along two axes: (i) *scope*, aggregating scores from all detectors versus selecting a subset, and (ii) *fusion rule*, how per-detector scores are combined.

Outlier Ensemble (OE) instantiates three canonical score-level rules: *AVG*, mean of detector scores to reduce variance. *MAX*, pointwise maximum to highlight strong outlier evidence and reduce bias, and *AOM*, which averages maxima over random detector subsets to balance bias-variance [33]. Iterative Outlier Ensemble (IOE) iteratively refines a pseudo ground truth by updating with the closest detector score vector until convergence [32]. SELECT performs *component selection* via two complementary schemes: a *Vertical* strategy that prefers detectors agreeing on anomalous samples (model consensus), and a *Horizontal* strategy that, per sample, prefers detectors that appear most reliable for that sample. The selected outputs are then ensembled using consensus techniques [51]. HITS Adapting hub/authority centrality, HITS builds a bipartite graph between detectors and data points to produce detector reliability weights [32], [52]. Existing ensemble selection techniques remain biased toward group consensus, even when

the collective decision is incorrect. RAMSeS mitigates this by employing a GA-Ens to ensure consistency and scalability.

F. Benchmarking-Based Evaluation Paradigms

TSB-UAD provides an end-to-end TSAD suite emphasizing standardized evaluation protocols and reproducibility [53]. Similarly, Schmidl et al. [54] conduct a comprehensive comparison of classical and deep models, revealing variance across datasets and metrics. AutoTSAD introduces a label-free framework that synthesizes regime-rich series with injected anomalies to optimize configurations and ensembles [3]. Recently, TSB-AutoAD evaluates selection, ensemble, and generation strategies across domains, exposing strong variability and trade-offs between accuracy and computational cost, and proposing selective ensemble as a lightweight hybrid [9]. These efforts demonstrate that detector performance is highly context-dependent and that a simple ensemble can be accurate but computationally expensive [3], [9], [54]. Building on these insights, RAMSeS incorporates an ensemble, reinforcement learning and robustness evaluations to provide systematic, dataset-independent tests for model selection.

III. PROBLEM FORMULATION AND RAMSeS OVERVIEW

We formalize the problem of model selection for TSAD and outlines our framework, RAMSeS.

A. Problem formulation

We are considering problems of single model and ensemble selection for TSAD. Before defining the objective, we first introduce time series, anomaly types and model candidates.

A Time series $X = \{x_t\}_{t=1}^T$ can be univariate ($x_t \in \mathbb{R}$) or multivariate ($x_t \in \mathbb{R}^{d_x}$). A fixed *sliding window* per series is generated. Given window length w_X and stride s_X , we obtain windows $\{W_i\}_{i=1}^{N_X}$ with $W_i = \{x_t\}_{t=t_i}^{t_i+w_X-1}$.

Anomaly types include *point*, *contextual*, and *collective* anomalies [8]. Point anomalies affect a single timestamp, such as *spikes* and *flips*. Contextual anomalies violate local behavior, such as seasonality or local trends. Collective anomalies arise when a subsequence is abnormal as a group although individual points appear normal [8].

Candidates: A set of base anomaly detection models $\mathcal{D} = \{d_1, \dots, d_M\}$ serve as candidates. For each window W_i , a model d_m outputs a normalized anomaly score $s_{i,m} \in [0, 1]$.

Objective: With the given time series dataset X and the set of candidates \mathcal{D} , the optimization goals for ensemble and model selection are slightly different. The former aims at obtaining an ensemble, i.e., a subset of candidates, and the latter aims at finding the best candidate both, with the goal of achieving high detection quality, measured by common scores, such as the harmonic mean (F1) of precision and recall or the area under the curve AUC-PR, or a combination of such scores. With the objective of maximizing the F1, we select:

$$\text{Single model: } d^* = \arg \max_{m \in \{1, \dots, M\}} F1(d_m) \quad (1)$$

$$\text{Ensemble: } S^* = \arg \max_{S \subseteq \{1, \dots, M\}} F1(E^{(S)}) \quad (2)$$

Here $E^{(S)}$ denotes the stacking ensemble built from subset S .

B. RAMSeS Overview

Fig. 2 presents RAMSeS. From a shared pool of base models, our framework proceeds along two branches: an ensemble selection and a single-model selection branch. Together, these branches provide application-specific flexibility.

The need for two-branch design. The two branches in RAMSeS serve complementary but distinct purposes. The ensemble branch targets robustness and generalization across heterogeneous time series by leveraging model diversity, whereas the single-model branch focuses on efficiency and interpretability. Integrating both into a single unified branch would require reconciling conflicting objectives: ensemble optimization favors diversity and cross-model aggregation, while model selection prioritizes decisive ranking and parsimony. RAMSeS adaptively switches between robustness-oriented and latency-oriented configurations without retraining.

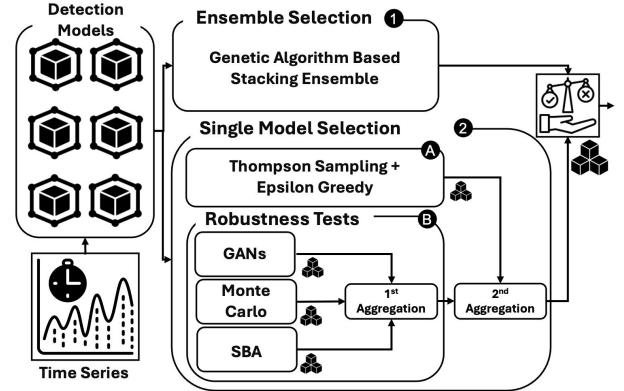


Fig. 2. RAMSeS framework overview.

1) *Ensemble Branch ① in Fig. 2:* RAMSeS employs a self-supervised approach that combines a meta-learner and a genetic algorithm (GA) [55]–[59]. Rather than relying on simple uniform (AVG, MAX) or manual selection, the GA systematically searches for high-performing *subsets of base models*. Each candidate ensemble combines the outputs of selected base models via a *fixed* meta-learner, chosen *a priori* based on empirical best practices. Namely, we use Random Forest (RF) as a meta-learner [60]–[63], in our framework, other meta-learners can be used, e.g., Logistic Regression [64]–[67], or Support Vector Machines (SVM) [68]–[72]. The meta-learner is fixed within the time series for all optimized subsets, as searching for a different meta-learner for each ensemble may increase the probability of overfitting [56], [72]. The ensemble selection module outputs an optimized subset of base models.

2) *Single-Model Selection Branch ② in Fig. 2:* In parallel to the ensemble branch, RAMSeS searches for individual models that perform well on time series. For this purpose, it searches in parallel through linear Thompson sampling and robustness-based test (A and B in Fig. 2). The outputs of both approaches are then merged into an aggregated ranking.

a) *LinTS A in Fig. 2:* To enable adaptive model selection under uncertainty, RAMSeS formulates the task as a contextual bandit and applies *LinTS* with ε -greedy exploration.

Each candidate model is treated as an arm. Selection is conditioned on contextual features extracted from the input stream, allowing RAMSeS to adaptively identify effective models based on localized feedback and historical performance [73].

b) *Robustness-based tests* **B** in Fig. 2: Three parallel heuristics perturb time series to rank models based on distinct aspects of robustness and sensitivity. First, *GAN-based perturbation testing* evaluates robustness against semantically plausible perturbations by generating *borderline*, hard-to-detect anomalies across all time series [74], [75]. Second, *SBA analysis* examines sensitivity near decision boundaries by injecting anomalies via local-statistics-scaled noise around threshold regions [11]. Third, *MC simulations* test stability under diverse anomaly patterns through MC-based brute-force tests using multiple randomized anomaly settings [76], [77]. As shown in Fig. 2, resulting robustness rankings are then combined through *Markov-based rank aggregation* into a single ranking, presenting the first aggregation. Our Markov ranking approach is described in more detail in V-C. Finally, we use another Markov aggregation approach, to obtain one unified ranking among both single model selection branches.

Example: SKAB 1-1 Water Circulation To illustrate RAMSeS’s operation, we use SKAB 1-1 [29], a water circulation monitoring dataset with 8 sensor features including: accelerometers, flow rate, pressure, temperature, voltage. The time series exhibits a pump failure anomaly where flow rate drops from 32 L/min to 3.2 L/min with correlated temperature decrease. We use 8 pre-trained base models: 1 RNN, 1 MD, 1 DGHL, 1 CBLOF, 2 KNN, and 2 LOF variants. As depicted in Fig. 3, the dataset is split into offline (train+test) and online deployment with re-optimization every $N = 5$ windows.

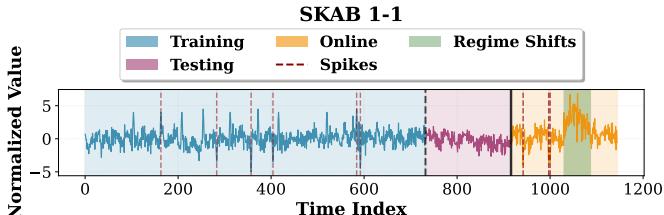


Fig. 3. SKAB 1-1 time series

IV. ENSEMBLE SELECTION

RAMSeS’s ensemble branch searches subsets of base models and trains a meta-learner to produce a stacked ensemble.

Design rationale. Stacking combines different detection models with complementary strengths by training a meta-learner on their outputs [55]. A stacking ensemble comprises level-0 base learners whose predictions are used as features for a level-1 meta-learner [56]. Fixing the meta-learner *a priori* restricts the search space to the possible combinations of base-model subsets, which (i) reduces overfitting risk, (ii) stabilizes the optimization, and (iii) makes convergence behavior interpretable. The ensemble search space is a discrete subset-selection problem over heterogeneous base models.

Existing solutions for searching ensemble space use grid search or bayesian optimization. *Grid search* scales with

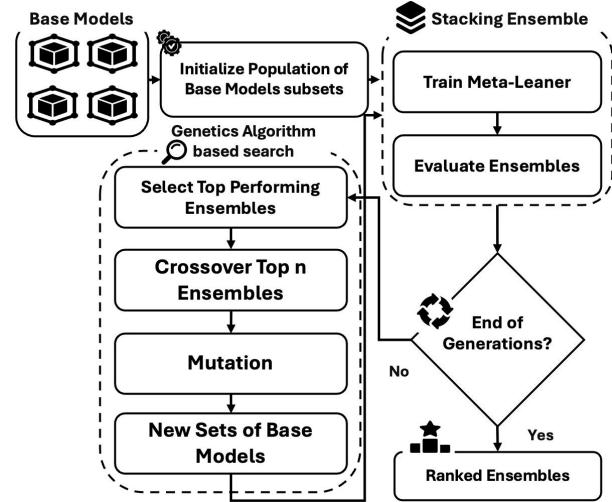


Fig. 4. Stacking ensemble via a genetic algorithm.

combinatorial complexity as $\mathcal{O}(2^M)$ or $\binom{M}{k}$ for fixed sizes and becomes infeasible even at moderate number of models M [34]. *Bayesian optimization* works best for low-dimensional continuous hyperparameters with a smooth response surface. However, our ensemble selection task involves choosing discrete subsets of heterogeneous models, which violates these assumptions and requires custom kernels over sets. This causes slow convergence [34]. Our goal is to avoid searching all subsets and take advantage of the predictions by the meta-learner. Considering the possible subsets of models as successive genes, where any extension of a model subset is the child of another, a genetic algorithm focuses on traversing high-performing subset generations. This way, the number of evaluations reduces drastically and converges faster [34].

Algorithm 1: GA based stacking-ensemble

Input: G : #generations, \mathcal{D} : set of pre-trained base models, P : population size, T : validation data, N : selection number, μ : mutation rate, ϕ : fixed meta-learner
Output: \hat{S}_G : optimized model subset with trained meta-learner
 $J_\sigma(E^{(\hat{S}_G)})$: best objective value (F1/AUC-PR)

```

1. foreach  $i \in \{1, \dots, P\}$  do  $\mathcal{S}_0[i] \leftarrow$  random subset from  $\mathcal{D}$  (initialization)
2. for  $g \leftarrow 1$  to  $G$  do
3.   foreach  $S \in \mathcal{S}_{g-1}$  do
4.     Train  $\phi$  on stacked base-model outputs of  $S$  over  $T$  (training)
5.     Compute F1 and/or AUC – PR on the validation fold
6.   end
7.    $\mathcal{S}_{elite} \leftarrow$  top- $k$  subsets by  $J_\sigma$ 
8.   for  $i \leftarrow 1$  to  $P$  do
9.      $S_1, S_2 \leftarrow$  select  $N$  parents from  $\mathcal{S}_{elite}$  (selection)
10.     $S' \leftarrow$  crossover( $S_1, S_2$ ) (crossover)
11.    if rand()  $< \mu$  then
12.      |  $S' \leftarrow$  mutate( $S'$ ) (mutation)
13.    end
14.    add  $S'$  to  $\mathcal{S}_g$ 
15.  end
16.   $\hat{S}_g \leftarrow \arg \max_{S \in \mathcal{S}_g} J_\sigma(E^{(S)})$ 
17. end
18. return  $\hat{S}_G, J_\sigma(E^{(\hat{S}_G)})$ 

```

A. Optimization procedure.

The workflow of the GA approach is depicted in Figure 4 and formalized in Algorithm 1. The GA iterates through a population of candidate subsets that are evaluated by the meta-learner in a self-supervised manner. To start with a represen-

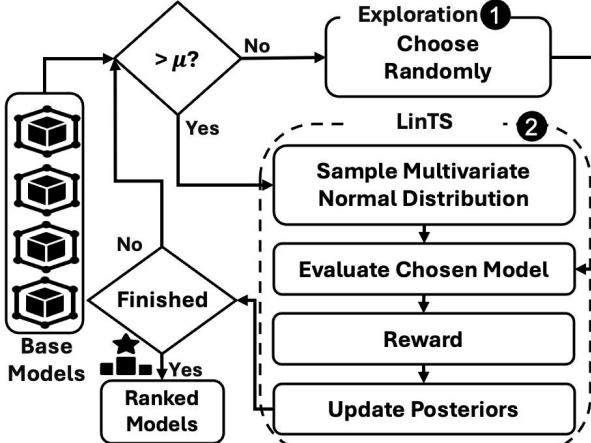


Fig. 5. Linear Thompson sampling with ε -greedy for model selection.

tative population, the GA initializes a population $\{S^{(p)}\}$ by randomly sampling distinct subsets from \mathcal{D} (initialization in Algorithm 1). For each $S^{(p)}$, RAMSeS trains the fixed meta-learner ϕ on stacked base-model outputs. It then computes $F1$ and/or $AUC-PR$ for each subset on a held-out validation fold (training in Algorithm 1). The GA then selects the top- n parent ensembles based on $F1$ and/or $AUC-PR$ for the crossover step. In this step, the models of the selected parent subsets are crossed to form new subsets (selection in Algorithm 1). Finally, the *Mutation* step injects exploratory variations, such as adding, removing, or replacing base models, in a subset. The mutation rate is chosen based on best practices and can be adaptively adjusted based on observed convergence dynamics (mutation in Algorithm 1). This process repeats for a fixed number of generations. In the end, the best-scoring subset \hat{S} is the chosen ensemble.

Example. On SKAB 1-1 III-B2b, the GA-Ens branch initializes a population of 20 ensembles with RF as a meta-learner and $\mu = 0.1$. GA converges within 20 generations on ensemble $\{\text{RNN}, \text{KNN_2}, \text{LOF_2}, \text{CBLOF}\}$. This diversity captures complementary failure modes: RNN models temporal flow patterns during normal operation (32 L/min steady-state), LOF_2 detects density shifts when flow drops abruptly, CBLOF identifies cluster deviations in the 8-sensor space, and KNN_2 captures local proximity patterns. The meta-learner learns to weight LOF_2 and CBLOF higher during multi-sensor anomalies (flow-temperature correlation breakdown) and RNN during stable temporal regimes.

V. SINGLE MODEL SELECTION

The single model selection branch of RAMSeS has two components: (i) a *LinTS* with ε greedy module to produce model ranking, and (ii) a *robustness and sensitivity* module that derives complementary rankings. Both rankings are aggregated using *Markov-based rank aggregation* to obtain a consensus ordering. Next, we will explain each part in detail.

A. Linear Thompson Sampling (*LinTS*)

LinTS with ε -greedy is a lightweight and uncertainty-aware approach to balance models exploration and exploitation. In

Algorithm 2: LinTS with ε -greedy

Input: X : time-series stream, \mathcal{D} : set of candidate models, w : window size for segmentation, ε_0 : initial exploration probability, κ : exploration decay rate, N : number of evaluation windows, α : reward weight for F1 vs. AUC-PR

Output: \hat{m} : models ranking based on posterior means

1. **Initialize:**
2. Segment X into windows $\{W_t\}_{t=1}^N$
3. For each model $m \in \mathcal{D}$, initialize posterior parameters:
4. $\mu_m \leftarrow \vec{0}, \Sigma_m \leftarrow I$
5. Set $\varepsilon_t \leftarrow \varepsilon_0$ (initialization)
6. **for** $t \leftarrow 1$ to N **do**
7. **Sample model** m_t using a hybrid strategy:
8. With probability ε_t : select $m_t \sim \text{Uniform}(\mathcal{D})$
9. Else: sample $\tilde{\theta}_m \sim \mathcal{N}(\mu_m, \Sigma_m)$ for each $m \in \mathcal{D}$; choose $m_t = \arg \max_m \tilde{\theta}_m^\top x_t$ (selection)
10. **Evaluate** m_t on W_t and compute reward:
11. $r_t \leftarrow \alpha \cdot \text{F1}(m_t, W_t) + (1 - \alpha) \cdot \text{AUC-PR}(m_t, \mathcal{B}_t)$
12. % where \mathcal{B}_t is a rolling buffer of recent windows (reward)
13. **Update posteriors** for m_t via Bayesian linear regression:
14. $\Sigma_{m_t}^{-1} \leftarrow \Sigma_{m_t}^{-1} + x_t x_t^\top$
15. $\mu_{m_t} \leftarrow \Sigma_{m_t} (\Sigma_{m_t}^{-1} \mu_{m_t} + x_t r_t)$ (Posterior Update)
16. **Anneal exploration:**
17. $\varepsilon_{t+1} \leftarrow \varepsilon_0 \cdot \exp(-\kappa t)$ (annealing)
18. **end**
19. **return** \hat{m}

our setting, each sliding window yields immediate rewards (F1 and/or AUC-PR) that reflect local data characteristics. The Bayesian posterior over model rewards captures both performance and uncertainty, guiding the selection toward models with high expected reward and strong evidence.

Design rationale. We adopt LinTS because it extends classical bandit formulations with contextual awareness through a Bayesian linear reward model [73], [78], [79]. The algorithm begins with sliding-window preparation and alternates between exploration, random choice with probability ε_t , and exploitation, posterior sampling and maximization. Each model is scored via F1 and/or AUC-PR rewards, and Bayesian posterior updates guide subsequent selections. The overall architecture is shown in Figure 5 and detailed in Algorithm 2.

Priors and initialization. For each model i , reward parameter $\theta_i \in \mathbb{R}^d$ follows a Gaussian prior $\theta_i \sim \mathcal{N}(\mu_{i,0}, \Sigma_{i,0})$ with mean $\mu_{i,0} = \vec{0}$ and covariance $\Sigma_{i,0} = \lambda^{-1} I_d$. Initializing with a ridge prior ($\lambda > 0$) stabilizes early updates and ensures $\Sigma_{i,t}$ remains positive definite. Rewards are normalized to $[0, 1]$ before updating the posterior so that a unit-variance likelihood assumption holds, (initialization in Algorithm 2).

Our LinTS algorithm operates as follows. For each window W_t , we construct a contextual representation $x_t \in \mathbb{R}^d$. Each model i maintains a Bayesian linear reward model

$$\mathbb{E}[r_t^{(i)} | x_t] = \theta_i^\top x_t, \quad (3)$$

with model-specific posterior parameters $(\mu_{i,t}, \Sigma_{i,t})$. As mentioned in the selection step of Algorithm 2, at time t , with probability ε_t we select a model uniformly at random (exploration), otherwise, we draw $\tilde{\theta}_i \sim \mathcal{N}(\mu_{i,t}, \Sigma_{i,t})$ and select $\arg \max_i \tilde{\theta}_i^\top x_t$ (exploitation via LinTS). The exploration rate decays as $\varepsilon_t = \varepsilon_0 \exp(-\kappa t)$. For the selected model m_t applied to W_t , we compute the reward

$$r_t = \alpha \cdot \text{F1}(m_t, W_t) + (1 - \alpha) \cdot \text{AUC-PR}(m_t, W_t), \quad (4)$$

Algorithm 3: GAN-based perturbations

- Input:** *data*: original data; *labels*: original labels;
input_dim: input dimensions; *epochs*; *batch_size*:
batch size; *noise_dim*: noise dimensions
Output: updated data, labels, indices of injected points
1. **Initialize** generator and discriminator
 2. **Train** with label smoothing and noise for *epochs*
 3. **Generate** borderline points with the trained generator
 4. **Score** candidates with the discriminator and assign labels
 5. **Integrate** generated points via sliding windows
 6. **Return** updated arrays and injection indices
-

where $\alpha \in [0, 1]$ weights thresholded performance (F1) versus threshold-free ranking (AUC-PR). As depicted in reward step in Algorithm 2, LinTS computes rewards *exclusively* on batches of injected windows with synthetic labels \tilde{y} . After observing (x_t, r_t) , we update the posterior via

$$\Sigma_{i,t+1} = (\Sigma_{i,t}^{-1} + x_t x_t^\top)^{-1} \quad (5)$$

$$\mu_{i,t+1} = \Sigma_{i,t+1}^{-1} (\Sigma_{i,t}^{-1} \mu_{i,t} + x_t r_t). \quad (6)$$

After processing N windows, models are ranked according to their posterior rewards, and this LinTS ranking is passed to the final aggregation along with the robustness-based ranking.

Cold start. At initialization, higher ε_0 encourages broad exploration. ε_t then decreases to prioritize exploitation. To stabilize rewards in short windows, we optionally apply exponential smoothing to r_t before updating the posterior. Exponential smoothing averages recent rewards with past ones, giving higher weight to recent observations to reduce noise.

Example. Applying LinTS to SKAB 1-1, it explores all 8 models over 50 validation windows and selects the RNN based on accumulated posterior means. This selection aligns with the properties of SKAB 1-1 from Figure 3: RNN’s sequential architecture captures temporal dependencies in pump flow, pressure, and temperature dynamics, outperforming proximity-based methods, such as LOF or KNN, that ignore temporal correlations between the 8 sensors.

B. Robustness and Sensitivity Tests

Models robustness and sensitivity is evaluated via three complementary tests. Each test yields a model ranking which is merged by Markov-based rank aggregation.

1) *GAN-Based Robustness Evaluation:* This module tests models against anomalies that mimic realistic drifts while preserving temporal structure.

Architecture and training. We instantiate a generator G and discriminator D as two-layer Multi-Layer Perceptrons (MLPs) [74], [75] with 256 hidden units, ReLU activations, and dropout for regularization. The generator maps $G : \mathbb{R}^d \rightarrow \mathbb{R}^d$ with a \tanh output layer, while the discriminator maps $D : \mathbb{R}^d \rightarrow [0, 1]$. Optimization follows Algorithm 3 using binary cross-entropy losses and Adam optimizers with a learning rate of 10^{-4} . To improve training stability, we apply label smoothing for real and fake targets and add Gaussian noise

to both. Training proceeds for a fixed number of epochs with mini-batches, and seeds are fixed to ensure reproducibility.

Data preparation. GAN training operates on the same per-window data stream. To match the generator’s \tanh output layer, inputs are linearly scaled to $[-1, 1]$. GAN training uses the clean, non-augmented, split to avoid leakage. Augmentation occurs *after* training, during robustness testing.

Injection. After training, we generate a candidate pool of points $\mathcal{C} = \{x_k^{(g)}\}_{k=1}^K$ with $x_k^{(g)} = G(z_k)$, $z_k \sim \mathcal{N}(0, I)$. We score candidates with D and measure ambiguity against the decision boundary via

$$\delta_k = |D(x_k^{(g)}) - \tau| \quad (7)$$

where τ denotes the discriminator decision threshold separating normal from anomalous samples. We then select the B most ambiguous points by minimizing the total ambiguity:

$$\mathcal{I}_B^* = \arg \min_{\mathcal{I} \subseteq \{1, \dots, K\}, |\mathcal{I}|=B} \sum_{k \in \mathcal{I}} \delta_k, \quad \mathcal{X}_B^* = \{x_k^{(g)} : k \in \mathcal{I}_B^*\}. \quad (8)$$

For surrogate labels we use the same boundary:

$$\hat{y}(x) = \mathbb{I}[D(x) \geq \tau], \quad x \in \mathcal{X}_B^*, \quad (9)$$

yielding an augmented set that contains both near-normal ($\hat{y}=0$) and near-anomalous ($\hat{y}=1$) behaviors.

Temporal integration policy. To respect chronology, we interleave the selected borderline points \mathcal{X}_B^* into the stream at regular intervals within sliding windows, using an injection budget ρ , default $\rho \approx 0.1$ of the original number of samples. Integration preserves ordering, updates label masks, and records injection indices for traceability and visualization.

Evaluation and ranking. Each candidate model is re-evaluated on the GAN-augmented data. Then we compute F1 and/or AUC-PR and produce model ranking. This ranking forms one of the outputs of the robustness block.

2) *Off-by-Threshold (SBA):* This analysis measures sensitivity to deviations from normality by injecting statistical border points, based on local context. Given a time series $X \in \mathbb{R}^{d \times n}$, SBA injects synthetic anomalies at regular intervals by adding zero-mean Gaussian noise with per-feature scaling derived from a contextual window of length w_{ctx} . For each injected point, we draw

$$\tilde{x}_j \sim \mathcal{N}(0, \sigma_j^2 s^2), \quad s \sim \mathcal{U}[\gamma_{\min}, \gamma_{\max}], \quad (10)$$

where σ_j is the local standard deviation for feature j in the contextual window. Points with $s \leq 1$ are labeled normal ($y=0$), while points with $s > 1$ are labeled as statistical border anomalies ($y=1$), creating a dataset with known labels near decision boundaries. We then evaluate all candidate models on this augmented data, compute F1 and AUC-PR, and derive an SBA ranking that identifies models robust to near-threshold variations. The perturbed data is not re-used in other components of our system.

3) *Monte Carlo Simulation (MC)*: We perform Monte Carlo stress testing to evaluate model stability under diverse anomaly patterns. In each of R randomized trials, synthetic anomalies of varying magnitudes ($s \in [\gamma_{\min}, \gamma_{\max}]$) are injected at random positions in the input streams, and all models are re-evaluated. The resulting F1 and/or AUC-PR scores are averaged across trials to obtain a robust Monte Carlo ranking [80].

C. Markov-Based Rank Aggregation

We aggregate multiple rankings over the same model set $\mathcal{D} = \{1, \dots, D\}$ into a single consensus ordering via a Markov-chain construction from pairwise preferences [81]. Given rankings $\mathcal{R} = \{r^{(1)}, \dots, r^{(K)}\}$, we count preferences to form $C \in \mathbb{R}^{D \times D}$, where C_{ij} is the number of times model i is ranked ahead of j . We then obtain a row-stochastic transition matrix P by normalizing rows:

$$P_{ij} = \begin{cases} \frac{C_{ij}}{\sum_{\ell \neq i} C_{i\ell}}, & \text{if } j \neq i \text{ and } \sum_{\ell \neq i} C_{i\ell} > 0, \\ 0, & \text{if } j = i \text{ and } \sum_{\ell \neq i} C_{i\ell} > 0, \\ \frac{1}{D}, & \text{if } \sum_{\ell \neq i} C_{i\ell} = 0, \end{cases} \quad (11)$$

The consensus score is the stationary distribution v^* satisfying $v^* = v^*P$, which we compute by power iteration from a uniform initialization until ℓ_1 convergence. Models are finally sorted in *descending* order of v^* to produce the consensus ranking. We apply this operation (i) to aggregate the three robustness rankings, GAN, off-by-threshold, Monte Carlo, and (ii) to aggregate the robustness with the LinTS ranking.

D. Robustness Evaluation Mechanism

We evaluate the robustness of base models via three tests that probe orthogonal modes: (I) As explained in Equation 7, the *GAN* learns from clean data and produces points near the discriminator’s decision boundary. Models that only memorize training patterns fail on these cases. (II) *SBA* tests sensitivity of base models by injecting Gaussian noise scaled by local standard deviation within overlapping windows. Unlike GAN’s distribution-learned perturbations, SBA uses context-dependent noise to probe decision boundaries under measurement uncertainty, for example peak-period temperature fluctuations near the threshold. (III) The *Monte Carlo simulation* tests stochastic stability through random perturbations. Providing a randomized baseline independent of GAN or SBA approaches.

Preventing circularity. Models are never trained on components perturbed augmented data. As depicted in Figure 2, each method operates on a data copy, and we aggregate model rankings via Markov aggregation (Equation 11), which combines rankings from all three tests into a consensus ranking.

Example. On SKAB 1-1, we observe the following behaviour: With 100 epochs the *GAN* generates perturbed flow-temperature pairs near the 32 L/min operating point, choosing *LOF_2* for its resilience to borderline cases. *SBA*: injects near-threshold degradation (31-33 L/min), also selecting *LOF_2* for boundary sensitivity. *MC* adds noise across all features, e.g., accelerometer vibration during pump failure, selecting *CBLOF*.

After Markov aggregation, *LOF_2* is selected as the single-model candidate. This divergence from LinTS’s choice (*RNN*) is instructive: while *RNN* excels on normal temporal patterns (32 L/min steady-state), it fails to generalize to regimes absent from training. *LOF_2*’s density-based approach treats this as a sparse region in the 8-dimensional sensor space, detecting multi-sensor correlation shifts (flow-temperature-vibration coupling) that violate normal operational density.

E. Online Anomaly Detection and Adaptive Model Selection

In RAMSeS, online real-time anomaly detection operates in a continuously adaptive loop. At initialization, both the top-ranked single model and the optimized ensemble (learned offline) are deployed. For each new batch of incoming data, both candidates emit anomaly decisions. These can be presented for human/system feedback. Figure 6 shows the complete cycle, initialization, detection, feedback, and refinement.

Window-based adaptation mechanism. To implement the online adaptive loop, the time series is partitioned into 80% for offline model selection and 20% for online inference. The online portion is further divided into overlapping windows of size $w = 0.05 \times \text{online_size}$ with step size $s = 0.05w$ (95% overlap) to preserve temporal context. Every N windows, the system triggers re-optimization: a sliding window is constructed by concatenating the N most recent online windows while dropping an equal number of samples from the beginning of the offline data, maintaining constant training size and oracle knowledge [13], [49]. This sliding mechanism adapts to distribution shifts while preventing catastrophic forgetting of earlier patterns. During this re-optimization step the selected ensemble or model might be updated based on performance.

Example. During the online phase on SKAB 1-1, RAMSeS deploys the ensemble $\{\text{RNN}, \text{KNN}_2, \text{LOF}_2, \text{CBLOF}\}$ and the single model *LOF_2*. At timestamp 1030, the pump failure anomaly occurs: Volume Flow drops from 32 L/min to 3.2 L/min coupled with Temperature decrease from 75°C to 74°C. The ensemble branch detects this multi-sensor correlation with $F1=0.87$ by combining *LOF_2*’s density detection with *CBLOF*’s cluster analysis. The single-branch *LOF_2* fails to detect the anomaly ($F1=0.52$), missing the subtle temperature correlation. Despite *LOF_2*’s theoretical advantage for density shifts, the isolated single-model deployment lacks the multi-sensor context that the ensemble provides through *CBLOF*’s cluster analysis. After $N = 5$ windows, RAMSeS triggers re-optimization: the ensemble branch re-runs GA on the sliding window (concatenating recent online data while dropping early offline samples), confirming $\{\text{RNN}, \text{KNN}_2, \text{LOF}_2, \text{CBLOF}\}$ remains optimal. The single-branch re-executes LinTS and robustness tests, switching to *CBLOF* as the top choice based on its superior performance on cluster deviations observed in recent windows.

VI. EXPERIMENTS AND RESULTS

We analyze the strengths and weaknesses of RAMSeS with experiments on several well-known benchmarks.

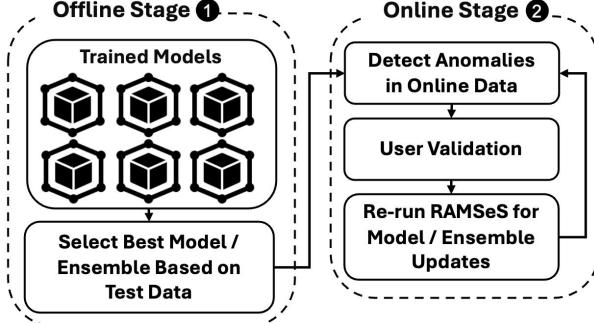


Fig. 6. Continuous online real-time model selection for anomaly detection.

Datasets We evaluate the performance of our framework on three diverse time-series datasets that vary in dimensionality, domain, and anomaly characteristics. The *UCR Anomaly Archive (UCR)* [21] contains average $\sim 73k$ time-points from 250 univariate time series (i_UCR) entities spanning domains, such as medicine and environment. The *Server Machine Dataset (SMD)* [30] comprises average $\sim 27k$ time-points from 28 server machines (SMD i-j) in a large-scale internet services infrastructure (38-dimensions). Each machine forms an entity with 38 features capturing CPU usage, memory utilization, and network throughput. The *Skoltech Anomaly Benchmark (SKAB)* [29] contains average $\sim 2k$ time-points, 34 time series (SKAB i-j) sequences from water-pump operations, each with 8 sensor features recorded in normal and faulty states.

Anomaly Detection algorithms Our pool of models follows the TSB-AutoAD [9] taxonomy in Table I, covering three families, Neural Networks, Statistical, and Foundation Models, with applicability to both univariate, and multivariate settings. FMs are excluded from the RAMSeS candidate pool, as they showed inconsistent performance. For completeness and comparability, we include FMs only in the *TSB-AutoAD* setting, which evaluates FMs in its testbed models [9].

A. Hyperparameter Configuration

Table II lists all hyperparameters used in RAMSeS. To avoid overfitting to specific datasets, we tuned these hyperparameters on a randomly selected 20% subset of time series from UCR, SMD, and SKAB. This provides sufficient diversity to capture varying characteristics, i.e., length, dimensionality, anomaly patterns. Unless otherwise noted, the selected values remain fixed across all remaining time series in our experiments.

GA-Ens. GA with $P \leq 20$ and $G \leq 20$ achieves F1 saturation while minimizing computational overhead. Mutation rate $\mu \leq 0.2$ sustains diversity without degrading performance. Random Forest (RF) is chosen as the meta-learner because it achieves comparable F1 to SVM while being faster. These choices reflect fundamental GA trade-offs: P controls exploration breadth, G control convergence depth, and mutation balances diversity versus stability.

LinTS. The single-model branch uses LinTS to balance exploration and exploitation. LinTS begins with $\epsilon_0 = 0.2$ (20% random exploration) and decays by 0.99 per window, shifting toward exploitation as evidence accumulates. The reward function weights F1 with $\alpha = 0.7$, which has been learned during

TABLE I
BASE MODELS GROUPED BY FAMILY: NEURAL NETWORKS (NN), STATISTICAL (STAT) OR FOUNDATION MODELS (FM) AND APPLICABILITY (U/M).

Base models	Category	Dim	Brief Description
AutoEncoder [82]	NN	U&M	Reconstruction error
RNN [83]	NN	U&M	Vanilla recurrent prediction error
DGHL [84]	NN	M	Gen-ConvNet with latent factor
LSTMVAE [25]	NN	U&M	LSTM-based VAE
Donut [85]	NN	U&M	VAE likelihood scoring
OmniAnomaly [86]	NN	U&M	Stochastic RNN likelihood
USAD [36]	NN	U&M	Adversarial autoencoders
TranAD [37]	NN	U&M	Self-conditioned + adversarial
TimeNet [87]	NN	U&M	Temporal-variation features
FITS [88]	NN	U&M	Frequency interpolation score
ABOD [89]	Stat	M	Angle-based outlier factor
(Sub)-MCD [90]	Stat	U&M	Robust Mahalanobis
Sub-OCSVM [91]	Stat	U&M	One-class SVM boundary
RM	Stat	U&M	Simple moving-average residuals
(Sub)-LOF [23]	Stat	U&M	Local density deviation
(Sub)-KNN [24]	Stat	U&M	k NN distance score
KMeansAD [92]	Stat	U&M	Distance to centroid
CBLOF [93]	Stat	U&M	Cluster-weighted LOF
POLY [94]	Stat	U	Local polynomial residuals
(Sub)-IForest [95]	Stat	U&M	Isolation via partitions
(Sub)-HBOS [96]	Stat	U&M	Histogram bin height
(Sub)-PCA [97]	Stat	U&M	Distance to PC subspace
OFA [98]	FM	U&M	Fine-tuned GPT-2
Lag-Llama [99]	FM	U	Decoder-only with lags
Chronos [100]	FM	U	Pretrained T5 on TS
TimesFM [101]	FM	U	Decoder-only with patching
MOMENT [102]	FM	U	Pretrained T5 encoder

Dim: U = univariate, M = multivariate. “(Sub)-” denotes a subsequence variant.

parameter tuning. A 50-window exploration horizon provides sufficient statistical evidence for reward estimation while maintaining responsiveness to regime changes. Lower settings, e.g., 10 windows, led to premature convergence on suboptimal models, while larger horizons, e.g., 200 windows, delayed adaptation when anomaly patterns shifted. This aligns with guidance for stable arm selection in bandit algorithms [73].

GAN. Adversarial perturbations test robustness by generating borderline anomalies that follow time-series properties. The generator uses a dense layer with 256 units and dropout of 0.4 to prevent memorization of training data. Following the recommendations of [74], [75], both generator and discriminator are trained with Adam optimizer at learning rate 10^{-4} . Training runs for 100 epochs to ensure stable generation of boundary cases [75]. Discriminator loss plateaus after ~ 80 epochs for all datasets. The additional 20 epochs provide robustness and improve boundary case quality.

SBA and Monte Carlo hyperparameters. SBA sensitivity analysis uses 10% near-threshold augmentation to probe decision boundaries under measurement uncertainty. Monte Carlo noise injection runs ≤ 10 trials as a stochastic stress test.

Online phase hyperparameters. Window size is set to 5% of the online data length, ensuring adaptation to varying time series lengths. Consecutive windows overlap by 95% using a step size of 5% of the window size, creating a sliding window effect. Re-optimization is triggered every ≥ 5 windows to balance model freshness against computational overhead.

TABLE II
RAMSeS HYPERPARAMETERS FOR REPRODUCIBILITY

Entity	Parameter	Value	Rationale
GA-Ens	Population size (P)	≤ 20	F1 saturation (VI-H)
	Number of Generations (G)	≤ 20	Cost-effective (VI-H)
	Mutation rate (μ)	≤ 0.2	Sustains diversity
	Meta-learner	RF	Nonlinear aggregation
LinTS	Windows	50	Posterior convergence
	Epsilon (ϵ_0)	0.2	Early exploration
	(ϵ_0) decay	0.99	Exploitation preferred
	Reward (α)	0.7	Detection accuracy
GAN	Epochs	100	Adversarial converge
	Learning rate	10^{-4}	Stable training
	Dropout	0.4	Prevent overfitting
SBA	Injection factor	0.1	10% test data
	Scale factor	[0.95, 1.05]	Near-threshold
MC	Simulations	≤ 10	Statistical confidence
	Noise level (σ)	0.1	Gaussian robustness
Online	Window size	5%	of online data
	Window step	5%	max overlap
	Re-opt (N)	≥ 5 windows	Adaptive balance

B. Baselines for Comparison

We compare RAMSeS against the top Automated TSAD architectures. Unless otherwise noted, we report *F1* computed on test splits using the range-based event evaluation, so that sequence-level detections are assessed consistently across time series. For each time series, the ensemble and single-model selections are produced *without access to labels*.

(1) *Automated Ensemble and single model selection (TSB-AutoAD [9])*. We use the TSB-AutoAD framework of Liu et al. [9], which provides a unified benchmark and reference implementations of automated TSAD solutions spanning model selection and model ensembling.

(2) *Single model selection (UMS) [8]*. UMS selects a single model by aggregating surrogate criteria prediction error, model centrality, and performance under synthetic anomaly injection.

(3) *Ensemble selection (AutoTSAD [3])*. AutoTSAD is an unsupervised system for univariate time series that combines data generation, model tuning, and selective score ensembling.

(4) *Meta-learning model selection (MSAD [49])*. MSAD uses time series classification to predict which anomaly detector will perform best. A pre-trained classifier operates on extracted features from windowed segments to select a detector.

Tuning policy. Unless otherwise stated, we do *not* tune base models in RAMSeS: each base model is instantiated with a set of randomly sampled configurations, and this fixed pool is used for all experiments.

C. System Level Comparison

Table III compares RAMSeS against four TSAD baselines: TSB-AutoAD, UMS, MSAD, and AutoTSAD. All experiments were conducted on a single machine (AMD EPYC 7543P 32-Core @ 2.80 GHz, 503 GB RAM, Linux 6.1.0), using the hyperparameter values in Table II.

Detection Accuracy. Table III shows that RAMSeS achieves the highest F1 scores on UCR and SMD datasets, outperforming all baselines. Notably, RAMSeS demonstrates strong per-

TABLE III

AVERAGE SYSTEM-LEVEL PERFORMANCE: RAMSeS VS. AUTOMATED TSAD BASELINES ACROSS ALL DATASETS AND TIME SERIES (TS). TIME (T) IN SECONDS, MEMORY (M) IN MB. BEST IN **BOLD**. TIMEOUT IS 48 HOURS PER TIME SERIES.

Method	UCR (250 TS)			SMD (27 TS)			SKAB (18 TS)		
	F1	T	M	F1	T	M	F1	T	M
TSB	0.45	5012	534	0.55	4603	825	0.68	368	702
UMS	0.66	2661	2580	0.80	1520	1796	0.86	546	1312
MSAD	0.07	14918	639	0.13	260	608	0.54	74	286
AutoTSAD		Timeout			Not applicable			Not applicable	
RAMSeS	0.83	1704	785	0.89	586	618	0.86	647	512

formance on large-scale univariate data, e.g., UCR, and high-dimensional multivariate scenarios, e.g., SMD. On SKAB, RAMSeS and UMS achieve the highest F1 performance. SKAB’s short sequences with simple patterns are well-suited to UMS’s lightweight heuristics. This suggests, that while RAMSeS excels on complex, large-scale workloads, simpler methods may suffice for short, well-structured time series.

Computational Efficiency. As depicted in Table III, RAMSeS achieves faster runtime than TSB and UMS on UCR and SMD. Moreover, RAMSeS maintains moderate memory consumption (500–800 MB) across all datasets, consuming less than UMS and less than TSB on SMD and SKAB. MSAD achieves the lowest runtime on SMD and SKAB by using a pre-trained time series classifier for lightweight prediction-based selection. However, on longer time series as in UCR, MSAD is significantly slower than RAMSeS due to training latency. AutoTSAD times out on UCR after 48 hours per entity due to its exhaustive hyperparameter search for base models.

Practical Implications. RAMSeS manages accuracy-speed-memory trade-offs across workloads, excelling on large-scale and high-dimensional scenarios. For simple datasets (<2k timesteps), lightweight selectors may suffice, but RAMSeS’s two-branch architecture ensures robust performance when dataset characteristics, e.g., length, dimensionality, pattern complexity, are unknown a priori.

D. Component-level Runtime and Memory Analysis

To assess the viability of RAMSeS, we report per-component runtime and system-level memory footprint across the three datasets for our offline phase. Table IV reports average runtime and memory across benchmark datasets. Runtime is reported by module. Markov aggregation contributes negligible overhead (<0.01s) and is omitted. RAMSeS’s runtime and memory depend on the time series characteristics: sequence length drives the number of LinTS windows as more windows require more posterior updates, dimensionality scales meta-learner training cost in GA-Ens as higher-dimensional feature spaces increase meta-learner complexity and latency. The memory footprint is lower than 800MB for all datasets.

Bottleneck on UCR. Table IV shows that LinTS consumes 1,445 seconds on UCR, representing 84.8% of the total single-branch runtime. This overhead stems from posterior updates over 50 windows on long univariate series with an average of 73,000 timesteps. Each window triggers a Gaussian update

TABLE IV
AVERAGE END-TO-END RUNTIME (SECONDS) AND MEMORY (MB) PER ENTITY.

Dataset	GA	LinTS	GAN	SBA	MC	Single Branch	System Memory
UCR	75	1445	71	39	73	1628	785
SMD	280	173	38	30	64	306	618
SKAB	134	337	75	42	88	513	512

involving mean and covariance inversion for each candidate model. With UCR’s longer sequences, the number of windows grows linearly with series length, and each covariance inversion scales cubical in the feature dimension. Consistent with LinTS complexity [73], posterior updates dominate runtime when the number of exploration rounds is large.

Bottleneck on SMD. The comparably high dimensionality of 38 features dominates the computational cost of both base models inference and meta-learner fitting. Table IV shows that GA-Ens accounts for 47.7% of the total runtime on SMD due to the meta-learner, which trains over 38 features across $P = 20$ models for $G = 20$ generations. Each generation requires training the RF on stacked predictions from base models.

Bottleneck on SKAB. Table IV shows that RAMSeS exhibits more evenly distributed costs across components on SKAB compared to UCR and SMD. The shorter sequence length, average 2,000 timesteps, reduces the number of LinTS windows, mitigating the posterior update bottleneck observed on UCR. The moderate dimensionality of 8 features, compared to 38 in SMD, reduces meta-learner training cost in GA-Ens.

E. Online Inference, Scalability and Window Size Effect

We evaluate RAMSeS with regard to: (1) online per-window performance, latency and memory, (2) scalability vs. candidate pool size, and (3) sensitivity to window size. All experiments use hyperparameters from Table II.

1) *Online Phase Performance:* To simulate the real-time scenarios we inject 8 random regime shifts to the online stream and test the performance of RAMSeS against UMS and MSAD baseline. AutoTSAD is only applicable to univariate time-series and TSB is also excluded due to low performance, $F1 \leq 10\%$. Fig. 7 compares average F1 score, latency, and memory consumption across 50 overlapping windows with 5% step size, on SMD for: RAMSeS with re-optimization (every 10 windows), RAMSeS without re-optimization (fixed selection), and the baselines. Both RAMSeS variants achieve on average 40% higher F1 than UMS and 60% higher F1 compared to MSAD. The re-optimization variant shows average performance gain after each update cycle, most notably after window 10 (the first update), where F1 jumps relative to the fixed variant, establishing it as the highest performer. However, re-optimization incurs higher memory during updates. This trade-off suggests practitioners should enable or disable re-optimization depending on whether their use case prioritizes accuracy or resource constraints.

2) *Online Scalability vs. Candidate Pool Size:* Fig. 8 shows runtime and memory consumption in relation to the candidate pool size. Both resource footprints are approximately constant from 3 to 14, which is expected as RAMSeS is only

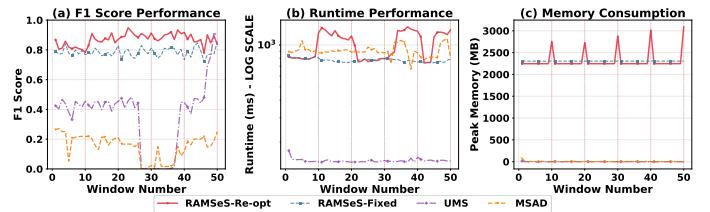


Fig. 7. Online phase avg. F1 & overhead on SMD for RAMSeS & Baselines

TABLE V
BRANCH AVERAGE PERFORMANCE COMPARISON BASED ON F1-SCORE.

Dataset	Domain	N	GA-Ens $F1 \pm \sigma$	Single Sel. $F1 \pm \sigma$	Final Sel. $F1 \pm \sigma$
UCR-Med	Healthcare	68	0.77 ± 0.06	0.80 ± 0.09	0.82 ± 0.07
UCR-Env	Environment	13	0.62 ± 0.01	0.80 ± 0.09	0.81 ± 0.04
UCR-Ind	Industry	33	0.72 ± 0.07	0.80 ± 0.09	0.82 ± 0.07
SMD	IT	25	0.88 ± 0.07	0.87 ± 0.05	0.89 ± 0.05
SKAB	Industry	15	0.78 ± 0.04	0.55 ± 0.30	0.80 ± 0.07

evaluating and using the models not retraining them in online phase for each window.

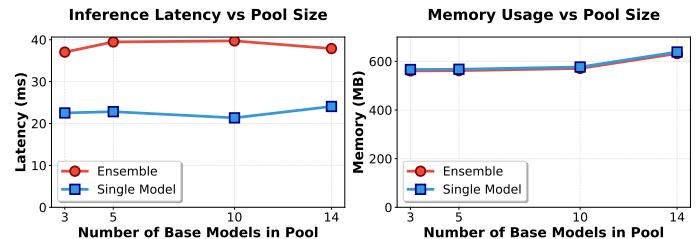


Fig. 8. Scalability: F1 and latency vs. candidate pool size for SKAB dataset.

3) *Altering Online Window Size:* Fig. 9 examines average performance and latency vs. window sizes varying from 22 to 114 timesteps over three time series from SMD. RAMSeS performs poorly with smaller window sizes due to insufficient context fed to the detectors, and the detectors exhibit behavior similar to random guessing. Performance increases significantly as window size grows, reaching a plateau where further increases yield diminishing returns. Runtime overhead increases with larger window sizes because each window requires processing more samples for detection.

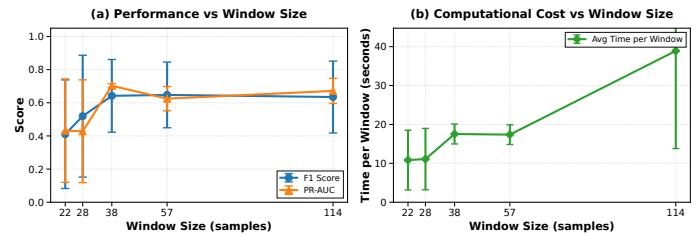


Fig. 9. Window size sensitivity on SKAB dataset.

F. Branch Performance Comparison

Table V compares GA-Ens against single-model selection across datasets and domains, using the exact hyperparameter values from Table II fixed throughout all experiments.

Univariate vs. Multivariate. Table V shows the single-branch achieves higher F1 across all univariate UCR subsets. With 1

feature, GA-Ens meta-learner lacks information about relevant cross-feature relationships. In contrast, single-model selection identifies detectors that robustly capture temporal patterns within the univariate stream through LinTS exploration and stress testing. The ensemble-branch excels on multivariate series but highly depends on the relationships between features. On SMD with 38 features both branches perform similarly. However on SKAB with 8 features, the ensemble branch captures cross-sensor dependencies, e.g., valve failure manifesting across pressure and temperature simultaneously, that single-model selection cannot exploit. On *UCR-Medical* LinTS’s evaluation captures regime shifts, e.g., arrhythmia onset, better than GA-Ens. GAN generates borderline anomalies that validate single-model generalization. On *UCR-Environment*: Big single-branch F1 advantage =0.80 vs. GA-Ens=0.62. SBA, 10% injection, scale [0.95,1.05]) matches temperature anomaly patterns (near-threshold spikes). On *Industrial (SKAB)*: Sensor correlations (valve failure → pressure drop + temperature rise) require ensemble. SKAB’s poor single-branch performance stems from insufficient data for LinTS: <30 points per window, which is inadequate for stable covariance estimation.

Guideline for practitioners. Dimensionality and cross-feature dependencies determine the branch choice: ensemble methods leverage complex inter-feature relationships in multivariate settings, while single-model selection suffices for simple patterns with sufficient data points. Table V confirms this, as single-branch excels on univariate UCR, GA-Ens excels on multivariate SMD/SKAB. However, we can also observe that single-branch takes longer on the long UCR series, while the ensemble-branch adds overhead for high-dimensional data, requiring practitioners to trade-off accuracy versus efficiency. *In summary*, (1) when accuracy is paramount, data is multivariate with inter-feature relationships, prefer ensemble-branch. (2) When latency is critical and data is univariate, prefer single-branch. (3) For unknown characteristics, deploy both branches and select adaptively based on online performance.

G. Ensemble Branch Analysis

To better understand the ensemble branch, we investigate its hyperparameter effects and runtime efficiency.

1) *Effect of Population Size and Generations:* To evaluate the convergence of GA-Ens, we varied the P and G for $\mu = 0.2$ and SVM as the meta-learner. Table VI reports the resulting F1 scores. For $P \in \{10, 50, 100\}$ and $G \in \{100, 1000, 10000\}$, both the chosen ensembles and their F1 remain nearly stable. For example, entities such as SKAB_1-1, and SMD_3-10 consistently converge to the same or very similar sets of base models. This indicates that the GA reliably identifies strong ensemble configurations without extensive tuning of P or G .

2) *Effect of the Mutation Rate (μ):* Table VII reports the F1 for each setting. To study the impact of μ in the GA-Ens process, we vary μ in $\{0, 0.05, 0.2, 1.0\}$ while keeping the $P = 100$, $G = 1000$, and meta-learner = RF, to assess the effect of mutation rate on heavy loading. For all values of μ , the resulting ensembles exhibit *stable* F1 scores. For example, on SKAB_1-1 and SKAB_1-2, F1 remains above

TABLE VI
F1 ACROSS P SIZES AND G . META-LEARNER= SVM AND $\mu= 0.2$.

Entity	P 10			P 50			P 100		
	G 100	1000	10000	100	1000	10000	100	1000	10000
SKAB 1-1	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
SKAB 1-2	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
SMD 3-4	0.77	0.76	0.76	0.76	0.76	0.76	0.76	0.76	0.76
SMD 3-10	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
011_UCR	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77
012_UCR	0.79	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78

TABLE VII
F1 vs. μ . $P = 100$, $G = 1000$,
META-LEARNER = RF.

Entity	0	0.05	0.2	1	Entity			SVM	RF	LR
					SKAB 1-1	SKAB 1-2	SMD 3-4	SMD 3-10	011_UCR	012_UCR
SKAB 1-1	0.87	0.87	0.87	0.87	0.87	0.87	0.76	0.71	0.77	0.78
SKAB 1-2	0.86	0.86	0.86	0.86	0.86	0.86	0.75	0.71	0.76	0.63
SMD 3-4	0.74	0.75	0.74	0.75	0.76	0.76	0.76	0.71	0.78	0.61
SMD 3-10	0.76	0.76	0.76	0.76	0.76	0.76	0.76	0.71	0.78	0.62
011_UCR	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.61
012_UCR	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.78	0.79	0.62

TABLE VIII
F1 vs. META-LEARNS. $P = 100$, $G = 1000$, $\mu = 0.2$.

Entity	SVM	RF	LR	Entity		
				SKAB 1-1	SKAB 1-2	SMD 3-4
SKAB 1-1	0.87	0.87	0.78	0.87	0.86	0.79
SKAB 1-2	0.87	0.87	0.78	0.86	0.86	0.66
SMD 3-4	0.76	0.76	0.74	0.76	0.74	0.63
SMD 3-10	0.71	0.71	0.63	0.71	0.71	0.63
011_UCR	0.77	0.77	0.61	0.77	0.78	0.61
012_UCR	0.78	0.78	0.62	0.78	0.79	0.62

0.86, for both, despite differences in ensemble composition. A moderate μ , e.g., 0.2, sustains diversity and helps avoid premature convergence. High μ values, e.g., 1.0, do not degrade performance in our study, suggesting that P contains multiple viable configurations.

3) *Effect of the Meta-Learner:* Table VIII summarizes the F1 for the GA-Ens process and meta-learner variants. To assess the influence of the meta-learner in our *GA-Ens*, we compare SVM, RF, and LR, for $P = 100$, $G = 1000$, and $\mu = 0.2$. Across nearly all entities, SVM outperforms both RF and LR. For example, on SKAB_1-1 and SKAB_1-2, SVM achieves an F1 score of 0.87 versus RF at 0.86 and LR at 0.78, for SKAB_1-1 and 0.79 for SKAB_1-2.

H. Runtime and Convergence of RAMSeS Ensemble

To analyze the runtime/accuracy trade-off of the *GA-Ens* during the *offline* selection stage we focus on SKAB1_1. For each configuration, we ran three repeats and report medians with interquartile range. Wall-clock time includes the GA search and meta-learner training with cached base-model scores. The *online* stage operates with the chosen ensemble and only triggers lightweight adaptation after accumulating user feedback as explained in V-E. Fig. 10A depicts runtime to grow linearly with *work* ($P \times G$). A power-law fit describes the variance, as shown in Fig. 10A. F_1 reaches a plateau quickly as shown in Fig. 10D. Figures. 10B, E depict the influence of μ on runtime and F1. At a fixed workload ($P=50$, $G=10$), runtime increases moderately with μ for RF/SVM, while LR remains nearly flat. F_1 is insensitive to μ for all meta-learners. Across all workloads of P and G , RF achieves near-top F_1 and is faster than SVM and LR as depicted in Figs. 10C, F, aligning with our results in Table VIII.

I. Single Model Selection Branch Analysis

To better understand behavior of single-model selection branch, Table IX lists F1 score for four variants of our scoring module: *LinTS*, *GAN*, *SBA*, and *MC*. Across all entities, the

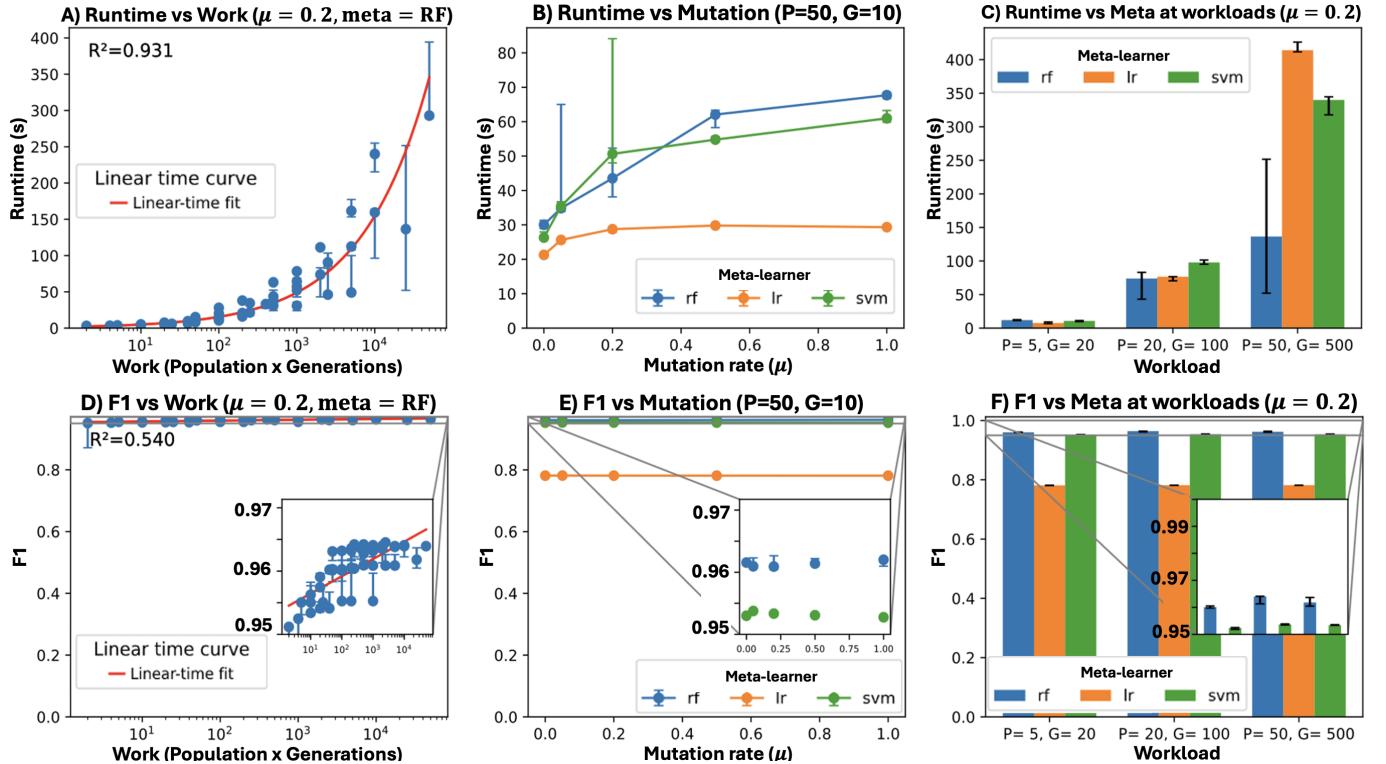


Fig. 10. Runtime-accuracy of the GA-based stacking ensemble on SKAB1_1. (A) runtime vs. work ($P \times G$), $R^2=0.931$. (B) runtime vs. μ across meta-learners. (C) runtime vs. meta-learner at three workloads. (D) F_1 vs. work, $R^2=0.540$. (E) F_1 vs. μ . (F) F_1 vs. meta-learner. Points are medians over 3 runs.

selected models achieve nearly identical F_1 and agree on the same model. This shows that the components are not contradicting, but supporting each other.

TABLE IX
F1 OF COMPONENTS.

Entity	LinTS	GAN	SBA	MC
SKAB 1-1	0.88	0.88	0.88	0.87
SKAB 1-2	0.87	0.88	0.88	0.87
SMD 3-4	0.90	0.86	0.90	0.90
SMD 3-10	0.88	0.89	0.88	0.88
011_UCR	0.86	0.86	0.86	0.86
012_UCR	0.86	0.86	0.86	0.88

TABLE X
RANKINGS OF COMPONENTS ON SMD 3–10.

	GAN	SBA	MC	Rob.	LinTS	Final
RNN4	RNN3	RNN3	RNN3	RNN3	RNN3	RNN3
RNN3	RNN4	RNN4	RNN4	RNN4	LOF4	RNN1
RNN2	RNN2	RNN2	RNN2	RNN1	RNN4	RNN2
RNN1	RNN1	RNN1	RNN1	RNN1	k-NN3	RNN2
LOF1	LOF1	LOF1	LOF1	LOF1	RNN4	LOF4
LOF2	LOF2	LOF2	LOF2	LOF2	k-NN1	LOF1

J. Markov-Based Rank Aggregation

Table X reports the per-heuristic rankings, the intermediate aggregated rankings within the robustness group and *LinTS*, and the final ranking. The results show strong agreement among the robustness-based methods, with models such as RNN3 and RNN4 consistently occupying the top positions. *LinTS* also ranks RNN3 highly, while introducing a slight variance, e.g., LOF4 and RNN1 appear among the top-5 candidates. Markov aggregation over robustness signals yields an ordering favoring RNN3. Notably, *final* aggregated ranking selects RNN3, RNN1, and RNN4 as the top three models. This outcome aligns with the F_1 results, confirming that the aggregation procedure integrates complementary signals.

VII. LIMITATIONS AND FUTURE DIRECTIONS

We now point out several limitations of RAMSeS.

Memory Consumption at inference time. Re-optimization during online phase introduces additional computational overhead. As depicted in Fig. 7, periodic re-execution of GA-Ens, LinTS and GAN incurs higher memory, limiting applicability in resource-constrained environments. Future work could adopt re-optimization triggered only when distribution shifts are detected or accuracy degradation.

Meta-Hyperparameter Configuration. As described in Table II, RAMSeS introduces additional hyperparameters, which may limit the ease of deployment. Future work can automate tuning via transfer learning [103].

Dataset Scope. Our evaluation focuses on univariate and multivariate numerical time series. RAMSeS's applicability to categorical, mixed-type, or event-based sequences remains unexplored and future work.

Interpretability. The ensemble's meta-learner and Markov aggregation operate as black-box mechanisms. Future work could integrate explainability techniques such as SHAP [104], [105] values or even LLMs to enhance transparency and trust.

VIII. CONCLUSION

We present RAMSeS, a novel framework to address the persistent challenges of model selection for TSAD. In comprehensive evaluations across diverse time-series, RAMSeS outperforms prior methods in both accuracy and adaptability. A central strength of the proposed framework lies in its integration of ensemble learning with adaptive model optimization. By leveraging ensemble techniques in combination with genetic algorithms, RAMSeS intelligently selects and coordinates multiple base models, capturing diverse anomalies.

patterns. Moreover, RAMSeS introduces a dynamic model selection mechanism based on Thompson sampling and ϵ -greedy reinforcement learning. This enables continuous policy refinement in response to shifting data distributions. To further ensure robustness, the framework incorporates adversarial and statistical validation through GAN, MC, and sensitivity analyses. These components assess the resilience of candidate models under noisy, imbalanced, or perturbed conditions providing robustness guarantees.

AI-GENERATED CONTENT ACKNOWLEDGEMENT

We used ChatGPT (GPT-5) and Grammarly for grammar and spelling corrections, light stylistic polishing, and LaTeX table formatting. ChatGPT also suggested concise wording for figure/axis labels and helped create simple decorative icons for schematic illustrations. No technical content, datasets, analyses, or experimental results were generated by AI.

REFERENCES

- [1] E. Keogh, J. Lin, and A. Fu, “Hot sax: Efficiently finding the most unusual time series subsequence,” *Proceedings of the Fifth IEEE International Conference on Data Mining*, 2005.
- [2] D. Wagner, T. Michels, F. C. Schulz, A. Nair, M. Rudolph, and M. Kloft, “Timesead: Benchmarking deep multivariate time-series anomaly detection,” *TMLR*, 2023.
- [3] S. Schmidl, F. Naumann, and T. Papenbrock, “Autotsad: Unsupervised holistic anomaly detection for time series data,” *PVLDB*, vol. 17, no. 11, pp. 2987–3002, 2024.
- [4] H. Jiang, C. Liu, Q. Jin, J. Paparrizos, and A. J. Elmore, “PIDS: attribute decomposition for improved compression and query performance in columnar storage,” *PVLDB*, vol. 13, no. 6, pp. 925–938, 2020.
- [5] K. Hishida, C. Liu, J. Paparrizos, and A. Elmore, “Beyond compression: A comprehensive evaluation of lossless floating-point compression,” *PVLDB*, vol. 18, no. 11, pp. 4396–4409, 2025.
- [6] J. Paparrizos, C. Liu, B. Barbaroli, J. Hwang, I. Edian, A. J. Elmore, M. J. Franklin, and S. Krishnan, “Vergedb: A database for iot analytics on edge devices,” in *CIDR*, 2021.
- [7] Q. Liu, P. Boniol, T. Palpanas, and J. Paparrizos, “Time-series anomaly detection: Overview and new trends,” *PVLDB*, vol. 17, no. 12, pp. 4229–4232, 2024.
- [8] M. Goswami, C. I. Challu, L. Callot, L. Minorics, and A. Kan, “Unsupervised model selection for time series anomaly detection,” in *ICLR*, 2023.
- [9] Q. Liu, S. Lee, and J. Paparrizos, “Tsb-autoad: Towards automated solutions for time-series anomaly detection,” *PVLDB*, vol. 18, no. 11, pp. 4364–4379, 2025.
- [10] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *CSUR*, vol. 41, no. 3, pp. 1–58, 2009.
- [11] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich, “A survey on anomaly detection for technical systems using lstm networks,” *Computers in Industry*, vol. 131, p. 103498, 2021.
- [12] P. Boniol, M. Linardi, F. Roncallo, T. Palpanas, M. Meftah, and E. Remy, “Unsupervised and scalable subsequence anomaly detection in large data series,” *VLDBJ*, vol. 30, no. 6, pp. 909–931, 2021.
- [13] P. Boniol, J. Paparrizos, T. Palpanas, and M. J. Franklin, “SAND: streaming subsequence anomaly detection,” *PVLDB*, vol. 14, no. 10, pp. 1717–1729, 2021.
- [14] P. Boniol, J. Paparrizos, Y. Kang, T. Palpanas, R. S. Tsay, A. J. Elmore, and M. J. Franklin, “Theseus: Navigating the labyrinth of time-series anomaly detection,” *PVLDB*, vol. 15, no. 12, pp. 3702–3705, 2022.
- [15] P. Boniol, J. Paparrizos, and T. Palpanas, “New trends in time series anomaly detection,” in *EDBT*, 2023, pp. 847–850.
- [16] ———, “An interactive dive into time-series anomaly detection,” in *ICDE*, 2024, pp. 5382–5386.
- [17] Q. Liu and J. Paparrizos, “The elephant in the room: Towards A reliable time-series anomaly detection benchmark,” in *NeurIPS*, 2024.
- [18] S. Bhattacharyya, S. Jha, K. K. Tharakunnel, and J. C. Westland, “Data mining for credit card fraud: A comparative study,” *Decis. Support Syst.*, vol. 50, no. 3, pp. 602–613, 2011.
- [19] H. Liao, C. R. Lin, Y. Lin, and K. Tung, “Intrusion detection system: A comprehensive review,” *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, 2013.
- [20] M. Goldstein and S. Uchida, “A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data,” *PloS one*, vol. 11, no. 4, p. e0152173, 2016.
- [21] R. Wu and E. J. Keogh, “Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress,” *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 3, pp. 2421–2429, 2023.
- [22] D. Campos, T. Kieu, C. Guo, F. Huang, K. Zheng, B. Yang, and C. S. Jensen, “Unsupervised time series outlier detection with diversity-driven convolutional ensembles,” *PVLDB*, vol. 15, no. 3, pp. 611–623, 2021.
- [23] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander, “LOF: identifying density-based local outliers,” in *SIGMOD*, 2000, pp. 93–104.
- [24] S. Ramaswamy, R. Rastogi, and K. Shim, “Efficient algorithms for mining outliers from large data sets,” in *SIGMOD*, 2000, pp. 427–438.
- [25] D. Park, Y. Hoshi, and C. C. Kemp, “A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder,” *IEEE Robotics Autom. Lett.*, vol. 3, no. 2, pp. 1544–1551, 2018.
- [26] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, “Long short term memory networks for anomaly detection in time series,” in *ESANN*, 2015.
- [27] D. Jung, N. Ramanan, M. Amjadi, S. R. Karingula, J. Taylor, and C. J. N. C. Jr., “Time series anomaly detection with label-free model selection,” *CoRR*, vol. abs/2106.07473, 2021.
- [28] X. Qiu, Z. Li, W. Qiu, S. Hu, L. Zhou, X. Wu, Z. Li, C. Guo, A. Zhou, Z. Sheng, J. Hu, C. S. Jensen, and B. Yang, “TAB: unified benchmarking of time series anomaly detection methods,” *CoRR*, vol. abs/2506.18046, 2025.
- [29] I. D. Katser and V. O. Kozitsin, “Skoltech anomaly benchmark (skab),” <https://www.kaggle.com/dsv/1693952>, 2020.
- [30] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, “Robust anomaly detection for multivariate time series through stochastic recurrent neural network,” in *KDD*, 2019, pp. 2828–2837.
- [31] G. J. J. Van den Burg and C. K. I. Williams, “An evaluation of change point detection algorithms,” *arXiv preprint arXiv:2003.06222*, 2020.
- [32] M. Q. Ma, Y. Zhao, X. Zhang, and L. Akoglu, “The need for unsupervised outlier model selection: A review and evaluation of internal evaluation strategies,” *SIGKDD Explor.*, vol. 25, no. 1, pp. 19–35, 2023.
- [33] C. C. Aggarwal and S. Sathe, “Theoretical foundations and algorithms for outlier ensembles,” *SIGKDD Explor.*, vol. 17, no. 1, pp. 24–47, 2015.
- [34] H. Alibrahim and S. A. Ludwig, “Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization,” in *CEC*. IEEE, 2021, pp. 1551–1559.
- [35] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, “Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding,” in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 387–395.
- [36] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, “USAD: unsupervised anomaly detection on multivariate time series,” in *KDD*, 2020, pp. 3395–3404.
- [37] S. Tuli, G. Casale, and N. R. Jennings, “Tranad: Deep transformer networks for anomaly detection in multivariate time series data,” *PVLDB*, vol. 15, no. 6, pp. 1201–1214, 2022.
- [38] J. Xu, H. Wu, J. Wang, and M. Long, “Anomaly transformer: Time series anomaly detection with association discrepancy,” in *ICLR*, 2022.
- [39] K. Ding, S. Ding, A. Morozov, T. Fabarisov, and K. Janschek, “Online error detection and mitigation for time-series data of cyber-physical systems using deep learning based methods,” in *2019 15th European Dependable Computing Conference (EDCC)*. IEEE, 2019, pp. 7–14.
- [40] S. Ding, A. Morozov, T. Fabarisov, and S. Vock, “Krakenbox: Deep learning-based error detector for industrial cyber-physical systems,” 11 2021.
- [41] S. Ding, A. Morozov, S. Vock, M. Weyrich, and K. Janschek, “Model-based error detection for industrial automation systems using lstm networks,” *Accepted to 7th International Symposium on Model-Based Safety and Assessment*, 2020.

- [42] S. Ding, S. Ayoub, and A. Morozov, *Tool Paper: Time Series Anomaly Detection Platform for MATLAB Simulink*, 09 2022, pp. 204–218.
- [43] *Adaptive and Scalable Approach for Time Series Anomaly Detection Using Transformer*, ser. ASME International Mechanical Engineering Congress and Exposition, vol. Volume 11: Safety Engineering, Risk and Reliability Analysis; Research Posters, 2024.
- [44] S. Ding, N. Chakraborty, and A. Morozov, “Imu sensor faults detection for uav using machine learning,” 09 2022.
- [45] Y. Ying, J. Duan, C. Wang, Y. Wang, C. Huang, and B. Xu, “Automated model selection for time-series anomaly detection,” *arXiv preprint arXiv:2009.04395*, 2020.
- [46] S. Ding, A. Wolf, and A. Morozov, “Automated and self-adapting approach to ai-based anomaly detection,” 01 2023, pp. 3056–3063.
- [47] J. E. Zhang, D. Wu, and B. Boulet, “Time series anomaly detection via reinforcement learning-based model selection,” in *CCECE*, 2022, pp. 193–199.
- [48] M. V. Ngo, T. Luo, and T. Q. Quek, “Adaptive anomaly detection for internet of things in hierarchical edge computing: A contextual-bandit approach,” *ACM Transactions on Internet of Things*, vol. 3, no. 1, pp. 1–23, 2021.
- [49] E. Sylligardos, P. Boniol, J. Paparrizos, P. Trahanias, and T. Palpanas, “Choose wisely: An extensive evaluation of model selection for anomaly detection in time series,” *PVLDB*, vol. 16, no. 11, pp. 3418–3432, 2023.
- [50] P. Boniol, E. Sylligardos, J. Paparrizos, P. E. Trahanias, and T. Palpanas, “Adecimo: Model selection for time series anomaly detection,” in *ICDE*, 2024, pp. 5441–5444.
- [51] S. Rayana and L. Akoglu, “Less is more: Building selective anomaly ensembles,” *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 4, pp. 42:1–42:33, 2016.
- [52] J. M. Kleinberg, “Authoritative sources in a hyperlinked environment,” *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.
- [53] J. Paparrizos, Y. Kang, P. Boniol, R. S. Tsay, T. Palpanas, and M. J. Franklin, “TSB-UAD: an end-to-end benchmark suite for univariate time-series anomaly detection,” *PVLDB*, vol. 15, no. 8, pp. 1697–1711, 2022.
- [54] S. Schmidl, P. Wenig, and T. Papenbrock, “Anomaly detection in time series: A comprehensive evaluation,” *PVLDB*, vol. 15, no. 9, pp. 1779–1797, 2022.
- [55] D. H. Wolpert, “Stacked generalization,” *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.
- [56] J. Chen, G. Liu, Y. Cao, R. Chen, G. Xiao, and X. Zhang, “Ist-sbilstm: Multi-level stacking ensemble approach using gravitational search algorithm and bilstm for accurate state of health prediction in lithium-ion batteries,” *Expert Syst. Appl.*, vol. 296, p. 129009, 2026.
- [57] N. Damanik and C. Liu, “Advanced fraud detection: Leveraging K-SMOTEENN and stacking ensemble to tackle data imbalance and extract insights,” *IEEE Access*, vol. 13, pp. 10 356–10 370, 2025.
- [58] D. Treder-Tschechlov, M. Fritz, H. Schwarz, and B. Mitschang, “Ensemble clustering based on meta-learning and hyperparameter optimization,” *PVLDB*, vol. 17, no. 11, pp. 2880–2892, 2024.
- [59] R. M. Cruz, R. Sabourin, G. D. Cavalcanti, and T. I. Ren, “Meta-des: A dynamic ensemble selection framework using meta-learning,” *Pattern recognition*, vol. 48, no. 5, pp. 1925–1935, 2015.
- [60] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
- [61] G. Biau, “Analysis of a random forests model,” *J. Mach. Learn. Res.*, vol. 13, pp. 1063–1095, 2012.
- [62] Y. Sun and P. Kosmas, “Meta-forests: Domain generalization on random forests with meta-learning,” in *Asian Conference on Machine Learning*. PMLR, 2024, pp. 1292–1307.
- [63] E. Taşçı, “A meta-ensemble classifier approach: Random rotation forest,” *Balkan Journal of Electrical and Computer Engineering*, vol. 7, no. 2, pp. 182–187, 2019.
- [64] D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression, Second Edition*. Wiley, 2000.
- [65] J. D. Conklin, “Applied logistic regression,” *Technometrics*, vol. 44, no. 1, pp. 81–82, 2002.
- [66] T. Ke, H. Cao, Z. Ling, and F. Zhou, “Revisiting logistic-softmax likelihood in bayesian meta-learning for few-shot classification,” *NeurIPS*, vol. 36, pp. 34 590–34 602, 2023.
- [67] I. Khan, X. Zhang, M. Rehman, and R. Ali, “A literature survey and empirical study of meta-learning for classifier selection,” *IEEE Access*, vol. 8, pp. 10 262–10 281, 2020.
- [68] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [69] B. L. Milenova, J. S. Yarmus, and M. M. Campos, “Svm in oracle database 10g: removing the barriers to widespread adoption of support vector machines,” in *VLDB*, 2005, pp. 1152–1163.
- [70] Y. Yao, D. Li, H. Jie, L. Chen, T. Li, J. Chen, J. Wang, F. Li, and Y. Gao, “Simplets: An efficient and universal model selection framework for time series forecasting,” *PVLDB*, vol. 16, no. 12, pp. 3741–3753, 2023.
- [71] M. Claesen, F. D. Smet, J. A. K. Suykens, and B. D. Moor, “Ensemblesvm: A library for ensemble learning using support vector machines,” *CoRR*, vol. abs/1403.0745, 2014.
- [72] Y. Liao, M. Li, Q. Sun, and P. Li, “Advanced stacking models for machine fault diagnosis with ensemble trees and SVM,” *Appl. Intell.*, vol. 55, no. 3, p. 251, 2025.
- [73] S. Agrawal and N. Goyal, “Thompson sampling for contextual bandits with linear payoffs,” in *ICML*. PMLR, 2013, pp. 127–135.
- [74] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, “Generative adversarial networks: introduction and outlook,” *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.
- [75] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *NeurIPS*, vol. 27, 2014.
- [76] M. A. M. Sadr, Y. Zhu, and P. Hu, “An anomaly detection method for satellites using monte carlo dropout,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 59, no. 2, pp. 2044–2052, 2023.
- [77] S. Raychaudhuri, “Introduction to monte carlo simulation,” in *WSC*. WSC, 2008, pp. 91–100.
- [78] Y. Ban and J. He, “Generic outlier detection in multi-armed bandit,” in *KDD*. ACM, 2020, pp. 913–923.
- [79] K. Ding, J. Li, and H. Liu, “Interactive anomaly detection on attributed networks,” in *WSDM*. ACM, 2019, pp. 357–365.
- [80] S. Raychaudhuri, “Introduction to monte carlo simulation,” in *2008 Winter simulation conference*. IEEE, 2008, pp. 91–100.
- [81] X. Du, W. Pan, B. Jiang, L. Ding, Y. Pan, C. Yuan, and Y. Xiang, “CRA: identifying key classes using markov-chain-based ranking aggregation,” *Axioms*, vol. 11, no. 10, p. 491, 2022.
- [82] M. Sakurada and T. Yairi, “Anomaly detection using autoencoders with nonlinear dimensionality reduction,” in *MLSDA@PRICAI*, 2014, p. 4.
- [83] S. Chang, Y. Zhang, W. Han, M. Yu, X. Guo, W. Tan, X. Cui, M. Witbrock, M. A. Hasegawa-Johnson, and T. S. Huang, “Dilated recurrent neural networks,” in *NIPS*, 2017, pp. 77–87.
- [84] C. I. Challa, P. Jiang, Y. N. Wu, and L. Callot, “Deep generative model with hierarchical latent factors for time series anomaly detection,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 1643–1654.
- [85] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, “Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications,” in *WWW*, 2018, pp. 187–196.
- [86] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, “Robust anomaly detection for multivariate time series through stochastic recurrent neural network,” in *KDD*, 2019, pp. 2828–2837.
- [87] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, “Timesnet: Temporal 2d-variation modeling for general time series analysis,” in *ICLR*, 2023.
- [88] Z. Xu, A. Zeng, and Q. Xu, “FITS: modeling time series with 10k parameters,” in *ICLR*, 2024.
- [89] H. Kriegel, M. Schubert, and A. Zimek, “Angle-based outlier detection in high-dimensional data,” in *KDD*, 2008, pp. 444–452.
- [90] M. Sakurada and T. Yairi, “Anomaly detection using autoencoders with nonlinear dimensionality reduction,” in *MLSDA@PRICAI*, 2014, p. 4.
- [91] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, “Support vector method for novelty detection,” in *NIPS*, 1999, pp. 582–588.
- [92] T. Yairi, Y. Kato, and K. Hori, “Fault detection by mining association rules from house-keeping data,” in *Proc. of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, vol. 3, no. 9, 2001.
- [93] Z. He, X. Xu, and S. Deng, “Discovering cluster-based local outliers,” *Pattern Recognit. Lett.*, vol. 24, no. 9–10, pp. 1641–1650, 2003.
- [94] Z. Li, H. Ma, and Y. Mei, “A unifying method for outlier and change detection from data streams based on local polynomial fitting,” in

- PAKDD*, ser. Lecture Notes in Computer Science, vol. 4426, 2007, pp. 150–161.
- [95] F. T. Liu, K. M. Ting, and Z. Zhou, “Isolation forest,” in *ICDM*, 2008, pp. 413–422.
- [96] M. Goldstein and A. Dengel, “Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm,” *KI-2012: poster and demo track*, vol. 1, pp. 59–63, 2012.
- [97] C. C. Aggarwal, “An introduction to outlier analysis,” in *Outlier analysis*, 2016, pp. 1–34.
- [98] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, “One fits all: Power general time series analysis by pretrained LM,” in *NeurIPS*, 2023.
- [99] K. Rasul, A. Ashok, A. R. Williams, A. Khorasani, G. Adamopoulos, R. Bhagwatkar, M. Bilos, H. Ghonia, N. V. Hassen, A. Schneider, S. Garg, A. Drouin, N. Chapados, Y. Nevyvaka, and I. Rish, “Lag-llama: Towards foundation models for time series forecasting,” *CoRR*, vol. abs/2310.08278, 2023.
- [100] A. F. Ansari, L. Stella, A. C. Türkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. Pineda-Arango, S. Kapoor, J. Zschiegner, D. C. Maddix, H. Wang, M. W. Mahoney, K. Torkkola, A. G. Wilson, M. Bohlke-Schneider, and B. Wang, “Chronos: Learning the language of time series,” *Trans. Mach. Learn. Res.*, vol. 2024, 2024.
- [101] A. Das, W. Kong, R. Sen, and Y. Zhou, “A decoder-only foundation model for time-series forecasting,” in *ICML*, 2024.
- [102] M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski, “MOMENT: A family of open time-series foundation models,” in *ICML*, 2024.
- [103] Y. Sui, X. Wang, T. Cui, T. Xiao, C. He, S. Zhang, Y. Zhang, X. Yang, Y. Sun, and D. Pei, “Bridging the gap: Llm-powered transfer learning for log anomaly detection in new software systems,” in *ICDE*. IEEE, 2025, pp. 4414–4427.
- [104] N. Takeishi, “Shapley values of reconstruction errors of PCA for explaining anomaly detection,” in *ICDM Workshops*. IEEE, 2019, pp. 793–798.
- [105] S. Shetiya, I. P. Swift, A. Asudeh, and G. Das, “Shapley values for explanation in two-sided matching applications,” in *EDBT*. OpenProceedings.org, 2024, pp. 584–596.